Independent Submission
Internet-Draft
Intended status: Informational
Expires: 3 September 2026

B. El Khatabi
TrackOne Project

2 March 2026

Verifiable Telemetry Ledgers for Resource-Constrained Environments
draft-elkhatabi-verifiable-telemetry-ledgers-00

Abstract

This document specifies a verifiable telemetry ledger profile for resource-constrained sensing environments. The profile defines how a gateway accepts framed telemetry, applies anti-replay policy, projects accepted frames into canonical facts, builds deterministic daily Merkle commitments, and anchors daily artifacts with external timestamp proofs. OpenTimestamps (OTS) is the default anchoring mechanism; optional parallel attestation methods (RFC 3161 timestamp protocol and peer signatures) are also described.

The goal is interoperability and independent auditability, not new cryptographic primitives.

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 3 September 2026.

Copyright Notice

Table of Contents

1. Introduction

Long-lived telemetry deployments such as environmental monitoring, heritage conservation, and infrastructure health need evidence that measurements were not silently altered after a collection. A coastal sensor array that reports temperature every six hours over a five-year deployment produces tens of thousands of records; stakeholders such as regulators, researchers, and insurers may need to verify, years later, that the data they rely on is the same data the sensors produced.

Existing standards provide important building blocks:

* Deterministic CBOR encoding ([RFC8949]),

* Timestamping via trusted timestamp authorities ([RFC3161]),

* CBOR-based Merkle tree proofs ([COSE-MERKLE]), and

* Supply-chain transparency architectures ([SCITT]).

However, none of these building blocks defines a complete operational profile for the constrained case: devices with intermittent uplinks, limited compute budgets, and no persistent Internet connectivity at the collection point. SCITT assumes a transparency service is reachable for receipt-oriented workflows. COSE Merkle proofs define proof encodings, but not batching policy or anchoring lifecycle. RATS addresses device identity attestation, but not telemetry commitment.

The discussion below compares this profile directly with [SCITT] and [COSE-MERKLE] because those documents define adjacent, but not identical, transparency and proof disclosure models.

This document fills that gap. It specifies a practical profile that combines these building blocks for low-power telemetry systems:

1. Emit encrypted framed telemetry.

2. Ingest and validate frames with anti-replay.

3. Project accepted frames into canonical facts.

4. Build deterministic daily Merkle commitments.

5. Chain days with previous-day root linkage.

6. Anchor the day artifact using external timestamp proofs.

7. Verify independently from disclosed artifacts.

The profile is informed by TrackOne pre-production validation, including constrained uplink simulations, hardware-in-loop testing, and daily batching and verification workflows.

1.1. Relationship to Existing Work

This document is complementary to, not a replacement for, the following work:

* _SCITT_ ([SCITT]): SCITT defines architectures for transparent, append-only logs of signed statements with receipts. This profile differs in that it is optimized for disconnected, daily-batch operation where a transparency service is not assumed to be continuously reachable.

* _COSE Merkle Tree Proofs_ ([COSE-MERKLE]): COSE-MERKLE defines proof encodings. This profile defines a batching and commitment contract and may use COSE-based proof encodings in a future revision.

* _RATS_: This profile explicitly defers device identity, attestation, and key lifecycle to deployment-specific mechanisms.

* _RFC 8949_ ([RFC8949]): This profile defines a TrackOne deterministic CBOR commitment profile for gateway-side commitments. It does not attempt to define a general-purpose CBOR profile beyond the commitment path described here.

2. Conventions and 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 all capitals, as shown here.

Terms:

* Frame: One NDJSON telemetry unit containing header and AEAD fields.

* Fact: Canonical telemetry record projected from an accepted frame.

* Commitment profile: The serialization, hash, and Merkle rules that produce deterministic commitment outputs.

* Ledger set for day D (F_D): The set of accepted facts committed for day D.

* Day artifact: The authoritative canonical day record written as day/YYYY-MM-DD.cbor.

* Authoritative: Used for the canonical artifact that verifiers MUST treat as the cryptographic source of truth.

* Projection: A non-authoritative representation (for example JSON) derived from an authoritative artifact.

* OTS metadata sidecar: proofs/YYYY-MM-DD.ots.meta.json, linking an artifact digest to an OTS proof path.

* Replay unit: The pair (dev_id, fc), where fc is a frame counter for device dev_id.

* Day boundary: A UTC calendar day boundary. Day labels in this profile use YYYY-MM-DD in UTC.

* Disclosure class: The level of artifact disclosure associated with a verification claim.

3. System Roles

* Device (Pod): Produces framed telemetry.

* Gateway: Validates, decrypts, applies anti-replay, projects facts, batches, and anchors day artifacts.

* Verifier: Recomputes commitments and validates proofs from disclosed artifacts.

* OTS Calendar(s): Provides OTS attestations for day artifact hashes.

* Optional TSA: An RFC 3161 timestamp authority over the same digest.

* Optional Peers: Co-sign daily roots for short-term provenance.

4. Data and Commitment Model

4.1. Frame Contract

A frame is transported as NDJSON with fields:

```
{
  "hdr": { "dev_id": 101, "msg_type": 1, "fc": 42, "flags": 0 },
  "nonce": "base64-24B",
  "ct": "base64-ciphertext",
  "tag": "base64-16B"
}
```

Figure 1

Gateways MUST validate header field presence, header ranges, and AEAD authentication before fact emission.

* hdr.dev_id MUST be an unsigned integer in the range 0..65535.

* hdr.msg_type MUST be an unsigned integer in the range 0..255.

* hdr.fc MUST be an unsigned integer in the range 0..(2^32-1).

* hdr.flags MUST be an unsigned integer in the range 0..255.

* nonce MUST be base64 text that decodes to exactly 24 bytes.

* tag MUST be base64 text that decodes to exactly 16 bytes.

Frames that fail parse, range, or AEAD validation MUST be rejected before fact commitment and MUST NOT produce committed facts.

4.2. Anti-Replay Semantics

The replay unit is (dev_id, fc).

* A gateway MUST consume at most one frame per (dev_id, fc) into the ledger.

* Duplicate or out-of-window frames MUST NOT produce committed facts.

* The RECOMMENDED default replay window is 64 frame counter values per device.

* Frames arriving with fc more than window_size behind the highest accepted counter for a device MUST be rejected.

* Frames arriving with fc more than window_size ahead of the highest accepted counter for a device MUST also be rejected, because an excessive forward jump is treated as out of window by the current gateway profile.

Structured Rejection Evidence

Gateways SHOULD produce structured rejection evidence for rejected frames. A rejection record SHOULD include at minimum:

* device_id,

* fc (or null if unavailable),

* reason,

* observed_at_utc, and

* frame_sha256.

Rejection evidence is an audit artifact and MUST NOT be hashed into ledger commitments.

Replay State Persistence

Gateways SHOULD persist replay state across restart. If replay state is lost, gateways SHOULD record a continuity break event and SHOULD NOT silently re-accept counters that could already have been committed.

Scope Limitation

This profile provides tamper-evidence for committed facts. It does not prove the absence of selective pre-commitment drops by a malicious gateway.

4.3. Frame-to-Fact Projection

The frame-to-fact projection transforms a validated, decrypted frame into a canonical fact suitable for commitment. The committed fact is a structured record derived from the decrypted payload, not a copy of transport bytes.

1. The gateway MUST verify AEAD authentication.

2. The gateway MUST decrypt the ciphertext.

3. The gateway MUST parse the decrypted plaintext according to the frame’s msg_type.

4. The gateway MUST construct a fact object.

5. The gateway MUST serialize that fact object under the gateway commitment profile.

6. The canonical fact bytes are then hashed for Merkle inclusion.

A committed fact MUST contain at minimum:

* device_id,

* timestamp,

* nonce, and

* payload.

Ciphertext, raw transport bytes, and the authentication tag MUST NOT be part of the committed fact object. The exact payload schema is deployment-specific; the deterministic projection contract is the normative requirement.

4.4. Deterministic CBOR Commitment Encoding

This section does not define a new general-purpose CBOR variant. It records the narrow deterministic CBOR encoding used for commitment bytes in the current TrackOne gateway and ledger implementation. The identifier trackone-cbor-map-v1 names this commitment recipe so verifiers can tell which byte-level rules were used.

The authoritative commitment artifacts, namely CBOR fact artifacts and the canonical day artifact, use a constrained subset of deterministic encoding under Section 4.2.1 of [RFC8949]. For TrackOne commitment bytes, the following concrete choices apply:

* All commitment-path items MUST use definite-length encoding.

* Integers MUST use the shortest encoding width permitted by [RFC8949].

* Map keys MUST be CBOR text strings.

* Map keys MUST be sorted by encoded key length ascending, then by lexicographic order of the encoded key bytes.

* Finite floating-point values MUST be encoded using the shortest of float16, float32, or float64 that exactly preserves the value.

* NaN, positive infinity, and negative infinity MUST be rejected in commitment paths.

* CBOR tags MUST NOT appear in commitment bytes.

* Supported values are unsigned integers, negative integers, byte strings, text strings, arrays, maps, booleans, null, and deterministic finite floats.

Implementations MUST NOT accept generic CBOR serializers as authoritative commitment encoders. An encoder is acceptable only if it yields the same bytes as these rules.

JSON projections of fact artifacts and day artifacts are optional and non-authoritative. They MUST NOT be used as commitment inputs. When produced, such projections SHOULD follow [RFC8785].

Device-side or embedded components MAY use other internal encodings, including different deterministic CBOR layouts optimized for local constraints. Those encodings are not the authoritative commitment encoding described here unless they are explicitly identified by a distinct commitment_profile_id and verified under their own rules.

4.5. Deterministic Commitment Tree Calculation

For a given day D, the current commitment profile computes a daily root from the canonical fact commitment bytes produced under Section 4.4. The following steps describe that calculation.

Leaf Digests

* Each canonical fact byte string is hashed with SHA-256, yielding a 32-byte leaf digest.

Digest Ordering

* To make the daily root independent of file order or ingest order, leaf digests MUST be sorted in ascending byte order before reduction.

* Lowercase hexadecimal is a representation format for artifacts and examples only; internal Merkle computation operates on raw hash bytes.

* Sorting by lowercase hexadecimal is equivalent to bytewise ascending order over the raw digests.

Pairwise Reduction

* The sorted digests are reduced pairwise by computing SHA-256(left_child_bytes || right_child_bytes), where both operands are raw 32-byte digests.

* If a layer has an odd number of digests, the final digest is duplicated to form the last pair.

* The current commitment profile does not prepend domain-separation bytes to leaf or parent hashes.

Empty Day

* If no facts are committed for the day, the daily root is the SHA-256 digest of zero bytes: e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855.

The resulting daily root is deterministic for the set of committed facts. Because the leaf digests are sorted before reduction, the result depends on the committed fact set rather than on ingestion order.

Any future change to this calculation that alters commitment bytes (for example, adding domain separation) MUST use a new commitment_profile_id.

4.6. Day Artifact Schema

The authoritative day artifact is a CBOR-encoded day record produced under Section 4.4. The day record contains the following fields:

* version (uint): day-record schema version, currently 1.

* site_id (tstr): site identifier.

* date (tstr): UTC day label in YYYY-MM-DD form.

* prev_day_root (tstr): previous day root as 64 lowercase hexadecimal characters.

* batches (array): array of batch objects.

* day_root (tstr): deterministic day root as 64 lowercase hexadecimal characters.

Each batch object contains:

* version (uint): batch-record schema version, currently 1.

* site_id (tstr): site identifier.

* day (tstr): UTC day label in YYYY-MM-DD form.

* batch_id (tstr): batch identifier.

* merkle_root (tstr): batch Merkle root as 64 lowercase hexadecimal characters.

* count (uint): number of committed facts in the batch.

* leaf_hashes (array of tstr): sorted leaf hashes as lowercase hexadecimal strings.

day/YYYY-MM-DD.cbor is authoritative. The corresponding day/YYYY-MM-DD.json file is a projection only.

This document uses normative field tables rather than CDDL. A future revision may add a formal CDDL appendix if broader independent implementations require it.

4.7. Day Chaining

Day records include prev_day_root.

* The genesis day for a site MUST set prev_day_root to 64 ASCII zero characters, representing 32 zero bytes.

* Non-genesis days MUST set prev_day_root to the previous committed day’s day_root.

Because day labels are UTC-based, chaining semantics are also defined on UTC day boundaries.

5. Artifacts and Verification Bundles

Illustrative artifact layout:

* facts/<fact-id>.cbor — authoritative canonical facts

* facts/<fact-id>.json — optional projections

* day/YYYY-MM-DD.cbor — authoritative canonical day artifact

* day/YYYY-MM-DD.json — optional projection

* blocks/YYYY-MM-DD-00.block.json — block and batch metadata

* day/YYYY-MM-DD.cbor.sha256 — convenience digest

* day/YYYY-MM-DD.cbor.ots — OTS proof

* proofs/YYYY-MM-DD.ots.meta.json — OTS binding metadata

Deployments MAY store artifacts differently and MAY export them as bundles. The path shapes above are illustrative.

At minimum, an OTS sidecar MUST bind:

* artifact,

* artifact_sha256, and

* ots_proof.

Verifiers MUST recompute the day artifact digest and compare it with the sidecar before accepting any proof validation result.

6. Anchoring and Verification

6.1. Anchoring Contract

The generic anchoring contract is simple: a gateway computes the SHA-256 digest of the authoritative day artifact and submits that digest to one or more external timestamping channels. Verifiers MUST first recompute the day artifact digest locally; proof validation occurs only after digest binding validation succeeds.

A deployment conforming to this profile MUST use at least one anchoring channel. OTS is the default channel described by this document; RFC 3161 and peer signatures are optional parallel channels.

6.2. OTS Anchoring Profile

When [OTS] is used, the gateway stamps SHA-256(day/YYYY-MM-DD.cbor) and stores an OTS proof plus an OTS sidecar.

6.2.1. OTS Anchoring Lifecycle

1. _Submission_: the gateway submits the day artifact digest to one or more OTS calendars.

2. _Pending_: a calendar may return an incomplete proof while awaiting Bitcoin commitment.

3. _Upgrade_: the gateway or a background process may later upgrade the proof to include completed attestations.

4. _Verification_: a verifier recomputes the artifact digest and validates the proof.

Gateways SHOULD submit to multiple independent calendars to reduce single-calendar unavailability risk.

6.2.2. Handling Delayed or Failed Anchoring

* *Calendar unreachable*: the gateway MUST still write the day artifact and SHOULD retry later.

* *Upgrade delay*: verifiers SHOULD treat pending proofs as unanchored and flag the condition, not as invalid.

* *Proof retention*: gateways MUST retain proof files for at least the operational retention period of the corresponding day artifacts.

6.3. Optional Parallel Attestation

Deployments MAY also produce:

* An [RFC3161] timestamp response over the same day-artifact digest.

* A peer signature quorum over (site_id, day, day_root, context).

When multiple channels are present, verifiers SHOULD validate all available channels independently and report per-channel results.

If a verifier is configured in strict mode for optional channels, failure of those channels MUST cause overall verification failure.

6.4. Verification

Verifiers SHOULD apply checks in the following fail-fast order:

1. Validate that disclosed artifacts are sufficient for the claimed disclosure class.

2. Recompute the Merkle root from canonical fact CBOR artifacts.

3. Compare the recomputed root to day_root.

4. Recompute SHA-256(day/YYYY-MM-DD.cbor) and compare it to the sidecar artifact_sha256.

5. Validate the OTS proof when OTS is required or present.

6. Validate optional RFC 3161 and peer attestations as configured.

Verifier implementations SHOULD expose machine-usable failure categories:

* malformed or missing artifacts,

* Merkle mismatch,

* missing or invalid OTS proof,

* sidecar mismatch or digest mismatch, and

* optional-channel failure.

7. Disclosure Classes

Verification claims depend on what artifacts are disclosed. This profile defines three disclosure classes.

* *Class A (Public Recompute)*: sufficient material for independent fact-level recomputation.

* *Class B (Partner Audit)*: controlled disclosure with redacted or partitioned fact material.

* *Class C (Anchor-Only)*: existence and timestamp evidence only.

A Class A bundle MUST include:

* all canonical fact artifacts required to recompute the claimed day root,

* the canonical day artifact,

* block and batch metadata, and

* the OTS proof plus its sidecar metadata.

A Class A bundle SHOULD also include a machine-readable verification manifest and SHOULD record the commitment_profile_id.

A Class B bundle MUST include:

* the canonical day artifact,

* block and batch metadata,

* the OTS proof plus sidecar metadata, and

* cryptographic commitments to any withheld or partitioned fact material, and

* a policy statement describing withheld or partitioned fact material.

Class B outputs MUST NOT be represented as publicly recomputable.

A Class C disclosure MUST be labeled as existence and timestamp evidence only and MUST NOT claim fact-level reproducibility.

A Class C bundle MUST include:

* the canonical day artifact, and

* at least one timestamp proof artifact (for example, an OTS proof or an RFC 3161 response).

If a verification manifest is present, it SHOULD include:

* disclosure_class,

* commitment_profile_id,

* artifact path and digest entries,

* per-channel anchor status, and

* a list of checks executed.

8. Versioning

This profile has several independent version surfaces:

* Document revision (for example -00, -01) is editorial and is not part of commitment output.

* Artifact schema versions are carried by the version fields in day and batch records.

* commitment_profile_id identifies the canonical CBOR, hash, and Merkle rules that define commitment outputs.

The commitment profile defined in this document is trackone-cbor-map-v1. If a verifier encounters an unsupported commitment_profile_id, it MUST reject the verification claim rather than silently using a fallback interpretation.

9. Conformance Vectors

Determinism claims in this profile are testable. Implementations that claim conformance to trackone-cbor-map-v1 MUST be able to reproduce the published conformance vectors for that profile.

Published vector sets SHOULD include coverage for:

* empty day,
* single fact,
* odd leaf count,
* power-of-two leaf count,
* duplicate leaf hashes,
* genesis chaining,
* non-genesis chaining, and
* a full Class A disclosure example.

Cross-implementation checks MUST verify byte-for-byte parity across independent implementations. Any mismatch in canonical bytes or roots is a conformance failure.

Published vector bundles MUST include the commitment_profile_id.

10. Security Considerations

This profile does not introduce new cryptographic primitives. Security depends on correct composition of existing primitives and on operational discipline.

Replay and Duplicate Suppression

The (dev_id, fc) replay unit enforces single-consumption: at most one committed fact per unique replay unit.

Tamper Evidence

Once a day artifact is anchored, mutation of that artifact changes its digest and invalidates the proof. Day chaining extends this property across days.

Proof Substitution

Sidecar metadata binds an artifact digest to a proof path. Verifiers MUST recompute the artifact digest independently.

Calendar Trust and Withholding

A compromised or unavailable calendar can delay or withhold proofs. Multi-calendar submission reduces single-calendar dependency but does not eliminate coordinated compromise risk.

Operational Misuse

Test or placeholder proofs MUST NOT be treated as production attestations.

Confidentiality Boundary

This profile addresses integrity and timing provenance only. Payload confidentiality remains the responsibility of the deployment's AEAD and key-management layers.

Pre-Commitment Censorship

This profile proves inclusion and timestamping of committed facts. It does not prove completeness of all observed or emitted frames.

11. Privacy Considerations

Telemetry payloads may include sensitive operational data. Operators SHOULD:

* minimize personally identifiable data in committed artifacts,
* separate identity metadata from measurement payload when possible,
* apply retention and access controls, and
* publish only data appropriate for the chosen disclosure class.

Privacy-preserving disclosures remain valid, but they MUST NOT be described as publicly recomputable unless Class A conditions are met.

12. IANA Considerations

This document has no IANA actions.

In particular, this revision does not request:

* a CBOR tag allocation,
* a media type registration, or
* a new registry entry.

The current profile is identified by in-band version and profile fields, not by IANA allocation.

13. Implementation Status

Note to RFC Editor: Please remove this section before publication.

This section records implementation status at the time of posting, per [RFC7942].

This draft reflects the TrackOne pre-production implementation snapshot at release 0.1.0-alpha.6.

A pre-production reference implementation exists and is aligned with the current 0.1.0-alpha.6 release. It is suitable for interoperability review and validation, but it is not a production deployment.

* Repository: https://github.com/bilalobe/trackone
* License: MIT

14. Interoperability Notes and Open Questions

* Media type strategy for canonical CBOR day artifacts.
* Whether a future revision should define a formal CDDL appendix.
* Whether future disclosure bundles should adopt COSE-MERKLE proof encodings.
* Registry strategy for disclosure and anchor-status vocabularies.
* Whether a future commitment profile should introduce domain separation.

15. References

15.1. 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/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/info/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>.

15.2. Informative References

[RFC3161] Adams, C., Cain, P., Pinkas, D., and R. Zuccherato, "Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)", RFC 3161, DOI 10.17487/RFC3161, August 2001, <https://www.rfc-editor.org/rfc/rfc3161>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/rfc/rfc7942>.

[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/rfc/rfc8785>.

[SCITT] Birkholz, H., Delignat-Lavaud, A., Fournet, C., Deshpande, Y., and S. Lasker, "An Architecture for Trustworthy and Transparent Digital Supply Chains", Work in Progress, Internet-Draft, draft-ietf-scitt-architecture, 2024, <https://datatracker.ietf.org/doc/draft-ietf-scitt-architecture/>.

[COSE-MERKLE] Steele, O., Birkholz, H., Delignat-Lavaud, A., and C. Fournet, "COSE Merkle Tree Proofs", Work in Progress, Internet-Draft, draft-ietf-cose-merkle-tree-proofs, 2025, <https://datatracker.ietf.org/doc/draft-ietf-cose-merkle-tree-proofs/>.

[OTS] OpenTimestamps Project, "OpenTimestamps Protocol and Tooling", <https://opentimestamps.org/>.

Appendix A. Example Day Record (JSON Projection)

This appendix shows a non-authoritative JSON projection of a day artifact. The authoritative artifact is the corresponding CBOR file.

```
{
  "version": 1,
  "site_id": "an-001",
  "date": "2025-10-07",
  "prev_day_root": "<genesis root hex>",
  "batches": [
    {
      "version": 1,
      "site_id": "an-001",
      "day": "2025-10-07",
      "batch_id": "an-001-2025-10-07-00",
      "merkle_root": "9d1f...c2",
      "count": 10,
      "leaf_hashes": [
        "01ab...",
        "7fe2..."
      ]
    }
  ],
  "day_root": "9d1f...c2"
}
```

Figure 2

Appendix B. Example Conformance Vector Bundle

The values in this appendix were generated by the Python gateway code path and cross-checked against trackone_core._native (Rust) for deterministic CBOR and Merkle parity. This vector bundle reflects the 0.1.0-alpha.6 implementation snapshot.

```
commitment_profile_id:
  trackone-cbor-map-v1

fixture fact_a:
  device_id: pod-101
  timestamp: 2026-03-01T12:00:00Z
  nonce: ""
  payload.temp_c: 21.5
```

```
   fixture fact_b:
     device_id: pod-102
     timestamp: 2026-03-01T12:10:00Z
     nonce: "n1"
     payload.temp_c: 22.0

   fixture fact_c:
     device_id: pod-103
     timestamp: 2026-03-01T12:20:00Z
     nonce: "n2"
     payload.temp_c: 22.5

   fixture fact_d:
     device_id: pod-104
     timestamp: 2026-03-01T12:30:00Z
     nonce: "n3"
     payload.temp_c: 23.0

   empty-day-v1:
     artifact_sha256:
       c00c984fdd78476f1044fa52eae94606
       6f403460e6585044c39b125a13ee3d7e
     batch_merkle_root:
       e3b0c44298fc1c149afbf4c8996fb924
       27ae41e4649b934ca495991b7852b855
     day_root:
       e3b0c44298fc1c149afbf4c8996fb924
       27ae41e4649b934ca495991b7852b855

   single-fact-v1:
     fact_cbor_hex:
       a4656e6f6e636560677061796c6f6164a16674656d705f63f94d606964657669
       63655f696467706f642d3130316974696d657374616d7074323032362d30332d
       30315431323a30303a30305a
     fact_leaf_hash:
       bb154e441ccdebec09969f1911b46394
       20f7830825b75b02ac52512aa5d32591

   odd-leaf-layer-v1:
     artifact_sha256:
       6f81c6de96dc635ff29f73a60457205b
       a0874a97b2ad6f9f88b1f61870592825
     day_root:
       6c96b4f201e5f6f1badfef6c84d4003a
       b12a7034daeb20fa7f59c33f43c5ae18
     leaf_hashes_sorted:
       26e4affe56412f9e1d4323b27d3ca54c
       4add4fa971800bc25568c4b175d55581
```

```
      bb154e441ccdebec09969f1911b46394
      20f7830825b75b02ac52512aa5d32591
      e2003581ac4364cb322005c465c8d565
      e69f5578af1a614e2762c222a46fd7a5

  power-of-two-v1:
    artifact_sha256:
      81cc87aaf2ecb8b7d9420faa910814aa
      47dd5c8b1ead76d2da19bef55afa48a8
    day_root:
      57bd26f73115f130dcf877a10c434ba2
      8686196daf81f5e48388833303600e73

  duplicate-leaf-hash-v1:
    artifact_sha256:
      4fafb987ef0df50e5e382a09d140793a
      84180f4a86e67924eab1184e20a11c00
    day_root:
      9166c21933341729c08b3a1f61710d9d
      f5efc5aa00d3af9f596c2e166c65b54e

  genesis-chain-v1:
    artifact_sha256:
      4fb6d4570d4662c63b682e2f2d993e9f
      a01669217b61ff64400b981b50b1a8c2
    day_root:
      bb154e441ccdebec09969f1911b46394
      20f7830825b75b02ac52512aa5d32591

  non-genesis-chain-v1:
    artifact_sha256:
      8969bafb62ad9e9aaa6c8460a52320ba
      107975d06352d6562107c5070d792f7e
    day_root:
      e2003581ac4364cb322005c465c8d565
      e69f5578af1a614e2762c222a46fd7a5

  class-a-bundle-v1:
    disclosure_class: A
    day_root:
      6c96b4f201e5f6f1badfef6c84d4003a
      b12a7034daeb20fa7f59c33f43c5ae18
    artifact_sha256:
      6f81c6de96dc635ff29f73a60457205b
      a0874a97b2ad6f9f88b1f61870592825
```

Figure 3

The full machine-readable vector set is maintained in the TrackOne repository alongside the reference implementation.

Acknowledgments

Early structural drafts of this document were prepared with AI writing assistance. All technical content, design decisions, and normative requirements were reviewed against the TrackOne implementation.

The author thanks the OpenTimestamps project for the public calendar infrastructure used during validation.

Author’s Address

Bilal El Khatabi
TrackOne Project
Email: elkhatabibilal@gmail.com