

Independent Submission  
Internet-Draft  
Intended status: Informational  
Expires: 25 September 2026

B. El Khatabi  
TrackOne Project  
24 March 2026

Verifiable Telemetry Ledgers for Resource-Constrained Environments  
draft-elkhatabi-verifiable-telemetry-ledgers-02

Abstract

This document specifies a verifiable telemetry ledger profile for resource-constrained sensing environments. It defines the gateway-side contract for accepting framed telemetry, enforcing anti-replay policy, projecting accepted frames into canonical facts, computing deterministic daily Merkle commitments, chaining day records, and binding authoritative day artifacts to external timestamp evidence. OpenTimestamps (OTS) is the default anchoring channel; RFC 3161 timestamp responses and peer signatures are optional parallel channels.

The goal is interoperability and independent auditability, not new cryptographic primitives. Verification claims are limited by the disclosed artifacts, the claimed disclosure class, and the checks the verifier actually executes. A successful result establishes internal consistency and correct proof binding for the disclosed bundle; it does not by itself establish dataset completeness, physical truth of measurements, or safety for autonomous actuation.

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 25 September 2026.

## Copyright Notice

Copyright (c) 2026 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

## Table of Contents

1. Introduction . . . . .	3
1.1. Relationship to Existing Work . . . . .	6
1.2. SCITT and RATS Integration Pattern . . . . .	6
2. Conventions and Terminology . . . . .	7
3. System Roles . . . . .	8
4. Data and Commitment Model . . . . .	8
4.1. Frame Contract . . . . .	9
4.2. Anti-Replay Admission Criterion . . . . .	9
4.3. Frame-to-Fact Projection . . . . .	11
4.4. Deterministic CBOR Commitment Encoding . . . . .	13
4.5. Deterministic Commitment Tree Calculation . . . . .	14
4.6. Day Artifact Schema . . . . .	15
4.7. Day Chaining . . . . .	16
5. Artifacts and Verification Bundles . . . . .	17
6. Anchoring and Verification . . . . .	18
6.1. Anchoring Contract . . . . .	18
6.2. OTS Anchoring Profile . . . . .	19
6.2.1. OTS Anchoring Lifecycle . . . . .	19
6.2.2. Handling Delayed or Failed Anchoring . . . . .	19
6.2.3. Proof Status Vocabulary . . . . .	20
6.3. Optional Parallel Attestation . . . . .	20
6.4. Verification . . . . .	20
7. Disclosure Classes . . . . .	22
8. Versioning . . . . .	24
9. Conformance Vectors . . . . .	25
10. Security Considerations . . . . .	26
11. Privacy Considerations . . . . .	27
12. IANA Considerations . . . . .	27
13. Interoperability Notes and Open Questions . . . . .	28
14. References . . . . .	28
14.1. Normative References . . . . .	28
14.2. Informative References . . . . .	28
Appendix A. Example Day Record (JSON Projection) . . . . .	29
Appendix B. Illustrative Conformance Vector Bundle . . . . .	30
Appendix C. Minimal Pod Requirements . . . . .	32

Appendix D. Verification Manifest CDDL . . . . .	32
Acknowledgments . . . . .	33
Author's Address . . . . .	33

## 1. Introduction

Long-lived telemetry deployments such as environmental monitoring, heritage conservation, and infrastructure health need evidence that measurements were not silently altered after collection, even when the collection point has intermittent connectivity and the evidence must be verified long after the original ingest path is gone. Stakeholders such as operators, auditors, researchers, regulators, and insurers may need to confirm that disclosed telemetry is the same telemetry the gateway committed on the day it was accepted.

This profile standardizes the gateway-side interoperability surface for that problem: admission semantics for framed telemetry, deterministic frame-to-fact projection, commitment encoding, day artifact structure, disclosure classes, and verifier-facing proof binding.

TrackOne is the open-source reference implementation and documentation set from which this profile has been extracted. The reference implementation publishes machine-readable CDDL for the authoritative commitment family and for TrackOne SCITT statement payloads, together with schemas and examples, but those repository artifacts are implementation evidence rather than normative sources. See [TRACKONE].

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]).

This document does not replace those building blocks. It profiles them for a constrained batch workflow in which a gateway may need to accept frames locally, produce authoritative day artifacts, and complete external proof collection later. SCITT is relevant at publication and transparency boundaries, COSE Merkle proofs are relevant to proof encoding, and RATS is relevant to device identity attestation, but none of those specifications alone defines the commitment contract used here.

The discussion below positions this profile relative to [SCITT], [COSE-MERKLE], and [RFC9334] because those documents define adjacent, but not identical, trust and disclosure boundaries.

Concretely, this profile defines the following contract 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 of disclosed artifacts.

Device reporting cadence and gateway artifact cadence are distinct. A pod may report at fixed or irregular intervals, but this profile uses one UTC day as the batching unit for authoritative artifacts. Other batching intervals are possible design choices for future profiles; they are not implicit variations of this one. One UTC day is used in the current profile because it gives a predictable rollover boundary, bounded artifact size, and a stable unit for later disclosure and audit.

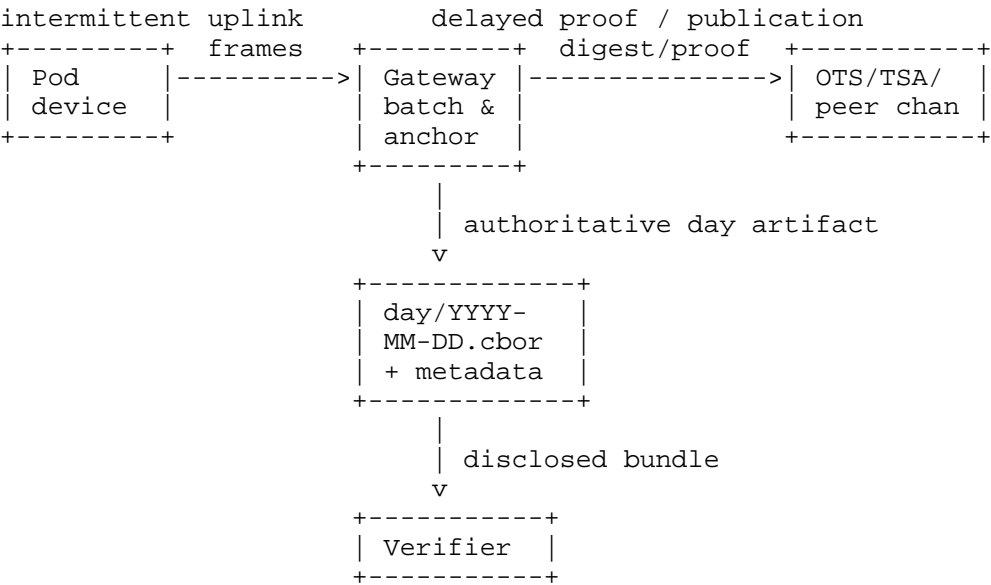


Figure 1: Reference Architecture and Trust Boundaries

Figure 1 shows the intended trust boundary. In the current reference deployment, intermittency primarily applies on the pod-to-gateway path.

External timestamping or later publication may also be delayed, but those delays do not change the gateway-side commitment contract. A disclosed bundle MAY later be registered with a SCITT service.

SCITT is not required to create the gateway-side commitment artifact.

The design is informed by TrackOne reference-implementation validation, including constrained uplink simulation, hardware-in-loop testing, and day-batch verification workflows.

Verification claims are intentionally scoped. A successful result means that the disclosed artifacts are internally consistent, that the authoritative day artifact was bound correctly to the disclosed proof material, and that the checks reported by the verifier succeeded. It does not prove that every observed frame was committed, that the measurements are physically correct, or that any downstream decision is safe.

### 1.1. Relationship to Existing Work

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

- \* `_SCITT_ ([SCITT])`: SCITT defines transparency-service architectures for signed statements and receipts. This profile can be published through such systems, but it standardizes the ledger and verification contract that exists before registration with a transparency service.
- \* `_COSE Merkle Tree Proofs_ ([COSE-MERKLE])`: COSE-MERKLE defines proof encodings. This profile defines the batching, canonicalization, and verification contract that such encodings could carry in a future revision.
- \* `_RATS_ ([RFC9334])`: This profile explicitly defers device identity, attestation, and key lifecycle to deployment-specific mechanisms.
- \* `_RFC 8949_ ([RFC8949])`: This profile defines a deterministic CBOR commitment profile for gateway-side commitments. It does not attempt to define a general-purpose CBOR profile beyond that commitment path.

### 1.2. SCITT and RATS Integration Pattern

This profile can be used alongside SCITT rather than instead of it. The current TrackOne implementation defines day-scoped and bundle-scoped SCITT statement payloads that bind verifier-visible TrackOne artifacts, but the transparency service remains responsible for statement registration, receipt issuance, and service policy. This document therefore standardizes the gateway-side commitment and disclosure contract that may later be carried inside a SCITT statement.

Likewise, this profile does not replace RATS. Device identity, attestation, and evidence appraisal remain deployment-specific inputs. The gateway logic defined here assumes that any required device identity or attestation decisions have already been handled by local policy or by a companion architecture such as RATS.

## 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 UTF-8 JSON text serialized as a single newline-delimited record, containing header and AEAD fields ([NDJSON]).
- \* **Fact:** Canonical telemetry record admitted after gateway validation and projection. The term refers to the committed record shape, not to physical truth of the underlying measurement.
- \* **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.
- \* **Block metadata:** A standalone projection of a batch object (for example blocks/YYYY-MM-DD-00.block.json) exported for convenience. When disclosed, it MUST match the corresponding batch object in the authoritative day artifact.
- \* **OTS metadata sidecar:** day/YYYY-MM-DD.ots.meta.json, a separate non-authoritative metadata file associated with an OTS proof and authoritative day artifact, linking the artifact digest to the proof path. It is not a commitment input.
- \* **Anchor evidence:** The proof artifacts and binding metadata disclosed for an external timestamping or attestation channel.
- \* **Peer signature quorum:** A deployment-defined set or threshold of peer signatures over the same day-artifact digest or day root, treated as one optional parallel attestation channel.

- \* **Replay unit:** The pair (dev\_id, fc), where fc is a frame counter for device dev\_id.
- \* **Verification scope:** The set of disclosed artifacts, checks executed, and claim boundaries that a verifier actually asserts for a verification result.
- \* **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

In this document, optional parallel attestation channels are not required for baseline conformance. A conforming deployment **MUST** implement at least one anchoring channel, with OTS as the default channel described here. RFC 3161 timestamp responses and peer signatures **MAY** be absent entirely; when present, they **MUST** bind to the same authoritative day-artifact digest and verifiers **MUST** report their results separately.

- \* **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.
- \* **TSA:** An RFC 3161 timestamp authority over the same digest, when that optional channel is used.
- \* **Peers:** Co-sign daily roots for short-term provenance, when that optional channel is used.

### 4. Data and Commitment Model

This section separates the reference wire framing from the reusable commitment contract. The most deployment-specific surface is the framed transport in Section 4.1. The interoperable core is the deterministic frame-to-fact projection, commitment encoding, Merkle calculation, day artifact, disclosure model, and verification behavior.



#### 4.1. Frame Contract

A frame is transported as newline-delimited JSON (NDJSON; [NDJSON]) with fields:

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

Figure 2

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<sup>32</sup>-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.

The wire shape in this section is the TrackOne reference transport profile. Other deployments MAY use a different transport framing if they preserve the acceptance semantics needed to produce the same canonical fact objects under Section 4.3.

The minimum pod-side requirements needed to emit such frames are summarized in Appendix C.

#### 4.2. Anti-Replay Admission Criterion

For framed telemetry, the replay unit is (`dev_id`, `fc`). A gateway MUST commit a fact from a frame only if all of the following hold:

1. No fact has already been committed for the same replay unit (`dev_id`, `fc`).

2. The frame counter `fc` is within the configured acceptance window for that device relative to durable replay state.
3. No continuity-break, reset, or replay-state-loss condition requires explicit resynchronization before further acceptance.

Frames that do not satisfy these conditions MUST NOT produce committed facts.

The RECOMMENDED default acceptance window is 64 frame counter values per device. Frames with `fc` more than `window_size` behind the highest accepted counter for a device MUST be rejected. In the TrackOne reference gateway profile, frames with `fc` more than `window_size` ahead of the highest accepted counter MUST also be rejected, rather than silently accepted across a large discontinuity.

#### \*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 under this profile MUST contain at minimum:

- \* `pod_id`,
- \* `fc`,
- \* `ingest_time`,
- \* `pod_time`,
- \* `kind`, and
- \* `payload`.

Interoperability depends on two distinct layers: the projection contract that maps an accepted frame into a fact object, and the commitment profile that serializes that fact object and reduces the resulting digests. Independent implementations claiming the same result MUST agree on both layers.

A commitment profile or deployment profile claiming conformance MUST fix the type and source of each committed field so independent implementations can construct identical fact bytes.

For each committed field, the applicable projection contract MUST state whether the field is transport-derived, payload-derived, gateway-assigned, or deployment metadata-derived. Independent implementations MUST NOT substitute a different field source while claiming the same projection contract.

For trackone-canonical-cbor-v1, the committed fact object has six logical fields: pod\_id, fc, ingest\_time, pod\_time, kind, and payload. Its authoritative commitment encoding is a fixed CBOR array whose positional elements are:

1. schema/version discriminator, currently 1;
2. pod\_id encoded as an 8-byte byte string;
3. fc;
4. ingest\_time;
5. pod\_time or null;
6. kind as the profile-defined family discriminator; and
7. payload encoded under the applicable payload family.

The TrackOne-specific field definitions are:

- \* pod\_id MUST be a deterministic pod identifier. When framed telemetry is used, the profile MUST define a deterministic mapping from hdr.dev\_id or equivalent deployment alias to pod\_id. In reader-facing JSON projections and vector notes, the TrackOne reference profile renders the identifier as lowercase hexadecimal text such as 0000000000000065. In authoritative CBOR commitment bytes, the same identifier is encoded as the corresponding 8-byte byte string.
- \* fc MUST be the accepted frame counter as a non-negative integer.
- \* ingest\_time MUST be a UTC integer timestamp fixed by the projection contract.
- \* pod\_time MUST be either a pod-supplied integer timestamp or null when the pod does not supply one.

- \* kind MUST identify the fact family under the applicable projection contract. In authoritative CBOR commitment bytes, this field is the numeric family discriminator defined by the commitment profile. For the current profile, the discriminator mapping is Env=1, Pipeline=2, Health=3, and Custom=250.
- \* payload MUST be the parsed plaintext object associated with the accepted frame.

A different deployment profile MAY define additional logical fields, but such an extension is not compatible with trackone-canonical-cbor-v1 unless it is given a distinct commitment\_profile\_id and corresponding conformance vectors.

An implementation claiming parity with trackone-canonical-cbor-v1 MUST reproduce that exact logical record and its fixed array encoding before applying the deterministic CBOR rules in Section 4.4.

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 and MUST be published for any commitment profile that claims interoperability.

Published conformance vectors for a commitment profile MUST include the post-projection fact objects used as commitment inputs, not only transport frames.

#### 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 TrackOne reference gateway and ledger implementation. The identifier trackone-canonical-cbor-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 bitwise 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:  
e3b0c44298fclcl149afbf4c8996fb92427ae41e4649b934ca495991b7852b855.

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.

The batch objects embedded in the day artifact are the authoritative batch metadata for verification.

- \* For each batch object, `count` MUST equal the length of `leaf_hashes`.
- \* For each batch object, `merkle_root` MUST equal the Merkle reduction of that batch's `leaf_hashes` under Section 4.5.
- \* The multiset union of all batch `leaf_hashes` for the day MUST equal the day's leaf digest multiset from which `day_root` is computed.

`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` - optional standalone block-metadata projection of one batch object
- \* `day/YYYY-MM-DD.cbor.sha256` - convenience digest
- \* `day/YYYY-MM-DD.cbor.ots` - OTS proof
- \* `day/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.

Standalone block metadata files are convenience projections of batch objects already carried in the authoritative day artifact. They are not additional commitment inputs. Verifiers that process disclosed standalone block-metadata projections MUST compare them with the corresponding batch object in the authoritative day artifact and reject mismatches.

Every verification bundle MUST disclose the `commitment_profile_id`. A verifier-facing bundle SHOULD disclose it in a machine-readable verification manifest together with artifact digests, disclosure class, channel status, and the executed or skipped verification checks. If a deployment uses other local verifier metadata, the resulting verification scope MUST still be made explicit in verifier output.

In the TrackOne reference workflow, the verification manifest is the normal machine-readable contract presented to verifiers. This document therefore treats the manifest as the primary disclosure

surface for bundle-level verification metadata. A minimum interoperable manifest shape is defined in Appendix D. The current TrackOne workflow also publishes a stricter JSON Schema-backed manifest that carries additional deployment-local pipeline fields, but those extra fields are not required by this document.

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.

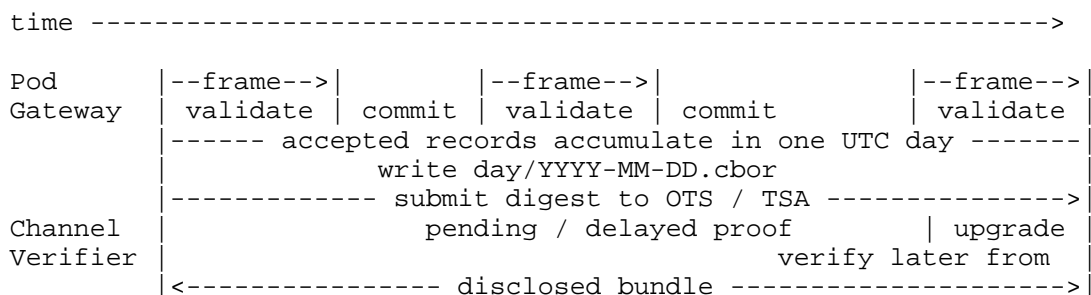


Figure 3: Illustrative Commitment and Proof Lifecycle

Figure 3 illustrates the current lifecycle. The pod-to-gateway transport may be intermittent, while proof completion or later publication may lag behind the authoritative day-artifact write.

## 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.

OpenTimestamps is referenced here as a deployed public timestamping ecosystem rather than an IETF-standardized proof format. Implementations claiming OTS support depend on the interoperable behavior of the public OTS project, its calendar servers, and compatible client tooling. OTS proof-format interoperability is therefore defined operationally by that ecosystem and its reference implementations.

### 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

If OTS submission fails, times out, or yields only an incomplete proof, the gateway MUST still write the authoritative day artifact and MUST treat OTS as a separate channel whose state is not yet complete. The gateway MAY retain a placeholder or incomplete proof artifact and MAY later replace or upgrade it as additional OTS evidence becomes available. Until a valid proof is disclosed and verified, verifiers MUST report the OTS channel as pending, missing, or failed according to the disclosed artifacts and local policy, rather than treating the day artifact itself as invalid. Any later replacement or upgrade of the OTS proof MUST continue to bind to the same authoritative day-artifact digest.

Gateways MUST retain disclosed OTS proof artifacts for at least the operational retention period of the corresponding day artifacts.

### 6.2.3. Proof Status Vocabulary

Verifiers and bundle manifests SHOULD use a consistent status vocabulary for OTS and optional parallel attestation channels:

- \* `verified`: proof validation succeeded for the disclosed artifact binding.
- \* `pending`: a proof exists but is incomplete or awaiting upgrade; this is not equivalent to `invalid`.
- \* `missing`: the expected proof or channel artifact is absent.
- \* `failed`: validation was attempted and did not succeed.
- \* `skipped`: validation was not attempted because of disclosure class, verifier configuration, or local policy.

Whether `missing`, `pending`, or `skipped` is verifier-fatal depends on the disclosure class, local verifier policy, and whether the relevant channel is required.

### 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 MUST first determine the applicable `commitment_profile_id` from disclosed bundle metadata. When the authoritative day artifact does not carry that identifier in-band, the verifier MUST obtain it from the verification manifest or equivalent bundle metadata and MUST reject absent or unsupported values.

Verifiers MUST determine the applicable verification scope from the disclosed artifacts, the claimed disclosure class, and local verifier policy. Reported outcomes MUST NOT claim checks or assurances outside that scope.

Verifiers SHOULD apply checks in the following fail-fast order, subject to the claimed disclosure class:

1. Validate that disclosed artifacts are sufficient for the claimed disclosure class and disclose a `commitment_profile_id`.
2. Validate the authoritative day artifact and any disclosed verification manifest or equivalent bundle metadata.
3. For Class A bundles, recompute canonical fact leaf digests from disclosed fact CBOR artifacts, validate the batch metadata contract, and recompute `day_root`. Compare the recomputed result to the authoritative `day_root`.
4. For Class B bundles, validate the authoritative day artifact, validate any disclosed batch metadata, and report that public fact-level recomputation was not attempted for withheld material. If the bundle discloses withheld-material commitments or policy artifacts, validate those artifacts under the applicable bundle policy.
5. For Class C bundles, skip fact-level and batch-level recomputation and treat the result as anchor-only evidence.
6. Recompute `SHA-256(day/YYYY-MM-DD.cbor)` and compare it to the `sidecar artifact_sha256`.
7. Validate the OTS proof when OTS is required or present.
8. Validate optional RFC 3161 and peer attestations as configured.

When batch metadata is within the exercised verification scope, verifiers MUST apply the following checks before accepting a result:

- \* each batch count equals the length of its `leaf_hashes`;
- \* each batch `merkle_root` equals the Merkle reduction of its `leaf_hashes`;
- \* the union multiset of batch `leaf_hashes` equals the leaf digest multiset derived from disclosed facts when fact artifacts are available; and
- \* any standalone block-metadata projection matches the corresponding batch object in the authoritative day artifact.

Verifier implementations SHOULD expose machine-usable failure categories:

- \* malformed or missing artifacts,
- \* missing or unsupported commitment\_profile\_id,
- \* Merkle mismatch,
- \* batch metadata mismatch,
- \* missing or invalid OTS proof,
- \* sidecar mismatch or digest mismatch,
- \* insufficient disclosure for the claimed verification level, and
- \* optional-channel failure.

Verifier output SHOULD state the claimed disclosure class, the verification scope actually exercised, the per-channel proof status, which checks were executed, which checks were skipped, and whether the resulting claim is public recompute, partial verification, or anchor-only evidence.

Verifier output MUST NOT be represented as proving more than the exercised verification scope. In particular, a successful result does not by itself establish dataset completeness, physical truth of measurements, or suitability for autonomous actuation or sanctions.

## 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 verifier claim for any disclosure class MUST be limited to the verification scope supported by the disclosed artifacts.

Class A is the public-recompute case. A Class A bundle MUST include:

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

- \* the canonical day artifact, including its authoritative batch objects,
- \* any disclosed standalone block-metadata projections, 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`.

Class A permits fact-level recomputation, batch-metadata validation, day-root recomputation, and anchor validation when the corresponding artifacts are disclosed and checked.

Class B is the controlled-disclosure case. Class B outputs MUST NOT be represented as publicly recomputable. A Class B bundle SHOULD include the canonical day artifact, any disclosed standalone block-metadata projections, at least one timestamp proof plus its binding metadata, any disclosed commitments covering withheld material, and a policy artifact describing the withheld or partitioned material.

This document defines the reporting boundary for Class B but does not require one universal withheld-material artifact format. Class B validation is therefore limited to the authoritative day artifact, disclosed batch metadata, any disclosed withheld-material commitments, and any anchor channels present in the bundle. Verifier output SHOULD explicitly state when fact-level recomputation was partial or was not attempted for withheld material.

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 plus the metadata needed to bind it to the day artifact digest and disclose `commitment_profile_id`.

Class C verification validates artifact-digest binding and external timestamp evidence only. It does not establish fact-level reproducibility.

Class C verifier output SHOULD be reported as anchor-only evidence.

\*Bundle Selection Guidance\*

- \* Class A is appropriate when the goal is public recomputation from disclosed canonical facts and authoritative artifacts, for example when a regulator or external researcher needs independent replay of the disclosed day.
- \* Class B is appropriate when the goal is partner or regulator review over a controlled disclosure set that still carries commitment and anchor evidence, for example when some record content must remain withheld while the committed day artifact is still audited.
- \* Class C is appropriate when the goal is existence and timestamp evidence without public fact-level reproducibility, for example when a party only needs to show that a committed artifact existed by a given time.

When a verification manifest is present, it SHOULD include:

- \* `disclosure_class`,
- \* `commitment_profile_id`,
- \* artifact path and digest entries,
- \* per-channel anchor status,
- \* a list of checks executed, and
- \* a list of checks skipped together with the reason for each skip.

## 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-canonical-cbor-v1`. If a verifier encounters an unsupported `commitment_profile_id`, it MUST reject the verification claim rather than silently using a fallback interpretation.



This revision defines exactly one `commitment_profile_id` and does not define an allocation policy or registry for additional profile identifiers. Future `commitment_profile_id` values are out of scope for this document and would require separate specification and review.

Bundles disclose the applicable `commitment_profile_id` via a verification manifest or equivalent bundle metadata. The required OTS sidecar metadata does not currently carry that identifier.

## 9. Conformance Vectors

Determinism claims in this profile are testable. When conformance vectors are published for a commitment profile, implementations that claim conformance to that profile **MUST** be able to reproduce them. For `trackone-canonical-cbor-v1`, the authoritative machine-readable vector corpus is published alongside the TrackOne reference implementation. The appendix in this document is explanatory only.

Published vector sets **SHOULD** include coverage for:

- \* post-projection fact fixtures with fixed field types and values,
- \* 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. Its security depends on correct composition of existing primitives, accurate verifier reporting, and operational discipline around replay state, artifact handling, and proof collection.

### \*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, placeholder, or incomplete proofs MUST NOT be treated as production attestations.

### \*Decision Boundary\*

This profile is not a standalone safety case. A successful verification result MUST NOT be used as the sole basis for autonomous actuation, safety-critical control, or regulatory sanction without deployment-specific policy, corroborating evidence, and human review.

### \*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.

#### \*Hash-Domain Separation\*

The current commitment profile does not prepend domain-separation bytes to leaf or parent hashes. This is acceptable for trackone-canonical-cbor-v1 because the profile is frozen around the published implementation and conformance vectors, but implementers **MUST** treat that choice as part of the profile contract rather than as an accidental omission. Any future profile that introduces explicit leaf or parent domain separation **MUST** use a new `commitment_profile_id`.

## 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.

It 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.

In particular, this revision defines only the document-specific `commitment_profile_id` `trackone-canonical-cbor-v1`. It does not establish a registry or general allocation mechanism for future `commitment-profile` identifiers.

### 13. Interoperability Notes and Open Questions

- \* Media type strategy for canonical CBOR day artifacts.
- \* Whether a future revision should require a verification manifest for every interoperable producer profile rather than treating equivalent local bundle metadata as deployment-specific.
- \* How a future revision should reference the published commitment-family CDDL and vector corpus once multiple independent producers rely on them as primary conformance artifacts.
- \* Whether a future revision should standardize a universal Class B withheld-material artifact format.
- \* 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.

### 14. References

#### 14.1. 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>>.

#### 14.2. Informative References

- [RFC3161] Adams, C., Cain, P., Pinkas, D., and R. Zuccherato, "Internet X.509 Public Key Infrastructure Timestamp Protocol (TSP)", RFC 3161, DOI 10.17487/RFC3161, August 2001, <<https://www.rfc-editor.org/rfc/rfc3161>>.
- [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>>.
- [RFC8610] Bormann, C. and P. Hoffman, "Concise Data Definition Language (CDDL): A Notational Convention to Express CBOR and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <<https://www.rfc-editor.org/rfc/rfc8610>>.
- [NDJSON] NDJSON Project, "NDJSON: Newline Delimited JSON", <<https://github.com/ndjson/ndjson-spec>>.
- [TRACKONE] TrackOne Project, "TrackOne Reference Implementation", 2026, <<https://github.com/bilalobe/trackone>>.
- [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", 2016, <<https://opentimestamps.org/>>.
- [RFC9334] Birkholz, H., Thaler, D., Richardson, M., Smith, N., and W. Pan, "Remote Attestation procedureS (RATS) Architecture", RFC 9334, DOI 10.17487/RFC9334, January 2023, <<https://www.rfc-editor.org/rfc/rfc9334>>.

#### 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": "<64 zero hex chars>",
  "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 4

## Appendix B. Illustrative Conformance Vector Bundle

The authoritative machine-readable conformance vector corpus for trackone-canonical-cbor-v1 is published with the TrackOne reference implementation. The figure below is a reader-oriented sketch of the bundle shape and naming only; it is not the authoritative vector source.

Wrapped hexadecimal values in this appendix are presentation-only; a verifier or implementer should concatenate adjacent lines without inserting whitespace.

The fixtures below are reader-oriented logical record sketches, not the authoritative CBOR array encoding. They show the current post-projection logical content and the active commitment profile identifier, while the published vector corpus carries the exact encoded bytes and digests.

In these reader-oriented sketches, kind is shown using the symbolic family name. In authoritative CBOR commitment bytes, the same field carries the numeric discriminator defined by the current profile; for the fixtures below, Custom corresponds to 250.

```
commitment_profile_id:
  trackone-canonical-cbor-v1

fixture fact_a:
  pod_id: "00000000000000065"
  fc: 1
  ingest_time: 1772366400
  pod_time: null
  kind: Custom
  payload.temp_c: 21.5

fixture fact_b:
  pod_id: "00000000000000066"
  fc: 2
  ingest_time: 1772367000
  pod_time: null
  kind: Custom
  payload.temp_c: 22.0

fixture fact_c:
  pod_id: "00000000000000067"
  fc: 3
  ingest_time: 1772367600
  pod_time: null
  kind: Custom
  payload.temp_c: 22.5

class-a-bundle-v1:
  disclosure_class: A
  commitment_profile_id: trackone-canonical-cbor-v1
  required_artifact_1: facts/<fact-id>.cbor
  required_artifact_2: day/YYYY-MM-DD.cbor
  required_artifact_3: day/YYYY-MM-DD.cbor.ots
  required_artifact_4: day/YYYY-MM-DD.ots.meta.json
  verifier_check_1: day_artifact_validation
  verifier_check_2: fact_level_recompute
  verifier_check_3: ots_verification
```

Figure 5

The published machine-readable vector set carries the exact canonical bytes, digests, expected roots, and the applicable `commitment_profile_id`. This appendix remains illustrative and is not an authoritative conformance corpus.

## Appendix C. Minimal Pod Requirements

This appendix records the minimum expectations for a constrained pod that emits framed telemetry under the current TrackOne reference transport profile. It is descriptive of the current deployment model, not a claim that all conforming deployments share identical hardware.

- \* The pod is not required to sign assertions or to emit SCITT statements.
- \* The pod MUST be able to produce framed transport messages consistent with Section 4.1, including fresh 24-byte transport nonces under the local AEAD policy.
- \* Nonce generation MUST rely on a cryptographically strong pseudorandom source or an equivalent construction with explicit seed discipline. Weak or predictable PRNG state is out of profile.
- \* The pod or an immediately adjacent trusted transport component MUST preserve enough durable state to avoid replay ambiguity across the intended connectivity window.
- \* If uplink availability is intermittent, accepted-but-unsubmitted telemetry MUST either be durably buffered or be retransmittable from durable local state according to deployment policy.
- \* This document does not require one fixed storage budget. The practical tradeoff is between expected outage duration, frame rate, local retention policy, and the gateway's replay/admission contract.

In the current reference deployment, the intermittency assumption primarily applies on the pod-to-gateway path. External timestamping or publication channels may also be delayed, but those delays do not change the pod-side framing requirements.

## Appendix D. Verification Manifest CDDL

This appendix gives a minimum interoperable CDDL ([RFC8610]) shape for a verifier-facing manifest. It captures the bundle metadata needed by this document. A concrete deployment MAY publish a stricter JSON Schema or carry additional deployment-local fields, but it MUST NOT omit these fields when claiming parity with this manifest shape.



```
verification-manifest = {
  "disclosure_class": disclosure-class,
  "commitment_profile_id": tstr,
  "artifacts": { + tstr => artifact-ref },
  "channels": {
    ? "ots": channel-status,
    ? "tsa": channel-status,
    ? "peers": channel-status
  },
  "checks_executed": [* tstr],
  "checks_skipped": [* skipped-check]
}

artifact-ref = {
  "path": tstr,
  "sha256": hex64
}

skipped-check = {
  "check": tstr,
  "reason": tstr
}

channel-status = {
  "status":
    "verified" / "pending" / "missing" / "failed" / "skipped",
  ? "detail": tstr
}

disclosure-class = "A" / "B" / "C"
hex64 = text .regexp "[0-9a-f]{64}"
```

#### 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  
Morocco  
Email: elkhatabibilal@gmail.com