Operations and Management Area
Internet-Draft
Intended status: Standards Track
Expires: 29 November 2026

L. Melegassi
Catellix

28 May 2026

The MVPS Operational Log Format: Append-Only, Hash-Chained, Externally-Anchored Audit Logs
draft-melegassi-opsawg-mvps-logging-00

Abstract

Multi-Vantage Path Snapshot (MVPS) deployments run in critical environments where the operational record must be auditable: an investigator, regulator, or independent witness must be able to detect any after-the-fact alteration of what the system observed and did. This document specifies the MVPS operational log format: an append-only stream of structured records, each cryptographically chained to its predecessor, with completeness guaranteed by an external anchor reusing the Coherent-Witness Trust (CWT) checkpoint and the MVPS Proof Envelope.

The format guarantees that recorded events cannot be silently edited, reordered, or interior-deleted, and -- once a head is anchored -- that the exact log prefix is pinned and tail truncation is exposed. The document is explicit about the limits: a bare hash chain cannot prove its own completeness (an anchor is REQUIRED); the chain is binding,
not confidential. Every property is validated by scripts/validate_logging_format.py (8/8 PASS, exit 0) and recorded in evidence/logging_format_receipt.json.

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 29 November 2026.

Copyright Notice

Table of Contents

1. Introduction

MVPS is deployed where the cost of an undetected lie about the past is high: critical infrastructure, regulated networks, contested environments. In such settings the operational log is not a debugging convenience; it is evidence. An evidence log must satisfy one property above all: any alteration of the recorded past is DETECTABLE by a party who did not write it.

This document specifies that log. Records are append-only and hash-chained (each commits to the digest of its predecessor), so the head digest is a binding commitment to the entire ordered history. Completeness -- the guarantee that no tail was dropped -- is provided by anchoring the head externally, reusing machinery the MVPS family already defines.

1.1. Relationship to CWT and the Proof Envelope

The Coherent-Witness Trust draft [I-D.melegassi-santos-ippm-mvps-cwt] defines a per-coordination-window Merkle checkpoint and a witness cosignature surface, but explicitly states it is NOT a long-running append-only log. This document is that log. It REUSES the CWT checkpoint as one anchor target and the MVPS Proof Envelope [I-D.melegassi-ippm-mvps-proof-envelope] as another: a periodic log head MAY be bound as an artifact in an envelope manifest, inheriting the envelope's tamper-evidence and optional post-quantum anchor.

1.2. Scope and Non-Goals

This document specifies:

- the record format and its canonical encoding,
- the chain link and head computation,
- the verification procedure,

- the external-anchor requirement for completeness,
- a redaction mechanism compatible with verification.

This document does NOT:

- provide confidentiality (the chain is binding, not hiding; Section 6 and Section 9);
- guarantee completeness without an anchor (impossible; Section 5, Section 7);
- define a transport; records MAY be shipped over syslog [RFC5424], files, or any ordered channel.

2. Terminology

Record: one log entry (Section 3).

Leaf hash: SHA-256 over the canonical record body with the record_hash field removed.

Chain link: the field prev_hash of record i equals the record_hash of record i-1 (or GENESIS = 64 zero hex digits for i = 0).

Head: the record_hash of the last record (GENESIS if empty).

Anchor: a (head_hash, count) pair published through an external, witnessed channel (a CWT checkpoint or a Proof Envelope manifest).

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "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.

3. Log Record Format

A record is a structured object with the following top-level fields:

seq	unsigned integer, contiguous from 0, MUST equal the record's position;
prev_hash	64 hex digits (SHA-256 of the previous leaf, or GENESIS for seq 0);
ts	RFC 3339 UTC timestamp of record creation;
event	an open structured payload (Section 3.3);
record_hash	64 hex digits, the leaf hash of this record.

3.1. Canonical Encoding and the Leaf Hash

The leaf hash is computed over the canonical JSON [RFC8785] encoding of the record with the record_hash field removed:

```
leaf(r) = SHA-256( JCS( r without "record_hash" ) )
```

Canonicalization makes the digest independent of key insertion order; two encoders MUST agree on the leaf hash. A CBOR profile [RFC8949] MAY be used on constrained links using CBOR deterministic encoding.

3.2. The Chain Link

For seq 0, prev_hash MUST be GENESIS. For seq i >= 1:

```
prev_hash_i = record_hash_{i-1}
```

```
record_hash_i = leaf(r_i)
```

The head after n records is record_hash_{n-1}. By linked timestamping [HS91] the head is a binding commitment to the entire ordered prefix under SHA-256 collision resistance.

3.3. Event Taxonomy and Severity

The event payload SHOULD carry:

- kind — a dotted taxonomy token, e.g. "vantage.join", "bundle.observe", "alarm.raise", "vantage.byzantine", "anchor.checkpoint";
- sev — one of debug, info, notice, warn, error, audit.

Records with sev = "audit" MUST be retained for the deployment's full retention period and MUST be covered by at least one anchor.

4. Verification Procedure

A verifier accepts a log L = (r_0, ..., r_{n-1}) iff, for every i:

1. seq_i == i;
2. prev_hash_i == (GENESIS if i == 0 else record_hash_{i-1});
3. record_hash_i == leaf(r_i).

Any failure localizes the first altered position. Verification is O(n) hashes and requires no secret.

5. External Anchoring (REQUIRED for completeness)

A bare chain proves internal consistency but CANNOT prove it has not been truncated at the tail: any prefix of a valid log is itself a valid log (Section 7, T-LOG-TRUNC-1). Therefore:

- Operators MUST publish anchors a = (head_hash, count) at a chosen cadence (per N records or per T seconds).
- An anchor SHOULD be a CWT checkpoint [I-D.melegassi-santos-ippm-mvps-cwt] (witness-cosigned) or a Proof Envelope manifest entry [I-D.melegassi-ippm-mvps-proof-envelope].
- Completeness is guaranteed only up to the most recently anchored head. Records after the last anchor enjoy edit/reorder/delete evidence but not truncation evidence.

Given an anchor, a verifier rejects any log whose head or count disagrees, exposing truncation and rollback.

6. Redaction

To remove a sensitive value v while keeping the record verifiable, a writer MAY replace v with a salted commitment:

```
c = SHA-256( salt || v )
```

retaining salt in the record. The leaf hash is computed over the committed form, so verification is unchanged and the record stays in the chain. A holder of v reproduces c; a party without v learns only c. This is BINDING, not hiding: it is not encryption and provides no confidentiality guarantee in the IND sense (Section 9).

7. Tamper-Evidence Properties

The following are proved in the companion (docs/MVPS_LOGGING_PROOF.txt) and validated by scripts/validate_logging_format.py:

T-LOG-CHAIN-1	a well-formed chain verifies end to end;
T-LOG-EDIT-1	editing any record's payload breaks the chain;
T-LOG-REORDER-1	reordering two records breaks the chain;
T-LOG-DELETE-1	deleting an interior record breaks the chain;
T-LOG-TRUNC-1	tail truncation is NOT self-detectable; it is detectable with an external anchor;
T-LOG-ANCHOR-1	an anchor (head_hash, count) binds the exact prefix;
T-LOG-CANON-1	canonical encoding is key-order independent;
T-LOG-REDACT-1	salted redaction preserves the commitment.

Each reduces to SHA-256 collision resistance and linked timestamping [HS91], plus witness EUF-CMA [RFC9162] for the anchor.

8. Numerical Receipt

scripts/validate_logging_format.py builds a five-record log, exercises each tamper class, and writes evidence/logging_format_receipt.json with the demo head digest, platform metadata, severity vocabulary, anchor targets, the explicit non-claims, and a SHA-256 of its own canonical body. At time of writing all eight checks PASS (exit 0).

9. Security Considerations

The log provides tamper-EVIDENCE, not tamper-PREVENTION: an attacker with write access can still destroy the file, but cannot do so silently once a head has been anchored (Section 5). Operators MUST anchor at a cadence matched to their threat model; the window between anchors is the maximum silently-truncatable tail.

The chain is not confidential. Sensitive fields MUST be redacted (Section 6) or the whole transport encrypted; the commitment scheme provides binding only.

Quantum considerations follow the Proof Envelope [I-D.melegassi-ippm-mvps-proof-envelope]: SHA-256 retains a halved (Grover) preimage margin of 2^128, which is finite and sufficient; anchors MAY migrate to a post-quantum witness signature without changing any record on the wire.

10. IANA Considerations

This document requests no IANA actions. The event "kind" taxonomy is an open, deployment-defined namespace; a later revision MAY request a registry if interoperability requires it.

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, May 2017.

[RFC8785] Rundgren, A., Jordan, B., and S. Erdtman, "JSON Canonicalization Scheme (JCS)", RFC 8785, June 2020.

[RFC9162] Laurie, B., Messeri, E., and R. Stradling, "Certificate Transparency Version 2.0", RFC 9162, December 2021.

11.2. Informative References

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

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, December 2020.

[HS91] Haber, S. and W. Stornetta, "How to Time-Stamp a Digital Document", Journal of Cryptology 3(2), 1991.

[I-D.melegassi-santos-ippm-mvps-cwt] Melegassi, L. and J. Santos, "MVPS Coherent-Witness Trust", draft-melegassi-santos-ippm-mvps-cwt-00, 2026.

[I-D.melegassi-ippm-mvps-proof-envelope] Melegassi, L., "MVPS Proof Envelope", draft-melegassi-ippm-mvps-proof-envelope-00, 2026.

Appendix A. Worked Example

Five records (vantage.join, bundle.observe, alarm.raise, anchor.checkpoint, vantage.byzantine) chained from GENESIS produce a head digest reproduced exactly by the validator. Editing the alarm.raise record’s d2 value from 38.7 to 0.0 causes verification to fail at seq 2, demonstrating T-LOG-EDIT-1. The full transcript is the output of scripts/validate_logging_format.py.

Author’s Address

Leonardo Melegassi
Catellix
Brazil
Email: melegassi@catellix.com