Workgroup: Network Working Group
Internet-Draft: draft-hardt-gnap-advanced-01
Published: 15 August 2020
Intended Status: Standards Track
Expires: 16 February 2021
Authors: D. Hardt, Ed.
SignIn.Org

The Grant Negotiation and Authorization Protocol - Advanced Features

Abstract

TBD

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 16 February 2021.

Table of Contents

1. Introduction

This document includes additional features for the Grant Negotiation and Authorization Protocol (GNAP) [GNAP], and presumes familiarity and knowledge of GNAP.

Terminology

This document uses the following terms defined in [GNAP]:

*authN

*authZ

*Access

*Access URI

*Claim

*Grant Client (GC)

*Registered Client

*Grant

*Grant Server (GS)

*Grant URI

*Grant Request

*Grant Response

*GS URI

*Interaction

*NumericDate

*Resource Owner (RO)

*Resource Server (RS)

*User

Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this specification are to be interpreted as described in [RFC2119].

Certain security-related terms are to be understood in the sense defined in [RFC4949]. These terms include, but are not limited to, "attack", "authentication", "authorization", "certificate", "confidentiality", "credential", "encryption", "identity", "sign", "signature", "trust", "validate", and "verify".

Unless otherwise noted, all the protocol parameter names and values are case sensitive.

Some protocol parameters are parts of a JSON document, and are referred to in JavaScript notation. For example, foo.bar refers to the "bar" boolean attribute in the "foo" object in the following example JSON document:

```
{
    "foo" : {
        "bar": true
    }
}
```

2. Grant Management APIs

In addition to creating and reading a Grant as specified in GNAP The GC MAY list, update, delete, and discover a Grant.

2.1. List Grants

The GC MAY list the Grants provided to the GC by doing an a GET on the GS URI. The GS MUST respond with a list of Grant URIs [format TBD] or one of the following errors:

- *TBD

from Error Responses Section 8.

2.2. Update Grant

The GC updates a Grant by doing an HTTP PUT of a JSON document to the corresponding Grant URI.

The JSON document MUST include the following from the [GNAP] Grant Request JSON:

- *iat

- *uri set to the Grant URI

and MAY include the following from the [GNAP] Grant Request JSON:

- *user

- *interaction

- *authorization or authorizations

- *claims

The GS MUST respond with one of the standard GNAP responses (Grant Response, Interaction Response, Wait Response) or one of the following errors:

- *TBD

from Error Responses Section 8.

Following is a non-normative example where the GC wants to update the Grant Request with additional claims:

```
{
    "iat"       : 15790460234,
    "uri"       : "https://as.example/endpoint/grant/example3",
    "claims": {
        "oidc": {
            "userinfo" : {
                "email"          : { "essential" : true },
                "name"           : { "essential" : true },
                "picture"        : null
            }
        }
    }
}
```

2.3. Delete Grant

The GC deletes a Grant by doing an HTTP DELETE of the corresponding Grant URI.

The GS MUST respond with OK 200, or one of the following errors:

- *TBD

from Error Responses Section 8.

2.4. Grant Options

The GC can get the supported operations for a Grant by doing an HTTP OPTIONS of the corresponding Grant URI.

The GS MUST respond with the supported methods

[Format TBD]

or one of the following errors:

- *TBD

from Error Responses Section 8.

3. Authorization Management APIs

In addition to reading an Authorization as specified in [GNAP], The GC MAY update, delete, and discover an Authorization.

3.1. Update Access

The GC updates an Authorization by doing an HTTP PUT to the corresponding Access URI of the following JSON. All of the following MUST be included.

- ***iat** - the time of the request as a NumericDate.
- ***uri** - the Access URI.
- ***access** - the new access requested per the [GNAP] Grant Request JSON "access" object.

The GS MUST respond with a GNAP Access JSON document, or one of the following errors:

- *TBD

from Error Responses Section 8.

3.2. Delete Access

The GC deletes an Access by doing an HTTP DELETE to the corresponding Access URI.

The GS MUST respond with OK 200, or one of the following errors:

- *TBD

from Error Responses Section 8.

A GS MAY indicate support for this feature by including the "DELETE" method in the Access URI OPTIONS response.

3.3. Access Options

The GC can get the supported operations for an Access by doing an HTTP OPTIONS of the corresponding Access URI.

The GS MUST respond with the supported methods

[Format TBD]

or one of the following errors:

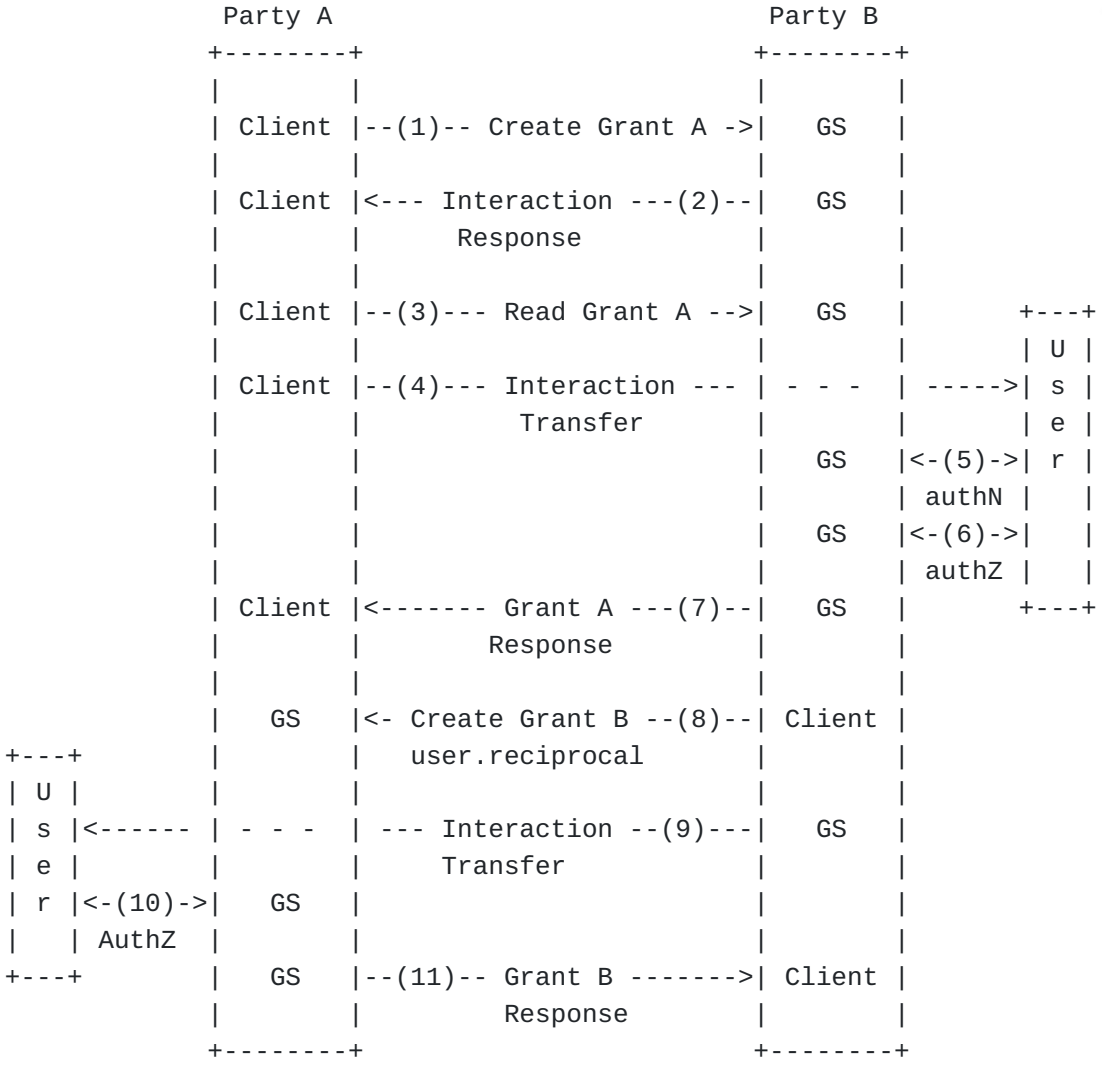
- *TBD

from Error Responses Section 8.

4. Reciprocal Grant

Party A and Party B both want to obtain a Grant from the other party. Each party will be both Grant Client and Grant Server. This would require two complete GNAP flows with an awkward redirect between them, and the User may have to authenticate multiple times as context is lost. Reciprocal Grant simplifies the User experience.

In the following sequence, steps 1 - 7 & 9 are a standard GNAP sequence.



Client = Grant Client

1. **Create Grant A** Party A makes a Create Grant request to the Party B GS URI.

2. **Interaction Response** Party B returns an interaction response containing the Grant A URI.

3. **Read Grant A** Party A does an HTTP GET of the Grant A URI.

4. **Interaction Transfer** Party A transfers User interaction to the Party B.

5. **User Authentication** Party B authenticates the User.

6. **User Authorization** If required, Party B interacts with the User to determine which identity claims and/or authorizations in the Grant A Request are to be granted.

7. **Create GrantB** Party B creates its Grant B Request with user.reciprocal set to the Grant A URI that will be in the step (2) Grant A Response, and sends it with an HTTP POST to the Party A GS URI. This enables Party A to correlate the Grant B Request and its Grant and the User.

8. **Grant S Response** Party B responds to Party A's Create Grant A Request with a Grant A Response.

9. **Interaction Transfer** Party B redirects the User to the Completion URI at Party A.

10. **User Authorization** If required, Party A interacts with the User to determine which identity claims and/or authorizations in Party B's Grant B Request are to be granted.

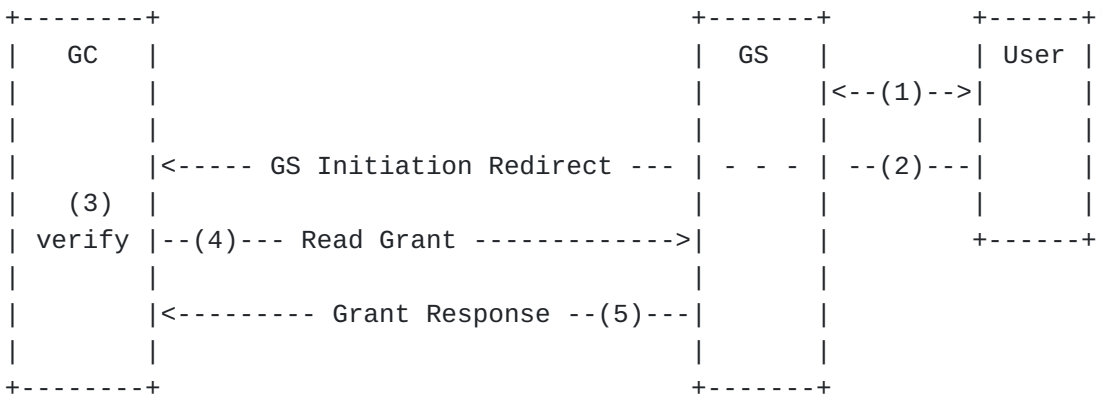
11. **Grant B Response** Party A responds with the Grant B Response.

*__reciprocal__ - a new attribute of the [GNAP] Request JSON user object. MUST be set to a Grant URI.

5. GS Initiated Grant

The User is at the GS, and wants to interact with a Registered Client. The GC has previously configured an initiation_uri at the GS, and the Grant it requires.

In this sequence, the GS creates a Grant and redirects the User to the GC's initiation_uri passing a Grant URI:



1. **User Interaction** The GS interacts with the User to determine the GC and what identity claims and / or authorizations to provide. The GS creates a Grant and corresponding Grant URI.

2. **GS Initiated Redirect** The GS redirects the User to the GC's initiation_uri, adding a query parameter with the name "grant_uri" and the value being the URL encoded Grant URI.

3. **Client Verification** The GC verifies the Grant URI starts with a GS URI from a GS the GC trusts.

4. **Read Grant** The GC does an HTTP GET of the Grant URI.

5. **Grant Response** The GS responds with a Grant Response.

*__initiation_uri__ - a URI at the GC that contains no query or fragment. How the GS learns the GC initiation_uri and require Grant is out of scope of this document.

6. User Exists

The GC may want to provide a different experience to the User depending on if a User already exists at the GS. By including one or more identifiers in the Grant Request user.identifiers object, and setting user.exists to true, the GS MAY include a user.exists attribute in a GNAP Interaction Response. The value is true if the GS has a user with one or more of the GC provided identifers, and false if not.

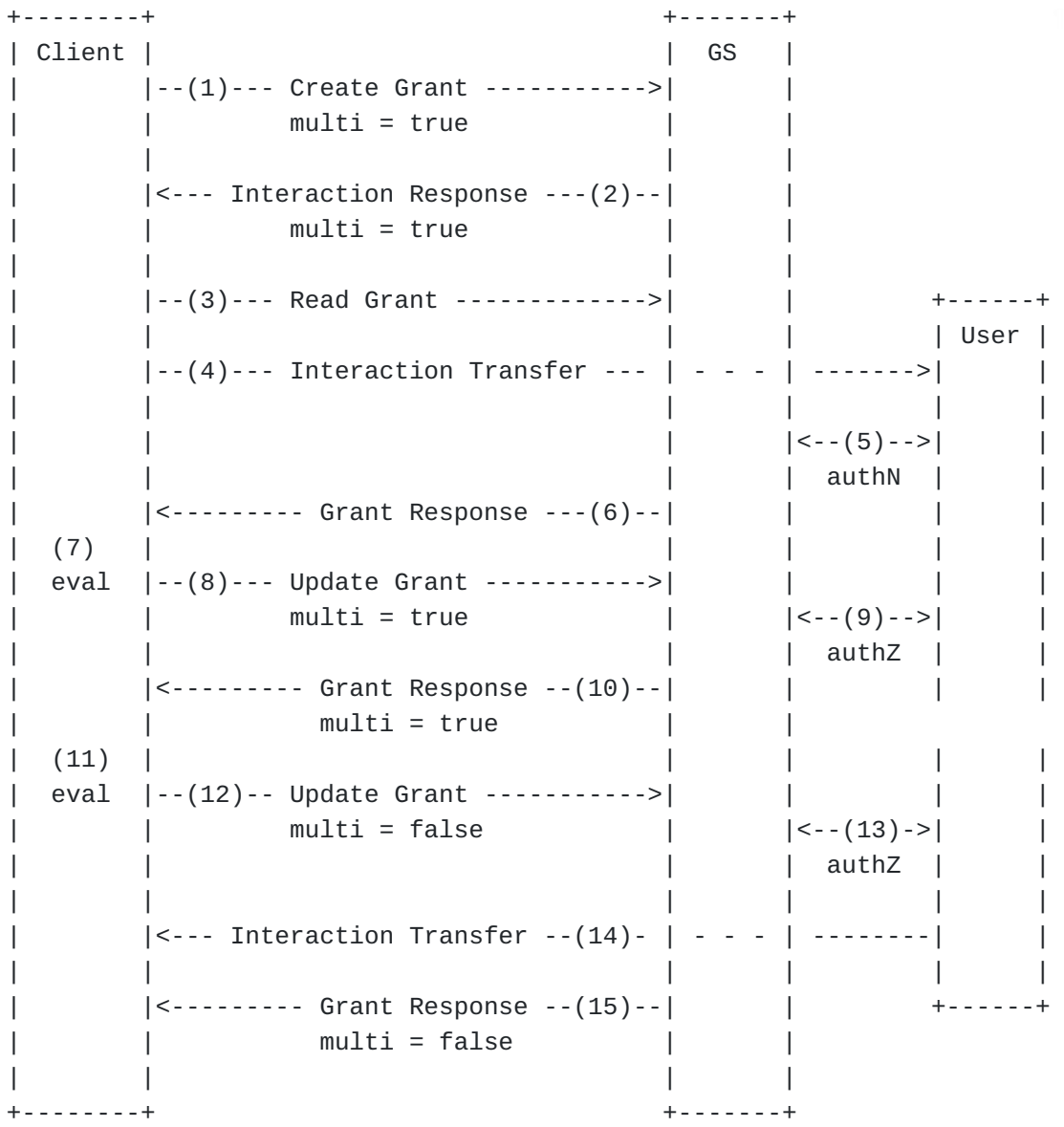
*__exists__ - a new attribute of the "user" object. If present in a GNAP Grant Request, it MUST be set to true.

A GS indicates support for this feature by returning the features.user_exists attribute in the GS Options response set to true.

7. Multiple Interactions

There are situations where the GC can not, or prefers not, to ask for all identity claims and/or authorizations it requires.

In this example sequence, the GC requests an identity claim to determine who the User is. Once the GC learns who the User is, the GC updates the Grant for additional identity claims which the GS prompts the User for and returns to the GC. Once those additional claims are received, the GC updates the Grant with the remaining identity claims required.



Client = Grant Client

1. **Create Grant** The GC creates a Grant Request (CreateGrant) including an identity claim and interaction.global.multi set to true, and sends it with an HTTP POST to the GS GS URI.

2. **Interaction Response** The GS sends an Interaction Response containing the Grant URI and an interaction object, and interaction.global.multi set to true.

3. **Read Grant** The GC does an HTTP GET of the Grant URI.

4. **Interaction Transfer** The GC transfers User interaction to the GS.

5. **User Authentication** The GS authenticates the User.

6. **Grant Response** The GS responds with a Grant Response including the identity claim from User authentication and interaction.global.multi set to true.

7. **Grant Evaluation** The GC queries its User database and does not find a User record matching the identity claim.

8. **Update Grant** The GC creates an Update Grant Request Section 2.2 including the initial identity claims required and interaction.global.multi set to true, and sends it with an HTTP PUT to the Grant URI.

9. **User AuthN** The GS interacts with the User to determine which identity claims in the Update Grant Request are to be granted.

10. **Grant Response** The GS responds with a Grant Response including the identity claims released by the User and interaction.global.multi set to true.

11. **Grant Evaluation** The GC evaluates the identity claims in the Grant Response and determines the remaining User identity claim required.

12. **Update Grant** The GC creates an Update Grant Request Section 2.2 including the remaining required identity claims and interaction.global.multi set to false, and sends it with an HTTP PUT to the Grant URI.

13. **User AuthZ** The GS interacts with the User to determine which identity claims in the Update Grant Request are to be granted.

14. **Interaction Transfer** The GS transfers User interaction to the GC.

15. **Grant Response** The GS responds with a Grant Response including the identity claims released by the User and interaction.global.multi set to false.

*__multi__ - a new boolean attribute of the GNAP interaction.global object.

A GS indicates support for this feature by returning the features.interaction_multi attribute in the GS Options response set to true.

8. Error Responses

*TBD

9. Acknowledgments

TBD

10. IANA Considerations

TBD

11. Security Considerations

TBD

12. Normative 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>.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007, <https://www.rfc-editor.org/info/rfc4949>.

[GNAP] Hardt, D., "The Grant Negotiation and Authorization Protocol", June 2020, <https://tools.ietf.org/html/draft-hardt-xauth-protocol>.

Appendix A. Document History

A.1. draft-hardt-gnap-advanced-00

*Initial version

A.2. draft-hardt-gnap-advanced-01

*renamed verb to method

*renamed Authorization to Access

*renamed Client to Grant Client (GC)

Appendix B. GS API Table

Below is a consolidated table of GS APIs from [GNAP] and this document:

request	http method	uri	response
Create Grant	POST	GS URI	interaction, wait, or Grant
List Grants	GET	GS URI	Grant list
Verify Grant	PATCH	Grant URI	Grant

request	http method	uri	response
Read Grant	GET	Grant URI	wait, or grant
Update Grant	PUT	Grant URI	Interaction, wait, or Grant
Delete Grant	DELETE	Grant URI	success
Read Access	GET	Access URI	Access
Update Access	PUT	Access URI	Access
Delete Access	DELETE	Access URI	success
GS Options	OPTIONS	GS URI	metadata
Grant Options	OPTIONS	Grant URI	metadata
Access Options	OPTIONS	Access URI	metadata

Table 1

Author's Address

Dick Hardt (editor)
SignIn.Org
United States

Email: dick.hardt@gmail.com