Workgroup: GNAP
Internet-Draft:
draft-ietf-gnap-resource-servers-03
Published: 13 March 2023
Intended Status: Standards Track
Expires: 14 September 2023
Authors: J. Richer, Ed.
Bespoke Engineering
F. Imbault
acert.io

Grant Negotiation and Authorization Protocol Resource Server Connections

Abstract

GNAP defines a mechanism for delegating authorization to a piece of software, and conveying that delegation to the software. This extension defines methods for resource servers (RS) to communicate with authorization servers (AS) in an interoperable fashion.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 14 September 2023.

Table of Contents

1. Introduction

The core GNAP protocol does not define any one specific mechanism for the resource server (RS) to communicate with the authorization server (AS), allowing the connection between these components to be solved orthogonally to the core protocol's concerns. For example, the RS and AS roles could be fulfilled by the same piece of software with common storage, obviating the need for any connecting protocol. However, it is often desirable to have the RS and AS communicate at runtime for a variety of purposes, including allowing the RS to validate and understand the rights and privileges associated with a grant of access represented by an access token issued by the AS, or negotiating the capabilities of either party. These types of connections are particularly useful for connecting an AS and RS from different vendors, allowing interoperable distributed deployments of GNAP-protected systems.

This specification defines several means for an RS and AS to communicate these aspects with each other, including structured access tokens and RS-facing APIs. This specification also discusses methods for an RS to derive a downstream token for calling another chained RS.

The means of the authorization server issuing the access token to the client instance and the means of the client instance presenting the access token to the resource server are the subject of the GNAP core protocol specification [GNAP].

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.

This document contains non-normative examples of partial and complete HTTP messages, JSON structures, URLs, query components, keys, and other elements. Some examples use a single trailing backslash \ to indicate line wrapping for long values, as per [RFC8792]. The \ character and leading spaces on wrapped lines are not part of the value.

Terminology specific to GNAP is defined in the terminology section of the core specification [GNAP], and provides definitions for the protocol roles: Authorization Server (AS), Client, Resource Server (RS), Resource Owner (RO), End-user; as well as the protocol elements: Attribute, Access Token, Grant, Privilege, Protected

Resource, Right, Subject, Subject Information. The same definitions are used in this document.

2. Access Tokens

Access tokens are a mechanism for an AS to provide a client instance limited access to an RS. These access tokens are artifacts representing a particular set of access rights granted to the client instance to act on behalf of the RO. While the format of access tokens varies in different systems (see discussion in Section 2.2), the concept of an access token is consistent across all GNAP systems.

2.1. General-purpose Access Token Model

Access tokens represent a common set of aspects across different GNAP deployments. This is not intended to be a universal or comprehensive list, but instead to provide guidance to implementors when developing data structures and associated systems across a GNAP deployment. These data structures are communicated between the AS and RS either by using a structured token or an API-like mechanism like token introspection. This general-purpose data model does not assume either approach, and in fact both can be used together to convey different pieces of information. Where possible, mappings to concrete data fields in common standards understood by the RS are provided for each item in the model.

2.1.1. Value

All access tokens have a unique value. This is the string that is passed on the wire between parties. The AS chooses the value, which can be structured as in Section 2.2 or unstructured. When the token is structured, the token value also has a *format* known to the AS and RS, and the other items in this token model are contained within the token's value in some fashion. When the token is unstructured, the values are usually retrieved by the RS using a service like token introspection described in Section 3.3.

The access token value is conveyed the value field of an access_token response from Section 3.2 of [GNAP].

The format and content of the access token value is opaque to the client software. While the client software needs to be able to carry and present the access token value, the client software is never expected nor intended to be able to understand the token value itself.

2.1.2. Issuer

The access token is issued by the AS in a standard GNAP transaction. The AS will often need to identify itself in order to recognize tokens that it has issued, particularly in cases where tokens from multiple different AS's could be presented.

This information is not usually conveyed directly to the client instance, since the client instance should know this information based on where it receives the token from.

In a [JWT] formatted token or a token introspection response, this corresponds to the iss claim.

2.1.3. Audience

The access token is intended for use at one or more RS's. The AS can identify those RS's to allow each RS to ensure that the token is not receiving a token intended for someone else. The AS and RS have to agree on the nature of any audience identifiers represented by the token, but the URIs of the RS are a common pattern.

In a [JWT] formatted token or token introspection response, this corresponds to the aud claim.

In cases where more complex access is required, the location field of objects in the access array can also convey audience information. In such cases, the client instance might need to know the audience information in order to differentiate between possible RS's to present the token to.

2.1.4. Key Binding

Access tokens in GNAP are bound to the client instance's registered or presented key, except in cases where the access token is a bearer token. For all tokens bound to a key, the AS and RS need to be able to identify which key the token is bound to, otherwise an attacker could substitute their own key during presentation of the token. In the case of an asymmetric algorithm, the model for the AS and RS need only contain the public key, while the client instance will also need to know the private key in order to present the token appropriately. In the case of a symmetric algorithm, all parties will need to either know or be able to derive the shared key.

The source of this key information can vary depending on circumstance and deployment. For example, an AS could decide that all tokens issued to a client instance are always bound to that client instance's current key. When the key needs to be dereferenced, the AS looks up the client instance to which the token was issued and finds the key information there. The AS could

alternatively bind each token to a specific key that is managed separately from client instance information. In such a case, the AS determines the key information directly. This approach allows the client instance to use a different key for each request, or allows the AS to issue a key for the client instance to use with the particular token.

In all cases, the key binding also includes a proofing mechanism, along with any parameters needed for that mechanism such as a signing or digest algorithm. If such information is not stored, an attacker could present a token with a seemingly-valid key using an insecure and incorrect proofing mechanism.

This value is conveyed to the client instance in the key field of the access_token response in Section 3.2 of [GNAP]. Since the common case is that the token is bound to the client instance's registered key, this field can be omitted in this case since the client will be aware of its own key.

In a [JWT] formatted token, this corresponds to the cnf (confirmation) claim. In a token introspection response, this corresponds to the key claim.

In the case of a bearer token, all parties need to know that a token has no key bound to it, and will therefore reject any attempts to use the bearer token with a key in an undefined way.

2.1.5. Flags

GNAP access tokens can have multiple data flags associated with them that indicate special processing or considerations for the token. For example, whether the token is a bearer token, or should be expected to be durable across grant updates.

The client can request a set of flags in the access_token request in [GNAP].

These flags are conveyed from the AS to the client in the flags field of the access_token response in Section 3.2 of [GNAP].

For token introspection, flags are returned in the flags field of the response.

2.1.6. Access Rights

Access tokens are tied to a limited set of access rights. These rights specify in some detail what the token can be used for, how, and where. The internal structure of access rights are detailed in Section 8 of [GNAP].

The access rights associated with an access token are calculated from the rights available to the client instance making the request, the rights available to be approved by the RO, the rights actually approved by the RO, and the rights corresponding to the RS in question. The rights for a specific access token are a subset of the overall rights in a grant request.

These rights are requested by the client instance in the access field of the access_token request in Section 2.1 of [GNAP].

The rights associated with an issued access token are conveyed to the client instance in the access field of the access_token response in Section 3.2 of [GNAP].

In token introspection responses, this corresponds to the access claim.

2.1.7. Time Validity Window

The access token can be limited to a certain time window outside of which it is no longer valid for use at an RS. This window can be explicitly bounded by an expiration time and a not-before time, or it could be calculated based on the issuance time of the token. For example, an RS could decide that it will accept tokens for most calls within an hour of a token's issuance, but only within five minutes of the token's issuance for certain high-value calls.

Since access tokens could be revoked at any time for any reason outside of a client instance's control, the client instance often does not know or concern itself with the validity time window of an access token. However, this information can be made available to it using the expires_in field of an access token response in Section 3.2 of [GNAP].

The issuance time of the token is conveyed in the iat claim of a [JWT] formatted token or a token introspection response.

The expiration time of a token, after which it is to be rejected, is conveyed in the exp claim of a [JWT] formatted token or a token introspection response.

The starting time of a token's validity window, before which it is to be rejected, is conveyed in the nbf claim of a [JWT] formatted token or a token introspection response.

2.1.8. Token Identifier

Individual access tokens often need a unique internal identifier to allow the AS to differentiate between multiple separate tokens. This

value of the token can often be used as the identifier, but in some cases a separate identifier is used.

This separate identifier can be conveyed in the jti claim of a [JWT] formatted token or a token introspection response.

This identifier is not usually exposed to the client instance using the token, since the client instance only needs to use the token by value.

2.1.9. Authorizing Resource Owner

Access tokens are approved on behalf of a resource owner (RO). The identity of this RO can be used by the RS to determine exactly which resource to access, or which kinds of access to allow. For example, an access token used to access identity information can hold a user identifier to allow the RS to determine which profile information to return. The nature of this information is subject to agreement by the AS and RS.

This corresponds to the sub claim of a [JWT] formatted token or a token introspection response.

The RO information is not generally returned to the client instance, and in many cases returning this information to the client instance would be a privacy violation on the part of the AS. Since the access token represents a specific delegated access, the client instance needs only to use the token at its target RS. Following the profile example, the client instance does not need to know the account identifier to get specific attributes about the account represented by the token.

2.1.10. Client Instance

Access tokens are issued to a specific client instance by the AS. The identity of this instance can be used by the RS to allow specific kinds of access, or other attributes about the access token. For example, an AS that binds all access tokens issued to a particular client instance to that client instance's most recent key rotation would need to be able to look up the client instance in order to find the key binding detail.

This corresponds to the client_id claim of a [JWT] formatted token or the instance_id field of a token introspection response.

The client is not normally informed of this information separately, since a client instance can usually correctly assume that it is the client instance to which a token that it receives was issued.

2.1.11. Label

When multiple access tokens are requested or a client instance uses token labels, the parties will need to keep track of which labels were applied to each individual token. Since labels can be re-used across different grant requests, the token label alone is not sufficient to uniquely identify a given access token in a system. However, within the context of a grant request, these labels are required to be unique.

A client instance can request a specific label using the label field of an access_token request in Section 2.1 of [GNAP].

The AS can inform the client instance of a token's label using the label field of an access_token response in Section 3.2 of [GNAP].

This corresponds to the label field of a token introspection response.

2.1.12. Parent Grant Request

All access tokens are issued in the context of a specific grant request from a client instance. The grant request itself represents a unique tuple of:

- The AS processing the grant request
- The client instance making the grant request
- The RO (or set of RO's) approving the grant request (or needing to approve it)
- The access rights granted by the RO
- The current state of the grant request, as defined in Section 1.5 of [GNAP]

The AS can use this information to tie common information to a specific token. For instance, instead of specifying a client instance for every issued access token for a grant, the AS can store the client information in the grant itself and look it up by reference from the access token.

The AS can also use this information when a grant request is updated. For example, if the client instance asks for a new access token from an existing grant, the AS can use this link to revoke older non-durable access tokens that had been previously issued under the grant.

A client instance will have its own model of an ongoing grant request, especially if that grant request can be continued using the API defined in Section 5 of [GNAP] where several pieces of statefulness need to be kept in hand. The client instance might need to keep an association with the grant request that issued the token in case the access token expires or does not have sufficient access rights, so that the client instance can get a new access token without having to restart the grant request process from scratch.

Since the grant itself does not need to be identified in any of the protocol messages, GNAP does not define a specific grant identifier to be conveyed between any parties in the protocol. Only the AS needs to keep an explicit connection between an issued access token and the parent grant that issued it.

2.1.13. Continuation Access Token

When an access token is used for the grant continuation API defined in Section 5 of [GNAP], the AS needs to be able to separate this access token from others usable at RS's. The AS can do this through the use of a flag on the access token data structure, by using a special internal access right, or any other means at its disposal.

A client instance will need to store continuation access tokens separately from access tokens used at the RS in order to keep them from being confused with each other and used at the wrong endpoints.

The client instance is given continuation access tokens only as part of the continue field of the grant response in Section 3.1 of [GNAP].

An RS should never see a continuation access token, so any attempts to process one should fail.

2.2. Access Token Formats

When the AS issues an access token for use at an RS, the RS needs to have some means of understanding what the access token is for in order to determine how to respond to the request. The core GNAP protocol makes neither assumptions nor demands on the format or contents of the access token, and in fact, the token format and contents are opaque to the client instance. However, such token formats can be the topic of agreements between the AS and RS.

Self-contained structured token formats allow for the conveyance of information between the AS and RS without requiring the RS to call the AS at runtime as described in Section 3.3. Structured tokens can also be used in combination with introspection, allowing the token itself to carry one class of information and the introspection response to carry another.

Some token formats, such as Macaroons and Biscuits, allow for the RS to derive sub-tokens without having to call the AS as described in Section 4.

The supported token formats can be communicated dynamically at runtime between the AS and RS in several places.

- The AS can declare its supported token formats as part of RS-facing discovery Section 3.1
- The RS can require a specific token format be used to access a registered resource set Section 3.4
- The AS can return the token's format in an introspection response Section 3.3

In all places where the token format is listed explicitly, it **MUST** be one of the registered values in the GNAP Token Formats Registry Section 6.1.

3. Resource-Server-Facing API

To facilitate runtime and dynamic connections, the AS can offer an RS-Facing API consisting of one or more of the following optional pieces.

- Discovery
- Introspection
- Token chaining
- Resource reference registration

3.1. RS-facing AS Discovery

A GNAP AS offering RS-facing services can publish its features on a well-known discovery document using the URL .well-known/gnap-as-rs appended to the grant request endpoint URL.

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

introspection_endpoint (string): **OPTIONAL**. The URL of the endpoint offering introspection. The location **MUST** be a URL [RFC3986] with a scheme component that **MUST** be https, a host component, and

optionally, port, path and query components and no fragment components. Section 3.3

token_formats_supported (array of strings): A list of token formats supported by this AS. The values in this list **MUST** be registered in the GNAP Token Format Registry. Section 6.1

resource_registration_endpoint (string): The URL of the endpoint offering resource registration. The location **MUST** be a URL [RFC3986] with a scheme component that **MUST** be https, a host component, and optionally, port, path and query components and no fragment components. Section 3.4

grant_request_endpoint (string): **REQUIRED**. The location of the AS's grant request endpoint, used by the RS to derive downstream access tokens. The location **MUST** be a URL [RFC3986] with a scheme component that **MUST** be https, a host component, and optionally, port, path and query components and no fragment components. This URL **MUST** be the same URL used by client instances in support of GNAP requests. Section 4

key_proofs_supported (array of strings) **OPTIONAL**. A list of the AS's supported key proofing mechanisms. The values of this list correspond to possible values of the proof field of the key section of the request.

3.2. Protecting RS requests to the AS

Unless otherwise specified, the RS **MUST** protect its calls to the AS using any of the signature methods defined by GNAP. This signing method **MUST** cover all of the appropriate portions of the HTTP request message, including any body elements, tokens, or headers required for functionality.

The RS **MAY** present its keys by reference or by value in a similar fashion to a client instance calling the AS in the core protocol of GNAP, described in [GNAP]. In the protocols defined here, this takes the form of the resource server identifying itself using a key field or by passing an instance identifier directly.

```
"resource_server": {
    "key": {
        "proof": "httpsig",
        "jwk": {
            "kty": "EC",
            "crv": "secp256k1",
            "kid": "2021-07-06T20:22:03Z",
            "x": "-J9OJIZj4nmopZbQN7T8xv3sbeS5-f_vBNSy_EHnBZc",
            "y": "sjrS51pLtu3P4LUTVvyAIxRfDV_be2RYpI5_f-Yjivw"
        }
    }
}
```

or by reference:

```
"resource_server": "7C7C4AZ9KHRS6X63AJAO"
```

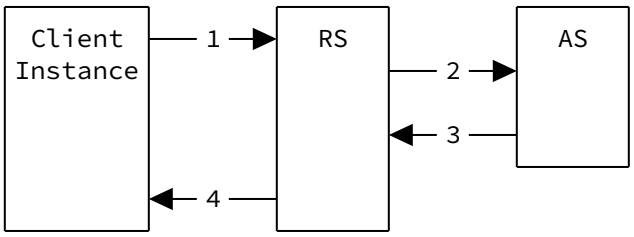
The AS **MAY** require an RS to pre-register its keys or could allow calls from arbitrary keys in a trust-on-first-use model.

3.3. Token Introspection

The AS issues access tokens representing a set of delegated access rights to be used at one or more RSs. The AS can offer an introspection service to allow an RS to validate that a given access token:

- *has been issued by the AS
- *has not expired
- *has not been revoked
- *is appropriate for the RS identified in the call

When the RS receives an access token, it can call the introspection endpoint at the AS to get token information.



1. The client instance calls the RS with its access token.

2. The RS introspects the access token value at the AS. The RS signs the request with its own key (not the client instance's key or the token's key).

3. The AS validates the access token value and the Resource Server's request and returns the introspection response for the token.

4. The RS fulfills the request from the client instance.

The RS signs the request with its own key and sends the value of the access token as the body of the request as a JSON object with the following members:

access_token (string): **REQUIRED**. The access token value presented to the RS by the client instance.

proof (string): **RECOMMENDED**. The proofing method used by the client instance to bind the token to the RS request.

resource_server (string or object): **REQUIRED**. The identification used to authenticate the resource server making this call, either by value or by reference as described in Section 3.2.

access (array of strings/objects): **OPTIONAL**. The minimum access rights required to fulfill the request. This **MUST** be in the format described in the Resource Access Rights section of [GNAP].

```
POST /introspect HTTP/1.1
Host: server.example.com
Content-Type: application/json
Signature-Input: sig1=...
Signature: sig1=...
Digest: sha256=...

{
    "access_token": "OS9M2PMHKUR64TB8N6BW7OZB8CDFONP219RP1LT0",
    "proof": "httpsig",
    "resource_server": "7C7C4AZ9KHRS6X63AJAO"
}
```

The AS **MUST** validate the access token value and determine if the token is active. An active access token is defined as a token that

*was issued by the processing AS,

*has not been revoked,

*has not expired, and

*is appropriate for presentation at the identified RS.

The AS responds with a data structure describing the token's current state and any information the RS would need to validate the token's presentation, such as its intended proofing mechanism and key material.

active (boolean): **REQUIRED**. If true, the access token presented is active, as defined above. If any of the criteria for an active token are not true, or if the AS is unable to make a determination (such as the token is not found), the value is set to false and other fields are omitted.

If the access token is active, additional fields from the single access token response structure defined in [GNAP] are included. In particular, these include the following:

access (array of strings/objects): **REQUIRED**. The access rights associated with this access token. This **MUST** be in the format described in the Resource Access Rights section of [GNAP]. This array **MAY** be filtered or otherwise limited for consumption by the identified RS, including being an empty array.

key (object/string): **REQUIRED** if the token is bound. The key bound to the access token, to allow the RS to validate the signature of

the request from the client instance. If the access token is a bearer token, this **MUST NOT** be included.

flags (array of strings): **OPTIONAL**. The set of flags associated with the access token.

exp (integer): **OPTIONAL**. The timestamp after which this token is no longer valid. Expressed as a integer seconds from UNIX Epoch.

iat (integer): **OPTIONAL**. The timestamp at which this token was issued by the AS. Expressed as a integer seconds from UNIX Epoch.

nbf (integer): **OPTIONAL**. The timestamp before which this token is not valid. Expressed as a integer seconds from UNIX Epoch.

aud (string or array of strings): **OPTIONAL**. Identifiers for the resource servers this token can be accepted at.

sub (string): **OPTIONAL**. Identifier of the resource owner who authorized this token.

iss (string): **REQUIRED**. Grant endpoint URL of the AS that issued this token.

instance_id (string): **OPTIONAL**. The instance identifier of the client instance that the token was issued to.

The response **MAY** include any additional fields defined in an access token response and **MUST NOT** include the access token value itself.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
    "active": true,
    "access": [
        "dolphin-metadata", "some other thing"
    ],
    "key": {
        "proof": "httpsig",
        "jwk": {
                "kty": "RSA",
                "e": "AQAB",
                "kid": "xyz-1",
                "alg": "RS256",
                "n": "kOB5rR4Jv0GMeL...."
        }
    }
}
```

3.4. Registering a Resource Set

If the RS needs to, it can post a set of resources as described in the Resource Access Rights section of [GNAP] to the AS's resource registration endpoint along with information about what the RS will need to validate the request.

access (array of objects/strings): **REQUIRED**. The list of access rights associated with the request in the format described in the "Resource Access Rights" section of [GNAP].

resource_server (string or object): **REQUIRED**. The identification used to authenticate the resource server making this call, either by value or by reference as described in Section 3.2.

token_format_required (string): **OPTIONAL**. The token format required to access the identified resource. If the field is omitted, the token format is at the discretion of the AS. If the AS does not support the requested token format, the AS **MUST** return an error to the RS.

token_introspection_required (boolean): **OPTIONAL**. If present and set to true, the RS expects to make a token introspection request as described in Section 3.3. If absent or set to false, the RS does not anticipate needing to make an introspection request for tokens relating to this resource set.

The RS **MUST** identify itself with its own key and sign the request.

```
POST /resource HTTP/1.1
Host: server.example.com
Content-Type: application/json
Signature-Input: sig1=...
Signature: sig1=...
Digest: ...

{
    "access": [
        {
            "actions": [
                "read",
                "write",
                "dolphin"
            ],
            "locations": [
                "https://server.example.net/",
                "https://resource.local/other"
            ],
            "datatypes": [
                "metadata",
                "images"
            ]
        },
        "dolphin-metadata"
    ],
    "resource_server": "7C7C4AZ9KHRS6X63AJAO"

}
```

The AS responds with a reference appropriate to represent the resources list that the RS presented in its request as well as any additional information the RS might need in future requests.

resource_reference (string): **REQUIRED**. A single string representing the list of resources registered in the request. The RS **MAY** make this handle available to a client instance as part of a discovery response as described in [GNAP] or as documentation to client software developers.

instance_id (string): **OPTIONAL**. An instance identifier that the RS can use to refer to itself in future calls to the AS, in lieu of sending its key by value.

introspection_endpoint (string): **OPTIONAL**. The introspection endpoint of this AS, used to allow the RS to perform token introspection. Section 3.3

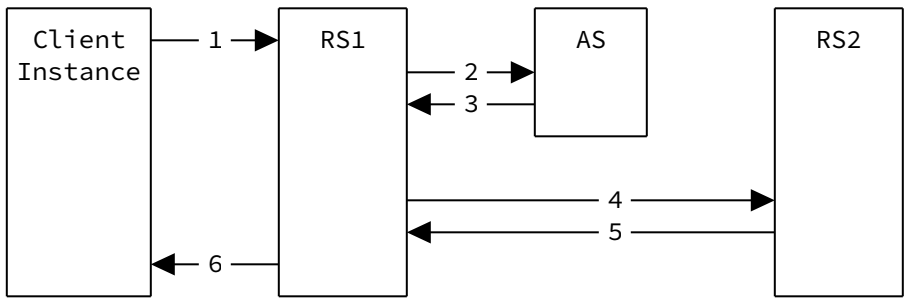
```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
    "resource_reference": "FWWIKYBQ6U56NL1"
}
```

4. Deriving a downstream token

Some architectures require an RS to act as a client instance and use a derived access token for a secondary RS. Since the RS is not the same entity that made the initial grant request, the RS is not capable of referencing or modifying the existing grant. As such, the RS needs to request or generate a new token access token for its use at the secondary RS. This internal secondary token is issued in the context of the incoming access token.

While it is possible to use a token format (Section 2) that allows for the RS to generate its own secondary token, the AS can allow the RS to request this secondary access token using the same process used by the original client instance to request the primary access token. Since the RS is acting as its own client instance from the perspective of GNAP, this process uses the same grant endpoint, request structure, and response structure as a client instance's request.



1. The client instance calls RS1 with an access token.

2. RS1 presents that token to the AS to get a derived token for use at RS2. RS1 indicates that it has no ability to interact with the RO. Note that RS1 signs its request with its own key, not the token's key or the client instance's key.

3. The AS returns a derived token to RS1 for use at RS2.

4. RS1 calls RS2 with the token from (3).

5. RS2 fulfills the call from RS1.

6. RS1 fulfills the call from the original client instance.

If the RS needs to derive a token from one presented to it, it can request one from the AS by making a token request as described in [GNAP] and presenting the existing access token's value in the "existing_access_token" field.

Since the RS is acting as a client instance, the RS **MUST** identify itself with its own key in the client field and sign the request just as any client instance would, as described in Section 3.2.

```
POST /tx HTTP/1.1
Host: server.example.com
Content-Type: application/json
Detached-JWS: ejy0...

{
    "access_token": {
        "access": [
            {
                "actions": [
                    "read",
                    "write",
                    "dolphin"
                ],
                "locations": [
                    "https://server.example.net/",
                    "https://resource.local/other"
                ],
                "datatypes": [
                    "metadata",
                    "images"
                ]
            },
            "dolphin-metadata"
        ]
    },
    "client": "7C7C4AZ9KHRS6X63AJAO",
    "existing_access_token": "OS9M2PMHKUR64TB8N6BW7OZB8CDFONP219RP1LT0"
}
```

The AS responds with a token for the downstream RS2 as described in [GNAP]. The downstream RS2 could repeat this process as necessary for calling further RS's.

5. Acknowledgements

(TODO: the ACK section should probably be split between the documents)

6. IANA Considerations

IANA is requested to create the following registries.

6.1. Token Formats Registry

This document defines a GNAP token format, for which IANA is asked to create and maintain a new registry titled "GNAP Token Formats". Initial values for this registry are given in Section 6.1.2. Future assignments and modifications to existing assignment are to be made through the Expert Review registration policy [RFC8126].

The Designated Expert (DE) is expected to ensure that all registrations follow the template presented in Section 6.1.1. The DE is expected to ensure that the format's definition is sufficiently unique from other formats provided by existing parameters. The DE is expected to ensure that the format's definition specifies the format of the access token in sufficient detail to allow for the AS and RS to be able to communicate the token information.

6.1.1. Registry Template

Name The name of the format.

Status Whether or not the format is in active use. Possible values are Active and Deprecated.

Description Human-readable description of the access token format.

Reference The specification that defines the token.

6.1.2. Initial Registry Contents

Name	Status	Description	Reference
jwt-signed	Active	JSON Web Token, signed with JWS	[JWT]
jwt-encrypted	Active	JSON Web Token, encrypted with JWE	[JWT]
macaroon	Active	Macaroon	
biscuit	Active	Biscuit	
zcap	Active	ZCAP	

Table 1

6.2. Token Introspection Registry

This document defines GNAP token introspection, for which IANA is asked to create and maintain a new registry titled "GNAP Token

Introspection". Initial values for this registry are given in Section 6.2.2. Future assignments and modifications to existing assignment are to be made through the Expert Review registration policy [RFC8126].

The Designated Expert (DE) is expected to ensure that all registrations follow the template presented in Section 6.2.1. The DE is expected to ensure that the claim's definition is sufficiently orthogonal to other claims defined in the registry so as avoid overlapping functionality. The DE is expected to ensure that the claim's definition specifies the syntax and semantics of the claim in sufficient detail to allow for the AS and RS to be able to communicate the token values.

6.2.1. Registry Template

Name The name of the claim.

Type The JSON data type of the claim value.

Reference The specification that defines the token.

6.2.2. Initial Registry Contents

The table below contains the initial contents of the GNAP Token Introspection Registry.

Name	Type	Reference
active	boolean	Section 3.3 of This document
access	array of strings/objects	Section 3.3 of This document
key	object/string	Section 3.3 of This document
flags	array of strings	Section 3.3 of This document
exp	integer	Section 3.3 of This document
iat	integer	Section 3.3 of This document
nbf	integer	Section 3.3 of This document
aud	string or array of strings	Section 3.3 of This document
sub	string	Section 3.3 of This document
iss	string	Section 3.3 of This document
instance_id	string	Section 3.3 of This document

Table 2

6.3. Resource Set Registration Request Parameters

This document defines a means to register a resource set for a GNAP AS, for which IANA is asked to create and maintain a new registry titled "GNAP Resource Set Registration Request Parameters". Initial values for this registry are given in Section 6.3.2. Future assignments and modifications to existing assignment are to be made through the Expert Review registration policy [RFC8126].

The Designated Expert (DE) is expected to ensure that all registrations follow the template presented in Section 6.3.1. The DE is expected to ensure that the parameter's definition is sufficiently orthogonal to other claims defined in the registry so as avoid overlapping functionality. The DE is expected to ensure that the parameter's definition specifies the syntax and semantics of the claim in sufficient detail to allow for the AS and RS to be able to communicate the resource set.

6.3.1. Registry Template

Name The name of the parameter.

Type The JSON data type of the parameter value.

Reference The specification that defines the token.

6.3.2. Initial Registry Contents

The table below contains the initial contents of the GNAP Resource Set Registration Request Parameters Registry.

Name	Type	Reference
access	array of strings/objects	Section 3.4 of This document
resource_server	string or object	Section 3.4 of This document
token_format_required	string	Section 3.4 of This document
token_introspection_required	boolean	Section 3.4 of This document

Table 3

6.4. Resource Set Registration Response Parameters

This document defines a means to register a resource set for a GNAP AS, for which IANA is asked to create and maintain a new registry titled "GNAP Resource Set Registration Response Parameters". Initial values for this registry are given in Section 6.4.2. Future assignments and modifications to existing assignment are to be made through the Expert Review registration policy [RFC8126].

The Designated Expert (DE) is expected to ensure that all registrations follow the template presented in Section 6.4.1. The DE is expected to ensure that the parameter's definition is sufficiently orthogonal to other claims defined in the registry so as avoid overlapping functionality. The DE is expected to ensure that the parameter's definition specifies the syntax and semantics

of the claim in sufficient detail to allow for the AS and RS to be able to communicate the resource set.

6.4.1. Registry Template

Name The name of the parameter.

Type The JSON data type of the parameter value.

Reference The specification that defines the token.

6.4.2. Initial Registry Contents

The table below contains the initial contents of the GNAP Resource Set Registration Response Parameters Registry.

Name	Type	Reference
resource_reference	string	Section 3.4 of This document
instance_id	string	Section 3.4 of This document
introspection_endpoint	string	Section 3.4 of This document

Table 4

6.5. RS-Facing Discovery

This document defines a means to for a GNAP AS to be discovered by a GNAP RS, for which IANA is asked to create and maintain a new registry titled "GNAP RS-Facing Discovery". Initial values for this registry are given in Section 6.5.2. Future assignments and modifications to existing assignment are to be made through the Expert Review registration policy [RFC8126].

The Designated Expert (DE) is expected to ensure that all registrations follow the template presented in Section 6.5.1. The DE is expected to ensure that the claim's definition is sufficiently orthogonal to other claims defined in the registry so as avoid overlapping functionality. The DE is expected to ensure that the claim's definition specifies the syntax and semantics of the claim in sufficient detail to allow for RS to be able to communicate with the AS.

6.5.1. Registry Template

Name The name of the parameter.

Type The JSON data type of the parameter value.

Reference The specification that defines the token.

6.5.2. Initial Registry Contents

The table below contains the initial contents of the GNAP RS-Facing Discovery Registry.

Name	Type	Reference
introspection_endpoint	string	Section 3.1 of This document
token_formats_supported	array of strings	Section 3.1 of This document
resource_registration_endpoint	string	Section 3.1 of This document
grant_request_endpoint	string	Section 3.1 of This document
key_proofs_supported	array of strings	Section 3.1 of This document

Table 5

7. Security Considerations

[[TBD: There are a lot of security considerations to add.]]

All requests have to be over TLS or equivalent as per [BCP195]. Many handles act as shared secrets, though they can be combined with a requirement to provide proof of a key as well.

8. Privacy Considerations

[[TBD: There are a lot of privacy considerations to add.]]

When introspection is used, the AS is made aware of a particular token being used at a particular AS, and the AS would not otherwise have insight into this.

When the client instance receives information about the protecting AS from an RS, this can be used to derive information about the resources being protected without releasing the resources themselves.

9. References

9.1. Normative References

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

(DTLS)", May 2015, <https://www.rfc-editor.org/info/bcp195>.

[GNAP] Richer, J. and F. Imbault, "Grant Negotiation and Authorization Protocol", Work in Progress, Internet-Draft, draft-ietf-gnap-core-protocol-13, 27 February 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-gnap-core-protocol-13>.

[JWT] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/rfc/rfc7519>.

[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/rfc/rfc2119>.

[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/rfc/rfc3986>.

[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/rfc/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/rfc/rfc8259>.

[RFC8792] Watsen, K., Auerswald, E., Farrel, A., and Q. Wu, "Handling Long Lines in Content of Internet-Drafts and RFCs", RFC 8792, DOI 10.17487/RFC8792, June 2020, <https://www.rfc-editor.org/rfc/rfc8792>.

9.2. Informative References

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/rfc/rfc8126>.

Appendix A. Document History

*-03

-Added token model.

-Added IANA sections.

*-02

-Editorial and formatting fixes.

*-01

-Better described RS authentication.

-Added access token format registry.

-Filled out introspection protocol.

-Filled out resource registration protocol.

-Expanded RS-facing discovery mechanisms.

-Moved client-facing RS response back to GNAP core document.

*-00

-Extracted resource server section.

Authors' Addresses

Justin Richer (editor)
Bespoke Engineering

Email: ietf@justin.richer.org
URI: https://bspk.io/

Fabien Imbault
acert.io

Email: fabien.imbault@acert.io
URI: https://acert.io/