Identity Assertion Authorization Grant

Abstract

This specification provides a mechanism for an application to use an identity assertion to obtain an access token for a third-party API using Token Exchange [RFC8693] and JWT Profile for OAuth 2.0 Authorization Grants [RFC7523].

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://aaronpk.github.io/draft-parecki-oauth-identity-assertion-authz-grant/draft-parecki-oauth-identity-assertion-authz-grant.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-parecki-oauth-identity-assertion-authz-grant/.

Discussion of this document takes place on the Web Authorization Protocol Working Group mailing list (mailto:oauth@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/oauth/.

Source for this draft and an issue tracker can be found at https://github.com/aaronpk/draft-parecki-oauth-identity-assertion-authz-grant.

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 3 September 2024.

Copyright Notice

Copyright (c) 2024 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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction
2. Conventions and Definitions
   2.1. Roles
3. Overview
4. User Authentication
5. Token Exchange
   5.1. Processing Rules
   5.2. Response
      5.2.1. Error Response
   5.3. Identity Assertion Authorization Grant JWT
6. Access Token Request
   6.1. Processing Rules
   6.2. Response
7. Security Considerations
   7.1. Client Authentication
   7.2. Step-Up Authentication
8. IANA Considerations
   8.1. Media Types
   8.2. OAuth URI Registration
9. References
   9.1. Normative References
   9.2. Informative References
Appendix A. Use Cases
   A.1. Enterprise Deployment
      A.1.1. Preconditions
1. Introduction

The draft specification Identity Chaining Across Trust Domains [I-D.ietf-oauth-identity-chaining] defines how to request a JWT authorization grant from an Authorization Server and exchange it for an Access Token at another Authorization Server in a different trust domain. The specification is an application of a combination of OAuth 2.0 Token Exchange [RFC8693] and JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants [RFC7523]. The draft supports multiple different use cases by leaving many details of the token exchange request and JWT authorization grant unspecified. This specification defines the additional details necessary to support interoperable implementations when using identity tokens as the input to the token exchange request.

2. Conventions and Definitions

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.

2.1. Roles

**Client** Application that wants to obtain an OAuth 2.0 access token on behalf of a signed-in user to an external/3rd party application's API (Resource Server below). In [I-D.ietf-oauth-identity-chaining], this is the Client in trust domain A.

**Resource Application** Application that provides an OAuth 2.0 Protected Resource. In [I-D.ietf-oauth-identity-chaining], this is the Protected Resource in trust domain B.

**Authorization Server (IdP)** The Identity Provider that is trusted by a set of applications in an organization's app ecosystem. In [I-D.ietf-oauth-identity-chaining], this is the Authorization Server in trust domain A, which is also trusted by the Authorization Server of the Protected Resource in trust domain B.

3. Overview

The example flow is for an enterprise acme
<table>
<thead>
<tr>
<th>Role</th>
<th>App URL</th>
<th>Tenant URL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td><a href="https://wiki.example">https://wiki.example</a></td>
<td><a href="https://acme.wiki.example">https://acme.wiki.example</a></td>
<td>SaaS Wiki app that embeds content from one or more resource applications</td>
</tr>
<tr>
<td>Resource Application</td>
<td><a href="https://chat.example">https://chat.example</a></td>
<td><a href="https://acme.chat.example">https://acme.chat.example</a></td>
<td>Chat and communication app</td>
</tr>
<tr>
<td>Identity Provider</td>
<td><a href="https://idp.example">https://idp.example</a></td>
<td><a href="https://acme.idp.example">https://acme.idp.example</a></td>
<td>Identity Provider</td>
</tr>
</tbody>
</table>

Table 1

Sequence Diagram
1. User logs in to the Client, the Client obtains the Identity Assertion (e.g. SAML Assertion or OpenID Connect ID Token)

2. Client uses the Identity Assertion to request an Identity Assertion Authorization Grant for the Resource Application from the IdP

3. Client exchanges the Identity Assertion Authorization Grant JWT for an Access Token at the Resource Application's token endpoint

4. Client makes an API request with the Access Token
4. User Authentication

The Client initiates an authentication request with the IdP using OpenID Connect or SAML.

The following is an example using OpenID Connect:

302 Redirect
Location: https://acme.idp.example/authorize?response_type=code&scope=op

The user authenticates with the IdP, and is redirected back to the Client with an authorization code, which it can then exchange for an ID Token.

Note: The Enterprise IdP may enforce security controls such as multi-factor authentication before granting the user access to the Client.

POST /token HTTP/1.1
Host: acme.idp.example
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code
&code=.....

HTTP/1.1 200 Ok
Content-Type: application/json

{
  "id_token": "eyJraWQiOiJzMTZ0cVNtODhwREo4VGZCXzdrSEtQ...",
  "token_type": "Bearer",
  "access_token": "7SliwCQP1brGdjBtsaMnXo",
  "scope": "openid"
}

5. Token Exchange

The Client makes a Token Exchange [RFC8693] request to the IdP's Token Endpoint with the following parameters:

*requested_token_type=urn:ietf:params:oauth:token-type:id-jag

*resource - The token endpoint of the Resource Application.

*scope - The space-separated list of scopes at the Resource Application to include in the token

*subject_token - The identity assertion (SAML Assertion or OpenID Connect ID Token) for the target end-user
subject_token_type - For SAML2 Assertion:

Client authentication (e.g. client_id and client_secret, or the more secure private_key_jwt method using client_assertion and client_assertion_type)

The example below uses an ID Token as the Identity Assertion, and uses private_key_jwt as the client authentication method, (tokens truncated for brevity):

```
POST /oauth2/token HTTP/1.1
Host: acme.idp.example
Content-Type: application/x-www-form-urlencoded

grant_type=urn:ietf:params:oauth:grant-type:token-exchange
&requested_token_type=urn:ietf:params:oauth:token-type:id-jag
&resource=https://acme.chat.example/oauth2/token
&scope=chat.read+chat.history
&subject_token=eyJraWQiOiJzMTZ0cVNtODhwREo4VGZCZdrSEtQ...
&subject_token_type=urn:ietf:params:oauth:token-type:id_token
&client_assertion_type=urn:ietf:params:oauth:client-assertion-type:jwt-bearer
&client_assertion=eyJhbGciOiJSUzI1NiIsImtpZCI6IjIyIn0...
```

5.1. Processing Rules

The IdP **MUST** validate the subject token, and **MUST** validate that the audience of the Subject Token (e.g. the aud claim of the ID Token) matches the client_id of the client authentication of the request.

The IdP evaluates administrator-defined policy for the token exchange request and determines if the client should be granted access to act on behalf of the subject for the target audience and scopes.

The IdP may also introspect the authentication context described in the SSO assertion to determine if step-up authentication is required.

5.2. Response

If access is granted, the IdP creates a signed Identity Assertion Authorization Grant JWT and returns it in the token exchange response defined in Section 2.2 of [RFC8693]:
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache

{
  "issued_token_type": "urn:ietf:params:oauth:token-type:id-jag",
  "access_token": "eyJhbGciOiJIUzI1NiIsI...",
  "token_type": "N_A",
  "scope": "chat.read chat.history",
  "expires_in": 300
}

*issued_token_type* - urn:ietf:params:oauth:token-type:id-jag

*access_token* - The Identity Assertion Authorization Grant JWT. (Token Exchange requires the access_token response parameter for historical reasons, even though this is not an OAuth access token.)

*token_type* - N_A (As defined by Token Exchange.)

*scope* - The list of scopes granted by the IdP. This may be fewer scopes than the application requested based on various policies in the IdP.

*expires_in* - The lifetime in seconds of the authorization grant.

5.2.1. Error Response

On an error condition, the IdP returns an OAuth 2.0 Token Error response as defined in Section 5.2 of [RFC6749], e.g:

HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
  "error": "invalid_grant",
  "error_description": "Audience validation failed"
}

5.3. Identity Assertion Authorization Grant JWT

The Identity Assertion Authorization Grant JWT is issued and signed by the IdP, and describes the intended audience of the authorization grant as well as the client to which it was issued and the subject
identifier of the original identity assertion, using the following claims:

*iss - The IdP issuer URL

*sub - The User ID at the IdP

*aud - Token endpoint of the Resource Application's authorization server

*client_id - Client ID as registered with the Resource Application's authorization server.

*scopes - Array of scopes at the Resource Application granted to the Client

*jti - Unique ID of this JWT

*exp, iat - as defined by JWT

The typ of the JWT indicated in the JWT header MUST be oauth-id-jag+jwt.

An example JWT shown with expanded header and payload claims is below:

```
{
  "typ": "oauth-id-jag+jwt"
}
.
{
  "jti": "9e43f01b64a33f20116179",
  "iss": "https://acme.idp.example",
  "sub": "U019488227",
  "aud": "https://acme.chat.example/oauth2/token",
  "client_id": "f53f19f9311af35",
  "exp": 1311281970,
  "iat": 1311280970,
  "scopes": [ "chat.read", "chat.history" ]
}
.
```

signature

Implementation notes:

*If the IdP is multi-tenant and uses the same issuer for all tenants, the Resource Application will already have IdP-specific logic to determine the tenant from OIDC/SAML (e.g. the hd claim in Google) and will need to use that if the IdP also has only one client registration for the Resource Application.*
sub should be an opaque ID, as iss+sub is unique. The IdP might want to also include the user's email here, which it should do as a new email claim. This would let the app dedupe existing users who may have an account with an email address but have not done SSO yet.

6. Access Token Request

The Client makes an access token request to the Resource Application's token endpoint using the previously obtained Identity Assertion Authorization Grant as a JWT Assertion [RFC7523].

*grant_type=urn:ietf:params:oauth:grant-type:jwt-bearer

*assertion - The Identity Assertion Authorization Grant JWT obtained in the previous token exchange step

*Client Authentication - the Client authenticates with its credentials as registered with the Resource Application's authorization server

For example:

POST /oauth2/token HTTP/1.1
Host: acme.chat.example
Authorization: Basic yZS1yYW5kb20tc2VjcmV0v3JOkF0XG5Qx2
grant_type=urn:ietf:params:oauth:grant-type:jwt-bearer
assertion=eyJhbGciOiJIUzI1NiIsI...

6.1. Processing Rules

All of Section 5.2 of [RFC7521] applies, in addition to the following processing rules:

*Validate the JWT typ is oauth-id-jag+jwt (per [RFC8725])

*The aud claim MUST identify the token endpoint of the Resource Application as the intended audience of the JWT.

6.2. Response

The Resource Application token endpoint responds with an OAuth 2.0 Token Response, e.g.:
7. Security Considerations

7.1. Client Authentication

This specification SHOULD only be supported for confidential clients. Public clients SHOULD redirect the user with an OAuth 2.0 Authorization Request.

7.2. Step-Up Authentication

In the initial token exchange request, the IdP may require step-up authentication for the subject if the authentication context in the subject's assertion does not meet policy requirements. An insufficient_user_authentication OAuth error response may be returned to convey the authentication requirements back to the client similar to OAuth 2.0 Step-up Authentication Challenge Protocol [RFC9470].

TBD: It may make more sense to request the Identity Assertion Authorization Grant as an additional response_type on the authorization request if using OIDC for SSO when performing a step-up to skip the need for additional token exchange round-trip.
8. IANA Considerations

8.1. Media Types

This section registers oauth-id-jag+jwt, a new media type [RFC2046] in the "Media Types" registry [IANA.MediaTypes] in the manner described in [RFC6838]. It can be used to indicate that the content is a Identity Assertion Authorization Grant JWT.

8.2. OAuth URI Registration

This section registers urn:ietf:params:oauth:token-type:id-jag in the "OAuth URI" subregistry of the "OAuth Parameters" registry [IANA.oauth-parameters].

*URN: urn:ietf:params:oauth:token-type:id-jag

*Common Name: Token type URI for a Identity Assertion JWT Authorization Grant

*Change Controller: IESG

*Specification Document: This document

9. References

9.1. Normative References

[I-D.ietf-oauth-identity-chaining]

[IANA.MediaTypes] "*** BROKEN REFERENCE ***".


9.2. Informative References


Appendix A. Use Cases

A.1. Enterprise Deployment

Enterprises often have hundreds of SaaS applications. SaaS applications often have integrations to other SaaS applications that are critical to the application experience and jobs to be done. When a SaaS app needs to request an access token on behalf of a user to a 3rd party SaaS integration's API, the end-user typically needs to complete an interactive delegated OAuth 2.0 flow, as the SaaS application is not in the same security or policy domain as the 3rd party SaaS integration.

It is industry best practice for an enterprise to connect their ecosystem of SaaS applications to their Identity Provider (IdP) to centralize identity and access management capabilities for the organization. End-users get a better experience (SSO) and administrators get better security outcomes such multi-factor authentication and zero-trust. SaaS applications today enable the administrator to establish trust with an IdP for user authentication.

This specification can be used to extend the SSO relationship of multiple SaaS applications to include API access between these applications as well. This specification enables federation for Authorization Servers across policy or administrative boundaries. The same enterprise IdP that is trusted by applications for SSO can be extended to broker access to APIs. This enables the enterprise to centralize more access decisions across their SaaS ecosystem and provides better end-user experience for users that need to connect multiple applications via OAuth 2.0.

A.1.1. Preconditions

*The Client has a registered OAuth 2.0 Client with the IdP Authorization Server

*The Client has a registered OAuth 2.0 Client with the Resource Application

*Enterprise has established a trust relationship between their IdP and the Client for SSO and Identity Assertion Authorization Grant

*Enterprise has established a trust relationship between their IdP and the Resource Application for SSO and Identity Assertion Authorization Grant

*Enterprise has granted the Client permission to act on behalf of users for the Resource Application with a set of scopes
A.2. Email and Calendaring Applications

Email clients can be used with arbitrary email servers, and cannot require pre-established relationships between each email client and each email server. When an email client uses OAuth to obtain an access token to an email server, this provides the security benefit of being able to use strong multi-factor authentication methods provided by the email server's authorization server, but does require that the user go through a web-based flow to log in to the email client. However, this web-based flow is often seen as disruptive to the user experience when initiated from a desktop or mobile native application, and so is often attempted to be minimized as much as possible.

When the email client needs access to a separate API, such as a third-party calendaring application, traditionally this would require that the email client go through another web-based OAuth redirect flow to obtain authorization and ultimately an access token.

To streamline the user experience, this specification can be used to enable the email client to use the identity assertion to obtain an access token for the third-party calendaring application without any user interaction.

A.2.1. Preconditions

*The Client does not have a pre-registered OAuth 2.0 client at the IdP Authorization Server or the Resource Application

*The Client has obtained an Identity Assertion (e.g. ID Token) from the IdP Authorization Server

*The Resource Application is configured to allow the Identity Assertion Authorization Grant from unregistered clients

Appendix B. Relationship to RFC7522

"SAML 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants" [RFC7522] describes a mechanism for using a SAML assertion as an authorization grant to obtain an access token. Directly exchanging a SAML assertion for an access token is limited in the ways this can be securely deployed.

This specification adds an intermediate step of exchanging a SAML assertion for the intermediate Identity Assertion Authorization Grant, which is then later exchanged for an access token. By adding this intermediate step, this provides the security benefit of being able to indicate which API the access token is being requested, enabling the authorization server to enforce policies before issuing the authorization grant. Without this step, policies must be enforced...
at each resource application's authorization server, which doesn't scale well and is impossible in some deployments.

Acknowledgments

The authors would like to thank the following people for their contributions and reviews of this specification: Brian Campbell.

Authors' Addresses

Aaron Parecki
Okta

Email: aaron@parecki.com

Karl McGuinness
Okta

Email: public@karlmcguinness.com