Web Authorization Protocol
Internet-Draft
Intended status: Standards Track
Expires: 3 January 2026

A. Parecki
Okta
K. McGuinness
Independent
B. Campbell
Ping Identity
2 July 2025

Identity Assertion Authorization Grant
draft-parecki-oauth-identity-assertion-authz-grant-05

Abstract

This specification provides a mechanism for an application to use an identity assertion to obtain an access token for a third-party API by coordinating through a common enterprise identity provider 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://drafts.aaronpk.com/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/. Subscribe at https://www.ietf.org/mailman/listinfo/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.

This Internet-Draft will expire on 3 January 2026.

Copyright Notice

Table of Contents

1. Introduction

In typical enterprise scenarios, applications are configured for single sign-on to the enterprise identity provider (IdP) using OpenID Connect or SAML. This enables users to access all the necessary enterprise applications using a single account at the IdP, and enables the enterprise to manage which users can access which applications.

When one application wants to access a user's data at another application, it will start an interactive OAuth flow ([RFC6749]) to obtain an access token for the application on behalf of the user. This OAuth flow enables a direct app-to-app connection between the two apps, and is not visible to the IdP used to log in to each app.

This specification enables this kind of "Cross App Access" to be managed by the enterprise IdP, similar to how the IdP manages single sign-on to individual applications.

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 combines 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 in enterprise scenarios when two applications are configured for single sign-on to the same enterprise identity provider. In particular, this specification uses identity assertions as the input to the token exchange request. This way, the same enterprise identity provider that is trusted by applications for single sign-on can be extended to broker access to APIs.

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 The 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. This is also sometimes referred to as the "Requesting Application".

Resource Application The 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. The Resource Application is made up of both an Authorization Server and a Resource Server as defined in Section 1.1 of [RFC6749].

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.

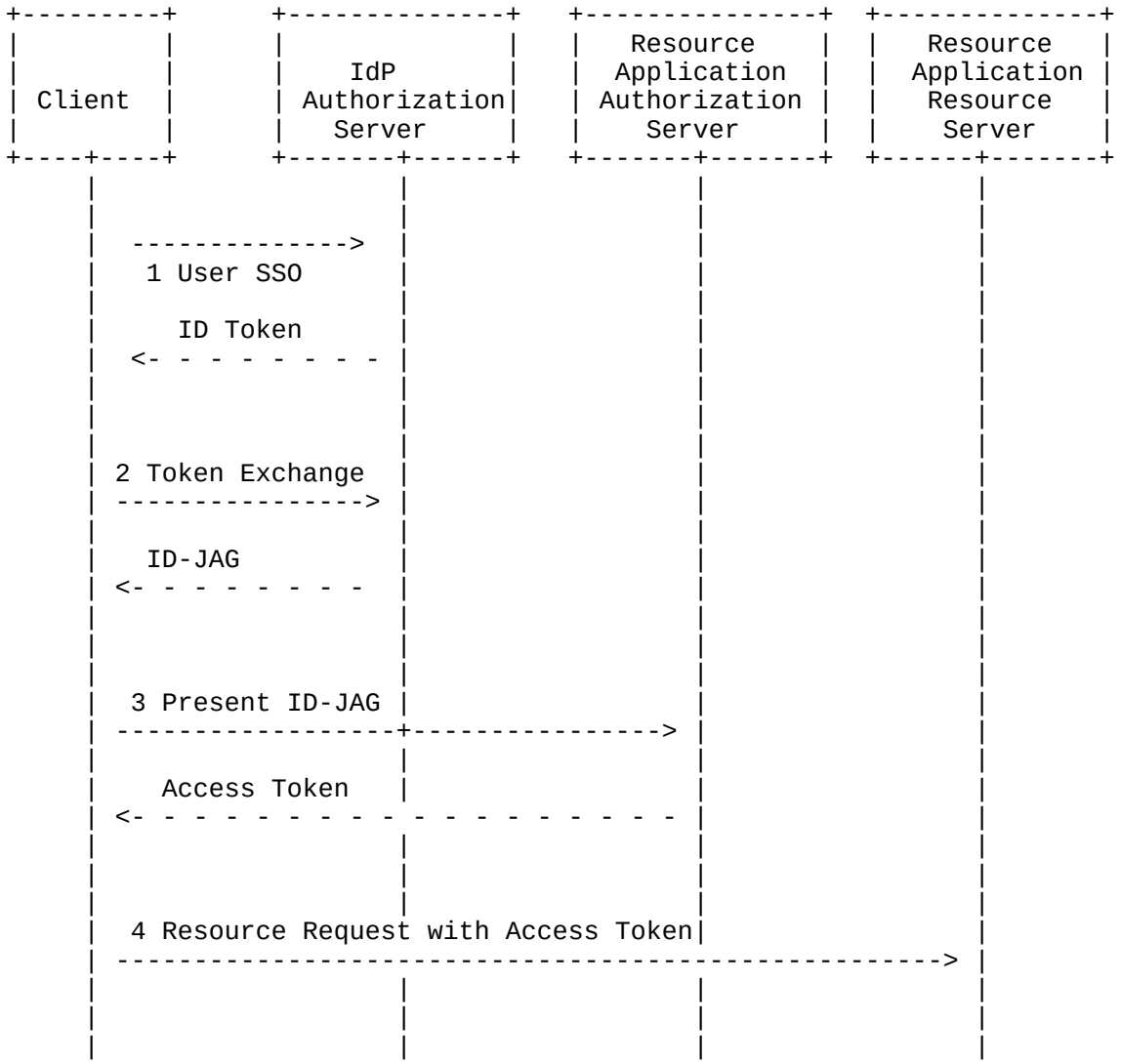
3. Overview

The example flow is for an enterprise acme, which uses a wiki app and chat app from different vendors, both of which are integrated into the enterprise’s Identity Provider using OpenID Connect.

Role	App URL	Tenant URL	Description
Client	https://wiki.example	https://acme.wiki.example	Wiki app that embeds content from one or more resource applications
Resource Application	https://chat.example	https://acme.chat.example	Chat and communication app
Identity Provider	https://idp.example	https://acme.idp.example	Identity Provider

Table 1

Sequence Diagram



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

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

This specification is constrained to deployments where all Resource Application Resource Servers are leveraging the same IdP Authorization Server for Single-Sign-On (SSO) and session management services. The IdP provides a consistent trust boundary enabling the set of Resource Application Authorization Servers to honor the JWT Authorization Grant (ID-JAG) issued by the IdP. This specification also assumes that the Resource Server Authorization Servers delegate user authorization authority to the IdP (e.g. the IdP is trusted to ensure the scopes identified in the ID-JAG have been correctly authorized before issuing the ID-JAG 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=openid&client_id=..
.
```

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: REQUIRED - The value urn:ietf:params:oauth:token-type:id-jag indicates that an ID Assertion JWT is being requested.

audience: REQUIRED - The Issuer URL of the Resource Application's authorization server as defined in Section 2 of [RFC8414].

resource: OPTIONAL - The Resource Identifier of the Resource Application's resource server as defined in Section 2 of [RFC8707].

scope: OPTIONAL - The space-separated list of scopes at the Resource Application that is being requested.

subject_token: REQUIRED - The identity assertion (e.g. the OpenID Connect ID Token or SAML assertion) for the target end-user.

subject_token_type: REQUIRED - An identifier, as described in Section 3 of [RFC8693], that indicates the type of the security token in the subject_token parameter. For an OpenID Connect ID Token: urn:ietf:params:oauth:token-type:id_token, or for a SAML assertion: urn:ietf:params:oauth:token-type:saml2.

The additional parameters defined in Section 2.1 of [RFC8693] actor_token and actor_token_type are not used in this specification.

Client authentication to the authorization server is done using the standard mechanisms provided by OAuth 2.0. Section 2.3.1 of [RFC6749] defines password-based authentication of the client (client_id and client_secret), however, client authentication is extensible and other mechanisms are possible. For example, [RFC7523] defines client authentication using bearer JSON Web Tokens using client_assertion and client_assertion_type.

The example below uses an ID Token as the Identity Assertion, and uses a JWT Bearer Assertion ([RFC7523]) 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
&audience=https://acme.chat.example/
&scope=chat.read+chat.history
&subject_token=eyJraWQiOiJzMTZ0cVNtODhwREo4VGZCXzdrSEtQ...
&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: REQUIRED - urn:ietf:params:oauth:token-type:id-jag

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

token_type: REQUIRED - N_A (because this is not an OAuth access token.)

scope: OPTIONAL if the scope of the issued token is identical to the scope requested by the client; otherwise, it is REQUIRED. Various policies in the IdP may result in different scopes being issued from the scopes the application requested.

expires_in: RECOMMENDED - The lifetime in seconds of the authorization grant.

refresh_token: OPTIONAL according to Section 2.2 of [RFC8693]. In the context of this specification, this parameter SHOULD NOT be used.

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 resource owner, using the following claims:

iss: REQUIRED - The IdP issuer URL as defined in Section 4.1.1 of [RFC7519]

sub: REQUIRED - The subject identifier (e.g. user ID) of the resource owner at the Resource Application as defined in Section 4.1.2 of [RFC7519]

aud: REQUIRED - The Issuer URL (Section 2 of [RFC8414]) of the Resource Application's authorization server as defined in Section 4.1.3 of [RFC7519]

resource: OPTIONAL - The Resource Identifier (Section 2 of [RFC8707]) of the Resource Application's resource server (either a single URI or an array of URIs)

client_id: REQUIRED - An identifier of the client that this JWT was issued to, which MUST be recognized by the Resource Application's authorization server. For interoperability, the client identifier SHOULD be a client_id as defined in Section 4.3 of [RFC8693].

jti: REQUIRED - Unique ID of this JWT as defined in Section 4.1.7 of [RFC7519]

exp: REQUIRED - as defined in Section 4.1.4 of [RFC7519]

iat: REQUIRED - as defined in Section 4.1.6 of [RFC7519]

scope: OPTIONAL - a JSON string containing a space-separated list of scopes associated with the token, in the format described in Section 3.3 of [RFC6749]

The typ of the JWT indicated in the JWT header MUST be oauth-id-jag+jwt. Using typed JWTs is a recommendation of the JSON Web Token Best Current Practices [RFC8725].

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

```
{
  "typ": "oauth-id-jag+jwt"
}
.
{
  "jti": "9e43f81b64a33f20116179",
  "iss": "https://acme.idp.example",
  "sub": "U019488227",
  "aud": "https://acme.chat.example/",
  "client_id": "f53f191f9311af35",
  "exp": 1311281970,
  "iat": 1311280970,
  "scope": "chat.read chat.history"
}
.
signature
```

The authorization server MAY add additional claims as necessary.

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 the OpenID Connect ID Token (e.g. a custom hd claim in Google) or SAML assertion, 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 as defined by [RFC7523].

grant_type: REQUIRED - The value of grant_type is urn:ietf:params:oauth:grant-type:jwt-bearer

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

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 Issuer URL of the Resource Application’s authorization server as the intended audience of the JWT.

* The client_id claim MUST identify the same client as the client authentication in the request.

6.2. Response

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

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
  "token_type": "Bearer",
  "access_token": "2YotnFZFEjr1zCsicMWpAA",
  "expires_in": 86400,
  "scope": "chat.read chat.history"
}
```

7. Authorization Server (IdP) Metadata

An IdP can advertise its support for this profile in its OAuth Authorization Server Metadata [RFC8414]. Identity and Authorization Chaining Across Domains [I-D.ietf-oauth-identity-chaining] defines a new metadata property identity_chaining_requested_token_types_supported for this purpose.

To advertise support for the Identity Assertion Authorization Grant, the authorization server SHOULD include the following value in the identity_chaining_requested_token_types_supported property:

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

8. Security Considerations

8.1. Client Authentication

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

8.2. 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].

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

{
  "error": "insufficient_user_authentication",
  "error_description": "Subject doesn't meet authentication requirements",
  "max_age": 5
}
```

The Client would need to redirect the user back to the IdP to obtain a new assertion that meets the requirements and retry the token exchange.

TBD: It may make more sense to request the Identity Assertion Authorization Grant in the authorization request if using OpenID Connect for SSO when performing a step-up to skip the need for additional token exchange round-trip.

8.3. Cross-Domain Use

This specification is intended for cross-domain uses where the Client, Resource App, and Identity Provider are all in different trust domains. In particular, the Identity Provider MUST NOT issue access tokens in response to an ID-JAG it issued itself. Doing so could lead to unintentional broadening of the scope of authorization.

9. IANA Considerations

9.1. 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.

9.2. 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

9.3. JSON Web Token Claims Registration

This section registers resource in the "JSON Web Token Claims" subregistry of the "JSON Web Token (JWT)" registry [IANA.JWT]. The "JSON Web Token Claims" subregistry was established by [RFC7519].

* Claim Name: resource
* Claim Description: Resource
* Change Controller: IESG
* Specification Document(s): Section 5.3

10. References

10.1. Normative References

[I-D.ietf-oauth-identity-chaining]
Schwenkschuster, A., Kasselman, P., Burgin, K., Jenkins, M. J., and B. Campbell, "OAuth Identity and Authorization Chaining Across Domains", Work in Progress, Internet-Draft, draft-ietf-oauth-identity-chaining-04, 27 February 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-oauth-identity-chaining-04>.

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

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

[IANA.oauth-parameters]
IANA, "OAuth Parameters",
<https://www.iana.org/assignments/oauth-parameters>.

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, DOI 10.17487/RFC2046, November 1996, <https://www.rfc-editor.org/rfc/rfc2046>.

[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>.

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

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <https://www.rfc-editor.org/rfc/rfc6838>.

[RFC7519] 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>.

[RFC7521] Campbell, B., Mortimore, C., Jones, M., and Y. Goland, "Assertion Framework for OAuth 2.0 Client Authentication and Authorization Grants", RFC 7521, DOI 10.17487/RFC7521, May 2015, <https://www.rfc-editor.org/rfc/rfc7521>.

[RFC7523] Jones, M., Campbell, B., and C. Mortimore, "JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants", RFC 7523, DOI 10.17487/RFC7523, May 2015, <https://www.rfc-editor.org/rfc/rfc7523>.

[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>.

[RFC8693] Jones, M., Nadalin, A., Campbell, B., Ed., Bradley, J., and C. Mortimore, "OAuth 2.0 Token Exchange", RFC 8693, DOI 10.17487/RFC8693, January 2020, <https://www.rfc-editor.org/rfc/rfc8693>.

[RFC8707] Campbell, B., Bradley, J., and H. Tschofenig, "Resource Indicators for OAuth 2.0", RFC 8707, DOI 10.17487/RFC8707, February 2020, <https://www.rfc-editor.org/rfc/rfc8707>.

[RFC8725] Sheffer, Y., Hardt, D., and M. Jones, "JSON Web Token Best Current Practices", BCP 225, RFC 8725, DOI 10.17487/RFC8725, February 2020, <https://www.rfc-editor.org/rfc/rfc8725>.

10.2. Informative References

[RFC8414] Jones, M., Sakimura, N., and J. Bradley, "OAuth 2.0 Authorization Server Metadata", RFC 8414, DOI 10.17487/RFC8414, June 2018, <https://www.rfc-editor.org/rfc/rfc8414>.

[RFC9470] Bertocci, V. and B. Campbell, "OAuth 2.0 Step Up Authentication Challenge Protocol", RFC 9470, DOI 10.17487/RFC9470, September 2023, <https://www.rfc-editor.org/rfc/rfc9470>.

[RFC9728] Jones, M.B., Hunt, P., and A. Parecki, "OAuth 2.0 Protected Resource Metadata", RFC 9728, DOI 10.17487/RFC9728, April 2025, <https://www.rfc-editor.org/rfc/rfc9728>.

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 capabilites 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 distruptive 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

A.3. LLM Agent using Enterprise Tools

AI agents, including those based on large language models (LLMs), are designed to manage user context, memory, and interaction state across multi-turn conversations. To perform complex tasks, these agents often integrate with external systems such as SaaS applications, internal services, or enterprise data sources. When accessing these systems, the agent operates on behalf of the end user, and its actions are constrained by the user's identity, role, and permissions as defined by the enterprise. This ensures that all data access and operations are properly scoped and compliant with organizational access controls.

A.3.1. Preconditions

* The LLM Agent has a registered OAuth 2.0 Client (com.example.ai-agent) with the Enterprise IdP (cyberdyne.idp.example)
* The LLM Agent has a registered OAuth 2.0 Client (4960880b83dc9) with the External Tool Application (saas.example.net)
* Enterprise has established a trust relationship between their IdP and the LLM Agent for SSO
* Enterprise has established a trust relationship between their IdP and the External Tool Application for SSO and Identity Assertion Authorization Grant
* Enterprise has granted the LLM Agent permission to act on behalf of users for the External Tool Application with a specific set of scopes

A.3.2. LLM Agent establishes a User Identity with Enterprise IdP

LLM Agent discovers the Enterprise IdP's OpenID Connect Provider configuration based on a configured issuer that was previously establshed.

> Note: IdP discovery where an agent discovers which IdP the agent should use to authenticate a given user is out of scope of this specification.

```
GET /.well-known/openid-configuration
Host: cyberdyne.idp.example
Accept: application/json

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

{
  "issuer": "https://cyberdyne.idp.example/",
  "authorization_endpoint": "https://cyberdyne.idp.example/oauth2/authorize",
  "token_endpoint": "https://cyberdyne.idp.example/oauth2/token",
  "userinfo_endpoint": "https://cyberdyne.idp.example/oauth2/userinfo",
  "jwks_uri": "https://cyberdyne.idp.example/oauth2/keys",
  "registration_endpoint": "https://cyberdyne.idp.example/oauth2/register",
  "scopes_supported": [
    "openid", "email", "profile"
  ],
  "response_types_supported": [
    "code"
  ],
  "grant_types_supported": [
    "authorization_code", "refresh_token", "urn:ietf:params:oauth:grant-type:token-exchan
ge"
  ],
  "identity_chaining_requested_token_types_supported": ["urn:ietf:params:oauth:token-type
:id-jag"],
  ...
}
```

LLM Agent discovers all necessary endpoints for authentication as well as support for the Identity Chaining requested token type urn:ietf:params:oauth:token-type:id-jag

A.3.3. IdP Authorization Request (with PKCE)

LLM Agent generates a PKCE code_verifier and a code_challenge (usually a SHA256 hash of the verifier, base64url-encoded) and redirects the end-user to the Enterprise IdP with an authorization request

```
GET /authorize?
  response_type=code
  &client_id=com.example.ai-agent
  &redirect_uri=https://ai-agent.example.com/oauth2/callback
  &scope=openid+profile+email
  &state=xyzABC123
  &code_challenge=E9Melhoa2OwvFrEMTJguCHaoeK1t8URWbuGJSstw-cM
  &code_challenge_method=S256
Host: cyberdyne.idp.example
```

A.3.4. User authenticates and authorizes LLM Agent

Enterprise IdP authenticates the end-user and redirects back to the LLM Agent's registered client redirect URI with an authorization code:

```
https://ai-agent.example.com/oauth2/callback?code=SplxlOBeZQQYbYS6WxSbIA&state=xyzABC123
```

LLM Agent exchanges the code and PKCE code_verifier to obtain an ID Token and Access Token for the IdP's UserInfo endpoint

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

grant_type=authorization_code
&code=SplxlOBeZQQYbYS6WxSbIA
&redirect_uri=https://ai-agent.example.com/oauth2/callback
&client_id=com.example.ai-agent
&code_verifier=dBjftJeZ4CVP-mB92K27uhbUJU1p1r_wW1gFWFOEjXk

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

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

LLM Agent validates the ID Token using the published JWKS for the IdP

```
{
  "iss": "https://cyberdyne.idp.example/",
  "sub": "1997e829-2029-41d4-a716-446655440000",
  "aud": "com.example.ai-agent",
  "exp": 1984444800,
  "iat": 1684441200,
  "auth_time": 1684440000,
  "name": "John Connor",
  "email": "john.connor@cyberdyne.example",
  "email_verified": true
}
```

LLM Agent now has an identity binding for context

A.3.5. LLM Agent calls Enterprise External Tool

LLM Agent tool calls an external tool provided by an Enterprise SaaS Application (Resource Server) without a valid access token and is issued an authentication challenge per Protected Resource Metadata [RFC9728].

> Note: How agents discover available tools is out of scope of this specification

```
GET /tools
Host: saas.example.net
Accept: application/json

HTTP/1.1 401 Unauthorized
WWW-Authenticate: Bearer resource_metadata=
  "https://saas.example.net/.well-known/oauth-protected-resource"
```

LLM Agent fetches the external tool resource’s OAuth 2.0 Protected Resource Metadata per [RFC9728] to dynamically discover an authorization server that can issue an access token for the resource.

```
   GET /.well-known/oauth-protected-resource
   Host: saas.example.net
   Accept: application/json

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

   {
      "resource":
        "https://saas.example.net/",
      "authorization_servers":
        [ "https://authorization-server.saas.com/" ],
      "bearer_methods_supported":
        ["header", "body"],
      "scopes_supported":
        ["agent.tools.read", "agent.tools.write"],
      "resource_documentation":
        "https://saas.example.net/tools/resource_documentation.html"
    }
```

LLM Agent discovers the Authorization Server configuration per [RFC8414]

```
GET /.well-known/oauth-authorization-server
Host: authorization-server.saas.com
Accept: application/json

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

{
  "issuer": "https://authorization-server.saas.com/",
  "authorization_endpoint": "https://authorization-server.saas.com/oauth2/authorize",
  "token_endpoint": "https://authorization-server.saas.com/oauth2/token",
  "jwks_uri": "https://authorization-server.saas.com/oauth2/keys",
  "registration_endpoint": "authorization-server.saas.com/oauth2/register",
  "scopes_supported": [
    "agent.read", "agent.write"
  ],
  "response_types_supported": [
    "code"
  ],
  "grant_types_supported": [
    "authorization_code", "refresh_token", "urn:ietf:params:oauth:grant-type:jwt-bearer"
  ],
  ...
}
```

LLM Agent has learned all necessary endpoints and supported capabilites to obtain an access token for the external tool.

If the urn:ietf:params:oauth:grant-type:jwt-bearer grant type is supported the LLM can first attempt to silently obtain an access token using an Identity Assertion Authorization Grant from the Enterprise's IdP otherwise it can fallback to interactively obtaining a standard authorization_code from the SaaS Application's Authorization Server

> Note: This would benefit from an Authorization Server Metadata [RFC8414] property to indicate whether the Identity Assertion Authorization Grant form of jwt-bearer would be accepted by this authorization server. There are other uses of jwt-bearer that may be supported by the authorization server as well, and is not necessarily a reliable indication that the Identity Assertion Authorization Grant would be supported. See issue #16 (https://github.com/aaronpk/draft-parecki-oauth-identity-assertion-authz-grant/issues/16).

A.3.6. LLM Agent obtains an Identity Assertion Grant for Enterprise External Tool from the Enterprise IdP

LLM Agent makes an Identity Assertion Grant Token Exchange [RFC8693] request for the external tool's resource from the user's Enterprise IdP using the ID Token the LLM Agent obtained when establishing an identity binding context along with scopes and the resource identifier for the external tool that was returned in the tool's OAuth 2.0 Protected Resource Metadata

```
POST /oauth2/token HTTP/1.1
Host: cyberdyne.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
&audience=https://authorization-server.saas.com/
&resource=https://saas.example.net/
&scope=agent.read+agent.write
&subject_token=eyJraWQiOiJzMTZ0cVNtODhwREo4VGZCXzdrSEtQ...
&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...
```

If access is granted, the Enterprise 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": "agent.read agent.write",
  "expires_in": 300
}
```

Identity Assertion Authorization Grant JWT claims:

```
{
  "alg": "ES256",
  "typ": "oauth-id-jag+jwt"
}
.
{
  "jti": "9e43f81b64a33f20116179",
  "iss": "https://cyberdyne.idp.example",
  "sub": "1llb-b4c0-0000-8000-t800b4ck0000",
  "aud": "https://authorization-server.saas.com",
  "resource": "https://saas.example.net/",
  "client_id": "com.example.ai-agent",
  "exp": 1984445160,
  "iat": 1984445100,
  "scope": "agent.read agent.write"
}
.
signature
```

A.3.7. LLM Agent obtains an Access Token for Enterprise External Tool

LLM Agent makes a token request to the previously discovered external tool's Authorization Server token endpoint using the Identity Assertion Authorization Grant obtained from the Enterprise IdP as a JWT Assertion as defined by [RFC7523].

The LLM Agent authenticates with its client credentials that were registered with the SaaS Authorization Server

> Note: How the LLM Agent registers with the Authorization Server (e.g static or dynamic client registration), and whether or not it has credentials, is out-of-scope of this specification

```
POST /oauth2/token HTTP/1.1
Host: authorization-server.saas.com
Authorization: Basic yZS1yYW5kb20tc2VjcmV0v3JOkF0XG5Qx2

grant_type=urn:ietf:params:oauth:grant-type:jwt-bearer
assertion=eyJhbGciOiJIUzI1NiIsI...
```

SaaS Authorization Server validates the Identity Assertion Authorization Grant using the published JWKS for the trusted Enterprise IdP

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
  "token_type": "Bearer",
  "access_token": "2YotnFZFEjr1zCsicMWpAA",
  "expires_in": 86400,
  "scope": "agent.read agent.write"
}
```

A.3.8. LLM Agent makes an authorized External Tool request

LLM Agent tool calls an external tool provided by the Enterprise SaaS Application (Resource Server) with a valid access token

```
GET /tools
Host: saas.example.net
Authorization: Bearer 2YotnFZFEjr1zCsicMWpAA"
Accept: application/json

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

{
  ...
}
```

Acknowledgments

The authors would like to thank the following people for their contributions and reviews of this specification: Kamron Batmanghelich, Sofia Desenberg, Meghna Dubey, George Fletcher, Pieter Kasselman, Kai Lehmann, Dean H. Saxe, Filip Skokan.

Document History

[[To be removed from the final specification]]

-05

* Use audience instead of resource to reference the authorization server issuer

* Add optional resource to indicate the resource server identifier

* Added a section on how to advertise support in the IdP metadata

-04

* Improved section references to other specs

* Editorial clarifications

* Updated references to RFC9728

* Rewrote intro

* Added Brian Campbell as co-author

* Changed "SHOULD NOT" to "MUST NOT" issue access tokens in response to an ID-JAG it issued itself

-03

* Added example for AI Agent

-02

* Changed the aud property to the Issuer URL instead of the token endpoint

-01

* Corrected the scope property in the JWT to match token exchange and JWT access token profile

* Formatting and editorial fixes

-00

* Initial revision

Authors' Addresses

Aaron Parecki
Okta
Email: aaron@parecki.com

Karl McGuinness
Independent
Email: public@karlmcguinness.com

Brian Campbell
Ping Identity
Email: bcampbell@pingidentity.com