

Web Authorization Protocol
Internet-Draft
Intended status: Standards Track
Expires: 22 November 2026

A. Parecki
Okta
K. McGuinness
Independent
B. Campbell
Ping Identity
21 May 2026

Identity Assertion JWT Authorization Grant
draft-ietf-oauth-identity-assertion-authz-grant-04

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 an identity provider that the downstream Resource Authorization Server already trusts for single sign-on (SSO), using Token Exchange [RFC8693] and JWT Profile for OAuth 2.0 Authorization Grants [RFC7523]. This pattern is informally referred to as Cross-App Access (XAA).

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/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 22 November 2026.

Copyright Notice

Copyright (c) 2026 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	4
2. Conventions and Definitions	5
2.1. Roles	5
2.2. Terms	6
3. Identity Assertion JWT Authorization Grant	7
3.1. ID-JAG Claims	8
3.2. SAML NameID Subject Identifier	11
3.2.1. Subject Identifier Format	11
3.2.2. Subject Identifier Processing	12
4. Cross-Domain Access	13
4.1. Overview	13
4.2. User Authentication	16
4.3. Token Exchange	17
4.3.1. Example: Token Exchange using ID Token	20
4.3.2. Example: Token Exchange using Refresh Token	20
4.3.3. Processing Rules	21
4.3.4. Response	22
4.4. Access Token Request	27
4.4.1. Processing Rules	28
4.4.2. Response	30

4.4.3. Refresh Token	30
4.5. SAML 2.0 Subject Token Interoperability	31
5. Cross-Domain Client ID Handling	33
6. Tenant Relationships with Issuer and Client ID	34
6.1. Issuer and Tenant Relationship	34
6.2. Client ID and Tenant Relationship	35
6.3. Subject Identifier Uniqueness with Tenants	35
6.4. Tenant Context in Token Exchange	36
7. Authorization Server Metadata	36
7.1. IdP Authorization Server Metadata	36
7.2. Resource Authorization Server Metadata	37
8. Client Metadata	37
9. Security Considerations	38
9.1. Client Authentication	38
9.2. Step-Up Authentication	38
9.3. Cross-Domain Use	39
9.4. Metadata Disclosure	39
9.5. Subject Identifier Trust	40
9.6. Subject Identifier Disclosure	40
9.7. Actor Delegation Extensions	40
9.8. Sender Constraining Tokens	41
9.8.1. Proof-of-Possession	41
10. IANA Considerations	46
10.1. Media Types	46
10.2. OAuth URI Registration	46
10.3. OAuth Authorization Server Metadata Registration	47
10.4. OAuth Dynamic Client Registration Metadata Registration	47
10.5. JSON Web Token Claims Registration	47
10.6. Security Event Identifier Format Registration	48
11. References	48
11.1. Normative References	48
11.2. Informative References	51
Appendix A. Use Cases	52
A.1. Enterprise Deployment	52
A.1.1. Preconditions	52
A.2. Customer Identity for Developer SaaS Components	53
A.2.1. Preconditions	53
A.3. Email and Calendaring Applications	54
A.3.1. Preconditions	55
A.4. AI Agent using External Tools	55
A.4.1. Preconditions	55
A.4.2. Example Sequence	56
Acknowledgments	63
Document History	63
Authors' Addresses	65

1. Introduction

In many deployments, applications are configured for single sign-on to a common identity provider (IdP) using OpenID Connect or SAML. This enables users to access multiple applications using a single account at the IdP, and enables the operator of that IdP to manage which users can access which applications and enforce policy for access to those applications. Enterprise workforce identity providers are one common example, but the same trust pattern can also arise in customer identity, platform identity, and other federated application ecosystems.

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 defines Cross-App Access (XAA): a pattern in which an application's access to another application's API is mediated by the IdP that both applications already trust for SSO and subject resolution. XAA extends the IdP's role from single sign-on to brokering cross-domain API access, using the same trust relationships that already exist for SSO.

XAA is built on top of OAuth 2.0 by profiling "Identity Chaining Across Trust Domains" [I-D.ietf-oauth-identity-chaining], which combines OAuth 2.0 Token Exchange [RFC8693] and the JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants [RFC7523] to chain a JWT authorization grant across trust domains. Identity Chaining intentionally leaves many details unspecified in order to support multiple use cases. This specification profiles it for XAA by defining:

- * the Token Exchange request used to obtain an Identity Assertion JWT Authorization Grant (Section 3);
- * the ID-JAG itself, including its claims and format;
- * the processing rules applied by the Resource Authorization Server when exchanging an ID-JAG for an Access Token; and
- * the authorization server and client metadata used to discover support for the profile.

In particular, this profile requires an Identity Assertion (such as an OpenID Connect ID Token or SAML 2.0 assertion) as the Token Exchange `subject_token` input.

Unlike profiles in which the IdP issues access tokens directly to be consumed by the Resource Server (such as the OAuth 2.0 JWT Profile for Access Tokens [RFC9068]), this specification preserves the Resource Authorization Server as the issuer of access tokens for its protected resources. This lets the IdP centralize federated identity, SSO trust, and the decision to broker cross-app access, while the Resource Authorization Server keeps independent control over its own access tokens. The Resource Authorization Server determines whether to honor a given ID-JAG, what scopes or authorization details to grant, and what access token to issue, applying its own local policy.

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 application ecosystem. In this specification, the IdP is the issuer that the Resource Authorization Server already trusts for SSO and subject resolution for the target user accounts. 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.

2.2. Terms

The following terms are used in this document:

Common OAuth and token processing terms such as client, authorization server, resource server, resource owner, access token, refresh token, token, grant, assertion, subject_token, subject_token_type, actor_token, and actor_token_type are used as defined in OAuth 2.0 [RFC6749], JSON Web Token (JWT) [RFC7519], JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants [RFC7523], OAuth 2.0 Token Exchange [RFC8693], and OAuth 2.0 Authorization Server Metadata [RFC8414], unless otherwise specified by this document.

OpenID Connect terms such as end-user, Relying Party, OpenID Provider, ID Token, subject identifier, and pairwise subject identifier are used as defined in OpenID Connect Core 1.0 [OpenID.Core], unless otherwise specified by this document.

Identity Assertion A security token issued by the IdP Authorization Server that conveys claims about the End-User and can be used as the subject_token input to Token Exchange. In this specification, the Identity Assertion is typically an OpenID Connect ID Token or a SAML 2.0 assertion.

Trust Domain A deployment-specific security and administrative boundary within which a set of entities, identifiers, credentials, and policy decisions are mutually trusted according to established trust relationships. In this specification, the IdP Authorization Server operates in trust domain A, while the Resource Authorization Server and Resource Server operate in trust domain B.

Cross-Domain Involving two or more trust domains where an assertion, grant, or authorization decision produced in one trust domain is relied upon in another. In this specification, the IdP Authorization Server operates in trust domain A, while the Resource Authorization Server and Resource Server operate in trust domain B.

Cross-App Access (XAA) The application-ecosystem name for the pattern this specification profiles, in which an application obtains an access token at another application's API by coordinating through the IdP both applications trust for SSO. XAA is realized over OAuth 2.0 by using an Identity Assertion JWT Authorization Grant (Section 3) as a specific instance of cross-domain identity chaining ([I-D.ietf-oauth-identity-chaining]).

Subject Resolution The process by which the Resource Authorization Server determines which local subject represents the End-User identified by the ID-JAG. Subject resolution can use stable identifiers and other trusted claims in the ID-JAG, such as `iss`, `sub`, `tenant`, `aud_sub`, `email`, or deployment-specific claims, and can include JIT provisioning when permitted by policy.

JIT provisioning The process by which the Resource Authorization Server or a relying service creates a new local account or subject record, or updates an existing one, using trusted identity claims presented during the transaction rather than requiring separate pre-provisioning. In this specification, JIT provisioning may occur as part of subject resolution when permitted by policy.

Tenant A deployment-specific administrative, organizational, or customer boundary within an issuer or relying service that scopes subjects, clients, policy, and other identifiers. In single-tenant deployments, the issuer may uniquely identify the tenant context; in multi-tenant deployments, the tenant context may need to be conveyed explicitly, for example by the `tenant` or `aud_tenant` claim.

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 can be extended with 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 that is trusted by the Resource Authorization Server for SSO and subject resolution to delegate access to the client. The Resource Authorization Server still applies local policy when deciding whether to honor the ID-JAG and what access token to issue.

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 sub claim identifies the End-User in the subject namespace of the ID-JAG issuer. When the Resource Authorization Server uses a different subject namespace for SSO, such as a SAML Assertion Subject <NameID> namespace, the ID-JAG MAY include sub_id to carry the SSO subject identifier. When both sub and sub_id are present, they MUST identify the same End-User. 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.

sub_id: OPTIONAL - Subject Identifier as defined in [RFC9493]. The sub_id claim MAY be used to identify the same End-User as the sub claim using a different identifier namespace from the ID-JAG iss and sub. This is useful when the Resource Authorization Server resolves users by a non-JWT subject identifier, such as a SAML Assertion Subject <NameID>, in its SSO trust relationship with the IdP Authorization Server. See Section 3.2.1 for the SAML NameID Subject Identifier Format defined by this specification, and Section 3.2.2 for processing rules.

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 than client identifier of the OAuth 2.0 client requesting an ID-JAG from the IdP Section 4.3 of [RFC8693] as it represents an 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.

act: OPTIONAL - Actor claim as defined in Section 4.1 of [RFC8693]. When present, this claim identifies the actor that is acting on behalf of the subject (sub).

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 subject resolution, including JIT provisioning, 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 subject resolution.

3.2. SAML NameID Subject Identifier

3.2.1. Subject Identifier Format

This specification defines the SAML NameID Subject Identifier Format for use in the sub_id claim [RFC9493]. The SAML NameID Subject Identifier Format is identified by the name saml-nameid and identifies a subject using a SAML 2.0 Assertion Subject <NameID> value within the context of a SAML issuer [OASIS.saml-core-2.0-os].

This format is intended for deployments where the Resource Authorization Server resolves users using SAML SSO subject identifiers rather than the ID-JAG issuer's iss and sub values.

A Subject Identifier in this format MUST contain:

format: A JSON string containing the value saml-nameid identifying this Subject Identifier Format, as defined in [RFC9493].

issuer: A JSON string containing the SAML issuer entity identifier for the assertion or configured mapping containing the SAML Assertion Subject <NameID>, as defined by Section 8.3.6 of [OASIS.saml-core-2.0-os]. SAML <Issuer> attributes are not represented by this format.

nameid: A JSON string containing the SAML Assertion Subject <NameID> value as defined in [OASIS.saml-core-2.0-os].

A Subject Identifier in this format MAY contain:

nameid_format: A JSON string containing the value of the Format attribute on the SAML Assertion Subject <NameID> as defined in [OASIS.saml-core-2.0-os].

name_qualifier: A JSON string containing the value of the

NameQualifier attribute on the SAML Assertion Subject <NameID> as defined in [OASIS.saml-core-2.0-os].

sp_name_qualifier: A JSON string containing the value of the SPNameQualifier attribute on the SAML Assertion Subject <NameID> as defined in [OASIS.saml-core-2.0-os].

sp_provided_id: A JSON string containing the value of the SPProvidedID attribute on the SAML Assertion Subject <NameID> as defined in [OASIS.saml-core-2.0-os].

The following is a non-normative example of a sub_id claim using the SAML NameID Subject Identifier Format:

```
"sub_id": {  
  "format": "saml-nameid",  
  "issuer": "https://idp.example.com/exk33qwjcwDda7luK346",  
  "nameid": "foo@example.com",  
  "nameid_format": "urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress",  
  "sp_name_qualifier": "https://resource.example.com/saml/sp"  
}
```

3.2.2. Subject Identifier Processing

When the Resource Authorization Server resolves users by SAML Assertion Subject <NameID>, the IdP Authorization Server MAY include a sub_id claim using the SAML NameID Subject Identifier Format (Section 3.2.1). The sub_id claim identifies the same subject as the ID-JAG sub claim, but in the SAML subject namespace used by the Resource Authorization Server for SSO.

When constructing a SAML NameID Subject Identifier, the IdP Authorization Server MUST derive its members from the SAML Assertion Subject <NameID> value it would use for SSO to the target Resource Authorization Server, or from an equivalent configured mapping that records the SAML <NameID> value and attributes the IdP Authorization Server would issue for that subject and Resource Authorization Server. Each optional field MUST be included exactly when the corresponding attribute is present on that <NameID> or in the configured mapping. The SAML NameID sub_id MUST identify the same subject as the ID-JAG sub claim.

The Resource Authorization Server MAY use sub_id for subject resolution when it supports the indicated Subject Identifier Format. The Resource Authorization Server MUST compare every member of the SAML NameID Subject Identifier that is part of the set of identifier fields it uses for subject resolution for that SAML issuer. The Resource Authorization Server MUST NOT resolve the subject using only

the nameid value unless local policy explicitly defines nameid alone as the subject identifier for that SAML issuer. In particular, when the SAML Assertion Subject <NameID> is scoped to a Service Provider, the sp_name_qualifier value is part of the subject namespace.

If the Resource Authorization Server requires a SAML NameID Subject Identifier for subject resolution, it MUST reject the token request with an invalid_grant error as defined in Section 5.2 of [RFC6749] if any of the following is true:

- * the ID-JAG does not contain a sub_id claim using the SAML NameID Subject Identifier Format;
- * the sub_id claim is malformed or uses a Subject Identifier Format that the Resource Authorization Server does not support;
- * the sub_id claim is not authorized by local policy for the validated ID-JAG issuer.

See Section 9.5 and Section 9.6 for security considerations on the sub_id claim.

4. Cross-Domain Access

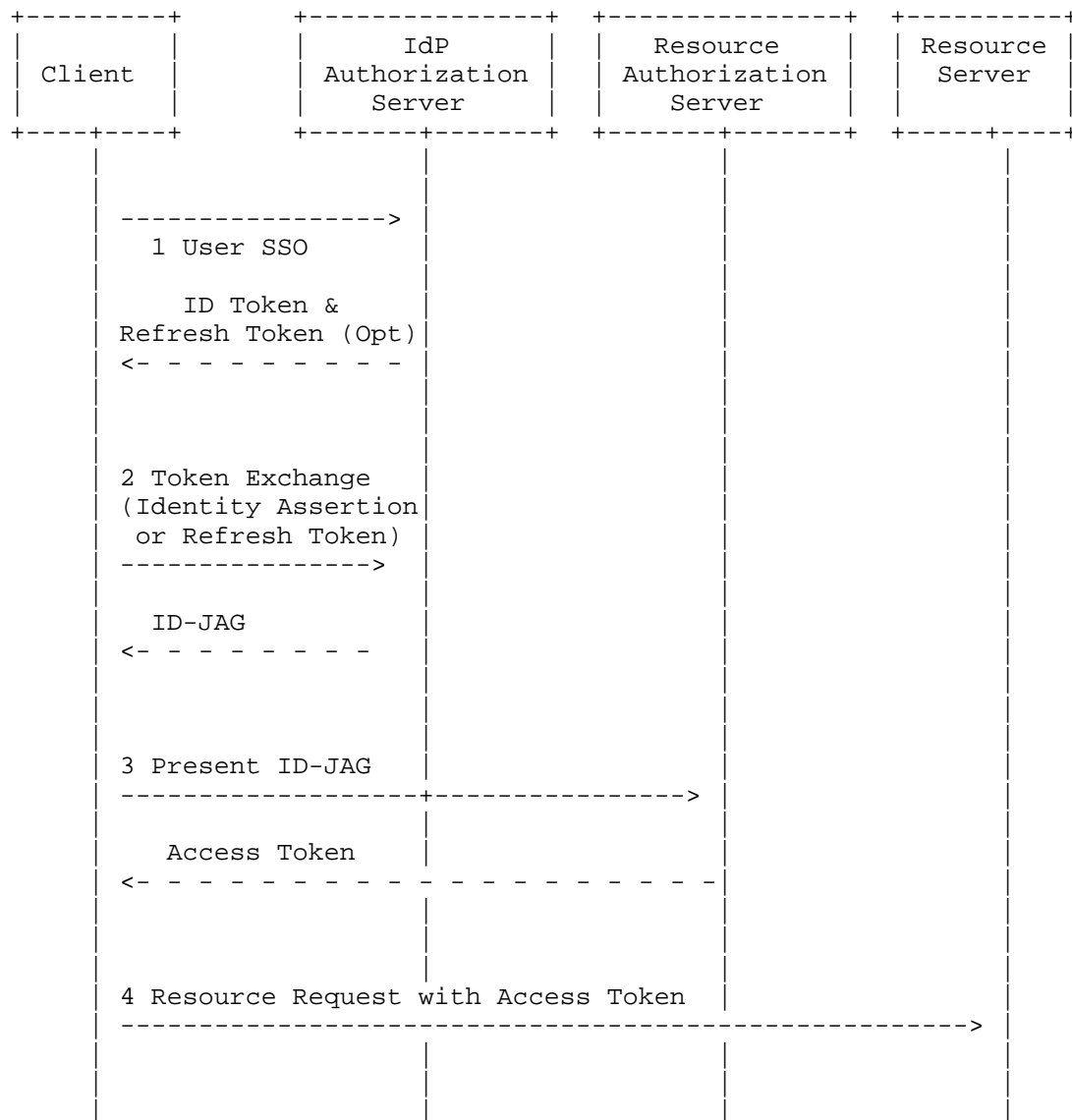
4.1. Overview

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. Enterprise is a common deployment shape for this profile, but it is not the only one. The same pattern applies anywhere the Resource Authorization Server already trusts the issuing IdP for SSO and subject resolution.

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 the Client has a Relying Party relationship with the IdP Authorization Server for SSO, and the Resource Authorization Server independently trusts that same IdP Authorization Server for SSO and subject resolution for the user represented in the ID-JAG. The IdP Authorization Server provides the trusted identity context that allows the Resource Authorization Server to evaluate the ID-JAG. The Resource Authorization Server not only delegates user authentication to that IdP, but also relies on the IdP-issued grant as input to delegated authorization for the scopes, resources, and authorization details conveyed in the ID-JAG. The Resource Authorization Server does not need to obtain user consent directly from the resource owner again at the token exchange step. The Resource Authorization Server still applies local policy to decide whether to honor the grant, whether to narrow or reject the requested access, and what access token to issue. A deployment MAY trust more than one IdP Authorization Server for this purpose, but for each trusted issuer the Resource Authorization Server MUST be configured to recognize that issuer, resolve identities asserted by that issuer to the appropriate local account or principal, and associate the ID-JAG with the correct client relationship.

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_access&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 also support implementation-specific audience values, such as URNs, that identify pre-established trust relationships with Resource Authorization Servers. When such a value is used, the IdP Authorization Server resolves it to the corresponding Resource Authorization Server identifier and issues the ID-JAG with that identifier in the `aud` claim.

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.

actor_token: OPTIONAL - A security token that identifies the actor, as described in Section 2.1 of [RFC8693].

actor_token_type: REQUIRED when **actor_token** is present, as described in Section 2.1 of [RFC8693] - An identifier, as described in Section 3 of [RFC8693], that indicates the type of the security token in the **actor_token** parameter.

This specification does not define normative processing requirements for **actor_token** or whether an act claim is included in the issued ID-JAG. Future profiles or extensions MAY define how **actor_token** is validated, how it influences policy evaluation, and whether it results in an act claim in the issued ID-JAG.

This specification profiles the audience and resource parameters of OAuth 2.0 Token Exchange [RFC8693] for interoperable use with ID-JAG. In this profile, **audience** identifies the Resource Authorization Server to which the ID-JAG is issued, and **resource** identifies the protected resource for which access is requested. This profile uses

audience for the Resource Authorization Server rather than defining a new parameter. The ID-JAG is issued by the IdP Authorization Server for processing by the Resource Authorization Server, and the Resource Authorization Server validates the aud claim to determine whether the ID-JAG was issued for it. The resource parameter continues to identify the protected resource, as defined in [RFC8707]. This convention provides a single interoperable interpretation of audience and resource, including deployments in which one Resource Authorization Server governs multiple protected resources or tenant-specific resources.

For example, if a SaaS provider operates a Resource Authorization Server at `https://authorization-server.saas-tool.example/` and protected resources at `https://api.saas-tool.example/files` and `https://api.saas-tool.example/messages`, a client requesting access to the Files API uses:

```
audience=https://authorization-server.saas-tool.example/  
resource=https://api.saas-tool.example/files
```

If a Resource Authorization Server at `https://login.saas-tool.example/` governs tenant-specific resources such as `https://api.saas-tool.example/acme/` and `https://api.saas-tool.example/fabrikam/`, the audience value remains the Resource Authorization Server identifier and the resource value distinguishes the protected resource. For example:

```
audience=https://login.saas-tool.example/  
resource=https://api.saas-tool.example/acme/
```

and:

```
audience=https://login.saas-tool.example/  
resource=https://api.saas-tool.example/fabrikam/
```

If the IdP Authorization Server supports an implementation-specific audience value such as `urn:example:idp:saas-tool` for that same Resource Authorization Server, the client MAY send:

```
audience=urn:example:idp:saas-tool  
resource=https://api.saas-tool.example/files
```

In that case, the IdP Authorization Server resolves the requested audience value to `https://authorization-server.saas-tool.example/` and issues the ID-JAG with aud set to `https://authorization-server.saas-tool.example/`.

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...
```

4.3.2. 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...
```

4.3.3. 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.

If an actor_token is present, any processing of it is outside the scope of this specification. Future profiles or extensions MAY define validation requirements, policy evaluation rules, and issued token content related to actor_token.

The IdP Authorization Server 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.

4.3.4. 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.

4.3.4.1. 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

```

4.3.4.2. 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 yZSlYyW5kb20tc2VjcmV0v3JOkF0XG5Qx2
Content-Type: application/x-www-form-urlencoded

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": "2YotnFZFEjrlzCsicMWpAA",
  "expires_in": 86400,
  "authorization_details": [
    {
      "type": "chat_read",
      "actions": ["read"],
      "locations": ["https://api.chat.example/channels"]
    },
    {
      "type": "chat_history",
      "actions": ["read"],
      "datatypes": ["message"]
    }
  ]
}
```

4.3.4.3. 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 yZSlYw5kb20tc2VjcmV0v3JOkF0XG5Qx2
Content-Type: application/x-www-form-urlencoded
```

```
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. This client continuity requirement preserves the OAuth client binding across the exchange, but it does not by itself identify or authenticate any actor represented in an `act` claim.

- * If the Resource Authorization Server uses the `sub_id` claim for subject resolution, it MUST process the claim according to the indicated Subject Identifier Format. For SAML NameID subject identifiers, the Resource Authorization Server MUST apply the processing rules in Section 3.2.2.

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": "2YotnFZFEjrlzCsicMWpAA",
  "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 MAY re-submit the original Identity Assertion JWT Authorization Grant to obtain a new Access Token. The ID-JAG replaces the use of 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 Server 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 4.3.2).

4.5. SAML 2.0 Subject Token Interoperability

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...c2FtbDppc3NlZXI+PC9zYWlsOkFzc2VydGlvbj4=
&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 pairwise 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 subject resolution across 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 Metadata

There are two complementary metadata parameters that can be used by the two different authorization servers. `identity_chaining_requested_token_types_supported` indicates which token types an IdP Authorization Server can issue for identity chaining, while `authorization_grant_profiles_supported` indicates which authorization grant profiles a Resource Authorization Server can process.

7.1. IdP Authorization Server Metadata

An IdP Authorization Server can advertise the identity chaining token types it can issue in its OAuth Authorization Server Metadata [RFC8414]. Identity and Authorization Chaining Across Domains [I-D.ietf-oauth-identity-chaining] defines the `identity_chaining_requested_token_types_supported` metadata parameter for this purpose.

To advertise support for issuing an Identity Assertion JWT Authorization Grant via Token Exchange, the IdP Authorization Server SHOULD include the following value in `identity_chaining_requested_token_types_supported`:

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

7.2. Resource Authorization Server Metadata

A Resource Authorization Server can advertise support for authorization grant profiles in its OAuth Authorization Server Metadata [RFC8414] using the `authorization_grant_profiles_supported` metadata parameter.

The value of `authorization_grant_profiles_supported` MUST be a JSON array of strings. Each string MUST identify a supported authorization grant profile.

Inclusion of a profile identifier in `authorization_grant_profiles_supported` indicates only that the Resource Authorization Server implements the processing rules for that profile. It does not indicate that any particular issuer, tenant, client, subject, audience, or authorization request will be accepted.

To advertise support for the Identity Assertion JWT Authorization Grant profile, the Resource Authorization Server SHOULD include the following value in the `authorization_grant_profiles_supported` property:

`urn:ietf:params:oauth:grant-profile:id-jag`

A Resource Authorization Server that includes `urn:ietf:params:oauth:grant-profile:id-jag` in `authorization_grant_profiles_supported` for this specification MUST also include `urn:ietf:params:oauth:grant-type:jwt-bearer` in `grant_types_supported`.

8. Client Metadata

A client can advertise support for authorization grant profiles in its client metadata using the `authorization_grant_profiles_supported` parameter. This parameter MAY be registered via OAuth 2.0 Dynamic Client Registration [RFC7591] or published in a Client ID Metadata Document [I-D.ietf-oauth-client-id-metadata-document].

The value of `authorization_grant_profiles_supported` MUST be a JSON array of strings. Each string MUST identify an authorization grant profile that the client implements.

Inclusion of a profile identifier in `authorization_grant_profiles_supported` indicates only that the client implements the processing rules for that profile. It does not indicate that any particular IdP Authorization Server, Resource Authorization Server, audience, resource, or scope will be requested.

To advertise support for the Identity Assertion JWT Authorization Grant profile, the client SHOULD include the following value in the `authorization_grant_profiles_supported` property:

`urn:ietf:params:oauth:grant-profile:id-jag`

A client that includes `urn:ietf:params:oauth:grant-profile:id-jag` in `authorization_grant_profiles_supported` for this specification MUST also include both `urn:ietf:params:oauth:grant-type:token-exchange` and `urn:ietf:params:oauth:grant-type:jwt-bearer` in its `grant_types`, since the client uses Token Exchange to obtain the ID-JAG from the IdP Authorization Server and the JWT Bearer Token grant to redeem it at the Resource Authorization Server.

9. Security Considerations

9.1. 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.

9.2. 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.

9.3. Cross-Domain Use

This specification is intended for cross-domain uses where the Client, Resource Authorization Server, and IdP are in different trust domains. In particular, the IdP MUST NOT issue access tokens in response to an ID-JAG it issued itself in the same domain. Doing so could lead to unintentional broadening of the scope of authorization. This does not preclude a single piece of software from being both an IdP issuing ID-JAGs as well as a Resource Authorization Server consuming ID-JAGs, as long as the ID-JAG crosses a trust domain such as a tenant or customer boundary.

An ID-JAG is specific to the trust relationship between the issuing IdP Authorization Server and the Resource Authorization Server identified by the aud claim. When a deployment involves additional downstream hops, the same ID-JAG MUST NOT be reused as the authorization grant for a different downstream Resource Authorization Server. For each subsequent hop, a new ID-JAG MAY be issued by the IdP Authorization Server trusted by that downstream Resource Authorization Server for SSO and subject resolution, or other mechanisms MAY be used.

9.4. Metadata Disclosure

Advertising issuer-specific trust relationships in publicly accessible metadata can disclose federation topology, business relationships, tenant configuration, or other deployment-sensitive information.

Resource Authorization Servers MUST NOT use `authorization_grant_profiles_supported` to disclose issuer allow lists or other profile-specific trust relationships.

Resource Authorization Servers MAY provide a protected discovery mechanism by which an authenticated client can determine whether an Identity Assertion JWT Authorization Grant from a particular issuer would be accepted for that client. If such a mechanism is provided, the Resource Authorization Server MUST require client authentication before disclosing issuer-specific acceptance information. The response MUST be specific to the authenticated client and MAY also be scoped by tenant, resource, or other local policy context.

9.5. Subject Identifier Trust

When the ID-JAG includes a `sub_id` claim, the Resource Authorization Server MUST NOT use the `sub_id.issuer` value to establish trust in the ID-JAG issuer. The ID-JAG MUST first be validated using the `iss` claim, signature, audience, expiration, and client binding according to this specification.

The Resource Authorization Server MUST use a SAML NameID `sub_id` only when the validated ID-JAG issuer is explicitly associated with the SAML issuer identified by `sub_id.issuer` through local configuration or trusted federation metadata.

9.6. Subject Identifier Disclosure

The IdP Authorization Server SHOULD include `sub_id` only when needed by the target Resource Authorization Server for subject resolution. Because the ID-JAG is visible to the Client that presents it, `sub_id` can disclose additional user identifiers, such as email-address NameIDs or stable enterprise account identifiers, that the Client might not otherwise receive. The IdP Authorization Server SHOULD minimize the subject identifiers included in the ID-JAG to those required by the Resource Authorization Server.

9.7. Actor Delegation Extensions

This specification allows Token Exchange requests for ID-JAG to carry `actor_token`, but it does not define normative processing requirements for it. Future profiles or extensions can define how `actor_token` is validated, authorized, and reflected in the issued ID-JAG, including whether an `act` claim is included.

Profiles or extensions that define use of actor_token need to consider delegation risks. In particular, a client could attempt to combine a valid subject_token with an unrelated or less-trusted actor_token to obtain an ID-JAG that overstates the actor's authority.

Such profiles or extensions should define how actor_token is validated, how the relationship between the authenticated client, subject, and actor is authorized, how any resulting act claim is derived, and how unnecessary disclosure of actor identity or attributes is minimized across trust domains.

When such profiles or extensions use an act claim, they should preserve the distinction between the actor identified by act and the resource owner identified by sub. The authenticated client identity is also not a substitute for actor identity.

9.8. Sender Constraining Tokens

9.8.1. 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).

9.8.1.1. 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: eyJ0eXAiOiJkcG9wK2p3dCI6ImFsZyI6IkVTMjU2IiwiaWandrIjp7Imt0eSI6IkVDI...

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-DWpqq30jZyJGHTN0d2Hg1BV3uiguA4I"
  }
}
```

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.

9.8.1.2. 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.

9.8.1.2.1. 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: eyJ0eXAiOiJkcG9wK2p3dCI6ImFsZyI6IkVTMjU2IiwiaWandrIjp7Imt0eSI6IkVDI...

grant_type=urn:ietf:params:oauth:grant-type:jwt-dpop
&assertion=eyJhbGciOiJIUzI1NiIsInR5cCI6Imlm9hdXRoLWlkLWphZytqd3QifQ...
```

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": "2YotnFZFEjrlzCsicMWpAA",
  "expires_in": 86400,
  "scope": "chat.read chat.history"
}
```

9.8.1.2.2. 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"  
}
```

9.8.1.2.3. 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).

9.8.1.2.4. 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"
}
```

10. IANA Considerations

10.1. 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.

10.2. 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

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

- * URN: `urn:ietf:params:oauth:grant-profile:id-jag`
- * Common Name: Authorization grant profile identifier for an Identity Assertion JWT Authorization Grant
- * Change Controller: IETF
- * Specification Document: This document

10.3. OAuth Authorization Server Metadata Registration

This section registers `authorization_grant_profiles_supported` in the "OAuth Authorization Server Metadata" registry of the "OAuth Parameters" registry [IANA.oauth-parameters].

- * Metadata Name: `authorization_grant_profiles_supported`
- * Metadata Description: JSON array of supported authorization grant profile identifiers
- * Change Controller: IETF
- * Specification Document: This document

10.4. OAuth Dynamic Client Registration Metadata Registration

This section registers `authorization_grant_profiles_supported` in the "OAuth Dynamic Client Registration Metadata" registry of the "OAuth Parameters" registry [IANA.oauth-parameters], established by [RFC7591].

- * Metadata Name: `authorization_grant_profiles_supported`
- * Metadata Description: JSON array of authorization grant profile identifiers supported by the client
- * Change Controller: IETF
- * Specification Document: Section 8

10.5. 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

10.6. Security Event Identifier Format Registration

This section registers the saml-nameid Subject Identifier Format in the "Security Event Identifier Formats" registry established by [RFC9493].

- * Format Name: saml-nameid
- * Format Description: SAML NameID Subject Identifier Format
- * Change Controller: IETF
- * Reference: Section 3.2.1

11. References

11.1. Normative References

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

Schwenkschuster, A., Kasselmann, P., Burgin, K., Jenkins, M. J., Campbell, B., and A. Parecki, "OAuth Identity and Authorization Chaining Across Domains", Work in Progress, Internet-Draft, draft-ietf-oauth-identity-chaining-12, 11 May 2026, <<https://datatracker.ietf.org/doc/html/draft-ietf-oauth-identity-chaining-12>>.

[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-11, 28 April 2026, <<https://datatracker.ietf.org/doc/html/draft-ietf-oauth-rfc7523bis-11>>.

[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>>.
- [OASIS.saml-core-2.0-os] Cantor, S., Kemp, J., Philpott, R., and E. Maler,
"Assertions and Protocol for the OASIS Security Assertion
Markup Language (SAML) V2.0", OASIS Standard saml-core-
2.0-os, March 2005, <[https://docs.oasis-
open.org/security/saml/v2.0/saml-core-2.0-os.pdf](https://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf)>.
- [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](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>>.
- [RFC7591] Richer, J., Ed., Jones, M., Bradley, J., Machulak, M., and P. Hunt, "OAuth 2.0 Dynamic Client Registration Protocol", RFC 7591, DOI 10.17487/RFC7591, July 2015, <<https://www.rfc-editor.org/rfc/rfc7591>>.
- [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>>.
- [RFC9493] Backman, A., Ed., Scurtescu, M., and P. Jain, "Subject Identifiers for Security Event Tokens", RFC 9493, DOI 10.17487/RFC9493, December 2023, <<https://www.rfc-editor.org/rfc/rfc9493>>.

11.2. 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>>.
- [RFC9068] Bertocci, V., "JSON Web Token (JWT) Profile for OAuth 2.0 Access Tokens", RFC 9068, DOI 10.17487/RFC9068, October 2021, <<https://www.rfc-editor.org/rfc/rfc9068>>.
- [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

The following use cases are illustrative and not exhaustive. Enterprise workforce identity is a common deployment context, but this profile can also apply where the IdP Authorization Server trusted by the Resource Authorization Server for SSO and subject resolution is a CIAM layer, platform identity system, or other application-domain identity provider.

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. Customer Identity for Developer SaaS Components

In deployments where the shared identity layer serves customer-facing applications, the IdP is a Customer Identity and Access Management (CIAM) platform that signs end-users into a customer-facing application or suite of first-party applications rather than a workforce IdP used to sign employees into internal or SaaS productivity tools.

These customer applications often embed or depend on third-party developer SaaS components such as communications, analytics, fraud detection, document processing, support tooling, or logging and observability services. A first-party application may need to request an access token on behalf of the signed-in customer so that these components can be invoked seamlessly as part of the product experience, without interrupting the customer with separate delegated OAuth authorization prompts for each downstream service.

This specification can be used in these deployments when the CIAM platform serves as the common identity layer trusted by the first-party applications and by the authorization servers for the third-party developer SaaS components. In that model, the CIAM platform can broker an Identity Assertion JWT Authorization Grant that allows the component provider to issue an access token scoped to the customer and to the specific component APIs needed by the first-party application.

A.2.1. Preconditions

A.2.1.1. Deployment and Client Preconditions

- * The first-party application has a registered OAuth 2.0 Client with the CIAM IdP Authorization Server
- * The first-party application has a registered OAuth 2.0 Client with the Resource Authorization Server for the developer SaaS component

- * The Resource Authorization Server is able to map the customer identity conveyed by the CIAM platform to the corresponding account, tenant, or subject established through JIT provisioning at the developer SaaS component

A.2.1.2. Trust Relationship Preconditions

- * The organization operating the customer application has established a trust relationship between its CIAM platform and the first-party application for customer sign-in and issuance of identity assertions
- * The organization operating the customer application and the developer SaaS provider have established a trust relationship between the CIAM platform and the Resource Authorization Server for Identity Assertion JWT Authorization Grant
- * The organization operating the customer application has granted the first-party application permission to act on behalf of signed-in customers for the developer SaaS component with a defined set of scopes

A.3. 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.3.1. 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.4. AI Agent using External Tools

AI agents are designed to perform complex tasks on behalf of users, often requiring integration with external tools provided by SaaS applications, internal services, or enterprise data sources. When accessing these systems, the agent often operates on behalf of the end user, and its actions are constrained by the user's identity, role, and permissions. In agentic systems, the relevant IdP for a given ID-JAG hop is the IdP Authorization Server the downstream Resource Authorization Server already trusts for SSO and subject resolution at that boundary. That may be an enterprise workforce IdP, a CIAM or product identity layer, or an internal platform IdP depending on the deployment.

A.4.1. Preconditions

A.4.1.1. Deployment and Client Preconditions

- * The Enterprise IdP at `idp.cyberdyne-corp.example` authenticates the enterprise's users, issues identity assertions, and is the IdP Authorization Server trusted by the External Tool Resource Authorization Server for SSO and subject resolution for this hop
- * The External Tool API (resource server) at `api.saas-tool.example` and its authorization server at `authorization-server.saas-tool.example` are operated by a SaaS tool vendor, in a different trust domain from the enterprise IdP
- * The AI Agent is an OAuth 2.0 client with client ID `https://ai-agent-app.example/`
- * The Enterprise IdP (`idp.cyberdyne-corp.example`) recognizes the AI Agent (`https://ai-agent-app.example/`) as a trusted client, either through static registration or dynamic discovery via `[I-D.ietf-oauth-client-id-metadata-document]`

- * The External Tool Resource Authorization Server (authorization-server.saas-tool.example) recognizes the AI Agent (https://ai-agent-app.example/) as a trusted client, either through static registration or dynamic discovery via [I-D.ietf-oauth-client-id-metadata-document]

Note: This example uses a URL as the client ID, following the Client Identity Metadata Document pattern [I-D.ietf-oauth-client-id-metadata-document] to allow servers to dynamically discover client metadata. Alternatively, clients may be statically registered at the IdP and authorization server and given static opaque client IDs, with metadata configured directly in the IdP and Authorization Server.

A.4.1.2. Trust Relationship Preconditions

- * The enterprise has established a trust relationship between their IdP (idp.cyberdyne-corp.example) and the AI Agent for SSO
- * The enterprise has established a trust relationship between their IdP (idp.cyberdyne-corp.example) and the External Tool Resource Authorization Server (authorization-server.saas-tool.example) for SSO and Identity Assertion JWT Authorization Grant
- * The enterprise has granted the AI Agent permission to act on behalf of users for the External Tool Resource Authorization Server with a specific set of scopes

A.4.2. Example Sequence

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

A.4.2.1. AI Agent establishes a User Identity with Enterprise IdP

AI Agent (https://ai-agent-app.example/) discovers the Enterprise IdP's (idp.cyberdyne-corp.example) 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: idp.cyberdyne-corp.example
Accept: application/json
```

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

```
{
  "issuer": "https://idp.cyberdyne-corp.example/",
  "authorization_endpoint": "https://idp.cyberdyne-corp.example/oauth2/authorize",
  "token_endpoint": "https://idp.cyberdyne-corp.example/oauth2/token",
  "userinfo_endpoint": "https://idp.cyberdyne-corp.example/oauth2/userinfo",
  "jwks_uri": "https://idp.cyberdyne-corp.example/oauth2/keys",
  "registration_endpoint": "https://idp.cyberdyne-corp.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-exchange"
  ],
  "identity_chaining_requested_token_types_supported": ["urn:ietf:params:oauth:token-type:id-jag"],
  ...
}
```

AI 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.4.2.2. IdP Authorization Request (with PKCE)

AI Agent (`https://ai-agent-app.example/`) 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 (`idp.cyberdyne-corp.example`) with an authorization request

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

A.4.2.3. User authenticates and authorizes AI Agent

Enterprise IdP (idp.cyberdyne-corp.example) authenticates the end-user and redirects back to the AI Agent's registered redirect URI with an authorization code:

```
https://ai-agent-app.example/callback?code=Splx10BeZQQYbYS6WxSbIA&state=xyzABC123
```

AI Agent (https://ai-agent-app.example/) 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: idp.cyberdyne-corp.example
Authorization: Basic yZSlyYW5kb20tc2VjcmV0v3JOkF0XG5Qx2
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code
&code=Splx10BeZQQYbYS6WxSbIA
&redirect_uri=https://ai-agent-app.example/callback
&code_verifier=dBjftJeZ4CVP-mB92K27uhbUJU1plr_wW1gFWFOEjXk

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

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

AI Agent validates the ID Token using the published JWKS for the Enterprise IdP (idp.cyberdyne-corp.example)

```
{
  "iss": "https://idp.cyberdyne-corp.example/",
  "sub": "1997e829-2029-41d4-a716-446655440000",
  "aud": "https://ai-agent-app.example/",
  "exp": 19844448400,
  "iat": 1984444800,
  "auth_time": 1984444800,
  "name": "John Connor",
  "email": "john.connor@cyberdyne-corp.example",
  "email_verified": true
}
```

AI Agent now has an identity binding for context

A.4.2.4. AI Agent calls External Tool

AI Agent (<https://ai-agent-app.example/>) calls the External Tool API (Resource Server) at api.saas-tool.example without a valid access token and is issued an authentication challenge per Protected Resource Metadata [RFC9728].

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

```
GET /tools
Host: api.saas-tool.example
Accept: application/json

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

AI Agent fetches the External Tool API'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: api.saas-tool.example
Accept: application/json

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

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

AI Agent discovers the External Tool Resource Authorization Server (authorization-server.saas-tool.example) configuration per [RFC8414]

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

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

```
{
  "issuer": "https://authorization-server.saas-tool.example/",
  "authorization_endpoint": "https://authorization-server.saas-tool.example/oauth2/authorize",
  "token_endpoint": "https://authorization-server.saas-tool.example/oauth2/token",
  "jwks_uri": "https://authorization-server.saas-tool.example/oauth2/keys",
  "registration_endpoint": "https://authorization-server.saas-tool.example/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"
  ],
  "authorization_grant_profiles_supported": ["urn:ietf:params:oauth:grant-profile:id-jag"],
  ...
}
```

AI 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-profile:id-jag` authorization grant profile is supported, the AI Agent can first attempt to silently obtain an access token using an Identity Assertion JWT Authorization Grant from the IdP Authorization Server trusted by the External Tool Resource Authorization Server for SSO, otherwise it can fallback to interactively obtaining a standard `authorization_code` from the External Tool Resource Authorization Server.

A.4.2.5. AI Agent obtains an Identity Assertion JWT Authorization Grant for External Tool from the Enterprise IdP

AI Agent (`https://ai-agent-app.example/`) makes an Identity Assertion JWT Authorization Grant Token Exchange [RFC8693] request to the Enterprise IdP (`idp.cyberdyne-corp.example`) using the ID Token obtained when establishing an identity binding context, along with scopes and the resource identifier for the External Tool API (`api.saas-tool.example`) that was returned in the external tool's OAuth 2.0 Protected Resource Metadata

```
POST /oauth2/token HTTP/1.1
Host: idp.cyberdyne-corp.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-tool.example/
&resource=https://api.saas-tool.example/
&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 (idp.cyberdyne-corp.example) 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://idp.cyberdyne-corp.example/",
  "sub": "1997e829-2029-41d4-a716-446655440000",
  "aud": "https://authorization-server.saas-tool.example/",
  "resource": "https://api.saas-tool.example/",
  "client_id": "https://ai-agent-app.example/",
  "exp": 1984445160,
  "iat": 1984445100,
  "scope": "agent.read agent.write"
}
.
signature

```

A.4.2.6. AI Agent obtains an Access Token for External Tool

AI Agent (<https://ai-agent-app.example/>) makes a token request to the External Tool Resource Authorization Server (authorization-server.saas-tool.example) token endpoint using the Identity Assertion JWT Authorization Grant obtained from the Enterprise IdP (idp.cyberdyne-corp.example) as a JWT Assertion as defined by [RFC7523].

Note: How the AI Agent registers with the External Tool Resource 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-tool.example
Authorization: Basic yZSlyYW5kb20tc2VjcmV0v3JOkF0XG5Qx2
Content-Type: application/x-www-form-urlencoded

```

```

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

```

External Tool Resource Authorization Server (authorization-server.saas-tool.example) validates the Identity Assertion JWT Authorization Grant using the published JWKS for the trusted Enterprise IdP (idp.cyberdyne-corp.example)

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

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

A.4.2.7. AI Agent makes an authorized External Tool request

AI Agent (<https://ai-agent-app.example/>) calls the External Tool API (Resource Server) at api.saas-tool.example with a valid access token

```
GET /tools
Host: api.saas-tool.example
Authorization: Bearer 2YotnFZFEjrlzCsicMWpAA
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 Kasselmann, Kai Lehmann, Dean H. Saxe, Filip Skokan, Phil Whipps.

Document History

[[To be removed from the final specification]]

-04

- * Clarified XAA vs ID-JAG terminology in abstract and intro
- * Added compatibility via new `sub_id` claim for Resource Apps configured with SAML SSO, and tightened definition of sub

- * Added client metadata property for DCR and CIMD

- * Clarified cross-domain use restriction

-03

- * Added a section defining terms used in the document

- * Updated language to be less specific to "enterprise" and more about the relationships between the IdP and Resource Authorization Server

- * Clarified the use of the resource and audience parameters in the token exchange request

- * Removed language discouraging the use of the actor_token in the token exchange request

- * Added a new AS metadata parameter `authorization_grant_profiles_supported` to enable a Resource Authorization Server to publish support for this profile

-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