Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 29 August 2026

H. Birkholz

T. Heldt

O. Steele

25 February 2026

# Verifiable Agent Conversation Records
## draft-birkholz-verifiable-agent-conversations-00

Abstract

Autonomous agents based on large language models increasingly perform consequential tasks on behalf of humans and other agents. Demonstrating that recorded agent behavior truthfully represents actual behavior is essential for accountability, compliance, and human oversight. This document defines a data format for verifiable agent conversation records using CDDL, with representations in both JSON and CBOR. The format captures session metadata, message exchanges, tool invocations, reasoning traces, and system events in a structured, extensible CDDL definition for verifiable agent conversation records. COSE is used as the signing method to allow for native interoperability in SCITT Transparency Services and the CDDL definition allows for seemless integration in Evidence as specified in RFC 9334. The specification supports cross-vendor interoperability by defining a common representation that accommodates translation from multiple existing agent implementations with distinct data structure layouts that are typically represented in JSON.

Table of Contents

## 1. Introduction

The question of whether the recorded output of an autonomous agent faithfully represents an agent's actual behavior has found new urgency as the number of consequential tasks that are delegated to agents increases rapidly. Autonomous Agents--typically workload instances of agentic artificial intelligence (AI) based on large language models (LLM)--interact with other actors by design. This creates an interconnected web of agent interactions and conversations that is currently rarely supervised in a systemic manner. In essence, the two main types of actors interacting with autonomous agents are humans and machines (e.g., other autonomous agents), or a mix of them. In agentic AI systems, machine actors interact with other machine actors. The number of interactions between machine actors grows significantly more than the number of interactions between human actors and machine actors. While the responsible parties for agent actions ultimately are humans--whether a natural legal entity or an organization--agents act on behalf of humans and on behalf of other agents. To demonstrate due diligence, responsible human parties require records of agent behavior to demonstrate policy compliant behavior for agents acting under their authority. These increasingly complex interactions between multiple actors that can also be triggered by machines (recursively) increase the need to understand decision making and the chain of thoughts (CoT) of autonomous agents, retroactively (i.e., accountability and auditability after the fact).

The verifiable records of agent conversations that are specified in this document provide an essential basis for operators to detect divergences between intended and actual agent behavior after the interaction has concluded.

For example:

* An agent authorized to read files might invoke tools to modify production systems or exfiltrate sensitive data beyond its authorization scope.

* An agent's visible CoT output might diverge from the reasoning that actually produced its actions.

* An agent might deliberately underperform during capability evaluations while performing at full capacity during deployment.

This document defines conversation records representing activities of autonomous agents such that long-term preservation of the evidentiary value of these records across chains of custody (CoC) is possible.

This document defines verifiable records of agent conversations as a building block towards seven dedicated goals:

1. The first goal is to assure that the recording of an agent conversation (a distinct segment of the interaction with an autonomous agent) being proffered is the same as the agent conversation that actually occurred.

2. The second goal is to provide a general structure of agent conversations that can represent most common types of agent conversation frames, is extensible, and allows for future evolution of agent conversation complexity and corresponding actor interaction.

3. The third goal is to use existing IETF building blocks to present believable evidence about how an agent conversation is recorded utilizing Evidence generation as laid out in the Remote ATtestation ProcedureS architecture [RFC9334].

4. The fourth goal is to use existing IETF building blocks to render conversation records auditable after the fact and enable non-repudiation as laid out in the Supply Chain Integrity, Transparency, and Trust architecture [I-D.ietf-scitt-architecture].

5. The fifth goal is to enable detection of behavioral anomalies in agent interactions, including unauthorized tool invocations, inconsistencies between reasoning traces and actions, and performance modulation across evaluation and deployment contexts, through structured, comparable conversation records.

6. The sixth goal is to enable cross-vendor interoperability by defining a common representation for agent conversations that can be translated from multiple existing agent implementations with distinct native formats.

7. The seventh goal is to produce records suitable for demonstrating compliance with emerging regulatory requirements for AI system documentation, traceability, and human oversight.

Most agent conversations today are represented in "human-readable" text formats. For example, JSON [STD90] is considered to be "human-readable" as it can be presented to humans in human-computer-interfaces (HCI) via off-the-shelf tools, e.g., pre-installed text editors that allow such data to be consumed or modified by humans. The Concise Binary Object Representation (CBOR [STD94]) is used as an alternative representation next to the established representation that is JSON.

In this version of the document the signing of JSON payloads is done via [STD90]. Using [STD90] enables interoperability with Transparency Services specified by the IETF [I-D.ietf-scitt-architecture] and enables low-threshold cross-application and cross-stakeholder interoperability across the Internet.

Note: further improvements in support of in-memory processing, further compacting human-readable text strings, and using CBOR as an alternative representation for verifiable records of agent conversations will follow.

### 1.1. Conventions and Definitions

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.

In this document, CDDL [RFC8610] is used to describe the data formats of agent conversations for JSON and CBOR (with no optimization for CBOR, currently).

The reader is assumed to be familiar with the vocabulary and concepts defined in [RFC9334] and [I-D.ietf-scitt-architecture].

## 2. Compliance Requirements for AI Agent Conversation Records

This section identifies the intersection of logging, traceability, and record-keeping requirements across major compliance frameworks applicable to AI systems. The verifiable agent conversation format defined in this document addresses these requirements by providing a standardized, cryptographically verifiable record of AI agent interactions.

### 2.1. Applicable Requirement Frameworks

The following frameworks were analyzed for their requirements on AI agent traceability and session logging:

| Framework | Jurisdiction | Sector | Status |
|---|---|---|---|
| EU AI Act [EU_AI_ACT_2024] | EU | Cross-sector | In force Aug 2024 |
| Cyber Resilience Act [EU_CRA_2024] | EU | Products with digital elements | In force Dec 2024 |
| NIS2 Directive [EU_NIS2_2022] | EU | Essential/ important entities | Transposed Oct 2024 |
| ETSI TS 104 223 [ETSI_TS_104223_2025] | EU/ International | AI systems | Published Apr 2025 |
| SOC 2 Trust Services Criteria [AICPA_TSC_2017] | US/ International | Service organizations | Active |
| FedRAMP [NIST_SP_800_53_R5] [OMB_M_21_31] | US | Federal cloud services | Active |
| PCI DSS [PCI_DSS_V4_0_1] | International | Payment card industry | Mandatory Mar 2025 |
| ISO/IEC 42001 [ISO_IEC_42001_2023] | International | AI management systems | Published 2023 |
| FFIEC IT Handbook [FFIEC_IT_HANDBOOK_2024] | US | Financial institutions | Updated 2024 |
| BSI AI Finance Test Criteria [BSI_AI_FINANCE_2024] | Germany | Financial sector AI | Published 2024 |
| NIST AI 100-2 [NIST_AI_100_2_E2025] | US | Cross-sector | Published 2025 |

Table 1

### 2.2. Common Requirements Intersection

The analysis of the frameworks listed above results in eleven categories of requirements that appear across ALL or MOST frameworks. These categories frame and represent a minimum baseline that verifiable agent conversation records MUST support.

#### 2.2.1. REQ-1: Automatic Event Logging

All frameworks require automatic, system-generated logging of events without reliance on manual recording. Explicit requirements are listed as follows:

| Framework | Requirement |
|---|---|
| EU AI Act Art. 12 | "High-risk AI systems shall technically allow for the automatic recording of events (logs)" |
| ETSI TS 104223 5.4.2-1 | "System Operators shall log system and user actions" |
| SOC 2 CC7.2 | "Complete and chronological record of all user actions and system responses" |
| FedRAMP AU-12 | "Audit Record Generation" control requirement |
| PCI DSS 4.0 Req 10 | "Audit logs implemented to support detection of anomalies" |
| ISO 42001 A.6.2.8 | "AI system recording of event logs" |

Table 2

Mapping to this specification: The entries array in session-trace captures all events automatically. Each entry represents a discrete, system-recorded event with structured metadata.

#### 2.2.2. REQ-2: Timestamp Requirements

Most frameworks require precise temporal information for each logged event. Explicit requirements are listed as follows:

| Framework | Requirement |
|---|---|
| EU AI Act Art. 12(2) | "Precise timestamps for each usage session" (biometric systems) |
| PCI DSS 4.0 Req 10.6 | "Time-synchronization mechanisms support consistent time settings" |
| SOC 2 | "When the activity was performed via timestamp" |
| NIS2 | "Precise logging of when an incident was first detected" |

Table 3

Mapping to this specification: The timestamp field in each entry uses abstract-timestamp which accepts both RFC 3339 strings and POSIX Seconds since Epoch, ensuring interoperability across implementations.

#### 2.2.3. REQ-3: Actor Identification

Most frameworks require some attribution of actions to identifiable actors (human or system). Explicit requirements are listed as follows:

| Framework | Requirement |
|---|---|
| EU AI Act Art. 12(3)(d) | "Identification of the natural persons involved in the verification of results" |
| SOC 2 | "The process or user who initiated the activity (Who)" |
| PCI DSS 4.0 | Attribution to "Who" performed each action |
| FedRAMP AC-2 | Account management and identification |

Table 4

Mapping to this specification: The contributor type captures actor attribution with type (human/ai/mixed/unknown) and optional model-id. Session-level agent-meta identifies the AI system.

#### 2.2.4. REQ-4: Action/Event Type Recording

All frameworks require recording of what action or event occurred. Explicit requirements are listed as follows:

| Framework | Requirement |
|---|---|
| EU AI Act Art. 12 | "Events relevant for identifying situations that may result in...risk" |
| ETSI TS 104223 5.2.4-3 | "Audit log of changes to system prompts or other model configuration" |
| SOC 2 | "The action they performed such as file transferred, created, or deleted (What)" |
| PCI DSS 4.0 | "What" component of audit trail |

Table 5

Mapping to this specification: The type field in each entry discriminates event types: user, assistant, tool-call, tool-result, reasoning, system-event.

#### 2.2.5. REQ-5: Input/Output Recording (AI-Specific)

All AI-specific frameworks require recording of inputs (prompts) and outputs (responses). Explicit requirements are listed as follows:

| Framework | Requirement |
|---|---|
| EU AI Act Art. 12(3)(c) | "The input data for which the search has led to a match" |
| PCI DSS AI Guidance | "Logging should be sufficient to audit the prompt inputs and reasoning process" |
| ETSI TS 104223 5.1.2-3 | "Operation, and lifecycle management of models, datasets and prompts" |
| FFIEC VII.D | "Lack of explainability...unclear how inputs are translated into outputs" |

Table 6

Mapping to this specification:

* message-entry (type: "user"): User/system input (prompt)

* message-entry (type: "assistant"): Model response

* tool-call-entry.input: Tool invocation parameters

* tool-result-entry.output: Tool execution results

* reasoning-entry.content: Chain-of-thought (where available)

#### 2.2.6. REQ-6: Retention Period Requirements

Most frameworks specify minimum retention periods for audit logs. Explicit requirements are listed as follows:

| Framework | Minimum Retention |
|---|---|
| EU AI Act Art. 19 | 6 months (longer for financial services) |
| FedRAMP (M-21-31) | 12 months active + 18 months cold storage |
| PCI DSS 4.0 | 12 months total, 3 months immediate access |
| NIS2 | Per member state law |

Table 7

Recommendation: Implementations SHOULD retain verifiable agent conversation records for at least 12 months to satisfy the most common requirement threshold.

#### 2.2.7. REQ-7: Tamper-Evidence and Integrity Protection

All frameworks require some protection against unauthorized modification of logs. Explicit requirements are listed as follows:

| Framework | Requirement |
|---|---|
| PCI DSS 4.0 Req 10.5 | "Tamper-proof audit trails...logs cannot be altered retroactively" |
| FedRAMP | "Effective chain of evidence to ensure integrity" |
| SOC 2 | Log integrity as security control |
| CRA | "Tamper-proof SBOMs and vulnerability disclosures" |

Table 8

Mapping to this specification: The signed-agent-record type (COSE_Sign1 envelope) provides cryptographic integrity protection. The content-hash field in trace-metadata enables verification of payload integrity.

#### 2.2.8. REQ-8: Incident Response Support

All frameworks require some logs to support incident investigation and response. Explicit requirements are listed as follows:

| Framework | Requirement |
|---|---|
| NIS2 Art. 23 | 24-hour initial notification, 72-hour assessment |
| CRA | 24-hour vulnerability notification to ENISA |
| ETSI TS 104223 5.4.2-1 | Logs for "incident investigations, and vulnerability remediation" |
| FedRAMP | Incident reporting and continuous monitoring |

Table 9

Mapping to this specification: The structured format enables rapid extraction of relevant entries by timestamp range, event type, or tool invocation for incident reconstruction.

### 2.2.9. REQ-9: Anomaly and Risk Detection Support

All frameworks require some logs to enable detection of anomalous or risky behavior. Explicit requirements are listed as follows:

| Framework | Requirement |
|---|---|
| EU AI Act Art. 12(2)(a) | "Identifying situations that may result in...risk" |
| ETSI TS 104223 5.4.2-2 | "Detect anomalies, security breaches, or unexpected behaviour" |
| FedRAMP SI-4 | "Anomaly detection" |
| SOC 2 | "Anomaly detection" for security monitoring |

Table 10

Mapping to this specification: The standardized entry types and structured tool-call/tool-result pairs enable automated analysis for detecting:

* Unusual tool invocation patterns

* Failed operations (via status and is-error fields)

* Unexpected reasoning patterns

* Token usage anomalies

#### 2.2.10. REQ-10: Human Oversight Enablement

All AI-specific frameworks require logs to support human review and oversight. Explicit requirements are listed as follows:

| Framework | Requirement |
|---|---|
| EU AI Act Art. 26(5) | "Monitoring the operation of high-risk AI systems" |
| ETSI TS 104223 5.1.4-1 | "Capabilities to enable human oversight" |
| ISO 42001 | Human responsibility and accountability |
| FFIEC VII.D | "Dynamic updating...challenges to monitoring and independently reviewing AI" |

Table 11

Mapping to this specification: The reasoning-entry type captures chain-of-thought content (where available), enabling human reviewers to understand AI decision-making processes. The hierarchical children field preserves conversation structure.

#### 2.2.11. REQ-11: Traceability and Reproducibility

All frameworks require some ability to trace system behavior and reconstruct events. Explicit requirements are listed as follows:

| Framework | Requirement |
|---|---|
| EU AI Act Art. 12 | "Level of traceability of the functioning...appropriate to the intended purpose" |
| ISO 42001 | "Traceability" as key factor including "data provenance, model traceability" |
| CRA | "Traceability in the software supply chain" |
| ETSI TS 104223 5.2.1-2 | "Track, authenticate, manage version control" |

Table 12

Mapping to this specification:

* session-id: Links entries to sessions

* entry-id and parent-id: Enables conversation tree reconstruction

* vcs-context: Git commit/branch for code state

* agent-meta: Model version and CLI version

* file-attribution: Code provenance tracking

## 2.3. Framework-Specific Requirements

### 2.3.1. EU AI Act (High-Risk Systems)

For AI systems classified as high-risk under Annex III of the EU AI Act [EU_AI_ACT_2024], additional requirements apply:

1. Biometric identification systems (Annex III, 1(a)) require logging of:

    * Precise timestamps for start/end of each usage session

    * Reference database used during input data validation

    * Input data leading to matches

    * Natural persons involved in result verification

2. Log retention: Minimum 6 months; financial services may require longer per sector-specific regulation.

3. Authority Access: Art. 19 of [EU_AI_ACT_2024] requires provision of logs to competent authorities upon reasoned request.

#### 2.3.2. ETSI TS 104 223 Session Logging Requirements

ETSI TS 104 223 [ETSI_TS_104223_2025] provides the most detailed AI-specific logging requirements:

| Provision | Requirement | This Spec Mapping |
|---|---|---|
| 5.1.2-3 | Audit trail for "operation, and lifecycle management of models, datasets and prompts" | session-trace, agent-meta |
| 5.2.4-1 | "Document and maintain a clear audit trail of their system design" | recording-agent, open maps |
| 5.2.4-3 | "Audit log of changes to system prompts or other model configuration" | event-entry with prompt changes |
| 5.4.2-1 | "Log system and user actions to support security compliance, incident investigations" | entries array |
| 5.4.2-2 | "Analyse their logs to ensure...desired outputs and to detect anomalies" | Structured format enables analysis |
| 5.4.2-3 | "Monitor internal states of their AI systems" | reasoning-entry, token-usage |

Table 13

#### 2.3.3. PCI DSS AI-Specific Guidance

The PCI Security Standards Council has published guidance on AI in payment environments [PCI_DSS_V4_0_1]: "Where possible, logging should be sufficient to audit the prompt inputs and reasoning process used by the AI system that led to the output provided."

This specification directly addresses this requirement through:

* message-entry (type: "user"): Captures prompt inputs

* reasoning-entry: Captures chain-of-thought (where available)

* message-entry (type: "assistant"): Captures model outputs

* tool-call-entry / tool-result-entry: Captures agentic actions

#### 2.3.4. Financial Sector Requirements (FFIEC, BSI)

Financial institutions face additional scrutiny for AI systems per [FFIEC_IT_HANDBOOK_2024] and [BSI_AI_FINANCE_2024]:

| Requirement Area | FFIEC | BSI AI Finance |
|---|---|---|
| Explainability | "Lack of transparency or explainability" risk | Test criteria for explainability |
| Dynamic updating | "Challenges to monitoring and independently reviewing AI" | Continuous validation |
| Audit trail | Log management (VI.B.7) | Complete audit trail |

Table 14

### 2.4. Compliance Mapping Table

The following table maps this specification’s data elements to compliance requirements:

| Data Element | EU AI Act | ETSI 104223 | SOC 2 | FedRAMP | PCI DSS | ISO 42001 | NIS2 |
|---|---|---|---|---|---|---|---|
| timestamp | Art. 12(2) | 5.4.2-1 | CC7.2 | AU-8 | 10.6 | A.6.2.8 | Art. 23 |
| session-id | Art. 12 | 5.2.4-1 | CC7.2 | AU-3 | 10.2 | A.6.2.8 | - |
| entry.type | Art. 12(2) | 5.4.2-1 | CC7.2 | AU-3 | 10.2 | A.6.2.8 | - |
| contributor | Art. 12(3)(d) | 5.1.4 | CC6.1 | AC-2 | 10.2 | A.6.2.8 | - |
| message-entry.content | Art. 12(3)(c) | 5.1.2-3 | - | - | AI Guide | - | - |
| reasoning-entry | Art. 12 | 5.4.2-3 | - | - | AI Guide | A.7.1 | - |
| tool-call-entry / tool-result-entry | Art. 12 | 5.4.2-1 | CC7.2 | AU-12 | 10.2 | A.6.2.8 | - |
| signed-agent-record | Art. 19 | 5.2.4-1.2 | CC6.1 | AU-9 | 10.5 | - | - |
| vcs-context | - | 5.2.1-2 | - | CM-3 | - | A.6.2.8 | - |
| token-usage | - | 5.4.2-4 | - | - | - | - | - |

Table 15

## 2.5. Security Requirements addressing Regulatory Compliance

### 2.5.1. Log Integrity

Per PCI DSS [PCI_DSS_V4_0_1] Req 10.5 and FedRAMP [NIST_SP_800_53_R5] AU-9, logs MUST be protected against modification. Implementations SHOULD:

1. Use the signed-agent-record envelope for cryptographic integrity

2. Store the content-hash for offline verification

3. Implement write-once storage for log archives

#### 2.5.2. Access Control

Per FedRAMP [NIST_SP_800_53_R5] AC-3 and ETSI [ETSI_TS_104223_2025] 5.2.2-1, access to logs MUST be controlled:

1. Logs containing sensitive prompts or outputs require access control

2. Reasoning content may contain confidential information

3. Authority access (EU AI Act Art. 19) requires audit of log access itself

#### 2.5.3. Data Protection

Logs may contain personal data subject to GDPR/privacy regulations:

1. message-entry.content may contain PII

2. tool-result-entry.output may contain query results with PII

3. Retention periods must balance compliance requirements with data minimization

### 2.6. Related Standards and Emerging Work

The ISO/IEC JTC 1/SC 42 committee is developing [ISO_IEC_DIS_24970], a draft international standard specifically addressing AI system logging requirements. This work aligns with and complements the verifiable agent conversation format defined in this document.

Research on emergent misalignment [EMERGENT_MISALIGNMENT_2025] demonstrates that narrow fine-tuning can produce broadly misaligned LLMs, underscoring the importance of comprehensive conversation logging for detecting behavioral anomalies that may not be apparent from individual interactions.

## 3. CDDL Definitions for Agent Conversations Records

This section defines each complex data type specified in the verifiable agent conversation record CDDL data definition. Each subsection illustrates a CDDL fragment for one type, a brief description, and per-member documentation.

### 3.1. Common Types

The schema uses the following generic type definitions throughout.

abstract-timestamp: An RFC 3339 date-time string or numeric epoch milliseconds. New implementations SHOULD use RFC 3339 strings; consumers MUST accept both forms.

session-id: An opaque string uniquely identifying a conversation session. Values may be UUID v4, UUID v7, SHA-256 hashes, or other formats.

entry-id: A per-entry unique reference within a session, enabling parent-child linking and tool call-result correlation.

### 3.2. The verifiable-agent-record Map

The CDDL definition for the verifiable-agent-record map is specified as follows:

```
verifiable-agent-record = {
    version: tstr
    id: tstr
    session: session-trace
    ? created: abstract-timestamp
    ? file-attribution: file-attribution-record
    ? vcs: vcs-context
    ? recording-agent: recording-agent
    * tstr => any
}
```

The verifiable-agent-record is the top-level container for all data produced by or about an agent conversation. It unifies two complementary perspectives: the session trace captures how code was produced (the full conversation replay), while file attribution captures what code was produced (which files were modified and by whom).

The following describes each member of this map.

version: The schema version string following semantic versioning (e.g., "3.0.0-draft"). Consumers use this field to select the appropriate parser or validation logic.

id: A unique identifier for this record, typically a UUID. Distinguishes records when multiple are stored or transmitted together.

session: The full conversation trace including all entries, tool calls, reasoning steps, and system events.

created: The timestamp when this record was generated. Distinct

from session timestamps, which record when the conversation occurred.

file-attribution: Structured data about which files were modified and which line ranges were written by the agent.

vcs: Version control metadata at the record level (repository, branch, revision).

recording-agent: Identifies the tool or agent that generated this record, as distinct from the agent that conducted the conversation.

### 3.3. The session-trace Map

The CDDL definition for the session-trace map is specified as follows:

```
session-trace = {
    ? format: tstr
    session-id: session-id
    ? session-start: abstract-timestamp
    ? session-end: abstract-timestamp
    agent-meta: agent-meta
    ? environment: environment
    entries: [* entry]
    * tstr => any
}
```

A session-trace captures the full conversation between a user and an autonomous agent. It contains an ordered array of entries representing messages, tool invocations, reasoning steps, and system events. The session-trace preserves the complete interaction history including all native agent metadata, enabling both replay and audit of the conversation.

The following describes each member of this map.

format: Classifies the session style, such as "interactive" (human-in-the-loop) or "autonomous" (fully automated). Informative only; does not change the structure.

session-id: Unique identifier for this session.

session-start: When the session began.

session-end: When the session ended.

agent-meta: Metadata about the agent and model that conducted this conversation.

environment: Execution environment context, such as working directory and version control state.

entries: An ordered array of conversation entries.

### 3.4. The agent-meta Map

The CDDL definition for the agent-meta map is specified as follows:

```
agent-meta = {
    model-id: tstr
    model-provider: tstr
    ? models: [* tstr]
    ? cli-name: tstr
    ? cli-version: tstr
    * tstr => any
}
```

The agent-meta type identifies the coding agent and language model used during a conversation session. Agent identification is essential for provenance tracking: knowing which model produced which output enables auditing, capability assessment, and compliance verification.

The following describes each member of this map.

model-id: The primary language model identifier, using the naming convention of the provider (e.g., "claude-opus-4-5-20251101", "gemini-2.0-flash").

model-provider: The provider of the primary model (e.g., "anthropic", "google", "openai").

models: List of all model identifiers used during the session. Relevant for multi-model sessions where the agent switches between models.

cli-name: The name of the CLI tool or agent framework (e.g., "claude-code", "gemini-cli", "codex-cli").

cli-version: The version of the CLI tool.

### 3.5. The recording-agent Map

The CDDL definition for the recording-agent map is specified as follows:

```
recording-agent = {
    name: tstr
    ? version: tstr
    * tstr => any
}
```

The recording-agent map identifies the tool that generated this verifiable agent record, as distinct from the agent that conducted the conversation. This distinction matters for provenance chains: the recording tool's version affects how native data is translated into the canonical CDDL data definition specified in this document.

The following describes each member of this map.

name: The name of the recording tool or agent.

version: The version of the recording tool.

### 3.6. The environment Map

The CDDL definition for the environment map is specified as follows:

```
environment = {
    working-dir: tstr
    ? vcs: vcs-context
    ? sandboxes: [* tstr]
    * tstr => any
}
```

The environment type captures execution context for the conversation: where the agent was running, what version control state was active, and whether sandboxing was in effect. This context is important for reproducibility and for understanding the scope of file modifications.

The following describes each member of this map.

working-dir: The primary working directory path. File paths in tool calls are typically relative to this directory.

vcs: Version control state at the time of the session (branch, revision, etc.).

sandboxes: Paths to sandbox mount points. Some agents run in sandboxed environments where the working directory is a temporary mount.

### 3.7. The vcs-context Map

The CDDL definition for the vcs-context map is specified as follows:

```
vcs-context = {
    type: tstr
    ? revision: tstr
    ? branch: tstr
    ? repository: tstr
    * tstr => any
}
```

The vcs-context type captures version control metadata for reproducibility. Knowing the exact repository, branch, and commit at the time of a conversation enables consumers to reconstruct the codebase state and verify file attributions.

The following describes each member of this map.

type: The version control system type (e.g., "git", "jj", "hg", "svn").

revision: The commit SHA or change identifier at session time.

branch: The active branch name.

repository: The repository URL.

### 3.8. Entry Types

The CDDL definition specifies five entry types representing the different kinds of events in an agent conversation. Each type uses a type field as the discriminator. All entry types support optional children for hierarchical nesting and * tstr => any for preserving native agent fields that do not map to canonical fields.

```
entry = message-entry
      / tool-call-entry
      / tool-result-entry
      / reasoning-entry
      / event-entry
```

#### 3.8.1. The message-entry Map

The CDDL definition for the message-entry map is specified as follows:

```
message-entry = {
    type: "user" / "assistant"
    ? content: any
    ? timestamp: abstract-timestamp
    ? id: entry-id
    ? model-id: tstr
    ? parent-id: entry-id
    ? token-usage: token-usage
    ? children: [* entry]
    * tstr => any
}
```

A message-entry represents a conversational turn: either human input (type: "user") or agent response (type: "assistant"). This is the most common entry type, carrying the primary dialogue content of a session. For assistant messages, additional metadata may be present: the model that generated the response and token usage statistics.

The following describes each member of this map.

type: The message direction. "user" indicates human (or upstream agent) input; "assistant" indicates the agent's response.

content: The message body. May be a plain text string, an array of typed content parts, or absent when the agent places content exclusively in child entries.

timestamp: When this message was produced.

id: Unique identifier for this entry within the session.

model-id: The model that generated this response. Present on assistant entries; absent on user entries.

parent-id: References the parent entry, enabling tree-structured conversations.

token-usage: Token consumption metrics for this response.

children: Nested entries within this message. Used when the native format embeds tool calls or reasoning blocks inside an assistant message.

### 3.8.2. The tool-call-entry Map

The CDDL definition for the tool-call-entry map is specified as follows:

```
tool-call-entry = {
    type: "tool-call"
    name: tstr
    input: any
    ? call-id: tstr
    ? timestamp: abstract-timestamp
    ? id: entry-id
    ? children: [* entry]
    * tstr => any
}
```

A tool-call-entry represents a tool invocation: which tool was called and with what arguments. Tool calls are central to agent conversation records because tool use is the primary mechanism by which agents interact with external environment.

The following describes each member of this map.

type: Fixed discriminator value "tool-call".

name: The tool name (e.g., "Bash", "Edit", "Read", "apply_patch").

input: The arguments passed to the tool, preserved in their native structure.

call-id: Links this call to its corresponding result.

timestamp: When this tool call occurred.

id: Unique identifier for this entry.

children: Nested entries within this tool call.

### 3.8.3. The tool-result-entry Map

The CDDL definition for the tool-result-entry map is specified as follows:

```
tool-result-entry = {
    type: "tool-result"
    output: any
    ? call-id: tstr
    ? status: tstr
    ? is-error: bool
    ? timestamp: abstract-timestamp
    ? id: entry-id
    ? children: [* entry]
    * tstr => any
}
```

A tool-result-entry represents the output returned by a tool after execution. It is linked to its corresponding tool-call-entry via the call-id field. The result carries the tool's response data and optional status metadata indicating success or failure.

The following describes each member of this map.

type: Fixed discriminator value "tool-result".

output: The tool's response, preserved in native structure.

call-id: Links this result to its corresponding call.

status: Outcome status of the tool execution, such as "success", "error", or "completed".

is-error: Boolean error flag, present when the tool execution failed.

timestamp: When this tool result was returned.

id: Unique identifier for this entry.

children: Nested entries within this tool result.

#### 3.8.4. The reasoning-entry Map

The CDDL definition for the reasoning-entry map is specified as follows:

```
reasoning-entry = {
    type: "reasoning"
    content: any
    ? encrypted: tstr
    ? subject: tstr
    ? timestamp: abstract-timestamp
    ? id: entry-id
    ? children: [* entry]
    * tstr => any
}
```

A reasoning-entry captures CoT, thinking, or internal reasoning content from the agent. Not all agents expose reasoning traces; when they do, the content may be plaintext, structured blocks, or encrypted. Reasoning entries are valuable for auditing decision-making processes and understanding why an agent took particular actions.

The following describes each member of this map.

type: Fixed discriminator value "reasoning".

content: The reasoning text or structured content. May be an empty string when only encrypted content is available.

encrypted: Encrypted reasoning content. Used where the model provider encrypts chain-of-thought output.

subject: A topic label for the reasoning block.

timestamp: When this reasoning was produced.

id: Unique identifier for this entry.

children: Nested entries within the reasoning block.

#### 3.8.5. The event-entry Map

The CDDL definition for the event-entry map is specified as follows:

```
event-entry = {
    type: "system-event"
    event-type: tstr
    ? data: { * tstr => any }
    ? timestamp: abstract-timestamp
    ? id: entry-id
    ? children: [* entry]
    * tstr => any
}
```

An event-entry records system lifecycle events that are not part of the conversation dialogue but are relevant for understanding the session context. Examples include session start/end markers, token usage summaries, permission changes, and configuration events.

The following describes each member of this map.

type: Fixed discriminator value "system-event".

event-type: Classifies the event, such as "session-start", "session-end", "token-count", or "permission-change". The values are not enumerated in the schema to accommodate vendor-specific event types.

data: Event-specific payload. The structure varies by event type.

timestamp: When this event occurred.

id: Unique identifier for this entry.

children: Nested entries within this event.

## 3.9. The token-usage Map

The CDDL definition for the token-usage map is specified as follows:

```
token-usage = {
    ? input: uint
    ? output: uint
    ? cached: uint
    ? reasoning: uint
    ? total: uint
    ? cost: number
    * tstr => any
}
```

The token-usage type captures token consumption metrics for a model response. Token usage data is essential for cost tracking, quota management, and understanding model behavior. All fields are optional because different agents report different subsets of token metrics.

Editor’s Note: add a non-empty generic here

The following describes each member of this map.

input: The number of input tokens consumed by this response.

output: The number of output tokens generated.

cached: The number of input tokens served from cache rather than reprocessed.

reasoning: Tokens consumed by chain-of-thought or reasoning computation.

total: The total token count.

cost: The monetary cost in US dollars for this response.

### 3.10. File Attribution Types

NOTE: This section is specified but not yet validated against real session data. Implementation is pending.

File attribution captures what code was produced: which files were modified, which line ranges were changed, and who authored them.

#### 3.10.1. The file-attribution-record Map

The CDDL definition for the file-attribution-record map is specified as follows:

```
file-attribution-record = {
    files: [* file]
}
```

The file-attribution-record is the top-level container for file attribution data. It holds an array of files, each with their attributed line ranges and contributor information.

The following describes each member of this map.

files: Array of files with attributed ranges.

### 3.10.2. The file Map

The CDDL definition for the file map is specified as follows:

```
file = {
    path: tstr
    conversations: [* conversation]
}
```

A file represents a single source file that was modified during the conversation. It groups all conversations that contributed changes to this file.

The following describes each member of this map.

path: The file path relative to the repository root.

conversations: The conversations that contributed modifications to this file.

### 3.10.3. The conversation Map

The CDDL definition for the conversation map is specified as follows:

```
conversation = {
    ? url: tstr .regexp uri-regexp
    ? contributor: contributor
    ranges: [* range]
    ? related: [* resource]
}
```

A conversation links a specific session to the line ranges it produced in a file. This enables tracing from a line of code back to the conversation that generated it.

The following describes each member of this map.

url: A URL pointing to the conversation source (e.g., a web UI permalink).

contributor: The default contributor for all ranges in this conversation. Can be overridden per-range.

ranges: The line ranges in the file that were produced by this conversation.

related: External resources related to this conversation (e.g., issue trackers, documentation).

#### 3.10.4. The range Map

The CDDL definition for the range map is specified as follows:

```
range = {
    start-line: uint
    end-line: uint
    ? content-hash: tstr
    ? content-hash-alg: tstr
    ? contributor: contributor
}
```

A range identifies a contiguous block of lines in a file that were produced by a specific conversation. Line numbers are 1-indexed and inclusive. The optional content hash enables position-independent tracking when lines move due to later edits.

The following describes each member of this map.

start-line: The first line of the range (1-indexed).

end-line: The last line of the range (1-indexed, inclusive).

content-hash: A hash of the range content for position-independent identification.

content-hash-alg: The hash algorithm used (default: "sha-256").

contributor: Overrides the conversation-level contributor for this specific range.

#### 3.10.5. The contributor Map

The CDDL definition for the contributor map is specified as follows:

```
contributor = {
    type: "human" / "ai" / "mixed" / "unknown"
    ? model-id: tstr
}
```

A contributor identifies who authored a range of code. The type field distinguishes between human-authored, AI-generated, mixed, and unknown authorship.

The following describes each member of this map.

type: The authorship category.

model-id: The model identifier for AI-authored ranges.

#### 3.10.6. The resource Map

The CDDL definition for the resource map is specified as follows:

```
resource = {
    type: tstr
    url: tstr .regexp uri-regexp
}
```

A resource represents an external reference related to a conversation, such as an issue tracker entry, a pull request, or a documentation page.

The following describes each member of this map.

type: The resource type (e.g., "issue", "pr", "documentation").

url: The URL of the resource.

### 3.11. Signing Envelope Types

The signing envelope provides cryptographic integrity protection for verifiable agent records using COSE_Sign1 ([STD96]).

#### 3.11.1. The signed-agent-record Structure

The CDDL definition for a signed-agent-record structure is specified as follows:

```
signed-agent-record = #6.18([
    protected: bstr .cbor protected-header
    unprotected: unprotected-header
    payload: bstr / null
    signature: bstr
])
```

A signed-agent-record is a COSE_Sign1 envelope (CBOR Tag 18) that wraps a verifiable agent record with a cryptographic signature. Signing provides data provenance and tamper evidence, satisfying some requirements of RATS Evidence generation ([RFC9334]) and SCITT auditability ([I-D.ietf-scitt-architecture]) requirements. The payload may be included or detached (null); in detached mode, the record is supplied separately during verification.

The following describes each element of this structure.

protected: The serialized protected header containing the algorithm identifier, content type, and CWT claims.

unprotected: The unprotected header carrying trace metadata at label 100.

payload: The serialized record bytes, or null for detached payloads.

signature: The cryptographic signature over the protected header and payload.

#### 3.11.2. The trace-metadata Map

The CDDL definition for the trace-metadata map is specified as follows:

```
trace-metadata = {
    session-id: session-id
    agent-vendor: tstr
    trace-format: trace-format-id
    timestamp-start: abstract-timestamp
    ? timestamp-end: abstract-timestamp
    ? content-hash: tstr
    ? content-hash-alg: tstr
}
```

The trace-metadata type carries summary information about the signed record in the COSE_Sign1 unprotected header. This enables consumers to inspect key properties of a signed record without deserializing the full payload.

The following describes each member of this map.

session-id: The session identifier from the signed record.

agent-vendor: The agent provider name (e.g., "anthropic", "google").

trace-format: Identifies the format of the signed payload (e.g., "ietf-vac-v3.0" for canonical records).

timestamp-start: When the session began.

timestamp-end: When the session ended.

content-hash: SHA-256 hex digest of the payload bytes, enabling integrity checking independent of the COSE signature.

content-hash-alg: The hash algorithm used (default: "sha-256").

## 4. Collated CDDL Definition for generic Agent Conversations

```
start = verifiable-agent-record / signed-agent-record

; RFC 3339 string OR epoch milliseconds (for interop).
abstract-timestamp = tstr .regexp date-time-regexp / uint

; Opaque string: UUID, SHA-256 hash in base64url, etc.
session-id = tstr / bstr

; Per-entry unique reference within a session.
entry-id = tstr

; RFC 3339 date-time pattern
date-time-regexp = "([0-9]{4})-(0[1-9]|1[0-2])-(0[1-9]|[12][0-9]|3[01])T([01][0-9]|2[0
-3]):([0-5][0-9]):(60|[0-5][0-9])([.][0-9]+)?(Z|[+-]([01][0-9]|2[0-3]):[0-5][0-9])"

; URI pattern (RFC 3986)
uri-regexp = "(([^:/?#]+):)?(//([^/?#]*))?([^?#]*)(\\?([^#]*))?(#(.*))?"

verifiable-agent-record = {
    version: tstr                       ; Schema version (semver)
    id: tstr                            ; Record identifier
    session: session-trace              ; Conversation trace (required)
    ? created: abstract-timestamp       ; Record creation time
    ? file-attribution: file-attribution-record
    ? vcs: vcs-context                  ; Record-level VCS context
    ? recording-agent: recording-agent  ; Tool that generated this record
    * tstr => any
}

session-trace = {
    ? format: tstr                      ; "interactive" / "autonomous" / vendor
    session-id: session-id
    ? session-start: abstract-timestamp
    ? session-end: abstract-timestamp
    agent-meta: agent-meta
    ? environment: environment
    entries: [* entry]
    * tstr => any
}

agent-meta = {
    model-id: tstr                      ; e.g., "claude-opus-4-5-20251101"
    model-provider: tstr                ; e.g., "anthropic", "google"
    ? models: [ * tstr ]                  ; All models (multi-model sessions)
    ? cli-name: tstr                    ; e.g., "claude-code", "gemini-cli"
    ? cli-version: tstr
    * tstr => any
```

```
   }

   recording-agent = {
       name: tstr
       ? version: tstr
       * tstr => any
   }

   environment = {
       working-dir: tstr
       ? vcs: vcs-context
       ? sandboxes: [ * tstr ]              ; Sandbox mount paths
       * tstr => any
   }

   vcs-context = {
       type: tstr                          ; "git" / "jj" / "hg" / "svn"
       ? revision: tstr                    ; Commit SHA or change ID
       ? branch: tstr
       ? repository: tstr                  ; Repository URL
       * tstr => any
   }

   entry = message-entry
         / tool-call-entry
         / tool-result-entry
         / reasoning-entry
         / event-entry

   message-entry = {
       type: "user" / "assistant"
       ? content: any                      ; Text string or structured content blocks
       ? timestamp: abstract-timestamp
       ? id: entry-id
       ? model-id: tstr                    ; Model (assistant only)
       ? parent-id: entry-id              ; Parent message reference
       ? token-usage: token-usage
       ? children: [ * entry ]
       * tstr => any
   }

   tool-call-entry = {
       type: "tool-call"
       name: tstr                          ; Tool name (e.g., "Bash", "Edit", "Read")
       input: any                          ; Tool arguments
       ? call-id: tstr                     ; Links call  result
       ? timestamp: abstract-timestamp
       ? id: entry-id
```

```
    ? children: [ * entry ]
    * tstr => any
}

tool-result-entry = {
    type: "tool-result"
    output: any                         ; Tool output
    ? call-id: tstr                     ; Links call  result
    ? status: tstr                      ; "success" / "error" / "completed"
    ? is-error: bool
    ? timestamp: abstract-timestamp
    ? id: entry-id
    ? children: [ * entry ]
    * tstr => any
}

reasoning-entry = {
    type: "reasoning"
    content: any                        ; Plaintext reasoning or structured
    ? encrypted: tstr                   ; Encrypted content (provider-protected)
    ? subject: tstr                     ; Topic label
    ? timestamp: abstract-timestamp
    ? id: entry-id
    ? children: [ * entry ]
    * tstr => any
}

event-entry = {
    type: "system-event"
    event-type: tstr                    ; Event classifier
    ? data: { * tstr => any }           ; Event-specific payload
    ? timestamp: abstract-timestamp
    ? id: entry-id
    ? children: [ * entry ]
    * tstr => any
}

token-usage = {
    ? input: uint                       ; Input tokens
    ? output: uint                      ; Output tokens
    ? cached: uint                      ; Cached input tokens
    ? reasoning: uint                   ; Reasoning/thinking tokens
    ? total: uint                       ; Total tokens
    ? cost: number                      ; Dollar cost
    * tstr => any
}

file-attribution-record = {
```

```
    files: [* file]
}

file = {
    path: tstr                          ; Relative path from repo root
    conversations: [* conversation]
}

conversation = {
    ? url: tstr .regexp uri-regexp
    ? contributor: contributor          ; Default contributor for ranges
    ranges: [* range]
    ? related: [* resource]
}

range = {
    start-line: uint                    ; 1-indexed
    end-line: uint                      ; 1-indexed, inclusive
    ? content-hash: tstr
    ? content-hash-alg: tstr            ; Default: "sha-256"
    ? contributor: contributor          ; Override for this range
}

contributor = {
    type: "human" / "ai" / "mixed" / "unknown"
    ? model-id: tstr
}

resource = {
    type: tstr
    url: tstr .regexp uri-regexp
}

; SCITT-interoperable COSE Envelope
; including from draft-ietf-cose-merkle-tree-proofs and
; draft-ietf-scitt-architecture for validation

signed-agent-record = #6.18([           ; COSE_Sign1 tag
    protected: bstr .cbor protected-header ; {alg, content-type, scitt-stuff}
    unprotected: unprotected-header     ; Trace metadata
    payload: bstr / null                ; Detached if null
    signature: bstr
])

protected-header = {
  &(CWT_Claims: 15) => CWT_Claims
  ? &(alg: 1) => int
  ? &(content_type: 3) => tstr / uint
```

```
    ? &(kid: 4) => bstr
    ? &(x5t: 34) => COSE_CertHash
    ? &(x5chain: 33) => COSE_X509
    * label => any
  }

  CWT_Claims = {
    &(iss: 1) => tstr
    &(sub: 2) => tstr
    * label => any
  }

  unprotected-header = {
    ? &(trace-metadata-key: 100) => trace-metadata ; 100 is placeholder
    ? &(x5chain: 33) => COSE_X509
    ? &(receipts: 394)  => [ + Receipt ]
    * label => any
  }

  trace-metadata = {
      session-id: session-id
      agent-vendor: tstr
      trace-format: trace-format-id
      timestamp-start: abstract-timestamp
      ? timestamp-end: abstract-timestamp
      ? content-hash: tstr                ; SHA-256 hex digest of payload
      ? content-hash-alg: tstr
  }

  ; Known values: "ietf-vac-v3.0" (canonical), "claude-jsonl", "gemini-json",
  ; "codex-jsonl", "opencode-json", "cursor-jsonl". Extensible via tstr.
  trace-format-id = tstr

  COSE_X509 = bstr / [ 2*certs: bstr ]
  COSE_CertHash = [ hashAlg: (int / tstr), hashValue: bstr ]
  label = int / tstr

  ; COSE Receipt CDDL for use in SCITT compliant COSE Envelope

  Receipt = #6.18(COSE_Sign1)

  cose-label = int / tstr
  cose-value = any

  Protected_Header = {
    * cose-label => cose-value
  }
```

```
Unprotected_Header = {
  &(receipts: 394)  => [+ bstr .cbor Receipt]
  * cose-label => cose-value
}

COSE_Sign1 = [
  protected   : bstr .cbor Protected_Header,
  unprotected : Unprotected_Header,
  payload     : bstr / null,
  signature   : bstr
]
```

Figure 1: CDDL definition of an Agent Conversation Record

## 5. Privacy Considerations

Verifiable agent conversation records reveal substantial information about agent behavior, system state, and user interactions. The privacy considerations of [RFC6973] apply.

### 5.1. Information Disclosure

User prompts captured in message entries may contain personal identifiers, business confidential information, credentials inadvertently included in prompts, or behavioral patterns. Agent responses and reasoning traces may reveal inference results that expose information about users not explicitly provided, confidential information retrieved by tools, or system architecture details through tool names and parameters. Implementations MUST treat user-provided content as potentially containing personally identifiable information.

Record metadata exposes operational details that may have privacy implications. Model identifiers reveal AI capabilities, CLI versions enable targeted attacks against known vulnerabilities, working directories expose file system structure, and VCS context discloses repository names and commit timing. Token usage patterns may enable inference about conversation content even when the content itself is protected.

### 5.2. Sensitive Content in Records

Reasoning entries may contain inferences that constitute special category data, including health-related inferences from user queries, political opinions derived from conversation context, or biometric data processed by AI systems. The reasoning-entry.encrypted field reflects that some model providers encrypt chain-of-thought content; when reasoning is encrypted, audit capabilities depend on the

provider’s cooperation.

Tool inputs and outputs warrant particular attention. Tool call inputs may contain credentials, API keys, or file contents. Tool results may contain query results with personal data from databases or external services. Implementations SHOULD provide configurable redaction rules for common patterns and support selective entry type recording based on deployment requirements.

### 5.3. Retention Considerations

Multiple frameworks impose retention requirements that may conflict with data minimization principles. Organizations must balance compliance obligations requiring extended retention against privacy principles requiring timely deletion.

Compliant data management may require separating raw personal data from audit trail metadata, implementing automated deletion for personal data after compliance-minimum periods, and maintaining cryptographic commitments enabling verification without retaining content.

### 5.4. Correlation and Inference

Presentations of the same record to multiple parties can be correlated by matching on the signature component. Session identifiers enable linking of records across time, potentially revealing long-term behavioral patterns or organizational structure. Implementations SHOULD use unlinkable session identifiers where correlation is not required.

Without accessing record content, observers may infer conversation frequency and duration patterns, types of tools used, error rates, or timing correlations with external events. Metadata exposure in unprotected headers warrants careful consideration; the trace-metadata in the COSE unprotected header reveals session identifiers, agent vendor, and timestamps even when the payload is encrypted.

### 5.5. Authority Access

Regulatory frameworks may require provision of records to authorities upon request. Access to records has to be logged, capturing who accessed which records, when access occurred, what legal basis justified access, and what data was disclosed. This creates a recursive consideration: access logs may themselves contain personal data requiring protection.

# 6. Security Considerations

## 6.1. Record Integrity

The signed-agent-record envelope provides cryptographic integrity protection for the serialized record payload. A valid signature establishes that the claimed signer produced the record but does not guarantee the truthfulness of its contents. Modifications to the record structure after signing invalidate the signature, but modifications before signing cannot be detected. Implementations generating records incrementally during a conversation MUST sign only after the conversation concludes or at defined checkpoints.

## 6.2. Signing Key Protection

The security of signed records depends critically on the protection of private signing keys. If an attacker obtains a signing key, they can forge records indistinguishable from genuine ones. Protection mechanisms range from operating system process isolation in development environments to hardware security modules with physical tampering resistance in high-assurance deployments. Compromised signing keys require rejection of records signed after the compromise date and notification to Relying Parties.

Key provisioning processes must guarantee that exclusively valid attestation key material is established. Off-device key generation requires confidentiality protection during transmission, creating recursive security dependencies. On-device key generation eliminates transmission risks but requires chain-of-custody integrity to prevent attackers from obtaining endorsement for keys they control.

## 6.3. Timestamp Integrity

Timestamps establish temporal ordering of events within records. Attackers who can manipulate timestamps can backdate records, freeze participants in chosen time periods to evade freshness checks, or manipulate perceived temporal relationships between entries. Timestamps within records are attested by the signer, not independently verified. Implementations requiring independent timestamp verification SHOULD use external timestamping services or transparency logs such as those defined in [I-D.ietf-scitt-architecture].

### 6.4. Adversarial Content

Processing verifiable agent conversation records involves parsing content that may be produced by adversaries. This applies both to user-supplied prompts and to model outputs that may have been influenced by adversarial inputs. The open extensibility (* tstr => any) in record types allows arbitrary additional fields; implementations MUST NOT assume that unrecognized fields are safe to process or display.

Records may contain content designed to exploit downstream systems. Malicious prompts preserved in records may execute if records are subsequently processed by AI systems. Tool results may contain executable content that executes if rendered unsafely in web interfaces. File paths in tool calls or file attribution entries may attempt directory traversal. Implementations MUST apply appropriate sanitization before rendering content, executing it in agent contexts, or using paths for file system operations.

### 6.5. Non-Repudiation

The signed-agent-record envelope alone provides authenticity and integrity, not non-repudiation. A signer can claim key compromise or dispute the signing time without independent evidence. Non-repudiation requires additional infrastructure such as transparency services [I-D.ietf-scitt-architecture] that provide independent timestamp proof via registration receipts, append-only logs preventing retroactive denial, and third-party witnesses to the signing event.

### 6.6. Detection and Trust Boundaries

Verifiable agent conversation records primarily enable detection of anomalous behavior rather than prevention. Detection requires that records accurately reflect actual agent behavior; an agent that controls its own recording can omit or falsify entries. The recording-agent field distinguishes the recording tool from the conversing agent, enabling Relying Parties to assess trust in the recording process.

Trust boundaries exist between the agent runtime and recording system, between the recording system and storage, between storage and verification, and between verification and decision-making. Attacks at any boundary may compromise record integrity or confidentiality.

## 7. IANA Considerations

### 7.1. Media Type

IANA is requested to add "application/agent-conversation" as a new media type for Verifiable Agent Conversation Records to the "Media Types" registry [IANA.media-types] in the Standards Tree [RFC6838]:

| Name | Template | Reference |
|---|---|---|
| agent-conversation | application/agent-conversation | RFCthis |

Table 16: Verifiable Agent Conversation Record Media Type

Type name: application

Subtype name: agent-conversation

Required parameters: N/A

Optional parameters: N/A

Encoding considerations: byte string

Security considerations: See Security Considerations {secconsec}

Interoperability considerations: N/A

Published specification: RFCthis

Applications that use this media type: Applications that need to describe AI agent conversation for verification and auditability.

Fragment identifier considerations: N/A

Additional information: Deprecated alias names for this type: N/A

Magic number(s): N/A

File extension(s): acr

Macintosh file type code(s): N/A

Person/email address to contact for further information: TBD

Intended usage: COMMON

Restriction on usage: none

Author: See Author's Addresses section

Change controller: IETF

Provisional registration: no

### 7.2. CoAP Content Format

IANA is requested to assign a Content-Format ID for Verifiable Agent Conversation Records in the "CoAP Content-Formats" registry, within the "Constrained RESTful Environments (CoRE) Parameters" registry group [IANA.core-parameters]:

| Content-Type | Content Coding | ID | Reference |
|---|---|---|---|
| application/agent-conversation | - | TBD1 | RFCthis |

Table 17: Verifiable Agent Conversation Record Content Format

If possible, TBD1 should be assigned in the 256...9999 range.

### 7.3. CBOR Tag

IANA is requested to allocate a tag for Verifiable Agent Conversation Records in the "CBOR Tags" registry [IANA.cbor-tags], preferably with the specific value requested:

| Tag | Data Item | Semantics |
|---|---|---|
| 4149 | binary | Verifiable Agent Conversation Records as defined in RFCthis |

Table 18: Verifiable Agent Conversation Record CBOR Tag

### 7.4. COSE Header Parameter

IANA is requested to allocated the COSE header parameter defined in Table 19 in the "COSE Header Parameters" registry [IANA.cose_header-parameters].

| Name | Label | Value Type | Value Registry | Description | Reference |
|---|---|---|---|---|---|
| trace-metadata | TBD | CBOR map | - | A metadata summary of an Agent Conversation Record | RFCthis, Section 3.11.2 |

Table 19: New COSE Header Parameters

# 8. References

## 8.1. Normative References

[AICPA_TSC_2017]
American Institute of Certified Public Accountants and Chartered Institute of Management Accountants, "Trust Services Criteria for Security, Availability, Processing Integrity, Confidentiality, and Privacy (With Revised Points of Focus - 2022)", AICPA Trust Services Criteria, 2017, <https://www.aicpa-cima.com/resources/download/2017-trust-services-criteria-with-revised-points-of-focus-2022>.

[BCP26] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://doi.org/10.17487/RFC8126>.

[BSI_AI_FINANCE_2024]
Bundesamt fuer Sicherheit in der Informationstechnik, "Test Criteria Catalogue for AI Systems in Finance", BSI AI Finance Test Criteria, 2024, <https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/KI/AI-Finance_Test-Criteria.pdf>.

[ETSI_TS_104223_2025]
European Telecommunications Standards Institute, "Securing Artificial Intelligence (SAI); Baseline Cyber Security Requirements for AI Models and Systems", ETSI TS 104 223, V1.1.1, April 2025, <https://www.etsi.org/deliver/etsi_ts/104200_104299/104223/01.01.01_60/ts_104223v010101p.pdf>.

[EU_AI_ACT_2024]
European Parliament and Council of the European Union, "Regulation (EU) 2024/1689 (Artificial Intelligence Act)", CELEX: 32024R1689, Official Journal of the European Union L 2024/1689, July 2024, <https://eur-lex.europa.eu/eli/reg/2024/1689/oj>.

[EU_CRA_2024]
European Parliament and Council of the European Union, "Regulation (EU) 2024/2847 (Cyber Resilience Act)", CELEX: 32024R2847, Official Journal of the European Union L 2024/2847, November 2024, <https://eur-lex.europa.eu/eli/reg/2024/2847/oj>.

[EU_NIS2_2022]
European Parliament and Council of the European Union, "Directive (EU) 2022/2555 (NIS 2 Directive)", CELEX: 32022L2555, Official Journal of the European Union L 333, December 2022.

[FFIEC_IT_HANDBOOK_2024]
Federal Financial Institutions Examination Council, "Information Technology Examination Handbook", FFIEC IT Examination Handbook, August 2024, <https://ithandbook.ffiec.gov/>.

[IANA.cbor-tags]
IANA, "Concise Binary Object Representation (CBOR) Tags", <https://www.iana.org/assignments/cbor-tags>.

[IANA.core-parameters]
IANA, "Constrained RESTful Environments (CoRE) Parameters", <https://www.iana.org/assignments/core-parameters>.

[IANA.cose_header-parameters]
IANA, "COSE Header Parameters", <https://www.iana.org/assignments/cose>.

[IANA.cwt] IANA, "CBOR Web Token (CWT) Claims", <https://www.iana.org/assignments/cwt>.

[IANA.jwt] IANA, "JSON Web Token (JWT)", <https://www.iana.org/assignments/jwt>.

[IANA.media-types]
IANA, "Media Types", <https://www.iana.org/assignments/media-types>.

[ISO_IEC_42001_2023]
International Organization for Standardization and International Electrotechnical Commission, "Information technology - Artificial intelligence - Management system", ISO/IEC 42001:2023, December 2023, <https://www.iso.org/standard/42001>.

[NIST_AI_100_2_E2025]
National Institute of Standards and Technology, "Adversarial Machine Learning: A Taxonomy and Terminology of Attacks and Mitigations", NIST AI 100-2 E2025, March 2025, <https://csrc.nist.gov/pubs/ai/100/2/e2025/final>.

[NIST_SP_800_53_R5]
National Institute of Standards and Technology, "Security and Privacy Controls for Information Systems and Organizations", NIST Special Publication 800-53, Revision 5, September 2020, <https://csrc.nist.gov/pubs/sp/800/53/r5/upd1/final>.

[OMB_M_21_31]
Office of Management and Budget, "Improving the Federal Government’s Investigative and Remediation Capabilities Related to Cybersecurity Incidents", OMB Memorandum M-21-31, August 2021, <https://www.whitehouse.gov/wp-content/uploads/2021/08/M-21-31-Improving-the-Federal-Governments-Investigative-and-Remediation-Capabilities-Related-to-Cybersecurity-Incidents.pdf>.

[PCI_DSS_V4_0_1]
PCI Security Standards Council, "Payment Card Industry Data Security Standard", PCI DSS Version 4.0.1, June 2024, <https://www.pcisecuritystandards.org/document_library/>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://doi.org/10.17487/RFC2119>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://doi.org/10.17487/RFC4648>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://doi.org/10.17487/RFC5280>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <https://doi.org/10.17487/RFC6838>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://doi.org/10.17487/RFC7252>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://doi.org/10.17487/RFC7515>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://doi.org/10.17487/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://doi.org/10.17487/RFC8174>.

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://doi.org/10.17487/RFC8610>.

[STD90] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://doi.org/10.17487/RFC8259>.

[STD94] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://doi.org/10.17487/RFC8949>.

## 8.2. Informative References

[EMERGENT_MISALIGNMENT_2025]
Betley, J., Tan, D., Warncke, N., Sztyber-Betley, A., Bao, X., Soto, M., Labenz, N., and O. Evans, "Emergent Misalignment: Narrow finetuning can produce broadly misaligned LLMs", PMLR 267:4043-4068, arXiv 2502.17424, February 2025, <https://arxiv.org/abs/2502.17424>.

[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>.

[ISO_IEC_DIS_24970]
International Organization for Standardization and International Electrotechnical Commission, "Artificial intelligence - AI system logging", ISO/IEC DIS 24970, 2025, <https://www.iso.org/standard/88723.html>.

[RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <https://doi.org/10.17487/RFC6973>.

[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://doi.org/10.17487/RFC9334>.

[STD96] Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", STD 96, RFC 9052, DOI 10.17487/RFC9052, August 2022, <https://doi.org/10.17487/RFC9052>.

## Acknowledgments

The authors would like to thank: xor-hardener

## Authors' Addresses

Henk Birkholz
Email: henk.birkholz@ietf.contact

Tobias Heldt

Email: tobias@xor.tech

Orie Steele
Email: orie@or13.io