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

A. Parecki
Okta
K. McGuinness
Independent
B. Campbell
Ping Identity
2 March 2026

Identity Assertion JWT Authorization Grant
draft-ietf-oauth-identity-assertion-authz-grant-02

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.oauth.net/oauth-identity-assertion-authz-grant/draft-ietf-oauth-identity-assertion-authz-grant.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-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/oauth-wg/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/.

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 an Identity Assertion 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. The application has a direct relationship with the IdP Authorization Server for single sign-on as a Relying Party and another independent OAuth 2.0 client relationship with the Resource Authorization Server in trust domain B.

IdP Authorization Server (IdP) An OpenID Connect Provider (OP) [OpenID.Core] or SAML 2.0 Identity Provider that issues Identity Assertions for single sign-on and cross-domain authorization grants Section 3 for a set of trusted applications in an organization's application ecosystem. In [I-D.ietf-oauth-identity-chaining], this is the Authorization Server in trust domain A, which is also trusted by the Resource Authorization Server in trust domain B.

Resource Authorization Server (AS) Issues OAuth 2.0 access tokens for protected resources provided by the Resource Server. In [I-D.ietf-oauth-identity-chaining], this is the Authorization Server in trust domain B, and trusts cross-domain authorization grants Section 3 from the IdP Authorization Server.

Resource Server (RS) Hosts protected resources and validates access tokens issued by the Resource Authorization Server. In [I-D.ietf-oauth-identity-chaining], this is the Protected Resource in trust domain B. The Resource Server has no direct trust relationship with the IdP Authorization Server. Instead, it validates access tokens issued by its trusted Resource Authorization Server to determine who should have access to resources.

3. Identity Assertion JWT Authorization Grant

The Identity Assertion JWT Authorization Grant (ID-JAG) is a profile of the JWT Authorization Grant [RFC7523] that grants a client delegated access to a resource in another trust domain on behalf of a user without a direct user-approval step at the authorization server. In addition to traditional OAuth scope-based authorization, this specification supports Rich Authorization Requests (RAR) [RFC9396], allowing clients to request limited authorization using structured authorization details.

An ID-JAG is issued and signed by an IdP Authorization Server similar to an ID Token [OpenID.Core], and contains claims about an End-User. Instead of being issued for a client (Relying Party in [OpenID.Core]) as the intended audience for the assertion, it is instead issued with an audience of an Authorization Server in another trust domain (Resource Authorization Server). It replaces the need for the client to obtain an authorization code from the Resource Authorization Server to delegate access to the client, and instead uses the IdP Authorization Server which is trusted by the Resource Authorization Server to delegate access to the client.

As described in [OpenID.Core], ID Tokens are only intended to be processed by the Relying Party (indicated by the ID Token audience) or the Issuer (e.g. for revocation), and not by other actors in a different trust domain such as an Authorization Server.

3.1. ID-JAG Claims

The following claims are used within the Identity Assertion JWT Authorization Grant:

iss: REQUIRED - The issuer identifier of the IdP Authorization Server as defined in [RFC8414].

sub: REQUIRED - Subject Identifier. An identifier within the IdP Authorization Server for the End-User, which is intended to be consumed by the Client as defined in [OpenID.Core]. The identifier MUST be the same as the subject identifier used in an

Identity Assertion for the Resource Authorization Server as a Relying Party for SSO. A public subject identifier MUST be unique when scoped with issuer (iss+sub) for a single-tenant issuer and MUST be unique when scoped with issuer and tenant (iss+tenant+sub) for multi-tenant issuer. See Section 5 for additional considerations.

aud: REQUIRED - The issuer identifier of the Resource Authorization Server as defined in [RFC8414].

client_id: REQUIRED - The client identifier of the OAuth 2.0 [RFC6749] client at the Resource Authorization Server that will act on behalf of the resource owner (sub). This identifier MAY be different that client identifier of the OAuth 2.0 client requesting an ID-JAG from the IdP Section 4.3 of [RFC8693] as it represents and independent client relationship to another Authorization Server in a different trust domain. See Section 5 for additional considerations.

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].

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

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].

authorization_details: OPTIONAL - A JSON array of authorization detail objects as defined in Section 2 of [RFC9396]. This claim enables Rich Authorization Requests (RAR) support, allowing structured authorization requests beyond simple scope strings.

tenant: OPTIONAL - JSON string that represents the tenant identifier for a multi-tenant issuer as defined in [OpenID.Enterprise]

auth_time: OPTIONAL - Time when End-User authenticated as defined in [OpenID.Core].

acr: OPTIONAL - Authentication Context Class Reference that was satisfied when authenticating the End-User as defined in [OpenID.Core].

amr: OPTIONAL - Identifiers for authentication methods used when authenticating the End-User as defined in [OpenID.Core].

aud_tenant: OPTIONAL - A JSON string that represents a Resource Authorization Server tenant identifier. This claim is only included when the Resource Authorization Server is multi-tenant and the IdP knows the tenant identifier. When aud_tenant is present, the aud_sub claim represents the identifier the Resource Authorization Server has for the account within the context of that specific Resource Authorization Server tenant. The combination of aud + aud_tenant and aud_sub MUST be unique within the Resource Authorization Server.

aud_sub: OPTIONAL - The Resource Authorization Server’s identifier for the End-User as defined in [OpenID.Enterprise].

email: OPTIONAL - End-User’s e-mail address as defined in Section 5.1 of [OpenID.Core].

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 as described in Section 3.11 of [RFC8725].

A non-normative example JWT 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,
  "resource": "https://acme.chat.example/api",
  "scope": "chat.read chat.history",
  "auth_time": 1311280970,
  "amr": [
    "mfa",
    "phrh",
    "hwk",
    "user"
  ]
}
.
signature
```

The ID-JAG may contain additional authentication, identity, or authorization claims that are valid for an ID Token [OpenID.Core] as the grant functions as both an Identity Assertion and authorization delegation for the Resource Authorization Server.

It is RECOMMENDED that the ID-JAG contain an email [OpenID.Core] and/or aud_sub [OpenID.Enterprise] claim. The Resource Authorization Server MAY use these claims for account resolution or just-in-time (JIT) account creation, for example when the user has not yet SSO'd into the Resource Authorization Server. Additional Resource Authorization Server specific identity claims MAY be needed for account resolution or JIT account creation.

4. Cross-Domain Access

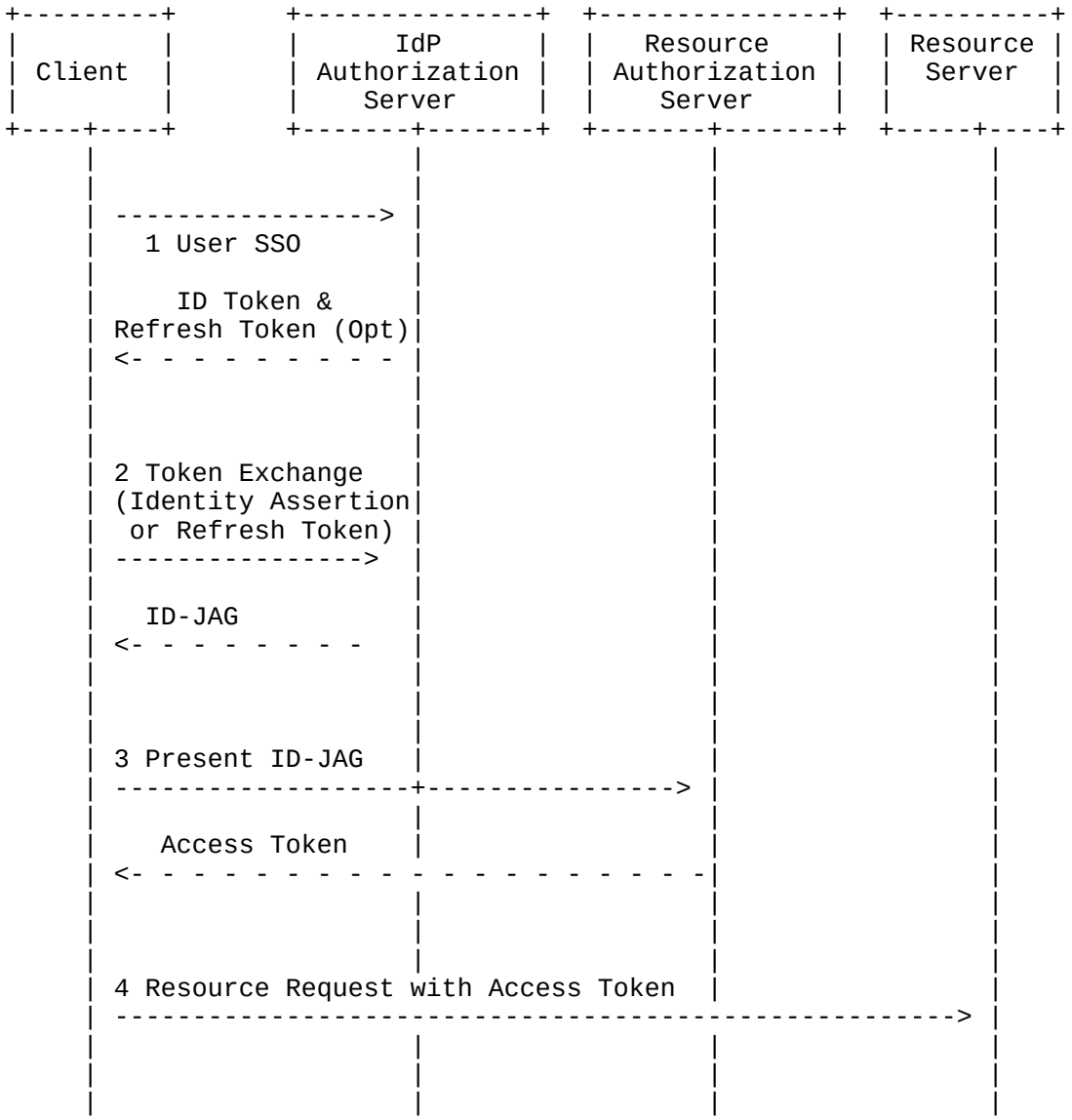
4.1. Overview

The example flow is for an enterprise acme, which uses a multi-tenant wiki app and chat app from different vendors, both of which are integrated into the enterprise's multi-tenant 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 servers
Resource Authorization Server	https://chat.example	https://acme.chat.example	Authorization Server for an chat and communication app
Identity Provider Authorization Server	https://idp.example	https://acme.idp.example	Enterprise Identity Provider
Resource Server	https://api.chat.example	https://api.chat.example	Public API for the chat and communications app

Table 1

Sequence Diagram



1. User authenticates with the IdP Authorization Server, the Client obtains an Identity Assertion (e.g. OpenID Connect ID Token or SAML 2.0 Assertion) for the user and optionally a Refresh Token (when using OpenID Connect) and signs the user in

2. Client uses the Identity Assertion or a previously issued Refresh Token from the IdP to request an Identity Assertion JWT Authorization Grant for the Resource Authorization Server from the IdP Authorization Server

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

4. Client makes an API request to the Resource Server with the Access Token

This specification is constrained to deployments where a set of Resource Authorization Servers for applications used by an organization are trusting the same IdP Authorization Server for Single Sign-On (SSO). The IdP Authorization Server provides a consistent trust boundary and user identity for the set of Resource Authorization Servers to honor the ID-JAG issued by the IdP. The Resource Authorization Server not only delegates user authentication but also delegates user authorization authority to the IdP Authorization Server for the scopes and resource specified in the ID-JAG and does not need obtain user consent directly from the resource owner.

4.2. User Authentication

The Client initiates an authentication request with the IdP Authorization Server 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%20offline_ac
cess&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 and optionally a Refresh Token when offline_access scope is requested per [OpenID.Core].

Note: The IdP Authorization Server 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",
  "refresh_token": "tGzv3JOkF0XG5Qx2TlKWIA"
  "scope": "openid offline_access"
}
```

4.3. Token Exchange

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

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

audience: REQUIRED - The identifier of the Resource Authorization Server in another trust domain as the intended audience for the ID-JAG. IdP Authorization Servers MUST support the issuer identifier of the Resource Authorization Server as defined in Section 2 of [RFC8414]. IdP Authorization Servers MAY support additional IdP-specific unique identifiers for Resource Authorization Servers in other trust domains as an extension point; when such identifiers are used, the IdP maps them to the corresponding Resource Authorization Server audience for the purposes of issuing the ID-JAG.

resource: OPTIONAL - The Resource Identifier of the Resource Server as defined in Section 2 of [RFC8707].

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

authorization_details: OPTIONAL - A JSON string containing a JSON

array of authorization detail objects as defined in Section 2 of [RFC9396]. This parameter enables Rich Authorization Requests (RAR) support, allowing structured authorization requests beyond simple scope strings.

subject_token: REQUIRED - Either the Identity Assertion (e.g. the OpenID Connect ID Token or SAML 2.0 Assertion) for the target resource owner, or a Refresh Token previously issued by the IdP Authorization Server for that resource owner. Implementations of this specification MUST accept Identity Assertions. They MAY additionally accept Refresh Tokens to allow the client to obtain a new ID-JAG without performing a new single sign-on round trip when the Identity Assertion has expired.

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, for a SAML 2.0 Assertion: urn:ietf:params:oauth:token-type:saml2, and for a Refresh Token (when supported): urn:ietf:params:oauth:token-type:refresh_token.

When a Refresh Token is used as the subject token, the client still requests requested_token_type=urn:ietf:params:oauth:token-type:id-jag; this allows the client to refresh an Identity Assertion JWT Authorization Grant without fetching a new Identity Assertion from the user-facing SSO flow.

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 Resource 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.

4.3.1. Example: Token Exchange using ID Token

This example uses an ID Token as the subject_token and a JWT Bearer Assertion [RFC7523] for client authentication (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/
&resource=https://api.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...
```

4.3.1.1. Example: Token Exchange using Refresh Token

This non-normative example shows using a Refresh Token as the subject_token (when supported by the IdP Authorization Server) to obtain an ID-JAG without acquiring a new Identity Assertion:

```
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/
&resource=https://api.chat.example/
&scope=chat.read+chat.history
&subject_token=tGzv3JOkF0XG5Qx2TlKWIA
&subject_token_type=urn:ietf:params:oauth:token-type:refresh_token
&client_assertion_type=urn:ietf:params:oauth:client-assertion-type:jwt-bearer
&client_assertion=eyJhbGciOiJSUzI1NiIsImtpZCI6IjIyIn0...
```

4.3.2. Processing Rules

The IdP MUST validate the subject token:

* If the subject token is an Identity Assertion, the IdP MUST validate the assertion and MUST validate that the audience of the assertion (e.g. the aud claim of the ID Token or SAML Audience) matches the client_id of the client authentication of the request.

* If the subject token is a Refresh Token, the IdP MUST validate it the same way it would for a standard refresh_token grant at the token endpoint: the token is issued by the IdP, bound to the authenticated client, unexpired, not revoked, and the requested scopes and audience remain within the authorization context of the Refresh Token.

* If the subject token is a Refresh Token, the IdP Authorization Server SHOULD retrieve or assemble the subject’s claims needed for the ID-JAG in the same way it would when issuing a new Identity Assertion during a token request, so that the resulting ID-JAG reflects current subject attributes and policy.

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, resources, scopes, and authorization details.

When processing the request:

* If resource is present, the IdP MUST process it according to Section 2 of [RFC8707] and evaluate policy to determine the granted resources. The granted resources MAY be a subset of the requested resources based on policy.

* If scope is present, the IdP MUST process it according to Section 3.3 of [RFC6749] and evaluate policy to determine the granted scopes. The granted scopes MAY be a subset of the requested scopes based on policy.

* If authorization_details is present, the IdP MUST parse it as a JSON array and process each authorization detail object according to [RFC9396]. The IdP evaluates policy for each authorization detail and determines which authorization details to include in the issued ID-JAG. The IdP MAY modify, filter, or omit authorization details based on policy.

* If both resource and authorization_details are present, the IdP MUST process both. The IdP SHOULD ensure consistency between the resource identifiers and authorization details, as they may represent overlapping authorization requests. The IdP MAY derive resource identifiers from authorization details or vice versa, or process them independently based on policy.

* If both scope and authorization_details are present, the IdP MUST process both. The IdP SHOULD ensure consistency between the scopes and authorization details, as they may represent overlapping authorization requests. The IdP MAY derive scopes from authorization details or vice versa, or process them independently based on policy.

* The IdP MUST include the granted resource (if any), scope (if any), and authorization_details (if any) in the issued ID-JAG. If the IdP modifies the requested resources, scopes, or authorization details, it MUST reflect the granted values in the ID-JAG.

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

4.3.3. Response

If access is granted, the IdP creates a signed Identity Assertion JWT Authorization Grant (Section 3) 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 JWT Authorization Grant. (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.

authorization_details: OPTIONAL - A JSON array of authorization detail objects as defined in Section 2.2 of [RFC9396]. This parameter MUST be included if the client requested authorization details and the IdP granted authorization details that differ from what was requested, or if the IdP modified the authorization details.

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.

4.3.3.1. Issued Identity Assertion JWT Authorization Grant

The following is a non-normative example of the issued token

```
{
  "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,
  "resource": "https://api.chat.example/",
  "scope": "chat.read chat.history",
  "auth_time": 1311280970,
  "amr": [
    "mfa",
    "phrh",
    "hwk",
    "user"
  ]
}
.
signature
```

4.3.3.2. Example with Rich Authorization Requests (RAR)

The following is a non-normative example demonstrating the use of Rich Authorization Requests (RAR) [RFC9396] with ID-JAG:

Token Exchange Request with authorization_details:

```
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/
&authorization_details=[{"type":"chat_read","actions":["read"],"locations":["https://api.
chat.example/channels"]},{"type":"chat_history","actions":["read"],"datatypes":["message"
]}]
&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...
```

Token Exchange Response:

```
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",
  "authorization_details": [
    {
      "type": "chat_read",
      "actions": ["read"],
      "locations": ["https://api.chat.example/channels"]
    },
    {
      "type": "chat_history",
      "actions": ["read"],
      "datatypes": ["message"]
    }
  ],
  "expires_in": 300
}
```

Issued Identity Assertion JWT Authorization Grant with authorization_details:

```
{
  "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,
  "authorization_details": [
    {
      "type": "chat_read",
      "actions": ["read"],
      "locations": ["https://api.chat.example/channels"]
    },
    {
      "type": "chat_history",
      "actions": ["read"],
      "datatypes": ["message"]
    }
  ],
  "auth_time": 1311280970
}
.
signature
```

Access Token Request:

```
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...
```

Access Token Response:

```
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,
  "authorization_details": [
    {
      "type": "chat_read",
      "actions": ["read"],
      "locations": ["https://api.chat.example/channels"]
    },
    {
      "type": "chat_history",
      "actions": ["read"],
      "datatypes": ["message"]
    }
  ]
}
```

4.3.3.3. 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"
}
```

4.4. Access Token Request

The Client makes an access token request to the Resource Authorization Server's token endpoint using the previously obtained Identity Assertion JWT Authorization Grant as a JWT Bearer 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 JWT Authorization Grant

obtained in the previous token exchange step

The Client authenticates with its credentials as registered with the Resource 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...
```

4.4.1. 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 Section 3.11 of [RFC8725])

* The Resource Authorization Server MUST validate the aud (audience) claim of the ID-JAG. The aud claim MUST contain the issuer identifier of the Resource Authorization Server as defined in [RFC8414]. The aud claim MAY be a string containing a single issuer identifier, or an array containing a single issuer identifier. If the aud claim is an array, it MUST contain exactly one element, and that element MUST be the issuer identifier of the Resource Authorization Server. If the aud claim does not match the Resource Authorization Server’s issuer identifier, the Resource Authorization Server MUST reject the JWT with an invalid_grant error as defined in Section 5.2 of [RFC6749]. This validation prevents audience injection attacks and ensures the ID-JAG was intended for this specific Resource Authorization Server.

* The client_id claim MUST identify the same client as the client authentication in the request. The Resource Authorization Server MUST validate that the client_id claim in the ID-JAG matches the authenticated client making the request. If they do not match, the Resource Authorization Server MUST reject the request with an invalid_grant error.

When processing authorization information from the ID-JAG:

* If the resource claim is present, the Resource Authorization Server MUST process it according to Section 2 of [RFC8707]. The Resource Authorization Server evaluates the resource identifiers

and determines which resources to grant access to based on policy. The granted resources MAY be a subset of the resources in the ID-JAG issued by the IdP Authorization Server.

* If the scope claim is present, the Resource Authorization Server MUST process it according to Section 3.3 of [RFC6749]. The Resource Authorization Server evaluates the scopes and determines which scopes to grant in the access token based on policy. The granted scopes MAY be a subset of the scopes in the ID-JAG issued by the IdP Authorization Server.

* If the authorization_details claim is present, the Resource Authorization Server MUST parse it as a JSON array and process each authorization detail object according to [RFC9396]. The Resource Authorization Server evaluates policy for each authorization detail and determines which authorization details to grant. The Resource Authorization Server MAY modify, filter, or omit authorization details based on policy.

* If both resource and authorization_details claims are present, the Resource Authorization Server MUST process both. The Resource Authorization Server SHOULD ensure consistency between the resource identifiers and authorization details when issuing the access token. The Resource Authorization Server MAY derive resource identifiers from authorization details or vice versa, or process them independently based on policy.

* If both scope and authorization_details claims are present, the Resource Authorization Server MUST process both. The Resource Authorization Server SHOULD ensure consistency between the scopes and authorization details when issuing the access token. The Resource Authorization Server MAY derive scopes from authorization details or vice versa, or process them independently based on policy.

* The Resource Authorization Server MUST include the granted resource (if any), scope (if any), and authorization_details (if any) in the access token response. The response format follows Section 2 of [RFC8707] for resource, Section 5.1 of [RFC6749] for scope, and Section 2.2 of [RFC9396] for authorization_details.

4.4.2. Response

The Resource Authorization Server’s 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"
}
```

4.4.3. Refresh Token

The Resource Authorization Server SHOULD NOT return a Refresh Token when an Identity Assertion JWT Authorization is exchanged for an Access Token per Section 5.2 of [I-D.ietf-oauth-identity-chaining].

When the access token has expired, clients SHOULD re-submit the original Identity Assertion JWT Authorization Grant to obtain a new Access Token. The ID-JAG replaces the use Refresh Token for the Resource Authorization Server.

If the ID-JAG has expired, the Client SHOULD request a new ID-JAG from the IdP Authorization Server before presenting it to the Resource Authorization Sever using the original Identity Assertion from the IdP (e.g ID Token)

If the ID Token is expired, the Client MAY use the Refresh Token obtained from the IdP during SSO to obtain a new ID Token which it can exchange for a new ID-JAG. If the Client is unable to obtain a new Identity Assertion with a Refresh Token then it SHOULD re-authenticate the user by redirecting to the IdP.

If the IdP Authorization Server supports Refresh Tokens as a subject_token in Token Exchange, the client can skip renewing the Identity Assertion and directly request a new ID-JAG by presenting the Refresh Token (see Section 4.3.1.1).

4.5. SAML 2.0 Identity Assertion Interopability

Clients using SAML 2.0 for SSO with the IdP Authorization Server can obtain an ID-JAG without changing their SSO protocol to OpenID Connect by first exchanging the SAML 2.0 assertion for a Refresh Token using Token Exchange. This enables protocol transition to OAuth and allows the client to later use the Refresh Token as a subject_token to obtain an ID-JAG without prompting the user for a new Identity Assertion.

The OpenID Connect scopes openid offline_access SHOULD be requested (additional scopes are allowed) when requesting a Refresh Token from the IdP Authorization Server.

The IdP Authorization Server MUST map the SAML Audience to a Client ID and ensure the client's authentication matches that mapping before issuing the Refresh Token.

The following non-normative example shows a SAML 2.0 assertion where the Audience value (from AudienceRestriction) corresponds to the Service Provider Entity ID (SPAuthority / SPEntityID) and MUST be mapped to the OAuth Client ID that the IdP Authorization Server associates with that SAML SP registration.

```
<saml2:Assertion xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
    ID="_123456789" IssueInstant="2025-03-01T12:34:56Z" Version="2.0">
  <saml2:Issuer>https://idp.example.com/</saml2:Issuer>
  <saml2:Subject>
    <saml2:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
      alice@example.com
    </saml2:NameID>
    <saml2:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
      <saml2:SubjectConfirmationData
          NotOnOrAfter="2025-03-01T12:39:56Z"
          Recipient="https://client.example.com/assertion-consumer"/>
    </saml2:SubjectConfirmation>
  </saml2:Subject>
  <saml2:Conditions NotBefore="2025-03-01T12:34:56Z" NotOnOrAfter="2025-03-01T13:34:56Z">
    <saml2:AudienceRestriction>
      <saml2:Audience>https://client.example.com/sp-entity-id</saml2:Audience>
    </saml2:AudienceRestriction>
  </saml2:Conditions>
  <saml2:AttributeStatement>
    <saml2:Attribute Name="given_name">
      <saml2:AttributeValue>Alice</saml2:AttributeValue>
    </saml2:Attribute>
  </saml2:AttributeStatement>
  <saml2:AuthnStatement AuthnInstant="2025-03-01T12:30:00Z">
    <saml2:AuthnContext>
      <saml2:AuthnContextClassRef>
        urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
      </saml2:AuthnContextClassRef>
    </saml2:AuthnContext>
  </saml2:AuthnStatement>
</saml2:Assertion>
```

When this assertion is used as the subject_token in Token Exchange, the IdP Authorization Server MUST verify that the Audience / SPEntityID maps to the OAuth Client ID that is authenticated for the token request. This prevents a client from presenting an assertion issued for a different SAML SP.

```
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:refresh_token
&scope=openid+offline_access+email
&subject_token=PHNhbWxwOkFzc2VydGlvbiB4bWxuczp...c2FtbDppc3N1ZXI+PC9zYW1sOkFzc2VydGlvbj4=
&subject_token_type=urn:ietf:params:oauth:token-type:saml2
&client_assertion_type=urn:ietf:params:oauth:client-assertion-type:jwt-bearer
&client_assertion=eyJhbGciOiJSUzI1NiIsImtpZCI6IjIyIn0...

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:refresh_token",
  "access_token": "vF9dft4qmTcXkZ26zL8b6u",
  "token_type": "N_A",
  "scope": "openid offline_access email",
  "expires_in": 1209600
}
```

5. Cross-Domain Client ID Handling

There are three separate OAuth/OpenID Connect/SAML relationships involved in this flow:

* Client to IdP Authorization Server (OpenID Connect or SAML)

* Client to Resource Authorization Server (OAuth)

* Resource Authorization Server to IdP Authorization Server (OpenID Connect or SAML)

Each relationship is typically represented by independent client registrations between each party. For example, the IdP Authorization Server typically issues a Client ID for both the Client and Resource Authorization Server to use for single sign-on with OpenID Connect as a Relying Party. Similarly, the Resource Authorization Server

typically issues a Client ID for the Client to use for API access to the Resource Server. The Client may choose to use different client credentials with each registration.

In this flow, the IdP Authorization Server accepts a Token Exchange request from the Client, and issues an ID-JAG that will be consumed by the Resource Authorization Server. This means the IdP Authorization Server needs to know about the relationship between the Client and the Resource Authorization Server, in order to include a client_id claim in the ID-JAG that will be recognized by the Resource Authorization Server.

This can be handled by the IdP Authorization Server maintaining a record of each client_id used between Clients and Resource Authorization Servers, which will need to be obtained by out-of-band mechanisms. The Client still needs to authenticate using its registered credential with the Resource Authorization Server when presenting the ID-JAG for the mapped client_id. Requiring a confidential client helps to prevent the IdP Authorization Server from delegating access to any of the valid clients for the Resource Authorization Server.

Note: The IdP Authorization Server is also responsible for mapping subject identifiers across Clients and trust domains in the ID-JAG. The same user may have a pair-wise subject identifier issued in an ID Token for SSO to the Client and another with SSO to the Resource Authorization Server as a Relying Party. The Resource Authorization Server needs consistent subject identifiers for account resolution for both SSO and API access. The IdP Authorization Server needs to ensure that the subject identifier issued in the ID-JAG is the same identifier for the user that it would have included in an ID Token intended for the Resource Authorization Server.

Alternatively, if clients use "Client ID Metadata Document" [I-D.ietf-oauth-client-id-metadata-document] as their client identifiers, this acts as a shared global namespace of Client IDs and removes the need for the IdP Authorization Server to maintain a mapping of each client registration.

6. Tenant Relationships with Issuer and Client ID

In multi-tenant deployments, the relationship between tenants, issuers, and client identifiers is critical for proper identity and authorization management. This section explains how these components relate to each other in the context of Identity Assertion JWT Authorization Grants.

6.1. Issuer and Tenant Relationship

An Authorization Server may operate as either a single-tenant or multi-tenant issuer:

* *Single-tenant issuer*: The issuer identifier (iss) uniquely identifies both the Authorization Server and the tenant. All clients and users belong to a single tenant context. The issuer identifier alone is sufficient to identify the tenant.

* *Multi-tenant issuer*: The issuer identifier (iss) identifies the Authorization Server, but multiple tenants may be hosted by the same issuer. In this case, the tenant identifier (tenant) claim is used in conjunction with the issuer identifier to uniquely identify the tenant context. The combination of iss + tenant uniquely identifies the tenant.

When an IdP Authorization Server issues an ID-JAG, it MUST include the tenant claim if the issuer is multi-tenant and the tenant context is relevant for the Resource Authorization Server. The IdP MUST determine the appropriate tenant identifier based on the subject’s tenant membership and the target Resource Authorization Server’s tenant requirements.

6.2. Client ID and Tenant Relationship

The relationship between client_id and tenant depends on the deployment model:

* *Tenant-scoped client identifiers*: In some deployments, the client_id is unique only within a tenant context. The same client_id value may exist in different tenants, and the combination of tenant + client_id (or iss + tenant + client_id for multi-tenant issuers) uniquely identifies the client registration.

* *Global client identifiers*: In other deployments, the client_id is globally unique across all tenants. The client_id alone uniquely identifies the client, regardless of tenant context.

The IdP Authorization Server MUST understand the client identifier model used by the Resource Authorization Server when including the client_id claim in an ID-JAG. For tenant-scoped client identifiers, the IdP MUST ensure that the client_id included in the ID-JAG is valid within the tenant context indicated by the tenant claim (if present) or the issuer’s tenant context.

6.3. Subject Identifier Uniqueness with Tenants

As specified in Section 3, subject identifiers (sub) have different uniqueness requirements based on tenant configuration:

* For single-tenant issuers: The subject identifier MUST be unique when scoped with issuer (iss + sub).

* For multi-tenant issuers: The subject identifier MUST be unique when scoped with issuer and tenant (iss + tenant + sub).

The IdP Authorization Server MUST ensure that the sub claim in the ID-JAG follows the appropriate uniqueness rules for the target Resource Authorization Server. When the Resource Authorization Server is multi-tenant, the IdP MUST include the tenant claim in the ID-JAG to ensure proper subject identifier scoping.

6.4. Tenant Context in Token Exchange

When a Client requests an ID-JAG via Token Exchange, the IdP Authorization Server determines the tenant context from:

1. The subject token (e.g., ID Token or SAML assertion) used in the token exchange request, which may contain tenant information

2. The authenticated client’s tenant membership

3. The target Resource Authorization Server’s tenant requirements

The IdP MUST evaluate policy to determine if the requested audience (Resource Authorization Server) requires tenant information, and if so, which tenant identifier to include in the issued ID-JAG. The tenant identifier in the ID-JAG MUST match the tenant context that the Resource Authorization Server expects for the specified client_id and sub.

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 JWT 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 use the existing authorization code grant and redirect the user to the Resource Authorization Server with an OAuth 2.0 Authorization Request where the user can interactively consent to the access delegation.

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 JWT 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.

8.4. Sender Constraining Tokens

8.4.1. Proof-of-Possession

Identity Assertion JWT Authorization Grant may support key binding to enable sender-constrained tokens as described in Section 4 of [I-D.ietf-oauth-identity-chaining] and [I-D.parecki-oauth-jwt-dpop-grant]. This provides additional security by binding tokens to a specific cryptographic key, preventing reuse by parties that do not have access to the private key.

Proof-of-possession is demonstrated by the client presenting a DPoP proof JWT (as defined in [RFC9449]) in a DPoP HTTP header. The DPoP proof demonstrates that the client possesses the private key corresponding to a public key. This public key can be bound to tokens, ensuring that only the holder of the private key can use those tokens.

The cnf (confirmation) claim, as defined in [RFC7800], is used to bind a public key to a JWT. When an ID-JAG contains a cnf claim with a jkt property as defined in [RFC9449], it indicates that the ID-JAG is bound to that specific key (identified by its JWK SHA-256 Thumbprint), and proof of possession of the corresponding private key MUST be demonstrated when using the ID-JAG.

The following sections describe the processing rules for proof-of-possession at two stages: during the Token Exchange (when requesting an ID-JAG from the IdP) and during the ID-JAG exchange (when exchanging the ID-JAG for an access token at the Resource Authorization Server).

8.4.1.1. Proof-of-Possession During Token Exchange

When a client requests an ID-JAG from the IdP Authorization Server via Token Exchange, the client MAY include a DPoP proof in the request. This demonstrates possession of a key that can be bound to the ID-JAG.

The client generates a key pair and includes a DPoP proof JWT in the DPoP header of the Token Exchange request:

```
POST /oauth2/token HTTP/1.1
Host: acme.idp.example
Content-Type: application/x-www-form-urlencoded
DPoP: eyJ0eXAiOiJkcG9wK2p3dCIsImFsZyI6IkVTMjU2IiwiandrIjp7Imt0eSI6IkVDI...

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/
&resource=https://api.chat.example/
&scope=chat.read+chat.history
&subject_token=eyJraWQiOiJzMTZ0cVNtODhwREo4VGZCXzdrSEtQ...
&subject_token_type=urn:ietf:params:oauth:token-type:id_token
```

The IdP Authorization Server processes the request as follows:

1. If a DPoP proof is present, the IdP MUST validate it according to Section 4.3 of [RFC9449]. The htm claim MUST be POST, and the htu claim MUST match the token endpoint URL.

2. If the DPoP proof is valid, the IdP MUST include a cnf claim in the issued ID-JAG containing a jkt property with the JWK SHA-256 Thumbprint computed from the DPoP proof’s jwk header parameter as defined in Section 6.1 of [RFC9449]. This enables the Resource Authorization Server to validate the key binding for the ID-JAG using simple string comparison of the JWK SHA-256 Thumbprint.

The cnf claim format follows Section 6.1 of [RFC9449]:

```
{
  "jti": "9e43f81b64a33f20116179",
  "iss": "https://acme.idp.example",
  "sub": "U019488227",
  "aud": "https://acme.chat.example/",
  "client_id": "f53f191f9311af35",
  "exp": 1311281970,
  "iat": 1311280970,
  "resource": "https://api.chat.example/",
  "scope": "chat.read chat.history",
  "cnf": {
    "jkt":"0ZcOCORZNYy-DWpqq30jZyJGHTN0d2HglBV3uiguA4I"
  }
}
```

1. The token exchange response does not explicitly indicate whether key binding was successfully performed by the IdP. The token_type response parameter for an ID-JAG is always N_A per Section 2.2.1 of [RFC8693]. The client SHOULD inspect the ID-JAG to determine if a cnf claim is present and whether it represents

the same key as the DPoP proof. This enables the client to detect if the IdP successfully processed the DPoP proof in the token exchange request and bound the issued ID-JAG, preventing the IdP from silently ignoring the DPoP proof and mitigating downgrade attacks.

2. If no DPoP proof is presented, the IdP issues an ID-JAG without a cnf claim.

8.4.1.2. Proof-of-Possession During ID-JAG Exchange

When a client exchanges an ID-JAG for an access token at the Resource Authorization Server, the processing rules depend on whether the ID-JAG contains a cnf claim and whether the client presents a DPoP proof.

8.4.1.2.1. ID-JAG Contains cnf Claim and DPoP Proof is Presented

If the ID-JAG contains a cnf claim and the client presents a DPoP proof, the Resource Authorization Server MUST:

1. Validate the DPoP proof according to Section 4 of [RFC9449].

2. Extract the JWK SHA-256 Thumbprint from the DPoP proof by computing the thumbprint of the jwk header parameter in the DPoP proof according to [RFC7638].

3. Extract the JWK SHA-256 Thumbprint from the jkt property of the cnf claim in the ID-JAG.

4. Compare the two thumbprints. They MUST match exactly. If they do not match, the request MUST fail with an invalid_grant error.

5. If the thumbprints match, the Resource Authorization Server MAY issue a sender-constrained access token (e.g., a DPoP-bound token) per the Resource Server configuration. The issued access token SHOULD be bound to the same key.

Example request:

```
POST /oauth2/token HTTP/1.1
Host: acme.chat.example
Content-Type: application/x-www-form-urlencoded
DPoP: eyJ0eXAiOiJkcG9wK2p3dCIsImFsZyI6IkVTMjU2IiwiandrIjp7Imt0eSI6IkVDI...

grant_type=urn:ietf:params:oauth:grant-type:jwt-dpop
&assertion=eyJhbGciOiJIUzI1NiIsInR5cCI6Im9hdXRoLWlkLWphZytqd3QifQ...
```

Example successful response with DPoP-bound token:

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

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

8.4.1.2.2. ID-JAG Contains cnf Claim but DPoP Proof is Not Presented

If the ID-JAG contains a cnf claim but the client does not present a DPoP proof, the Resource Authorization Server MUST reject the request with an invalid_grant error, as the ID-JAG requires proof of possession.

Example error response:

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

{
  "error": "invalid_grant",
  "error_description": "Proof of possession required for this authorization grant"
}
```

8.4.1.2.3. ID-JAG Does Not Contain cnf Claim and DPoP Proof is Presented

If the ID-JAG does not contain a cnf claim but the client presents a DPoP proof, the Resource Authorization Server:

1. MUST validate the DPoP proof according to Section 4 of [RFC9449].

2. MAY issue a sender-constrained access token (e.g., a DPoP-bound token) per the Resource Server configuration at the Authorization Server, binding the access token to the key demonstrated in the DPoP proof.

3. The access token response will indicate the token type (e.g., DPoP for DPoP-bound tokens, or Bearer for unconstrained tokens).

8.4.1.2.4. ID-JAG Does Not Contain cnf Claim and DPoP Proof is Not Presented

If the ID-JAG does not contain a cnf claim and the client does not present a DPoP proof:

1. The Resource Authorization Server MAY issue an unconstrained Bearer token.

2. However, if the Resource Server configuration at the Authorization Server requires constrained tokens for that Resource Server, the request MUST fail with an invalid_grant error.

Example error response when constrained tokens are required:

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

{
  "error": "invalid_grant",
  "error_description": "Sender-constrained tokens required for this resource server"
}
```

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.media-types] in the manner described in [RFC6838]. It can be used to indicate that the content is an Identity Assertion JWT Authorization Grant.

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 an Identity Assertion JWT Authorization Grant

* Change Controller: IETF

* Specification Document: This document

9.3. JSON Web Token Claims Registration

This section registers the following claims 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: IETF
* Specification Document(s): Section 3
* Claim Name: aud_tenant
* Claim Description: Resource Authorization Server tenant identifier
* Change Controller: IETF
* Specification Document(s): Section 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-08, 9 February 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-oauth-identity-chaining-08>.

[I-D.ietf-oauth-rfc7523bis]
Jones, M. B., Campbell, B., Mortimore, C., and F. Skokan, "Updates to OAuth 2.0 JSON Web Token (JWT) Client Authentication and Assertion-Based Authorization Grants", Work in Progress, Internet-Draft, draft-ietf-oauth-rfc7523bis-05, 12 January 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-oauth-rfc7523bis-05>.

[I-D.parecki-oauth-jwt-dpop-grant]
Parecki, A., "OAuth 2.0 JWT Authorization Grant with DPoP Binding", Work in Progress, Internet-Draft, draft-parecki-oauth-jwt-dpop-grant-01, 30 January 2026, <https://datatracker.ietf.org/doc/html/draft-parecki-oauth-jwt-dpop-grant-01>.

[IANA.jwt] IANA, "JSON Web Token (JWT)", <https://www.iana.org/assignments/jwt>.

[IANA.media-types]
IANA, "Media Types", <https://www.iana.org/assignments/media-types>.

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

[OpenID.Core]
Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., and C. Mortimore, "OpenID Connect Core 1.0 incorporating errata set 2", December 2023, <https://openid.net/specs/openid-connect-core-1_0.html>.

[OpenID.Enterprise]
Hardt, D. and K. McGuinness, "OpenID Connect Enterprise Extensions 1.0 - draft 01", September 2025, <https://openid.net/specs/openid-connect-enterprise-extensions-1_0.html>.

[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>.

[RFC7638] Jones, M. and N. Sakimura, "JSON Web Key (JWK) Thumbprint", RFC 7638, DOI 10.17487/RFC7638, September 2015, <https://www.rfc-editor.org/rfc/rfc7638>.

[RFC7800] Jones, M., Bradley, J., and H. Tschofenig, "Proof-of-Possession Key Semantics for JSON Web Tokens (JWTs)", RFC 7800, DOI 10.17487/RFC7800, April 2016, <https://www.rfc-editor.org/rfc/rfc7800>.

[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>.

[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>.

[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>.

[RFC9396] Lodderstedt, T., Richer, J., and B. Campbell, "OAuth 2.0 Rich Authorization Requests", RFC 9396, DOI 10.17487/RFC9396, May 2023, <https://www.rfc-editor.org/rfc/rfc9396>.

[RFC9449] Fett, D., Campbell, B., Bradley, J., Lodderstedt, T., Jones, M., and D. Waite, "OAuth 2.0 Demonstrating Proof of Possession (DPoP)", RFC 9449, DOI 10.17487/RFC9449, September 2023, <https://www.rfc-editor.org/rfc/rfc9449>.

10.2. Informative References

[I-D.ietf-oauth-client-id-metadata-document] Parecki, A. and E. Smith, "OAuth Client ID Metadata Document", Work in Progress, Internet-Draft, draft-ietf-oauth-client-id-metadata-document-01, 1 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-oauth-client-id-metadata-document-01>.

[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 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 Authorization Server
* Enterprise has established a trust relationship between their IdP and the Client for SSO and Identity Assertion JWT Authorization Grant
* Enterprise has established a trust relationship between their IdP and the Resource Authorization Server for SSO and Identity Assertion JWT Authorization Grant
* Enterprise has granted the Client permission to act on behalf of users for the Resource Authorization Server 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 Authorization Server

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

* The Resource Authorization Server is configured to allow the Identity Assertion JWT 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 JWT 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. Example Sequence

The steps below describe the sequence of the LLM agent obtaining an access token using an Identity Assertion JWT Authorization Grant (Section 3).

A.3.2.1. 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 established.

> 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.3.2.2. 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.3.2.3. 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.3.2.4. 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 capabilities 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 JWT 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 JWT 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 JWT Authorization Grant would be supported. See issue #16 (https://github.com/aaronpk/draft-parecki-oauth-identity-assertion-authz-grant/issues/16).

A.3.2.5. LLM Agent obtains an Identity Assertion JWT Authorization Grant for Enterprise External Tool from the Enterprise IdP

LLM Agent makes an Identity Assertion JWT Authorization 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 JWT Authorization Grant 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 JWT Authorization Grant 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": "4960880b83dc9",
  "exp": 1984445160,
  "iat": 1984445100,
  "scope": "agent.read agent.write"
}
.
signature
```

A.3.2.6. 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 JWT 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 JWT 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.3.2.7. 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, Bingrong He, Pieter Kasselman, Kai Lehmann, Dean H. Saxe, Filip Skokan, Phil Whipps.

Document History

[[To be removed from the final specification]]

-02

* Added reference and examples of a RAR authorization_details object in the Token Exchange and ID-JAG

* Added refresh token as an optional subject token input to the Token Exchange for SAML interop

* Described how to use DPoP with the ID-JAG exchange

* Added aud_tenant and aud_sub claims to ID-JAG to support multi-tenant systems

-01

* Moved ID-JAG definition to document root instead of nested under Token Exchange

* Added proposed OpenID Connect tenant claim

* Added authentication claims from ID Token

* Adopted standard OAuth 2.0 role names instead of Resource App or Resource App’s Authorization Server

* Updated sequence diagram

* Updated all inconsistent references of ID-JAG to "Identity Assertion JWT Authorization Grant"

* Updated section references with more specific links

* Added reference to scope parameter in ID-JAG processing rules

* Added a section discussing client ID mapping and reference to Client ID Metadata Document

* Added recommendations for refresh tokens

-00

* Initial revision as adopted working group draft

Authors’ Addresses

Aaron Parecki
Okta
Email: aaron@parecki.com

Karl McGuinness
Independent
Email: public@karlmcguinness.com

Brian Campbell
Ping Identity
Email: bcampbell@pingidentity.com