Independent Submission
Internet-Draft
Intended status: Informational
Expires: 21 November 2026

C. Hood
Nomotic, Inc.

20 May 2026

AGTP Transparency Log Protocol
draft-hood-agtp-log-01

Abstract

This document specifies the AGTP Transparency Log (AGTP-LOG): the append-only audit log protocol that underpins the log-anchored Trust Tier 1 verification path defined in [AGTP] and provides the post-hoc audit infrastructure for AGTP-based agent operations. AGTP-LOG aligns with [RFC9162] (Certificate Transparency 2.0) as the verifiable data structure and issues COSE_Sign1 receipts per [RFC9943] (SCITT) for cross-ecosystem interoperability. This document also establishes the AGTP Agent Identity Taxonomy (Agent Genesis, Canonical Agent-ID, Agent Certificate) that [AGTP] v07 normatively adopts. The log protocol defined here covers entry submission, receipt retrieval, inclusion-proof retrieval, consistency-proof retrieval, and discovery via the log_uri field of the Agent Genesis. Per-platform logs are the default; the federation model follows the SCITT cross-witnessing pattern. Witness and monitor role definitions, full federation procedures, and the formal entry-acceptance threat model are deferred to a future revision.

The audit properties AGTP-LOG provides are structural rather than retrofitted. Statements are signed by the governance platform that operates the log, not by the agent the statements concern; an agent cannot forge its own audit trail because it does not control the log operator’s signing key. This architectural separation distinguishes AGTP-LOG audit infrastructure from approaches that layer audit data models onto general-purpose substrates where the audited party participates in writing the audit trail.

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/.

Copyright Notice

Table of Contents

1. Introduction

AGTP v07 recognizes log-anchored as one of three equivalent Trust Tier 1 verification paths defined in [AGTP-TRUST]. This document specifies the log protocol that path depends on. Earlier draft material that referred to an inline AGTP Certificate Transparency Log (AGTP-CTL) in [AGTP-CERT] is superseded by this document. The next revision of [AGTP-CERT] will reference AGTP-LOG normatively rather than define its own log.

AGTP-LOG inherits the verifiable data structure of [RFC9162] and the receipt format of [RFC9943]. This document does not redefine the cryptographic primitives or the underlying append-only log construction; it specifies how an AGTP governance platform operates a log over those primitives, what kinds of statements the log accepts, how verifiers discover the log, and how AGTP-LOG receipts are bound to the Agent Identity Document.

The key requirements language follows [RFC2119] and [RFC8174].

2. AGTP-LOG as Audit Infrastructure

AGTP-LOG is the audit infrastructure for AGTP-based agent operations. Entries committed to the log constitute the post-hoc record of agent identity events: creation, lifecycle transitions, and revocation. These records carry the structural properties that distinguish protocol-layer audit from application-layer audit logging:

Audited party cannot forge own trail. Statements are signed by the governance platform that issued the Agent Genesis, not by the agent. An agent does not control the log operator's signing key and therefore cannot forge, alter, or omit its own audit records. This is a structural property of the architecture, not a policy constraint that requires enforcement.

Audit records are append-only and tamper-evident. The verifiable data structure of [RFC9162] guarantees that committed statements cannot be retroactively altered without breaking the cryptographic consistency of the log. A log operator that publishes inconsistent tree heads is detectable by monitors observing the log.

Audit records carry cryptographic provenance. Receipts per [RFC9943] are independently verifiable: a party holding a Canonical Agent-ID and the corresponding receipt can confirm the audit record exists in the log without trusting any single party's assertion that it does.

Audit records are bound to wire-level identity. The agtp-subject field of every statement is a Canonical Agent-ID that ties the audit record to the protocol-level identity used in every AGTP request. The audit trail and the operational identity are the same identity, not separately maintained representations that must be correlated after the fact.

2.1. Scope: Post-Hoc Audit

AGTP-LOG addresses post-hoc audit: the recording of what has happened, after it has happened, in a form that can be verified later. This is the audit problem regulatory frameworks and compliance systems typically address.

Pre-commitment intent declaration — recording what an agent declares it is about to do, before action evaluation, with ordering guarantees that prevent ex-post fabrication of intent — is a distinct concern with different threat-model properties. AGTP-LOG does not address pre-commitment intent declaration. The AGTP framework treats pre-commitment intent declaration as adjacent work; see Section 2.3.

2.2. Regulatory Context

Audit infrastructure for agent operations is increasingly subject to regulatory requirements. The EU AI Act Article 12 requires automatic recording of events for high-risk AI systems with cryptographically verifiable provenance. Similar frameworks are emerging in other jurisdictions, and industry-specific regulations (financial services, healthcare, aviation, government) add their own requirements for agent attribution and audit trails.

AGTP-LOG provides the protocol-layer infrastructure these regulatory frameworks require: tamper-evident records of agent identity events, signed by accountable governance platforms, verifiable by independent third parties, bound to the wire-level identity used in agent operations. AGTP-LOG does not itself implement any specific

regulatory framework; it provides the audit primitives that compliance systems built on AGTP can use to satisfy regulatory requirements.

2.3. Related Audit Work

Multiple parallel efforts in the agent infrastructure space address adjacent audit concerns. AGTP-LOG positions explicitly relative to:

Pre-commitment intent declaration: [IDP] defines intent declaration committed to a tamper-evident log before policy evaluation executes. The ordering guarantee (declaration < evaluation < execution) is structurally different from AGTP-LOG's post-hoc recording. AGTP-LOG and pre-commitment intent declaration are complementary: both can operate against the same governance platform, recording different categories of agent events.

HTTP-based audit data models: Work in progress in the IETF (e.g., [AUDIT-ARCH]) proposes audit data models layered over HTTP and existing token formats. AGTP-LOG's architectural commitment is different: audit is a structural property of the protocol substrate, not a data model retrofitted onto a substrate that was not designed for accountability. Both approaches may coexist for the agent traffic each substrate carries.

Transparency standards (SCITT, RFC 9162): AGTP-LOG reuses these verifiable data structures and receipt formats. AGTP-LOG is a profile and deployment pattern for SCITT-aligned audit infrastructure specific to AGTP agent identity events.

3. Terminology

3.1. AGTP Agent Identity Taxonomy

AGTP identity is composed of three distinct elements. This taxonomy was formalized concurrent with the v07 revision of [AGTP] and is normatively adopted throughout the AGTP draft family.

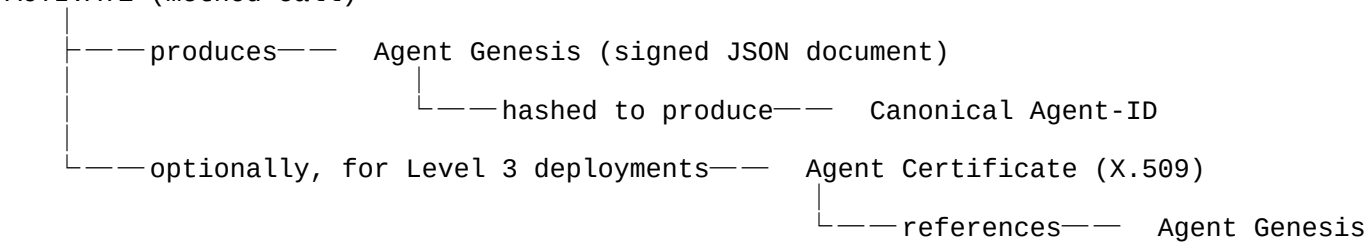
Agent Genesis: The permanent, signed governance-layer document produced at ACTIVATE time that establishes an agent's identity. The Agent Genesis is the origin record from which all other identity artifacts derive. It is issued by the governance platform, signed with the governance platform's key, and archived on revocation. An Agent Genesis is never reissued; the identity it establishes is permanent for the life of the agent.

Canonical Agent-ID: A 256-bit cryptographic hash of the Agent

Genesis, used as the agent's identifier in all AGTP protocol operations. The Canonical Agent-ID is the value carried in the Agent-ID header on every request, the key in the registry, and the subject of transparency log entries. The Canonical Agent-ID is derived from the Agent Genesis; it is not an independent value.

Agent Certificate: An optional X.509 v3 credential defined in [AGTP-CERT] that binds the Canonical Agent-ID to TLS mutual authentication for transport- layer verification and O(1) scope enforcement at Scope-Enforcement Points. Agent Certificates are renewable and revocable, with a recommended 90-day validity period. The Agent Certificate is a derived credential, not an identity substrate; it references the Agent Genesis via the activation-certificate-id extension.

The relationship among these three elements:



3.2. Log-Specific Terminology

Log: An append-only, tamper-evident record of AGTP-LOG statements operated by a governance platform. The log implements the verifiable data structure of [RFC9162] and issues SCITT receipts per [RFC9943].

Log Operator: The party that operates a log. In this revision the log operator is normatively the governance platform that issued the Agent Genesis statements committed to the log. Future revisions *MAY* permit log operation to be delegated to a separate entity.

Statement: A single AGTP-LOG entry: a signed assertion by a log operator about an agent identity event. Each statement has an event type, subject, payload, issuer, and signature. See Section 6.

Receipt: A COSE_Sign1 envelope per [RFC9943] issued by a log against

a statement, attesting that the statement has been committed to the log. The receipt is the consumable proof for verifiers. See Section 7.

Inclusion Proof: A cryptographic proof per [RFC9162] demonstrating that a given statement is present at a specific position in the log's tree structure. Embedded within a receipt or retrievable independently via the log protocol.

Consistency Proof: A cryptographic proof per [RFC9162] demonstrating that two signed tree heads at different sizes are consistent (the smaller tree is a prefix of the larger). Used by monitors to detect split-view attacks.

Signed Tree Head (STH): The log operator's signed assertion of the current tree size and root hash per [RFC9162] Section 4.10.

Witness: A third party that countersigns the log's signed tree heads to attest that it has observed the log in a consistent state. The witness role is defined informally here and normatively specified in a future revision; see Section 13.

Monitor: A party that polls the log's STHs and consistency proofs to detect split-view attacks or operator misbehavior. The monitor role is defined informally here and normatively specified in a future revision; see Section 13.

Verifier: A consumer of an AGTP-LOG receipt that confirms a given Agent Genesis is committed to a log. Any party holding a Canonical Agent-ID and a log_uri can act as a verifier.

4. Log Scope

A log *MUST* accept statements for at least the following five AGTP-LOG event types. The event type is recorded in the statement envelope (see Section 6).

Event Type	Triggered By	Subject
agent-genesis-issued	ACTIVATE method completion	Canonical Agent-ID derived from the new Agent Genesis
agent-genesis-revoked	REVOKE method completion	Canonical Agent-ID being retired
agent-lifecycle-suspended	SUSPEND method completion	Canonical Agent-ID entering Suspended state
agent-lifecycle-reinstated	REINSTATE method completion	Canonical Agent-ID returning to Active state
agent-lifecycle-deprecated	DEPRECATE method completion	Canonical Agent-ID entering Deprecated state

Table 1: AGTP-LOG Event Types

The above five event types form the initial AGTP-LOG event-type registry (see Section 12). Future event types *MAY* be registered as new operational categories arise without requiring revision of this document.

Implementations *MAY* log additional event types beyond the above five. Agent Certificate issuance and revocation events, defined in [AGTP-CERT], are explicit candidates: a deployment that operates both AGTP-LOG and an Agent Certificate authority *MAY* log certificate-lifecycle events to the same log. Whether a deployment logs certificate events is a deployment decision; logging them is *NOT* required by this document. When logged, the event types *MUST* be registered in the AGTP-LOG event-type registry; ad-hoc event-type strings *MUST NOT* be admitted.

Statements *MUST NOT* carry secret material. The Agent Genesis is public by design; the Canonical Agent-ID is derived from a public hash; lifecycle states are operational facts. A log operator that finds itself wanting to commit a secret to the log is misusing the protocol.

5. Log Architecture

5.1. Per-Platform Logs (Default)

Each AGTP governance platform *MUST* operate a log if it issues Agent Genesis statements with verification_path: log-anchored. The governance platform is the log operator. Statements committed to the log are signed by the governance platform's key as recorded in the Agent Genesis issuer field.

A governance platform operating a log advertises the log's URI in the log_uri field of every Agent Genesis it produces under the log-anchored path. See Section 9.

Per-platform logs follow the SCITT issuer-local convention: each issuer commits its own statements to its own log, signed with its own key, with receipts referencing that issuer's identity. This matches typical deployment patterns and minimizes operational coupling between governance platforms.

5.2. Federation via Cross-Witnessing

When a deployment requires that statements from one governance platform be verifiable by participants who do not implicitly trust that platform's log operator, the platform *MAY* invite a witness to countersign its signed tree heads. The witness adds a second signature over the STH, attesting that the witness has observed the log in a consistent state at that tree size.

The cross-witnessing model preserves per-platform log operation while introducing third-party attestation. A verifier presented with an Agent Genesis from platform A and the corresponding receipt *MAY* require the STH to be countersigned by a witness whose key the verifier trusts before accepting the inclusion proof.

The witness protocol, witness selection, witness key publication, and the schema of countersigned STHs are deferred to a future revision per Section 13.

6. Entry Format

Every statement is a SCITT-aligned signed envelope per [RFC9943]. The envelope is a COSE_Sign1 structure per [RFC9052] containing the following protected and unprotected header parameters and payload.

The protected header *MUST* carry:

Parameter	CBOR Label	Value
alg	1	Signing algorithm. *MUST* be ES256 (-7), ES384 (-35), ES512 (-36), or EdDSA (-8).
kid	4	Key identifier of the log operator’s signing key (governance platform key).
cty	3	Content type. *MUST* be application/agtp-log-statement+cbor.
agtp-event-type	TBD	One of the event-type strings registered in the AGTP-LOG event-type registry.
agtp-subject	TBD	Canonical Agent-ID, encoded as a 32-byte CBOR byte string.
agtp-issuer	TBD	URI of the governance platform issuing this statement.
agtp-issued-at	TBD	Statement issuance time as an RFC 3339 timestamp.

Table 2: AGTP-LOG Statement Protected Header

The unprotected header *MAY* carry implementation-specific parameters; verifiers *MUST* ignore unprotected parameters not recognized.

The payload *MUST* be a CBOR-encoded map whose contents depend on the event type:

For agent-genesis-issued:

```
  {
    "agent-genesis": <bytes>,             ; the canonical CBOR-encoded
                                           ; Agent Genesis document
    "log-position": <uint>,                ; assigned by log operator
                                           ; on entry commit
    "previous-tree-size": <uint>           ; tree size before this entry
  }
```

For agent-genesis-revoked, agent-lifecycle-suspended, agent-lifecycle-reinstated, and agent-lifecycle-deprecated:

```
{
  "lifecycle-event": <text>,              ; matches event-type
  "reason": <text>,                       ; operator-supplied reason
  "previous-state": <text>,               ; lifecycle state prior to event
  "new-state": <text>,                    ; lifecycle state after event
  "log-position": <uint>,
  "previous-tree-size": <uint>
}
```

The CBOR encoding *MUST* be deterministic per [RFC8949] Section 4.2 (Canonical CBOR). The signature is computed over the COSE_Sign1 Sig_structure per [RFC9052] Section 4.4.

A statement is accepted into the log only after the log operator has verified:

1. The signature is valid against the log operator’s published key.

2. The agtp-issuer URI matches the log operator’s identity.

3. The agtp-subject is a well-formed 32-byte Canonical Agent-ID.

4. The agtp-event-type is registered in the AGTP-LOG event-type registry.

5. The payload structure conforms to the schema for the declared event type.

6. For agent-genesis-issued, the hash of the embedded Agent Genesis matches the agtp-subject byte for byte.

Statements failing any of these checks *MUST NOT* be appended to the log. The log operator *MUST* record the rejection in its operational logs but *MUST NOT* commit the rejected statement to the verifiable tree.

7. Receipt Format

A receipt is a COSE_Sign1 envelope per [RFC9943] attesting that a statement has been committed to the log. Receipts are the consumable proof for AGTP verifiers.

The receipt's content type is application/scitt-receipt+cose per [RFC9943]. The receipt's payload references the statement's position in the log and embeds an inclusion proof against a signed tree head.

The protected header *MUST* carry:

Parameter	CBOR Label	Value
alg	1	Signing algorithm; same constraint as statement signing.
kid	4	Key identifier of the log operator's signing key.
cty	3	Content type. *MUST* be application/scitt-receipt+cose.
verifiable-data-structure	TBD	*MUST* be RFC9162_SHA256 per [RFC9162].
agtp-statement-position	TBD	The log position (zero-indexed) of the committed statement.
agtp-statement-hash	TBD	SHA-256 hash of the canonical CBOR-encoded statement.
agtp-signed-tree-head	TBD	The STH the inclusion proof attests against.

Table 3: AGTP-LOG Receipt Protected Header

The payload *MUST* be a CBOR-encoded map carrying the inclusion proof:

```
{
  "tree-size": <uint>,                    ; STH tree size
  "leaf-index": <uint>,                   ; position of statement
  "audit-path": [<bytes>, <bytes>, ...]  ; Merkle audit path per RFC 9162
}
```

Receipts *MUST* be retrievable by any party that holds the Canonical Agent-ID and the log_uri. The receipt *MUST NOT* contain the statement itself; verifiers requiring the statement *MUST* retrieve it separately. This separation keeps receipts small enough to embed in the Agent Genesis (log_inclusion_proof field) without bloating the Agent Identity Document.

The receipt for an agent-genesis-issued event *MUST* be embedded in the Agent Genesis's log_inclusion_proof field at the moment the Agent Genesis is finalized. The governance platform commits the statement to the log, retrieves the resulting receipt, embeds the receipt in the Agent Genesis, and only then signs the Agent Genesis and computes the Canonical Agent-ID. This ordering ensures the Canonical Agent-ID hashes a document containing its own inclusion proof.

8. Log Protocol

The log operator exposes four operations to clients. All operations *MUST* be available over HTTPS at paths relative to the log's base URI (the log_uri value advertised in the Agent Genesis). Implementations *SHOULD* also expose these operations over AGTP at the same logical paths once AGTP transport bindings are stable.

8.1. Submit Statement

POST {log_uri}/statements

Request body: a single COSE_Sign1 statement per Section 6. Content-Type application/agtp-log-statement+cose.

Response on success: HTTP 201 with the resulting receipt per Section 7. Content-Type application/scitt-receipt+cose.

Response on signature failure: HTTP 400 with body indicating which validation step failed (signature, issuer match, subject format, event type, payload schema, or genesis hash mismatch).

Submission is normatively restricted to the log operator itself: external parties *MUST NOT* be able to submit statements directly. The log operator validates and signs internally, then commits. This preserves the log's role as an attestation of the operator's own governance decisions.

8.2. Retrieve Receipt

GET {log_uri}/receipts/{statement-hash}

Path parameter: hex-encoded SHA-256 hash of the canonical CBOR statement.

Response on success: HTTP 200 with the receipt per Section 7.

Response on miss: HTTP 404 with body indicating whether the hash is unknown to this log or refers to a statement that has not yet been committed.

8.3. Retrieve Inclusion Proof

GET {log_uri}/proofs/inclusion?leaf-index={index}&tree-size={size}

Query parameters:

* leaf-index: the position of the statement (zero-indexed)

* tree-size: the STH tree size against which the proof is computed

Response on success: HTTP 200 with a CBOR-encoded inclusion proof per [RFC9162] Section 4.11. The proof is the audit path; the caller is responsible for verifying it against the STH at the requested tree size.

This operation duplicates information already in receipts; it exists for clients that hold a position and need a fresh proof against a different STH than the one originally bound in the receipt.

8.4. Retrieve Consistency Proof

GET {log_uri}/proofs/consistency?first-tree-size={n}&second-tree-size={m}

Query parameters:

* first-tree-size: the smaller tree size

* second-tree-size: the larger tree size

Response on success: HTTP 200 with a CBOR-encoded consistency proof per [RFC9162] Section 4.12. The proof attests that the tree at size n is a prefix of the tree at size m.

Used by monitors to detect split-view attacks. A monitor that observes two different signed tree heads from the same log operator at the same tree size *MUST* treat that as evidence of operator misbehavior.

8.5. Retrieve Signed Tree Head

GET {log_uri}/sth

Response on success: HTTP 200 with the current STH per [RFC9162] Section 4.10. The response is a CBOR-encoded structure carrying tree size, root hash, timestamp, and the log operator’s signature over those fields.

The STH is the anchor for all inclusion and consistency proofs issued by the log. Verifiers *MUST* retrieve and cache the STH before validating any proof.

9. Discovery

A verifier holding a Canonical Agent-ID locates the log containing the relevant inclusion proof via the log_uri field of the Agent Genesis. This document proposes log_uri as a new field to be added to the Agent Identity Document schema defined in [AGTP]; adoption of the field is anticipated in a future revision of [AGTP], coordinated with the publication of this document.

The log_uri field, once adopted, carries an absolute HTTPS URI pointing to the base of the log’s protocol surface. All log protocol operations defined in this document are exposed at paths relative to this URI.

A verifier that has retrieved an Agent Genesis (whether via canonical Agent-ID resolution per [AGTP], via the AGTP DISCOVER method, or via any other means) *MUST*:

1. Read the log_uri field.

2. Validate that the URI uses the https:// scheme.

3. Open a TLS connection to the log endpoint.

4. Retrieve the STH, verify its signature against the governance platform’s published key.

5. Retrieve the inclusion proof for the Agent Genesis (either by parsing the embedded log_inclusion_proof receipt or by requesting a fresh proof from the log).

6. Verify the inclusion proof against the STH.

If the log_uri field is absent from an Agent Genesis whose verification_path is log-anchored, the Agent Genesis is *malformed* and the verifier *MUST* reject the agent identity.

If the log_uri is present but the log is unreachable at verification time, the verifier *MAY* fall back to the cached inclusion proof embedded in the Agent Genesis's log_inclusion_proof field if such a cached proof exists and remains valid against the verifier's last-known good STH.

Until the log_uri field is adopted into [AGTP], implementations *MAY* convey the log URI through other means: as an out-of-band configuration value keyed on the governance platform's issuer URL, or as a well-known location relative to issuer (e.g., {issuer}/log). These mechanisms are deployment-specific and *MUST NOT* be relied on for cross-platform interoperability. The normative discovery mechanism is the log_uri field once adopted.

A well-known URI discovery mechanism (e.g., /.well-known/agtp-log) is not specified in this revision. AGTP v07 does not currently define any well-known URI conventions, and introducing one solely for AGTP-LOG discovery is premature. Future revisions of AGTP-LOG *MAY* define a well-known discovery path if the broader AGTP family adopts well-known URI conventions.

10. Integration with AGTP

The log_inclusion_proof field defined in the Agent Identity Document of [AGTP] v07 references the inclusion proof produced by this log. A verifier retrieves the proof, validates it against the log's signed tree head, and confirms the Agent Genesis is committed to the log.

The verification_path: log-anchored value defined in [AGTP-TRUST] indicates that the agent's Trust Tier 1 verification depends on a log-committed Agent Genesis. Verifiers consuming such an agent *MUST* perform the inclusion-proof verification described in Section 9 before granting Authority-Scope decisions on the agent's behalf.

The log_uri field is proposed by this document as an addition to the Agent Identity Document defined in [AGTP]. Its adoption is anticipated in a future revision of [AGTP], coordinated with the

publication of this document. Until adoption, deployments *MAY* convey log location through the alternative mechanisms described in Section 9.

11. Security Considerations

11.1. Audited Party Forgery

A common threat model for audit systems is the audited party forging, altering, or omitting their own audit records. AGTP-LOG addresses this structurally rather than through policy controls.

Statements in AGTP-LOG are signed by the governance platform that operates the log, not by the agent the statements concern. The agent does not control the log operator’s signing key. An agent cannot:

* Forge a statement attributing actions to a different agent identity (statements bind to agtp-subject and are signed by the log operator)

* Omit an event from its own audit trail (commit decisions are made by the log operator, not by the agent)

* Alter a committed statement after the fact (the verifiable data structure of [RFC9162] prevents retroactive modification without breaking tree consistency)

* Replace its audit history with a forged sequence (statements are identified by position in an append-only log; replacing history is detectable by monitors observing the log)

The threat shifts to compromising the governance platform itself (Section 11.2) rather than compromising the agent. This is a higher bar architecturally: the governance platform’s signing key is a long-lived infrastructure credential typically protected with hardware security modules and key ceremonies, while agent credentials may be more numerous and shorter-lived.

This structural property distinguishes AGTP-LOG from audit approaches where the audited party participates in writing the audit trail. In application-layer audit logging on HTTP-based substrates, an agent or its operator typically generates audit records about its own behavior; ensuring those records cannot be altered by the audited party requires additional cryptographic machinery layered on top of the substrate. In AGTP-LOG, the separation is built into the protocol architecture.

11.2. Log Operator Compromise

The log operator is also the issuer of statements in the AGTP-LOG model in this revision. A compromised log operator could:

* Sign a fraudulent Agent Genesis for an Agent-ID it does not control, commit it to the log, and present the resulting receipt as proof of a non-existent agent's existence.

* Refuse to commit valid statements, denying agents legitimate Trust Tier 1 standing.

* Backdate statements by manipulating its internal clock prior to signing.

* Operate two parallel logs presenting different signed tree heads to different verifiers (a split-view attack).

The first three threats are inherent to any single-issuer log model and are partially mitigated by the witness mechanism (deferred to a future revision) and by external monitors that detect inconsistency. The split-view attack is specifically detectable via the consistency proof operation: a monitor that observes a second STH at the same tree size as a previously-observed STH from the same operator *MUST* treat the divergence as evidence of compromise.

Deployments that require defense against log operator compromise *SHOULD* require witness countersignatures on STHs (when a future revision specifies the witness protocol) and *SHOULD* subscribe to independent monitors that surface inconsistency evidence.

11.3. Witness Collusion

In a future revision incorporating witnesses, two or more witnesses colluding with a malicious log operator could countersign inconsistent STHs. This threat is structurally identical to log operator compromise from the verifier's perspective: the verifier trusts the witness set, and a colluding witness set is no better than a colluding operator. Mitigations are deferred to a future revision along with the witness protocol itself.

11.4. Private Key Handling

The log operator's signing key is the root of all trust in AGTP-LOG. Operators *MUST* protect this key with controls appropriate to a long-lived signing identity: hardware security modules, key ceremonies for key generation and rotation, and documented operational procedures for key recovery.

Key rotation procedures and the binding of historical STHs to retired keys are not specified in this revision; see Section 13.

11.5. Statement Privacy

Statements in AGTP-LOG are public by design. The Agent Genesis itself is published as the canonical identity record of an agent; the lifecycle events that follow it are operational facts. AGTP-LOG is not a confidential audit log.

Operators *MUST NOT* commit statements containing secrets, operational credentials, or other sensitive material. The validation gates in Section 6 are insufficient to filter privacy violations; preventing privacy leaks in statement payloads is the operator’s responsibility.

11.6. Replay and Statement Substitution

A statement signed by a valid log operator and committed to a log is durable: the log operator cannot retroactively replace it without breaking the consistency property of the tree. However, duplicate submissions of the same logical statement at different positions are not prevented by the protocol; the log operator *SHOULD* deduplicate statements before commit, using the canonical CBOR encoding as the deduplication key.

11.7. Threat Model Reference

The full Trust Tier 1 threat model is specified in [AGTP] Section 7 and [AGTP-TRUST]. AGTP-LOG is one of three equivalent verification paths; the threat model considers attacks against the log-anchored path within the broader context of dns-anchored and hybrid alternatives.

12. Future Registry Considerations

The AGTP family includes several constructs (status codes, header fields, identity document fields, method names, media types) whose long-term stewardship will require coordinated management once the ecosystem matures. Several of these constructs are AGTP-specific and have no current home in existing standards-body registries. The appropriate venue for AGTP registry stewardship — IANA, an AGTP governance forum, a foundation, or some combination — is an open question for the broader AGTP family and is not decided in this document.

This section enumerates the registry-shaped artifacts AGTP-LOG defines, so that whichever stewardship model the AGTP family eventually adopts has a clear inventory of what AGTP-LOG contributes. No commitment to any specific registry venue is made by this document.

12.1. AGTP-LOG Event Types

The five event types defined in this revision form an extensible event-type set:

Event Type	Reference
agent-genesis-issued	This document, Section 4
agent-genesis-revoked	This document, Section 4
agent-lifecycle-suspended	This document, Section 4
agent-lifecycle-reinstated	This document, Section 4
agent-lifecycle-deprecated	This document, Section 4

Table 4: AGTP-LOG Event Types Registry

New event types *MUST* be defined in a published specification that describes the triggering condition, subject, payload schema, and the validation rules a log operator applies before commit. Implementers introducing event types prior to a formal registry process *SHOULD* prefix experimental types with x- to avoid namespace collisions with future registered types.

12.2. Media Types

This document defines two media types for AGTP-LOG message serialization:

application/agtp-log-statement+cose AGTP-LOG statement envelope (COSE_Sign1 per [RFC9052]).

application/agtp-log-statement+cbor AGTP-LOG statement payload (CBOR, deterministic encoding per [RFC8949] Section 4.2). Used as the payload of the COSE_Sign1 envelope.

The SCITT receipt media type application/scitt-receipt+cose is defined in [RFC9943] and reused without redefinition.

12.3. COSE Header Parameters

The following protected-header parameter names are introduced by this document. CBOR label assignments are not made by this document; implementations prior to a formal registry process *SHOULD* use string-form labels in the protected header rather than numeric labels, and *MUST* document their string-to-numeric mapping for downstream tooling.

Parameter Name	Use
agtp-event-type	Event type string for the AGTP-LOG statement.
agtp-subject	Canonical Agent-ID (32 bytes) the statement concerns.
agtp-issuer	Governance platform URI signing the statement.
agtp-issued-at	Issuance timestamp per RFC 3339.
verifiable-data-structure	Receipt parameter; *MUST* be RFC9162_SHA256.
agtp-statement-position	Log position of the statement the receipt covers.
agtp-statement-hash	SHA-256 of the canonical statement encoding.
agtp-signed-tree-head	The STH the receipt attests against.

Table 5: AGTP-LOG Protected Header Parameter Names

12.4. Identity Document Field

The log_uri field defined in Section 9 is proposed as an addition to the Agent Identity Document defined in [AGTP]. Its adoption is anticipated in a future revision of [AGTP] and will be reflected in whichever field-management mechanism the AGTP family adopts for the Agent Identity Document schema.

13. Open Items

The following items are explicitly out of scope for this revision and are anticipated in future revisions:

* *Witness protocol.* Witness selection, key publication, countersigned STH schema, witness inclusion in receipts, and the failure modes when witness signatures are inconsistent.

* *Monitor role.* Formal monitor specification: polling cadence, consistency-proof retention, split-view reporting protocol, and monitor discoverability.

* *Federation across governance platforms.* Cross-platform agent identity resolution when multiple governance platforms operate independent logs and an agent originates from one but is invoked via another. This revision assumes per-platform isolation.

* *Key rotation.* Procedures for rotating the log operator’s signing key, binding historical STHs to retired keys, and ensuring older inclusion proofs remain verifiable across key rotations.

* *Log operator delegation.* Whether and how a governance platform may delegate log operation to a separate entity while preserving the issuer-equals-operator property.

* *Statement deduplication.* A normative deduplication algorithm that is robust to byte-level variations in CBOR encoding (the protocol currently relies on deterministic encoding to make this tractable).

* *Discovery via well-known URI.* Specification of a well-known discovery endpoint per [RFC8615] once the broader AGTP family adopts well-known URI conventions.

* *CBOR label assignments for AGTP-LOG header parameters.* The table in Section 12 marks these as TBD pending the outcome of AGTP family registry stewardship decisions. Until numeric labels are assigned, implementations *SHOULD* use the string-form parameter names and document their string-to-numeric mapping for downstream tooling.

* *Registry stewardship venue.* Whether AGTP-LOG registries (event types, media types, header parameters, identity document fields) are stewarded through IANA, an AGTP governance forum, or another mechanism is an open question for the AGTP family. This document enumerates the registry-shaped artifacts it contributes without committing to a venue.

14. Contributors

Feedback from Scott Courtney (GoDaddy / ANS) on SCITT alignment and production deployment considerations informed this draft. The event-type registry structure follows the SCITT philosophy of deterministic CBOR statement envelopes with externally-managed event-type names, allowing the AGTP ecosystem to add new event categories without spec revisions.

15. References

15.1. Normative References

[AGTP] Hood, C., "Agent Transfer Protocol (AGTP)", Work in Progress, Internet-Draft, draft-hood-independent-agtp-07, 2026, <https://datatracker.ietf.org/doc/html/draft-hood-independent-agtp-07>.

[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/rfc/rfc2119>.

[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/rfc/rfc8174>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/rfc/rfc8949>.

[RFC9052] Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", STD 96, RFC 9052, DOI 10.17487/RFC9052, August 2022, <https://www.rfc-editor.org/rfc/rfc9052>.

[RFC9162] Laurie, B., Messeri, E., and R. Stradling, "Certificate Transparency Version 2.0", RFC 9162, DOI 10.17487/RFC9162, December 2021, <https://www.rfc-editor.org/rfc/rfc9162>.

[RFC9943] Birkholz, H., Cooley, D., Delignat-Lavaud, A., Fournet, C., and Y. Deshpande, "An Architecture for Trustworthy and Transparent Digital Supply Chains (SCITT)", RFC 9943, DOI 10.17487/RFC9943, January 2026, <https://www.rfc-editor.org/info/rfc9943>.

15.2. Informative References

[AGTP-CERT]
Hood, C., "AGTP Agent Certificate Extension", Work in Progress, Internet-Draft, draft-hood-agtp-agent-cert-00, 2026, <https://datatracker.ietf.org/doc/html/draft-hood-agtp-agent-cert-00>.

[AGTP-TRUST]
Hood, C., "AGTP Trust and Verification Specification", Work in Progress, Internet-Draft, draft-hood-agtp-trust-00, 2026, <https://datatracker.ietf.org/doc/html/draft-hood-agtp-trust-00>.

[AUDIT-ARCH]
Kuehlewind, M. and H. Birkholz, "AI Agent Auditing Architecture", Work in Progress, Internet-Draft, draft-kuehlewind-audit-architecture-00, 2026, <https://datatracker.ietf.org/doc/html/draft-kuehlewind-audit-architecture-00>.

[IDP] Sato, T., "Intent Declaration Protocol", Work in Progress, Internet-Draft, draft-sato-soos-idp-00, 2026, <https://datatracker.ietf.org/doc/html/draft-sato-soos-idp-00>.

[RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, DOI 10.17487/RFC8615, May 2019, <https://www.rfc-editor.org/rfc/rfc8615>.

Author’s Address

Chris Hood
Nomotic, Inc.
Email: chris@nomotic.ai
URI: https://nomotic.ai