TBD
Internet-Draft
Intended status: Informational
Expires: 28 August 2026

J. M. Barney
R. Pioli
D. Watson
Independent

24 February 2026

Contextual Agent Authorization Mesh (CAAM)
draft-barney-caam-00

Abstract

This document specifies the Contextual Agent Authorization Mesh (CAAM), an authorization profile composable with the Agent Registration and Discovery Protocol (ARDP) I-D.pioli-agent-discovery and other discovery mechanisms. CAAM defines the Post-Discovery Authorization Handshake: the runtime authorization layer that governs agent behavior after an agent has been discovered through ARDP but before it is permitted to execute tool calls or delegate authority.

CAAM provides a sidecar-based authorization mediator for enforcing Relationship-Based Access Control (ReBAC), purpose-bound delegation, and cryptographically verifiable intent propagation in Human-to-Agent (H2A) and Agent-to-Agent (A2A) flows. It bridges identity provenance frameworks -- the Interoperability Profiling for Secure Identity in the Enterprise (IPSIE) IPSIE and the Secure Production Identity Framework for Everyone (SPIFFE) SPIFFE -- with the ARDP control plane, leveraging OpenID XAA for coarse-grained delegation, a Knowledge Graph for real-time relationship inference, and the Remote ATtestation procedureS (RATS) architecture RFC9334 for cryptographically verifiable attestation of agent execution environments.

The Session Context Object (SCO) defined herein is encoded as a JSON Web Token (JWT) RFC7519 or a CBOR Web Token (CWT) RFC8392, and carries a new "ctx" (Contextual Assertion) claim that binds the agent’s delegated authority to a specific purpose, session, and trust chain.

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/.

Copyright Notice

Table of Contents

1. Introduction

The evolution of enterprise infrastructure toward autonomous and semi-autonomous agentic systems has introduced a fundamental gap in identity and access management. Traditional security architectures, designed for static workloads and direct human-to-service interactions, are insufficient for managing the non-deterministic, ephemeral, and multi-hop nature of Large Language Model (LLM) agents.

Current security models address identity at two distinct layers: the human user (via OpenID Connect and IPSIE IPSIE) and the machine workload (via SPIFFE SPIFFE). Autonomous agents occupy a hybrid space -- a software workload that possesses its own identity but operates with the delegated authority of a human user or another agent. This hybridity creates a gap where traditional Role-Based Access Control (RBAC) fails to account for the ephemeral and purpose-driven nature of agentic tasks.

Security Dimension	Workload	Human	Agent
Identity Type	SVID	User ID	Composite
Lifespan	Long-lived	Session	Ephemeral
Decision Logic	Deterministic	Human	Probabilistic
Trust Anchor	Attestor	IdP	Discovery+Ctx
Authorization	Static	Role	Contextual

Table 1

As agents move from simple request-response patterns to independent reasoning and tool invocation, the risk of "identity dilution" and "authorization loops" grows. Without a dedicated mesh to manage these interactions, organizations face a choice between over-privileging agents -- expanding the blast radius of potential compromises -- or restricting them to functional uselessness. CAAM resolves this tension by enforcing policy at the most granular level of the agent’s internal reasoning loop.

1.1. Relationship to Existing Standards

CAAM is not intended as a replacement for existing identity, authorization, or attestation protocols. It functions as an orchestration profile -- a connective layer that composes outputs from multiple mature standards into a single, coherent authorization decision tailored to autonomous agent ecosystems.

CAAM occupies three complementary roles:

* GNAP Extension for Non-Deterministic Clients: The Grant Negotiation and Authorization Protocol (GNAP) RFC9635 assumes a client capable of participating in structured negotiation flows. Autonomous agents are non-deterministic clients whose resource requirements emerge dynamically during multi-step reasoning. CAAM extends the GNAP model by introducing the Session Context Object (SCO) as a purpose-bound grant envelope that constrains the agent's evolving resource requests to the boundaries of the original human intent.

* Policy Enforcement Point for RATS: The RATS architecture RFC9334 defines the roles of Attester, Verifier, and Relying Party but does not prescribe where in an application's request path the Attestation Result SHOULD be consumed or how it SHOULD influence fine-grained authorization decisions. The CAAM sidecar serves as a dedicated Policy Enforcement Point (PEP) that ingests RATS Attestation Results and combines them with identity provenance and data sensitivity signals to produce the Contextual Risk Score (CRS) defined in contextual-risk-scoring.

* Trust Framework for Multi-Hop Delegation: OAuth 2.1 and Token Exchange RFC8693 provide mechanisms for single-hop delegation and impersonation. They do not natively address the compound trust problem arising in N-hop agentic chains (User -> Agent A -> Agent B -> Agent C), where each intermediate agent introduces a new trust boundary. CAAM provides Scope Attenuation, Depth-Limited Tokens, and Intent-Binding to ensure that delegated authority degrades gracefully rather than accumulating across hops.

1.2. The Post-Discovery Authorization Handshake

CAAM provides the Post-Discovery Authorization Handshake for I-D.pioli-agent-discovery. The ARDP control plane enables an orchestrating client to discover an agent's endpoint, capabilities, and network address. However, ARDP does not prescribe the authorization protocol that governs the agent's behavior once a session is established.

CAAM fills this gap. After an ARDP RESOLVE operation returns an agent's endpoint, the CAAM sidecar mediates a mutual attestation handshake between the client and the discovered agent before any tool call is permitted. This handshake establishes:

1. Mutual identity verification via SPIFFE SVIDs and RATS Attestation Evidence.

2. A purpose-bound Session Context Object (SCO) that constrains the agent's authority to the specific task.

3. A JIT Scoped Token (via the Ghost Token Pattern) that replaces any long-lived credential with a short-lived, nonce-bound, single-use token.

Without CAAM, an ARDP-discovered agent could be invoked with a static bearer token that carries no purpose binding, no environmental attestation, and no delegation-depth limit -- the failure mode observed in real-world incidents where agents used static bearer tokens without purpose binding.

1.3. The Multi-Hop Intent Binding Problem

The core standardization gap that CAAM addresses is Multi-Hop Intent Binding: the problem of proving, at each hop in a delegation chain, that the current agent's request remains within the semantic bounds of the original user's intent -- without over-privileging the entire chain.

Existing standards address adjacent concerns but leave this problem unresolved:

* OAuth XAA handles coarse-grained identity delegation across application boundaries. It establishes who is delegating what scope to which application. However, XAA does not track whether a sub-delegated agent three hops downstream is still operating within the purpose for which the original grant was issued.

* RATS RFC9334 provides cryptographic assurance of the execution environment's integrity. It establishes where the agent is running and whether that environment is trustworthy. However, RATS Evidence contains no semantic information about the agent's current task or whether that task aligns with the original user's intent.

* CAEP/SSE (Shared Signals and Events) enables reactive session-level signals (e.g., "user logged out," "device compromised"). These signals operate at the granularity of sessions and are

propagated asynchronously. They do not provide the synchronous, per-request intent verification required to gate individual tool calls within a multi-hop agentic flow.

CAAM bridges this gap by binding each request in the delegation chain to a cryptographically signed Intent-Signature derived from the original SCO, verified at every hop by the CAAM sidecar before a JIT Scoped Token is synthesized.

1.3.1. Standardization Gap Analysis

The following table identifies the authorization dimensions relevant to autonomous agent ecosystems and maps the coverage provided by existing standards against the extensions introduced by CAAM.

Dimension	OAuth/GNAP	RATS	CAEP	CAAM
Delegation	1-hop	N/A	N/A	N-hop+Scope Attenuation
Env Trust	N/A	HW/SW	N/A	CRS input
Risk Signals	N/A	N/A	Async	Sync narrowing
Intent	Static	Env-only	None	Per-request
Credentials	Bearer	N/A	N/A	Ghost Token
Inference	N/A	N/A	N/A	Firewall
Granularity	Grant	Attestation	Session	Request
Decision	Scope	Pass/fail	Event	CRS tiers

Table 2

2. 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.

Contextual Mesh: A decentralized, sidecar-based authorization layer

that enforces purpose-bound and context-aware access controls across a network of autonomous agents.

Agent Principal: The composite identity of an agent, incorporating its hardware-attested workload identity (SPIFFE ID) and its delegated human user context (IPSIE claims).

Inference Boundary: A documented and machine-enforceable specification that defines the permissible and prohibited conclusions an agent MAY draw from the combination of authorized data sources.

Session Context Object (SCO): A cryptographically signed, transient data structure encoded as a JWT RFC7519 or CWT RFC8392 that encapsulates the intent, purpose, and provenance of an agentic interaction.

Tool-Call Interception: The process by which an authorization sidecar monitors and regulates the outbound network/API request calls made by an agent. steps of an LLM agent to prevent policy violations.

Narrowed Persona: The reduced capability set advertised by an agent after the Resolver applies contextual filtering based on participant relationships.

Ghost Token: A credential management pattern where the raw delegation token is never exposed to the agent; instead, the mesh synthesizes a short-lived, purpose-scoped replacement at the moment of use.

Contextual Risk Score (CRS): A real-valued score S in the range [0, 1] assigned to every request within the mesh, encoding the combined risk of provenance, environment trust, and data sensitivity.

Intent-Signature: A cryptographic binding between the original user’s purpose (as encoded in the SCO) and each subsequent request in a delegation chain, implemented as a Chained JWS RFC7515.

3. Protocol Overview

3.1. Architectural Foundations

CAAM leverages the existing maturity of identity provenance standards while extending them to meet agentic requirements.

3.1.1. SPIFFE/SPIRE as the Workload Identity Root

The Secure Production Identity Framework for Everyone (SPIFFE) SPIFFE provides a platform-agnostic standard for identifying software workloads. Every agent instance in a CAAM-compliant deployment MUST be registered with a SPIRE server and issued a SPIFFE Verifiable Identity Document (SVID), typically in the form of an X.509 certificate. This SVID provides the agent with a non-repudiable identity that is automatically rotated and cryptographically bound to the execution environment. This machine-level identity is the first layer of the CAAM trust model.

3.1.2. IPSIE as the Human Provenance Layer

While SPIFFE identifies the workload, the IPSIE profiles IPSIE identify the delegating human. When a human user initiates a task requiring agent assistance, their IPSIE-compliant identity token serves as the original source of authority. CAAM uses this provenance to bind the agent’s workload identity to the human’s session identity, creating a composite principal.

3.1.3. ARDP and the Discovery-Authorization Nexus

The Agent Registration and Discovery Protocol (ARDP) I-D.pioli-agent-discovery establishes a control plane for agents to advertise capabilities and network endpoints. CAAM extends ARDP by introducing "AuthZ-at-Discovery," where the security capabilities and policy compliance of an agent are verified before a session is established.

Implementations that support CAAM MUST advertise the following metadata in the ARDP registry: the agent’s SPIFFE trust domain, supported RATS Evidence types, Inference Boundary hash, and policy manifest URI.

3.1.4. RFC 9334 (RATS) Alignment

The CAAM Mesh adopts the RATS architecture RFC9334. Within the mesh, the following roles are defined:

* Attester: The Agent or Sub-Agent providing Evidence. Every agent in a CAAM-compliant deployment MUST be capable of producing RATS Evidence that can be independently verified. Evidence includes runtime state, TPM 2.0 platform quotes, and signed container image digests.

* Verifier: The CAAM Sidecar component that processes Evidence against Endorsements from the SPIRE trust domain. The Verifier produces an Attestation Result that encodes the trustworthiness of the agent's current execution environment.

* Relying Party: The Resource Server that consumes the Attestation Result to authorize the tool call. The Relying Party MUST NOT process raw Evidence; it relies on the Verifier's signed Attestation Result.

3.2. The CAAM Sidecar Model

The CAAM sidecar is the core enforcement mechanism. It is a lightweight proxy that runs alongside the agent workload, intercepting all internal and external communications.

3.2.1. Intercepting the Outbound Tool Calls

The tool-call loop is the fundamental interaction cycle of an autonomous agent:

1. Thought: The LLM generates a plan or tool call.

2. Action: The agent executes the tool call.

3. Observation: The agent receives results and reasons further.

The CAAM sidecar MUST intercept this loop at the transition between Thought and Action. By acting as a semantic gateway, the sidecar analyzes the intent of the tool call before it reaches the network.

3.2.2. Impersonation vs. Delegation

CAAM utilizes Token Exchange RFC8693 to manage relationships in multi-hop chains. It distinguishes between Impersonation (agent acts as the user) and Delegation (agent acts on behalf of the user while maintaining its own identity).

In the CAAM delegation model, the resulting access token MUST include an act (actor) claim that captures the entire lineage:

Step	Token Components	Identity State
User starts	IPSIE Token	sub: User
A calls B	A SVID + Token	sub: User, act: A
B calls RS	B SVID + Prev	act: B { act: A }

Table 3

To maintain least privilege, CAAM mandates Scope Attenuation: each subsequent token in the chain MUST have an equal or smaller scope than its predecessor. Implementations MUST NOT permit scope expansion during delegation.

3.2.3. The Ghost Token Pattern

While OAuth XAA handles coarse delegation between applications, CAAM introduces the Ghost Token Pattern to prevent raw credential exposure:

1. Grant: The User provides a long-lived XAA grant to the Lead Agent, representing the ceiling of the agent’s authority.

2. Shadowing: The Lead Agent MUST NOT possess the raw XAA token. The token resides exclusively in the CAAM Vault. The agent holds only an opaque reference.

3. Synthesis: Upon a tool request, the Mesh synthesizes a JIT Scoped Token that merges the XAA permission ceiling with the current RATS Attestation Evidence and the active SCO. This token MUST be short-lived (less than 60 seconds TTL), bound to a cryptographic nonce, scoped to the exact tool call, and non-replayable.

3.3. The Session Context Object (SCO)

The Session Context Object (SCO) is the central data structure of the CAAM protocol. It MUST be encoded as a JWT RFC7519 for HTTP-based transports or as a CWT RFC8392 for constrained environments.

3.3.1. The "ctx" Claim

This document defines a new JWT/CWT claim:

ctx (Contextual Assertion): A JSON object (or equivalent CBOR map)

that carries the contextual metadata required for CAAM authorization decisions. The "ctx" claim is registered per iana-considerations.

The "ctx" claim MUST contain the following members:

* "purpose": A string describing the human-readable intent of the agentic task. This value is set at origination and MUST NOT be modified by intermediate agents.

* "scope_ceiling": An array of OAuth scope strings representing the maximum authority derived from the original XAA grant.

* "max_hops": An integer indicating the remaining delegation depth. This value MUST be decremented by 1 at each delegation hop. When "max_hops" reaches 0, further delegation MUST be denied.

* "zookie": A consistency token (as defined by the Zanzibar model ZANZIBAR) ensuring that relationship queries reflect the most recent state of the Knowledge Graph.

* "rats_result": A reference to the Verifier’s Attestation Result for the current agent, encoded as a URI or an embedded Evidence structure per RFC9334.

* "crs": The current Contextual Risk Score, a decimal value in the range [0, 1].

The SCO MUST be signed by the originating Identity Provider at creation. At each delegation hop, the delegating agent’s CAAM sidecar MUST append an act claim and sign the extended SCO with its SPIFFE SVID key, forming a Chained JWS RFC7515.

The following is a non-normative example of an SCO JWT payload:

```
{
  "iss": "https://idp.corp.internal",
  "sub": "user:jonathan@corp.com",
  "aud": "agent:finance-bot@corp.internal",
  "iat": 1740355200,
  "exp": 1740358800,
  "act": {
    "sub": "spiffe://corp.internal/agent/fb"
  },
  "ctx": {
    "purpose": "Q4 revenue audit",
    "scope_ceiling": [
      "finance:read",
      "crm:read"
    ],
    "max_hops": 3,
    "zookie": "zk_v1_998877",
    "rats_result":
      "https://verifier.corp.internal/abc",
    "crs": 0.22
  }
}
```

3.3.2. Intent-Signature Mechanism

To solve the Multi-Hop Intent Binding problem, CAAM uses the Intent-Signature: a Chained JWS RFC7515 binding the original user's purpose to each subsequent request.

1. Origination: The CAAM mesh generates an SCO containing the "purpose", "scope_ceiling", and "max_hops" claims. The SCO is signed by the originating Identity Provider.

2. Propagation: At each hop, the delegating agent's sidecar appends its own "act" claim and a "sub_purpose" narrowing the task scope. The sidecar signs the extended SCO with its SPIFFE SVID key, producing a Chained JWS.

3. Verification: Before synthesizing a JIT Scoped Token, the receiving sidecar MUST verify the entire signature chain. Verification confirms that:

   * Each signature is valid against the signer's SPIFFE SVID.

   * The "sub_purpose" at the current hop MUST be validated against the permitted scope boundaries using deterministic policy evaluation (e.g., OPA/Cedar tag matching).

* The requested scope does not exceed the "scope_ceiling" after Scope Attenuation.

* The "max_hops" counter is non-negative.

* The current CRS is within the acceptable remediation tier.

4. Gating: If any verification step fails, the sidecar MUST deny the tool call and MAY trigger a HITL escalation per the CRS remediation tiers.

3.4. Policy Substrate: Knowledge Graphs and ReBAC

To avoid manual policy sprawl, CAAM mandates a Policy Inference Plane that treats the enterprise Knowledge Graph as the source of truth for relationship-based access decisions.

3.4.1. Relationship Ingestion

The mesh MUST ingest relationship triples from existing IAM data (via SCIM) and real-time collaboration signals. These relationships form the edges of the Knowledge Graph and MUST be updated continuously.

3.4.2. Common Ancestor Constraint

For multi-participant sessions, the Resolver MUST satisfy the Common Ancestor Constraint: all participants in a session MUST share a relationship path through the Knowledge Graph to the data silo being accessed. If any participant lacks this path, the agent’s capability set MUST be narrowed accordingly.

3.4.3. Zanzibar Model

Implementation of the Zanzibar model ZANZIBAR (e.g., SpiceDB) is RECOMMENDED. This allows the N-way intersection check to be performed using consistency tokens (e.g., Zanzibar-style ’zookies’) to ensure causal consistency during policy evaluation.

3.5. Protocol Integration

CAAM operates as a secondary control plane between the ARDP discovery layer I-D.pioli-agent-discovery and the execution protocol (e.g., MCP, HTTP, or gRPC).

3.5.1. OpenID XAA Binding

The initial delegation between the Human and the Agent occurs via OpenID XAA:

* The User grants the agent a scope (e.g., "client_data:read").

* This XAA grant is treated as a root node in the Knowledge Graph, providing the agent the potential to access data. The potential is narrowed by real-time context.

* The raw XAA token MUST be stored in the CAAM Vault and MUST NOT be exposed to the agent directly.

3.5.2. OpenID IPSIE Binding and Shared Signals

CAAM utilizes IPSIE IPSIE to manage the Agentic Session:

* The session_id in the ARDP RESOLVE call MUST map to an IPSIE session identifier.

* The Resolver MUST act as a Shared Signals Framework (SSF) receiver. If an SSF event indicates a change in session risk or participant list, the Resolver MUST immediately narrow any agent capabilities that no longer satisfy the Knowledge Graph intersection.

3.5.3. Post-Discovery CAAM Handshake

After a client successfully discovers an agent via the ARDP RESOLVE method, it initiates a separate, subsequent handshake to establish the cryptographic context required for contextual authorization. This is performed via an HTTP POST to the discovered agent’s /caam/authorize endpoint:

```
{
  "method": "POST",
  "endpoint": "/caam/authorize",
  "aid": "agent:finance-bot@corp.internal",
  "context": {
    "session_id": "ipsie-session-2026-v1",
    "xaa_ref": "vault:opaque-ref-001",
    "participants": [
      {
        "format": "email",
        "email": "jonathan@corp.com"
      },
      {
        "format": "email",
        "email": "guest@client2.com"
      }
    ],
    "consistency_token": "zk_v1_998877",
    "rats_evidence": {
      "attester_id":
        "spiffe://corp.internal/agent/fb",
      "evidence_type": "tpm2.0",
      "manifest_hash": "sha256:ab12cd34"
    }
  }
}
```

3.5.4. Capability Fuzzing (Narrowed Persona)

Upon RESOLVE, the Resolver performs contextual capability filtering in two phases:

Discovery-Time Narrowing:

1. Execute a Knowledge Graph traversal to determine whether all participants share a relationship with the target data silos.

2. If a participant lacks access to a data silo, the Resolver MUST remove tools associated with that silo from the agent’s capability advertisement.

3. The agent is discovered with a Narrowed Persona.

Session-Time Enforcement:

1. The sidecar MUST enforce the same filtering on every tool call via outbound request interception.

2. If an SSF event changes the participant list mid-session, the sidecar MUST re-execute the Knowledge Graph intersection and further narrow (MUST NOT expand) the agent's capabilities.

3.5.5. Protocol Phases

The CAAM protocol proceeds through four phases:

1. Discovery Phase: The client searches for an agent via ARDP. The Resolver returns the agent's endpoint along with its CAAM Capability Block (SPIFFE ID, supported policy languages, Inference Boundary hash).

2. Negotiation Phase: The client and the agent's sidecar perform mutual attestation. The client provides its identity proof and the SCO. The sidecar verifies the SCO against IPSIE risk signals and RATS Evidence.

3. Establishment Phase: Upon successful negotiation, a Contextual Session is established. The sidecar issues a JIT Scoped Token via the Ghost Token Pattern.

4. Enforcement Phase: The sidecar intercepts the tool call and performs real-time validation of every tool call against the SCO, Inference Boundary, and CRS threshold.

3.6. Contextual Risk Scoring (CRS)

Every request within the mesh MUST be assigned a Contextual Risk Score (CRS), S in the range [0, 1], calculated by the Verifier.

$$S = w_1 * P + w_2 * E + w_3 * D$$

Where P is Provenance, E is EnvTrust, D is DataSensitivity, $w_1 + w_2 + w_3 = 1$, and weights are configurable per deployment. The factors are:

* Provenance: The strength and freshness of the identity chain, hop count from the original user, SVID validity, and SCO integrity.

* EnvTrust: Trustworthiness of the execution environment per RATS Attestation Evidence -- TPM status, manifest integrity, geographic compliance.

* DataSensitivity: Classification of the target resource (public, internal, confidential, regulated PII).

Remediation Tiers:

CRS Range	Level	Action
S < 0.3	Nominal	JIT Token execution
0.3 <= S < 0.7	Elevated	Step-Up (MFA) REQUIRED
S >= 0.7	Critical	HITL REQUIRED

Table 4

The CRS MUST be recalculated on every tool call. A spike in CRS mid-session (e.g., from an SSF risk event) MUST trigger immediate re-evaluation and potential session downgrade.

3.7. Policy Orchestration

CAAM translates high-level intent into machine-enforceable policies via HEXA HEXA and IDQL:

1. Intent Capture: An administrator defines intent in IDQL.
2. Translation: The HEXA orchestrator translates IDQL to target-specific format (Rego for OPA OPA, Cedar CEDAR).
3. Distribution: Policy bundles are pushed to sidecars.
4. Enforcement: The sidecar evaluates tool calls against bundles, using SCO metadata for context.

Implementations MAY support both OPA and Cedar simultaneously.

4. Security Considerations

This section analyzes the threat landscape specific to autonomous agent ecosystems and describes how CAAM mitigates each identified threat. The analysis follows a defense-in-depth model where multiple independent controls reinforce each other.

4.1. Token Theft and Exfiltration

An attacker who gains access to an agent’s runtime environment (e.g., via container escape or memory dump) may attempt to exfiltrate tokens for use outside the legitimate agent context.

CAAM employs four independent defenses:

1. Ghost Token Pattern: The raw long-lived XAA delegation token never enters the agent's address space. It resides exclusively in the CAAM Vault. An attacker who compromises the agent runtime will find only opaque vault references, not actionable credentials.

2. JIT Token Ephemerality: The JIT Scoped Token synthesized for each tool call MUST have a maximum TTL of 60 seconds and MUST be bound to a single-use cryptographic nonce. After the first presentation, the Relying Party MUST record the nonce in a replay cache and reject any subsequent presentation.

3. Proof-of-Possession via DPoP: Every JIT Scoped Token MUST be sender-constrained using the Demonstrating Proof-of-Possession (DPoP) mechanism defined in RFC9449. The token includes a "jkt" (JWK Thumbprint) confirmation claim that binds it to the agent's SPIFFE SVID private key. At each tool call, the agent MUST present a DPoP proof JWT signed with the corresponding private key. A stolen token is unusable without the private key, which MUST be stored in a hardware-backed keystore (TPM 2.0 or Cloud KMS) and MUST NOT be exportable.

4. Environment Binding: The token embeds a hash of the RATS Attestation Result that was current at synthesis time. An attacker presenting the token from a different environment will fail the attestation verification, and the token MUST be rejected.

These defenses are conjunctive. Even if an attacker overcomes one layer (e.g., extracts a JIT token before it expires), the remaining layers (DPoP binding, environment binding, nonce exhaustion) independently prevent misuse.

4.2. Context Spoofing

Context spoofing occurs when an agent or its operator falsifies environmental signals to obtain a more permissive authorization decision. For example, an agent operating on a public network could claim to be executing within a corporate-office context to bypass geofencing policies.

CAAM mitigates context spoofing through the following requirements:

* Self-Asserted Context Prohibition: The CAAM Verifier MUST NOT accept self-asserted context claims from the agent. All environmental context (location, network posture, platform integrity) MUST be derived from independently verifiable sources: RATS Attestation Evidence RFC9334 from a hardware-rooted Attester, or corroborating signals from the SSF receiver.

* Attestation Evidence Validation: The Verifier MUST validate RATS Evidence against manufacturer Endorsements and the SPIRE trust domain's Reference Values. Evidence that cannot be traced to a trusted Endorser MUST be rejected, and the CRS MUST be set to the Critical tier (S >= 0.7), triggering mandatory HITL escalation.

* Coarse-Grained Context Defaults: To reduce the attack surface for context spoofing, context signals MUST be expressed at the coarsest granularity sufficient for the authorization decision. Implementations MUST use categorical labels (e.g., "corporate", "in-transit", "public") rather than precise measurements (e.g., GPS coordinates) unless the requested scope explicitly demands higher precision. This reduces the number of forgeable dimensions available to an attacker.

* Context Freshness: The Verifier MUST reject Attestation Evidence older than the deployment-configured freshness threshold. Stale evidence MAY indicate that an attacker captured a legitimate attestation from a trusted environment and is replaying it from an untrusted one.

4.3. Proof-of-Possession Binding

Bearer tokens are insufficient for agentic authorization because they can be used by any party that obtains them. CAAM requires that all JIT Scoped Tokens be sender-constrained.

The CAAM sidecar MUST implement the DPoP mechanism RFC9449 as follows:

1. Key Generation: During ARDP registration, each agent's CAAM sidecar generates an asymmetric key pair. The private key MUST be stored in a non-exportable hardware-backed keystore (TPM 2.0, HSM, or Cloud KMS). The public key is published as part of the agent's ARDP Capability Block.

2. Token Binding: When the CAAM Vault synthesizes a JIT Scoped Token, it includes a "cnf" (confirmation) claim containing the "jkt" (JWK Thumbprint) of the agent's DPoP key:

```
{
  "cnf": {
    "jkt": "sha256:0ZcOCORZNYy-..."
  }
}
```

1. Proof Presentation: On every tool call, the agent MUST present a DPoP proof JWT alongside the JIT Scoped Token. The proof JWT is signed with the agent's private key and includes the HTTP method, target URI, and a fresh "jti" to prevent replay.

2. Verification: The Relying Party MUST verify that the DPoP proof JWT was signed by the key whose thumbprint matches the "jkt" in the token's "cnf" claim. If the proof is missing, expired, or signed by a different key, the token MUST be rejected.

This binding ensures that a stolen JIT token cannot be used by an attacker who does not possess the agent's hardware-protected private key.

4.4. Prompt Injection as Privilege Escalation

Prompt injection attacks are a primary threat to autonomous systems. CAAM treats prompt injection as a Semantic Elevation of Privilege attack. The CAAM sidecar MUST remain isolated from the agent's reasoning space (Out-of-Band Policy Enforcement).

Even if an agent's internal state is compromised via prompt injection and it attempts an unauthorized action, the sidecar intercepts the resulting tool call. Because the sidecar's decision logic is deterministic and based on the externally verified SCO, it MUST block the action regardless of the agent's internal belief state.

4.5. Multi-Hop Identity Dilution

In an N-hop chain (User -> Agent A -> Agent B -> Agent C), each hop introduces a new trust boundary. Traditional OAuth 2.0 tokens, if not managed with Scope Attenuation, can allow an agent at the end of the chain to inherit the full permissions of the original user without explicit intent.

CAAM mitigates this by requiring that every token exchange include a "purpose" claim matching the original intent. The sidecar MUST verify that each sub-task was authorized in the original SCO. Combined with Scope Attenuation and Depth-Limited Tokens ("max_hops"), authority MUST degrade gracefully across hops rather than accumulating.

4.6. Confused Deputy Prevention

The Confused Deputy attack in agentic systems occurs when a malicious or compromised Agent A passes a crafted intent to a higher-privileged Agent B, tricking B into performing an action that A is not authorized to request. A related attack is token replay, where a captured JIT token is presented in a different environment or session context.

CAAM prevents both attacks through Contextual Binding -- the cryptographic coupling of every JIT Scoped Token to the specific SCO, environment attestation, DPoP key, and nonce from which it was derived:

* Environment Binding: The JIT token embeds a hash of the RATS Attestation Result that was current at synthesis time. If an attacker replays the token from a different environment, the Relying Party's Attestation Result will not match the embedded hash, and the token MUST be rejected.

* Session Binding: The JIT token includes the SCO's "jti" (JWT ID) and "session_id". A token synthesized for Session X MUST NOT be accepted in Session Y. The sidecar MUST validate that the presented token's session binding matches the active Contextual Session.

* Nonce Binding: Each JIT token is bound to a single-use cryptographic nonce. After the first use, the nonce is recorded in a replay cache. Any subsequent presentation of the same nonce MUST be rejected.

* Sender Binding: The JIT token is bound to the agent's DPoP key via the "cnf"/"jkt" claim per dpop-binding. Even if Agent A obtains a token intended for Agent B, A cannot produce a valid DPoP proof because A does not possess B's private key.

* Intent Binding: The JIT token carries the "purpose" and "sub_purpose" from the SCO. Agent B's sidecar MUST verify that the requested action falls within the stated purpose before executing. A crafted intent from Agent A that requests an out-of-scope action will fail this check even if Agent B has the technical capability to perform it.

These five bindings are conjunctive: all MUST hold for a JIT token to be accepted. The failure of any single binding invalidates the token.

4.7. Data-in-Use Protection (Future Work)

While CAAM provides robust controls for data-at-rest and data-in-transit via access policies, the next frontier is protecting data-in-use during processing by the agent’s LLM or inference engine.

As technology matures, agent workloads MAY be executed within Confidential Computing environments (e.g., secure enclaves such as Intel SGX or AMD SEV). Executing the agent within a Trusted Execution Environment (TEE) protects the "Observation" and reasoning phases from inspection or tampering by a compromised host OS or hypervisor. This provides cryptographic assurance against active data leakage and some forms of prompt injection at the execution layer itself.

Because deploying large, GPU-accelerated LLM workloads within secure enclaves is currently technically difficult, this is identified as an optional, future capability rather than a strict requirement for -00 implementations.

4.8. Policy and Knowledge Graph Integrity

The security of the CAAM mesh relies entirely on the correctness of the authorization policy (e.g., OPA or Cedar) and the accuracy of the Knowledge Graph. Implementations MUST adopt a Zero Trust approach to policy lifecycle management.

This approach SHOULD include:

* Signed Policy Bundles: All policy updates MUST be cryptographically signed and verified by the sidecar before enforcement.

* Verifiable Audit Trails: All mutations to the Knowledge Graph MUST be appended to a tamper- evident log.

* Policy-as-Code Pipelines: Changes to the ruleset MUST pass through automated CI/CD pipelines with mandatory security reviews and conflict-detection tests to prevent the introduction of overly permissive rules.

4.9. Agent Supply Chain Security

While RATS secures the agent’s runtime environment, a comprehensive Zero Trust model MUST also scrutinize the agent’s software provenance.

The CAAM framework SHOULD be extended to verify the software supply chain of discovered agents. Before initiating the Post-Discovery Handshake, the sidecar MAY require the agent (or the ARDP Registrar) to provide a signed Software Bill of Materials (SBOM). The sidecar can then verify that the agent's components, including the specific LLM model weights and inference dependencies, contain no known critical vulnerabilities before allowing execution.

4.10. Formal Threat Model

The following threats are specific to multi-agent orchestration. For each threat, the table identifies the attack vector, the primary CAAM mitigation, and the relevant section of this document.

Threat	Vector	Mitigation	Ref
Token Theft	Runtime exfil	DPoP+Vault+60s	token-theft
Context Spoof	Fake env	RATS+coarse ctx	context-spoofing
Confused Deputy	Intent A->B	5-way binding	confused-deputy
Signal Spoof	Fake TEE/geo	TPM Evidence	context-spoofing
Deleg Loop	Circular chain	max_hops+cycle	sco-definition
Token Replay	Captured JIT	Nonce+DPoP	dpop-binding
Infer Bypass	Cross-source	Firewall+DP	minimal-disclosure
Prompt Inject	Manipulated LLM	OOB enforce	N/A
Data-in-Use Exfil	Host OS/ Hyper	TEE (Future/ Opt)	data-in-use-protection
Policy Tampering	Malicious Rules	Signed Bundles	policy-integrity
Vulnerable Agent	Compromised Dep	Signed SBOM	supply-chain-security

Table 5

5. Privacy Considerations

5.1. Inference Isolation

A unique challenge in agentic security is Fuzzy Search Leakage: an agent with authorized access to multiple datasets may combine them in its internal memory to draw unauthorized inferences.

CAAM implements a Contextual Firewall that enforces isolation within the agent's reasoning context. If an agent retrieves data from Source A, the sidecar MUST restrict access to Source B for the duration of that sub-task if the combination is flagged as a high-risk inference vector in the Inference Boundary.

5.2. Minimal Disclosure

Context Providers -- entities that supply environmental, behavioral, or relational signals to the CAAM mesh -- MUST adhere to the principle of Minimal Disclosure.

5.2.1. Coarse-Grained Context by Default

All context signals MUST be expressed at the coarsest granularity sufficient for the authorization decision by default. The following table illustrates the REQUIRED default granularities:

Signal	Coarse (Default)	Fine (Opt-in)
Location	"corporate" / "public"	Country code
Network	"trusted" / "untrusted"	CIDR block
Time	"business-hours" / "off"	ISO 8601
Platform	"attested" / "unattested"	PCR values

Table 6

A Context Provider MUST NOT disclose fine-grained context unless both of the following conditions are met:

1. The requested OAuth scope explicitly requires the finer granularity (e.g., a geofencing scope that specifies country-level).

2. The current CRS is in the Nominal tier ($S < 0.3$). At Elevated or Critical CRS, the Verifier MUST reject fine-grained context requests and fall back to coarse defaults to reduce the information surface.

5.2.2. Additional Disclosure Constraints

* The SCO's "ctx" claim MUST NOT carry raw sensor data, biometric measurements, or personally identifiable information (PII) unless the requested scope explicitly requires it and the CRS permits it.

* Implementations SHOULD support selective disclosure mechanisms (e.g., SD-JWT) to enable verifiers to request only the claims they require from the SCO.

* When RATS Evidence is included in the SCO, the Verifier SHOULD produce an Attestation Result that abstracts the raw Evidence into a trust score or categorical assessment, ensuring that detailed platform measurements are not propagated beyond the Verifier.

* The CAAM mesh MUST NOT log or persist the full contents of the "ctx" claim beyond the lifetime of the Contextual Session. Implementations SHOULD retain only the CRS value and remediation tier for audit purposes.

The principle of Minimal Disclosure ensures that the CAAM mesh does not itself become a vector for privacy leakage by aggregating and propagating more context than is necessary for each authorization decision.

6. IANA Considerations

This section requests registration of claims, parameters, and a new registry with IANA.

6.1. JSON Web Token Claims Registration

This specification defines one new claim for registration in the "JSON Web Token Claims" registry established by Section 10.1 of RFC7519.

6.1.1. Registry Contents

Claim Name: ctx

Claim Description: CAAM Contextual Assertion. A JSON object containing purpose, scope ceiling, delegation depth, consistency token, attestation result reference, and contextual risk score for agentic authorization decisions.

Change Controller: IETF

Specification Document(s): sco-definition of this document

Claim Value Type: JSON object

The "ctx" claim is used within the Session Context Object (SCO) defined in sco-definition. Its value is a JSON object with the following members:

* "purpose" (string, REQUIRED): Human-readable intent of the agentic task. MUST NOT be modified by intermediate agents.

* "scope_ceiling" (array of strings, REQUIRED): Maximum OAuth scope derived from the original delegation grant.

* "max_hops" (integer, REQUIRED): Remaining delegation depth. Decremented by 1 at each hop.

* "zookie" (string, OPTIONAL): Consistency token for Knowledge Graph queries.

* "rats_result" (string, OPTIONAL): URI reference to the Verifier's Attestation Result per RFC9334.

* "crs" (number, REQUIRED): Contextual Risk Score in the range [0, 1].

The following is a non-normative example of the "ctx" claim value:

```
{
  "purpose": "Q4 revenue audit",
  "scope_ceiling": [
    "finance:read",
    "crm:read"
  ],
  "max_hops": 3,
  "zookie": "zk_v1_998877",
  "rats_result":
    "https://verifier.corp.internal/abc",
  "crs": 0.22
}
```

6.2. OAuth Parameters Registration

IANA is requested to register the following entry in the "OAuth Parameters" registry established by RFC 6749:

Parameter Name: caam

Parameter Usage Location: authorization request, token request

Change Controller: IETF

Specification Document(s): This document

The "caam" parameter carries a reference to the Session Context Object that MUST be validated by the authorization server before issuing tokens in a CAAM-compliant deployment.

6.3. OAuth Token Introspection Response

IANA is requested to register the following entry in the "OAuth Token Introspection Response" registry:

Response Parameter: ctx

Change Controller: IETF

Specification Document(s): sco-definition of this document

When a CAAM-compliant authorization server responds to an introspection request for a JIT Scoped Token, it SHOULD include the "ctx" claim to enable the Relying Party to perform contextual validation.

6.4. CAAM Agent Discovery Metadata Registry

IANA is requested to create a new registry titled "CAAM Agent Discovery Security Metadata" under the "CAAM" registry group.

6.4.1. Registration Policy

New entries require Specification Required per RFC 8126.

6.4.2. Initial Registry Contents

Attribute	Description	Req
caam_v1_supported	CAAM support	REQUIRED
sc_object_hash	SCO hash	OPTIONAL
inf_boundary_v1	Inference limits	RECOMMENDED
authz_policy_uri	Policy URI	REQUIRED
spiffe_trust_domain	Trust domain	REQUIRED

rats_evidence_type	Evidence type	RECOMMENDED
crs_threshold	Max CRS w/o MFA	OPTIONAL
max_delegation_hops	Max depth	RECOMMENDED
dpop_jwk_uri	DPoP public key	REQUIRED

Table 7

6.4.3. ARDP Registry Extensions

Additionally, this document requests the following new attributes for the ARDP registry defined by I-D.pioli-agent-discovery:

* Policy Compliance Hash: A cryptographic hash of the agent's active policy set, allowing clients to verify governance.

* Inference Boundary Declaration: A formal specification of the agent's semantic limits.

* Trust Anchor Reference: A pointer to the authority responsible for the agent's identity and behavioral attestation.

7. References

7.1. Normative References

[I-D.pioli-agent-discovery]
Pioli, R., "Agent Registration and Discovery Protocol (ARDP)", 2026, <https://datatracker.ietf.org/doc/draft-pioli-agent-discovery/>.

[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>.

[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>.

[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>.

[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>.

[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>.

[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>.

[RFC9334] Birkholz, H., Thaler, D., Richardson, M., Smith, N., and W. Pan, "Remote ATtestation procedureS (RATS) Architecture", RFC 9334, DOI 10.17487/RFC9334, January 2023, <https://www.rfc-editor.org/rfc/rfc9334>.

[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>.

7.2. Informative References

[CEDAR] Amazon Web Services, "Cedar Policy Language", n.d., <https://www.cedarpolicy.com/>.

[HEXA] "Hexa Policy Orchestrator", n.d., <https://github.com/hexa-org/policy-orchestrator>.

[IPSIE] OpenID Foundation, "Interoperability Profiling for Secure Identity in the Enterprise", n.d., <https://openid.net/wg/ipsie/>.

[OPA] "Open Policy Agent", n.d., <https://www.openpolicyagent.org/>.

[RFC9635] Richer, J., Ed. and F. Imbault, "Grant Negotiation and Authorization Protocol (GNAP)", RFC 9635, DOI 10.17487/RFC9635, October 2024, <https://www.rfc-editor.org/rfc/rfc9635>.

[SPIFFE] "Secure Production Identity Framework for Everyone", n.d., <https://spiffe.io/>.

[ZANZIBAR] Pang, R. and R. Lanber, "Zanzibar: Google’s Consistent, Global Authorization System", USENIX ATC 2019, 2019.

Appendix A. Appendix A. Mathematical Models for Inference Isolation

Let X be the agent's internal state, S_1 and S_2 be two data sources, and B be the inference boundary defined by policy. The sidecar ensures that for any inference I drawn by the agent:

$$P(I \mid S_1, S_2, X) = P(I \mid Auth(S_1, S_2), X)$$

Where the Auth function represents the set of permissible inferences under boundary B. If the combination is unauthorized, the sidecar applies a privacy-preserving transformation T such that the mutual information within the scratchpad is minimized:

$$I(T(S_1); T(S_2)) <= epsilon$$

Where epsilon is the privacy budget defined by enterprise policy.

Appendix B. Acknowledgments

The authors thank the IETF community and the contributors to the Agent Registration and Discovery Protocol for foundational work that informed this specification.

Appendix C. Document History

C.1. draft-barney-caam-00

* Initial submission.

Authors' Addresses

Jonathan M. Barney
Independent
United States of America
Email: jonathan.barney@gmail.com
URI: https://github.com/jmbarney5/Contextual-Agent-Authorization-Mesh

Roberto Pioli
Independent
Italy
Email: roberto.pioli@gmail.com
URI: https://github.com/roberto-pioli/agent-registration-discovery

Darron Watson
Independent
United States of America
Email: drwatson0874@gmail.com