OAuth Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 30, 2018

W. Denniss
Google

June 28, 2018

# OAuth 2.0 Incremental Authorization
# draft-ietf-oauth-incremental-authz-00

## Abstract

OAuth 2.0 authorization requests that include every scope the client might ever need can result in over-scoped authorization and a sub-optimal end-user consent experience.  This specification enhances the OAuth 2.0 authorization protocol by adding incremental authorization, the ability to request specific authorization scopes as needed, when they're needed, removing the requirement to request every possible scope that might be needed upfront.

## 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 December 30, 2018.

## Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors.  All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document.  Please review these documents carefully, as they describe your rights and restrictions with respect to this document.  Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction . . . . . . . . . . . . . . . . . . . . . . . . 2
2. Notational Conventions . . . . . . . . . . . . . . . . . . . 3
3. Terminology . . . . . . . . . . . . . . . . . . . . . . . . . 3
4. Incremental Auth for Confidential Clients . . . . . . . . . . 3
5. Incremental Auth for Public Clients . . . . . . . . . . . . . 4
6. Usability Considerations . . . . . . . . . . . . . . . . . . 4
   6.1. Handling Denials . . . . . . . . . . . . . . . . . . . . 4
7. Alternative Approaches . . . . . . . . . . . . . . . . . . . 5
   7.1. Alternative for Public Clients . . . . . . . . . . . . . 5
   7.2. Alternative for Confidential Clients . . . . . . . . . . 5
8. Privacy Considerations . . . . . . . . . . . . . . . . . . . 5
   8.1. Requesting Authorization In Context . . . . . . . . . . . 5
   8.2. Preventing Overbroad Authorization Requests . . . . . . . 6
   8.3. Authorization Correlation . . . . . . . . . . . . . . . . 6
9. Security Considerations . . . . . . . . . . . . . . . . . . . 6
   9.1. Public Client Impersonation . . . . . . . . . . . . . . . 6
10. IANA Considerations . . . . . . . . . . . . . . . . . . . . . 7
   10.1. OAuth Parameters Registry . . . . . . . . . . . . . . . 7
11. Normative References . . . . . . . . . . . . . . . . . . . . 7
Appendix A. Acknowledgements . . . . . . . . . . . . . . . . . . 8
Appendix B. Document History . . . . . . . . . . . . . . . . . . 8
Author's Address . . . . . . . . . . . . . . . . . . . . . . . . 8

## 1. Introduction

OAuth 2.0 clients may offer multiple features that requiring user authorization, but commonly not every user will use each feature. Without incremental authentication, applications need to either request all the possible scopes they need upfront, potentially resulting in a bad user experience, or track each authorization grant separately, complicating development.

The goal of incremental authorization is to allow clients to request just the scopes they need, when they need them, while allowing them to store a single authorization grant for the user that contains the sum of the scopes granted.  Thus, each new authorization request increments the scope of the authorization grant, without the client needing to track a separate authorization grant for each group of scopes.

## 2. Notational Conventions

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 Key words for use in RFCs to Indicate Requirement Levels [RFC2119]. If these words are used without being spelled in uppercase then they are to be interpreted with their normal natural language meanings.

## 3. Terminology

In addition to the terms defined in referenced specifications, this document uses the following terms:

"OAuth" In this document, OAuth refers to OAuth 2.0 [RFC6749].

## 4. Incremental Auth for Confidential Clients

For confidential clients, such as web servers that can keep secrets, the authorization endpoint SHOULD treat scopes that the user already granted differently on the consent user interface. Typically such scopes are hidden for new authorization requests, or at least there is an indication that the user already approved them.

By itself, this property of the authorization endpoint enables incremental authorization. The client can track every scope they've ever requested, and include those scopes on every new authorization request.

To avoid the need for confidential clients to re-request already authorized scopes, authorization servers MAY support an additional "include_granted_scopes" parameter in the authorization request. This parameter, enables the client to request tokens during the authorization grant exchange that represent the full scope of the user's grant to the application including any previous grants, without the app needing to track the scopes directly.

The client indicates they wish the new authorization grant to include previously granted scopes by sending the following additional parameter in the OAuth 2.0 Authorization Request (Section 4.1.1 of [RFC6749].) using the following additional parameter:

include_granted_scopes OPTIONAL. Either "true" or "false". When "true", the authorization server SHOULD include previously granted scopes for this client in the new authorization grant.

## 5. Incremental Auth for Public Clients

Unlike with confidential clients, it is NOT RECOMMEND to automatically approve OAuth requests for public clients without user consent (see Section 10.2 of OAuth 2.0 [RFC6749], and Section 8.6 of OAuth 2.0 [RFC8252]), thus authorization grants shouldn't contain previously authorized scopes in the manner described above for confidential clients.

Public clients (and confidential clients using this technique) should instead track the scopes for every authorization grant, and only request yet to be granted scopes during incremental authorization. In the past, this would result in multiple discrete authorization grants that would need to be tracked. To enable incrementing a single authorization grant for public clients, the client supplies their existing refresh token during the authorization code exchange, and receives new authorization tokens with the scope of the previous and current authorization grants.

The client sends the previous refresh token in the OAuth 2.0 Access Token Request (Section 4.1.3 of [RFC6749].) using the following additional parameter:

existing_grant  OPTIONAL.  The refresh token from the existing authorization grant.

When processing the token exchange, in addition to the normal processing of such a request, the token endpoint MUST verify that token provided in the "existing_grant" parameter is unexpired and unrevoked, and was issued to the same client id and relates to the same user as the current authorization grant. If this verification succeeds, the new refresh token issued in the Access Token Response (Section 4.1.4 of ) SHOULD include authorization for the scopes in the previous grant.

## 6. Usability Considerations

### 6.1. Handling Denials

A core principle of OAuth is that users may deny authorization requests for any reason. This remains true for incremental authorization requests. In the case of incremental authorization, clients may already have a valid authorization and receive a denial for an incremental authorization request (that is, an "access_denied" error code as defined in Section 4.1.2.1 of OAuth 2.0 [RFC6749]). Clients should SHOULD handle such errors gracefully and not discard any existing authorization grants if the user denies an incremental authorization request. Clients SHOULD NOT immediately request the

same incremental authorization again, as this may result in an infinite denial loop (and the end-user feeling badgered).

# 7. Alternative Approaches

## 7.1. Alternative for Public Clients

It is possible for OAuth clients to maintain multiple authorizations per user for feature-specific scopes without needing the feature documented in this specification. For example, an app could maintain an authorization for the contacts and one for calendar, and store them separately.

This specification offers a convenience that a single authorization grant can be managed that represents all the scope granted so far, rather than needing to maintain multiple, however it does require that all grants are made from a single end-user account (as authorization servers cannot typically combine grants from multiple users). Apps where users may wish to authorize separate end-user accounts for different features should consider using the alternative documented above.

## 7.2. Alternative for Confidential Clients

An alternative incremental auth design for confidential clients is to ask for authorization scopes as they are needed and keep a running record of all granted scopes. In this way each incremental authorization request would include all scopes granted so far, plus the new scope needed. Authorization servers can see the existing scopes and only display the new scopes for approval (and likely to inform the user of the existing grants). This approach can be performed using RFC 6749 without additions, but requires the client to keep track of every authorization grant.

Confidential clients can also use the alternative documented for public clients in Section 7.1.

# 8. Privacy Considerations

## 8.1. Requesting Authorization In Context

The goal of incremental authorization is to enhance end-user privacy by allowing clients to request only the authorization scopes needed in the context of a particular user action, rather than asking for ever possible scope upfront. For example, an app may offer calendar and contacts integration, and an extension of OAuth like OpenID Connect for sign-in. Such an app should first sign the user in with just the scopes needed for that. If later the user interacts with

the calendar or contacts features then, and only then, should the requires scopes be requested. By using this specification, apps can improve the privacy choices of end-users by only requesting the scopes they need in context.

Clients authorizing the user with an authorization server that supports incremental auth SHOULD ask for the minimal authorization scope for the user's current context, and use this specification to add authorization scope as required.

### 8.2. Preventing Overbroad Authorization Requests

When this specification is implemented, clients should have no technical reason to make overbroad authorization requests (i.e. requesting every possible scope, even ones they don't immediately need). It is therefore RECOMMENDED for authorization servers to limit the authorization scope that can be requested in a single authorization to what would reasonably be needed by a single feature.

### 8.3. Authorization Correlation

Incremental authorization is designed for use-cases where it's the same user authorizing each request, and thus all incremental authorization grants are correlated to that one user (by being merged into a single authorization grant). For applications where users may wish to connect different user accounts for different features (e.g. contacts from one account, and calendar from another) it is RECOMMENDED to instead allow multiple unrelated authorizations, as documented in Section 7.1.

The goal of this specification is to improve end-user privacy by giving them more choice over which scopes they grant access to. Previously many apps would request an overly large number of scopes upfront (typically for all the features of the app, rather than the subset that the user is currently wishing to use). The scopes in such authorization grants are necessarily correlated with the same user as they are contained in a single authorization grant. Implementing this specification doesn't change that attribute, but it does improve user privacy overall by empowering the user to grant access in a more granular way.

## 9. Security Considerations

### 9.1. Public Client Impersonation

As documented in Section 8.6 of RFC 8252 [RFC8252], some public clients are susceptible to client impersonation, depending on the type of redirect URI used. If the "include_granted_scopes" feature

documented in Section 4 is used by an impersonating client, it may receive a greater authorization grant than the user specifically approved for that client. For this reason, the "include_granted_scopes" feature MUST NOT be enabled for such public client requests.

Note that there is no such restriction on the use of "existing_grant" feature documented in Section 5. While it is designed for public clients, it MAY be supported for all client types.

## 10. IANA Considerations

This specification makes a registration request as follows:

### 10.1. OAuth Parameters Registry

This specification registers the following parameters in the IANA OAuth Parameters registry defined in OAuth 2.0 [RFC6749].

- Parameter name: include_granted_scopes

- Parameter usage location: authorization request

- Change controller: IESG

- Specification document(s): this document

- Parameter name: existing_grant

- Parameter usage location: token request

- Change controller: IESG

- Specification document(s): this document

## 11. 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>.

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/info/rfc6749>.

[RFC8252] Denniss, W. and J. Bradley, "OAuth 2.0 for Native Apps", BCP 212, RFC 8252, DOI 10.17487/RFC8252, October 2017, <https://www.rfc-editor.org/info/rfc8252>.

## Appendix A. Acknowledgements

This document was produced in the OAuth working group under the chairpersonship of Rifaat Shekh-Yusef and Hannes Tschofenig with Benjamin Kaduk, and Eric Rescorla serving as Security Area Directors.

The following individuals contributed ideas, feedback, and wording that shaped and formed the final specification:

Yanna Wu, Marius Scurtescu, Jason Huang, Nicholas Watson, and Breno de Medeiros.

## Appendix B. Document History

[[ to be removed by the RFC Editor before publication as an RFC ]]

00

- Now a working group draft.

draft-wdenniss-oauth-incremental-auth-01

- Added usability, privacy, and security considerations.
- Documented alternative approaches.

draft-wdenniss-oauth-incremental-auth-00

- Initial draft based on the implementation of incremental and "appcremental" auth at Google.

Author's Address

William Denniss
Google
1600 Amphitheatre Pkwy
Mountain View, CA  94043
USA

Email: wdenniss@google.com
URI:   https://wdenniss.com/incremental-auth