Network Working Group
Internet-Draft
Intended status: Informational
Expires: 19 November 2026

M. Kuehlewind
Ericsson
H. Birkholz
Fraunhofer SIT

18 May 2026

An Architecture for Auditing AI Agent Delegation and Interactions
draft-kuehlewind-audit-architecture-00

Abstract

This document describes an architecture for auditing of agent-driven interactions on the Internet. Autonomous and semi-autonomous software agents, including those based on artificial intelligence, increasingly act on behalf of users, organizations, and services. Existing auditing mechanisms often capture isolated system events but do not consistently represent delegation relationships, user intent, or evolving authorization. In agent-driven systems, auditability requires linking intent, delegation, authorization, and execution. The proposed architecture enables this through distributed audit record generation, propagation of audit context, optional attestation, and additonal logging for transparency.

Discussion Venues

This note is to be removed before publishing as an RFC.

Source for this draft and an issue tracker can be found at https://github.com/mirjak/draft-audit-architecture.

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 19 November 2026.

Table of Contents

1. Introduction

Autonomous and semi-autonomous software agents based on large language models (LLMs) and similar non-deterministic systems are deployed to take consequential actions on behalf of users and organizations. These agents interact across administrative and trust domains, delegate tasks and authority to other agents or tools, and initiate consequential actions without per-step human oversight. The question of whether the recorded actions of an agent faithfully represent what the agent actually did has acquired new urgency.

Autonomous agents may run long-lived workflows without tight user interaction or may be very short-lived, e.g. for a delegated sub-tasks. Agents may be authenticated to several services, request step-up approval from a human, spawn further sub-agents, and produc records that long outlive its own process. Existing auditing mechanisms often capture isolated system events but do not consistently represent delegation relationships, user intent, or evolving authorization. In agent-driven systems, auditability requires linking intent, delegation, authorization, and execution.

This document describes an architecture that enables this form of auditing through three layers: - distributed audit record generation by each participant, - propagation of audit context across protocol interactions, and - optional attestation and independent third-party logging for transparency that provides verifiable assurances about the content and origin of records.

The architecture in this document identifies roles and their duties, describes the classes of interaction that must be audited, and discusses an example data model to make audits interoperable across vendors, domains, and time.

Two principles frame the rest of this document:

1. Agents participate in _two distinct classes of interaction_ that must each be auditable: user-facing interactions (prompts, approvals, human-in-the-loop confirmations) and system-facing interactions (API calls, tool invocations, delegation to other agents or services). Effective auditing requires linking user intent to resulting system actions across protocol and administrative boundaries.

2. Unlike traditional delegated workflows in which authorization transitions are explicit and predefined, AI agent systems introduce dynamic, fine-grained authorization changes that arise during execution and are driven by agent decisions, sub-agent delegation, and human interaction. Auditing must therefore

capture authorization as a _time-evolving state_ and must correlate transitions across interactions and domains by maintaining common context.

1.1. Relationship to Other IETF Work

The architecture is designed to compose existing IETF building blocks to make verifiable what these layers already do rather than redefining them.

Remote attestation follows RATS [RFC9334] supplies the environmental evidence of the record’s origin: RATS Evidence, Attestation Results, and Endorsements are reused verbatim as the vocabulary for environmental claims about audit-record producers.

Transparency follows SCITT [I-D.ietf-scitt-architecture]that makes a record’s existence later un-deniable: SCITT Signed Statements, Receipts, and Transparent Statements are the canonical artifacts, and SCITT-compatible Transparency Services are the canonical substrate for non-repudiable custody.

The Verifiable Agent Conversations data model [I-D.birkholz-verifiable-agent-conversations] could be utilizd as an Interaction Record.

HTTP may be used as the transport mechanism for conveying audit context alongside requests. JSON-based formats, including JWT and COSE, can proide representations for audit records and attestations, along with mechanisms for cryptographic protection.

Authority and delegation are based on by OAuth 2.0 [RFC6749], Token Exchange [RFC8693], Transaction Tokens [I-D.ietf-oauth-transaction-tokens], Identity Chaining [I-D.ietf-oauth-identity-chaining], Identity Assertion Authorization Grants [I-D.ietf-oauth-identity-assertion-authz-grant], RAR [RFC9396], attestation-based client authentication [I-D.ietf-oauth-attestation-based-client-auth], DPoP [RFC9449], Status Lists [I-D.ietf-oauth-status-list], and SPIFFE client authentication [I-D.ietf-oauth-spiffe-client-auth].

Workload identity follows WIMSE [I-D.ietf-wimse-arch], which this document specializes for AI Agents.

The three principal acting roles as described in the next section and shown in Figure 1 have parallel counterparts in OAuth and WIMSE; a deployment may use both. The following table shows a simple mapping:

| Actor View (this document) | OAuth view [RFC6749] | WIMSE view [I-D.ietf-wimse-arch] | |---|---|---| | User | Resource Owner | Principal of a run; may also be a Workload in machine-only runs | | AI Agent | OAuth Client | Workload (with sub_profile=ai_agent) presenting a Workload Identity Credential | | External Service / Tool | Resource Server | Service-side Workload or external endpoint of another Trust Domain |

The auditing layer adds the cross-layer artifacts (audit records and context, agent interaction records, attestation references, and transparency receipts) that turn isolated layer events into a verifiable audit trail.

2. Motivating Use Cases

The need for interoperable auditing of agent-driven systems arises from both regulatory requirements and user trust expectations. The following examples highlight scenarios where traditional logging is insufficient and where an explicit auditing architecture for agents provides value.

The proposed auditing architecture provides traceability of data access and enables reconstruction of the full chain from user intent to execution, including the full delegation chain that might change dynamically and authorization decisions, providing the desired verifiable audit trail suitable for compliance and review.

2.1. Financial Transactions by Agents

Agents may execute financial operations such as payments or procurement actions on behalf of users, often involving multiple systems. Regulatory frameworks (e.g., SOX, PSD2) require that transactions be attributable, authorized, and auditable.

Traditional logs typically capture only execution events (e.g., “payment executed”), without clearly linking them to user intent, approvals, or delegation steps. This makes it difficult to verify whether actions were properly authorized.

2.2. Long-Running Autonomous Agents

Some agents operate continuously over extended periods, making decisions and performing actions based on changing conditions. For example, a procurement agent may manage ordering and inventory over days or weeks. In such scenarios, authorization evolves over time due to policy changes, approvals, or context-dependent decisions. Delegation paths may also change dynamically.

2.3. Everyday Agent Interaction

Even simple consumer use cases benefit from improved auditing. For example, an assistant may book a restaurant on behalf of a user by selecting a venue and interacting with a booking service. If the result is unexpected, traditional logs provide limited insight into how the decision was made or which services were involved.

2.4. Data Sharing and User Trust

Agents often access and share (sensitive) user data with external services. For example, an assistant may send a user's address to a delivery provider. Without detailed auditing, it is difficult to verify what data was accessed, what was shared, and with whom. Unintended disclosure can undermine user trust.

3. Architectural Overview

Figure 1 shows a high level end-to-end view of the proposed architecture: three principal acting roles produce records about their own behaviour. These records flow into supporting services that store, attest, and expose them to consumers that can verify their authenticity and provenance.

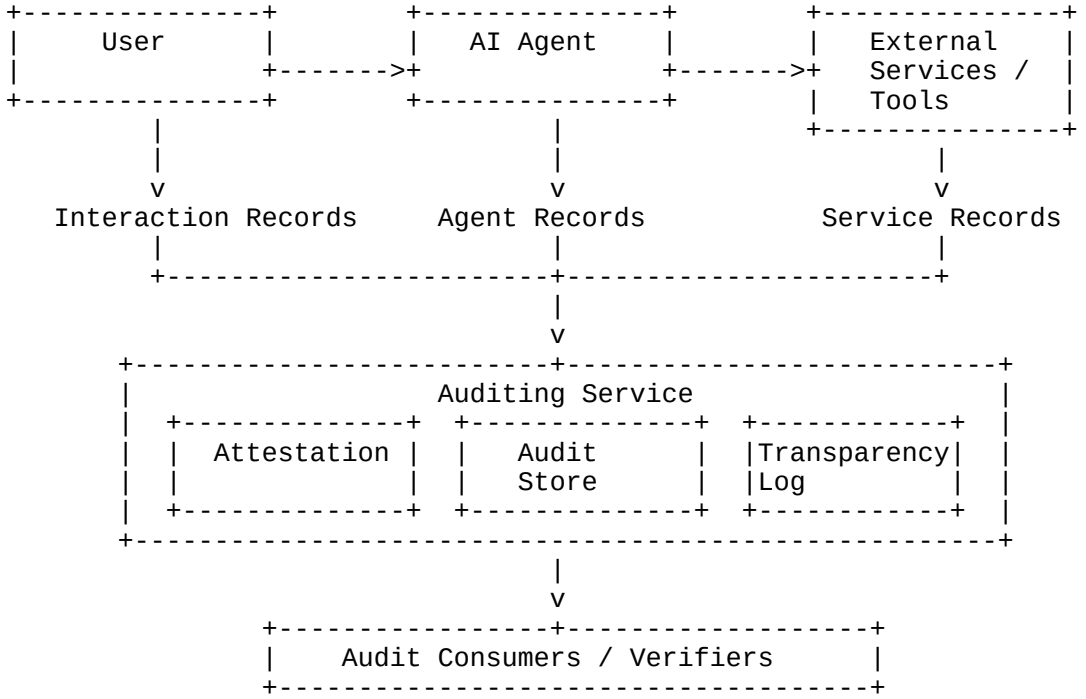

Figure 1: Roles view: principal acting roles, auditing services, and the records that flow among them.

The proposed architecture enables interoperable auditing of agent-driven interactions by combining distributed audit record generation, audit context propagation, and optional attestation and transparency logging. Audit information is produced by multiple actors operating across administrative domains and is later reconstructed and validated by audit consumers through a shared audit context and may be accompanied by attestations.

Audit records are generated distributively. Each principal acting role (user, agent, service/tool) records its own behaviour rather than relying on a central observer. These records are linked via a propagated audit context with a coherent trail: Interaction Records flow are generated by the User; Action and Delegation Records flowby the Agent; Service Records flow by external Services/Tools. Each actor may produce mutiple records to, e.g., audit actions, delegation, or authorization changes. Distributed record generation limits the trust placed in any single point. Two trust mechanisms are composed (attestation and transparency logging). An auditing system may select either, both, or neither according to the strength of evidence its Auditors require.

The auditing services (Attestation, Audit Store, and Transparency Log) are distinct from the interaction and authorization layers that drives the three acting roles. They can be or need to be operated by different parties dependeing on trust requirements of auditor. Attestations may be directly provided by the producer or supplied on request by an Attestation Service. An Audit Store canonicalises observable agent signals into the records and exposes them to Audit Consumers and Verifiers. The Store is logically distinct from the Agent it records, because an agent that records itself can produce useful telemetry but cannot, by itself, deliver non-repudiation to a third party. Transparency receipts provided by the Transparency Log are embedded in or referenced from the records.

3.1. Audit Records and Context

Audit records are generated independently by participating actors and reflect different perspectives of an interaction. User-facing systems produce records capturing intent, such as prompts or approvals. Agents produce records describing decisions, actions, and delegation steps. Services produce records reflecting execution outcomes.

These records are linked through a shared audit context that is propagated across protocol interactions. This context carries identifiers such as a trace identifier and references to prior events, enabling reconstruction of a causal chain. It also carries identity information, including the acting entity and, where applicable, the entity on whose behalf the action is performed.

Transport mechanisms such as HTTP are used for propagating this context, for example via headers or message metadata. Existing approaches such as W3C's trace context propagation can serve as a basis but require extension to include identity, delegation, and authorization state. As such, correlation is not performed by a centralized component. Instead, it emerges from consistent use of shared identifiers and structures across all participants.

3.2. Attestation Model

Audit records may include attestation evidence that provides verifiable assurances about their content. An attestation binds a statement to a cryptographic identity and allows relying parties to validate claims about events, delegation, or execution.

Attestations are generated at or on behalf of the entity asserting a claim and are associated with audit records at creation time. Attestation can have different levels of assurance, depending on the type of origin (ranging from self-assertions to RATS evidence). An agent or service may self-attest its actions using mechanisms such as JSON Web Signatures or COSE-based signatures. In other cases, the producer interacts (taking on the role of a RATS Attester) with an external remote attestation service (i.e., a RATS Verifier) to obtain an additional statement (an Attestation Result).

Further, the transparency service may attest that a record has been recorded in an append-only log. And the audit store may provide additional attestations, such as timestamping or proof of inclusion. But this does not replace attestations generated by record producers.

3.3. Identity Substrate

An AI Agent is treated as a specialisation of a Workload [I-D.ietf-wimse-arch], with a Workload Identifier [I-D.ietf-wimse-identifier] scoped within a Trust Domain and a Workload Identity Credential (e.g., a WIT [I-D.ietf-wimse-workload-creds]) bound to a key the workload generates and retains. The credential is never a bearer token and is normally short-lived. For auditing it is benefical if an Agent also carries a role profile (per [I-D.mcguinness-oauth-actor-profile]'s sub_profile convention) so that downstream parties can distinguish an

AI-driven Workload from a human-operated client or a traditional service. The role-profile vocabulary is the subject of a separate specification (WI-2 in Section 6).

4. Roles

A role is a function, not a deployment unit. The same entity may take on several roles.

4.1. Principle Agent Interaction Actors

The User is the human or organisation on whose behalf an Agent acts. Where the User is a natural person they are also the OAuth Principal of the run: the sub of any token issued for the run and the original authorizing party in any delegation chain. On the audit layer the User's duty is to issue intent in a verifiable form--prompt, approval, or signed grant--and to respond to step-up escalations.

An AI Agent is a Workload [I-D.ietf-wimse-arch] whose behaviour is driven, in whole or in part, by a non-deterministic decision process (typically an LLM). Agents operate within the authorization they were issued, propagate the upstream Principal's identity and the delegation context unmodified except where an exchange explicitly authorises a change, and emit observable signals, such as prompts, actions, tool calls, sub-agent invocations, terminations, that an Audit Store can canonicalise. Where an Agent signs records itself, it signs with a key bound to its Workload Identity Credential, never with a long-lived shared key. A Sub-Agent is an Agent invoked by another Agent rather than directly by a User. This distinction is positional and not categorical. A Sub-Agent receives a downscoped authorization, represented in the delegation chain, and re-binds the chain on outbound calls so downstream Services can attribute the call correctly and trace back to the parent.

A Tool is co-located with or directly invocable from an Agent runtime (filesystem, shell, sandbox, function call). A Service is a remote workflow reached via a network protocol. A Resource Server provides an access service to an authorization-protected resource. All three enforce the Agent's presented authorization at the point of effect and emit per-request records bound to the Agent's Workload Identity Credential. Those records reflect the canonical request as observed at the boundary, not as reported by the Agent.

4.2. Auditing Services

The Auditing Service canonicalises observable records (Interaction, Action, Delegation, Authorization Transition), signs them with a key bound to its own identity, and submits them for registration with a Transparency Log before they leave the operational environment. The Auditing Service is logically distinct from the User, Agent, or Tool it records: the architecture's accountability properties require either an independent Auditing Service.

The Audit Store is a generic role of storing audit records after they have been produced. Substrates range from append-only databases and SIEM-fed log stores to fully transparent registries. Deployment requires only availability of records may use any type of Audit Store implementation.

Attestation binds a record to the operational state of the environment in which it was produced. A record may carry inline Evidence about that environment, an Attestation Result derived from that Evidence by a Verifier, or a stable reference to either, and the resulting record carries a verifiable claim about _what was running_ and _in what configuration_ when the recorded action took place. Remote Attestation is the architecture's principal candidate for producing records that are authentic in this sense. RATS [RFC9334] is the canonical instance.

The Transparency Log binds a record to an append-only, non-equivocating statement sequence. Once a Transparency Log has issued a receipt for a signed statement, the record's existence and content at registration time can no longer be denied, and no Auditor with read access can be presented with an inconsistent view of the history. Transparency is the architecture's principal mechanism for producing records that are non-repudiable in this sense. SCITT [I-D.ietf-scitt-architecture] is the canonical instance.

Attestation and transparency answer different questions: attestation answers "is this record authentic?", transparency answers "did this record exist as claimed?". They compose freely as a record may carry attestation evidence and be transparently registered, and the resulting receipt then binds that combination immutably.

4.3. Auditor

The Auditor consumes Interaction, Action, Delegation, and Authorization Transition Records together with the receipts that bind them, and determines whether recorded behaviour matched the User's intent and the authorization in force at the time and, where appropriate, whether the recorded evidence supports or contradicts a claim of compliance. The Auditor is not required to trust the Agent: the entire point of the architecture is to make the Agent itself the auditable object.

4.4. Role Aggregation

An entity can assume one or more roles. An Agent's host process may also implement the Audit Recorder role for that Agent. Doing so simplifies deployment but introduces recorder-collusion threats that the architecture mitigates by either an independent Recorder or non-repudiable registration with a Transparency Service. An Auditor can be operated by an organisation distinct from the one operating the Recorder, the Transparency Service, the Auditing Service, or the Agent. The architecture's requirements apply across role boundaries even when those roles are co-located.

5. Record Types

This document proposes an initial set of four record types for agent auditing:

Interaction Records: capture user-facing events (prompts, model responses, instructions, approvals, refusals) and agent-to-agent dialogue treated as conversation.

The Verifiable Agent Conversations data model [I-D.birkholz-verifiable-agent-conversations] is the principal candidate format for the user-Agent dialogue subtype.

Human-in-the-loop escalations, like step-up approvals, refusals, or the rarer case of an action that should have been escalated but was not, appear as identifiable records bindable to the run's Interaction Record.

Action Records: capture system-facing events at the boundary where the action took effect.

An Action Record may carry inline Evidence, an Attestation Result, or a stable reference to either; where Evidence is unavailable, that absence is itself an audit-relevant fact and can be recorded.

Delegation Records: capture the assignment of authority from one entity to another,e.g., delegator, delegatee, scope, and constraints on use.

Authorization delegation records are used to audit which authority is transmitted along the chain from User through Agent through Sub-Agent to Tool or Service. Across that chain, the record remains append-only, meaning no actor removes or reorders prior actors. Any cross-domain transition is recorded with enough fidelity that an Auditing Service on either side can make sense of it without access to the other side’s pipeline.

Authorization Transition Records: capture changes in permission state over time: initial grants, step-up approvals, scope narrowing on exchange, revocation, and expiry.

Authorization is considered as a time-evolving state. The ordered sequence of Authorization Transition Records of a run reconstructs the authorization in force at any point within it.

Concrete illustrative shapes for each are given in Section 7. The four classes of records share a common correlation identifier in the Audit Context (see Section 7.1 and WI-11), so that a User’s stated intent and an Agent’s executed action remain linkable across protocol and administrative boundaries.

Consequential records are attested by an in-device or third-party service, stored in the Audit Store for retrival by an authorized auditor, and registered with a Transparency Service before they leave the operational environment, or as soon as network conditions permit.

6. Potential Work Items

The following work items are proposed for potential specifications that support this architecture:

* *WI-1: Audit Data Models and Semantics.* The canonical structure of Interaction, Action, Delegation, and Authorization Transition Records (Section 5), encoded in at least one IETF-recognised serialisation (CBOR/COSE or JSON/JWS) with support for detached payloads, and carrying actor identity unambiguously across User, Agent, Sub-Agent, Tool, and Service.

* *WI-2: Delegation-Chain Wire Profile.* Specify both a Cryptographic Delegation Chain carried in token bodies (nested act per [RFC8693]; acti/actc candidates from [I-D.mw-oauth-actor-chain]) as well as a Tracing Delegation Chain (a flat, lightweight Actor sequence suitable for audit context,

HTTP headers, and standalone records) with a defined reconciliation path between them. This item includes a representation for cross-domain transitions, and a sub_profile [I-D.mcguinness-oauth-actor-profile] vocabulary that distinguishes AI Agent, Sub-Agent, Tool, Service, and Human.

* *WI-3: Interaction Record Profile.* A canonical Interaction Record format for prompts, responses, instructions, approvals, refusals, tool-invocation traces, reasoning traces (where exposed by the model), and system events, potentially with an identifiable HITL subtype and a registration profile compatible with SCITT. [I-D.birkholz-verifiable-agent-conversations] is the principal candidate for the User-Agent dialogue subtype. This work item does not preclude additional profiles for non-conversational interactions, such as network device interaction.

* *WI-4: Action Record Profile.* A canonical Action Record produced at the boundary where each tool or service call took effect, bound to its parent Interaction Record via SC-2/SC-11 tracing identifiers, to its authorizing Token, and (when available) to the Attestation Result for the executing environment. Distinguishes the Recorder’s signing identity from the recorded Agent’s identity where the two are operationally separated.

* *WI-5: HITL Escalation Signalling.* Communication of step-up requests from Agent to User and the User’s response, e.g., approval, refusal, or timeout, in a form auditable end-to-end and bindable to the resulting authorization state.

* *WI-6: Profile of RATS Evidence.* How Evidence is referenced from Interaction and Action Records using [RFC9334]’s encoding-agnostic Conceptual Messages, and how Attestation Results derived from such Evidence are consumed by Identity Issuance Authorities, Services, and Auditors.

* *WI-7: Profile of SCITT Transparency.* A Registraton Policy profile of [I-D.ietf-scitt-architecture] for auditing records--admissible Issuers (Agents, Sub-Agents, Recorders) and required payload media types--and a Receipt presentation profile permitting Auditors to verify non-repudiation independently of any single Transparency Service.

* *WI-8: Authorization Transition Encoding.* A canonical Authorization Transition Record format carrying previous state, new state, triggering event, and responsible actor (see Section 7.4), reusing [I-D.ietf-oauth-status-list] where the state is a token-status state, and replayable to reconstruct authorization in force at any timestamp within a run.

* *WI-9: Auditor-Facing Query Interface.* An optional specialised query profile over the Audit Store and Transparency Log, surfacing records by session, workflow, principal, agent, tool, or time range, with authorization and privacy controls.

* *WI-10: Deployment and Operations Best Practices.* Recorder placement, Identity Issuance Authority configuration for ephemeral Workloads, Trust Domain partitioning, operational separation between Agent runtime and audit pipeline, and the privacy guidance on redaction, retention, and disclosure that Section 9 relies on.

* *WI-11: Audit Context Propagation Protocol Extensions.* This could be realized by an HTTP headers carrying the Audit Context, e.g. a workflow-wide Audit-Trace-ID, an immediate-redecessor Audit-Parent-ID, the current Audit-Actor, the upstream Audit-On-Behalf-Of, the SC-2 tracing chain as Audit-Delegation-Chain, and a reference to the current Audit-Auth-State. Alternatively, a single composite Audit-Context header could be defined that provides the audit context embedded in OAuth token claims (Section 7.1). The relationship to existing distributed-tracing conventions (W3C Trace Context, OpenTelemetry) need to be considered.

7. Illustrative Audit Record Examples

This section is informative. It illustrates the four classes of audit record introduced in Section 5 and the audit context introduced in SC-2 and SC-11 with concrete examples. The exact field names, claim names, and encodings shown here are placeholders pending the specifications defined in SC-1 through SC-11. They are intended to convey the relationships among the artifacts.

To enable interoperability, audit records require a common structure that captures identity, delegation, and causal relationships. Each record includes identifiers for correlation, such as a trace identifier and a reference to a preceding event. It identifies the acting entity and may include an “on-behalf-of” identity to represent delegation. The record also captures relevant authorization state, such as scope and validity, which may be derived from OAuth tokens or authorization responses.

7.1. Audit Context in an Access Token

An access token or similar credential MAY include an audit claim carrying correlation and tracing-chain information alongside the conventional OAuth claims. The cryptographic delegation chain remains in the OAuth act claim [RFC8693]; the audit claim carries the tracing-layer chain (SC-2) and the correlation identifiers used to link records:

```
{
  "iss": "authz.example",
  "sub": "agent-42",
  "aud": "calendar.service",
  "exp": 1715674800,

  "audit": {
    "trace_id": "trace-abc",
    "parent_id": "int-001",
    "delegation_chain": ["user-123", "agent-42"]
  },

  "authorization": {
    "scope": ["calendar.write"],
    "expires_at": "2026-05-14T11:00:00Z"
  }
}
```

The same information MAY be propagated using HTTP headers (SC-11) or included in standalone audit records.

7.2. Action Record

An Action Record produced at the boundary where a tool or service call took effect:

```
{
  "event_id": "act-456",
  "trace_id": "trace-abc",
  "parent_id": "int-001",
  "timestamp": "2026-05-14T10:00:00Z",
  "type": "action",

  "actor": { "type": "agent", "id": "agent-42" },
  "on_behalf_of": { "type": "user", "id": "user-123" },

  "action": {
    "type": "api_call",
    "target": "calendar.service",
    "operation": "create_event"
  },

  "delegation_chain": ["user-123", "agent-42"],

  "authorization": {
    "scope": ["calendar.write"],
    "expires_at": "2026-05-14T11:00:00Z"
  }
}
```

The parent_id references the Interaction Record that motivated the action. The delegation_chain is the SC-2 tracing chain. The cryptographic chain lives in the authorizing token and is not duplicated here.

7.3. Delegation Record

A Delegation Record produced when an Agent narrows or assigns authority to a Sub-Agent:

```
{
  "event_id": "del-789",
  "trace_id": "trace-abc",
  "parent_id": "act-456",
  "timestamp": "2026-05-14T10:01:00Z",
  "type": "delegation",

  "delegator": { "type": "agent", "id": "agent-42" },
  "delegatee": { "type": "agent", "id": "agent-sub-1" },

  "scope": ["email.send"],

  "constraints": {
    "expires_at": "2026-05-14T10:30:00Z"
  }
}
```

7.4. Authorization Transition Record

An Authorization Transition Record produced when a Principal's authorization state changes. This example illustrates a user approval:

```
{
  "event_id": "auth-001",
  "trace_id": "trace-abc",
  "parent_id": "int-002",
  "timestamp": "2026-05-14T10:02:00Z",
  "type": "authorization_transition",

  "previous_state": { "scope": ["calendar.read"] },
  "new_state":      { "scope": ["calendar.read", "calendar.write"] },

  "trigger": { "type": "user_approval" },
  "actor":   { "type": "user", "id": "user-123" }
}
```

An ordered sequence of such records reconstructs the authorization state in force at any point in a recorded run.

8. Security Considerations

The architecture's security properties are properties of the combination of mitigations supplied by the underlying layers (RATS, SCITT, WIMSE, OAuth) plus the bindings added by this architecture and any new potenial specification (Section 6). No single layer suffices on its own.

While the final architecture and set of specification require a detailed security and threat model analysis, some initial consideration are stated here:

First, the Agent is _not_ trusted; the architecture is designed precisely for the case where the Agent’s internal behaviour is the subject of scrutiny.

Second, the Audit Store is only partially trusted: its records are corroborated by independent Service-side records and, for consequential actions, by registration with a Transparency Service.

Third, neither a single Verifier nor a single Transparency Service is assumed sufficient: a deployment that requires defence against compromise of either re-appraises Evidence against independent Reference Values and Endorsements, registers records to multiple Transparency Services, and verifies mutual consistency in each case.

The architecture does _not_ solve the case of an adversarial Service that refuses to record what happened at its boundary, nor the case of operator collusion across all roles, nor the alignment of the model behind the Agent.

9. Privacy Considerations

Audit records of AI agent activity are intrinsically rich in information about the User, the User’s intent, the Agent’s reasoning trace where exposed, and the data the Agent processed in tool calls. As such privacy considerations are of special importance for auditing.

Prompt and response content frequently contain personally identifiable information, confidential business information, or content under contractual confidentiality. Records of this content registered to a Transparency Service may be visible to a broader set of parties than the original interlocutors. Subsequent specifications (WI-3) must permit detached payloads so that the registered Signed Statement may carry only a hash of the conversation, with the content held elsewhere under deployment controls; WI-10 best practice need to address redaction and retention policies.

The chain identifier required for cross-service correlation (Section 5) also enables correlation of an Agent's--and by extension a User's--activity across the Services it touches. Chain identifiers can be encrypted to specific Auditors when the Service does not need to correlate itself, and pairwise per-Service identifiers ([I-D.johansson-direct-presentation-arch]'s pairwise pattern) could substitute for a single global chain identifier where correlation is not required.

Tool-call outputs should be referenceable by hash rather than included inline in WI-4 Action Records. Where inline inclusion is required, encryption to a specific Auditor should be supported. Receipts [I-D.ietf-scitt-architecture] bind a registration to a position in the Transparency Service's data structure. the fact of registration leaks the existence of an interaction at a time even when the Statement payload is hash-only.

Where an Agent operates across organisational boundaries, the audit context propagated outward reveals the existence and rough size of the workflow (via Trace ID continuity), the identity or pseudonym of every Actor that participated earlier in the workflow (via the Tracing Delegation Chain), and the structural shape of the workflow (via the Parent ID graph). These are not theoretical concerns; the same primitives are the basis of operational distributed-tracing systems where the trace stream is routinely used for capacity planning, account profiling, and adversarial reconnaissance. To address this, the Tracing Delegation Chain could be replaced at an organisational boundary by an opaque identifier whose mapping is retained only at the originating side's Audit Store (the egress identity generalisation pattern of Section 3.3.8 of [I-D.ietf-wimse-arch]).

10. IANA Considerations

This document has no IANA actions.

11. References

11.1. Normative References

[I-D.ietf-oauth-attestation-based-client-auth]
Looker, T., Bastian, P., and C. Bormann, "OAuth 2.0 Attestation-Based Client Authentication", Work in Progress, Internet-Draft, draft-ietf-oauth-attestation-based-client-auth-08, 2 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-oauth-attestation-based-client-auth-08>.

[I-D.ietf-oauth-identity-assertion-authz-grant]
Parecki, A., McGuinness, K., and B. Campbell, "Identity Assertion JWT Authorization Grant", Work in Progress, Internet-Draft, draft-ietf-oauth-identity-assertion-authz-grant-03, 22 April 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-oauth-identity-assertion-authz-grant-03>.

[I-D.ietf-oauth-identity-chaining]
Schwenkschuster, A., Kasselman, P., Burgin, K., Jenkins, M. J., Campbell, B., and A. Parecki, "OAuth Identity and Authorization Chaining Across Domains", Work in Progress, Internet-Draft, draft-ietf-oauth-identity-chaining-12, 11 May 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-oauth-identity-chaining-12>.

[I-D.ietf-oauth-spiffe-client-auth]
Schwenkschuster, A., Kasselman, P., Rose, S., and S. Thorgersen, "OAuth SPIFFE Client Authentication", Work in Progress, Internet-Draft, draft-ietf-oauth-spiffe-client-auth-01, 2 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-oauth-spiffe-client-auth-01>.

[I-D.ietf-oauth-status-list]
Looker, T., Bastian, P., and C. Bormann, "Token Status List (TSL)", Work in Progress, Internet-Draft, draft-ietf-oauth-status-list-20, 20 April 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-oauth-status-list-20>.

[I-D.ietf-oauth-transaction-tokens]
Tulshibagwale, A., Fletcher, G., and P. Kasselman, "Transaction Tokens", Work in Progress, Internet-Draft, draft-ietf-oauth-transaction-tokens-08, 2 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-oauth-transaction-tokens-08>.

[I-D.ietf-scitt-architecture]
Birkholz, H., Delignat-Lavaud, A., Fournet, C., Deshpande, Y., and S. Lasker, "An Architecture for Trustworthy and Transparent Digital Supply Chains", Work in Progress, Internet-Draft, draft-ietf-scitt-architecture-22, 10 October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-scitt-architecture-22>.

[I-D.ietf-wimse-arch]
Salowey, J. A., Rosomakho, Y., and H. Tschofenig, "Workload Identity in a Multi System Environment (WIMSE) Architecture", Work in Progress, Internet-Draft, draft-ietf-wimse-arch-07, 2 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-wimse-arch-07>.

[I-D.ietf-wimse-identifier]
Rosomakho, Y. and J. A. Salowey, "Workload Identifier", Work in Progress, Internet-Draft, draft-ietf-wimse-identifier-02, 2 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-wimse-identifier-02>.

[I-D.ietf-wimse-workload-creds]
Campbell, B., Salowey, J. A., Schwenkschuster, A., Sheffer, Y., and Y. Rosomakho, "WIMSE Workload Credentials", Work in Progress, Internet-Draft, draft-ietf-wimse-workload-creds-01, 5 May 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-wimse-workload-creds-01>.

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/rfc/rfc6749>.

[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>.

[RFC9396] Lodderstedt, T., Richer, J., and B. Campbell, "OAuth 2.0 Rich Authorization Requests", RFC 9396, DOI 10.17487/RFC9396, May 2023, <https://www.rfc-editor.org/rfc/rfc9396>.

[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>.

11.2. Informative References

[I-D.birkholz-verifiable-agent-conversations]
Birkholz, H., Heldt, T., and O. Steele, "Verifiable Agent Conversation Records", Work in Progress, Internet-Draft, draft-birkholz-verifiable-agent-conversations-00, 25 February 2026, <https://datatracker.ietf.org/doc/html/draft-birkholz-verifiable-agent-conversations-00>.

[I-D.johansson-direct-presentation-arch]
Johansson, L., Zundel, B., and T. Cappalli, "A reference architecture for direct presentation credential flows", Work in Progress, Internet-Draft, draft-johansson-direct-presentation-arch-01, 4 November 2025, <https://datatracker.ietf.org/doc/html/draft-johansson-direct-presentation-arch-01>.

[I-D.mcguinness-oauth-actor-profile]
McGuinness, K., "OAuth Actor Profile for Delegation", Work in Progress, Internet-Draft, draft-mcguinness-oauth-actor-profile-00, 30 April 2026, <https://datatracker.ietf.org/doc/html/draft-mcguinness-oauth-actor-profile-00>.

[I-D.mw-oauth-actor-chain]
Prasad, A., Krishnan, R., Lopez, D., and S. Addepalli, "Cryptographically Verifiable Actor Chains for OAuth 2.0 Token Exchange", Work in Progress, Internet-Draft, draft-mw-oauth-actor-chain-00, 1 May 2026, <https://datatracker.ietf.org/doc/html/draft-mw-oauth-actor-chain-00>.

Authors' Addresses

Mirja Kuehlewind
Ericsson
Email: mirja.kuehlewind@ericsson.com

Henk Birkholz
Fraunhofer SIT
Email: henk.birkholz@ietf.contact