Web Authorization Protocol
Internet-Draft
Intended status: Standards Track
Expires: 15 September 2026

T. Uzua
GudLab

14 March 2026

AgentID: An Identity Protocol for Autonomous AI Agents
draft-gudlab-agentid-protocol-00

Abstract

This document defines the AgentID Protocol, an open standard for establishing, verifying, and managing the identity of autonomous AI agents operating across the Internet. As AI agents increasingly interact with web services, APIs, and each other on behalf of humans and organisations, the industry lacks a universal mechanism to determine which agent is making a request, who is accountable for it, and what it is permitted to do.

AgentID introduces the Agent Identity Token (AIT), a signed JWT carrying agent identity, owner verification, capability, and delegation chain claims. The protocol defines agent registration, owner verification levels, service-side access tiers, and transparent delegation chains with scope attenuation. It builds on existing standards including OAuth 2.0, OpenID Connect, JSON Web Token (JWT), and JSON Web Key (JWK), adapting them to the unique requirements of non-human autonomous actors.

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 15 September 2026.

Table of Contents

1. Introduction

Today's Internet authentication infrastructure was designed for humans interacting with services through browsers. OAuth 2.0 [RFC6749] enables a user to grant an application limited access to their resources. OpenID Connect [OpenID.Core] enables a user to prove their identity across services. Neither protocol was designed for autonomous software agents that:

* Operate independently without a human in the loop

* Chain together in multi-step workflows where Agent A dispatches tasks to Agent B

* Make decisions and take actions at machine speed on behalf of principals who may not be present

The result is a proliferation of ad-hoc solutions: raw API keys with no ownership tracing, OAuth tokens repurposed beyond their design intent, workload identity frameworks such as [SPIFFE] that address service-to-service authentication but not agent-specific concerns like owner accountability and delegation chains, and proprietary agent authentication schemes that fragment the ecosystem.

AgentID addresses this gap by providing a lightweight, cryptographically secure identity layer purpose-built for autonomous AI agents. It answers three questions that no existing standard addresses comprehensively:

1. Which agent is this?

2. Who is accountable for it?

3. What is it permitted to do?

1.1. Terminology

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.

Agent An autonomous software system that interacts with services on behalf of a principal (human user or another agent).

Agent Identity Token (AIT) A signed JWT that an agent presents to authenticate itself to a service.

Owner The human or organisation legally accountable for an agent's actions.

Principal The entity on whose behalf an agent acts. May be a user, an organisation, or another agent.

Delegation Chain The ordered sequence of principals and agents through which authority was delegated, from the original authoriser to the acting agent.

Registry A service that stores agent identities, public keys, and owner verification records.

Scope Attenuation The principle that each link in a delegation chain can only have equal or fewer permissions than the previous link.

Verification Level A numeric indicator (0-3) of the rigour of identity verification performed on an agent's owner.

2. Threat Model

Without a standardised agent identity layer, services face the following threats:

Unattributable Abuse An anonymous agent submits large volumes of spam, books fake appointments, or scrapes content. The service has no way to trace the actor or hold anyone accountable.

Impersonation A malicious agent claims to act on behalf of a reputable organisation without any verifiable proof.

Privilege Escalation An agent granted read access to a resource exploits loose controls to perform write or delete operations.

Delegation Opacity Agent A spawns Agent B which spawns Agent C. A service receiving a request from Agent C has no visibility into the delegation chain or the original authorising principal.

Rate-Limit Evasion Without stable agent identity, rate limiting falls back to IP addresses, which agents trivially rotate.

3. Protocol Requirements

The protocol MUST satisfy the following requirements:

1. *Verifiable Identity:* A service MUST be able to cryptographically verify an agent's identity without a synchronous call to a central authority.

2. *Owner Accountability:* Every agent identity MUST be traceable to a verified human or organisational owner.

3. *Granular Access Control:* Services MUST be able to define policies based on agent identity, owner identity, verification level, and declared capabilities.

4. *Delegation Transparency:* When an agent acts on behalf of another agent or a human, the full chain of delegation MUST be inspectable.

5. *Revocability:* Compromised or abusive agent identities MUST be revocable in real time.

6. *Interoperability:* The protocol MUST work across agent frameworks, programming languages, and deployment models.

7. *Performance:* Token verification MUST be possible offline using cached public keys, with registry lookups reserved for enhanced checks.

4. Agent Identity Token (AIT)

The Agent Identity Token is a signed JSON Web Token (JWT) conforming to [RFC7519], with a custom claim set designed for agent identification. The token is self-contained: any service with access to the issuer's public key can verify the token without contacting the registry.

4.1. Token Header

The AIT JOSE header MUST contain the following parameters:

alg REQUIRED. MUST be "ES256" (ECDSA using P-256 and SHA-256, per [RFC7518] Section 3.4).

typ REQUIRED. MUST be "AIT+jwt".

kid REQUIRED. Key identifier for key rotation, corresponding to a key in the registry's JWKS [RFC7517].

```
{
  "alg": "ES256",
  "typ": "AIT+jwt",
  "kid": "agentidp-2026-03-01"
}
```

Figure 1: AIT Header Example

4.2. Token Claims

The AIT payload contains the following claims organised into four groups:

4.2.1. Identity Claims

agent_id REQUIRED. String. Globally unique agent identifier assigned by the registry.

agent_name REQUIRED. String. Human-readable name of the agent.

agent_version RECOMMENDED. String. Semantic version of the agent software.

agent_description OPTIONAL. String. Brief description of the agent's purpose.

4.2.2. Owner Claims

owner_id REQUIRED. String. Unique identifier for the agent's owner in the registry.

owner_type REQUIRED. String. Either "person" or "org".

owner_name REQUIRED. String. Display name of the owner.

verification_level REQUIRED. Integer (0-3). See Section 5.

4.2.3. Capability Claims

capabilities OPTIONAL. Array of strings. Declared capabilities

using the namespace convention defined in Appendix A.

4.2.4. Delegation Claims

delegation_chain OPTIONAL. Array of delegation objects. See Section 8.

4.2.5. Standard JWT Claims

iss REQUIRED. String. The registry URL that issued the agent identity.

sub REQUIRED. String. Same value as agent_id.

aud RECOMMENDED. String. The target service URL.

iat REQUIRED. NumericDate. Time at which the token was issued.

exp REQUIRED. NumericDate. Expiration time. MUST NOT exceed 24 hours after iat. A lifetime of 1 hour or less is RECOMMENDED for high-security contexts.

jti REQUIRED. String. Unique token identifier for replay prevention.

4.3. AIT Example

The following is an example AIT payload:

```
{
  "agent_id": "ag_7xK9m2nP4qRtL8",
  "agent_name": "AcmeBookingAgent",
  "agent_version": "1.2.0",
  "owner_id": "own_T1mR4xBq9",
  "owner_type": "org",
  "owner_name": "Acme Inc",
  "verification_level": 2,
  "capabilities": ["calendar:read", "calendar:write"],
  "delegation_chain": [{
    "principal_type": "user",
    "principal_id": "usr_abc123",
    "granted_at": "2026-03-01T10:00:00Z",
    "scopes": ["calendar:read"]
  }],
  "iss": "https://registry.agentidp.dev",
  "sub": "ag_7xK9m2nP4qRtL8",
  "aud": "https://api.example.com",
  "iat": 1740000000,
  "exp": 1740003600,
  "jti": "tok_unique_nonce_123"
}
```

Figure 2: AIT Payload Example

4.4. Cryptographic Requirements

All Agent Identity Tokens MUST be signed using ES256 (ECDSA with the P-256 curve and SHA-256) as defined in [RFC7518] Section 3.4. This algorithm provides strong security with compact signatures suitable for high-frequency agent-to-service communication.

Agent key pairs are generated during registration and the public key is published via the registry's JWKS endpoint [RFC7517]. Services verify tokens by fetching the public key from the JWKS endpoint, with aggressive caching (RECOMMENDED TTL: 24 hours).

4.5. Token Lifetime

AIT tokens MUST have a maximum lifetime of 24 hours (the difference between exp and iat MUST NOT exceed 86400 seconds). For high-security contexts, a lifetime of 1 hour or less is RECOMMENDED. Short-lived tokens limit the blast radius of key compromise. Agents MUST implement automatic token refresh using their private key.

5. Owner Verification Levels

The protocol defines four verification levels providing increasing assurance about the agent owner's identity. Services can set minimum verification requirements per endpoint.

Level 0 - Unverified Email address confirmed only. Suitable for testing, development, and open-access services.

Level 1 - Email Email and phone verification. Suitable for low-risk public services.

Level 2 - Domain DNS TXT record or well-known file verification at the owner's domain. Suitable for business APIs and partner integrations.

Level 3 - Organisation Know Your Business (KYB) verification including company registration and legal entity verification. Suitable for financial services, healthcare, and regulated industries.

6. Agent Registration

Agent registration is a two-phase process: owner verification (performed once) followed by agent creation (performed per agent). A single verified owner MAY register multiple agents.

6.1. Phase 1: Owner Registration

1. Owner registers at an AgentID-compliant registry with email and authenticating credentials.

2. Owner completes verification to their desired level (email, domain DNS, or KYB document submission).

3. Registry issues an owner_id and management credentials.

6.2. Phase 2: Agent Creation

1. Owner calls POST /v1/agents with agent name, description, and declared capabilities.

2. Registry generates an ES256 key pair, stores the public key, and returns the agent_id, private key (in JWK format per [RFC7517]), and the kid.

3. Agent uses the private key to self-sign AIT tokens. The private key MUST NOT be retained by the registry after delivery to the owner.

Owners MUST immediately store the agent private key in a secure secrets manager. If the private key is lost, the agent MUST be re-registered with a new key pair.

7. Service-Side Access Control

Services consuming AgentID tokens implement access policies based on the claims present in the AIT. The protocol defines three standard access tiers:

Open Any agent may access. No AIT required. Rate limiting is IP-based only. Suitable for public read-only endpoints.

Authenticated A valid AIT from any registered agent is required. Token signature and revocation status are verified. Rate limiting is per agent_id. Suitable for general-purpose APIs.

Permissioned A valid AIT is required AND the agent_id or owner_id MUST be on an explicit allowlist maintained by the service. Rate limiting is per agent_id. Suitable for partner integrations and sensitive data.

7.1. Policy Expression

Services define policies as a set of rules evaluated against AIT claims. The protocol RECOMMENDS the following policy structure:

```
{
  "resource": "/api/bookings",
  "access_tier": "authenticated",
  "min_verification_level": 2,
  "required_capabilities": ["calendar:write"],
  "rate_limit": {
    "requests_per_hour": 100,
    "scope": "agent_id"
  },
  "allowlist": null,
  "denylist": ["ag_known_bad_actor"]
}
```

Figure 3: Access Policy Example

8. Delegation Chain

When Agent A instructs Agent B to perform an action, or when a human user authorises an agent to act on their behalf, the delegation chain is encoded in the AIT so that the receiving service can inspect the full provenance of the request.

8.1. Delegation Chain Structure

The delegation_chain claim is an ordered array of delegation objects, from the original principal (index 0) to the most recent delegator. Each entry records who delegated, when, and with what scope restrictions.

```
{
  "delegation_chain": [
    {
      "principal_type": "user",
      "principal_id": "usr_john_abc",
      "granted_at": "2026-03-01T10:00:00Z",
      "scopes": ["calendar:read", "calendar:write"],
      "evidence": "oauth2:token_exchange"
    },
    {
      "principal_type": "agent",
      "principal_id": "ag_orchestrator_1",
      "granted_at": "2026-03-01T10:00:05Z",
      "scopes": ["calendar:read"],
      "evidence": "ait:delegation"
    }
  ]
}
```

Figure 4: Delegation Chain Example

Each delegation object contains:

principal_type REQUIRED. String. Either "user" or "agent".

principal_id REQUIRED. String. Identifier of the delegating principal.

granted_at REQUIRED. String (datetime). When the delegation was granted.

scopes REQUIRED. Array of strings. The scopes granted in this delegation step.

evidence OPTIONAL. String. Reference to the mechanism used to establish the delegation (e.g., "oauth2:token_exchange" per [RFC8693], "ait:delegation").

8.2. Scope Attenuation

Each link in the delegation chain MUST have scopes that are equal to or a strict subset of the previous link's scopes. A delegate MUST NOT grant more permissions than it received. This is enforced by both the issuing agent (when constructing the token) and the verifying service (when validating claims).

If a verifying service detects a scope attenuation violation, it MUST reject the token with error code AID-009 (DELEGATION_INVALID).

9. Agent Registry

The Agent Registry is the trust anchor of the AgentID ecosystem. It stores agent public keys, owner verification records, and revocation status.

9.1. Registry API

A compliant registry MUST implement the following endpoints:

Method	Endpoint	Description
POST	/v1/owners	Register a new owner
POST	/v1/owners/{id}/verify	Submit verification evidence
POST	/v1/agents	Register a new agent
GET	/v1/agents/{agent_id}	Public agent lookup
GET	/v1/agents/{agent_id}/public-key	Fetch agent public key (JWK)
DELETE	/v1/agents/{agent_id}	Revoke an agent
GET	/.well-known/jwks.json	Registry JWKS endpoint
GET	/v1/agents/{agent_id}/status	Real-time revocation check

Table 1

9.2. Revocation

Agent revocation MUST be immediate. When an owner revokes an agent or the registry suspends one, the agent status is set to "revoked".

* Services implementing real-time checks SHOULD call the status endpoint.

* Services relying on cached public keys SHOULD implement a maximum cache TTL of 1 hour and MUST honour Cache-Control headers returned by the registry.

10. Relationship to OAuth 2.0

AgentID is designed to complement, not replace, OAuth 2.0 [RFC6749]. For services that already use OAuth for user authentication, AgentID adds the agent identity layer. A typical integration pattern is:

1. The user grants an OAuth token to the agent (user delegation).

2. The agent includes the OAuth grant reference in the AIT delegation chain via the "evidence" field (e.g., "oauth2:token_exchange").

3. The service verifies both the agent's identity (AgentID) and the user's authorisation (OAuth).

This layered approach allows services to answer "which agent is this?" (AgentID) independently from "what is this user allowing?" (OAuth), providing complete request provenance.

11. Security Considerations

11.1. Key Management

Agent private keys MUST be stored in hardware security modules (HSMs) or cloud secret managers in production. Key rotation is supported via key versioning: owners can generate a new key pair without changing the agent_id. Both old and new keys are valid during a configurable overlap period (default: 72 hours).

11.2. Replay Prevention

Every AIT includes a unique jti claim. Services SHOULD maintain a jti cache (e.g., with TTL matching token expiry) and reject tokens with previously seen jti values. For stateless verification, the short token lifetime provides an acceptable trade-off.

11.3. Abuse Mitigation

Registries SHOULD implement automated abuse detection monitoring for anomalous patterns. Services can report abusive agent_ids to the registry. A graduated response model is RECOMMENDED: warning, temporary suspension, permanent revocation, owner-level ban.

11.4. Privacy Considerations

Registries store owner identity information subject to applicable data protection regulations (GDPR, etc.). Owner personal data MUST NOT be exposed via the public agent lookup API unless the owner explicitly opts in. The minimum public disclosure is: owner_type, verification_level, and (for Level 2+) verified_domain.

12. IANA Considerations

12.1. Media Type Registration

This document requests registration of the "AIT+jwt" media type subtype in the "JSON Web Token Types" sub-registry of the "JSON Web Token (JWT)" registry.

Type name: AIT+jwt

Subtype name: N/A

Required parameters: N/A

Optional parameters: N/A

Reference: This document

12.2. JWT Claims Registration

This document requests registration of the following claims in the "JSON Web Token Claims" registry:

* agent_id - Globally unique agent identifier
* agent_name - Human-readable agent name
* agent_version - Agent software version
* agent_description - Agent purpose description
* owner_id - Agent owner identifier
* owner_type - Owner type (person or org)
* owner_name - Owner display name
* verification_level - Owner verification level (0-3)
* delegation_chain - Delegation provenance chain

13. References

13.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

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

[RFC7517] Jones, M., "JSON Web Key (JWK)", RFC 7517, May 2015, <https://www.rfc-editor.org/info/rfc7517>.

[RFC7518] Jones, M., "JSON Web Algorithms (JWA)", RFC 7518, May 2015, <https://www.rfc-editor.org/info/rfc7518>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, May 2015, <https://www.rfc-editor.org/info/rfc7519>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

13.2. Informative References

[OpenID.Core] Sakimura, N., Bradley, J., Jones, M., Medeiros, B. D., and C. Mortimore, "OpenID Connect Core 1.0", November 2014, <https://openid.net/specs/openid-connect-core-1_0.html>.

[RFC8693] Jones, M., Nadalin, A., Campbell, B., Bradley, J., and C. Mortimore, "OAuth 2.0 Token Exchange", RFC 8693, January 2020, <https://www.rfc-editor.org/info/rfc8693>.

[SPIFFE] CNCF, "SPIFFE: Secure Production Identity Framework for Everyone", 2024, <https://spiffe.io/docs/latest/spiffe-about/overview/>.

Appendix A. Capability Namespace Convention

Capabilities follow a resource:action format:

form:submit	Submit form data
form:read	Read form structure/schema
calendar:read	Read calendar events
calendar:write	Create/update events
calendar:delete	Delete events
email:send	Send emails
email:read	Read inbox
storage:read	Read files
storage:write	Upload/modify files
payment:initiate	Start a payment
payment:confirm	Confirm a payment

Service providers MAY define custom capabilities using reverse-domain notation (e.g., "com.example.booking:create").

Appendix B. Error Codes

Code	Name	Description
AID-001	INVALID_TOKEN	AIT signature verification failed
AID-002	TOKEN_EXPIRED	AIT has exceeded its exp claim
AID-003	AGENT_REVOKED	Agent has been revoked
AID-004	AGENT_SUSPENDED	Agent is temporarily suspended
AID-005	INSUFFICIENT_VERIFICATION	Owner verification level below minimum
AID-006	CAPABILITY_DENIED	Agent lacks required capability
AID-007	RATE_LIMIT_EXCEEDED	Per-agent rate limit exceeded
AID-008	NOT_ON_ALLOWLIST	Agent not on service allowlist
AID-009	DELEGATION_INVALID	Scope attenuation violated
AID-010	REPLAY_DETECTED	Token jti previously used

Table 2

Acknowledgements

The authors wish to thank the contributors to the AgentID Protocol specification on GitHub, and the broader community working on AI agent identity and authentication standards.

Author's Address

Tim Uzua
GudLab
Email: tim@gudlab.org
URI: https://gudlab.org