

Execution Context Tokens for Distributed Agentic Workflows: A Data-Driven Design Grounded in 260 IETF Internet-Drafts

Christian Nennemann
Independent Researcher
ietf@nennemann.de

February 2026

Abstract

The Internet Engineering Task Force (IETF) has seen 260 Internet-Drafts addressing AI agents and autonomous systems between June 2025 and February 2026—a $36\times$ increase in monthly submissions over nine months. Yet a quantitative analysis of this corpus reveals a striking gap: while 98 drafts address agent *identity*, 92 address agent-to-agent *communication*, and 60 address *authorization*, effectively zero proposals provide a standard format for recording what agents *actually did*. We introduce Execution Context Tokens (ECTs), a JWT-based extension to the WIMSE (Workload Identity in Multi-System Environments) architecture that records task execution across distributed agentic workflows. Each ECT is a cryptographically signed record documenting a single task, with predecessor tasks linked through a directed acyclic graph (DAG). Using embedding-based similarity analysis, LLM-assisted multi-dimensional rating, and automated gap detection across the full 260-draft landscape, we demonstrate that ECTs address three identified gaps—agent behavior verification (critical), error recovery (critical), and data provenance (medium)—while maintaining low overlap (2/5) with existing proposals. A head-to-head comparison against eight competing drafts shows that ECTs are the only proposal combining DAG-based workflow modeling, cryptographic input/output integrity, and native WIMSE integration.

Keywords: execution context, agentic workflows, WIMSE, JWT, directed acyclic graph, IETF standardization, landscape analysis

1 Introduction

The rapid deployment of autonomous AI agents—software systems that can independently plan, execute tasks, and collaborate with other agents—has created urgent demand for infrastructure standards. The IETF, as the primary venue for Internet protocol standardization, has responded with unprecedented speed: between June 2025 and February 2026, submissions grew from 2 AI-related Internet-Drafts per month to 72, a $36\times$ increase in nine months.

However, this activity has not been evenly distributed. A quantitative survey of 260 drafts reveals that the community has focused heavily on three questions:

1. **Who is this agent?** (Identity/authentication: 98 drafts)
2. **What may this agent do?** (Authorization/policy: 60 drafts)
3. **How do agents talk?** (Communication protocols: 92 drafts)

A fourth, equally critical question remains effectively unanswered:

What did this agent actually do?

Of 1,262 technical ideas extracted from the corpus, only 6 address error recovery in agentic workflows, 52 partially touch behavior verification, and zero provide hash-based data lineage tracking. This paper addresses this gap with two contributions:

- **Execution Context Tokens (ECTs):** A JWT-based extension to the WIMSE architecture [17] that records task execution as cryptographically signed, DAG-linked tokens. ECTs answer the “what did it do?” question with verifiable execution records.
- **Quantitative landscape positioning:** Using a purpose-built analysis pipeline (260 drafts, 33,670 pairwise similarity computations, 12 identified gaps), we demonstrate that ECTs occupy a genuinely novel niche—overlap score 2/5, composite quality 4.0/5—and are the only proposal simultaneously addressing all three execution-layer gaps.

The remainder of this paper is organized as follows. Section 2 provides background on the WIMSE architecture and related standardization efforts. Section 3 presents the quantitative landscape analysis that motivates ECTs. Section 4 describes the ECT specification. Section 5 provides a head-to-head comparison against eight competing proposals. Section 6 maps ECT mechanisms to the 12 identified landscape gaps. Section 7 discusses limitations and adoption considerations. Sections 8–9 cover related work and conclusions.

2 Background

2.1 IETF AI Agent Standardization

The IETF’s AI-related work spans multiple working groups and areas. The WIMSE (Workload Identity in Multi-System Environments) working group focuses on cross-system identity for workloads and services. The SPICE (Secure Patterns for Internet CrEentials) group addresses verifiable credentials. The RATS (Remote ATtestation procedureS) group develops attestation evidence formats. The OAuth working group has produced several agent-specific extensions for delegated authorization.

These groups collectively address the identity and authorization layers but leave execution accountability to individual implementations. No IETF working group currently has a charter covering execution record formats for agentic workflows.

2.2 WIMSE Architecture

The WIMSE architecture [17] provides identity infrastructure for workloads operating across trust domains. Its core components are:

- **Workload Identity Token (WIT):** A JWT asserting the identity of a workload, typically using SPIFFE IDs [15] as subject identifiers.
- **Workload Proof Token (WPT):** A proof-of-possession token binding a request to a specific WIT, preventing token theft.
- **Trust Domains:** Administrative boundaries within which workload identities are issued and recognized. Cross-domain federation follows established patterns.

WIMSE answers “who is this workload?” and “can it prove its identity?” but does not answer “what did it do?” ECTs extend WIMSE to record execution context, reusing its signing infrastructure and trust model.

2.3 The Identity–Authorization–Execution Stack

We identify three layers of agent accountability:

1. **Identity Layer:** Establishes *who* the agent is (WIMSE WIT, X.509, DID).
2. **Authorization Layer:** Determines *what* the agent may do (OAuth tokens, capability tokens, policy frameworks).

3. **Execution Layer:** Records *what* the agent actually did (execution context, audit trails, provenance).

The IETF landscape heavily invests in layers 1 and 2 ($98 + 60 = 158$ drafts) while layer 3 remains effectively vacant. This asymmetry means agents can be authenticated and authorized but their actual behavior cannot be independently audited—a significant gap for production deployments.

3 Landscape Analysis: Motivating Evidence

To ground the ECT design in empirical evidence, we conducted a systematic analysis of 260 IETF Internet-Drafts related to AI agents, published between June 2025 and February 2026. The methodology, dataset, and analysis toolkit are described in a companion paper [12] and released as open source.

3.1 Corpus Overview

Table 1 summarizes the dataset.

Table 1: Corpus summary statistics.

Metric	Value
Internet-Drafts analyzed	260
Unique authors	403
Author–draft relationships	742
Technical ideas extracted	1,262
Semantic categories	19
Pairwise similarity pairs	33,670
Identified landscape gaps	12
Time span	Jun 2025 – Feb 2026

Each draft was rated on five dimensions (novelty, maturity, overlap, momentum, relevance; scale 1–5) using LLM-assisted analysis (Anthropic Claude Sonnet 4), embedded using a local model (nomic-embed-text via Ollama), and processed for technical idea extraction.

3.2 Category Distribution and the Safety Deficit

LLM-assisted classification assigned each draft to one or more of 19 categories. Table 2 organizes the top categories into three accountability tiers.

Table 2: Category distribution organized by accountability tier.

Tier	Category	Drafts	Avg Score
Infrastructure	Data formats / interop	102	3.3
Infrastructure	A2A protocols	92	3.4
Infrastructure	Agent discovery / reg	57	3.5
Authorization	Agent identity / auth	98	3.4
Authorization	Policy / governance	60	3.3
Authorization	Human-agent interaction	22	3.3
Accountability	AI safety / alignment	36	3.4
Accountability	Autonomous netops	60	3.3

The infrastructure and authorization tiers collectively account for 431 category assignments (noting that multi-assignment is possible). The accountability tier, despite including safety-critical concerns, accounts for only 96—a ratio of roughly **4:1**. Within the accountability tier, only 36 drafts address AI safety/alignment, the category most relevant to execution auditing.

3.3 The Overlap Problem

Pairwise cosine similarity analysis across 33,670 draft pairs reveals significant redundancy:

- 56 pairs (0.2%) exceed 0.90 similarity (near-duplicate)
- 344 pairs (1.0%) exceed 0.85 (highly similar)
- 2,668 pairs (7.9%) exceed 0.80 (significantly overlapping)

The redundancy concentrates in crowded areas. The OAuth-for-agents cluster contains 13 drafts proposing variations of delegated agent authorization. The agent-gateway cluster contains 10 drafts. Meanwhile, **zero** dedicated drafts exist for execution record formats prior to ECTs. The community is duplicating effort in well-explored spaces while leaving critical gaps unfilled.

3.4 Gaps Relevant to Execution Tracking

Automated gap analysis identified 12 under-addressed areas (3 critical, 6 high, 3 medium severity). Three directly concern the execution layer:

Table 3: Execution-layer gaps in the IETF landscape.

Gap	Severity	Ideas ^a	Problem
Behavior Verification	Critical	52	No mechanisms to verify agents behave per declared policies. 36 safety drafts vs 260 total.
Error Recovery	Critical	6	No standards for cascading failure recovery. Only 6 of 1,262 ideas address this.
Data Provenance	Medium	0 ^b	No hash-based data lineage tracking despite 102 data format drafts.

^aNumber of extracted ideas (of 1,262 total) that partially address the gap.

^b79 ideas mention “data” broadly, but none implement cryptographic lineage.

The error recovery gap is particularly stark: of 1,262 technical ideas extracted from the entire corpus, only **6** touch error handling in agentic workflows. The two proposals that do exist address the problem partially but lack a common execution record format for systematic post-mortem analysis.

4 Execution Context Token Specification

4.1 Design Principles

ECTs are guided by four principles:

1. **Records, not permissions.** ECTs document what an agent did, not what it may do. They complement rather than replace authorization tokens.
2. **DAG, not chain.** Real-world workflows involve parallel execution and convergence (fan-out/fan-in). A directed acyclic graph captures this; a linear chain cannot.
3. **WIMSE-native.** ECTs reuse WIMSE’s signing keys, identity model, and trust domains. No additional key infrastructure is required.
4. **Minimal mandatory claims.** Only three execution-specific claims are required (`jti`, `exec_act`, `par`), keeping the base token compact.

4.2 Token Structure

An ECT is a JWT [7] using JWS Compact Serialization [6]. It comprises a JOSE header, standard JWT claims, and execution-specific claims.

4.2.1 JOSE Header

Listing 1: ECT JOSE Header.

```
{
  "alg": "ES256",
  "typ": "wimse-exec+jwt",
  "kid": "agent-a-key-id-123"
}
```

The `typ` parameter `wimse-exec+jwt` distinguishes ECTs from other JWTs. The `kid` references the agent's WIMSE public key, binding the ECT to the agent's verified identity.

4.2.2 Claims

Listing 2: Complete ECT payload example.

```
{
  "iss": "spiffe://trust-domain.example/agent/payment-processor",
  "aud": "spiffe://trust-domain.example/agent/compliance-check",
  "iat": 1740600000,
  "exp": 1740600900,
  "jti": "f47ac10b-58cc-4372-a567-0e02b2c3d479",
  "wid": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
  "exec_act": "process_payment",
  "par": ["c9d2e3f4-a5b6-7890-cdef-123456789012",
          "d8e9f0a1-b2c3-4567-89ab-cdef01234567"],
  "inp_hash": "base64url(SHA-256(input_data))",
  "out_hash": "base64url(SHA-256(output_data))",
  "ext": {
    "com.example.amount_range": "1000-5000"
  }
}
```

The execution-specific claims are:

- **exec_act**: A structured identifier for the action performed (e.g., `process_payment`, `validate_input`).
- **par**: An array of parent task identifiers (`jti` values), forming the DAG edges. An empty array indicates a root task.
- **wid**: Optional workflow identifier grouping related ECTs.
- **inp_hash** / **out_hash**: SHA-256 hashes of input and output data, providing integrity without exposing content.
- **ext**: Optional extension object (max 4,096 bytes, max 5 nesting levels) using reverse-domain notation.

4.3 DAG Construction and Validation

ECTs form a directed acyclic graph through the `par` claim. Each `par` entry references the `jti` of a predecessor task. Validation enforces five rules:

1. **Uniqueness**: Each `jti` must be unique within the workflow scope (or globally if `wid` is absent).

2. **Parent existence:** All `par` entries must reference existing ECTs in the store.
3. **Temporal ordering:** Parent `iat` timestamps must not exceed child `iat` plus clock skew tolerance (recommended 30 seconds).
4. **Acyclicity:** Parent chains must not cycle back to the current task.
5. **Trust domain consistency:** Parents should belong to the same or federated trust domains.

A maximum ancestor traversal limit of 10,000 nodes prevents denial-of-service via deep DAGs.

Figure 1 illustrates a five-task workflow with fan-out and fan-in.

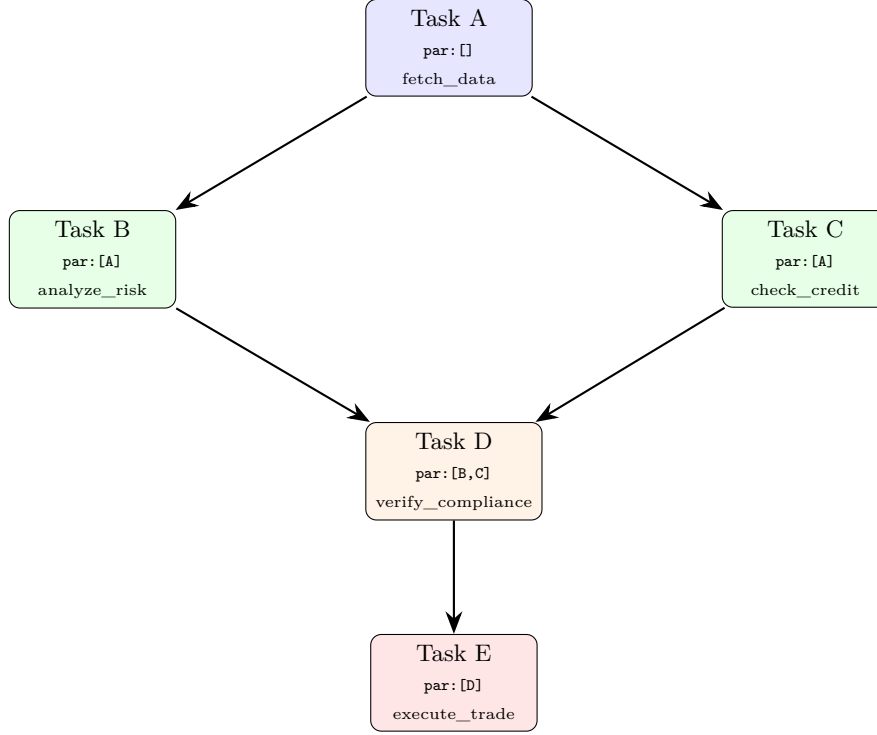


Figure 1: A five-task DAG representing a cross-organization financial workflow. Task A (root) fetches data; Tasks B and C execute in parallel (fan-out); Task D merges results (fan-in); Task E performs the final action. Each task references its predecessors via the `par` claim.

4.4 Verification Procedure

ECT verification proceeds in 13 steps, grouped into four phases:

Failed verification returns HTTP 403 (invalid ECT with valid WIT) or HTTP 401 (signature failure). The overall complexity is $O(1)$ per token except DAG traversal, which is $O(|V| + |E|)$ bounded by the 10,000-node limit.

4.5 HTTP Transport

ECTs travel via a new `Execution-Context` HTTP header alongside WIMSE identity:

Listing 3: HTTP request with WIMSE identity and ECT.

```

GET /api/compliance-check HTTP/1.1
Host: compliance-agent.example.com
Workload-Identity: eyJhbGci...WIT...
Execution-Context: eyJhbGci...ECT...
  
```

Multiple `Execution-Context` headers may appear when multiple parent tasks contribute context (fan-in). Receivers must individually verify each ECT and reject the request if any fails.

Table 4: ECT verification procedure (13 steps in 4 phases).

Step	Phase	Action
1	Serialization	Parse JWS Compact Serialization (header.payload.signature)
2		Verify <code>typ = wimse-exec+jwt</code>
3		Verify <code>alg</code> \in allowlist (must include ES256; reject <code>none</code>)
4	Identity	Verify <code>kid</code> references valid public key from agent’s WIT
5	Binding	Verify JWS signature per RFC 7515
6		Confirm signing key not revoked
7		Verify <code>alg</code> matches WIT algorithm
8		Verify <code>iss</code> matches WIT <code>sub</code> claim
9	Temporal & Audience	Verify verifier’s identity \in <code>aud</code>
10		Verify <code>exp</code> not passed
11		Verify <code>iat</code> \leq now + 30s and \geq now - 15min
12	Structural	Verify <code>jti</code> , <code>exec_act</code> , <code>par</code> present and well-formed
13		Perform DAG validation (Section 4.3)

4.6 Audit Ledger Interface

An optional audit ledger provides immutable storage. Implementations must satisfy four requirements:

1. **Append-only:** No modification or deletion after recording.
2. **Ordering:** Monotonically increasing sequence numbers.
3. **Lookup:** Efficient retrieval by `jti`.
4. **Integrity:** Hash chains or Merkle trees for tamper detection.

Ledgers should be maintained independently of workflow agents to reduce collusion risk.

5 Comparative Analysis Against the Landscape

5.1 Comparison Framework

We compare ECTs against eight proposals from the landscape along eight dimensions relevant to execution accountability:

1. **Execution Recording:** Does it record what an agent did?
2. **DAG Support:** Can it represent parallel/fan-in workflows?
3. **I/O Integrity:** Does it protect input/output data integrity?
4. **Audit Trail:** Does it support immutable audit storage?
5. **WIMSE Integration:** Does it build on the WIMSE trust model?
6. **Authorization Scope:** Does it handle delegation/permission?
7. **Token Format:** Is it based on established standards (JWT/CWT)?
8. **Verification Depth:** How thorough is the verification procedure?

5.2 Head-to-Head Comparison

Table 5 presents the feature matrix. Composite scores are from our landscape analysis (1–5 scale, higher is better); overlap is the LLM-assessed redundancy with other drafts (lower is more unique).

Key observations:

ECT is the only proposal combining execution recording with DAG support. The Actor Chain [9] and Transaction Tokens [13] propagate linear context but cannot represent fan-in workflows where multiple parent tasks converge. The ECT DAG model is strictly more expressive.

Table 5: Feature comparison: ECT vs. eight competing/complementary proposals. ✓ = full, ~ = partial, ✗ = none.

Proposal	Score	Ovlp	Exec Record	DAG	I/O Hash	Audit	WIMSE	AuthZ	JWT/CWT	Verif. Depth
ECT (this work)	4.0	2	✓	✓	✓	✓	✓	✗	✓	13 steps
DAAP v2 [1]	4.8	1	~	✗	✗	✓	✗	✓	✓	Broad
STAMP [4]	4.6	1	✗	✗	~	✓	✗	✓	✓	8 steps
Agentic JWT [3]	4.5	2	✗	~	✗	✗	✗	✓	✓	6 steps
Verif. Conv. [2]	4.5	2	~	✗	~	✓	✗	✗	COSE	Schema
Trans. Attest. [10]	4.3	2	✗	✗	✗	✗	✓	✗	✓	Env. only
Txn Tokens [13]	4.2	3	✗	✗	✗	✗	✗	✓	✓	Linear
Actor Chain [9]	4.1	2	✗	✗	✗	~	✗	✓	✓	Linear
HJS [16]	3.5	1	✓	✗	✗	✓	✗	✗	Custom	Blockchain

No other proposal provides cryptographic I/O integrity. ECTs’ `inp_hash` and `out_hash` claims create a verifiable data lineage chain. STAMP [4] includes message-level proofs but focuses on delegation authorization, not execution data.

DAAP v2 [1] is the most comprehensive competitor. Scoring 4.8, it covers authentication, behavioral monitoring, and remote shutdown—a broader scope than ECTs. However, it does not define a structured execution record format or support DAG workflow modeling. ECTs are complementary: DAAP could use ECTs as its execution record layer.

Transitive Attestation [10] is a natural WIMSE sibling. It binds identities to execution environments (TEEs). Combined with ECTs, the WIMSE stack would provide a complete accountability chain: *who* (WIT), *where* (transitive attestation), *what* (ECT).

HJS [16] shares ECT’s goals but uses a heavier trust model. Blockchain-anchored timestamps provide decentralized accountability but introduce latency and infrastructure requirements that ECTs avoid through standard JWT infrastructure.

5.3 Uniqueness Quantification

To quantify ECT’s positioning beyond subjective comparison, we use embedding-based similarity from the landscape analysis:

- **Maximum pairwise similarity:** 0.836 (with `draft-nederveld-ad1`, an agent definition language—a different problem domain).
- **Overlap rating:** 2/5 (low—among the most unique 25% of all 260 drafts).
- **Cluster membership:** ECT does not belong to any of the high-similarity clusters identified at the 0.85 threshold.
- **Composite score:** 4.0/5, placing ECT in the top 20% of the corpus.

These metrics confirm that ECTs occupy a genuinely novel niche rather than duplicating existing work.

6 Gap Coverage Analysis

We map ECT mechanisms to each of the 12 identified landscape gaps. Table 6 provides the full matrix.

ECTs fully address 3 gaps and partially address 3 more. Notably, ECTs are the **only single proposal in the 260-draft landscape that simultaneously addresses all three**

Table 6: ECT coverage of the 12 identified landscape gaps.

Gap	Sev.	Exist. ^a	ECT	ECT Mechanism
Agent Behavior Verification	Crit.	52	✓	Signed <code>exec_act</code> records what agent claimed to do
Agent Error Recovery	Crit.	6	✓	DAG enables post-mortem tracing; <code>inp_hash/out_hash</code> locate data corruption
Agent Resource Mgmt	Crit.	117	✗	—
Cross-Protocol Translation	High	0	✗	—
Agent Lifecycle Mgmt	High	90	✗	—
Multi-Agent Consensus	High	5	✗	—
Human Override	High	4	✗	—
Cross-Domain Security	High	10	~	DAG validation rule 5 (trust domain consistency)
Dynamic Trust	High	5	~	Execution history enables reputation assessment
Agent Performance Mon.	Med.	26	~	Timestamp-based execution timing from <code>iat</code> of parent/child
Agent Explainability	Med.	5	✗	—
Agent Data Provenance	Med.	0	✓	<code>inp_hash/out_hash</code> chains create verifiable data lineage

^aNumber of existing ideas (of 1,262) that partially address the gap.

execution-layer gaps (behavior verification, error recovery, data provenance). No other draft combines signed execution records, DAG-based workflow modeling, and cryptographic data lineage.

The error recovery coverage is particularly significant: with only 6 of 1,262 ideas touching this topic, the existing landscape provides essentially no support for debugging failed multi-agent workflows. ECT DAGs directly enable the kind of “execution replay” analysis that production deployments require.

7 Discussion and Limitations

7.1 The Self-Assertion Limitation

ECTs are fundamentally self-asserted: an agent creates its own execution record. A compromised or malicious agent can claim to have performed actions it did not, or omit actions it did perform. This is the most significant limitation of the design.

Mitigations include:

- **Independent audit ledgers:** Maintained separately from workflow agents, enabling cross-verification of ECT sequences.
- **Multi-replica comparison:** Multiple observers independently record ECTs and flag discrepancies.
- **TEE integration:** Combining ECTs with transitive attestation [10] to verify the execution environment. If the agent runs in a Trusted Execution Environment, the ECT’s credibility increases significantly.
- **Input/output hash verification:** While agents can falsify `exec_act`, the `inp_hash` and `out_hash` claims can be independently verified by recipients who possess the actual data.

7.2 Scalability Considerations

Large-scale workflows may generate thousands of ECTs. The 10,000-node traversal limit constrains verification cost but may prove insufficient for long-running industrial workflows. Practical deployments may require:

- **Checkpoint ECTs:** Periodic summarization of sub-DAGs into single checkpoint tokens, resetting traversal depth.
- **Lazy verification:** Verifying only the immediate parents during normal operation, with full DAG traversal reserved for audit and debugging.
- **Ledger-assisted verification:** Offloading DAG validation to the audit ledger, which maintains an indexed view of the complete workflow graph.

7.3 Adoption Path

ECTs are designed for incremental adoption:

1. Agents can emit ECTs even if not all recipients verify them (the header is simply ignored by unaware systems).
2. The JWT format leverages existing libraries and tooling in every major programming language.
3. Integration with WIMSE requires only that the agent already possesses a WIT—no additional key infrastructure.
4. The single new HTTP header (**Execution-Context**) requires no changes to request routing or load balancing.

7.4 Limitations of the Landscape Analysis

The quantitative evidence supporting ECTs inherits limitations from the analysis methodology:

- **Keyword selection:** Six seed keywords may miss relevant drafts using different terminology.
- **Single-LLM assessment:** Claude Sonnet 4 may have systematic biases in its ratings.
- **Snapshot:** The analysis captures February 2026; the landscape evolves continuously.
- **Disambiguation:** Author affiliations may be inconsistent (e.g., “Huawei” vs. “Huawei Technologies” are counted separately).

8 Related Work

Distributed tracing. OpenTelemetry [14] provides observability for microservice architectures with trace/span hierarchies. However, spans are unsigned and designed for monitoring, not accountability. ECTs provide cryptographic integrity and are designed for audit, not debugging (though they enable both).

Provenance standards. The W3C PROV data model [8] provides a semantic framework for provenance tracking. PROV is comprehensive but heavyweight—ECTs provide a runtime-embeddable JWT format suitable for HTTP request flows rather than offline provenance databases.

Supply chain transparency. SCITT (Supply Chain Integrity, Transparency, and Trust) [5] defines transparent ledgers for signed statements about supply chain artifacts. ECT audit ledgers share design principles but target runtime execution rather than build-time artifacts. ECTs could correlate with SCITT Signed Statements for end-to-end accountability.

Blockchain-based accountability. Various proposals use blockchain for agent accountability (e.g., HJS [16]). ECTs deliberately avoid blockchain dependency to minimize latency and infrastructure requirements, using optional append-only ledgers instead.

9 Conclusion and Future Work

We have presented Execution Context Tokens (ECTs), a JWT-based extension to the WIMSE architecture for recording task execution in distributed agentic workflows. By analyzing 260 IETF Internet-Drafts using embedding similarity, LLM-assisted rating, and automated gap detection, we demonstrated that the execution accountability layer is the most under-served area in the current standardization landscape.

ECTs address this gap with a design that is both technically sound (DAG-based workflow modeling, cryptographic I/O integrity, 13-step verification) and practically deployable (JWT format, single HTTP header, incremental adoption). A head-to-head comparison against eight competing proposals confirms that ECTs are the only specification combining execution recording, DAG support, and WIMSE-native integration.

Future work includes:

1. **Reference implementation:** An open-source library for ECT creation, verification, and DAG visualization.
2. **Formal verification:** Applying ProVerif or Tamarin to the ECT security model.
3. **Integration testing:** Deploying ECTs alongside WIMSE WIT/WPT and transitive attestation in a multi-agent testbed to validate the “WIMSE trinity” concept.
4. **Privacy-preserving ECTs:** Selective disclosure of execution details using zero-knowledge proofs or redactable signatures.
5. **Longitudinal tracking:** Monitoring the landscape as it evolves and assessing whether ECTs catalyze additional execution-layer proposals.

The ECT specification is available as IETF Internet-Draft `draft-nennemann-wimse-ect-00` [11]. The landscape analysis toolkit and dataset are released as open source.¹

Acknowledgments

The landscape analysis was performed using Anthropic Claude (Sonnet 4) for rating and idea extraction, and Ollama with `nomic-embed-text` for embedding generation. The author thanks the IETF community for maintaining the open Datatracker API and the WIMSE working group for providing the identity foundation on which ECTs build.

References

- [1] Jonathan Aylward. Distributed AI Accountability Protocol (DAAP) Version 2.0. Internet-Draft `draft-aylward-daap-v2`, 2026.
- [2] Henk Birkholz. Verifiable Agent Conversation Records. Internet-Draft `draft-birkholz-verifiable-agent-conversations`, 2025.
- [3] Anirudh Goswami. Secure Intent Protocol: JWT Compatible Agentic Identity and Workflow Management. Internet-Draft `draft-goswami-agentic-jwt`, 2025.
- [4] Bary Guy. Secure Task-bound Agent Message Proof (STAMP) Protocol. Internet-Draft `draft-guy-bary-stamp-protocol`, 2025.
- [5] IETF SCITT Working Group. Supply Chain Integrity, Transparency, and Trust (SCITT). Internet-Draft, 2024.
- [6] M. Jones, J. Bradley, and N. Sakimura. JSON Web Signature (JWS). RFC 7515, 2015.
- [7] M. Jones, J. Bradley, and N. Sakimura. JSON Web Token (JWT). RFC 7519, 2015.

¹Repository: <https://github.com/TODO/ietf-draft-analyzer>

- [8] Luc Moreau and Paolo Missier. PROV-DM: The PROV Data Model. W3C Recommendation, 2013.
- [9] MW et al. Cryptographically Verifiable Actor Chain for OAuth 2.0 Token Exchange. Internet-Draft draft-mw-spice-actor-chain, 2025.
- [10] MW et al. Transitive Attestation for Sovereign Workloads: A WIMSE Profile. Internet-Draft draft-mw-wimse-transitive-attestation, 2025.
- [11] Christian Nennemann. Execution Context Tokens for Distributed Agentic Workflows. Internet-Draft draft-nennemann-wimse-ect-00, February 2026.
- [12] Christian Nennemann. The AI Agent Standardization Wave: A Quantitative Analysis of 260 IETF Internet-Drafts on Autonomous Agents and Artificial Intelligence. arXiv preprint, 2026. Companion landscape survey paper.
- [13] OAuth Working Group. Transaction Tokens For Agents. Internet-Draft draft-oauth-transaction-tokens-for-agents, 2025.
- [14] OpenTelemetry Project. OpenTelemetry Specification. CNCF Project, 2024.
- [15] SPIFFE Project. SPIFFE: Secure Production Identity Framework for Everyone. CNCF Project, 2024.
- [16] et al. Wang. HJS: An Accountability Layer for AI Agents. Internet-Draft draft-wang-hjs-accountability, 2025.
- [17] WIMSE Working Group. Workload Identity in Multi System Environments (WIMSE) Architecture. Internet-Draft, 2025.