

oauth  
Internet-Draft  
Intended status: Informational  
Expires: 3 September 2026

R. Li  
H. Wang  
Huawei Int. Pte Ltd  
C. Liu  
Huawei Technologies  
T. Li  
Huawei Int. Pte Ltd  
2 March 2026

OAuth 2.0 Delegated Authorization  
draft-li-oauth-delegated-authorization-01

## Abstract

Delegated authorization enables a client to delegate a subset of its granted privileges to a subordinate access token (also known as a delegated access token). This mechanism allows the client to securely delegate authorization to a delegated party while maintaining fine-grained control over delegated permissions.

## 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://liuchunchi.github.io/li-oauth-delegated-authorization/draft-li-oauth-delegated-authorization.html>. Status information for this document may be found at <https://datatracker.ietf.org/doc/draft-li-oauth-delegated-authorization/>.

Discussion of this document takes place on the WG Working Group mailing list (<mailto:oauth@ietf.org>), which is archived at <https://datatracker.ietf.org/wg/oauth/about/>. Subscribe at <https://www.ietf.org/mailman/listinfo/oauth/>.

Source for this draft and an issue tracker can be found at <https://github.com/liuchunchi/li-oauth-delegated-authorization>.

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

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

This Internet-Draft will expire on 3 September 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 . . . . .	3
2. Requirements Language . . . . .	3
3. Terminology . . . . .	4
4. Overview . . . . .	4
5. Delegated Party Metadata . . . . .	6
5.1. Delegated Party Metadata Attributes . . . . .	6
5.2. Delegated Party Metadata Example . . . . .	7
5.3. WWW-Authenticate . . . . .	8
6. Delegation Tokens and Delegated Access Tokens . . . . .	9
6.1. Delegation Tokens . . . . .	9
6.1.1. Top Level Delegation Token . . . . .	10
6.2. Delegated Access Tokens . . . . .	10
7. Acquiring Delegation Tokens . . . . .	11
7.1. Authorization Code Grant . . . . .	11
7.2. Other Grant Types . . . . .	12
8. Creating Delegated Access Tokens . . . . .	12
9. Using Delegated Access Tokens . . . . .	12
10. Verification of Delegated Access Tokens . . . . .	13
10.1. Local Verification . . . . .	13
10.2. Token Introspection . . . . .	14
11. Privacy Considerations . . . . .	14
11.1. Privacy Benefits . . . . .	14
11.2. Relationship to Token Exchange . . . . .	14
11.3. Trade-offs . . . . .	15
12. Security Considerations . . . . .	15

13. Operational Considerations . . . . .	16
14. IANA Considerations . . . . .	16
14.1. Well-Known URIs Registry . . . . .	16
14.2. OAuth Parameters Registry . . . . .	16
14.3. OAuth Access Token Types Registry . . . . .	17
14.4. HTTP Field Name Registry . . . . .	17
14.5. OAuth Delegated Party Metadata Registry . . . . .	18
14.5.1. Registration Template . . . . .	18
14.5.2. Initial Registry Contents . . . . .	18
15. References . . . . .	19
15.1. Normative References . . . . .	19
15.2. Informative References . . . . .	22
Appendix A. Token Format . . . . .	22
A.1. Example 1 . . . . .	22
A.2. Example 2 . . . . .	23
Appendix B. Integration with Step-Up Authorization . . . . .	25
Appendix C. Use Cases . . . . .	25
C.1. Delegating Subset of Access Rights to Specialized AI Agents . . . . .	25
C.2. Third-Party Analytics Platform Integrated in an Enterprise SaaS . . . . .	26
Authors' Addresses . . . . .	27

## 1. Introduction

OAuth 2.0 [RFC6749] provides a framework for authorizing third-party applications to access protected resources on behalf of a resource owner. However, in existing implementations, access tokens issued to clients often contain excessive permissions that exceed actual requirements, creating security vulnerabilities and potential data exposure risks.

This specification extends OAuth 2.0 with a delegated authorization framework that enables clients to create subordinate access tokens with restricted permissions. This approach addresses the problem of over-privileged access tokens by implementing a two-token architecture that decouples initial authorization from final resource access.

## 2. Requirements Language

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.

### 3. Terminology

This specification uses the following terms defined in OAuth 2.0 [RFC6749]: authorization server, client, resource server, and resource owner.

The following additional terms are used throughout this document:

**\*Delegated Party (DP)\*:** An entity (e.g., a service, component, or application) authorized by the client to access protected resources on behalf of the resource owner.

**\*Delegated Resource\*:** A resource or API endpoint hosted by the delegated party that requires access to the resource owner's protected data at a target resource server.

**\*Delegation Token\*:** A token issued by the authorization server for the client that enables the client to create delegated access tokens.

**\*Delegated Access Token\*:** A token created by the client using the delegation token, with permissions being a subset of the delegation token's privileges and a more limited lifespan.

**\*Delegation Key\*:** A cryptographic key bound to the delegation token, used by the client to sign delegated access tokens. The delegation key is presented in the token request as the `delegation_key` parameter.

### 4. Overview

The delegated authorization framework introduces a hierarchical token structure where a client can obtain a delegation token from an authorization server and use it to issue subordinate access tokens with reduced permissions. This enables fine-grained access control while maintaining the security properties of the original authorization grant.

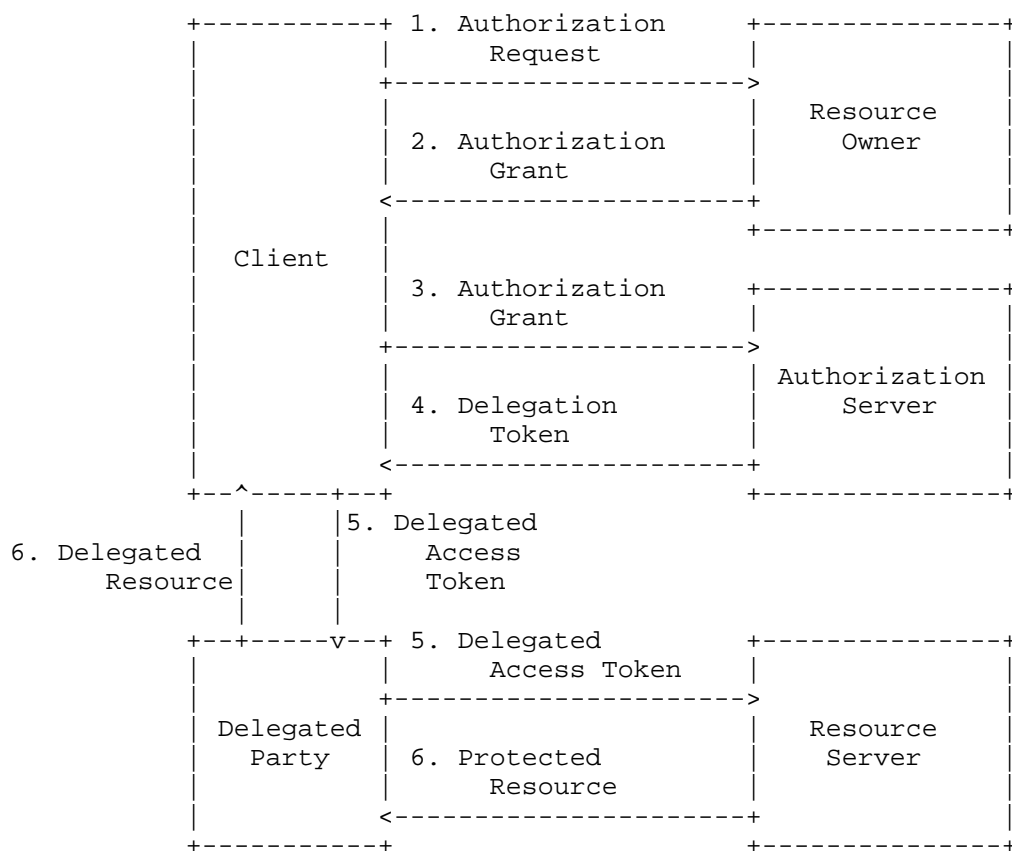


Figure 1: Delegated Authorization Framework Architecture

1. The client requests authorization from the resource owner. The client indicates in the authorization request that the requested authorization grant is for delegated authorization.
2. The client receives an authorization grant.
3. The client requests a delegation token by authenticating with the authorization server and presenting the authorization grant and its delegation key as defined in Section 3.
4. The authorization server authenticates the client and validates the authorization grant, and if valid, issues a delegation token.
5. The client calls the delegated party's API, presenting the delegated access token generated from the delegation token. The delegated access token is issued by the client using the

delegation key. The delegated party requests the target protected resource from the resource server and presents the delegated access token.

6. The resource server validates the delegated access token, and if valid, serves the resource. The delegated party receives the resource, optionally transforms it into a service-specific response (also known as delegated resource), and returns it to the client.

Both delegation token and delegated access token can be JSON Web Tokens (JWTs) [RFC7519] or CBOR Web Tokens (CWTs) [RFC8392].

## 5. Delegated Party Metadata

Before the OAuth 2.0 client retrieves a delegation token and generates a delegated access token for the delegated party, the client needs to obtain the authorization server endpoint and the permissions needed by the delegated party. Such information can be manually configured into the client, or it can be dynamically discovered through delegated party metadata.

Delegated party metadata enables OAuth 2.0 clients to obtain information needed to interact with a delegated party. The structure of the metadata format is similar to "OAuth 2.0 Authorization Server Metadata" [RFC8414] and "OAuth 2.0 Protected Resource Metadata" [RFC9728].

The delegated party metadata is retrieved from a well-known [RFC8615] location as a JSON [RFC8259] document. By default, the well-known URI string used is `/.well-known/oauth-delegated-party`.

### 5.1. Delegated Party Metadata Attributes

**\*resources\***: **\*RECOMMENDED\***. JSON array containing a list of target protected resources' resource identifiers, as defined in [RFC9728]. Either this attribute or **\*authorization\_servers\*** defined below **MUST** be present.

**\*authorization\_servers\***: **\*OPTIONAL\***. JSON array containing a list of OAuth authorization server issuer identifiers, as defined in [RFC8414]. Either this attribute or **\*resources\*** defined above **MUST** be present.

**\*permissions\_supported\***: **\*RECOMMENDED\***. JSON object indicating the permissions the delegated party may request. The **scopes** attribute lists supported scope values [RFC6749]; the **authorization\_details** attribute lists supported rich authorization request objects as

defined in [RFC9396]. These guide the client in constructing valid authorization and token requests. Either the scopes attribute or the authorization\_details attribute must be present.

**\*api\_permissions\***: **\*RECOMMENDED\***. JSON object mapping API endpoints (resource identifiers) to the permissions required to access them. Each value can include scopes and/or authorization\_details to specify the permissions needed for that endpoint/resource.

**\*delegated\_party\_documentation\***: **\*OPTIONAL\***: URL of a page containing human-readable information that developers might want or need to know when using the delegated party. The value of this field MAY be internationalized.

## 5.2. Delegated Party Metadata Example

The following is a non-normative example delegated party metadata:

```
{
  "resources": [
    "https://res1.example.com",
    "https://res2.example.net"
  ],
  "authorization_servers": [
    "https://as1.example.com",
    "https://as2.example.net"
  ],
  "permissions_supported": {
    "scopes": ["profile:read", "profile:write", "email:read", "email:write"],
    "authorization_details": [
      {
        "type": "payment",
        "actions": ["initiate", "status", "cancel"],
        "instructedAmount": {
          "notMoreThan": {
            "currency": "USD",
            "amount": "500.00"
          }
        }
      }
    ]
  },
  "api_permissions": {
    "/emails/list": {
      "scopes": ["email:read"]
    },
    "/balance/transfer": {
      "authorization_details": [
```

```
{
  {
    "type": "payment",
    "actions": ["initiate"],
    "instructedAmount": {
      "notMoreThan": {
        "currency": "USD",
        "amount": "500.00"
      }
    }
  }
},
"/balance/transfer/status": {
  "authorization_details": [
    {
      "type": "payment",
      "actions": ["status"]
    }
  ]
},
"/balance/transfer/cancel": {
  "authorization_details": [
    {
      "type": "payment",
      "actions": ["cancel"]
    }
  ]
}
},
"delegated_party_documentation": "https://dp.example.com/dp_documentation.html"
}
```

### 5.3. WWW-Authenticate

Upon receipt of a request for a delegated resource that lacks credentials, the delegated party can reply with a challenge using the 401 (Unauthorized) status code ([RFC9110] Section 15.5.2) and the WWW-Authenticate header field ([RFC9110] Section 11.6.1).

This specification introduces a new parameter in the WWW-Authenticate HTTP response header field to indicate the delegated party metadata URL:

**\*delegated\_party\_metadata\***: The URL of the delegated party metadata.

The response below is an example of a WWW-Authenticate header that includes the delegated party metadata URL. NOTE: '\' line wrapping per [RFC8792].



```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Bearer delegated_party_metadata=\
  "https://dp.example.com/.well-known/oauth-delegated-party"
```

## 6. Delegation Tokens and Delegated Access Tokens

This section defines the properties of delegation tokens and delegated access tokens. The delegated authorization framework uses a hierarchical token structure where tokens form a chain: a delegation token can be used to create subordinate tokens, which can either be another delegation token (to continue the chain) or a delegated access token (to access protected resources). The top-level token in the chain is issued by the authorization server, while subordinate tokens are created by clients.

### 6.1. Delegation Tokens

A delegation token is a token that can be used to create subordinate tokens. It contains a `max_delegation_depth` claim that limits the maximum length of the delegation chain.

**\*delegation\_key\*:** The cryptographic key bound to the delegation token, used by the client to sign subordinate tokens (either another delegation token or a delegated access token). The value is a JSON object representing the public key, using key format as defined in [RFC7517]. This claim **MUST** be present in delegation tokens. The corresponding private key is held by the client and used to create subordinate tokens. Subordinate tokens **MUST** include the parent delegation token in the `delegation_token` claim, creating a verifiable chain from the top-level token to the subordinate token.

**\*max\_delegation\_depth\*:** OPTIONAL. Integer value indicating the maximum depth of the delegation chain. If not present, the delegation chain is unrestricted. When a client creates a subordinate token from a delegation token:

- \* If the parent delegation token has a `max_delegation_depth` value, the subordinate delegation token **MUST** have a `max_delegation_depth` value smaller than the parent's value.
- \* If the parent delegation token does not have a `max_delegation_depth` value, the subordinate delegation token **MAY** set any `max_delegation_depth` value.

When `max_delegation_depth` reaches 1, the subordinate token **MUST** be a delegated access token (no further delegation is allowed).

The scope / authorization\_details, aud, exp, and nbf claims MAY be present in delegation tokens and can be reduced in subordinate tokens.

**\*scope\* / \*authorization\_details\*:** The scope (as defined in [RFC8693]) or authorization details (as defined in [RFC9396]) of the delegation token. When creating a subordinate token, the client MAY reduce the scope or authorization\_details to a subset of the parent token's permissions.

**\*aud\*:** The audience of the delegation token. When creating a subordinate token, the client MAY narrow the audience to a subset of the parent token's audience.

**\*exp\*:** The expiration time of the delegation token. When creating a subordinate token, the client MUST set an expiration time that is no later than the parent token's expiration time.

**\*nbf\*:** The not-before time of the delegation token. When creating a subordinate token, the client MAY set a not-before time that is later than the parent token's not-before time.

#### 6.1.1. Top Level Delegation Token

The top-level delegation token is issued by the authorization server and MAY contain the iss, sub, and jti claims. If present, these claims apply to the entire token chain. These claims MUST NOT be present in subordinate delegation tokens or delegated access tokens.

**\*iss\*:** The issuer of the delegation token. This claim applies to all subordinate tokens in the chain.

**\*sub\*:** The subject of the delegation token, typically representing the resource owner or the client. This claim applies to all subordinate tokens in the chain.

**\*jti\*:** Unique identifier for the delegation token. This claim applies to all subordinate tokens in the chain.

Subordinate delegation tokens and delegated access tokens MUST NOT include the iss, sub, or jti claims. Verifiers (resource servers and delegated parties) SHALL consider these claims from the top-level delegation token when validating the entire token chain.

#### 6.2. Delegated Access Tokens

A delegated access token is a token used to access protected resources. It cannot be used to create subordinate tokens.

The scope, aud, exp, and nbf claims in delegated access tokens follow the same rules as in delegation tokens, as described in Section 6.1.

**\*max\_delegation\_depth\***: Delegated access tokens do not need to include a max\_delegation\_depth claim, even if the parent delegation token has one. If a delegated access token includes max\_delegation\_depth, its value MUST be 0.

## 7. Acquiring Delegation Tokens

The client requests a delegation token using standard OAuth 2.0 grant types with additional parameters to distinguish delegation requests from standard token requests.

### 7.1. Authorization Code Grant

For authorization code grant type, the client MUST include a delegation=true parameter in the authorization request to indicate that the client is requesting a delegation token instead of an OAuth 2.0 access token.

Additionally, the authorization request MUST include either a scope parameter (as defined in [RFC6749] Section 3.3), an authorization\_details parameter (as defined in "Rich Authorization Requests" [RFC9396]), or both, that define the permissions granted to the requested delegation token.

In the token request, the client MUST include a delegation\_key parameter with the value of the delegation key. The delegation key is a public key for digital signature.

Additionally, the client MAY include in the token request either a scope parameter, an authorization\_details parameter, or both. The client MAY also include a delegation=true parameter in the token request.

In the token response, the authorization server MUST include an access\_token attribute whose value is the delegation token, and MUST include a token\_type attribute valued "Delegation", and MAY include a refresh\_token attribute which is the refresh token for obtaining a new delegation token via the refresh token grant.

Other procedures of the authorization code grant are as described in [RFC6749]. Use of Proof Key for Code Exchange (PKCE) [RFC7636] is RECOMMENDED.

## 7.2. Other Grant Types

Other OAuth 2.0 grant types, such as the refresh token grant or client credentials grant, MAY support delegated authorization by including the delegation and delegation\_key parameters when applicable. The authorization server MUST validate that the client is authorized to request delegation tokens using the given grant type.

## 8. Creating Delegated Access Tokens

The client creates delegated access tokens by:

1. Validating the delegation token's validity and permissions.
2. Generating a subordinate access token with (optionally) reduced privileges.
3. Applying cryptographic protection using the delegation key (digital signature).

The client MUST include the delegation token in the delegation\_token attribute of the delegated access token.

The client MUST ensure that the delegated access token's scope, lifetime, audience, and other claims do not exceed those of the delegation token. The client MAY generate single-use delegated access tokens that the resource server or authorization server only consider valid when validating it for the first time.

The client is RECOMMENDED to "sender-constrain" the delegated access tokens by binding the delegated access tokens with public keys or certificates where the corresponding private keys are owned by the delegated parties, via techniques similar to OAuth 2.0 mTLS [RFC8705] or OAuth 2.0 DPOP [RFC9449].

## 9. Using Delegated Access Tokens

When the client accesses a delegated resource on the delegated party, the client MUST include the delegated access token as a bearer token [RFC6750] in the Delegated-Authorization header, used by the target resource server to verify requests from the delegated party. The Delegated-Authorization header MAY be used in combination with an Authorization header used by the delegated party to verify the request from the client.

For example:

```
GET /dp-resource HTTP/1.1
Host: delegated-party.example.com
Authorization: Bearer mF_9.B5g1234
Delegated-Authorization: Bearer mF_9.B5f-4.1JqM
```

Upon receiving a delegated resource request with a Delegated-Authorization header, the delegated party sends a request to the target resource server for the respective target resource. The delegated party **MUST** include the received delegated access token as a bearer token in the Authorization header.

For example:

```
GET /target-resource HTTP/1.1
Host: resource.example.com
Authorization: Bearer mF_9.B5f-4.1JqM
```

## 10. Verification of Delegated Access Tokens

Resource servers verify delegated access tokens through either local validation using pre-configured public keys or remote validation via token introspection [RFC7662] at the authorization server.

### 10.1. Local Verification

The resource server verifies delegated access tokens by:

1. The resource server is pre-configured with the authorization server's public key, or it fetches the public key via the authorization server's JWKS endpoint [RFC7517].
2. Checking the digital signature of the delegation token (part of the delegated access token) against the authorization server's public key.
3. Checking the digital signature of the delegated access token against the delegation key bound to the delegation token.
4. Verifying the delegated access token's permissions and validity are within the scope of the delegation token.
5. Verifying the delegated access token is within validity period, and the delegated access token's permissions cover the resource request.

## 10.2. Token Introspection

The resource server sends the delegated access token to the authorization server via the token introspection endpoint [RFC7662]. The authorization server verifies the delegated access token against its keys.

## 11. Privacy Considerations

This section describes the privacy properties of the local delegation approach defined in this specification.

### 11.1. Privacy Benefits

When a client creates delegated access tokens locally using a delegation token, the authorization server only observes the initial delegation token request. After that, the client can independently issue subordinate tokens without further authorization server interaction. This provides the following privacy benefits:

- \* **\*Minimized Authorization Server Visibility\***: The authorization server does not learn which delegated parties are authorized, what resources they access, or when they access them.
- \* **\*No Access Pattern Correlation\***: The authorization server cannot track usage patterns or correlate activities across different delegated parties.
- \* **\*Reduced Data Collection\***: The authorization server stores only metadata about the initial delegation token, not about each delegated access token.
- \* **\*Network Traffic Reduction\***: Fewer network round-trips to the authorization server reduce exposure to network surveillance.

### 11.2. Relationship to Token Exchange

Token Exchange [RFC8693] provides a mechanism for delegating access by having the client exchange one token for another at the authorization server. That approach requires authorization server involvement for each delegation event, which necessarily exposes delegation metadata to the authorization server. The local delegation approach in this specification performs delegation entirely on the client side after the initial token issuance, providing an alternative with different privacy characteristics.

### 11.3. Trade-offs

While local delegation provides significant privacy benefits, implementations should consider:

- \* **\*Client Responsibility\***: The client must securely protect the delegation key and properly enforce delegation rules.
- \* **\*Auditability\***: Deployments requiring delegation event audits can implement client-side logging rather than relying on authorization server monitoring.
- \* **\*Revocation\***: If a delegation key is compromised, the authorization server may need to revoke the entire delegation token.

## 12. Security Considerations

This specification extends OAuth 2.0 to support delegated authorization through hierarchical token issuance. While this enables fine-grained privilege delegation, it also introduces new trust and security considerations.

- \* Delegation tokens **MUST NOT** be sent to resource servers without subordinate delegated access tokens. Resource servers and authorization servers **MUST NOT** treat delegation tokens as regular OAuth access tokens.
- \* Clients **MUST** protect the delegation key, as compromise allows an attacker to mint valid delegated access tokens within the scope of the delegation token.
- \* Delegated access tokens **SHOULD** have short lifetimes and be bound to specific audiences, methods, and sender keys (e.g., via DPoP or mTLS) to mitigate replay and token leakage risks. Resource servers **MUST** validate both the delegation token and the delegated access token, ensuring the latter does not exceed the former's permissions.
- \* Token introspection **CAN** be used in scenarios where the tokens are kept opaque from the delegated party and the resource server. If employed, token introspection responses **MUST NOT** reveal sensitive internal information. Authorization servers **SHOULD** enforce rate limiting and audit token issuance and validation activities.

### 13. Operational Considerations

Deployments of this specification should consider the following operational aspects:

- \* **\*Key Management\***: Clients **MUST** securely store and rotate delegation keys. Authorization servers **SHOULD** support key rotation for delegation tokens and provide mechanisms to revoke compromised keys.
- \* **\*Token Lifetimes\***: Delegation tokens **SHOULD** have longer lifetimes than delegated access tokens to reduce authorization server load, and **SHOULD** be refreshable using refresh tokens.
- \* **\*Metadata Caching\***: Delegated party metadata at the well-known URI `/.well-known/oauth-delegated-party` can be cached by clients; a reasonable default TTL (e.g., 24 hours) is **RECOMMENDED**.
- \* **\*Error Handling\***: Delegated parties and resource servers **SHOULD** provide clear error responses (e.g., invalid token, insufficient scope) without exposing implementation details.
- \* **\*Interoperability\***: Implementers **SHOULD** ensure compatibility with existing OAuth 2.0 features such as PKCE, Rich Authorization Requests, and sender-constrained tokens.

### 14. IANA Considerations

#### 14.1. Well-Known URIs Registry

This specification registers the following entry in the "Well-Known URIs" registry:

- \* **\*URI suffix\***: `oauth-delegated-party`
- \* **\*Reference\***: [this document]
- \* **\*Status\***: permanent
- \* **\*Change controller\***: IETF
- \* **\*Related information\***: (none)

#### 14.2. OAuth Parameters Registry

This specification registers the following parameters in the "OAuth Parameters" registry:



**\*Delegation:\***

- \* **\*Name\***: delegation
- \* **\*Parameter Usage Location\***: authorization request, token request
- \* **\*Change Controller\***: IETF
- \* **\*Reference\***: [this document]

**\*Delegation Key:\***

- \* **\*Name\***: delegation\_key
- \* **\*Parameter Usage Location\***: token request
- \* **\*Change Controller\***: IETF
- \* **\*Reference\***: [this document]

**14.3. OAuth Access Token Types Registry**

This specification registers the following parameters in the "OAuth Access Token Types" registry:

- \* **\*Name\***: Delegation
- \* **\*Additional Token Endpoint Response Parameters\***: (none)
- \* **\*HTTP Authentication Scheme(s)\***: Bearer
- \* **\*Change Controller\***: IETF
- \* **\*Reference\***: [this document]

**14.4. HTTP Field Name Registry**

This specification registers the following parameters in the "Hypertext Transfer Protocol (HTTP) Field Name" registry:

- \* **\*Field Name\***: Delegated-Authorization
- \* **\*Status\***: permanent
- \* **\*Structured Type\***: (none)
- \* **\*Reference\***: [this document]

#### 14.5. OAuth Delegated Party Metadata Registry

This specification establishes the "OAuth Delegated Party Metadata" registry for OAuth 2.0 delegated party metadata names. The registry records the delegated party metadata parameter and a reference to the specification that defines it.

##### 14.5.1. Registration Template

**\*Metadata Name\*:** The name requested (e.g., "resource"). This name is case sensitive. Names may not match other registered names in a case-insensitive manner unless the designated experts state that there is a compelling reason to allow an exception.

**\*Metadata Description\*:** Brief description of the metadata (e.g., "Resource identifier URL").

**\*Change Controller\*:** For IETF Stream RFCs, list "IETF". For others, give the name of the responsible party. Other details (e.g., postal address, email address, home page URI) may also be included.

**\*Specification Document(s)\*:** Reference to the document or documents that specify the parameter, preferably including URIs that can be used to retrieve copies of the documents. An indication of the relevant sections may also be included but is not required.

##### 14.5.2. Initial Registry Contents

**\*Resources\*:**

\* **\*Metadata Name\*:** resources

\* **\*Metadata Description\*:** JSON array containing a list of target protected resources' resource identifier URLs

\* **\*Change Controller\*:** IETF

\* **\*Specification Document(s)\*:** [this document]

**\*Authorization Servers\*:**

\* **\*Metadata Name\*:** authorization\_servers

\* **\*Metadata Description\*:** JSON array containing a list of authorization server issuer identifiers

\* **\*Change Controller\*:** IETF

\* \*Specification Document(s)\*: [this document]

\*Supported Permissions:\*

\* \*Metadata Name\*: permissions\_supported

\* \*Metadata Description\*: JSON object indicating the permissions the delegated party may request

\* \*Change Controller\*: IETF

\* \*Specification Document(s)\*: [this document]

\*API Permissions:\*

\* \*Metadata Name\*: api\_permissions

\* \*Metadata Description\*: JSON object mapping API endpoints (resource identifiers) to the permissions required to access them

\* \*Change Controller\*: IETF

\* \*Specification Document(s)\*: [this document]

\*Delegated Party Documentation:\*

\* \*Metadata Name\*: delegated\_party\_documentation

\* \*Metadata Description\*: URL of a page containing human-readable information that developers might want or need to know when using the delegated party

\* \*Change Controller\*: IETF

\* \*Specification Document(s)\*: [this document]

## 15. References

### 15.1. Normative References

- [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>>.

- [RFC6750] Jones, M. and D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage", RFC 6750, DOI 10.17487/RFC6750, October 2012, <<https://www.rfc-editor.org/rfc/rfc6750>>.
- [RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <<https://www.rfc-editor.org/rfc/rfc7515>>.
- [RFC7516] Jones, M. and J. Hildebrand, "JSON Web Encryption (JWE)", RFC 7516, DOI 10.17487/RFC7516, May 2015, <<https://www.rfc-editor.org/rfc/rfc7516>>.
- [RFC7517] Jones, M., "JSON Web Key (JWK)", RFC 7517, DOI 10.17487/RFC7517, May 2015, <<https://www.rfc-editor.org/rfc/rfc7517>>.
- [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>>.
- [RFC7636] Sakimura, N., Ed., Bradley, J., and N. Agarwal, "Proof Key for Code Exchange by OAuth Public Clients", RFC 7636, DOI 10.17487/RFC7636, September 2015, <<https://www.rfc-editor.org/rfc/rfc7636>>.
- [RFC7662] Richer, J., Ed., "OAuth 2.0 Token Introspection", RFC 7662, DOI 10.17487/RFC7662, October 2015, <<https://www.rfc-editor.org/rfc/rfc7662>>.
- [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>>.
- [RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <<https://www.rfc-editor.org/rfc/rfc8259>>.
- [RFC8392] Jones, M., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "CBOR Web Token (CWT)", RFC 8392, DOI 10.17487/RFC8392, May 2018, <<https://www.rfc-editor.org/rfc/rfc8392>>.
- [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>>.

- [RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, DOI 10.17487/RFC8615, May 2019, <<https://www.rfc-editor.org/rfc/rfc8615>>.
- [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>>.
- [RFC8705] Campbell, B., Bradley, J., Sakimura, N., and T. Lodderstedt, "OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Tokens", RFC 8705, DOI 10.17487/RFC8705, February 2020, <<https://www.rfc-editor.org/rfc/rfc8705>>.
- [RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <<https://www.rfc-editor.org/rfc/rfc9110>>.
- [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>>.
- [RFC9700] Lodderstedt, T., Bradley, J., Labunets, A., and D. Fett, "Best Current Practice for OAuth 2.0 Security", BCP 240, RFC 9700, DOI 10.17487/RFC9700, January 2025, <<https://www.rfc-editor.org/rfc/rfc9700>>.
- [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>>.
- [I-D.lombardo-oauth-step-up-authz-challenge-proto] Lombardo, J., Babeanu, A., Zehavi, Y., and G. Fletcher, "OAuth 2.0 step-up authorization challenge proto", Work in Progress, Internet-Draft, draft-lombardo-oauth-step-up-authz-challenge-proto-02, 30 June 2025, <<https://datatracker.ietf.org/doc/html/draft-lombardo-oauth-step-up-authz-challenge-proto-02>>.

## 15.2. Informative References

[RFC8792] Watsen, K., Auerswald, E., Farrel, A., and Q. Wu, "Handling Long Lines in Content of Internet-Drafts and RFCs", RFC 8792, DOI 10.17487/RFC8792, June 2020, <<https://www.rfc-editor.org/rfc/rfc8792>>.

## Appendix A. Token Format

The example tokens in this section are shown in Flattened JSON Serialization [RFC7515] [RFC7516], un-base64url-encoded and with comments for ease of reading. When used as JWTs [RFC7519], they should be represented in Compact Serialization [RFC7515] [RFC7516]. Similarly, they can be represented as CWTs [RFC8392].

### A.1. Example 1

In this example, the delegation token is a JWS token signed with HS256, and the delegated access token is a JWS token signed with RS256.

\*Delegation Token\*:

```
{
  "protected": {
    "_comment": "to be base64url-encoded",
    "alg": "HS256",
    "typ": "JWT",
    "kid": "as-key-1"
  },
  "payload": {
    "_comment": "to be base64url-encoded",
    "iss": "https://as1.example.com",
    "sub": "user@example.com",
    "aud": "https://res1.example.com",
    "iat": 1772177588,
    "exp": 1774769588,
    "scope": "email:read email:send",
    "delegation_key": {
      "kty": "RSA",
      "n": "0Z5t3Rwhfz4MJxdNzg2I6Bsf9H_3kcMKtH6iwJsLV2DK01MVcHb6wsxHZpqRePjm5Q1-pImr4pT67hCHyAR6kS4QbTrBZcjSe8lsvZiifiBVWZxz8ZAfdj6EFSOWhFLZ0GVqe9NMslwX1hHOwmhRmMjGjexISPsZk5qN3gwqmD36H1GS-rE-cWlXI9f4pVcCdYqE0rfhZ7hDn5mrpWWY8wJuQa2jFqX-Fhyvt_Zh-qaa5Wp8cLJPS_ceAcLeS5mOYE58beelp51PuCDpq7a9fN4K_EpcSD7Ay0GrJxgN_VcLa7DgOAuU6zYDzk2KnmQ3qz35Tj5hy8jDHgF1AaSTw",
      "e": "AQAB"
    }
  },
  "signature": "SBwdWhy4dissabBRoyyxMxRuVUbsFYv_1HZFeROLh5A"
}
```

\*Delegated Access Token\*:

```
{
  "protected": {
    "_comment": "to be base64url-encoded",
    "alg": "RS256",
    "typ": "JWT",
    "kid": "delegation-key-1"
  },
  "payload": {
    "_comment": "to be base64url-encoded",
    "sub": "https://dpl.example.com",
    "aud": "https://res1.example.com",
    "iat": 1772181188,
    "exp": 1772184788,
    "scope": "email:read",
    "delegationToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6ImFzLWtleS0xIn0.eyJpc3MiOiJodHRwczovL2FzMzUleGFtcGxlLnVnbSIsInN1YiI6InVzZXJAZXhhbXBsZS5jb20iLCJhdWQiOiJodHRwczovL3Jlc2EuZXhhbXBsZS5jb20iLCJpYXQiOjE3NzIxNzc1ODgsImV4cCI6MTc3NDc2OTU4OCwic2NvcGUiOiJlbWFpbDpyZWZkIGVtYWlsOnNlbnMqIlCjKZWxlZ2ZF0aW9uX2tleSI6eyJrdHkiOiJSU0EiLCJuIjoimFoldDNSd2hmejRNSnhkTnpnMkk2QnNmOUhfM2tjTUt0SDZpd0pzTFYyREtPMUlWY0hiNndzeEhacHFSZVBqbTVRMSlwSWlyNHBUjdoQ0h5QVI2alM0UWJUckJaY2pTZThsc3ZaSWlpZmlCVldaehO4WkFmZGo2RUZTT1dorKxaMEdWcWU5Tk1zbHdYMWhIT3dtaFJtTWpHamV4SVNQclprNXFOM2d3cWlEMzZIMudTLXJFLWNxbFhJOWY0cFZjQ2RZcUUwcmZoWjdORG41bXJwVldZOHDkdVFhMmpGcVgtRmh5dnRfWmgtcWFhNVdwOGNMslBTX2NlQWNmZVM1bu9ZRTU4YmVlbHA1MVBlQ0RwcTdhoWZONEtfRXBju0Q3QXkwR3JKEgdOX1ZjTEE3RGdPQXVVNnpZRHpRMktubVEzcxozNVRqNWwh5OGpeSGdGMUFhU1R3IiwizSi6IkFRQUIifX0.SBwdWhy4dissabBRoyyxMxRuVUBsfYv_lHZFeROLh5A"
  },
  "signature": "df5sTdp6j-ULv3To3yqYFFZWLRy41wu6jpKfsZdjMNMeRH3TXy5nSc-lhdFfJE1VGnyCa5Rf9YGlvacGVy84maFCq0zhNTiLL8xBYieCKTAhmMuWzvIUZFB0e1KBj2YXskpnU6CFTKhAlbw5lidZSGXdCEnygxaumpO_AxlyghU8RqrQgFlwAsoDLcounl-FROuHUzqKGWqYSbxellGJ3dTt5ftod7SjGqM2Quz4BvfQMpw82VVGq7YlvRJMuPe4ITUeikFNQ672bvVrjhWHuHg1Q9HuFj5iroHQwr6dLtfKVDBtw33A_lCnsWfk4vPWDJ-1fy3Klj4ht_bhBIVJA"
```

### A.2. Example 2

In this example, the delegation token chain consists of three tokens: a top-level delegation token issued by the authorization server, a subordinate delegation token created by the client, and a delegated access token created from the subordinate token. The top-level delegation token has `max_delegation_depth` of 3, which allows for two levels of delegation.

**\*Top-Level Delegation Token\*:**





```
{
  "protected": {
    "_comment": "to be base64url-encoded",
    "alg": "RS256",
    "typ": "JWT",
    "kid": "as-key-2"
  },
  "payload": {
    "_comment": "to be base64url-encoded",
    "iss": "https://as1.example.com",
    "sub": "user@example.com",
    "aud": "https://res1.example.com",
    "iat": 1772177588,
    "exp": 1774769588,
    "scope": "email:read email:write email:send",
    "delegation_key": {
      "kty": "RSA",
      "n": "301Eva-_zsENOhylq99puKjF6NBq8S8Ntw9sufyCB9LrTSs-xRoDQ7X6rYbPEOn5E-6ASZyUW2vFxLacj0Wir05hznlnfRLRTmJhud4im5CofAhuiPeL0Mbs3WFLdBfWVGXSP2O5UNkd7dJ2KqudqxNF81Nrt51RpmPR59pUZS3REFAYgtqhQ07WuCTzd2VpFttPFurDSiWcriyELNdgsT9wap0nyceSVLQgY9ZgR0VvsmHj_LUKielt5RCjTsaMHJH6x3QI0tjKG9Psbil00RlhA9WEg0O6E6KOC_StEBnK_Ntb2ArGPDj1NPzTqWOeeev84MM8VXdRzpTL0rRmHQ",
      "e": "AQAB"
    },
    "max_delegation_depth": 3
  },
  "signature": "p10jJQ2tASuFPPCXzdbuSHaZiPTF2jM1Z_My2_qPx0IAAJJ5RVo9Km2hFeXjqGcjkG5wEFxMw_q8JO4809bVaKNBalk_-UoBxf5KlMpldMFIUyV4D8Sugshh73iqVii5HW2MY4uS5Gj46GDUcX1NQrg8LW6Wa4zso0SPfIfunXF0dvBG24cCUoIFor1NzryZ7OcVt9kdC4_uXGoSmjtG1lXsRvgb216fZFuZV0F88zWFsDcBZn4i8qGcBS2XAHjW4jbH7Y108dvl2h0afftmNkG2ZgHPL3phj_i-unQsAxxCUI_8Mjtff5iwe586uQw_1iED2IUlEXOGX-Ld5GgTw"
```

**\*Subordinate Delegation Token\*:**

```
{
  "protected": {
    "_comment": "to be base64url-encoded",
    "alg": "RS256",
    "typ": "JWT",
    "kid": "delegation-key-2"
  },
  "payload": {
    "_comment": "to be base64url-encoded",
    "aud": "https://res1.example.com",
    "iat": 1772181188,
    "exp": 1772782388,
    "max_delegation_depth": 1,
    "scope": "email:read email:write",
    "delegationToken": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6ImFzLWtleS0yIn0.eyJpc3MiOiJodHRwczovL2FzMS5leGFtcGxlLmNvbSIsInN1YiI6InV4ZXJAZXhhbXBsZS5jb20iLCJhdWQiOiJodHRwczovL3Jlc2EuZXBsZS5jb20iLCJpYXQiOiE3NzIxZXhhbDODgsImV4cCI6MTc3NDc2OTU4OCwic2NvcGUiOiJlbWVpbDpyZWZkIGVtYWlsOndyaXRlIGVtYWlsOnNlbnQmIiLCJkZmVzL2Z0aW9uX2tleSI6eyJrdHkiOiJSU0EiLCJuIjoim08xRVZhLV96c0VOT2h5bHE5OXB1S2pGnk5CcThTOE50dzlzdWZ5Q0I5THJUU3MteFJvRFE3WDZyWWJQRU9uNUUtNkFTWnlVVzJ2RnhMYWNqMfdpcjAlaHpubG5mUkxSVG1KaHVkNgltNUNPZkFodWlQZUwwTWJzmlDGTGRCZldWR1hTUDJPNVVOa2Q3ZEoyS3F1ZHF4TkY4MU5ydDUxUnBtUFIlOXBVWlMzUkVmQVlnbHFOUta3V3VDVHpkMlZwRnR0UEZ1ckRTaVdjcm15RUxOZGdzVD13YXAwbmljZVNWTFFFnWtlaZl1wVnZzbUhqX0xVS2llMXQ1UkNqVHNhTUhKSDZ4M1FJMHRqS0c5UHNiaWwwTlJsaEE5V0VnT082RTZLT0NfU3RFQm5LX050YyJBckdQRGpsTlB6VHFXT2VlZXY4NE1NOFZYZFJ6cFRMT3JSbUhRIiwizSI6IkFRQUIifSwibWF4X2RlbgVnYXRpb25fZGVwdGgiOjN9.pl0jJQ2tASuFPpCXzdbuSHaZiPTf2jm1Z_my2qPx0IAAJJ5RVo9Km2hFeXjqGcjkG5wEFxmWw_q8JO4809bVaKNBalk_-UoBxf5KlMpldMFIUyV4D8Sugsh73iqvii5HW2MY4uS5Gj46GDUCXlNQRGZl6W6a4zsoSPfIfunXF0dvBG24cUoIFOrlNzryZ7OcVt9kdC4 uXGoSmjtGll1XsRvqb216fZFuZV0F88zWFSdCBZn4i8qCBS2XAHjW4jbH7Y108dvl2h0afftmNkG2ZqHPl3ph
```

```
j_i-unQsAxhCUI_8MjtfF5iwe586uQw_1iED2IUlEXOGX-Ld5GgTw"
  },
  "signature": "lnf8Wz5M_B4dVTO0jT0aQa0ZSqEbTZVGb1HU_dk1tJ4eCbKCB8lm00wJXc07Ys8gKqqb3T
S4tKUbnwwKo3_Dw3FGYjbDqDE7rBtdc0lqi6fKSecff2ujVCi2U2dLjX78_h7U0LP1XbJuhkjdBYa-_mwiygAXf0j
8cilXTigbTWJfgqczivmkJzl9a_5HJ8HYF5ohdGmVBhOfyRrwGikYoHEQB6Ye_6jwTD_IQsUJe0eYdD4xAdA7C2aw
ZAvhdYduKXit7WSyGuEhbyKopsm6WVZz4BkjWDjcJ2S7Xjb6RxPCGc_nNiAMsGTlnG9MIj-oGYy8gIWPSLEyF_6sk
8hiaQ"
}
```

\*Delegated Access Token\*:

```

{
  "protected": {
    "_comment": "to be base64url-encoded",
    "alg": "RS256",
    "typ": "JWT",
    "kid": "delegation-key-2-sub"
  },
  "payload": {
    "_comment": "to be base64url-encoded",
    "sub": "https://dpl.example.com",
    "aud": "https://res1.example.com",
    "iat": 1772184788,
    "exp": 1772188388,
    "scope": "email:read",
    "delegationToken": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6ImRlbGVnYXRpb24ta2V5LTl1IiwiaWF0IjoiMTY3MjE4NDc3IiwiaXNjaW50IjoiImRlbGVnYXRpb25Ub2t1biI6ImV5SmhiR2NpT2lKU1V6STFOaUlsZW5SNWNDSTZJa3BYVkJkZ01tdHBAQ0k2SW1GekxXZGxlUzB5SW4wLmV5SnBjM01pT2lKb2RIUndjem92TDJGek1TNWxlR0Z0Y0d4bExtTnZiU01zSW5OMVlpSTZJblZ6WlhKQVpYaGhiWEJzWlMlamIyMGlMQ0poZFdRaU9pSm9kSFJ3Y3pvdkwzSmxjekV1WlhoaGJYQnNaUzVqYjIwaUxDSnBZWFFpT2pFM056SXhOemMxT0Rnc01tVjRjQ0k2TVRjM05EYzJPVFU0T0N3aWMyTnZjR1VpT2lKbGJXRnBiRHB5WldGa0lhVnRZV2xzT25keWFYUmxJR1Z0WVdsc09uTmxibVFpTENKAlpXGxaMkYwYVc5dVgydGxlU0k2ZXlKcmRIa2lPaUpTVTBFaUxDSnVJam9pTTA4eFJWWmhMVjk2YzBWTlQyaDViSEU1TlhCMVMycEdOazVDY1RoVE9FNTBkemx6ZFdaNVEwSTVUSePvVTNNDGVGSnZSRkUzV0RaeVdXSlFSVTl1TlVVeE5rRlRXbmXWVnpKMLJuaElZV05xTUZkcGNqQTFhSHB1Ykc1bVvrefNWRzFLYUUhWa05HbHROVU5QWmtGb2RXbFFaVXd3VfdeUR1JDWmxkV1IxaFRVREpQTlZWt2EYUTNaRW95UzNGMVpIRjRUalk0TVUleWREVXhVbkJ0VUZJMU9YQlZXBE16VWtWbVFWBg5kSEZvVVRBM1YzVkrWSHBrtWxad1JuUjBVRVoxY2tSVGFwZGpjbWw1UlV4T1pHZHpWRGwzWVhBd2JubGpaVk5XVEZGblUUbGFaMUl3Vm5aemJVaHFYMHhWUzJsbElYUTFVa05xVkhOaFRVaEtTRFo0TTFGSK1IUnFTMGm1VUhoAwFXd3dUMUpzYUVFNvYwVm5UMDgyUlRaTFQwTmZVM1JGUW01TFgwNTBZakpCY2tkUVJHCHNUbE12VkhGWFQyVmxawFk0TkUxTk9GWllaRko2Y0ZSTVQzSlniVWhSSWl3aVpTSTZJa0ZSUUVJawZTd2liV0Y0WDJSbGJHVM5ZWFFwYjI1ZlphVndkR2dpT2pOOS5wMU9qSlEydeFTdUZQUENYemRidVNIYVppUFRGMmpNMVpfTXkyX3FQeDBJQUFKSjVSVm85S20yaEZlWGpxcUdjamtHNXdFRnhNd19xOEPNDhPOWJYUyU0QmExa18tVW9CeGYlS2xNcGxkTUZJVXlWNEQ4U3Vnc2hoNzNpcVZpaTVIVzJNWTR1UzVHaJQ2R0RVY1gxTlFSZzhmVzZXYTR6c28wU1BmSWZ1blhGMGR2QkcyNGNDVW9JRk9yMU56cnlaN09jVnQ5a2RDNF9lWEdevU2lqdEdsMVhzUnZnYjIxNmZaRnVaVjBGODh6V0ZzRGNCWm40aThxR2NCUzJYQUhqcVzRqYkg3WTFPOGR2bDJoMGFmZnRtTmtHMLpnsFBsm3Boal9pLXVUUXNBeGhDVUlfoElqdGZGNWl3ZTU4NnVRd18xaUVEMklVbEVYT2dYLUxkNUdnVHcifQ.lnf8Wz5M_B4dVTOOjT0aQa0ZSqEbTZVGB1HU_dk1tJ4eCbKCB8lm00wJXc07Ys8gKqgb3TS4tKUbnwwKo3_Dw3FGYjbdQDE7rBtdc0lqi6fKSecff2ujVCi2U2dLjX78_h7U0LP1XbJuhkjdBYa-_mwiygAXf0j8cilXTigbTWJfgqczVmKJz19a_5HJ8HYF5ohdGmVBhOfyRrwGikYoHEQB6Ye_6jwTD_IQSUJe0eYdD4xAdA7C2awZAvhdYduKXit7WSyGuEhbyKopsm6WVZz4BkjWDjcJ2S7Xjb6RxPCGc_nNiAMSGTlnG9MIj-oGYy8gIWPSLEyF_6sk8hiaQ"
  },
  "signature": "mn-0y0F3lx0IM-60UsBp1_M7uRhdt28dPqxt6-s1Onjm-obyF-p5XbQV9ji5Q56X4cb3jL16RkM789ODTgt5y4QNdoA2WhlS0WldSOzYkV1Xj2yrC0mZakYCHTPzzgvfeUjeNj-fDwmtgRb2Lfm6Z6dguT071WQkaIyH4IW5xVxroqbnltFM07uX3huIV9tC_is926rWaDw7eQDmlnhhjs_QSgUcTFleJy7h0jEtzs70iij5bQU-B9nn6OfXdpblj05vx3lCSnZfL2h2EaQYK3CbjrflbDnIYKtJ0NMx5D03KRcv_uPlG5HHRL5cuMz8eToKc08rc0Qs5PxDbEqV4A"
}

```

## Appendix B. Integration with Step-Up Authorization

This specification can be used together with step-up authorization [I-D.lombardo-oauth-step-up-authz-challenge-proto]. When a resource server returns an `insufficient_authorization` error, the client can create a new delegated access token with the required permissions from its existing delegation token.

This is particularly useful for AI agents that hold a delegation token. Upon receiving a step-up authorization challenge, the agent can automatically create a new delegated access token with the additional `authorization_details` required by the resource server, without requiring the user to re-authenticate.

Resource servers that support both delegated authorization and step-

up challenges SHOULD use the `body_instructions` parameter to specify the exact `authorization_details` or scope required.

## Appendix C. Use Cases

### C.1. Delegating Subset of Access Rights to Specialized AI Agents

Enterprise Identity and Access Management systems often employ Role Based Access Control (RBAC) or Attribute Based Access Control (ABAC), assigning a set of minimal permissions to the employee based on their role, department, or other attributes. AI Agent can be an employee's personal assistant, or a virtual employee of a certain department in general. The permissions delegated to an AI agent CAN be long-term, but an AI agent MUST NOT directly inherit all its owner's access

rights. Rather, they SHOULD be a subset of its owner, bound to specific service/API/database/codebase according to its specialty and dedicated workflow.

Role	Service / Component
Resource Owner	an enterprise, individual or a department
Client	agent's client application
Delegated Party	CI-CD agent, test agent, DEV agent, research agent
Authorization Server	enterprise IAM system
Resource Server	enterprise IT systems
Target Protected Resource	DEV/STAGE/PROD environments, internal knowledge database

Table 1: AI Agents

## C.2. Third-Party Analytics Platform Integrated in an Enterprise SaaS

In this scenario, a corporate customer uses a Software-as-a-Service (SaaS) Customer Relationship Management (CRM) application. The customer wishes to gain business insights by granting a specialized third-party analytics platform limited access to its CRM data.

The CRM application obtains a delegation token from the enterprise's identity provider. It then creates a narrowly scoped delegated access token for the analytics service. This token only permits read access to a predefined, non-sensitive subset of customer data (e.g., names and identifiers, but not personal email addresses). The analytics platform uses this token to pull data, generates an aggregated business intelligence report, and delivers it back to the CRM application for the corporate customer to view.

Role	Service / Component
Resource Owner	companyA (the tenant)
Client	SaaS CRM application
Delegated Party	analytics service
Authorization Server	enterprise IdP
Resource Server	CRM application server
Target Protected Resource	CRM application's data retrieval API

Table 2: Enterprise-SaaS

## Authors' Addresses

Ruochen Li  
 Huawei Int. Pte Ltd  
 Email: li.ruochen@h-partners.com

Haiguang Wang  
 Huawei Int. Pte Ltd  
 Email: wang.haiguang.shieldlab@huawei.com

Chunchi Peter Liu  
 Huawei Technologies  
 Email: liuchunchi@huawei.com

Tieyan Li  
 Huawei Int. Pte Ltd  
 Email: Li.Tieyan@huawei.com