OAuth 2.0
Incremental Auth

IETF 99 Prague, July 2017

William Denniss

Incremental Auth Problem Statement

Asking for the kitchen sink of scopes up-front is a bad thing.

Users should have the context of the authorization request.

E.g. Granting a calendar scope only makes sense in the context of interacting with a calendar-related feature.
Hi Bill

billd1600@gmail.com

Google OAuth 2.0 Playground wants to:

- View and manage the files in your Google Drive
- Manage your Blogger account
- Send email on your behalf
- Manage your contacts
- Manage your calendars
- Manage your YouTube account

Allow Google OAuth 2.0 Playground to do this?

By clicking Allow, you allow this app to use your information in accordance to their terms of service and privacy policies. You can remove this or any other app connected to your account in My Account.
Incremental Auth Definition

The ability to request additional scopes in subsequent requests resulting in a single authorization grant representing all scopes granted so far.
Typical Implementation

Consent screen should only display new scopes (or display new/existing scopes differently).

Single refresh token issued for the union of all granted scopes.
De-facto Incremental Auth for Confidential Clients

OAuth 2.0 doesn’t stop you returning an authorization grant with *more* scope, so many people have implemented this already for confidential clients.
De-facto Incremental Auth


- **Request relevant permissions**

  Permission requests should make sense to users, and should be limited to the critical information necessary to implement your application.

  **Don’t request access to information that you don’t need.** You may only request access to the Google user data that is necessary to implement existing features or services in your application. Don’t attempt to “future proof” your access to user data by requesting access to information that might benefit services or features that have not yet been implemented.

  **Request permissions in context where possible.** Request access to user data in context (via incremental auth) whenever you can, so that users understand why you need the data.
What about public clients?

Confidential client incremental auth techniques not suitable for public clients (risk of granting greater authorization to a counterfeit client than the user saw).

No standard way to implement incremental auth for public clients… until now.
OAuth 2.0 for Incremental Auth
Internet-Draft

New internet draft:
draft-wdenniss-oauth-incremental-auth-00

Defines a protocol for public client (native app) incremental auth, a.k.a. “appcremental auth”.

OAuth 2.0 for Incremental Auth
Internet-Draft

In addition to defining a protocol for public clients, we can use this opportunity to formalize incremental auth for all client types.

Implementors of confidential clients may benefit from more detailed analysis and security considerations.
New token endpoint param: `existing_grant`.

When exchanging the authorization code from subsequent (i.e. incremental) requests, pass the previous refresh token in `existing_grant`.

Resulting access and refresh tokens will contain a union of the scope.
Incremental Auth for Native Apps

App -> Authorization Endpoint

Initial Authorization Request
Authorization Request (Scope A)
Response: authorization_code
Token Request: authorization_code
Response: access_token, refresh_token

App -> Authorization Endpoint

Incremental Authorization Request
Authorization Request (Scope B)
Response: authorization_code (Scope B)
Token Request: authorization_code (Scope B), existing_grant (Scope A)
Response: access_token, refresh_token (Scope A, B)

Authorization Endpoint -> Token Endpoint
Don’t just union any two grants: the existing grant must be valid in its own right (not expired or revoked), and client_id must match.
Alternatives Considered: Authorization Param not Token Param

Could also pass previous grant in the authorization request.

Benefit: authorization server can know full context ahead of time.

Drawback: Tokens passed in a GET request.
Alternatives Considered: Access Token not Refresh Token

Alternative: access token as proof of existing grant.

Benefit: Works with other flows

Drawback: More susceptible to attack.

Native Apps SHOULD be using the code flow anyway.

OAuth 2.0 Incremental Auth
Running Code

Google’s OAuth server already supports incremental auth for public clients, as defined in this spec.

Try out my proof of concept:
https://github.com/WilliamDenniss/AppAuth-iOS/tree/appcremental
Demo

By clicking Allow, you allow this app to use your information in accordance to their terms of service and privacy policies. You can remove this or any other app connected to your account in My Account.

Log:
```
A9DX_R8WFvNUAhvCeU7IXW8NB6_DOEd232ZuLr-
Nz8h5Sy0AFOedPa530q25Nqrb128i1MW8tChxvEsks8
11:09:17: Performing token introspection: https://
www.googleapis.com/oauth2/v2/tokeninfo?access_token=ya29.GIuBLz2gV5yfQQVEQ0MQ/QG0yfwi
75vY/jb0b4BDX_R8WFvWb/h4In/4vC/n/7/XW8NB6_DOEd2
3ZuLr-
Nz8h5Sy0AFOedPa530q25Nqrb128i1MW8tChxvEsks8
11:09:18: Success: {
    "access_type" = "offline",
    aud = "581786658708-6p376uvjy9v1nndcj2a0o0apps8n5c.app
s.googleusercontent.com",
    azp = "581786658708-6p376uvjy9v1nndcj2a0o0apps8n5c.app
s.googleusercontent.com",
    exp = 1500372507;
    "expires_in" = 3599;
    scope = "https://www.googleapis.com/auth/calendar"
}
```

https://youtu.be/SPSCW4NUooU
Confidential Client Specification Details

Documents best practices, security considerations, and:

New authorization endpoint parameter
include_granted_scopes.

Confidential Client Specification Details

Allows client to specify they want the new grant to contain all granted scopes, including ones not requested directly in the current request.

If you track your scopes (like you have to with public clients), alternative is to simply include granted scopes manually.
Google implemented `include_granted_scopes` as a convenience to confidential clients.

https://developers.google.com/identity/protocols/OAuth2WebServer#incrementalAuth

By default only requested scopes are included in the grant.
Confidential Client Specification Details

Alternative is to always or never implement this behavior, each with drawbacks:

*Always*: No way for client to indicate they *don’t* want the full scope.

*Never*: Confidential clients need to track granted scopes manually (like native ones).
OAuth 2.0 Incremental Auth

Discuss