Web Authorization Protocol
Internet-Draft
Intended status: Standards Track
Expires: 14 August 2026

Y. Jia
S. Peng
Huawei

10 February 2026

OAuth 2.0 Scope Aggregation for Multi-Step AI Agent Workflows
draft-jia-oauth-scope-aggregation-00

Abstract

This document describes a scope-aggregated OAuth 2.0 authorization pattern for multi-step AI agent workflows. An AI agent aggregates the scopes required across a workflow and only initiates a single authorization procedure for the aggregated scope. This reduces repeated user consents and multiple authorization round-trips, improving authorization efficiency.

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 14 August 2026.

Table of Contents

1. Introduction

Recent advances in large language models (LLMs) enable AI agents to plan and execute multi-step workflows for complex tasks. With agent communication protocols, AI agents can call third-party tools and delegate tasks to other AI agents. When an AI agent is acting on behalf of a human user, these interoperations should require authentication and authorization.

The OAuth 2.0 authorization framework [RFC6749] has been a widely preferred method for granting an AI agent access to protected resources in existing agent communication protocols. Acting as an OAuth client, an AI agent must present a valid access token with appropriate scopes for each request to access a protected resource. If the token is missing or lacks the required scope, the resource server will reject the request with an HTTP 401/403 error response. The agent is then expected to perform authorization server discovery and initiate an OAuth flow to obtain an access token as shown in Figure 1, involving end user authentication and consent. This challenge-triggered flow repeats for each scope deficiency when an AI agent workflow spans across multiple protected resources, causing user fatigue and increased end-to-end latency.

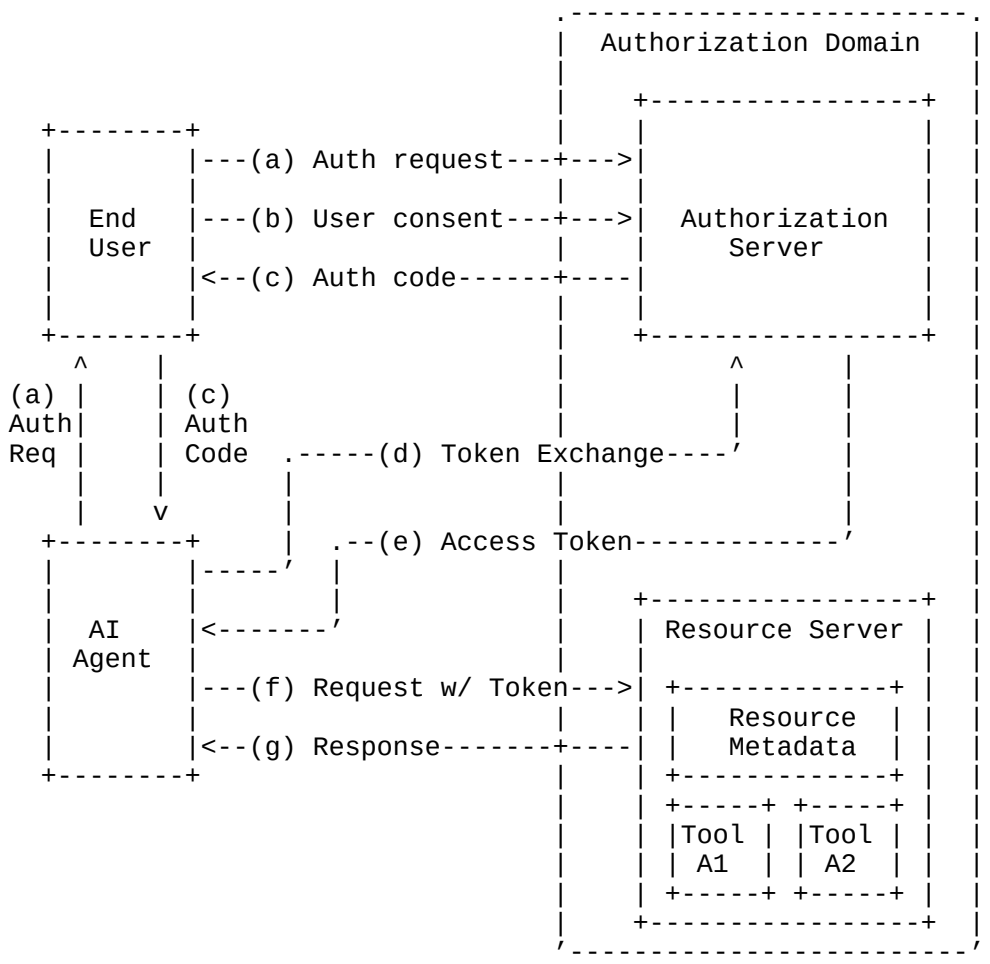


Figure 1: OAuth 2.0 authorization flow for AI agents

This document defines an authorization domain as a logical grouping of the scopes authorizable by a single trust anchor (e.g., an OAuth 2.0 authorization server). Within this domain, an AI agent can request multiple scopes (e.g., email:read, calendar:write) that share a common user identity and consent context. Given a multi-step workflow, an AI agent first computes the minimal aggregation of the required scopes within each authorization domain. Subsequently, the agent performs authorization server discovery and initiates a scope-aggregated authorization flow to obtain the access token. This approach reduces repeated user consent and multiple authorization round-trips while preserving least privilege.

1.1. 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.

2. Terminology

AI Agent
: An entity with built-in intelligence, which can perform actions to accomplish tasks, possibly on behalf of an end-user or another agent.

Agent Communication Protocol
: A protocol that defines how an agent discovers and utilizes server-exposed resources (e.g., tools, skills, etc.). For example, Model Context Protocol (MCP) protocol [MCP] defines how to discover and call tools, and Agent-to-Agent (A2A) protocol [A2A] defines how to invoke other AI agents.

Agent Workflow
: A sequence of steps to accomplish a task, which may be pre-built by a human or planned by an agent. These steps may include access to one or more resources on remote resource servers. An agent workflow may be structured as a chain or a tree with parallel branches.

Resource Server (RS)
: An OAuth 2.0 term for the server hosting one or more protected resources that require access token for access. The resources are invocable, server-exposed units of functionality.

Authorization Server (AS)
: An OAuth 2.0 term for the server that issues access tokens to AI agent after successfully authenticating the resource owner and obtaining consent for authorization.

Authorization Domain (AD)
: A logical grouping of scopes authorizable by a single trust anchor. For example, scopes like mail.read and calendar.write within a single ecosystem belong to the same authorization domain.

3. Security Member in Resource Metadata

For the purpose of resource discovery, an AI agent obtains resource metadata by invoking a protocol method or by retrieving a description document from a well-known URI. Such resources may be referred to as "tools" or "skills" in different contexts. This document extends the resource metadata object by defining a security member that MAY be included to indicate a resource's authorization requirements.

The resource metadata is represented in JSON format. It MUST contain the following members:

name
: A string that provides a unique identifier for the resource.

description
: A human-readable string that describes the resource.

input_schema
: A machine-readable schema (e.g., JSON schema) describing the parameters required to execute the resource.

This document defines an OPTIONAL security member within the resource metadata. When present, the security object indicates the authorization requirements for the resource and contains the following members:

type
: REQUIRED. An array of strings specifying one or more supported security schemes. This document defines the value "oauth2" for use with the OAuth 2.0 authorization framework. Other type values, such as "apikey", "basic", and "oidc", may also be used.

scopes
: REQUIRED. An array of strings specifying the scope values required to access the resource. If this member is present and "oauth2" is included in the type array, the AI agent MUST possess all listed scopes to gain access.

as_metadata
: OPTIONAL. A string containing the URL to the OAuth 2.0 Authorization Server (AS) metadata [RFC8414]. This allows the agent to dynamically discover the authorization endpoints.

Figure 2 illustrates the structure of the security member within a resource definition:

```
{
  "name": "resource_identifier",
  "description": "...",
  "input_schema": { ... },
  "security": {
    "type": ["oauth2"],
    "scopes": ["scope_A"],
    "as_metadata": "https://server.example.com/.well-known/..."
  }
}
```

Figure 2: Structure of Resource Metadata with Security Member

A comprehensive, non-normative example of a full list of resource metadata is provided in Appendix A.

3.1. Comparison to Existing Practice

This document defines a uniform mechanism for advertising per-resource authorization requirements, drawing inspiration from and addressing gaps in existing protocols.

Agent-to-Agent (A2A) Protocol [A2A]: The A2A protocol defines security scheme objects based on the OpenAPI 3.2 specification. OpenAPI includes a security object within its schema that serves a purpose analogous to the security member defined here. It allows an API to declare its required security schemes (e.g., OAuth 2.0) and the necessary scopes for each operation. This document aligns with that established principle, offering a potentially simplified version.

Model Context Protocol (MCP) [MCP]: The MCP protocol specifies OAuth 2.1 as the authorization framework. However, MCP does not standardize per-tool scope advertisement in tool metadata. This can lead to reactive, challenge-triggered authorization flows when executing an AI agent workflow. The security member defined in this document directly addresses this gap by providing a mechanism for an MCP server to advertise these requirements.

3.2. Backward Compatibility

Inclusion of the security member in resource metadata is OPTIONAL. A server SHOULD omit the security member for resources that do not require authorization. To allow for implementation flexibility, a server MAY also omit the security member for a protected resource.

An AI agent MUST ignore a security member if it does not understand the contents. Furthermore, all agents MUST be able to handle authorization reactively (e.g., via a step-up flow) upon receiving an authorization error.

AI agents that recognize and support the security member SHOULD use the information it contains to perform proactive and aggregated authorization, as described in Section 4.

4. Workflow Execution with OAuth Scope Aggregation

Using the security member in the resource metadata, an AI agent can follow the sequence below to execute a workflow:

1. Resource Discovery: The AI agent fetches the resource metadata from the resource server. Resource metadata MAY include a security member describing the required authentication scheme, scopes, and AS metadata URL.

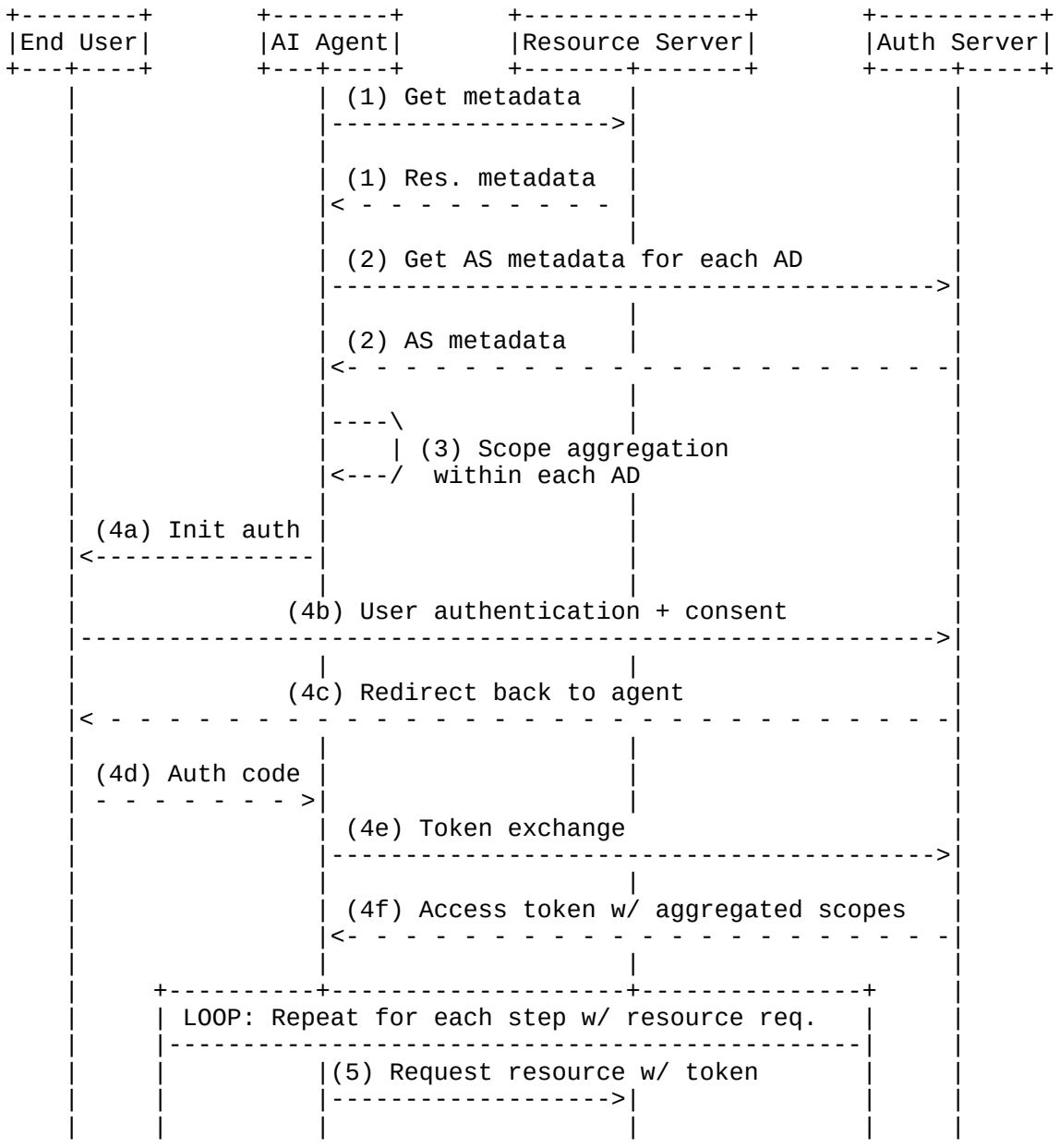
2. Authorization Domain Identification: Given a pre-defined or planned workflow, the AI agent iterates through each step and fetches the associated AS metadata to learn the authorization domain and authorization endpoints.

3. Scope Aggregation: For each authorization domain, the AI agent computes a least-privilege set of scopes required by the workflow, using the OPTIONAL security member defined in Section 3. The agent aggregates the required scopes across all resource invocations within the same authorization domain and removes duplicates. If, and only if, the authorization domain defines a hierarchical relationship among its scopes (e.g., a broader scope implies one or more narrower scopes), the agent MAY further reduce the requested scope set by removing any scope that is subsumed by another requested scope. For example, within a single authorization domain, an agent workflow (1) reads a document from the cloud drive with scope "drive.read", (2) updates the document with scope "drive.write", and (3) creates calendar events with scope "calendar.write". If the authorization domain defines that "drive.write" subsumes "drive.read", the aggregated scope set is ["drive.write", "calendar.write"].

4. Aggregated Scope Authorization: For each authorization domain, the AI agent initiates an OAuth 2.0 authorization code grant to obtain an access token with the aggregated scopes. During this step, the user is prompted to authorize the requested scopes.

5. Workflow Execution: The AI agent executes the multi-step workflow, using the obtained access tokens when accessing resources on resource servers.

This process is illustrated in Figure 3.



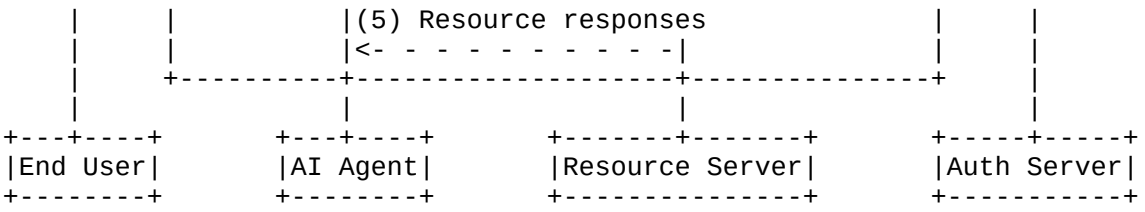


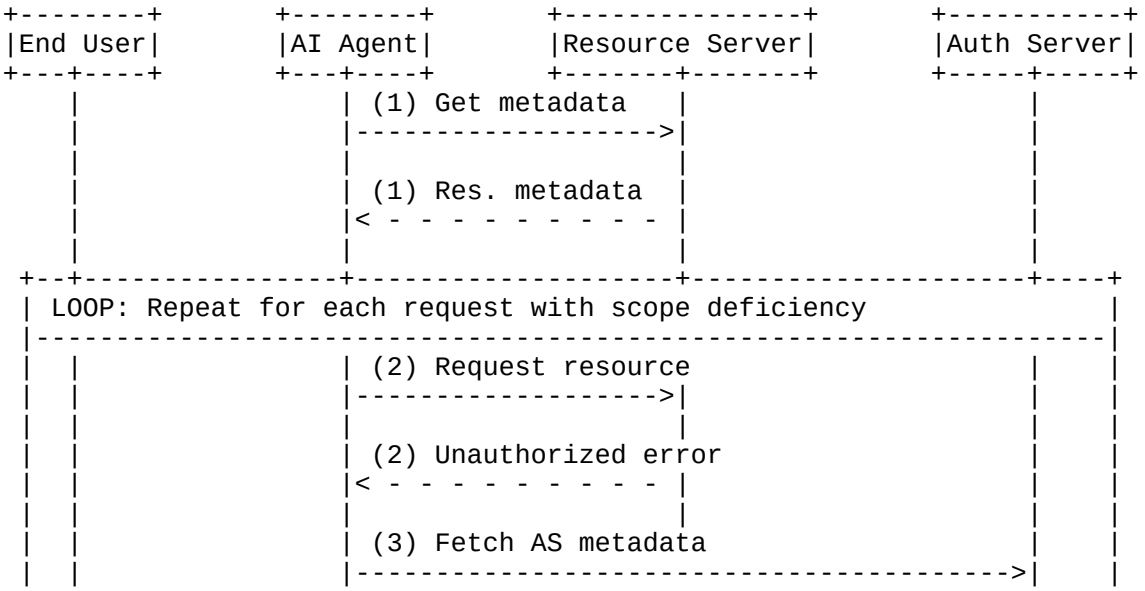
Figure 3: Sequence diagram of workflow execution with scope-aggregated authorization

5. Advantages

Compared to a reactive, challenge-triggered authorization flow, this scope-aggregated authorization strategy offers several benefits for multi-step AI agent workflows.

5.1. Reduced Consent Interactions and End-to-End Latency

In the absence of aggregated authorization, the end user is required to respond to multiple consent prompts for individual resources associated with distinct scopes. This interaction pattern is illustrated in Figure 4. On the contrary, the mechanism defined in this document enables the end user to authorize the AI Agent one single time per authorization domain. This approach mitigates user friction and consent fatigue, while simultaneously reducing the end-to-end latency of task completion.



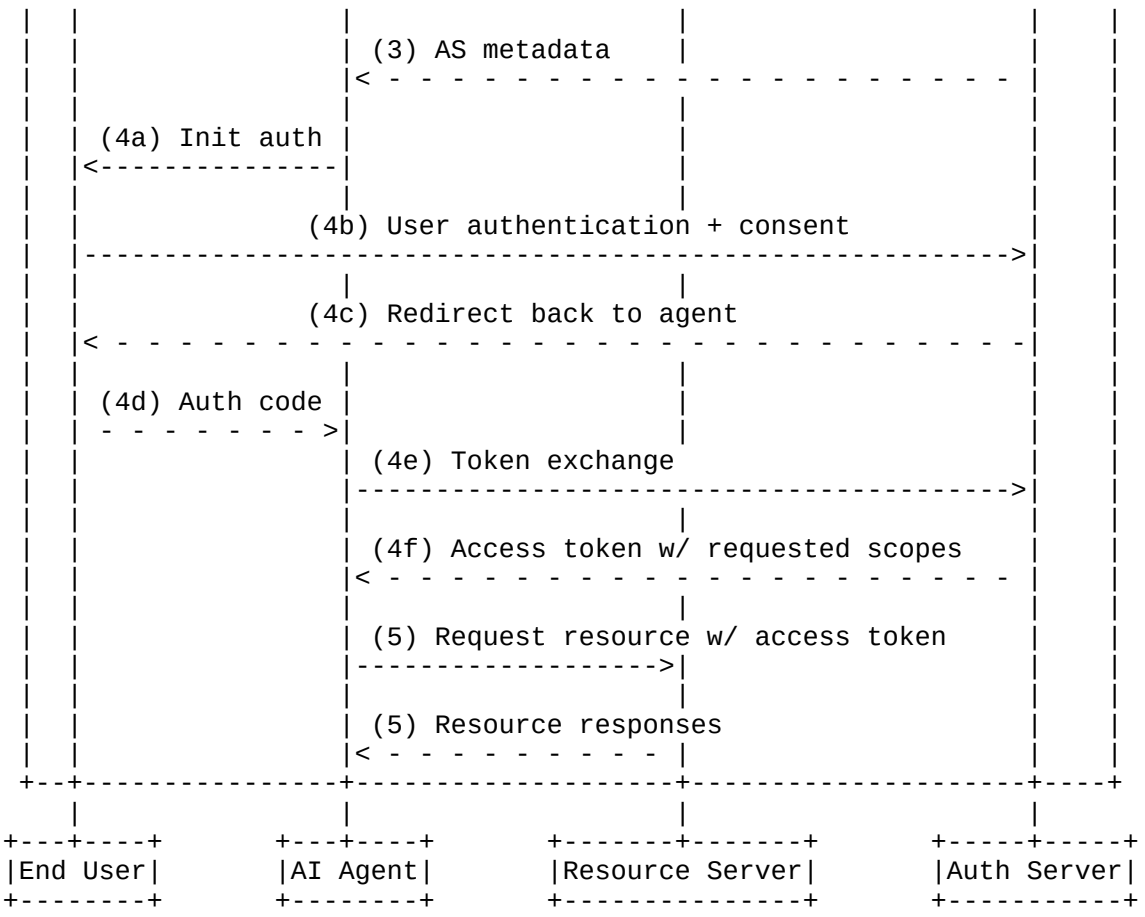


Figure 4: Sequence diagram of workflow execution without scope-aggregated authorization

5.2. Avoidance of Mid-Flow Failures due to Insufficient Privileges

By computing the required scopes and obtaining tokens in advance, the AI agent avoids mid-workflow failures caused by missing privileges. This increases the reliability of multi-step workflows, preventing partial completion and handling insufficient-scope conditions proactively rather than reactively.

6. IANA Considerations

This document includes no request to IANA.

7. Security Considerations

This section summarizes security risks that arise in scope-aggregated authorization for multi-step AI agent workflows and describes recommended mitigations. Implementations are RECOMMENDED to apply the mechanisms below, as appropriate to their threat model and deployment context.

7.1. Risk: Residual Privilege

In a sequential workflow involving Step A, B, and C, the AI agent still holds the permissions for Step A while executing Step B and C. More critically, after Step C is finished, the agent retains full access to Step A, B, and C.

Mitigation:

Token Downscoping [RFC8693] : Upon completing a step in the workflow, the AI agent SHOULD perform a token exchange request to the authorization server to exchange for a new token with a reduced scope set. This reduces the impact of token leakage during later steps.

Strict Token Revocation [RFC7009] : Upon successful completion or terminal failure of the workflow, the AI agent SHOULD immediately revoke the access token and any associated refresh tokens. The authorization server SHOULD limit refresh token lifetimes to reduce residual privilege.

Sender-Constrained Tokens [RFC9449] : It is RECOMMENDED to use sender-constrained access tokens that bind token use to a proof-of-possession (DPoP) key controlled by the AI agent. This reduces the risk that an aggregated token can be intercepted and replayed by a different actor.

7.2. Risk: User Blind Signing

When users are presented with a long set of aggregated scopes (e.g., "Read Mail", "Write Files", "Access Calendar", "Post to Slack"), they may approve the request without understanding its implications. This can undermine informed consent and increase the likelihood of over-authorization.

Mitigation:

Structured Consent UI: Authorization Servers SHOULD present requested privileges in a structured manner, grouped by workflow step, rather than as a flat list of scopes. Authorization Servers MAY additionally require step-level acknowledgment for high-risk privileges.

8. Acknowledgements

The authors would like to thank Yiyang Shao and Jinyang Li for useful discussion and ideas.

9. References

9.1. 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/info/rfc2119>.

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

[RFC7009] Lodderstedt, T., Ed., Dronia, S., and M. Scurtescu, "OAuth 2.0 Token Revocation", RFC 7009, DOI 10.17487/RFC7009, August 2013, <https://www.rfc-editor.org/info/rfc7009>.

[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/info/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/info/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/info/rfc8693>.

[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/info/rfc9449>.

9.2. Informative References

[A2A] Google, "Agent2Agent(A2A) Protocol", 2025, <https://a2a-protocol.org/latest>.

[MCP] Anthropic, "Model Context Protocol (MCP)", 2025, <https://modelcontextprotocol.io/docs/getting-started/intro>.

Appendix A. Example of a Resource List

This appendix is non-normative. Figure 5 shows an example resource list for a calendar server. In this example, both resources belong to the same authorization domain. For a workflow that accesses both resources, the AI agent can obtain an access token via OAuth 2.0 with the aggregated scope set ["calendar.read", "calendar.write"]. If the authorization domain defines that "calendar.write" subsumes "calendar.read", the requested scope set can be further reduced to ["calendar.write"].

```
[
  {
    "name": "CalendarReader",
    "description": "Read events from the user’s calendar",
    "input_schema": {
      "type": "object",
      "properties": {
        "start_date": {
          "type": "string",
          "format": "date-time"
        },
        "end_date": {
          "type": "string",
          "format": "date-time"
        }
      },
      "required": ["start_date"]
    },
    "security": {
      "type": ["oauth2"],
      "scopes": ["calendar.read"],
      "as_metadata": "https://auth.calendar.com/.well-known/
      oauth-authorization-server"
    }
  },
  {
    "name": "CalendarWriter",
    "description": "Create events on the user’s calendar",
    "input_schema": {
      "type": "object",
      "properties": {
        "summary": { "type": "string" },
        "start": { "type": "string", "format": "date-time" },
        "end": { "type": "string", "format": "date-time" }
      },
      "required": ["summary", "start", "end"]
    },
    "security": {
      "type": ["oauth2"],
      "scopes": ["calendar.write"],
      "as_metadata": "https://auth.calendar.com/.well-known/
      oauth-authorization-server"
    }
  }
]
```

Figure 5: An example of the resource list of a calendar server

Contributors

The following people contributed significantly to this document:

Yiyang Shao
Huawei
Email: shaoyiyang@huawei.com

Jinyang Li
Huawei
Email: lijinyang9@huawei.com

Authors' Addresses

Yukuan Jia
Huawei
Huawei Bld., No.156 Beiqing Rd.
Beijing
100095
China
Email: jiayukuan@huawei.com

Shuping Peng
Huawei
Huawei Bld., No.156 Beiqing Rd.
Beijing
100095
China
Email: pengshuping@huawei.com