Comments on draft-klrc-aiagent-auth-00

Delegation Chain Mechanics Gaps and Chain Splicing Vulnerability

Draft: draft-klrc-aiagent-auth-00

Authors: Pieter Kasselman, Rifaat Shekh-Yusef, Brian Campbell, David Waite

Comment by: Kieran Sweeney

Contact: kieran@kierans.net

Date: March 11, 2026

Submitted to: wimse@ietf.org, CC oauth@ietf.org

Summary

This comment provides detailed, constructive feedback on draft-klrc-aiagent-auth-00. I am developing a companion draft addressing delegation chain mechanics and have implementation experience with the patterns described in this framework. This comment aims to (a) affirm the core framing, (b) identify specific gaps in delegation chain semantics that warrant focused specification work, (c) flag a recently disclosed vulnerability relevant to the delegation sections, and (d) surface an architectural failure mode not addressed in the current security considerations.

1. What the Draft Gets Right

The decision to treat agents as workloads (Section 3) and anchor agent identity in WIMSE/SPIFFE is the correct foundation. The AIMS model (Section 4) provides a coherent layered stack that practitioners and standards authors can reason about systematically.

The use of existing, proven standards rather than inventing new protocols is the right call. The OAuth 2.0 delegation framework (Sections 10.1-10.3), WIMSE Proof Tokens (Section 9.2.1), and the Transaction Tokens approach (Section 10.4) are well-chosen building blocks.

The acknowledgment of the Human-in-the-Loop requirement (Section 10.6) and CIBA as a mechanism for asynchronous user consent is important and often overlooked in agent-focused work.

The document's explicit goal -- to identify gaps and guide future standardization -- is the right framing. This comment is written in that spirit.

2. The Core Gap: Delegation Chain Protocol Mechanics

Section 10.3 describes how agents receive credentials via three paths: user delegation, own authorization, and agent-to-agent access. For the agent-to-agent case (Section 10.3.3), the draft correctly identifies OAuth 2.0 Token Exchange (RFC 8693) as the relevant mechanism.

However, the delegation mechanics are left underspecified at three distinct layers, and existing authorization policy engines each address only one of them:

Layer 1: Per-action authorization (Cedar model). Cedar evaluates permit/deny decisions for each agent action before execution. It is deterministic and fast. However, Cedar is stateless -- each delegation hop is evaluated independently against policy, with no native model of delegation chains as a coherent structure. Revoking an intermediate agent's access requires the application layer to manually cascade updates.

Layer 2: Per-session authorization (Conditional Access / CAE model). Microsoft Entra's Continuous Access Evaluation gates access at the token-acquisition level. It handles critical revocation events with near-real-time propagation -- though measured latency can reach 15 minutes for event propagation and up to one day for policy changes. This is session-granular: it cannot revoke a specific tool-call authorization within an active session, and it has no awareness of multi-hop chain topology.

Layer 3: Per-chain authorization (FGA / OpenFGA model). OpenFGA models delegation chains as first-class relationship graph objects. It is the only major authorization engine that natively represents multi-hop delegation as a coherent structure. However, revocation via tuple deletion only cascades structurally in HIGHER_CONSISTENCY mode, which bypasses cache with significant performance impact. In the default MINIMIZE_LATENCY mode, revoked access may persist until TTL expires.

The critical finding from production deployments: no single engine provides deterministic per-action policy, risk-adaptive session gating, and chain-aware relationship authorization simultaneously. Production systems likely need Cedar or FGA for fine-grained enforcement layered with Conditional Access for coarse risk-gating -- but no standard composition pattern exists.

This means the following specific gaps exist in both RFC 8693 and this draft:

1. No chain verification algorithm. When Agent A delegates to Agent B which delegates to Agent C, there is no standard mechanism to verify that the resulting token represents a valid, unbroken chain -- that each step was authorized by the entity above it.

2. No scope attenuation rules. There is no normative requirement that Agent B's authority be a subset of or equal to Agent A's. An agent can request the same scopes as the original user grant, violating least-privilege for multi-hop scenarios.

3. No revocation propagation standard. RFC 8693 defers revocation to implementations. Neither CAE's 15-minute propagation window nor FGA's cache-dependent consistency is acceptable for machine-speed agents where a compromised token can exfiltrate data before revocation propagates.

4. No token format for delegation chains. The act and may_act claims (RFC 8693 Section 4) are underspecified for multi-hop scenarios. The may_act claim is optional; there is no guidance on whether it is normative for resource servers acting on delegated tokens.

5. No cross-IdP delegation semantics. Section 10.5 notes that cross-domain federation is a consideration but leaves the mechanism unspecified. When Agent A (authenticated via Okta) delegates to Agent B (authenticated via Entra), there is no interoperability protocol.

These gaps are correctly out-of-scope for a framework document. They are flagged here to motivate the companion work described in Section 6.

3. Chain Splicing Vulnerability in RFC 8693

On February 26, 2026, Arnav Chhaya posted to the OAuth WG mailing list documenting a delegation chain splicing vulnerability in RFC 8693. This is directly relevant to Sections 10.1-10.4 of this draft.

The attack: RFC 8693 Section 2.1-2.2 requires no cross-validation between the subject_token and the actor_token presented to a Security Token Service. A compromised intermediary can obtain a valid subject_token from delegation context A (e.g., User -> Agent A), obtain a valid actor_token from a completely different delegation context B (a different user's flow), present both to an STS which validates each independently, and receive a properly-signed token asserting a delegation relationship that never occurred.

The root cause: the may_act claim (RFC 8693 Section 4.4) validates actor identity but not delegation context provenance. There is no binding between the subject_token and actor_token that proves they originated from the same delegation event.

Relevance to this draft: Section 10.4 recommends Transaction Tokens as a risk reduction mechanism. Transaction Tokens address request-scoped context propagation but do not mitigate this splicing vulnerability, because the forgeable claim is in the token issuance step, not the request propagation step.

The security considerations section (Section 14) should explicitly address this vulnerability and recommend that implementations bind subject_token and actor_token to the same delegation context via a common delegation_id claim, make may_act processing normative at resource servers, and implement per-step signed delegation artifacts for multi-hop chains.

4. Tool-to-Service Access and the Credential Provisioning Gap

Section 10.7 describes the scenario where an agent invokes a tool that itself needs to access a protected resource on behalf of the user. The draft notes that this requires the tool to hold appropriate credentials but does not specify how those credentials are provisioned, managed, or attenuated.

This gap has documented security impact. The August 2025 Drift/Salesloft breach demonstrated OAuth tokens issued months earlier -- still valid due to no revocation process -- being used to infiltrate 700+ organizations. Current deployments handle tool credential provisioning either via static secrets in tool configuration (long-lived tokens with no scope limits and no centralized revocation) or ad-hoc OAuth flows per tool (no cross-tool scope visibility or centralized revocation). Neither approach scales to the multi-tool, multi-agent environments this framework is designed for.

The missing component is a normative specification for how tools request scoped, time-limited credentials on behalf of users, with the user's pre-authorization captured in a revocable delegation record. This is distinct from the agent authentication mechanisms in this draft -- it concerns how agents and tools obtain third-party credentials without those credentials being embedded in

prompts or configuration.

5. Architectural Failure Mode: IdP Unavailability

The Enterprise-Managed Authorization extension routes all agent-to-tool authorization through the enterprise IdP. This creates a critical single point of failure not addressed in the current security considerations.

The failure mode: Every step of the authorization flow requires the IdP to be reachable. If the IdP is unavailable, no new agent-to-tool connections can be established. Unlike human users, who can wait for systems to recover, agents running multi-step workflows will fail at the next authorization check. For agents using short-lived tokens (5-15 minutes, as recommended), an IdP outage exceeding the token TTL cascades to every agent in the organization simultaneously.

Multi-tenant IdP architectures amplify this: a single provider outage affects all tenants. The blast radius is qualitatively different from traditional SSO outages because agents operate at machine speed and cannot self-triage.

Section 14 should include explicit guidance on IdP availability requirements and define at minimum a fallback ordering when enterprise-managed authorization is unavailable. Potential mitigations -- token grace period extensions, fallback to standard authorization, multi-IdP federation, or IdP-independent credential caching -- each require spec-level definition to be deployed interoperably.

6. Potential Work Items for the Working Group

6.1 Delegation Chain Semantics (High Priority). A focused draft profiling RFC 8693 for multi-hop agent delegation: delegation chain representation, chain verification algorithm, mandatory attenuation semantics, cryptographic context binding (splicing resistance), and token exchange requirements. The author intends to submit a companion draft (draft-sweeney-oauth-agent-delegation-00) addressing this item.

6.2 Credential Broker Protocol (High Priority). A protocol for how agents and tools request scoped, time-limited credentials on behalf of users, with centralized delegation records and revocation. Addresses the tool-to-service access gap in Section 10.7.

6.3 RFC 8693 Security Addendum (High Priority). An explicit security advisory or BCP addendum addressing the delegation chain splicing vulnerability disclosed February 2026.

6.4 Enterprise Authorization Resilience Profile (Medium Priority). A profile defining fallback behavior when enterprise IdP is unavailable during agent authorization. Should specify minimum token grace periods, fallback authorization modes, and observability requirements.

6.5 Agent Context Claims (Medium Priority). A lightweight JWT extension for communicating AI agent context in OAuth tokens -- agent_id, model_id, agent_version, agent_policy_hash -- enabling model-aware authorization and audit-ready logging.

7. Editorial Comments

Section 5 (Agent Identifier): The requirement that an agent "MUST be assigned exactly one WIMSE identifier" creates challenges for agents operating across multiple cloud trust domains. Cloud-native workload identity federation (AWS IRSA, GKE Workload Identity, Azure Workload Identity) all issue OIDC tokens with different claim requirements, making a single WIMSE identifier non-trivially portable across cloud boundaries. A profile covering multi-cloud agent identity scenarios would be worth specifying.

Section 6 (Agent Credentials): The note that "static API keys are an antipattern" is correct and important. Consider making this a formal MUST NOT in the security considerations rather than a note. A normative prohibition would signal clearly to implementers and create leverage for the credential broker work described in item 6.2.

Section 10.6 (Human in the Loop): The description of CIBA for asynchronous consent is good. It would be worth noting that CIBA has latency constraints (typically 2-5 seconds for push notification round-trips) that may be prohibitive for real-time agent interactions. Implementations may need a "consent budget" pattern: upfront user authorization covering a bounded set of anticipated actions, rather than per-action CIBA challenges. This also connects to the revocation problem: pre-authorized consent budgets need a revocation mechanism if user intent changes mid-task.

Kieran Sweeney -- kieran@kierans.net