Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: September 29, 2026

R. Sharif
CyberSecAI Ltd

March 29, 2026

Agent Audit Trail: A Standard Logging Format for Autonomous AI Systems

draft-sharif-agent-audit-trail-00

Abstract

This document specifies a standard logging format for autonomous AI agent systems. The Agent Audit Trail (AAT) defines a JSON-based record structure with mandatory fields for agent identity, action classification, outcome tracking, and trust level reporting. Records are linked via tamper-evident hash chaining using SHA-256 per RFC 8785, with optional ECDSA signatures for non-repudiation.

The format addresses requirements from the EU AI Act (Regulation 2024/1689), which mandates automatic recording of events for high-risk AI systems effective August 2026. It also maps to SOC 2 Trust Services Criteria, ISO/IEC 42001, and PCI DSS v4.0.1 logging requirements.

The design is transport-agnostic and supports export to JSONL, Syslog (RFC 5424), and CSV while preserving chain integrity. Privacy is addressed through input/output hashing and tombstone-based deletion compatible with GDPR Article 17.

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 September 29, 2026.

Copyright Notice

Table of Contents

1. Introduction

1.1. The Problem

The EU Artificial Intelligence Act (Regulation 2024/1689) enters full application on 2 August 2026. Article 12 requires that

high-risk AI systems "shall technically allow for the automatic recording of events ('logs') over the lifetime of the system." Article 12(2) further specifies that logging capabilities shall conform to recognized standards or common specifications.

Despite this regulatory mandate, no standard exists for HOW autonomous AI agents should log their activities. Current approaches suffer from several deficiencies:

- Proprietary formats that vary across vendors, making cross-system auditing impossible.
- No tamper-evidence, allowing post-hoc modification of logs without detection.
- Inconsistent action taxonomies that prevent meaningful comparison of agent behavior across implementations.
- No linkage between agent identity and logged actions, making attribution unreliable.
- No session structure, making it impossible to reconstruct the full sequence of an agent's autonomous decision chain.

This document fills this gap by defining the Agent Audit Trail (AAT), a standard JSON-based logging format with tamper-evident chaining, a defined action taxonomy, and explicit regulatory mapping.

1.2. Design Goals

The AAT format is designed with the following goals:

- Regulatory compliance: Direct mapping to EU AI Act Article 12 requirements and other frameworks.
- Tamper evidence: Hash-chained records that make unauthorized modification detectable.
- Interoperability: A single format usable across different agent frameworks, model providers, and orchestration systems.
- Privacy by design: No raw personal data in records; cryptographic hashes of inputs and outputs instead.
- Transport agnostic: Exportable to JSONL, Syslog, and CSV without losing chain integrity.
- Incremental adoption: Mandatory fields are minimal; optional fields support progressive enhancement.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in capitalized form, as shown here.

Agent: An autonomous software system that uses one or more large language models to make decisions and take actions with limited or no human intervention per action.

Agent Audit Trail (AAT): An ordered sequence of audit records produced by an agent during a session, linked by hash

chaining.

Audit Record: A single JSON object representing one logged event in an agent's operation.

Session: A bounded sequence of agent operations that begins with a genesis record and ends with a session close record.

Genesis Record: The first record in a session, which has no parent and establishes the chain root.

Tombstone Record: A record that replaces a deleted record's content while preserving the hash chain.

Trust Level: A classification from L0 (no verification) to L4 (full mutual authentication with revocation checking) as defined in [MCPS].

Action Type: A controlled vocabulary value describing the category of agent activity being logged.

Chain Hash: The SHA-256 digest of the previous record's canonical JSON representation, linking records into a tamper-evident sequence.

3. Audit Record Format

Each audit record is a JSON object. Fields are divided into mandatory (MUST be present in every record) and optional (MAY be present based on the action type and deployment context).

3.1. Mandatory Fields

record_id: String. REQUIRED. A UUID version 4 [RFC9562] uniquely identifying this record. Implementations MUST generate a fresh UUIDv4 for each record. Duplicate record_id values within a session indicate a processing error and MUST be flagged by validators.

Example: "f47ac10b-58cc-4372-a567-0e02b2c3d479"

timestamp: String. REQUIRED. The time at which the event occurred, formatted per RFC 3339 [RFC3339] with mandatory UTC offset. Implementations SHOULD use UTC (indicated by "Z" suffix). Millisecond precision is RECOMMENDED. Microsecond precision is OPTIONAL.

Example: "2026-03-29T14:30:00.123Z"

agent_id: String. REQUIRED. A URI [RFC3986] uniquely identifying the agent instance. This SHOULD be a persistent identifier that survives agent restarts. When used with MCPS [MCPS], this MUST match the agent_id in the Agent Passport.

Example: "urn:agent:payment-bot.acme.example"

agent_version: String. REQUIRED. The semantic version [SEMVER] of the agent software. This allows correlation of behavior changes with software updates.

Example: "2.1.0"

session_id: String. REQUIRED. A UUID version 4 identifying the current session. All records within a single session MUST share the same session_id.

Example: "a1b2c3d4-e5f6-7890-abcd-ef1234567890"

action_type: String. REQUIRED. One of the registered action type values defined in Section 5. The initial registry contains: "tool_call", "tool_response", "decision", "delegation", "escalation", "error", "lifecycle".

action_detail: Object. REQUIRED. A JSON object containing action-type-specific fields as defined in Section 5. The structure of this object varies by action_type. Unknown fields within action_detail SHOULD be preserved by processors.

outcome: String. REQUIRED. The result of the action. One of the registered outcome values: "success", "failure", "timeout", "denied", "escalated". See Section 12.2 for the outcome registry.

- "success": The action completed as intended.
- "failure": The action did not complete due to an error.
- "timeout": The action exceeded its time budget.
- "denied": The action was blocked by a policy or authorization check.
- "escalated": The action was redirected to a human or higher-authority agent.

trust_level: String. REQUIRED. The trust level at which the agent was operating when this action occurred. One of: "L0", "L1", "L2", "L3", "L4".

- L0: No verification. The agent has no cryptographic identity.
- L1: Self-signed identity. The agent possesses a key pair but no external attestation.
- L2: Authority-signed identity. A Trust Authority has issued the agent’s passport.
- L3: Mutual authentication. Both parties have verified each other’s identity.
- L4: Full mutual authentication with revocation checking and continuous monitoring.

parent_record_id: String or null. REQUIRED. The record_id of the immediately preceding record in the session chain. For genesis records (the first record in a session), this field MUST be null. For all subsequent records, this MUST contain the record_id of the previous record.

prev_hash: String or null. REQUIRED. The SHA-256 hash of the canonical JSON representation (per RFC 8785 [RFC8785]) of the previous record. For genesis records, this field MUST be null. The hash MUST be encoded as a lowercase hexadecimal string (64 characters).

Example: "a7ffc6f8bf1ed76651c14756a061d662..."

3.2. Optional Fields

The following fields are OPTIONAL and MAY be included in any audit record:

human_override: Object. Present when a human intervened in or overrode the agent's action. Contains:

- "operator_id": String. An identifier for the human operator (SHOULD be a pseudonym or role, not a real name, for privacy).
- "reason": String. Free-text explanation of the override.
- "original_action": Object. The action the agent would have taken without intervention.

risk_score: Number. A value between 0.0 and 1.0 indicating the agent's assessed risk of the action. 0.0 indicates minimal risk; 1.0 indicates maximum risk.

model_id: String. The identifier of the language model used for the decision. Example: "gpt-4o-2025-03-01" or "claude-sonnet-4-20250514".

input_hash: String. The SHA-256 hash of the input provided to the agent for this action, encoded as lowercase hexadecimal. Used instead of raw input to preserve privacy.

output_hash: String. The SHA-256 hash of the output produced by the agent for this action, encoded as lowercase hexadecimal.

latency_ms: Number. The wall-clock time in milliseconds from action initiation to completion.

cost_estimate: Object. Estimated cost of the action:

- "amount": Number. The monetary amount.
- "currency": String. ISO 4217 currency code.
- "breakdown": Object. Optional sub-costs (e.g., "compute", "api_calls", "tokens").

sanctions_check: Object. Result of sanctions screening:

- "provider": String. The screening provider.
- "checked_at": String. RFC 3339 timestamp of check.
- "result": String. One of "clear", "match", "error".
- "list_version": String. Version of the sanctions list.

jurisdiction: String. ISO 3166-1 alpha-2 country code indicating the jurisdiction governing this action.

signature: String. An ECDSA P-256 signature over the canonical JSON of this record (excluding the signature field itself), encoded as Base64url per RFC 4648 Section 5. See Section 4.2.

3.3. Field Constraints

The following constraints apply to all audit records:

- All string fields MUST be valid UTF-8.
- The total size of a single audit record SHOULD NOT exceed 64 KB when serialized as JSON. Records exceeding 256 KB MUST be rejected by validators.
- Timestamps MUST NOT be backdated. The timestamp of record N+1 MUST be greater than or equal to the timestamp of record N within the same session.
- The action_detail object MUST contain at least one field relevant to the action_type.
- Implementations MUST NOT add fields with names beginning with "aat_" to action_detail, as this prefix is reserved for future extensions of this specification.

4. Tamper-Evident Chaining

4.1. Hash Computation

The prev_hash field creates a tamper-evident chain across all records in a session. The hash is computed as follows:

1. Take the complete JSON object of the previous record, INCLUDING all fields (mandatory and optional) that were present in the record as stored.
2. Serialize the JSON object using the JSON Canonicalization Scheme (JCS) defined in RFC 8785 [RFC8785]. JCS produces a deterministic byte sequence from any JSON value.
3. Compute the SHA-256 hash of the canonical byte sequence.
4. Encode the resulting 32-byte hash as a 64-character lowercase hexadecimal string.

The formula is:

```
prev_hash(N) = hex(SHA-256(JCS(record(N-1))))
```

For the genesis record (N=0), prev_hash MUST be null.

Implementations MUST use JCS (RFC 8785) for canonicalization. Alternative canonicalization schemes MUST NOT be used, as they would break chain verification across implementations.

4.2. Signature Envelope

When cryptographic non-repudiation is required, records MAY include an ECDSA P-256 signature. The signing procedure is:

1. Construct the complete audit record with all fields EXCEPT the "signature" field.
2. Serialize using JCS (RFC 8785).
3. Compute SHA-256 of the canonical bytes.
4. Sign the hash using ECDSA P-256 with the agent’s private key, per FIPS 186-5 [FIPS186-5].
5. Encode the signature as Base64url (RFC 4648 Section 5)

using IEEE P1363 fixed-length r||s encoding (64 bytes total: 32 bytes r, 32 bytes s).

6. Add the "signature" field to the record.

When verifying, the verifier MUST remove the "signature" field before computing the hash for comparison.

Note: When both prev_hash and signature are present, prev_hash is computed over the COMPLETE previous record INCLUDING its signature field. Only the current record's signature is excluded during signing of the current record.

When used with MCPS [MCPS], the signing key SHOULD be the same key used in the agent's Agent Passport, providing a direct binding between audit records and cryptographic identity.

4.3. Chain Verification

To verify a session's audit trail integrity, a verifier MUST:

1. Confirm the first record has parent_record_id = null and prev_hash = null.

2. For each subsequent record N (where N > 0):

   a. Compute hex(SHA-256(JCS(record(N-1)))).

   b. Compare the computed hash with record(N).prev_hash.

   c. If the values differ, the chain is broken at record N and the trail MUST be flagged as tampered.

3. If signatures are present, verify each signature using the agent's public key.

4. Verify that timestamps are monotonically non-decreasing.

5. Verify that parent_record_id of record N equals the record_id of record N-1.

A chain verification failure MUST be reported as a critical integrity error. Partial chain verification (e.g., verifying only the last K records) is NOT RECOMMENDED but MAY be used for performance reasons if the full chain has been previously verified.

5. Action Type Definitions

Each action type defines a specific structure for the action_detail object.

5.1. tool_call

Logged when the agent invokes an external tool or API.

action_detail fields:

- "tool_name": String. REQUIRED. The name of the tool being called.
- "tool_server": String. OPTIONAL. URI of the MCP server or API endpoint providing the tool.
- "parameters_hash": String. REQUIRED. SHA-256 hash of the serialized parameters sent to the tool.

- "tool_version": String. OPTIONAL. Version of the tool definition.

- "authorization": String. OPTIONAL. The authorization mechanism used (e.g., "bearer_token", "api_key", "mutual_tls").

5.2. tool_response

Logged when the agent receives a response from a tool.

action_detail fields:

- "tool_name": String. REQUIRED. The name of the tool that responded.

- "response_hash": String. REQUIRED. SHA-256 hash of the response payload.

- "response_size": Number. OPTIONAL. Size of the response in bytes.

- "parent_call_id": String. REQUIRED. The record_id of the corresponding tool_call record.

5.3. decision

Logged when the agent makes an autonomous decision.

action_detail fields:

- "decision_type": String. REQUIRED. Category of decision (e.g., "route", "approve", "reject", "classify", "generate").

- "reasoning_hash": String. OPTIONAL. SHA-256 hash of the agent's reasoning chain or chain-of-thought.

- "confidence": Number. OPTIONAL. Confidence score between 0.0 and 1.0.

- "alternatives_considered": Number. OPTIONAL. Count of alternative actions the agent evaluated.

- "policy_ref": String. OPTIONAL. Identifier of the policy or rule that governed this decision.

5.4. delegation

Logged when the agent delegates work to another agent.

action_detail fields:

- "delegate_agent_id": String. REQUIRED. URI of the agent receiving the delegation.

- "delegate_trust_level": String. REQUIRED. Trust level of the delegate agent.

- "task_description_hash": String. REQUIRED. SHA-256 hash of the delegated task description.

- "constraints": Array of String. OPTIONAL. Constraints imposed on the delegate.

- "timeout_ms": Number. OPTIONAL. Maximum time allowed for the delegate to complete the task.

5.5. escalation

Logged when the agent escalates to a human operator or higher-authority system.

action_detail fields:

- "escalation_reason": String. REQUIRED. Why the agent escalated (e.g., "confidence_below_threshold", "policy_requires_human", "risk_score_exceeded", "error_recovery").

- "escalation_target": String. REQUIRED. Identifier of the human or system receiving the escalation.

- "context_hash": String. OPTIONAL. SHA-256 hash of the context provided to the escalation target.

- "urgency": String. OPTIONAL. One of "low", "medium", "high", "critical".

5.6. error

Logged when the agent encounters an error condition.

action_detail fields:

- "error_code": String. REQUIRED. A machine-readable error code.

- "error_message": String. REQUIRED. A human-readable error description.

- "error_category": String. REQUIRED. One of "transport", "authentication", "authorization", "validation", "timeout", "internal", "external".

- "recoverable": Boolean. REQUIRED. Whether the agent can continue operating after this error.

- "stack_hash": String. OPTIONAL. SHA-256 hash of the stack trace, for debugging without exposing internals.

5.7. lifecycle

Logged for agent lifecycle events (start, stop, pause, configuration changes).

action_detail fields:

- "event": String. REQUIRED. One of "session_start", "session_end", "pause", "resume", "configuration_change", "key_rotation", "trust_level_change".

- "previous_state": String. OPTIONAL. The state before this lifecycle event.

- "new_state": String. OPTIONAL. The state after this lifecycle event.

- "trigger": String. OPTIONAL. What caused the lifecycle event (e.g., "scheduled", "manual", "policy",

"error_recovery").

6. Session Structure

6.1. Genesis Record

Every session MUST begin with a genesis record. The genesis record has the following characteristics:

- action_type MUST be "lifecycle".
- action_detail.event MUST be "session_start".
- parent_record_id MUST be null.
- prev_hash MUST be null.
- The action_detail SHOULD include the agent's configuration hash, enabled tools list, and operating parameters to establish a baseline for the session.

Example genesis action_detail:

```
{
  "event": "session_start",
  "new_state": "active",
  "trigger": "scheduled",
  "config_hash": "b5bb9d8014a0f9b1d6...",
  "enabled_tools": [
    "payment_transfer",
    "sanctions_check",
    "balance_query"
  ]
}
```

6.2. Ordered Chain

After the genesis record, all records MUST form a strictly ordered chain:

- Each record's parent_record_id MUST equal the previous record's record_id.
- Each record's prev_hash MUST equal hex(SHA-256(JCS(previous_record))).
- Timestamps MUST be monotonically non-decreasing.
- No gaps in the chain are permitted. If a record cannot be produced (e.g., due to a crash), a recovery record with action_type "error" MUST be inserted to document the gap when the agent resumes.

Branching (multiple records claiming the same parent) is NOT permitted within a single session. If an agent forks into parallel execution paths, each path MUST use a separate session_id and the delegation record in the parent session MUST reference the child session_id.

6.3. Session Close

Every session SHOULD end with a close record. The close record has the following characteristics:

- action_type MUST be "lifecycle".

- action_detail.event MUST be "session_end".

- action_detail MUST include a "session_hash" field containing the SHA-256 hash of the concatenation of all record hashes in the session, in order:

```
session_hash = hex(SHA-256(
  prev_hash(1) || prev_hash(2) || ... || prev_hash(N)
))
```

  where N is the close record itself and prev_hash values are the raw 32-byte digests (not hex-encoded) prior to concatenation.

- action_detail SHOULD include "record_count" (integer) and "duration_ms" (number) summarizing the session.

If an agent terminates abnormally without producing a close record, the session is considered "orphaned." Monitoring systems SHOULD detect orphaned sessions and produce a synthetic close record with outcome "failure" and action_detail.trigger "crash_recovery".

7. Retention Requirements

7.1. High-Risk Systems

For AI systems classified as high-risk under the EU AI Act (Annex III), audit trail records SHOULD be retained for a minimum of 12 months from the session close timestamp.

This aligns with Article 12(1) which states that logging capabilities shall be such that logs are kept for a period appropriate to the intended purpose of the high-risk AI system, of at least six months unless provided otherwise in applicable Union or national law.

The 12-month RECOMMENDATION in this specification exceeds the minimum 6-month requirement to account for audit cycles and incident investigation timelines.

7.2. General-Purpose Systems

For AI systems not classified as high-risk, audit trail records SHOULD be retained for a minimum of 6 months from the session close timestamp.

Deployments subject to financial regulations (e.g., PCI DSS, SOC 2) MAY require longer retention periods as specified by those frameworks.

7.3. Tombstone Records

When individual records must be deleted (e.g., pursuant to GDPR Article 17 right to erasure), the record MUST be replaced with a tombstone record that preserves chain integrity. A tombstone record:

- Retains the original record_id, timestamp, parent_record_id, and prev_hash.

- Sets action_type to "lifecycle".

- Sets action_detail to:

```
{
```

```
      "event": "record_deleted",
      "deletion_reason": "gdpr_art17",
      "deleted_at": "2026-06-15T10:00:00Z",
      "original_action_type": "tool_call"
    }
```

- Sets outcome to "success".
- Retains the signature field if originally present.

The original record's content is destroyed. Because the tombstone preserves the original record_id and prev_hash, subsequent records in the chain remain verifiable. However, the prev_hash of the NEXT record will no longer match the tombstone (since the content changed). To handle this, implementations MUST also store a "tombstone_hash" field in the tombstone record containing the original record's hash, allowing validators to accept the chain break.

8. Export Formats

Implementations MUST support at least one export format. JSONL is the RECOMMENDED primary format.

8.1. JSONL (Primary)

The primary export format is JSON Lines (JSONL), where each line contains exactly one complete audit record serialized as JSON. Lines are separated by a single newline character (U+000A).

- Each line MUST be a valid JSON object.
- The order of lines MUST match the chain order (genesis first, close last).
- The file SHOULD use UTF-8 encoding without a byte order mark (BOM).
- The file extension SHOULD be ".jsonl".

8.2. Syslog (RFC 5424)

For integration with existing logging infrastructure, audit records MAY be exported as Syslog messages per RFC 5424 [RFC5424]. The mapping is:

- FACILITY: local0 (16).
- SEVERITY: based on outcome -- success=6 (Informational), failure=3 (Error), timeout=4 (Warning), denied=5 (Notice), escalated=5 (Notice).
- APP-NAME: the agent_id (truncated to 48 characters).
- MSGID: the action_type.
- STRUCTURED-DATA: SD-ID "aat@IANA-PEN" containing record_id, session_id, trust_level, prev_hash.
- MSG: JSON serialization of the full audit record.

The prev_hash and chain integrity MUST be preserved in the structured data to enable reconstruction of the chain from Syslog archives.

8.3. CSV

For human review and spreadsheet analysis, audit records MAY be exported as CSV per RFC 4180 [RFC4180]. The mapping is:

- Header row: record_id, timestamp, agent_id, agent_version, session_id, action_type, outcome, trust_level, parent_record_id, prev_hash, action_detail.
- The action_detail column contains the JSON serialization of the action_detail object.
- CSV export is inherently lossy for optional fields. Implementations SHOULD document which optional fields are included.

CSV exports MUST NOT be used as the authoritative record. The JSONL format MUST be retained as the source of truth.

9. Regulatory Mapping

9.1. EU AI Act

The following table maps AAT features to EU AI Act articles:

Article 12 (Record-Keeping):
AAT provides automatic recording via the mandatory audit record format (Section 3). Hash chaining (Section 4) ensures records "allow the tracing back of the AI system's operation." The session structure (Section 6) provides the "period of each use" required by Art 12(1)(c).

Article 13 (Transparency):
The action_type taxonomy (Section 5) and decision records (Section 5.3) provide interpretability of agent behavior. The model_id field documents which model was used. The human_override field documents human interventions.

Article 14 (Human Oversight):
The escalation action type (Section 5.5) documents when and why agents escalated to humans. The human_override optional field (Section 3.2) captures human interventions. Trust levels document the degree of autonomous operation.

Article 72 (Reporting):
The export formats (Section 8) enable provision of logs to national competent authorities. The session_hash in session close records (Section 6.3) provides a verifiable summary for regulatory reporting.

9.2. SOC 2

SOC 2 Trust Services Criteria relevant to AAT:

- CC6.1 (Logical Access): trust_level and authorization fields document access controls.
- CC7.2 (System Monitoring): Continuous audit trail with tamper-evident chaining satisfies monitoring requirements.
- CC8.1 (Change Management): lifecycle action type records document configuration changes.

9.3. ISO/IEC 42001

ISO/IEC 42001 (AI Management System) clauses addressed:

- Clause 6.1.2 (AI Risk Assessment): risk_score and decision records support risk documentation.
- Clause 8.4 (AI System Operation): Full session audit trails document operational behavior.
- Clause 9.1 (Monitoring): Continuous logging with chain verification supports monitoring requirements.

9.4. PCI DSS v4.0.1

PCI DSS v4.0.1 requirements addressed by AAT:

- Requirement 10.2: AAT provides audit logs for all agent actions including tool calls, decisions, and errors.
- Requirement 10.3: Record fields (timestamp, agent_id, action_type, outcome) map directly to required audit trail entries.
- Requirement 10.5: Hash chaining and optional signatures protect audit trail integrity.
- Requirement 10.7: Retention requirements (Section 7) align with PCI DSS retention periods.

10. Privacy Considerations

10.1. Data Minimization

AAT is designed with privacy by default:

- Raw input and output data MUST NOT be stored in audit records. Implementations MUST use the input_hash and output_hash fields instead.
- The human_override.operator_id SHOULD be a pseudonymous identifier or role name, not a natural person's name.
- The reasoning_hash field in decision records stores a hash of the reasoning chain, not the reasoning itself.
- Tool parameters are recorded via parameters_hash, not in cleartext.
- Sanctions check results record only "clear", "match", or "error" -- not the details of what was screened.

Implementations that need to retain raw data for debugging MUST store it in a separate system with appropriate access controls, linked to the audit trail via record_id.

10.2. Right to Erasure

To support GDPR Article 17 (right to erasure) and similar regulations, AAT uses tombstone records (Section 7.3) rather than record deletion. This approach:

- Removes all personal data from the record.
- Preserves chain integrity for regulatory compliance.
- Documents the fact and reason for deletion.
- Is compatible with the EU AI Act's record-keeping

requirements, which do not require retention of personal data but do require retention of operational logs.

Data controllers MUST implement a process to identify which audit records contain personal data (even in hashed form) and respond to erasure requests by creating tombstone records within 30 days.

11. Security Considerations

11.1. Log Tampering

The primary threat to audit trails is unauthorized modification. AAT mitigates this through:

- Hash chaining: Any modification to a record invalidates all subsequent prev_hash values, making tampering detectable.
- Optional signatures: ECDSA P-256 signatures provide non-repudiation and prevent even the log storage system from undetectably modifying records.
- Session hashes: The session_hash in close records provides a single value that can be stored externally (e.g., on a blockchain or with a timestamp authority) to anchor the entire session.

Implementations SHOULD store session_hash values in a separate, append-only system to provide an independent verification point.

11.2. Log Injection

Attackers may attempt to inject false audit records into the trail. Mitigations include:

- Signature verification: When signatures are present, only records signed by the agent’s key are valid.
- Chain continuity: Injected records would break the hash chain unless the attacker can also modify all subsequent records.
- Timestamp monotonicity: Injected records with out-of-order timestamps are detectable.
- Record size limits: The 256 KB maximum prevents denial-of-service through oversized records.

Implementations MUST validate all records against the schema before accepting them into the audit trail.

11.3. Timing Attacks

Audit record timestamps may be manipulated if the agent controls its own clock. Mitigations include:

- Using NTP-synchronized clocks with drift monitoring.
- Cross-referencing timestamps with external systems (e.g., tool server response timestamps).
- Flagging sessions where timestamps show suspicious patterns (e.g., large jumps, regression).

Implementations SHOULD monitor for timestamp anomalies and flag them for human review.

11.4. Chain Breaks

Chain breaks can occur due to:

- System crashes during record writing.
- Storage corruption.
- Intentional tampering.

When a chain break is detected, implementations MUST:

1. Flag the break with a severity of "critical".
2. Record the break location (which record pair failed verification).
3. Preserve both the broken chain and any recovered data.
4. If the break is due to crash recovery, insert an error record documenting the gap.

Chain breaks in high-risk systems MUST trigger an alert to the system operator within 1 hour.

12. IANA Considerations

12.1. Action Type Registry

This document requests IANA to create the "Agent Audit Trail Action Types" registry. The registration policy is "Specification Required" per RFC 8126 [RFC8126].

Initial registry contents:

Value	Description	Reference
tool_call	Agent invokes a tool	Sec 5.1
tool_response	Agent receives tool reply	Sec 5.2
decision	Agent makes a decision	Sec 5.3
delegation	Agent delegates to agent	Sec 5.4
escalation	Agent escalates to human	Sec 5.5
error	Agent encounters error	Sec 5.6
lifecycle	Agent lifecycle event	Sec 5.7

New entries MUST include a value (lowercase ASCII string, max 32 characters), description, and reference to a published specification.

12.2. Outcome Registry

This document requests IANA to create the "Agent Audit Trail Outcomes" registry. The registration policy is "Specification Required" per RFC 8126.

Initial registry contents:

Value	Description	Reference
success	Action completed as intended	Sec 3.1

failure	Action failed due to error	Sec 3.1
timeout	Action exceeded time budget	Sec 3.1
denied	Action blocked by policy	Sec 3.1
escalated	Action redirected to human	Sec 3.1

13. References

13.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <https://www.rfc-editor.org/info/rfc3339>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8785] Rundgren, A., Jordan, B., and S. Erdtman, "JSON Canonicalization Scheme (JCS)", RFC 8785, DOI 10.17487/RFC8785, June 2020, <https://www.rfc-editor.org/info/rfc8785>.

[RFC9562] Davis, K., Peabody, B., and P. Leach, "Universally Unique IDentifiers (UUIDs)", RFC 9562, DOI 10.17487/RFC9562, May 2024, <https://www.rfc-editor.org/info/rfc9562>.

[FIPS186-5] National Institute of Standards and Technology, "Digital Signature Standard (DSS)", FIPS PUB 186-5, DOI 10.6028/NIST.FIPS.186-5, February 2023.

[RFC8126] 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://www.rfc-editor.org/info/rfc8126>.

13.2. Informative References

[MCPS] Sharif, R., "MCPS: Cryptographic Security Layer for the Model Context Protocol", draft-sharif-mcps-secure-mcp-02, March 2026.

[EU-AI-ACT] European Parliament and Council, "Regulation (EU) 2024/1689 laying down harmonised rules on artificial intelligence (Artificial Intelligence Act)", Official Journal of the European Union, L series, 2024/1689, August 2024.

[RFC5424] Gerhards, R., "The Syslog Protocol", RFC 5424, DOI 10.17487/RFC5424, March 2009,

            <https://www.rfc-editor.org/info/rfc5424>.

[RFC4180] Shafranovich, Y., "Common Format and MIME Type for
            Comma-Separated Values (CSV) Files", RFC 4180,
            DOI 10.17487/RFC4180, October 2005,
            <https://www.rfc-editor.org/info/rfc4180>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data
            Encodings", RFC 4648, DOI 10.17487/RFC4648,
            October 2006,
            <https://www.rfc-editor.org/info/rfc4648>.

[ISO42001] International Organization for Standardization,
            "Information technology -- Artificial intelligence --
            Management system", ISO/IEC 42001:2023, December
            2023.

[PCI-DSS] PCI Security Standards Council, "Payment Card
            Industry Data Security Standard Version 4.0.1",
            June 2024.

[SOC2]   American Institute of Certified Public Accountants,
            "SOC 2 -- SOC for Service Organizations: Trust
            Services Criteria", 2017.

[SEMVER]   Preston-Werner, T., "Semantic Versioning 2.0.0",
            <https://semver.org/>.

Appendix A. Example Audit Trail

The following example shows a complete audit trail for a payment agent session that processes a GBP 500 transfer. The session demonstrates tool calls, decisions, sanctions screening, and successful completion. prev_hash values are truncated for readability (shown as first 16 hex characters).

Record 1: Genesis (session start)


```
{
  "record_id": "a1000000-0000-4000-8000-000000000001",
  "timestamp": "2026-03-29T14:00:00.000Z",
  "agent_id": "urn:agent:payment-bot.acme.example",
  "agent_version": "2.1.0",
  "session_id": "sess-29mar-0001-4000-8000-abcdef123456",
  "action_type": "lifecycle",
  "action_detail": {
    "event": "session_start",
    "new_state": "active",
    "trigger": "api_request",
    "config_hash": "b5bb9d8014a0f9b1...",
    "enabled_tools": [
      "payment_transfer",
      "sanctions_check",
      "balance_query"
    ]
  },
  "outcome": "success",
  "trust_level": "L2",
  "parent_record_id": null,
  "prev_hash": null
}
```


Record 2: Sanctions screening tool call


```
{
  "record_id": "a1000000-0000-4000-8000-000000000002",
```

```
  "timestamp": "2026-03-29T14:00:00.150Z",
  "agent_id": "urn:agent:payment-bot.acme.example",
  "agent_version": "2.1.0",
  "session_id": "sess-29mar-0001-4000-8000-abcdef123456",
  "action_type": "tool_call",
  "action_detail": {
    "tool_name": "sanctions_check",
    "tool_server": "https://screening.acme.example/v2",
    "parameters_hash": "e3b0c44298fc1c14...",
    "authorization": "mutual_tls"
  },
  "outcome": "success",
  "trust_level": "L2",
  "parent_record_id":
    "a1000000-0000-4000-8000-000000000001",
  "prev_hash": "7d865e959b2466918a...",
  "input_hash": "9f86d081884c7d659a...",
  "latency_ms": 145
}
```

Record 3: Sanctions screening response

```
{
  "record_id": "a1000000-0000-4000-8000-000000000003",
  "timestamp": "2026-03-29T14:00:00.295Z",
  "agent_id": "urn:agent:payment-bot.acme.example",
  "agent_version": "2.1.0",
  "session_id": "sess-29mar-0001-4000-8000-abcdef123456",
  "action_type": "tool_response",
  "action_detail": {
    "tool_name": "sanctions_check",
    "response_hash": "2cf24dba5fb0a301...",
    "response_size": 256,
    "parent_call_id":
      "a1000000-0000-4000-8000-000000000002"
  },
  "outcome": "success",
  "trust_level": "L2",
  "parent_record_id":
    "a1000000-0000-4000-8000-000000000002",
  "prev_hash": "4e07408562bedb8b6...",
  "sanctions_check": {
    "provider": "acme_screening",
    "checked_at": "2026-03-29T14:00:00.290Z",
    "result": "clear",
    "list_version": "2026-03-29"
  }
}
```

Record 4: Decision to approve transfer

```
{
  "record_id": "a1000000-0000-4000-8000-000000000004",
  "timestamp": "2026-03-29T14:00:00.310Z",
  "agent_id": "urn:agent:payment-bot.acme.example",
  "agent_version": "2.1.0",
  "session_id": "sess-29mar-0001-4000-8000-abcdef123456",
  "action_type": "decision",
  "action_detail": {
    "decision_type": "approve",
    "reasoning_hash": "6b86b273ff34fce1...",
    "confidence": 0.97,
    "alternatives_considered": 2,
    "policy_ref": "payment-policy-v3.2"
  },
  "outcome": "success",
```

```
    "trust_level": "L2",
    "parent_record_id":
      "a1000000-0000-4000-8000-000000000003",
    "prev_hash": "ef2d127de37b942ba...",
    "risk_score": 0.12,
    "model_id": "claude-sonnet-4-20250514",
    "cost_estimate": {
      "amount": 500.00,
      "currency": "GBP"
    }
  }
```

Record 5: Payment execution tool call

```
  {
    "record_id": "a1000000-0000-4000-8000-000000000005",
    "timestamp": "2026-03-29T14:00:00.320Z",
    "agent_id": "urn:agent:payment-bot.acme.example",
    "agent_version": "2.1.0",
    "session_id": "sess-29mar-0001-4000-8000-abcdef123456",
    "action_type": "tool_call",
    "action_detail": {
      "tool_name": "payment_transfer",
      "tool_server": "https://payments.acme.example/v1",
      "parameters_hash": "d4735e3a265e16ee...",
      "authorization": "bearer_token"
    },
    "outcome": "success",
    "trust_level": "L2",
    "parent_record_id":
      "a1000000-0000-4000-8000-000000000004",
    "prev_hash": "e7f6c011776e8db7c...",
    "latency_ms": 890,
    "jurisdiction": "GB"
  }
```

Record 6: Session close

```
  {
    "record_id": "a1000000-0000-4000-8000-000000000006",
    "timestamp": "2026-03-29T14:00:01.210Z",
    "agent_id": "urn:agent:payment-bot.acme.example",
    "agent_version": "2.1.0",
    "session_id": "sess-29mar-0001-4000-8000-abcdef123456",
    "action_type": "lifecycle",
    "action_detail": {
      "event": "session_end",
      "previous_state": "active",
      "new_state": "closed",
      "trigger": "task_complete",
      "session_hash": "9c22ff5f21f0b81b...",
      "record_count": 6,
      "duration_ms": 1210
    },
    "outcome": "success",
    "trust_level": "L2",
    "parent_record_id":
      "a1000000-0000-4000-8000-000000000005",
    "prev_hash": "a3a2e67ad1b8d57e2..."
  }
```

Appendix B. EU AI Act Compliance Checklist

The following table maps EU AI Act Article 12 sub-requirements to specific AAT features:

Art 12 Requirement	AAT Feature
12(1) Automatic recording	Mandatory audit record format (Section 3)
12(1)(a) Recording of period of each use	session_id + ordered chain (Section 6)
12(1)(b) Reference database against which input data has been checked	agent_id (URI) + agent_version + model_id (Section 3)
12(1)(c) Input data for which search has led to a match	input_hash field (Section 3.2)
12(1)(d) Identification of natural persons involved in verification	human_override field with pseudonymous operator_id (Section 3.2)
12(2) Conform to recognised standards	This specification
12(3) Appropriate to intended purpose, at least 6 months	Retention requirements (Section 7): 12 months for high-risk, 6 months general
12(4) Providers of high-risk AI systems that are credit institutions	Export formats (Section 8) enable log provision to financial authorities
Art 13 Transparency	action_type taxonomy + decision records (Sec 5.3)
Art 14 Human oversight	escalation type (Sec 5.5) + human_override (Sec 3.2)

Appendix C. Implementation Notes

C.1. Performance Considerations

Hash computation adds overhead to each record. Benchmarks on commodity hardware show:

- JCS canonicalization: ~0.1 ms per record (typical size).
- SHA-256 hash: ~0.01 ms per record.
- ECDSA P-256 signing: ~1-2 ms per record.
- Total overhead with signing: ~2 ms per record.

For high-throughput agents (>1000 actions/second), implementations MAY batch records and compute hashes asynchronously, provided the chain order is preserved. The timestamp MUST reflect the actual event time, not the time the hash was computed.

C.2. Storage Estimates

A typical audit record (mandatory fields only) is approximately 500-800 bytes when serialized as JSON. With

optional fields, records range from 800-2000 bytes.

For a high-activity agent producing 10,000 records per day:

- Daily storage: ~10-20 MB (JSONL).
- Monthly storage: ~300-600 MB.
- 12-month retention: ~3.6-7.2 GB.

Implementations SHOULD apply compression (e.g., gzip) to archived sessions. Typical compression ratios for JSON audit data are 5:1 to 10:1.

C.3. Clock Synchronization

Accurate timestamps are critical for audit trail integrity. Implementations MUST:

- Use NTP or PTP for clock synchronization.
- Monitor clock drift and alert if drift exceeds 100 ms.
- Record the clock source in the genesis record's action_detail when available.

In distributed agent systems where multiple agents contribute to a workflow, each agent maintains its own audit trail with its own clock. Cross-agent timestamp correlation SHOULD use the delegation record timestamps as synchronization points.

C.4. Relationship to MCPS

AAT is designed to complement MCPS [draft-sharif-mcps-secure-mcp]. The relationship is:

- MCPS provides cryptographic identity (Agent Passports) and per-message signing for MCP protocol traffic.
- AAT provides the audit log format for recording what agents did and why.
- The agent_id in AAT records SHOULD match the agent_id in the MCPS Agent Passport.
- AAT signatures SHOULD use the same ECDSA P-256 key as the MCPS Agent Passport, providing a single cryptographic identity across both protocol security and audit logging.
- MCPS trust levels (L0-L4) are directly referenced in AAT records via the trust_level field.

Implementations that deploy both MCPS and AAT achieve both real-time protocol security and comprehensive audit logging under a unified cryptographic identity.

C.5. Validator Implementation

A conformant AAT validator MUST check:

1. Schema validation: All mandatory fields present with correct types.
2. Chain integrity: prev_hash values match computed hashes.
3. Temporal ordering: Timestamps are monotonically

non-decreasing.

4. Session structure: Genesis record is first, close record is last (if present).

5. Referential integrity: parent_record_id values reference existing records.

6. Action type conformance: action_detail contains required fields for the declared action_type.

A validator SHOULD produce a structured report indicating pass/fail for each check, with the specific record_id where failures occurred.

Author’s Address

Raza Sharif
CyberSecAI Ltd

Email: contact@agentsign.dev