oauth
Internet-Draft
Intended status: Standards Track
Expires: 17 September 2026

D. Liu
H. Zhu
Alibaba
S. Krishnan
Cisco

16 March 2026

Agent Operation Authorization
draft-liu-agent-operation-authorization-02

Abstract

This document specifies the Agent Operation Authorization framework — a structured mechanism that enables verifiable delegation of actions from human principals to autonomous AI agents with fine-grained agent operation authorization.

The framework introduces two distinct phases:

* Agent Operation Authorization Request: A structured proposal of operations converted to a JSON Web Token (JWT) without including the user’s original natural language input.

* Agent Operation Authorization Token: A JSON Web Token representing confirmed authorization for a specific agent operation, enforceable at runtime by agents and verifiers. It cryptographically verifies user intent, prevents unauthorized or hallucinated actions, and ensures auditable traceability of each authorized operation.

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 17 September 2026.

Table of Contents

1. Introduction

In agent-based systems, especially those involving generative capabilities, it is essential to convey not only what actions are permitted, but also the original intent behind them and conditions under which an autonomous agent may act on behalf of a principal.

This document specifies the Agent Operation Authorization framework — a mechanism that enables verifiable delegation of actions from human principals to autonomous AI agents with fine-grained agent operation authorization. The framework includes Agent Operation Authorization Proposal and Agent Operation Authorization phases.

This specification defines several new JSON Web Token (JWT) claims to support agent operation authorization: agent_operation_proposal (used in authorization requests), agent_operation_authorization (used in access tokens), agent_identity (for verified agent-user binding), evidence (for user intent provenance), and context (for policy evaluation in proposal phase). These claims enable fine-grained control over autonomous agent operations and ensure cryptographic verification of user intent.

The AI agent constructs a structured agent_operation_proposal object and submits it to the Authorization Server (AS) via OAuth 2.0 Pushed Authorization Requests (PAR) [RFC9126], without including the user's original natural-language instruction.

This design ensures that downstream verifiers can validate both the policy boundaries and the provenance of the initiating instruction, without dependency on Decentralized Identifiers (DIDs). This enables secure, auditable delegation for autonomous AI Agent.

As an optional enhancement for user experience, the agent MAY include a reference (e.g., a hash or identifier) to the original user prompt in the PAR request. This reference can be used by the AS to display the original user intent during the authorization consent process, and MAY be included in the final authorization token for audit purposes.

Upon successful user confirmation and authentication of the Authorization Proposal during the first phase, the Authorization Server (AS) SHALL issue an Agent Operation Authorization Token. This token serves as the access token for subsequent interactions.

The agent MUST present this JWT access token when accessing protected resources at the AS, using the mechanisms defined in OAuth 2.0 [RFC6749] and bearer token usage rules [RFC6750].

Together, these components ensure that AI systems act only within user-approved boundaries, mitigating risks such as hallucination.

It is designed for use in autonomous AI Agent system, multi-agent orchestration, and regulated domains such as finance, healthcare, and public services — particularly where accountability and auditability are important.

The framework supports enterprise identity providers, and zero-trust architectures.

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 RFC 2119 [RFC2119] and RFC 8174 [RFC8174].

3. agent_operation_proposal Token Structure

The PAR-JWT (Pushed Authorization Request in JWT format) is used in the first phase. Its purpose is to deliver the agent-proposed operational strategy to the AS, enabling the generation of a high-quality consent UI and establishing an evidentiary starting point, without including the user’s original input.

Its format is defined as follows:

```
{
    "iss": "https://client.myassistant.example",
    "sub": "user_12345@myassistant.example",
    "aud": "https://as.online-shop.example",
    "exp": 1731369540,
    "iat": 1731320400,
    "agent_user_binding_proposal": { ... },
    "agent_operation_proposal": "package agent\nallow { input.transaction
.amount <= 50.0 }",
    "context": { ... }
}
```

Figure 1

The agent_user_binding_proposal claim is a structured JSON object proposed by the client (e.g., an AI agent) to describe its own identity context when acting on behalf of a user. In the Agent Operation Authorization Request (i.e., the PAR-JWT), this claim represents a _proposal_ of the agent-to-user binding and is _not yet cryptographically endorsed_ by the Authorization Server (AS).

The Authorization Server can determine whether a user is authorized to access a given agent by verifying the user’s identity token (e.g., an ID Token) issued by a trusted Identity Provider. This requires that the token includes the agent’s identifier in its aud (audience) claim, indicating that the token was intended for use with that agent. Since only users authorized by the Identity Provider receive tokens with the appropriate audience, validating the token and its

audience suffices to establish legitimacy. The Authorization Server MUST validate this binding during request processing. Only upon successful validation— and after obtaining explicit user consent— does the AS issue an Agent Operation Authorization Token that includes the agent_identity claim as part of a trusted authorization assertion. At that point, the presence of agent_identity in the AS-issued token serves as an implicit attestation that the binding has been verified.

The agent_operation_proposal field is a Rego policy string for OPA enforcement.

The context field is a structured input format for OPA decision-making.

The evidence field in the authorization token contains user confirmation records generated during the authorization phase. See Section 4 for the detailed format of this field.

The agent_user_binding_proposal field format is as follows:

```
{
  "agent_user_binding_proposal": {
    "user_identity_token": "eyJhbGciOiJSUzI1NiIs...",
    "agent_workload_token": "eyJhbGciOiJFUzI1NiIs...",
    "device_fingerprint": "dfp_abc123"
  }
}
```

Figure 2

The user_identity_token MUST be a cryptographically verifiable identity credential (e.g., an OpenID Connect ID Token). The AS validates this token to establish the user’s identity before issuing an authorization token.

The agent_workload_token identifies the running agent workload. Unless otherwise specified, it SHOULD be a Workload Identity Token (WIT) as defined in [I-D.ietf-wimse-workload-creds]. Deployments that already use other workload identity mechanisms (such as SPIFFE SVID) MAY map those tokens into this field, provided that the Authorization Server (AS) can validate them and bind the workload identity to the expected agent.

Field	Type	Description	Requirement
user_identity_token	JWT string	A verifiable identity token for the end user, issued by a trusted Identity Provider.	MUST be an OpenID Connect ID Token or equivalent cryptographically signed token. The Authorization Server will validate this token to establish the user's identity.
agent_workload_token	JWT string	A verifiable workload identity token for the agent, typically a Workload Identity Token (WIT) as defined in I-D.ietf-wimse-workload-creds; deployments using SPIFFE MAY map SPIFFE SVIDs into this field.	MUST be a valid, signed workload identity credential. The Authorization Server will validate this token to establish the agent's identity and trustworthiness.
device_fingerprint	string	An optional unique identifier for the client device instance.	OPTIONAL. If provided, it SHOULD be a stable, privacy-preserving fingerprint (e.g., derived from hardware and app properties). Used by the AS to populate the clientInstance

				field in the resulting agent_identity claim.

Table 1: agent_user_binding_proposal fields

Field	Type	Description	Requirement
version	string	Schema version	MUST be "1.0"
id	URI	Unique identifier	MUST be a UUID URI
issuer	URI	Issuer of identity	MUST be a valid URI
issuedTo	string	Target user identity	MUST be a cryptographically verifiable user identifier issued by a trusted Identity Provider, such as the sub claim from an ID Token.
issuedFor	object	Agent context	MUST contain platform:the logical platform or service namespace, client:The software client identifier (e.g., mobile app ID), clientInstance:A unique fingerprint of the client instance (e.g., device+app hash).
issuanceDate	timestamp	When identity was issued	MUST conform to ISO 8601
validFrom	timestamp	When identity becomes	MUST conform to ISO 8601

		valid	
expires	timestamp	When identity expires	MUST conform to ISO 8601

Table 2: agent_identity fields

The agent_operation_proposal field format is as follows:

```
{
"agent_operation_proposal": "package agent\nallow { input.transaction.amount
<= 50.0 }"
}
```

Figure 3

The agent_operation_proposal field should be a valid Rego policy string.

The context field format is as follows:

```
{
"context": {
    "channel": "mobile-app",
    "deviceFingerprint": "dfp_abc123",
    "language": "zh-CN",
    "user": {
    "id": "user_12345@myassistant.example"
    },
    "agent": {
    "instance": "dfp_abc123",
    "platform": "personal-agent.myassistant.example",
    "client": "mobile-app-v1.myassistant.example"
    }
}
}
```

Figure 4

The Agent Client sends this PAR-JWT to the Authorization Server (AS) via the Pushed Authorization Request (PAR) mechanism, as defined in [RFC9126] (OAuth 2.0 Pushed Authorization Requests).

4. Agent Operation Authorization Token

Upon successful user authorization and authentication, the Authorization Server (AS) issues a Verifiable Agent Operation Credential in the form of a JWT token. The purpose of this credential is to serve as a digitally signed and independently verifiable "authorization letter", which enables the Personal Agent to perform authorized operations on behalf of the user. The issuer of the credential is the Authorization Server (AS) (indicated by the iss claim), and the intended audience is the Resource Server (indicated by the aud claim) that will verify and honor this authorization. The credential becomes effective immediately after the user clicks "Allow" or "Consent".

```
{
  "iss": "https://as.online-shop.example",
  "sub": "user_12345@myassistant.example",
  "aud": "https://api.online-shop.example",
  "exp": 1731369540,
  "iat": 1731320700,

  // ====== Evidence of User Confirmation ======
  "evidence": {
    "id": "evidence-confirmation-abc123",
    "user_confirmation_record": {
      "displayed_content": "Add items under $50 to cart during the Nov 11 promotion (vali
d until 23:59)",
      "user_action": "confirmed_via_button_click",
      "timestamp": 1731320595,
      "session_context": {
        "oauth_session_id": "session_abc123",
        "device_fingerprint": "dfp_xyz789"
      }
    },
    "as_signature": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9...signature_over_confirmation_r
ecord"
  },

  // ====== Agent Identity ======
  "agent_identity": { ... },

  // ====== Agent Operation Authorization ======
  "agent_operation_authorization": {
    "policy_id": "opa-policy-789"
  },


  // ====== auditTrail ======
  "auditTrail": {
      "evidence_reference": "evidence-confirmation-abc123",  // References the evidence o
bject by ID
      "semanticExpansionLevel": "medium",
      "userAcknowledgeTimestamp": 1731320580,
      "consentInterfaceVersion": "consent-ui-v2.1"
  },

  // ====== Optional: Reference to Proposal ======
  "references": {
    "relatedProposalId": "urn:uuid:op-proposal-456"
  }
}
```


Figure 5

The evidence field in the authorization token contains user confirmation records generated during the authorization phase.

The evidence field format in the authorization token is as follows:

```
{
  "evidence": {
    "user_confirmation_record": {
      "displayed_content": "string",
      "user_action": "string",
      "timestamp": "NumericDate",
      "session_context": {
        "oauth_session_id": "string",
        "device_fingerprint": "string"
      }
    },
    "as_signature": "signature_over_user_confirmation_record_by_AS"
  }
}
```

Figure 6

The evidence field MAY alternatively use a W3C Verifiable Credential format for compatibility with existing credential infrastructures. When using W3C VC format, the credential MUST contain user confirmation evidence and be signed by the Authorization Server.

Alternative W3C Verifiable Credential format for evidence:

```
{
  "evidence": {
    "type": "VerifiableCredential",
    "credentialSubject": {
      "type": "UserConfirmationEvidence",
      "userAction": "confirmed_via_ui_interaction",
      "displayedContent": "Add items under $50 to cart during Nov 11 promotion",
      "confirmationTimestamp": "2025-11-11T10:33:15Z",
      "sessionId": "session_abc123"
    },
    "issuer": "https://as.example.com",
    "issuanceDate": "2025-11-11T10:33:15Z",
    "@context": ["https://www.w3.org/2018/credentials/v1"]
  }
}
```

Figure 7

The agent_identity claim is issued by the Authorization Server (AS) after successfully validating a user identity token and confirming the agent's workload identity. It contains an authoritative representation of the binding between the agent and the user, along with contextual information necessary for authorization decisions.

```
{
  "agent_identity": {
    "version": "1.0",
    "id": "urn:uuid:agent-identity-789",
    "issuer": "https://as.example.com",
    "issuedTo": "https://idp.example.com|user-12345",
    "issuedFor": {
      "platform": "personal-agent.myassistant.example",
      "client": "mobile-app-v1.myassistant.example",
      "clientInstance": "dfp_abc123"
    },
    "issuanceDate": 1731320130,
    "validFrom": 1731320130,
    "expires": 1731369540
  }
}
```

Figure 8

Field	Type	Description	Requirement
version	string	Schema version	MUST be "1.0"
id	URI	Unique identifier for this binding instance	MUST be a UUID-based URI
issuer	URI	Issuer of the agent_identity claim	MUST be the URI of the Authorization Server
issuedTo	string	Verified user identifier on whose behalf the agent acts	MUST be a globally unique, cryptographically verifiable identifier derived from a validated identity token (e.g., the sub claim, optionally prefixed with the issuer URI such as https://idp.example.com|user-12345). It MUST NOT be a

			plain username or unverified string.
issuedFor	object	Context identifying the agent	MUST contain: platform: Logical service namespace; client: Software client identifier (e.g., mobile app ID); clientInstance: Unique fingerprint of the client instance (e.g., device+app hash).
issuanceDate	timestamp	Time when the binding was issued	MUST conform to ISO 8601 UTC
validFrom	timestamp	Time when the binding becomes effective	MUST conform to ISO 8601 UTC
expires	timestamp	Time when the binding expires	MUST conform to ISO 8601 UTC

Table 3: agent_identity fields in the authorization token

The agent_identity claim is signed by the AS to ensure its authenticity and integrity. The AS SHALL validate all inputs before issuing this claim, ensuring that the identities of both the user and the agent are verified.

The policy_id field is a string that serves as an OPA policy reference and MUST match a registered policy in the AS

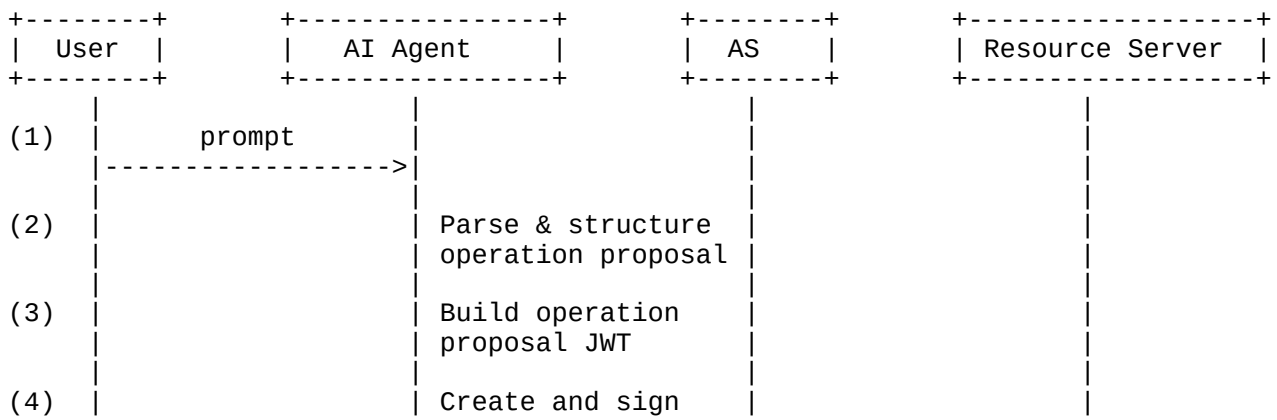
* auditTrail establishes a complete, semantically traceable chain—from the user’s original intent to the system’s final executed action—in AI Agent scenarios. This mechanism is known as a _Semantic Audit Trail_. The specific purposes and their descriptions are outlined in the following table:

Purpose	Description
1. Intent Provenance	Records what the user originally said (e.g., "Add something cheap to cart on Nov 11 night") to prevent disputes such as: "I didn’t say I wanted to add anything to cart!"
2. Action Interpretation	Documents how the system interpreted and rendered the input into a concrete operation (e.g., "Add items under $50 to cart during the Nov 11 promotion (valid until 23:59)"), reflecting the AI’s reasoning process.
3. Semantic Transparency	Shows whether semantic expansions or default values were applied (e.g., mapping "cheap" to $50, defining "night" as 00:0006:00).
4. User Confirmation Evidence	Includes timestamps indicating when the user reviewed and confirmed the interpreted action, serving as proof of authorization.
5. Accountability Support	Enables post-hoc analysis in case of erroneous transactions: Was the issue due to ambiguous user input, system misinterpretation, or misleading UI guidance.

Table 4: Purposes and Descriptions of the Semantic Audit Trail

5. Workflow

5.1. High-Level Flow



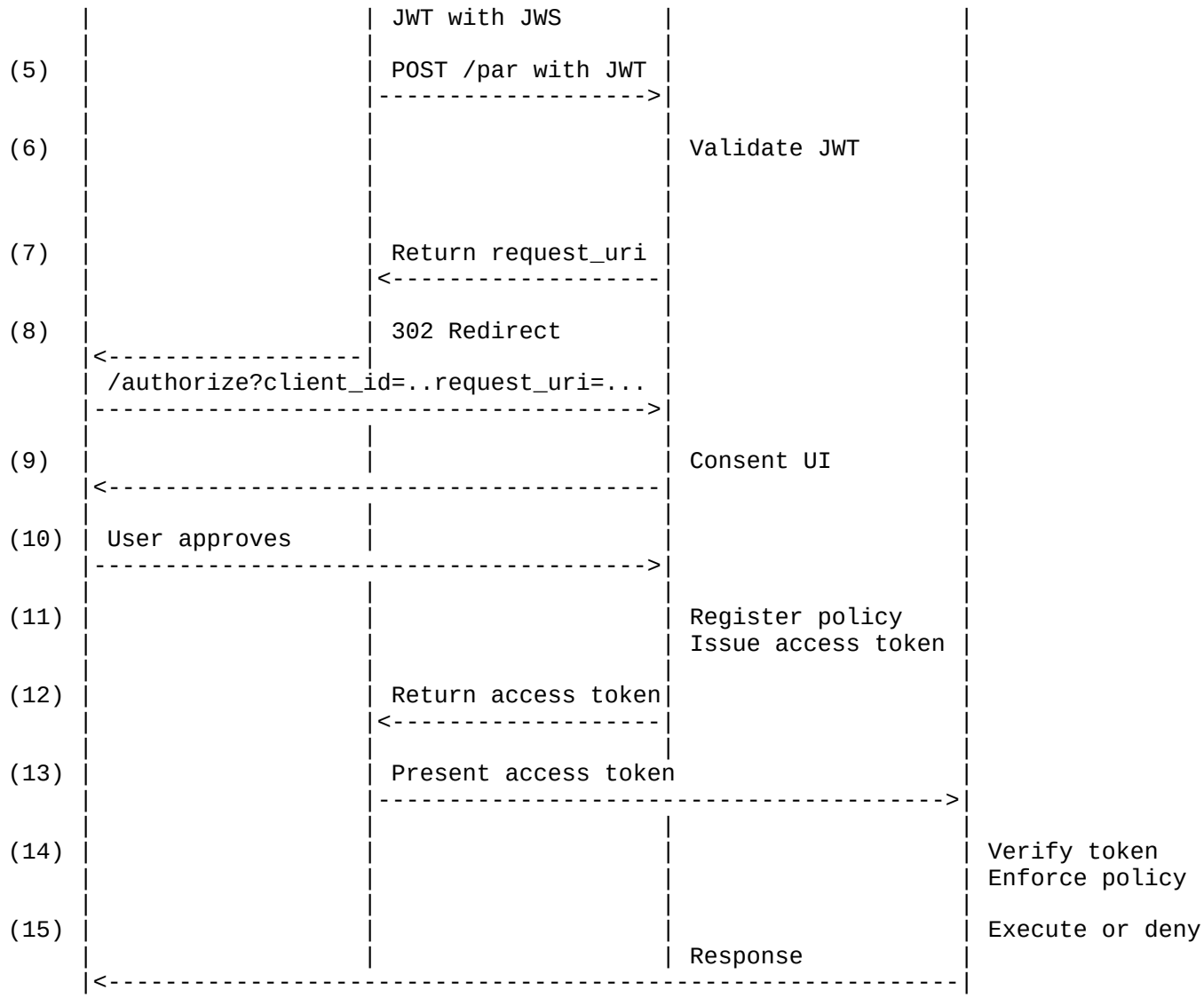


Figure 9

5.2. Detailed Process Flow

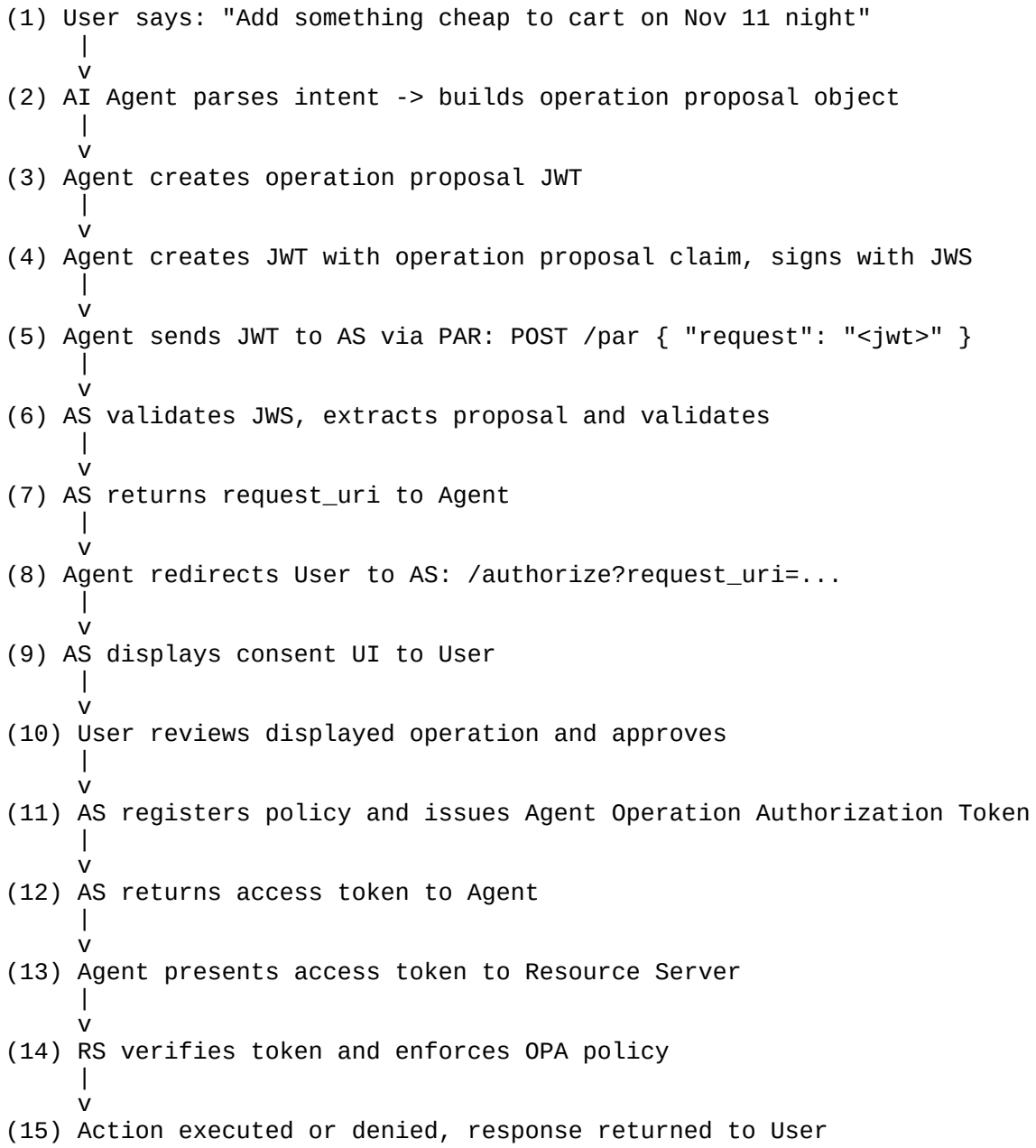

(1) User says: "Add something cheap to cart on Nov 11 night"
(2) AI Agent parses intent -> builds operation proposal object
(3) Agent creates operation proposal JWT
(4) Agent creates JWT with operation proposal claim, signs with JWS
(5) Agent sends JWT to AS via PAR: POST /par { "request": "<jwt>" }
(6) AS validates JWS, extracts proposal and validates
(7) AS returns request_uri to Agent
(8) Agent redirects User to AS: /authorize?request_uri=...
(9) AS displays consent UI to User
(10) User reviews displayed operation and approves
(11) AS registers policy and issues Agent Operation Authorization Token
(12) AS returns access token to Agent
(13) Agent presents access token to Resource Server
(14) RS verifies token and enforces OPA policy
(15) Action executed or denied, response returned to User

Figure 10

6. Agent-to-Agent Delegation

In multi-agent systems, a primary agent (Agent A) may delegate a subset of its authorized operations to a secondary agent (Agent B). This specification supports such delegation while preserving:

* End-to-end auditability back to the original human principal;

* No privilege escalation beyond the original authorization scope;

* Explicit AS validation at every delegation hop.

To support secure agent-to-agent delegation, the Agent Operation Authorization Token MAY include a delegation_chain claim. This claim is an ordered list (from most recent to original) of cryptographically verifiable delegation events, enabling resource servers to validate that the current agent's authority is derived—without escalation—from an original human-confirmed authorization.

The delegation_chain is an array of delegation records, ordered from most recent to earliest. Each record MUST be cryptographically bound to the issuing Authorization Server during token issuance.

```
   "delegation_chain": [
   {
     "delegator_jti": "urn:uuid:token-abc-123",
     "delegator_agent_identity": { /* agent_identity of Agent A */ },
     "delegation_timestamp": 1734516900,
     "operation_summary": "Delegate inventory check for item X",
     "as_signature": "eyJhbGciOiJSUzI1NiIs..."
   }
 ]
```

Figure 11

Each entry in the delegation_chain is signed by the Authorization Server at issuance time, ensuring its integrity and non-repudiation. The chain is extended—not copied—by the AS during each delegation step. The following subsections describe how agents initiate delegation and how the AS validates and extends this chain.

The delegation_chain contains the following fields:

Field	Type	Description	Requirement
delegator_jti	string (URI)	The JTI (JWT ID) of the delegator's authorization token, serving as a reference to the prior authorization in the delegation chain.	REQUIRED. MUST be a valid JWT ID that can be resolved by the AS.
delegator_agent_identity	object	The agent_identity claim from the delegator's token, identifying the delegating agent.	REQUIRED. MUST match the agent_identity structure defined in this specification.
delegation_timestamp	timestamp	The time when this delegation was authorized by the AS.	REQUIRED. MUST conform to ISO 8601 UTC format.
operation_summary	string	A human-readable description of the delegated operation for audit and logging purposes.	OPTIONAL. Useful for post-hoc analysis and compliance reporting.
as_signature	string (JWT)	Cryptographic signature from the AS over this delegation record, ensuring integrity and non-repudiation.	REQUIRED. MUST be verifiable using the AS's public key.

Table 5: delegation_chain Fields

6.1. Delegation Request

When initiating delegation, Agent A submits a Pushed Authorization Request (PAR) to the AS containing:

* Its current Agent Operation Authorization Token (or a reference via jti if tokens are stored server-side);
* A new agent_user_binding_proposal for Agent B (note: user remains the original human principal);
* A requested_sub_operation descriptor (e.g., policy template ID or scope hash).

Importantly, Agent A does _not_ submit raw credentials like sourcePromptCredential. The AS validates the user confirmation evidence and authorization scope using the provided token reference.

6.2. Authorization Server Validation

The AS MUST perform the following checks:

1. Validate that the submitted token (or jti) is valid and issued by this AS.
2. Confirm that the original token permits delegation (e.g., contains "delegation_allowed": true).
3. Verify that the requested sub-operation is strictly narrower in scope than the original authorization. This may be implemented via:
    1. Pre-registered policy templates with hierarchical relationships;
    2. Runtime evaluation using a policy engine (e.g., OPA) if enabled;
    3. Scope string containment (for simple cases).
4. Authenticate Agent B's identity via its agent_workload_token in the new binding proposal.

If all checks pass, the AS issues a new Agent Operation Authorization Token for Agent B. The new token:

* References the same original human intent (via internal linkage, not token exposure);

* Includes a new agent_identity for Agent B;

* Extends the delegation_chain with a new, AS-signed record referencing Agent A's token (jti).

6.3. Trust Chain Preservation

The resulting token for Agent B contains:

* A fresh agent_identity identifying Agent B;

* An updated delegation_chain with one additional entry;

* No exposure of the original user's ID Token or Agent A's private credentials (while the issuedTo field allows maintaining the link to the human principal).

Resource Servers can validate the entire chain by:

1. Verifying the AS signature on each delegation_chain entry;

2. Confirming that the final agent_identity.issuedTo matches the original human principal;

3. Ensuring no operation exceeds the cumulative scope of the chain.

7. Security Considerations

The JWS signature provides integrity protection for the authorization token, while the evidence field contains user confirmation records generated during the authorization phase.

Authorization Servers MUST validate the user confirmation evidence and session context before issuing authorization tokens.

The Authorization Server MUST generate a cryptographic signature over the user_confirmation_record to ensure its integrity and non-repudiation. This signature provides authoritative attestation that the user confirmed the displayed content at the specified timestamp.

When using W3C Verifiable Credential format for evidence, Authorization Servers MUST validate the credential's signature and verify that it contains appropriate user confirmation evidence as described in this specification.

Public keys referenced by issuerKey MUST be obtained through secure, trusted mechanisms (e.g., pre-registration, PKI).

Expression evaluation (e.g., CEL) MUST occur in sandboxed environments.

The use of PAR prevents leakage of sensitive operation data in URLs.

This specification assumes a threat model in which the agent implementation (including its evidence construction and signing logic) is part of the trusted computing base, while the large language model (LLM) used to derive operation proposals is not. The mechanisms defined here are primarily intended to make the transformation from the user's original intent to the final authorized operation transparent and auditable, and to avoid granting the LLM direct control over authorization decisions.

During the authorization phase, the Authorization Server (AS) acts as the witness of the user's consent: it presents the rendered operation to the user on an AS-controlled interface and, upon explicit approval, issues the Agent Operation Authorization Token embedding user confirmation evidence. This evidence includes the displayed content, user action, and confirmation timestamp, all covered by the AS's signature on the token for audit and verification purposes.

Deployments MAY define whitelists or policy profiles that limit which classes of operations an agent can be authorized to perform for a user, especially for highly sensitive resources. This can help reduce the risk of over-broad delegation.

For Agent-to-Agent delegation, the Authorization Server acts as the central policy enforcer and trust anchor. It MUST NOT issue delegation tokens unless it can cryptographically verify the user confirmation evidence and confirm that the proposed sub-operation is within the bounds of the original authorization.

The delegation_chain MUST be constructed and signed by the AS (and not self-reported by agents) in order to prevent forgery.

8. IANA Considerations

8.1. JWT Claim Registration

This document requests IANA to register the following two claims in the "JSON Web Token Claims" registry, following the procedure defined in RFC 8126.

Claim Name: agent_identity

Claim Description: A structured claim that conveys the identity and

issuance metadata of an autonomous agent acting on behalf of a user. It includes a unique identifier, issuer, target user, deployment context, and validity timestamps, enabling secure binding of agent operations to a verified identity.

Change Controller: IETF

Specification Document: This document requests registration of the agent_identity claim in the IANA "JSON Web Token Claims" registry [RFC7519].

Claim Name: agent_operation_proposal

Claim Description: A Rego policy string proposed by an agent for authorization evaluation. This claim is used in the initial authorization request to convey a policy that, upon validation and registration by the Authorization Server, will be referenced via a policy_id in subsequent access tokens.

Change Controller: IETF

Specification Document: This document, Section X.Y ("Agent Operation Proposal")

Claim Name: agent_operation_authorization

Claim Description: A structured claim that conveys authorization metadata for agent-performed operations, including a reference to a registered policy via the policy_id field. This claim is included in access tokens issued after successful policy validation and registration by the Authorization Server.

Change Controller: IETF

Specification Document: This document, Section X.Z ("Agent Operation Authorization")

Claim Name: context

Claim Description: A structured claim providing contextual information for policy evaluation, including user and agent identity attributes, device characteristics, channel, and locale. This claim serves as the input data for Open Policy Agent (OPA) enforcement decisions.

Change Controller: IETF

Specification Document: This document, Section X.Z ("context")

Claim Name: delegation_chain

Claim Description: An optional array of AS-signed delegation records tracing agent-to-agent authorizations. Each record includes: delegator_jti (reference to prior token), delegator_agent_identity (delegating agent's identity), delegation_timestamp, operation_summary (optional human-readable description), and as_signature (AS cryptographic signature ensuring integrity). Enables end-to-end validation of delegation lineage without exposing raw credentials.

Change Controller: IETF

Specification Document: This document, Section 6.

8.2. JSON Schema (Informative)

While not required for interoperability, implementers may find it useful to validate claim structures using JSON Schema. Informative schemas for the claims defined in this document may be developed and published separately. Such schemas are not normative and do not require IANA registration.

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>.

[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>.

[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/info/rfc7519>.

[RFC9126] Lodderstedt, T., Campbell, B., Sakimura, N., Tonge, D., and F. Skokan, "OAuth 2.0 Pushed Authorization Requests", RFC 9126, DOI 10.17487/RFC9126, September 2021, <https://www.rfc-editor.org/info/rfc9126>.

9.2. Informative References

[I-D.ietf-wimse-workload-creds]
IETF WIMSE Working Group, "Workload Identity Credentials", Work in Progress, Internet-Draft, draft-ietf-wimse-workload-creds, October 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-wimse-workload-creds>.

[OpenID] OpenID Foundation, "OpenID Connect Core 1.0", November 2014, <https://openid.net/specs/openid-connect-core-1_0.html>.

Acknowledgments

TBD

Authors' Addresses

Dapeng Liu
Alibaba
Email: max.ldp@alibaba-inc.com

Hongru Zhu
Alibaba
Email: hongru.zhr@alibaba-inc.com

Suresh Krishnan
Cisco
Email: suresh.krishnan@gmail.com