Shunyaya Symbolic Mathematical Network (SSM-NET)

Meaning-Carrying Internet Overlay

Status: Public Research Release (v2.1)
Date: November 10, 2025
Caution: Research/observation only. Not for critical decision-making.
License: Open standard. Free to implement with no fees or registration, provided strictly "as-is" with no warranty, no endorsement, and no claim of exclusive stewardship.
Citation: When implementing or adapting, cite the concept name **"Shunyaya Symbolic Mathematical Network (SSM-NET)"** as the origin of the manifest-first, stamped, alignment-lane overlay for network communication.

SECTION 0 — WHY SSM-NET EXISTS

0A. What SSM-NET is (in brief)

SSM-NET is a **manifest-first overlay** that adds **portable meaning** to any network exchange **without changing routing or encryption** (for the web, see the **HTTP-M** profile: SSM-NET over HTTP/HTTPS). It keeps the original bytes intact while carrying a tiny, bounded lane that expresses **live posture**, a **human band** from a declared rulebook, and a **compact continuity stamp** for replay.

- **Keep your bytes.** Original values remain unchanged by collapse parity: `phi((m,a)) = m` **(collapse parity)**.
- **Add a tiny lane.** A bounded stability dial lives beside the bytes; it is computed deterministically via:
  `a_c := clamp(a_raw, -1+eps_a, +1-eps_a) -> u := atanh(a_c) -> U += w*u ; W += w -> align := tanh( U / max(W, eps_w) ).`
- **Declare the rulebook.** Human bands (e.g., `"A++"` … `"CRITICAL"`) come only from **published cutpoints** in a `manifest_id`.
- **Stamp every exchange.** A one-line continuity anchor proves time and the committed content subset:
  `SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEX`.
- **Overlay, not a fork.** Carried as **headers/metadata**; legacy stacks that ignore them still function exactly as before.

Plain language. Today's requests and responses move bytes fast, but meaning, policy, and continuity are scattered. **SSM-NET makes meaning portable and provable on the wire**, so any independent party can later **replay what was said, under which policy, and when**—without private trust channels.

0B. What SSM-NET fixes (recurring internet gaps)

1. **Contextless payloads.** A response is "200 + bytes," but rarely states *how close to risk* the content is or *which rulebook* judged it safe.
2. **Policy drift after the fact.** Thresholds and escalation rules are undocumented or mutable; later reviews cannot prove the rulebook that actually applied.
3. **Unverifiable intermediaries.** Caches, mirrors, and relays preserve bytes—but lose provenance. Disputes collapse into screenshots and opinions.
4. **Cross-vendor misalignment.** Different services interpret the same situation differently, with no *canonical, bounded* dial to reconcile posture.

SSM-NET answer. Label-first disclosure (`value + band`), **optional public lane** (`align` when required for parity), **published manifests** (`manifest_id` with boundary-inclusivity text), and **stamped continuity** (`SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEX`) that chains over a **declared canonical subset**.

One-line takeaway: *Make bytes verifiable, posture reproducible, policy replayable, and intermediaries accountable—without changing the payload.*

0C. Core invariants (copy-ready)

- **Invariance (bytes):** `phi((m,a)) = m` guarantees legacy readers see original values exactly.
- **Bounded lane (deterministic):** `a_c := clamp(a_raw, -1+eps_a, +1-eps_a)` → `u := atanh(a_c)` → `U += w*u ; W += w` → `align := tanh( U / max(W, eps_w) ).`
- **Bands from manifests:** `band := cutpoint_map(align, manifest_id)`; boundary inclusivity is explicit in the manifest text.
- **Canonical subset:** A declared list (e.g., `["value","band","manifest_id"(,"align_ascii"?)]`) is what the stamp's `sha256` commits to.
- **Continuity stamp:** `SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEX`; linear chains; repairs append, never edit.
- **Disclosure defaults: label-first** (`value + band` public); numeric `align` is logs-only unless declared public.

0D. What SSM-NET is not

- **Not a new transport.** It does **not** replace IP/TCP/TLS or alter routing.
- **Not identity infrastructure.** Bands describe **content posture**, not people.
- **Not a mutation of payloads.** Original bytes are never rewritten; overlays ride beside them.

One-line takeaway for Section 0

"Keep your bytes. Add a tiny lane. Declare your rulebook. Stamp every exchange."

TABLE OF CONTENTS

SECTION 1 — SCOPE & NON-GOALS

1A. Scope (what this standard covers)

SSM-NET defines a **manifest-first overlay** for network exchanges that adds portable meaning and verifiable continuity **without altering transports**. The standard covers:

- **Headers/metadata overlay.** How to carry `manifest_id`, disclosure intent, a **canonical subset** declaration, a **body hash**, and a **continuity stamp** `SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEX`.
- **Deterministic lane math.** A bounded dial beside bytes using `a_c := clamp(a_raw, -1+eps_a, +1-eps_a) → u := atanh(a_c) → U += w*u ; W += w → align := tanh( U / max(W, eps_w) )`.
- **Human bands from rulebooks.** `band := cutpoint_map(align, manifest_id)` with **explicit boundary inclusivity**.
- **Canonical subset commitment.** A sender-declared list (e.g., `["value","band","manifest_id"(,"align_ascii"?)]`) that the stamp's `sha256` binds.
- **Verification behavior.** Receiver checks **continuity** (`prev`), **subset integrity**, and **band consistency** under the manifest; optional **local lane recompute** for parity.
- **Evidence publication. Well-known discovery** for manifests, checkpoints, and evidence bundles; optional **fetch** of manifests and checkpoints by URL.
- **Federation levels.** Interop across differing policies via **disclosure levels** (label-first → lanes reproducible → full evidence).
- **Profiles (bindings).** Naming and requirements for concrete transports (e.g., **HTTP-M** for the web), plus guidance for **APIs**, **streams**, **peer/mesh**, and **device** surfaces.

1B. Non-Goals (what this standard does not do)

- **No transport replacement.** SSM-NET rides **above** transport; it does not alter routing, congestion control, handshake semantics, or encryption of transport protocols (e.g., IP, TCP, TLS, UDP-based secure transports).
- **No identity or PII mandate.** Bands describe content posture, not people; identity is out of scope.
- **No payload mutation.** `phi((m,a)) = m` ensures original bytes remain untouched; overlays ride beside them.
- **No content moderation or policy creation.** The manifest freezes your thresholds and obligations; SSM-NET does not author them.
- **No consensus ledger.** Continuity is linear and append-only for each scope; distributed consensus and global ledgers are out of scope.
- **No encryption redesign.** Works over existing secure or insecure channels; cryptographic algorithms and key management are not specified.
- **No schema imposition.** SSM-NET commits to a declared subset; it does not dictate business fields or domain schemas.
- **No availability SLA.** Operational uptime, scaling, and failover are deployment concerns, not protocol guarantees.

1C. Interoperability and forward-compatibility

- **Overlay-safe by default.** Unknown SSM-NET headers or fields may be ignored by legacy stacks without breaking payload delivery.
- **Independent replay.** Any conforming receiver can recompute `sha256` over the **canonical subset** and inspect `prev` continuity without trusting the sender.
- **Manifest rotation, not edits.** Policy changes **mint a new `manifest_id`**; past outcomes stand.
- **Additive evolution.** New headers and fields may be added without invalidating older exchanges, provided the **canonical subset contract** is honored.
- **Text normalization for hashing.** Canonical subset strings **MUST be UTF-8 NFC** before hashing; numeric comparisons **MUST apply defined tolerances** at band boundaries.

1D. Conformance roles (high-level obligations)

- **Senders (MUST):**
  - Publish or reference a `manifest_id`.
  - Declare a **canonical subset** and emit `SSMCLOCK1|...|sha256=HEX|prev=HEX`.
  - Ensure `phi((m,a)) = m` (no payload mutation).
  - Normalize canonical subset fields to **UTF-8 NFC** before hashing; include the **declared order** and exact byte representation in the commitment.

- **Receivers (MUST):**
  - Verify **stamp continuity** and **subset integrity**; detect mismatches deterministically.
  - Map `align` to `band` under the referenced manifest (or **recompute `align` locally** if disclosed) using the exact pipeline `clamp → atanh → accumulate → tanh`.
- **Intermediaries (SHOULD):**
  - Preserve bytes and `SSMNET-Body-Hash`.
  - If adding their own observations, **append a new stamp**; **never rewrite history**.
- **Publishers (SHOULD):**
  - Expose **well-known endpoints** for manifests, checkpoints, and evidence packs.
  - Provide **boundary-inclusivity** text and **escalation promises** inside manifests.

1E. Deployment surfaces (illustrative)

- **Web responses and requests (HTTP-M).** Add headers while keeping bodies **byte-identical**.
- **Programmatic APIs.** REST/GraphQL-style responses carry the same overlay fields.
- **Streaming & messaging.** Frames or messages include envelopes with **canonical subset binding**.
- **Peer/mesh links.** Nodes exchange stamped envelopes with **local policy replay**.
- **Devices and gateways.** Lightweight headers or side-band metadata report **bands and stamps** alongside sensor bytes.

One-line takeaway for Section 1

"Overlay semantics only: carry rulebooks, lanes, and stamps beside unchanged bytes — verifiable by anyone, on any transport."

SECTION 2 — OBJECT MODEL

2A. Alignment lane (normative kernel)

A **bounded, replayable dial** lives beside the original bytes. It never mutates the payload (**collapse parity** `phi((m,a)) = m`) and is **order-invariant** across batch, stream, or shard-merge.

- **Safety clamp:** `a_c := clamp(a_raw, -1+eps_a, +1-eps_a)`
- **Rapidity space:** `u := atanh(a_c)`
- **Accumulate evidence:** `U += w*u ; W += w`
- **Fuse back to lane:** `align := tanh( U / max(W, eps_w) )`

Norms

- **MUST** implement the four-step pipeline above, with **clamp before `atanh`**.
- **MUST** keep `eps_a > 0`, `eps_w > 0` (small positive safeguards).
- **SHOULD** persist `(U, W)` per **scope** so the dial is continuous across messages.
- **MAY** set `w := 1` if no weighting policy is declared.
- **MAY** define **epoch rollover** for `(U,W)` (e.g., by time or count); rollover points **SHOULD** be continuity-stamped and documented in the manifest.

Determinism & invariants (clarifications)

- **Order-invariance:** The fused `align` **MUST** be identical (within numerical tolerance) whether inputs are processed as a batch, as a stream, or via shard-merge using `U_total := ΣU_i`, `W_total := ΣW_i`, then `align_total := tanh( U_total / max(W_total, eps_w) )`.
- **Shard/merge rule: MUST** compute per-shard `(U_i, W_i)` using the same `eps_a`, `eps_w`, `w`, then **fold by summation only** (no re-clamping at joins).
- **Payload invariance: MUST NOT** alter original bytes; the lane is beside the payload, never inside it (`phi((m,a)) = m`).

Recommended numeric practice (non-normative but strongly encouraged)

- **Float mode:** 64-bit floating point for reproducibility; embedded targets may use fixed-point with documented Q-format.
- **Tolerances:** target parity of `≤ 1e-6` for batch vs. stream and `≤ 1e-12` for shard-merge on identical inputs and parameters.
- **Canonical decimal (if public):** when exposing `align` publicly, also emit `align_ascii` as a fixed-width decimal (e.g., `"+0.732000"`) to avoid serializer drift.

Weighting guidance

- **Equal weights:** `w := 1` is acceptable where no trust, sample size, or importance weighting is defined.
- **Declared policy:** if a non-trivial `w` is used, the **manifest MUST** document how `w` is chosen; implementations **MUST** apply the same rule during parity checks.

Prohibited behaviors

- **MUST NOT** apply clamp **after** `atanh`.
- **MUST NOT** re-scale, re-center, or remap `align` outside the specified pipeline.
- **MUST NOT** change `eps_a`, `eps_w`, or `w` within a scope **without** stamping a policy/manifest change.

2B. Bands from a published manifest

Humans see **labels**, not raw dials. Bands are computed **only under a declared rulebook**.

- **Mapping:** `band := cutpoint_map(align, manifest_id)`
- **Boundary text:** the manifest **MUST** declare **inclusivity** at each cut (e.g., `[-1.00, -0.80) → "A++"`).
- **Rotation:** any policy change **MUST** mint a new `manifest_id`; past outcomes stand.
- **Tolerance:** manifests **SHOULD** specify numeric tolerance for boundary comparisons to ensure consistent decisions near cuts.

2C. Envelope (portable unit of meaning)

The minimal **self-describing** unit that can cross any transport.

```
{
  "value":        <opaque or structured>,
  "align":        <FloatAlign>,              // omit unless
align_public=true
  "align_ascii":  "+0.732000",               // include only if align is
public
  "band":         "<BandLabel>",
  "manifest_id":  "<ManifestID>",
  "stamp":        "SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEX"
}
```

Norms

- **MUST** carry `value`, `band`, `manifest_id`, `stamp`.
- `align` **MUST** be omitted unless disclosure is declared **public**.
- When `align` is present, `align_ascii` **SHOULD** be present if a canonical decimal is required.
- Envelopes **MAY** carry additional non-normative fields, but only the **declared canonical subset** is bound by the digest.

2D. Canonical subset & body commitment

A sender declares **exactly which fields are bound** by the stamp's digest and **commits to a byte-exact serialization** of those fields (and, when applicable, the **raw body bytes**).

- **Subset list (declared on the wire):**
  `["value","band","manifest_id"(,"align_ascii"?)]`
  The **order in this list is normative** and defines the **serialization order**.

- **Digest target:** `sha256( serialize(subset_fields) [+ raw_body_bytes_if_declared] )`
  `raw_body_bytes_if_declared` are the **exact bytes as transmitted on the transport**, **after any content encoding** that the wire actually carries.

Serialization rules (normative)

- **Text encoding:** all textual subset fields **MUST** be UTF-8 in NFC normalization; **no BOM**.
- **Newlines:** newline is `\n` (LF). Before hashing, **normalize all line endings to `\n`**.
- **Field order: exactly** the order named in the declared subset list.
- **No extras: MUST NOT** include fields not listed; **MUST NOT** omit listed fields.
- **Whitespace & formatting (inside the subset):** spacing, quoting, and separators **MUST** be stable per the profile's canonical form.
- **Numbers:** values committed as text **MUST** be deterministic (e.g., `align_ascii` uses **fixed sign** and **fixed decimals** such as `+0.732000`); **no locale separators**.
- **Escaping:** strings use a **single, profile-defined escape scheme**; no alternate equivalent encodings.
- **Binary fields:** when a subset field is binary, it **MUST** be committed **by bytes** (not by a re-encoded textual representation), using the profile's canonical serialization.
- **Line-trim option (if enabled):** `TRIM_WHITESPACE_PER_LINE` = **for text fields only**, after normalizing newlines to `\n`, **remove leading and trailing ASCII space (0x20) and tab (0x09) on each line**; **do not alter internal spaces**; **do not apply to binary fields**. Default is **disabled** unless explicitly declared by the manifest/profile.

Body commitment (when a body exists)

- Compute `sha256=<HEX>` over **exact raw body bytes** as sent on the wire (**e.g., if compressed on transport, hash the compressed bytes**).
- **Expose that commitment** alongside the subset declaration so receivers can recompute over the **same bytes**.

Norms

- **Senders MUST** expose the subset declaration and keep its **serialization stable**.
- **Receivers MUST** recompute `sha256` over the **same byte sequence** and compare for **equality (no tolerance)**.
- **Pretty-printing or whitespace changes outside the subset MUST NOT** affect verification.
- **Hidden/missing/misordered fields in the subset MUST** be treated as **verification failure**.

2E. Continuity stamp (sequence & time)

A compact, human-checkable line anchors time and order for a declared scope.

- **Format.** `SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEX`

    - `UTC_ISO` uses `YYYY-MM-DDThh:mm:ss.sssZ` (UTC; no timezone offsets).
    - `nonce` is a CSPRNG-derived token; its length is exactly the manifest's `nonce_bytes` **in bytes**; representation is ASCII as defined by the active profile (e.g., hex or base64url, no padding).
    - `sha256=HEX` commits to the canonical subset (and body if declared).
    - `prev=HEX` links to the immediately prior accepted digest within the same scope.
- **Scope & genesis.** The chain scope is the tuple the deployment declares (e.g., channel/end-point plus `manifest_id`). The first item in any scope uses `prev=NONE`.
- **HEAD.** The **HEAD** is the last accepted digest in a scope. It **MAY** be published to aid audits and fast catch-up.
- **Append-only repairs.** History is immutable. Corrections are new stamped notes appended after the fact; prior lines **MUST NOT** be edited or removed.
- **Clock tolerance.** Receivers **MUST** tolerate timestamp drift within `allowed_skew_s` (declared by policy). Outside that window, treat the declaration as **non-evidential** (payload remains byte-true; stamp is not accepted into continuity).
- **Nonce quality.** Nonces **MUST** be generated by a cryptographically secure PRNG. Reuse across the same scope is prohibited.
- **Rollover markers.** When epoch rollover is used for `(U, W)`, rollover boundaries **SHOULD** be indicated by a stamped note in the same scope.

2F. Disclosure modes (privacy-first defaults)

- **value-only:** Only bytes; lane and band not exposed.
- **value+band (label-first, default):** Human label is public; `align` kept private/logs-only.
- **full:** `value`, `band`, and `align` (plus `align_ascii`) are public.

Norms

- If `align` is public, receivers **SHOULD** recompute locally to parity-check the `band`.
- If `align` is private, receivers **MUST** still be able to verify **subset digest** and **stamp continuity**.

2G. Intermediaries (mirrors, caches, relays)

- **MUST** forward bytes **untouched**.
- **SHOULD** preserve the sender's `sha256` commitments.
- **MAY** append their own **stamped observation** (new `stamp`) without altering the original envelope.
- **MUST NOT** rewrite history or alter any previously committed subset.

2H. Epochs & Scope Lifecycle (normative)

(Purpose: specify how long-running scopes manage `(U,W)` continuity, when and how epochs reset, and how receivers verify parity without ambiguity.)

Purpose & context
Long-running streams can accumulate very large `(U,W)` values. SSM-NET permits **epoch boundaries** that reset the accumulator **without** losing continuity of stamped truth. Epochs are declared facts, not silent math edits.

Definitions

- **Scope:** the continuity domain for stamps and digest replay (e.g., a URL, stream id, topic, or device).
- **Epoch:** a declared boundary within a scope at which `(U,W)` **may** reset to zero for subsequent items.
- **Epoch id (`epoch_id`)**: stable, human-readable token for the current epoch; monotonic within a scope.

Kernel reminder: `a_c := clamp(a_raw, -1+eps_a, +1-eps_a) -> u := atanh(a_c) -> U += w*u ; W += w -> align := tanh( U / max(W, eps_w) ).`

Norms (MUST / SHOULD / MAY)

1. **Declaration on the wire (MUST).**
   - When an epoch starts, the sender **MUST** stamp the first post-reset item and declare `epoch_id` in the envelope or profile header for that item and all subsequent items in the epoch.
   - The continuity **stamp** still uses the same scope; `prev` links across the epoch boundary (append-only).
2. **Reset semantics (MUST).**
   - At an epoch boundary, the sender **MUST** reset accumulators: `U := 0`, `W := 0` for the **next** item; pre-boundary `(U,W)` are not reused.
   - There is **no** re-clamp at joins; the kernel stays exactly: clamp → atanh → accumulate → tanh.
3. **Manifest stability (MUST).**
   - Epoch creation **MUST NOT** change `eps_a`, `eps_w`, or `w`. Any change to these parameters requires a **new `manifest_id`** (rotation), not merely a new epoch.
4. **Receiver parity (SHOULD).**
   - Receivers **SHOULD** recompute parity per epoch independently (L2/L3), expecting continuity of stamps across the boundary and continuity of **truth**, not of `(U,W)`.

5. **Epoch id monotonicity (SHOULD).**
    - `epoch_id` **SHOULD** be strictly increasing (lexical or numeric) within a scope to simplify audit queries and partial replays.
6. **Gap handling (MAY).**
    - If the sender experiences outages or backfills, it **MAY** start a new epoch and declare it; the `prev` chain **MUST** remain linear.

Serialization & discovery

- **Envelope/header field (profile-level):** Include a single key/value (e.g., `SSMNET-Epoch: <epoch_id>`) on every item of that epoch.
- **Checkpoint (optional):** `/.well-known/ssmnet/checkpoint` **MAY** advertise `HEAD` plus `epoch_id` of the latest accepted item to speed resyncs.
- **Evidence bundle (optional): MAY** add `epochs.txt` listing `epoch_id`, first index/time, and last digest seen.

Examples (illustrative)

- *Epoch rollover due to size/time:*
    - Last item of `epoch_id=2025W44` stamps `prev=<HEX_A>` and yields digest `<HEX_B>`.
    - First item of `epoch_id=2025W45` stamps `prev=<HEX_B>`, resets `(U,W)` to zero, and proceeds.
    - Digest continuity proves append-only history; math parity within each epoch proves kernel determinism.
- *Operational rotation:*
    - Nightly rotation `epoch_id=YYYYMMDD` for very high-rate streams; manifests unchanged; disclosure unchanged.

Prohibited behaviors

- **MUST NOT** silently reset `(U,W)` without declaring a new `epoch_id`.
- **MUST NOT** change `eps_a`, `eps_w`, or `w` at an epoch boundary; that is a policy change → **new `manifest_id`**.
- **MUST NOT** fork the `prev` chain at an epoch boundary; continuity remains linear.

Audit expectations

- Auditors can:
    1. Walk stamps across epochs (linear `prev`).

2. Verify per-epoch lane parity with tolerances (`<= 1e-6` batch vs stream; `<= 1e-12` shard-merge).
3. Confirm that kernel parameters match the manifest before and after the epoch boundary.

Minimal receiver algorithm (epoch-aware, L2/L3)

```
state: HEAD := NONE, epoch_active := null
for each item in order:
  verify subset digest -> DIGEST
  require prev == HEAD or prev == "NONE" if genesis
  HEAD := DIGEST

  if epoch_id changes:
     reset local U := 0 ; W := 0

  if align is public:
     recompute a_c := clamp(a_raw, -1+eps_a, +1-eps_a)
     u := atanh(a_c)
     U += w*u ; W += w
     align_local := tanh( U / max(W, eps_w) )
     confirm band := cutpoint_map(align_local, manifest_id)
```

One-line takeaway for Section 2

"A small, deterministic lane + manifest-based bands + a declared subset + a one-line stamp = meaning that travels and can be replayed by anyone."

SECTION 3 — MANIFESTS (THE RULEBOOKS)

3A. Purpose of a manifest

A **manifest** is the frozen rulebook that defines how `align` is interpreted as a **human band**. It states what "safe," "normal," and "critical" mean **at the moment the exchange occurred**.

A manifest answers:

- **Which cutpoints** define each band?
- **Which escalation obligations** apply to each band?
- **What tolerances and numerical safety margins** are in force?
- **Under what assumptions or operating conditions** was this rulebook valid?

Once declared, a manifest is **never edited in place**. If the meaning of safety or response changes, a **new `manifest_id` is minted**. This prevents silent rewriting of expectations after the fact.

3B. Manifest fields (minimum required)

A manifest **MUST** include the following:

Field	Definition
`manifest_id`	A short, stable identifier for this rulebook. Used in envelopes.
`bands`	A list of human-readable labels and their dial intervals, with **boundary inclusivity** text.
`cutpoints`	The numerical boundaries in terms of `align` that map to each band.
`eps_a`, `eps_w`	The small positive constants for the clamp and accumulator safeguards.
`weight_rule`	Optional text specifying how `w` is chosen (e.g., equal, sample-based, trust-based).
`disclosure`	Whether `align` is **public** or **logs-only** under this manifest.
`assumptions`	Optional brief note on operating environment, calibration, or expected input behavior.
`cmp_tolerance`	Optional numeric tolerance guidance for boundary comparisons near cutpoints.
`text_norm`	Optional declaration of string normalization for hashing (e.g., `"utf8_nfc"`).
`epoch_policy`	Optional guidance on `(U,W)` rollover (by time or count) and how rollovers are stamped.
`evidence_links`	Optional well-known paths for checkpoints and evidence bundles relevant to this rulebook.

Example style (illustrative only):

```
manifest_id: "TRANSPORT_POSTURE.DEMO"
bands:
  - "A++" : [-1.00, -0.80]
  - "A0"  : (-0.80, +0.60]
  - "CRITICAL" : (+0.60, +1.00]
boundary_inclusivity:
  A++: left-inclusive, right-inclusive
  A0: left-open, right-inclusive
eps_a: 1e-6
eps_w: 1e-9
weight_rule: equal
disclosure: value+band
cmp_tolerance: 1e-9
text_norm: "utf8_nfc"
epoch_policy:
  mode: "count"
  rollover_after: 5000
  note: "Emit stamped rollover marker at each boundary."
```

```
assumptions: "Nominal input sampling rate; no guaranteed rate enforcement."
evidence_links:
  checkpoints: "/.well-known/ssmnet/checkpoint/TRANSPORT_POSTURE.DEMO"
  bundles: "/.well-known/ssmnet/evidence/TRANSPORT_POSTURE.DEMO"
```

3C. Rotation principle (critical invariant)

Rule: If the meaning of any band, threshold, or action stance changes, the `manifest_id` **MUST change**.

This means:

- You **do not rewrite meanings in place**.
- Historical records **retain the meaning** that applied at the moment they were produced.
- Audits and post-event reviews become **replayable and fair**.

Silent policy drift is impossible under this rule.

3D. Publication & discovery

Purpose. Make the rulebook **fetchable**, **immutable per `manifest_id`**, and **cache-friendly** so any independent verifier can replay bands exactly as declared without private coordination.

Well-known location (normative).
Publish each manifest at a stable path:
`/.well-known/ssmnet/manifest/<manifest_id>`

Receiver capabilities (MUST).

- Retrieve the **exact manifest text** used at emission time.
- Confirm **boundary inclusivity** at every cut (e.g., `(-0.80, +0.60]`).
- Confirm **disclosure mode** (e.g., `value+band` by default; `full` only if declared).
- Confirm any **assumptions/tolerances**, including `eps_a`, `eps_w`, any `weight_rule`, `cmp_tolerance`, and `text_norm`.
- Detect **epoch policy** notes when `(U,W)` rollover is in use.

Immutability (MUST).

- A published manifest **MUST NOT** change for a given `manifest_id`.
- Any policy change **MUST** mint a **new `manifest_id`**; **old manifests remain online** for replay.

Transport-agnostic representation (SHOULD).

- Use a **plain, canonical text** format with stable field names and decimal precision matching cut boundaries.
- Include **explicit boundary text** (e.g., "left-open, right-inclusive") alongside numeric ranges to avoid locale/serializer ambiguity.
- Declare charset; manifests **SHOULD** be UTF-8 and, when hashing is referenced, **NFC-normalized**.

Caching & integrity (SHOULD).

- Serve with strong caching metadata (e.g., long TTL plus validators).
- Provide a **content digest line** within the manifest text (e.g., `sha256=<HEX>` computed over the **byte-exact** manifest body) so verifiers can pin the exact bytes they replayed.

Minimal response expectations (MUST/SHOULD).

- **MUST** return the manifest body as **byte-stable** text.
- **SHOULD** use a deterministic charset (UTF-8) and **declare it**.
- **MUST** avoid embedding secrets or identity data; manifests describe **content posture**, not people.

Discovery notes (MAY).

- **MAY** expose an index at:
  `/.well-known/ssmnet/manifest/`
  listing available `manifest_id` entries and their digests.
- **MAY** link (non-normative) to related well-known endpoints for **checkpoint** and **evidence** to help operators find replay artifacts quickly.

One-line takeaway for 3D.

"Publish once, never edit: a stable, cacheable manifest per `manifest_id` so every verifier can fetch the same rulebook and reproduce the same bands."

3E. Reproducibility of bands

To independently verify that the **band** is correct:

1. Receiver recomputes `align` locally (if `align` is public or reconstructable).
2. Receiver applies `cutpoint_map(align, manifest_id)` using the **exact manifest text**.
3. Receiver confirms the declared **band** matches.

This is **vendor-neutral by design**.

3F. Manifest changes over time

To evolve policy:

1. Draft new thresholds or obligations.
2. Publish a **new manifest** with a **new `manifest_id`**.
3. Begin using the new `manifest_id` in envelopes.
4. **Never change** the old manifest text.
5. Do **not** rewrite past envelopes.

Past envelopes remain valid under the rulebook that existed at their time.
This is how the network maintains **meaning continuity** without needing trust in memory.

One-line takeaway for Section 3

"A manifest freezes the meaning of safety in time; if meaning changes, the `manifest_id` changes — history remains true."

SECTION 4 — WIRE SEMANTICS (PROFILE-AGNOSTIC)

4A. Sender declarations (outbound)

A conforming sender **attaches meaning beside unchanged bytes** by declaring:

- **Manifest reference:** include `manifest_id`.
- **Disclosure intent:** choose one of `value-only`, `value+band` (default), or `full`.
- **Canonical subset:** declare exactly which fields are bound by the digest (e.g., `["value","band","manifest_id"(,"align_ascii"?)]`).
- **Body commitment (if applicable):** compute `sha256=<HEX>` over the raw body bytes when a body exists.
- **Continuity stamp:** emit `SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEX`.

Norms

- **MUST** ensure `phi((m,a)) = m` (payload bytes never altered by the overlay).
- **MUST** serialize the canonical subset **byte-exactly** for digest parity.
- **SHOULD** keep `align` private unless `full` disclosure is explicitly chosen.

4B. Receiver validation (inbound)

A conforming receiver **verifies first, then trusts**. The goal is to accept only what can be replayed independently, without changing a single payload byte.

- **Subset integrity.** Recompute `sha256` over the declared canonical subset (and raw body bytes if declared) and compare to the advertised digest. If the **subset declaration** is missing or malformed, treat as a validation failure.
- **Continuity.** Verify that `prev` links to the last accepted item in the same scope; on success, update local `HEAD`. If `prev` does not chain, record a stamped incident and do not advance `HEAD`.
- **Band consistency.** Apply `band := cutpoint_map(align, manifest_id)` using the published manifest.
    - If **align is public**, recompute the lane via `a_c := clamp(a_raw, -1+eps_a, +1-eps_a) → u := atanh(a_c) → U += w*u ; W += w → align := tanh( U / max(W, eps_w) )` and confirm the band matches the manifest cutpoints (including boundary inclusivity).
    - If **align is not public** (label-first), validate the declared **band** deterministically from disclosed inputs and the manifest; store the manifest text and subset proof with the record.
- **Payload invariance.** Confirm that overlay fields do **not** mutate the body: `phi((m,a)) = m` must hold for accepted items.

Outcomes

- **Accept.** Digest parity OK, `prev` continuity OK, and band mapping consistent with the manifest. Update `HEAD`.
- **Flag.** If any check fails, **quarantine** the item and append a **stamped incident note** (repairs are append-only). Do not edit history or advance `HEAD` on the failing branch.

4C. Canonical subset & digest contract

The digest is only as good as the exact bytes it binds.
• **Declared list (example).** `["value","band","manifest_id"(,"align_ascii"?)]`
• **Digest target.** `sha256( serialize(subset_fields) [+ raw_body_bytes_if_declared] )`
• **Serializer rules (normative).** Field names **and** order in the declared list MUST be the order serialized; bytes bound by the digest MUST be exactly the on-the-wire bytes of those fields (no reformatting, no hidden fields). If `align_ascii` is present, it MUST be canonical decimal (e.g., `+0.732000`) with a fixed precision declared in the manifest/profile and identical everywhere it appears.
• **Newlines (normative).** Newline is `\n` (LF). Profiles that carry textual subset fields MUST serialize with LF only.

• **Per-line trim option (normative when enabled).** If the manifest/profile enables `TRIM_WHITESPACE_PER_LINE`, then **before** serialization of textual subset fields, trim leading/trailing **space (U+0020)** and **tab (U+0009)** from each line; do **not** alter interior whitespace; do **not** change newlines. If not enabled, no trimming is performed.
• **Stability rule.** Changes to whitespace, pretty-printing, or formatting **outside** the declared subset MUST NOT affect verification.
• **Transparency rule.** The subset declaration MUST be visible to receivers on the wire.
• **Intermediaries.** Intermediaries MUST NOT alter bytes of fields listed in the declared subset. If they add their own metadata, it MUST NOT be included in the upstream subset bound by the original digest.
• **Body commitment (if declared).** `raw_body_bytes_if_declared` are the exact bytes as sent on the wire; if transport applies encoding (e.g., compression), hash the encoded bytes.
• **Failure examples (illustrative).** Re-serializing fields in a different key order; changing numeric formatting of `align_ascii`; hashing decoded text instead of the raw body bytes when a body commitment is declared; enabling `TRIM_WHITESPACE_PER_LINE` but trimming characters other than space/tab; emitting CRLFs instead of LF.

4D. Continuity scopes, genesis, and repairs

Continuity is **linear per scope** (e.g., per resource, stream, or conversation).

- **Genesis.** First item uses `prev=NONE`.
- **HEAD.** Last accepted `sha256` for the scope; MAY be published for audits.
- **Acceptance.** A new item is accepted only if its `prev` equals the current HEAD for the applicable chain (see **4D.1**).
- **Repairs (append-only).** If a defect is found, emit a **new stamped note** that references the last valid HEAD; prior history is never edited.
- **Rollover markers.** When epoch rollover is used for `(U,W)`, rollover boundaries SHOULD be indicated by a stamped note in the same scope.
- **Failure.** Items that cannot chain to the current HEAD for their chain MUST be rejected as non-continuous (see fork procedures in **4D.1**).

4D.1 Branching & Chain IDs (normative)

To support deliberate forks without rewriting history, continuity is tracked **per `chain_id`** within a scope.

- **Field.** `chain_id` is an **opaque, binary-safe identifier** carried alongside the stamp. Recommended size is **16 bytes** rendered as fixed-length lowercase hex (or profile-declared encoding).
- **Default.** If no `chain_id` is present, receivers MUST treat it as `chain_id="default"`.
- **Validation rule.** For an incoming item, `prev` MUST equal the current **HEAD of the same `chain_id`**. Otherwise the item is an **orphan** and MUST be rejected unless a fork procedure (below) is in effect.

- **Fork procedure (creating a new line).**
  - Start a **new `chain_id`**.
  - Include a **fork note** that states: the **parent `chain_id`**, the **parent HEAD** at the moment of divergence, and a short **reason code**.
  - The new chain proceeds linearly from its own genesis (its first item uses `prev=NONE`).
- **Selection policy.** Receivers MAY track multiple chains per scope. Which chain is **active** for enforcement is a **local policy decision** (e.g., prefer a configured `chain_id`, or the latest HEAD of an approved set).
- **Merges.** History is never rewritten. If two chains must converge, emit a **bridge note** on the chosen chain that references the other chain's HEAD in a metadata field (informational). The **stamp's `prev` stays single-link**; no multi-parent merges.
- **Publication.** Publishers SHOULD expose a **HEAD map**: `{ chain_id → HEAD }` for audits.
- **Rotation.** When rotating keys/manifests or performing planned maintenance, either **continue** on the same `chain_id` (preferred) or start a **new `chain_id`** with a fork note indicating rotation.
- **Failure handling.** Items with unknown or disallowed `chain_id` MUST be treated as **non-compliant** for that receiver's policy but MAY be stored for forensic review.

4E. Disclosure modes on the wire

- **value-only:** sender exposes bytes only; still stamps subset/body for integrity.
- **value+band (label-first):** sender exposes `band` derived from manifest; `align` remains private/logs-only.
- **full:** sender exposes `value`, `band`, `align` (and optionally `align_ascii`) for public parity checks.

Privacy default: prefer **label-first**; disclose `align` publicly only when necessary for ecosystem parity.

4F. Intermediaries (mirrors, caches, relays)

- **MUST** forward payload bytes unchanged.
- **SHOULD** preserve original `sha256` commitments.
- **MAY** append their **own** stamped observation (new `stamp`) while keeping the prior chain intact.
- **MUST NOT** rewrite earlier stamps or canonical subset declarations.

One-line takeaway for Section 4

"Declare a manifest, bind a precise subset, stamp continuity, and let any receiver re-verify without trusting you—or changing a single payload byte."

SECTION 5 — PROFILES (BINDINGS)

5A. HTTP-M (web profile: SSM-NET over HTTP/HTTPS)

Purpose. Carry SSM-NET semantics as HTTP headers while keeping bodies byte-identical.

Request (client → server) headers

- `SSMNET-Manifest: <manifest_id>`
- `SSMNET-Disclosure: value-only | value+band | full`
- `SSMNET-Canonical-Subset: ["value","band","manifest_id"(,"align_ascii"?)]`
- `SSMNET-Body-Hash: sha256=<HEX>` // over raw body bytes, if present
- `SSMNET-Stamp: SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEX`

Response (server → client) headers

- `SSMNET-Manifest: <manifest_id>`
- `SSMNET-Band: <BandLabel>` // optional (label-first)
- `SSMNET-Align-Ascii: <+0.000000>` // optional; canonical signed decimal with fixed precision if public
- `SSMNET-Canonical-Subset: ["value","band","manifest_id"(,"align_ascii"?)]`
- `SSMNET-Body-Hash: sha256=<HEX>`
- `SSMNET-Stamp: SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEX`
- `SSMNET-Checkpoint: HEAD=<HEX>` // optional (per scope)

Norms

- **MUST** keep payload bytes unchanged (`phi((m,a)) = m`).
- **MUST** compute `sha256` over byte-exact body if `SSMNET-Body-Hash` is present.
- **SHOULD** default to label-first (`value+band`); expose `align_ascii` only if declared public.
- **MAY** publish well-known endpoints for audits (see Section 7).
- **MUST** ensure the canonical subset declaration is visible on the wire.

Minimal example (illustrative)

```
GET /resource
SSMNET-Manifest: NETWORK_POSTURE.DEMO
SSMNET-Disclosure: value+band
SSMNET-Canonical-Subset: ["value","band","manifest_id"]
SSMNET-Stamp: SSMCLOCK1|2025-11-09T12:05:00Z|n1|sha256=...|prev=NONE

200 OK
SSMNET-Manifest: NETWORK_POSTURE.DEMO
SSMNET-Band: A0
SSMNET-Canonical-Subset: ["value","band","manifest_id"]
SSMNET-Body-Hash: sha256=...
SSMNET-Stamp: SSMCLOCK1|2025-11-09T12:05:01Z|n2|sha256=...|prev=...
```

5B. WS-M (bidirectional streams)

Purpose. Carry envelopes and stamps in message metadata for duplex streams.

Binding

- Each message **MUST** include an envelope with `value`, `band`, `manifest_id`, `stamp`.
- Stream scope defines continuity; first message uses `prev=NONE`.
- Intermediaries **MAY** append their own stamped observations without altering prior frames.

Disclosure

- Default **label-first**; `align` only if declared public.
- When `align` is public, receivers **SHOULD** recompute locally to confirm `band`.

5C. API-M (programmatic APIs)

Purpose. Apply SSM-NET semantics to programmatic API requests and responses without altering transport semantics. SSM-NET rides as headers or API metadata while keeping payload bytes unchanged (`phi((m,a)) = m`).

Binding

- **Envelope placement.** Declarations MAY appear in protocol headers or a dedicated metadata block; they MUST be visible to receivers on the wire.
- **Body commitment.** If a body exists, `SSMNET-Body-Hash: sha256=<HEX>` MUST bind the exact raw body bytes as sent.
- **Canonical subset.** The declared list (e.g., `["value","band","manifest_id"(,"align_ascii"?)]`) MUST be present and the field order MUST define serialization order for digest computation.

- **Alignment format.** If `align_ascii` is disclosed, it MUST use canonical decimal with fixed precision (e.g., `+0.732000`).
- **Continuity stamp.** `SSMNET-Stamp: SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEX` anchors sequence and time.
- **Non-interference.** Intermediaries MUST NOT alter bytes of fields listed in the declared subset; added metadata MUST NOT be included in upstream commitments unless explicitly re-declared.
- **Error handling.** If any committed bytes or order differ, receivers MUST fail verification.

Illustrative mappings

HTTP-style JSON API

```
POST /api/do-thing
SSMNET-Manifest: NETWORK_POSTURE.DEMO
SSMNET-Disclosure: value+band
SSMNET-Canonical-Subset: ["value","band","manifest_id"]
SSMNET-Body-Hash: sha256=...
SSMNET-Stamp: SSMCLOCK1|2025-11-09T07:05:00Z|nA1|sha256=...|prev=NONE

{ "value": { "op": "do-thing", "x": 12 } }
```

Server response:

```
200 OK
SSMNET-Manifest: NETWORK_POSTURE.DEMO
SSMNET-Band: A0
SSMNET-Canonical-Subset: ["value","band","manifest_id"]
SSMNET-Body-Hash: sha256=...
SSMNET-Stamp: SSMCLOCK1|2025-11-09T07:05:01Z|nA2|sha256=...|prev=...

{ "value": { "status": "ok" } }
```

gRPC-style metadata (pseudo)

```
:authority: api.example
method: DoThing
metadata:
  SSMNET-Manifest: NETWORK_POSTURE.DEMO
  SSMNET-Disclosure: full
  SSMNET-Canonical-Subset: ["value","band","manifest_id","align_ascii"]
  SSMNET-Body-Hash: sha256=...
  SSMNET-Stamp: SSMCLOCK1|2025-11-09T07:06:00Z|nB1|sha256=...|prev=NONE
payload-bytes: <exact binary protobuf body>
```

Receiver recomputes `sha256` over the exact protobuf bytes as transmitted.

JSON-RPC sidecar fields (example)

```
{
  "jsonrpc": "2.0",
  "method": "doThing",
  "params": { "x": 12 },
  "id": 1,
```

```
  "_ssmnet": {
    "manifest": "NETWORK_POSTURE.DEMO",
    "disclosure": "value+band",
    "canonical_subset": ["value","band","manifest_id"],
    "body_hash": "sha256=...",
    "stamp": "SSMCLOCK1|2025-11-09T07:07:00Z|nC1|sha256=...|prev=NONE"
  }
}
```

Norms

- **MUST** keep payload bytes unchanged (`phi((m,a)) = m`).
- **MUST** ensure the canonical subset declaration is visible and stable.
- **MUST** recompute and compare `sha256` exactly (no tolerance).
- **SHOULD** default to label-first disclosure (`value+band`); disclose `align_ascii` only when declared public.
- **MAY** publish well-known discovery endpoints for manifest shelves and checkpoints.

5D. MESH-M (peer/mesh links)

Purpose. Peers exchange stamped envelopes over arbitrary links.

Binding

- Peers **MUST** preserve chains; repairs append-only.
- Federation level is negotiated per link (label-first → full evidence).
- Checkpoint gossip **MAY** publish `HEAD=<HEX>` for recovery.

5E. IOT-M (devices & gateways)

Purpose. Constrained devices carry bands and stamps with minimal overhead.

Binding
• Compact headers or side-band key–value pairs carry `manifest_id`, `band`, and `stamp`.
• Gateways **SHOULD** preserve device stamps and append their own observation as a new stamped note (append-only; no rewrites).
• `eps_a`, `eps_w`, and `weight_rule` **MUST** be documented in the referenced manifest for reproducibility.
• When `align` is disclosed, devices **MUST** emit canonical decimal text (e.g., `+0.732000`) and receivers **MUST** treat non-canonical forms as verification failures.
• The continuity line **MUST** follow `SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEX`, where `nonce` is generated via a crypto-secure method and sized in bytes per manifest policy.
• Devices **MUST NOT** alter bytes of any fields declared in the canonical subset; gateways **MUST NOT** reserialize those bytes when forwarding.
• For binary payloads, the body commitment **MUST** be computed over the exact on-the-wire bytes; if the transport compresses, hash the compressed bytes.

• Low-power links may carry the canonical subset declaration once per session; if so, each message **MUST** reference that declaration's identifier, and any change **MUST** re-declare before use.

One-line takeaway for Section 5

"Same semantics, many surfaces: headers for the web, frames for streams, and side-band KVs for devices—all carrying the same manifest, subset, and stamp."

SECTION 6 — GOLDEN FLOWS (END-TO-END EXAMPLES)

6A. Public GET (label-first discovery)

Intent. Serve a byte-identical resource while exposing portable meaning for anyone to replay.

Sender (server) behavior

- Choose **label-first** disclosure (`value+band`).
- Declare canonical subset (e.g., `["value","band","manifest_id"]`).
- Compute `sha256` over the subset (and body bytes if applicable).
- Emit continuity: `SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEX`.

Receiver checks

- Recompute `sha256` over declared subset [+ body if declared].
- Verify `prev` continuity; update `HEAD`.
- Confirm `band := cutpoint_map(align, manifest_id)` is **derivable** from manifest (even if `align` stays private).

Outcome

- Resource bytes unchanged (`phi((m,a)) = m`), plus portable provenance.

6B. Declared POST (create with evidence)

Intent. Sender submits content and **declares** policy posture at creation time.

Sender (client) behavior

- Stamp request body and subset:
  `SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEAD`.
- Include `manifest_id`, disclosure mode, canonical subset.
- If `full`, expose `align` computed from:
  `a_c := clamp(a_raw, -1+eps_a, +1-eps_a)` → `u := atanh(a_c)` → `U += w*u ; W += w` → `align := tanh( U / max(W, eps_w) )`.

Receiver (server) behavior

- Verify digest over subset and raw body.
- Verify chain continuity against server-side `HEAD`.
- Map `band` from manifest; on success, return created resource with its **own** stamp (new `prev`).

Outcome

- Two stamps exist (submission and creation), forming a **bidirectional, replayable** narrative.

6C. Mirror / CDN (intermediary preservation)

Intent. Intermediaries preserve bytes and provenance, optionally adding their **observation**.

Intermediary behavior

- **Forward bytes untouched.**
- Preserve original `sha256` and subset declaration.
- **Optionally** append a fresh stamped note (new `stamp`) attesting to mirror time/location.
- **Never** rewrite earlier stamps or subsets.

Receiver behavior

- Accept multiple stamps; check that each forms a single linear chain per scope (no forks unless documented).

Outcome

- Provenance survives caching and replication without proprietary trust.

6D. Streaming frame (duplex scope)

Intent. Apply the same semantics to message frames.

Sender behavior

- Treat the stream (topic/room/resource) as a **scope**.
- First frame uses `prev=NONE`; each subsequent frame chains `prev=<last_sha256>`.
- Default disclosure remains **label-first**; expose `align` only when necessary.

Receiver behavior

- Maintain `HEAD` per scope; verify parity and band mapping per frame.
- On dropout/rejoin, request latest `HEAD` (or consult checkpoint endpoint).

Outcome

- Frame sequences become replayable event logs with stable posture semantics.

6E. Evidence pack pull (self-service audit)

Intent. Allow any party to fetch exactly the artifacts required to **replay meaning** independently, without private coordination.

Sender behavior
• Publish at well-known paths:
  o `/.well-known/ssmnet/manifest/<manifest_id>`
  o `/.well-known/ssmnet/checkpoint` → `HEAD=<HEX>`
  o `/.well-known/ssmnet/evidence` → bundle containing `envelopes.jsonl`, `manifests.json`, `hashes.txt`, `checkpoint.txt`, and a tiny `verify.sh`.
• Ensure the canonical subset is clearly declared and the bundle's files are byte-for-byte consistent with stamps and subset digests.
• Prefer **label-first** disclosure in public bundles; if `align` is private, include sufficient manifest text so derivability of bands is auditable without revealing `align`.
• (Optional) Support range/time filters `?scope=<name>&from=<iso>&to=<iso>` for large chains; filtered bundles MUST preserve a valid chain from the first included item to `HEAD` or to the filtered end.

Verifier behavior
• Run `verify.sh`:
  o Recompute `sha256` over the declared canonical subset of each envelope (and raw body bytes if declared).
  o Validate the **linear** `prev` chain from `genesis` (`prev=NONE`) to `HEAD`.
  o Confirm `band := cutpoint_map(align, manifest_id)` for any envelope where `align` is public; when `align` is private, confirm that the declared band is **derivable** from the published manifest cutpoints.
• Treat any mismatch as a stamped incident (append-only); never edit prior records.

Outcome
• One command prints `ALL CHECKS PASSED` **or** pinpoints the first failing step with a precise reason (e.g., digest mismatch, chain break, boundary inclusivity error), while keeping payload bytes invariant (`phi((m,a)) = m`).

One-line takeaway for Section 6

"Same bytes, now replayable: GET, POST, mirrors, streams, and audits all share the same subset-digest + stamp chain + manifest-mapped bands."

SECTION 7 — WELL-KNOWN ENDPOINTS (DISCOVERY)

7A. Purpose

Well-known endpoints let any party **discover rulebooks, continuity, and evidence** without private coordination. They are **read-only**, **cache-friendly**, and **profile-agnostic**. All textual responses **SHOULD** be UTF-8 and, when used in hashing, **MUST** be normalized to **UTF-8 NFC** to ensure byte-stable verification.

7B. Endpoints (normative set)

Manifest (by ID)
`GET /.well-known/ssmnet/manifest/<manifest_id>`
• Returns: the exact manifest text used to derive bands, including boundary inclusivity, `eps_a`, `eps_w`, `weight_rule`, optional `cmp_tolerance`, `text_norm`, and any `epoch_policy` notes.
• MUST be immutable once published for a given `manifest_id`.
• SHOULD return a strong validator (e.g., `ETag`) and long cache lifetime.
• MAY include a short assumptions field (operating ranges, hysteresis, sampling notes).
• MUST NOT include identity or PII.
• Content-Type (SHOULD): a clear, text-based format that is byte-stable for hashing (e.g., `application/json` or `text/plain`).
• Notes: if a manifest embeds a self-digest line like `sha256=<HEX>`, it MUST be computed over the byte-exact manifest body.

Checkpoint (scope HEAD)
`GET /.well-known/ssmnet/checkpoint[?scope=<name>]`
• Returns: `HEAD=<HEX>` for the requested scope (or the default scope if omitted).

• Used for: resync, replay start points, and fork detection.
• SHOULD include `updated=<UTC_ISO>` to aid caches and humans.
• SHOULD support light caching (short TTL) and conditional requests.
• MUST remain read-only; publishing HEAD never alters historical stamps.
• MAY include a `rollover=<UTC_ISO>` hint if a `(U,W)` epoch boundary recently occurred.

Evidence bundle (self-service audit)
`GET /.well-known/ssmnet/evidence[?scope=<name>&from=<iso>&to=<iso>]`
• Returns: a compact bundle containing (at minimum):
○ `envelopes.jsonl` (canonical subset per line),
○ `manifests.json` (the referenced manifests),
○ `hashes.txt` (per-item `sha256` over the declared subset),
○ `checkpoint.txt` (`HEAD=<HEX>`),
○ `verify.sh` (tiny script that prints ALL CHECKS PASSED or pinpoints failure).
• MAY support time/range filtering for large scopes (`from`, `to`).
• SHOULD include a top-level `bundle.sha256` for the entire archive.
• MUST avoid PII in canonical subsets; posture, not identity.
• Content-Type (SHOULD): deterministic archive (e.g., `.zip`) with stable ordering to preserve digest reproducibility.

Manifest index (optional)
`GET /.well-known/ssmnet/manifest/`
• MAY list available `manifest_id` entries and their digests for discovery and tooling.

7C. Minimal response styles (illustrative)

Manifest (YAML-like, illustrative only)

```
manifest_id: "TRANSPORT_POSTURE.DEMO"
bands:
  - "A++": [-1.00, -0.80]
  - "A0" : (-0.80, +0.60]
  - "CRITICAL": (+0.60, +1.00]
boundary_inclusivity:
  A++: left-inclusive, right-inclusive
  A0:  left-open, right-inclusive
eps_a: 1e-6
eps_w: 1e-9
weight_rule: equal
disclosure: value+band
cmp_tolerance: 1e-9
text_norm: "utf8_nfc"
assumptions: "Nominal sampling; no guaranteed rate enforcement."
```

Notes (illustrative):

- Bands **MUST** map to explicit dial intervals; edges are defined in `boundary_inclusivity`.

- `eps_a` and `eps_w` are small positive safeguards used in the kernel: `a_c := clamp(a_raw, -1+eps_a, +1-eps_a) -> u := atanh(a_c) -> U += w*u ; W += w -> align := tanh( U / max(W, eps_w) ).`
- `disclosure` sets the public posture: **label-first** (`value+band`) by default; numeric `align` is private unless declared public.

Checkpoint

```
HEAD=1F3C4D...A902
scope=default
updated=2025-11-07T12:30:00Z
```

Notes (illustrative):

- `HEAD` is the last accepted `sha256` in scope; genesis uses `prev=NONE`.
- `scope` names **SHOULD** be stable and human-readable.

Evidence bundle layout

```
/evidence/
  manifests.json
  envelopes.jsonl
  hashes.txt
  checkpoint.txt
  verify.sh
```

Notes (illustrative):

- `hashes.txt` lists `sha256` over the **declared canonical subset** (and body if declared).
- `verify.sh` **SHOULD** recompute digests, walk the `prev` chain, and confirm band mapping under the referenced manifest.

7D. Caching, integrity, and privacy

- **Caching:**
  - `manifest/<manifest_id>` **SHOULD** be cacheable with **long TTL** (immutable).
  - `checkpoint` **MAY** be short-TTL and `ETag`/`Last-Modified` aware.
  - `evidence` **MAY** be range-/time-bounded; encourage compression.
- **Integrity:**
  - Each envelope's digest binds the declared **canonical subset** (e.g., `["value","band","manifest_id"(,"align_ascii"?)]`).
  - `hashes.txt` **MUST** list the `sha256` used in the stamp line `SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEX`.
  - Textual materials used in hashing **MUST** be **UTF-8 NFC** to avoid digest drift.
- **Privacy:**
  - **Default label-first:** publish bands and manifests; keep `align` private unless declared public.

- **Redact PII** from the canonical subset; SSM-NET does not require identity fields.

7E. Versioning and evolution

- **Manifest rotation:** Any policy change **MUST** mint a new `manifest_id`; past manifests remain published.
- **Endpoint stability:** Paths remain stable; new optional query params may be added **without** breaking older clients.
- **Backward compatibility:** Additional fields in `manifests.json` or `envelopes.jsonl` **MUST NOT** alter the meaning of previously defined fields or the **canonical subset** contract.

7F. Operational notes

- **Availability:** Endpoints are **informational**; outages **MUST NOT** block payload delivery.
- **Scope naming:** Choose stable, human-readable scope names (e.g., `default`, `orders`, `feed-alpha`).
- **Fork handling:** If a fork is intentionally created, publish a short `fork.txt` in the evidence bundle describing the procedure and recovery rules.
- **Rollover markers:** When `(U,W)` epoch rollover is used, **SHOULD** expose a stamped checkpoint near the boundary for simpler replay.

7G. Fetch, Pinning, and Offline Replay (client guidance)

Purpose. Define how receivers fetch manifests and checkpoints, **pin** the exact bytes they relied on, and continue verification **offline** without private coordination.

7G.1 Minimal client behavior (normative)

- **Fetch manifest by ID.** `GET /.well-known/ssmnet/manifest/<manifest_id>` → byte-stable text.
- **Compute pin.** `manifest_sha256 := sha256(manifest_bytes)` and cache both **bytes** and **pin**.
- **Verify band locally.** Apply `band := cutpoint_map(align, manifest_id)` using the **fetched bytes** (not a re-rendered variant).

- **Fetch checkpoint.** `GET /.well-known/ssmnet/checkpoint[?scope=<name>]` → `HEAD=<HEX>` (+ optional `updated`).
- **Persist for offline.** Store `manifest_bytes`, `manifest_sha256`, `HEAD`, and the envelopes you receive (declared canonical subset only).

7G.2 Pinning contract (normative)

- **Pin field (client-side).** Store `manifest_pin := "sha256=" + manifest_sha256`.
- **Replay rule.** When verifying any envelope that cites `<manifest_id>`, require that its manifest bytes produce the **same** `manifest_pin`. If not, treat as **E_MANIFEST_PIN** and require fresh fetch or operator review.
- **No in-place edits.** If a server changes the bytes for a given `<manifest_id>`, the pin will change; clients MUST treat this as a violation and prefer the cached older bytes. Proper evolution uses a **new manifest_id** (rotation principle).

7G.3 Offline replay (deterministic)

- **Inputs:** `manifests.json` (byte-exact texts), `envelopes.jsonl` (declared subset only), `hashes.txt`, `checkpoint.txt`.
- **Steps:**
  1. For each envelope line, compute `sha256(subset_bytes [+ body_bytes_if_declared])` and compare with `hashes.txt`.
  2. Rebuild continuity via the stamped `prev` chain: `SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEX_OR_NONE`.
  3. For lanes disclosed publicly, recompute `align` with the kernel: `a_c := clamp(a_raw, -1+eps_a, +1-eps_a) -> u := atanh(a_c) -> U += w*u ; W += w -> align := tanh( U / max(W, eps_w) )`.
  4. Apply the **pinned** manifest text to derive `band` via `cutpoint_map(align, manifest_id)`.
  5. Expect `ALL CHECKS PASSED` when the computed HEAD equals `checkpoint.txt` and all bands match manifest cuts.

7G.4 Caching & freshness (client hints)

- **Manifest cache:** long-TTL; validate by **pin**, not by timestamp.
- **Checkpoint cache:** short-TTL; accept conditional fetches; stale checkpoint only affects **recency**, not validity of prior stamps.
- **Network loss:** if fetch fails, continue using pinned manifest for verification; defer only actions that require a newer rulebook.

7G.5 Failure codes (wire-surface, illustrative)

- **E_MANIFEST_MISS:** fetch returns 404 or empty body.
- **E_MANIFEST_PIN:** bytes fetched do not match the previously stored `manifest_pin`.
- **E_CHECKPOINT_DRIFT:** computed HEAD from verified envelopes != published HEAD.
- **E_SUBSET_DECL:** declared canonical subset malformed or missing required fields.
- **E_BODY_HASH_MISMATCH:** digest over body/subset differs from advertised value.

7G.6 Security & privacy notes

- **No identity drag.** Fetching manifests and checkpoints MUST NOT require identity; they are public rulebooks and continuity summaries.
- **PII discipline.** Canonical subsets SHOULD exclude personal attributes; posture is about signals/systems.
- **Integrity over recency.** A pinned manifest guarantees reproducibility; a stale checkpoint cannot alter the validity of already-verified stamps.

7G.7 Minimal operator checklist (paste-ready)

- **[ ]** Serve manifests at stable paths; byte-stable; include edges and `eps_a`, `eps_w`, optional `weight_rule`.
- **[ ]** Publish checkpoint with `HEAD=<HEX>` and short TTL.
- **[ ]** Keep old manifests online; never edit bytes for an existing `<manifest_id>`.
- **[ ]** Include a one-line `sha256=<HEX>` within the manifest body (optional but recommended) to help humans pin visually.
- **[ ]** Provide evidence bundles for offline replay.

One-line takeaway for Section 7

“Manifests, HEAD, and evidence live at predictable, read-only URLs — so anyone can fetch, replay, and verify without asking for favors.”

SECTION 8 — ERROR MODEL (OVERLAY-SAFE)

8A. Philosophy

Errors **must expose facts without mutating history**. An error in SSM-NET is a *stamped disclosure* that something failed verification; it **does not** rewrite payload bytes or earlier stamps. Repairs are **append-only**.

8B. Signaling errors on the wire

A conforming implementation signals errors via minimal overlay fields (headers or side-band metadata) and an optional compact body.

Required behavior
• **MUST** keep the original payload bytes unchanged (`phi((m,a)) = m`).
• **MUST** include a machine-parsable error code.
• **SHOULD** include a short, human-readable reason (no secrets, no PII).
• **MUST** append a new continuity stamp for the error note:
`SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEAD`.
• **MUST NOT** remove or alter prior stamps, subsets, or manifests.

Recommended header fields (illustrative)
• `SSMNET-Error: <CODE>`
• `SSMNET-Error-Reason: <short-text>`
• `SSMNET-Stamp: SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEX`

Canonical error codes (normative set)
• `commit-mismatch` — recomputed sha256 does not equal declared digest.
• `unknown-manifest` — referenced `manifest_id` not retrievable/invalid.
• `subset-missing` — declared canonical subset not present or incomplete.
• `subset-order` — fields present but not in the declared order.
• `align-format` — `align_ascii` not in canonical numeric form.
• `stamp-invalid` — malformed stamp (missing fields or parse failure).
• `chain-fork` — `prev` does not chain to current HEAD and no fork procedure in effect.
• `clock-skew` — timestamp outside allowed skew window.
• `disclosure-violation` — disclosure exceeds manifest/profile policy.
• `policy-deny` — local policy prohibits acceptance despite syntactic validity.

Receiver actions (guidance)
• **MUST** record the error stamp and preserve the original exchange.
• **SHOULD** include `SSMNET-Checkpoint: HEAD=<HEX>` in follow-up diagnostics, if available.
• **MAY** provide an optional compact body with a single JSON object describing `code`, `reason`, and `hint`, avoiding any identity or PII.

One-line takeaway
Signal the problem, never rewrite history; append a stamped note, keep bytes intact, and make the failure independently replayable.

8C. Canonical error codes (normative set)

- **`E_POLICY_MISMATCH`** — Declared `band` is inconsistent with `manifest_id` cutpoints.
- **`E_BODY_HASH_MISMATCH`** — `sha256` over declared subset and/or raw body bytes does not match.
- **`E_STAMP_PREV`** — `prev` does not link to the current `HEAD` for this scope.
- **`E_MANIFEST_MISS`** — Referenced `manifest_id` not retrievable at well-known location.
- **`E_ALIGN_POLICY`** — Public `align` fails parity with the deterministic kernel (`a_c := clamp(...)` → `u := atanh(a_c)` → `U += w*u ; W += w` → `align := tanh(U / max(W, eps_w))`).
- **`E_SUBSET_DECL`** — `SSMNET-Canonical-Subset` missing, malformed, or unverifiable.
- **`E_DISCLOSURE`** — Declared disclosure mode conflicts with exposed fields.
- **`E_CHECKPOINT_DRIFT`** — Remote `HEAD` conflicts with published checkpoint for the same scope.
- **`E_FORMAT`** — Envelope or headers structurally invalid (syntax/encoding).

Extensibility: New codes **MAY** be added; existing semantics **MUST NOT** change.

8D. Minimal error body (optional)

When a body is permissible, return a compact, non-PII structure such as:

```
{
  "error": "E_BODY_HASH_MISMATCH",
  "reason": "Digest does not match declared canonical subset",
  "scope": "default",
  "head": "AB12...FF90",
  "stamp": "SSMCLOCK1|2025-11-07T12:45:33Z|n42|sha256=...|prev=AB12...FF90"
}
```

Norms

- **MUST** include `error`.
- **SHOULD** include `stamp` for the error note.
- **MUST NOT** echo secrets, tokens, or personal data.

8E. Receiver actions on failure

- **Log & stamp:** Record a **new** stamped incident note chaining from current `HEAD`.
- **Quarantine (SHOULD):** Segregate the failing item; do not accept into the verified chain.
- **Retry logic (MAY):**
  - For `E_MANIFEST_MISS`, retry retrieval with backoff.
  - For `E_CHECKPOINT_DRIFT`, refetch checkpoint, then reconcile per policy.
- **Operator notice (SHOULD):** Emit a small band-style UI chip (e.g., `"INCIDENT • E_BODY_HASH_MISMATCH"`) without exposing `align` unless public.

8F. Sender actions on failure

- **Do not mutate history.** Prepare a **fresh** envelope with corrected declarations; `prev` links to the last accepted `HEAD`.
- **Clarify manifest.** If policy text was ambiguous, publish an updated **new** `manifest_id`; never edit the old manifest.
- **Narrow disclosure.** If the error stems from overexposed fields, revert to **label-first** unless `full` is required by policy.

8G. Intermediary behavior

- **Pass-through first.** Preserve upstream bytes and declarations.
- **Annotate locally.** If detecting an error, emit its **own** stamped observation; **must not** rewrite upstream stamps.
- **Checkpoint awareness.** Intermediaries **MAY** publish a local `HEAD=<HEX>` to aid recovery.

8H. Privacy and safety

- **Least disclosure.** Error text **SHOULD** be short and abstract (e.g., “digest mismatch,” not “customer X invalid”).
- **No lane leakage.** Do not include `align` in errors unless already public.
- **No identity drag.** Error surfaces describe **content posture**, not people.

8I. Illustrative overlay snippets

HTTP-M style (illustrative)

```
SSMNET-Error: E_STAMP_PREV
SSMNET-Stamp: SSMCLOCK1|2025-11-07T12:47:02Z|n7|sha256=...|prev=1F3C...A902
```

Stream frame note (illustrative)

```
type: ssmnet.error
error: E_POLICY_MISMATCH
stamp: "SSMCLOCK1|2025-11-07T12:47:55Z|n8|sha256=...|prev=..."
```

One-line takeaway for Section 8

"Errors are stamped facts, not edits: signal the code, chain the note, keep bytes intact, and move forward append-only."

SECTION 9 — SECURITY, PRIVACY, AND ETHICS

9A. Security invariants (what the overlay guarantees)

- **Payload invariance.** Original bytes are never mutated by the overlay: `phi((m,a)) = m`.
- **Deterministic lane.** The lane math is fixed and replayable: `a_c := clamp(a_raw, -1+eps_a, +1-eps_a)` → `u := atanh(a_c)` → `U += w*u ; W += w` → `align := tanh( U / max(W, eps_w) )`.
- **Append-only continuity.** Time/order are chained by `SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEX`; repairs are **append-only**, never edits.
- **Subset binding.** Verification targets a **declared** canonical subset; formatting outside it cannot spoof integrity.

9B. Threats and mitigations (overlay-specific)

• **T1: Digest spoofing.**
Mitigation: Receivers **MUST** recompute `sha256` over the declared subset (and raw body bytes if declared) and compare to the advertised digest. Mismatch **MUST** be treated as verification failure.

• **T2: Chain forking / reordering.**
Mitigation: Verify `prev` against local `HEAD`; publish scope `HEAD` via well-known endpoints; quarantine branches unless a documented fork procedure exists.

• **T3: Manifest tampering or drift.**
Mitigation: Manifests are **immutable per `manifest_id`**; any change mints a **new `manifest_id`**. Keep historical manifests publicly retrievable for replay. Optionally, include a **manifest self-digest** (`sha256=<HEX>`) and/or a **time-stamp note** for provenance.

• **T4: Lane mis-compute (vendor drift).**
Mitigation: The normative kernel **must** follow `clamp -> atanh -> accumulate -> tanh` with eps guards. Where `align` is public, receivers **SHOULD** parity-check using the same inputs and weights.

• **T5: Intermediary mutation.**
Mitigation: Intermediaries **MUST NOT** alter bytes of fields in the declared subset and **MUST NOT** rewrite prior stamps. If annotating, **append** a new stamped note.

• **T6: Excess disclosure of internals.**
Mitigation: Default to **label-first** (`value+band`); disclose `align` only when required for ecosystem parity or explicitly declared public in the manifest/profile.

• **T7: Weak or reused nonces.**
Mitigation: Nonces **MUST** be generated via a **cryptographically secure** source and treated as **bytes** (length declared in manifest). Reuse across scopes **MUST** be avoided.

• **T8: Clock skew and ambiguous time anchors.**
Mitigation: Deploy **synchronized clocks** (e.g., NTP or hybrid). Receivers **SHOULD** apply a bounded `allowed_skew_s` tolerance and **MUST** validate monotonic linkage via `prev`. When in doubt, quarantine and seek resync using the published `HEAD`.

9C. Privacy posture (data minimization by design)

- **Label-first default.** Publish human band; keep numeric `align` private unless declared public.
- **PII avoidance.** Canonical subset **SHOULD NOT** include PII; manifests describe **content posture**, not identities.
- **Purpose limitation.** Bands communicate required operational stance (e.g., "REVIEW", "CRITICAL"), not user profiles.

- **Retention clarity.** Evidence bundles should carry a retention note; delete or rotate per policy without breaking published manifests and checkpoints.

9D. Ethical ground rules (humans before metrics)

- **Clarity over cosmetics.** Do not "soften" bands to look better. The dial is bounded for safety, not marketing.
- **Escalation honesty.** If a band implies an obligation, the obligation **must** be executed or explicitly overridden with a stamped note.
- **No identity drag.** Never convert posture labels into statements about people; posture labels describe **content/system state**.
- **Audit equity.** Keep manifests and checkpoints fetchable so independent reviewers can replay meaning without gatekeeping.

9E. Time, clocks, and anchors

- **UTC only.** Stamps use `UTC_ISO`; localized displays are UI-only.
- **Deterministic format.** `UTC_ISO` MUST be `YYYY-MM-DDThh:mm:ssZ`. Fractional seconds MAY appear **only** if declared in the manifest (e.g., `prec_ms=true`) and then MUST be present everywhere.
- **Monotonic order by chain.** `prev` establishes accepted order per scope; receivers MUST NOT rely on wall-clock order alone.
- **Skew window.** Manifests SHOULD declare `allowed_skew_s`. Receivers MAY accept stamps whose `UTC_ISO` differs from local time by ≤ `allowed_skew_s`; otherwise mark **out-of-window** and quarantine without altering payload.
- **Clock guidance.** Deployments SHOULD synchronize with a reliable source (e.g., NTP/precise time). Implementations SHOULD pair wall-clock (for stamps) with a monotonic counter (for local sequencing) to avoid reordering under leap/adjust events.
- **Genesis clarity.** The first item per scope uses `prev=NONE` and becomes the replay anchor.
- **Anchoring HEAD.** Publishers MAY expose a read-only `HEAD=<HEX>` (per scope) via a well-known endpoint to aid replay and fork detection.
- **Rollover notes.** If `(U,W)` epochs are used, emit a stamped rollover note at each boundary to assist analysis; history is never rewritten.
- **Drift recovery.** On detected drift or incorrect time, append a corrective stamped note that chains to the current `HEAD`; do not edit prior stamps.

9F. Operational hardening (recommended practices)

- **Keying & TLS.** Use transport security appropriate to your environment; SSM-NET does not replace encryption or key management.

- **Checksum discipline.** Hash over **byte-exact** bytes for bodies and canonical subsets; document serializers to avoid ambiguity.
- **Red-team the chain.** Periodically attempt fork, replay, and digest mismatch tests; publish results in evidence notes.
- **Scope hygiene.** Keep scope names stable (`default`, `orders`, `feed-alpha`) and document rollover procedures.

9G. Policy safety in manifests

- **Boundary inclusivity text.** Make cut edges explicit (e.g., `(-0.80, +0.60]`); ambiguous edges cause audit failures.
- **Hysteresis (optional).** To reduce flip-flop near edges, declare narrow buffers in the manifest narrative; document their math if used.
- **Assumptions.** State validity ranges ("sample rate ≥ X", "load ≤ Y"); out-of-assumption operation should trigger downgraded trust or separate bands.

9H. Minimal disclosure during incidents

- **Stamped incident notes.** On failure, append a short, non-PII stamped note referencing the failing item; do not reveal private `align` unless already public.
- **Containment first.** Quarantine failing scopes while preserving replay artifacts for post-incident review.

9I. Provenance & Minimization Audit (copy-ready checklist)

(Purpose: a periodic, evidence-based review that proves security posture, privacy minimization, and ethical obligations are upheld without touching payload bytes — collapse parity `phi((m,a)) = m` remains inviolable.)

Scope of the audit
Covers three pillars across all active scopes: **Security provenance**, **Privacy minimization**, and **Ethical obligations**. Verifiers rely only on published manifests, canonical subsets, stamped continuity, and offline evidence bundles.

Inputs required (read-only)

- Latest **manifests** used per scope (`/.well-known/ssmnet/manifest/<manifest_id>`).
- Current **checkpoint** (`HEAD=<HEX>` with `updated=<UTC_ISO>`).

- Most recent **evidence bundles** (manifests.json, envelopes.jsonl, hashes.txt, checkpoint.txt, verify.sh).
- Rotation log of `manifest_id` changes (append-only notes).

Security provenance — checks

1. **Digest correctness (subset + wire body if declared):** recompute `sha256( serialize(subset_fields) [+ raw_body_bytes_if_declared] )` for a sampled window; expect exact match with `hashes.txt`.
2. **Continuity chain:** walk `prev` to confirm append-only history; final equals `HEAD`. Detect forks; if present, require stamped fork note and resolution.
3. **Kernel determinism (if disclosure=full):** recompute lane via `a_c := clamp(a_raw, -1+eps_a, +1-eps_a) -> u := atanh(a_c) -> U += w*u ; W += w -> align := tanh( U / max(W, eps_w) )` and verify tolerances: batch vs stream `<= 1e-6`, shard-merge `<= 1e-12`.
4. **Intermediary preservation:** confirm upstream stamps and `SSMNET-Body-Hash` are forwarded unchanged; only **new** stamps are added.

Privacy minimization — checks

1. **Subset review:** canonical subset contains **no PII**; posture is expressed as `band` plus `manifest_id`.
2. **Disclosure posture:** public surfaces default to **label-first** (`value+band`); `align`/`align_ascii` appear only where `disclosure=full`.
3. **Decimal canonicality (if full):** `align_ascii` uses fixed sign + 6 decimals (e.g., `+0.732000`) to avoid serializer drift; matches recomputed float.
4. **Data residency / retention note:** manifests remain online; evidence bundles retain only what is necessary for replay; no secret material embedded.

Ethical obligations — checks

1. **Obligation encoding:** manifests declare human/agent obligations per band (e.g., review windows, safe-mode timing) with explicit boundary inclusivity.
2. **Actionability trace:** for sampled **CRITICAL** or escalatory bands, confirm that stamped timelines allow independent verification of whether obligations were met within declared windows.
3. **Proportionality:** verify that disclosure level (L1/L2/L3) is the minimum required to achieve safety and accountability for the use-case.
4. **No identity drag:** confirm that bands describe **system/content posture**, not people.

Sampling guidance

- Choose at least one **edge case** around each cut boundary (e.g., `(-0.80, +0.60]`).
- Include one **negative test** per bundle (single-byte flip in a working copy should fail deterministically).
- For long streams, include one **epoch boundary** sample and one **shard-merge** sample.

Expected audit artifacts

- **Provenance report** (text): scope, time window, `manifest_id` pins (sha256 of manifest text), computed vs expected `HEAD`, parity metrics.
- **Findings** (pass/fail with codes): `E_BODY_HASH_MISMATCH`, `E_STAMP_PREV`, `E_ALIGN_POLICY`, `E_SUBSET_DECL`, etc., each accompanied by a **new stamped note** (no history edits).
- **Remediation plan** (if any): rotate manifest, fix serializer, adjust disclosure, or update operator runbooks (Appendix J).

Failure handling (overlay-safe)

- Failures **do not** trigger retroactive edits. Publish stamped incident notes and remediate forward (e.g., manifest rotation, serializer fix), preserving all prior evidence.

One-line takeaway for Section 9

"Protect bytes, protect humans: bounded math, minimal disclosure, immutable manifests, and append-only stamps make meaning verifiable without exposing more than necessary."

SECTION 10 — FEDERATION LEVELS

10A. Purpose

Federation levels let independently governed systems interoperate **without forcing the highest disclosure**. Each peer **advertises a maximum level**; links **converge on the lowest common level** so posture and provenance remain portable across boundaries.

10B. Levels (normative)

- **L1 — Label-first (minimum)**
  - **Public:** `value` (bytes), `band`, `manifest_id`, `stamp`, canonical subset declaration.
  - **Private:** `align` (kept in logs).
  - **Receiver MUST:** verify `sha256` over the declared subset, verify the `prev` chain, fetch `manifest_id`, and confirm band **derivability** from the manifest's cutpoints.
  - **Use when:** privacy and simplicity dominate; public numeric parity on the lane is **not** required.

- **L2 — Lanes reproducible**
  - **Public: L1 +** `align` (or `align_ascii`) exposed.
  - **Receiver SHOULD:** recompute the lane via `a_c := clamp(a_raw, -1+eps_a, +1-eps_a) → u := atanh(a_c) → U += w*u ; W += w → align := tanh( U / max(W, eps_w) )`, then confirm `band := cutpoint_map(align, manifest_id)`.
  - **Use when:** independent **numeric parity checks** are desired across vendors.
- **L3 — Full evidence**
  - **Public: L2 +** downloadable **evidence bundle** (envelopes, manifests, hashes, checkpoint, `verify.sh`).
  - **Receiver MUST:** be able to reproduce **ALL CHECKS PASSED** offline.
  - **Use when:** third-party audits, regulatory replay, or **long-chain verification** are required.

10C. Negotiation

- **Advertise:** each peer declares its **max supported level** (`L1`/`L2`/`L3`).
- **Converge:** a link operates at `min(local_max, remote_max)`.
- **Downgrade rule:** when policies differ, federate at the **lower disclosure**; no peer may coerce higher disclosure.
- **Upgrade path:** peers may raise a link level **only** by mutual consent and explicit policy.

10D. What crosses the link (by level)

Item	L1	L2	L3
value (bytes)	yes	yes	yes
band, manifest_id	yes	yes	yes
align / align_ascii	no	yes	yes
stamp `SSMCLOCK1	UTC_ISO	nonce	sha256=HEX
canonical subset declaration	yes	yes	yes
evidence bundle (pull)	optional	optional	required

Table is descriptive; actual encodings follow the active profile (e.g., HTTP-M headers).

10E. Divergent policy handling

- **Different manifests**. Allowed. Each side uses its own `manifest_id` for local duties and enforcement.
- **Preserve sender envelope**. The sender's declared subset and band travel intact; **MUST NOT** be rewritten by receivers (collapse parity holds: `phi((m,a)) = m`).

- **Local remap for enforcement**. Receivers **MAY** map the sender's posture to their own bands using their local `manifest_id` and act locally; this remap is a **separate observation note** (append a new stamp) and is **not** part of the sender's committed subset.
- **Cross-validation (optional)**. At L2/L3, receivers **MAY** recompute `align` (where disclosed) to compare postures across manifests while acknowledging that band labels differ by policy.
- **Transparency**. When adding a local observation, receivers **SHOULD** cite the local `manifest_id` used for remap in that new stamped note.
- **Public stance**. Interfaces **SHOULD** publish which level they support (L1/L2/L3) and whether cross-manifest comparisons are informational only (no auto-action) or used for local duties.
- **Conflict rule**. Where sender and receiver policies disagree, **local policy governs local action**; the sender's envelope remains preserved for replay.

10F. Intermediaries across levels

- **Pass-through:** an `L1` mirror **MUST NOT** strip `L2`/`L3` fields; it may hide them from its own consumers but **must preserve upstream bytes and stamps**.
- **Append-only notes:** intermediaries may **add** their own stamped observation (new `stamp`) regardless of level; **never rewrite** upstream stamps.
- **Checkpoint relay:** at `L3`, intermediaries **SHOULD** expose `HEAD=<HEX>` to aid fast replay.

10G. Privacy posture by level

- **L1:** strongest default privacy; communicates only the **human stance** (`band`) with verifiable continuity.
- **L2: parity transparency**; reveals the numeric dial for independent math checks.
- **L3: audit transparency**; reveals artifacts sufficient for **offline reproduction**.

10H. Failure and fallback

• If a receiver at L2 cannot parity-check `align`, it MUST treat the session as L1 (label-first) and signal a non-fatal notice.
• If evidence cannot be fetched at L3, the receiver MUST continue at L2 or L1 (according to available fields) and log the deficiency; payload bytes remain valid under `phi((m,a)) = m`.
• If canonical subset text cannot be normalized for hashing, the receiver MUST refuse parity checks and fall back to continuity verification only.

• If a `prev` linkage is broken (chain gap), the receiver MUST accept the payload bytes unchanged under `phi((m,a)) = m`, mark the declaration **non-evidential**, and continue continuity from the last valid `HEAD` (or require a documented repair).

One-line takeaway for Section 10

"Declare your maximum, meet at the minimum: posture and provenance travel at L1, numeric parity at L2, and full offline replay at L3 — always append-only, never coercive."

SECTION 11 — OPERATIONS (DAY-0 → DAY-90)

11A. Day-0 — Overlay without friction (first hour)

Goal. Ship portable meaning with zero payload changes.

- **Publish a starter manifest** with explicit cutpoints and boundary inclusivity text.
- **Enable label-first disclosure** (`value+band`) by default.
- **Emit a continuity stamp** on outbound responses: `SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=NONE` (per-scope genesis).
- **Bind a canonical subset** (e.g., `["value","band","manifest_id"]`) and compute `sha256` over its byte-exact serialization.
- **Expose well-known paths:** `/.well-known/ssmnet/manifest/<manifest_id>` and `/.well-known/ssmnet/checkpoint`.
- **Keep lane private** (`align` logs-only) unless policy requires public disclosure.
- **Do not mutate payload bytes** at any point (collapse parity holds: `phi((m,a)) = m`).
- **Name the scope** (e.g., `default`, `orders`) and **persist `(U,W)` per scope** so continuity and replay are trivial.

Operator checklist (Day-0)

- **Payload invariance verified:** `phi((m,a)) = m`.
- **Digest parity confirmed** end-to-end for the declared canonical subset (and body, if declared).
- **Genesis recorded** with `prev=NONE`; local **HEAD** stored and checkpoint published.
- **Manifest reachable** at well-known; **disclosure default = label-first** confirmed.
- **Scope name and `(U,W)` state** initialized and durably persisted.

11B. Day-7 — Parity Sentinel & checkpoints

Goal: prove determinism across batch, stream, and shards.
• **Parity Sentinel:** run nightly tests confirming order-invariance of the lane kernel:
`a_c := clamp(a_raw, -1+eps_a, +1-eps_a) -> u := atanh(a_c) -> U += w*u ; W += w -> align := tanh( U / max(W, eps_w) ).`
Expect `|a_out_batch - a_out_stream| <= 1e-6`.
• **Publish checkpoints per scope:** `/.well-known/ssmnet/checkpoint -> HEAD=<HEX>` (include `updated=<UTC_ISO>`).
• **Incident stamping:** if any mismatch occurs, append stamped incident notes; never edit history.

Operator checklist (Day-7)
• Parity delta within tolerance.
• HEADs published and cached.
• Evidence export dry-run successful.

11C. Day-30 — Label-first discovery & evidence packs

Goal: enable self-service replay for internal and partner audits, without private trust channels.

- **Turn on discovery surface** in UI:
  Display something like: **"Rulebook: `<manifest_id>` • View"**, linking directly to the published manifest.
- **Ship evidence bundles** at:
  `/.well-known/ssmnet/evidence`
  containing:
  - `envelopes.jsonl` (canonical subset per line)
  - `manifests.json` (exact rulebooks referenced)
  - `hashes.txt` (per-item `sha256` over declared subset)
  - `checkpoint.txt` (`HEAD=<HEX>`)
  - `verify.sh` (single-command replay script)
- **Document rotation:** any policy change mints a **new `manifest_id`**; old manifests remain published for replay (never edited).
- **Privacy audit:** confirm the canonical subset contains **no PII**. **Posture ≠ identity.**

Operator checklist (Day-30)

- Running `verify.sh` on a fresh machine prints **ALL CHECKS PASSED**.
- Manifest rotation procedure is rehearsed (**mint new IDs**, never modify old manifests).
- Disclosure review confirms **label-first (`value+band`) remains default**, with `align` private unless explicitly declared public.

11D. Day-60 — Federation trials

Goal: interoperate at negotiated levels (L1/L2/L3).

- **Advertise maximum level** supported on each interface.
- **Negotiate to minimum level** with each peer; log the agreement.
- **Cross-vendor parity (L2+):** if `align` is public, receivers recompute lane and confirm `band := cutpoint_map(align, manifest_id)`.
- **Intermediary posture:** mirrors preserve bytes and stamps; may add their own stamped observation, never rewrite.

Operator checklist (Day-60)

- At least one external link operating at L1.
- One partner link at L2 or L3 validated with evidence replay.
- Fork handling documented and tested.

11E. Day-90 — Transparency cadence

Goal: institutionalize append-only accountability.

- **Monthly transparency notes**: publish bundle hashes, HEAD, and parity metrics (`max_delta_align`, `evidence_count`).
- **Red-team drills**: attempt `E_BODY_HASH_MISMATCH`, `E_STAMP_PREV`, `E_POLICY_MISMATCH`; verify error signaling and quarantine paths.
- **Capacity & retention**: document evidence retention windows and cold storage locations; ensure manifests remain hot and immutable.

Operator checklist (Day-90)

- Red-team results stamped and archived.
- Retention policy public; manifests immutable.
- Performance SLOs measured (header overhead, verify latency).

11F. Minimal runbooks (paste-ready)

Scope genesis

1. Set `prev=NONE`.
2. Emit first stamped item; store local `HEAD`.
3. Publish checkpoint at `/.well-known/ssmnet/checkpoint`.

Incident handling

1. Quarantine the failing item (do **not** discard payload; `phi((m,a)) = m` holds).
2. Append a stamped error note:
   `SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEAD`.
3. Prepare a corrected envelope and chain **from the latest `HEAD`**.
4. **Do not edit history.** History is append-only.

Manifest rotation

1. Draft new cutpoints and obligations.
2. Mint a **new `manifest_id`**.
3. Publish at the well-known manifest endpoint.
4. Begin using the new ID; **keep old manifests online** for replay.

One-line takeaway for Section 11

"Ship label-first on Day-0, prove determinism by Day-7, enable evidence replay by Day-30, federate by Day-60, and publish stamped transparency by Day-90—always append-only."

SECTION 12 — VERIFICATION (GOLDEN DIAGNOSTICS)

12A. Purpose

Establish a **deterministic, vendor-neutral** test pack that proves SSM-NET behavior across **math, chains, and transports**. Any conforming stack should pass these diagnostics **byte-for-byte** with identical outputs on independent implementations.

12B. Kernel determinism (lane math)

Test: Compute `align` from identical inputs via three paths: **batch**, **stream**, **sharded-merge**.

- **Pipeline (normative):** `a_c := clamp(a_raw, -1+eps_a, +1-eps_a) → u := atanh(a_c) → U += w*u ; W += w → align := tanh( U / max(W, eps_w) )`
- **Expect:** `|align_batch − align_stream| ≤ 1e-6` and `|align_batch − align_shardmerge| ≤ 1e-12` with identical `(eps_a, eps_w, w)`.

- **Fail conditions:** any order-sensitivity beyond tolerance; mismatch in fused value due to serializer drift, float mode differences, or missing clamp.

12C. Boundary accuracy (band cutpoints)

Test: Dial values **exactly at and around** each cut boundary.

- **Manifest cut:** e.g., `(-0.80, +0.60]` for `"A0"`.
- **Vectors:** `{align = -0.800000, -0.800001, -0.799999, +0.600000, +0.600001, +0.599999}`.
- **Expect:** Bands follow the manifest **boundary inclusivity** text verbatim.
- **Fail conditions:** off-by-one at edges; rounding to fewer decimals than the manifest declares; tolerance misapplied relative to `cmp_tolerance` (if present).

12D. Shard/merge invariance

Test: Split a series into `N` shards; compute per-shard `(U,W)`; fold as `U_total := ΣU_i`, `W_total := ΣW_i`; then `align_total := tanh( U_total / max(W_total, eps_w) )`.

- **Expect:** `align_total` equals monolith within `≤ 1e-12`.
- **Fail conditions:** re-clamping at shard joins; inconsistent `eps_w`; mixing different `w` policies within the same scope.

12E. Stamp chain continuity

Test: Validate `SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEX` across a scope.

- **Genesis:** first item uses `prev=NONE`.
- **Expect:** every `prev` equals prior accepted item's `sha256`; last equals **HEAD**.
- **Fork test:** introduce a synthetic fork; receiver **MUST** reject non-HEAD branch.
- **Epoch rollover note (if used):** a stamped rollover marker **SHOULD** appear near boundaries; continuity remains linear.
- **Fail conditions:** non-linear chain; reused `nonce` with different subset; `prev` mismatch; missing or malformed fields.

12F. Canonical subset digest parity

Test: Recompute `sha256` over the **declared canonical subset** (e.g., `["value","band","manifest_id"(,"align_ascii"?)]`) **plus raw body bytes if declared**.

- **Serializer stability.** The declared list **defines field order**. The serialized bytes **MUST** match exactly across implementations when the same inputs are provided.
- **Normalization (normative).**
  • All textual subset fields **MUST** be UTF-8 in **NFC** normalization.
  • Newline **MUST** be `\n` (LF) only.
  • No BOM.
  • If `TRIM_WHITESPACE_PER_LINE` is enabled by manifest/profile, trim only **leading and trailing** space (0x20) and tab (0x09) **per line**, without altering interior whitespace or binary data.
- **Body commitment.** When body commitment is declared, compute `sha256` over the **exact on-the-wire body bytes**. If the transport compresses, hash the **compressed** bytes.

Expect: Digest match across independent implementations and across batch vs. stream paths.

Fail conditions:

- Hidden or additional fields included in the subset.
- Fields **omitted** from the subset that were declared.
- Field **order** not matching the declared list.
- Whitespace or quoting differences **within** the committed subset.
- Non-canonical numeric form for `align_ascii` (e.g., missing sign or inconsistent precision).
- Hashing a **decoded** or **reformatted** body instead of the raw wire bytes.

12G. Profile conformance (HTTP-M example)

Test: Round-trip headers for request/response.

- **Expect:**
  - Payload bytes **identical** (`phi((m,a)) = m`).
  - `SSMNET-Body-Hash == sha256(raw_body_bytes)` where `raw_body_bytes` are **exactly** the bytes on the wire (after any content encoding).
  - Unknown SSMNET headers **safely ignored** by legacy stacks.
- **Fail conditions:** body rewrites; hashing over decoded text instead of raw bytes; header case/quoting errors that change digest input; misordered canonical subset fields.

12H. Privacy posture checks

Test: Run the same flow with **label-first** vs **full**.

- **Expect (label-first):** `band`, `manifest_id`, `stamp`, subset proof present; `align` **absent** from public fields.
- **Expect (full):** `align` and `align_ascii` exposed; receiver parity-checks kernel and band.
- **Fail conditions:** leaking `align` when not declared public; using `align` to imply identity traits (not allowed).

12I. Intermediary preservation

Test: Mirror/CDN adds its own stamped observation while forwarding bytes.

- **Expect:** upstream stamps and `SSMNET-Body-Hash` **preserved**; new stamp **appended** with `prev=<prior HEAD>`.
- **Fail conditions:** rewriting upstream stamps; recomputing digest over altered subset; dropping canonical subset declaration.

12J. Negative tests (must fail clearly)

- **Digest mismatch:** alter one byte in the subset; expect `E_BODY_HASH_MISMATCH`.
- **Broken prev:** point `prev` to an older item; expect `E_STAMP_PREV`.
- **Manifest missing:** reference unknown `manifest_id`; expect `E_MANIFEST_MISS`.
- **Lane parity fail (full mode):** perturb `eps_a` or `eps_w`; expect `E_ALIGN_POLICY`.
- **Subset declaration error:** remove a declared field from serialization; expect `E_SUBSET_DECL`.
- **Normalization error:** change text to non-NFC form; expect `E_TEXT_NORM`.
- **Boundary tolerance misuse:** apply tolerance opposite to declared inclusivity; expect `E_BOUNDARY_RULE`.

12K. Golden vector kit (minimal contents)

- **`vectors_lane.csv`** — rows of `a_raw, w, eps_a, eps_w, align_expected`.
  - Use **fixed-precision decimals** for any public `align_ascii` (e.g., `+0.732000`).
  - Include at least **three permutations** of the same series (**batch**, **stream**, **shard-merge**) to prove order-invariance.
- **`vectors_cuts.csv`** — `align` values at and around each manifest boundary with `band_expected`.
  - Cover both sides of each cut (e.g., `-0.800000, -0.800001, -0.799999`).

- **`envelopes.jsonl`** — one canonical-subset object per line, in **declared field order**.
  - Example order: `["value","band","manifest_id"(,"align_ascii"?)]`.
- **`manifests.json`** — exact manifest texts referenced by the vectors (**immutable copies**).
- **`hashes.txt`** — one line per envelope: `sha256=<HEX>` over the **declared canonical subset bytes** (and raw body, if declared).
- **`checkpoint.txt`** — `HEAD=<HEX>` for the set; genesis items use `prev=NONE`.
- **`verify.sh`** — recompute digests, follow the `prev` chain, and check band mapping under the referenced manifest; print **ALL CHECKS PASSED** on success.

Recommended packaging notes (non-normative but helpful)

- **Encoding:** UTF-8; **line endings:** `\n`; textual materials used in hashing **MUST** be **NFC**.
- **Decimal policy:** fixed width for any public `align_ascii` to avoid locale drift.
- **Tolerances:** lane parity ≤ `1e-6` (batch vs. stream) and shard/merge ≤ `1e-12`.
- **Negative tests:** include at least one altered byte to trigger a clear digest mismatch.
- **Archive:** deterministic ordering inside the bundle to keep `bundle.sha256` stable.

12L. Tolerances and math notes

- **Float mode.** 64-bit floats are recommended for general deployments.
  Embedded targets **MAY** use fixed-point arithmetic with a documented **Q-format**, provided that resulting `align` values remain within the required tolerances and canonical formatting (e.g., `+0.732000`) is preserved when `align_ascii` is disclosed.
- **Tolerances.**
  • Lane parity tolerance (batch vs. stream): `|a_out_batch - a_out_stream| <= 1e-6`.
  • Shard / merge tolerance across distributed accumulation: `<= 1e-12`, provided the same weight_rule and clamping parameters are used.
- **Clamping (normative).**
  `a_c := clamp(a_raw, -1+eps_a, +1-eps_a)` **MUST** precede `atanh(a_c)` to avoid domain errors and ensure stable, reproducible results.
  Accumulation then proceeds: `u := atanh(a_c) -> U += w*u ; W += w -> align := tanh( U / max(W, eps_w) )`.
- **Epoch rollover.**
  When `(U, W)` epochs are used, rollover boundaries **SHOULD** be marked by a stamped note.
  Parity requirements apply **within each epoch**, and continuity across epoch boundaries is maintained **via stamped sequence (`prev`)**, **not** by reusing old `(U, W)` state.

One-line takeaway for Section 12

“Prove it three ways: same lane fused everywhere, same bands at the edges, and the same stamped chain — so every independent verifier prints ALL CHECKS PASSED.”

SECTION 13 — UI HINTS (OPTIONAL BUT RECOMMENDED)

13A. Principles

- **Meaning without noise.** Show humans the **band** and **rulebook** first; keep numeric `align` hidden unless disclosure is **full**.
- **Zero mutation.** UI **must not** rewrite payload bytes; `phi((m,a)) = m` holds for all displays.
- **Stamped truth.** Surface continuity in a compact way; full evidence is one click away.

13B. Minimal components (paste-ready copy blocks)

Disclosure chip (always visible)

Rulebook: `<manifest_id>` • **View**

Band badge (label-first default)

Posture: `<BandLabel>`
Small color band (optional), text must read clearly in monochrome.

Continuity crumb

Stamped: `UTC_ISO` • `prev=…HEX`
Truncation allowed for `prev` (e.g., first/last 4 chars).

Evidence action

Export Evidence (downloads bundle with `envelopes.jsonl`, `manifests.json`, `hashes.txt`, `checkpoint.txt`, `verify.sh`)

Lane parity (full mode only)

Align: `+0.732000` (recomputed parity **OK**)
or
Align: `+0.732000` (parity **MISMATCH** → investigate)

13C. States & micro-copy

- **Normal (label-first):**
  - Chip: "**Rulebook:** `<manifest_id>` • View"
  - Badge: "**Posture:** `A0`"
- **Critical:**
  - Badge: "**Posture:** `CRITICAL` — *execute escalation now*"
  - Optional timer showing promised response window from the manifest.
- **Incident (verification fail):**
  - Banner: "**Incident:** `E_BODY_HASH_MISMATCH` — bytes preserved; investigation stamped."
  - Link: "View error note" (opens stamped entry)
- **Parity warning (full mode):**
  - Inline: "**Align parity drift** — check `eps_a`, `eps_w`, `w`."
  - CTA: "Open verify panel"

13D. Verify panel (human-readable drawer)

Show a short, reproducible checklist:

- **Canonical subset:** `["value","band","manifest_id"(,"align_ascii"?)]`
- **Digest:** `sha256=<HEX>` — **OK / FAIL**
- **Stamp:** `SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEX` — **OK / FAIL**
- **Manifest:** `<manifest_id>` — boundaries: `(-0.80, +0.60]` — **OK / FAIL**
- **Lane parity (full mode):** recompute via
  `a_c := clamp(a_raw, -1+eps_a, +1-eps_a)` → `u := atanh(a_c)` → `U += w*u ; W += w` → `align := tanh( U / max(W, eps_w) )` — **OK / FAIL**

End with a single line: **ALL CHECKS PASSED** or the first failing step.

13E. Accessibility & internationalization

- **Color-independent meaning.** Do not rely on hue alone; pair color with text labels ("A0", "CRITICAL").
- **Monospace for proofs.** Render stamps, hashes, and subsets in monospace to avoid glyph drift.
- **UTC display with local helper.** Show `UTC_ISO`; optionally add local time in parentheses for convenience.
- **Short, clear language.** Use phrases like "**Posture:** A0", "**Stamped**", "**Evidence**" to minimize translation risk.

13F. Privacy-first defaults

- **Label-first by default.** Show `band`, hide numeric `align`.
- **No identity drag.** UI elements describe **content posture**, not people.
- **Redaction discipline.** Ensure the canonical subset excludes PII; never render hidden fields in tooltips.

13G. Operator screens (control room patterns)

- **Stability trail sparkline.** A tiny history line with three states: *improving / steady / declining*.
- **Scope HEAD tile.** "**HEAD:** `<HEX>` • Updated `UTC_ISO` • **Publish**"
- **Rotation notice.** When `manifest_id` changes, show a small banner: "**New rulebook active** — `<manifest_id>`."

13H. Empty, error, and loading

- **Empty:** "No stamped items yet (genesis uses `prev=NONE`)."
- **Loading:** "Fetching manifest and checkpoint…"
- **Network error:** "Could not retrieve `<manifest_id>` — retry or check well-known endpoint."

13I. Dark mode & print fidelity

- **Dark mode:** maintain contrast for badges; ensure stamps/hashes are legible.
- **Print/PDF:** keep proofs in monochrome with sufficient line breaks; avoid truncating hashes.

One-line takeaway for Section 13

"Show bands and rulebooks first, stamps on demand, and parity only when public—make truth legible without leaking more than you must."

SECTION 14 — INDEPENDENCE, LEGAL POSITION, AND INTEROPERABILITY

(Why SSM-NET is not "just another header set")

14A. Independence and ownership

SSM-NET is an open standard. It is free to implement in any context — public, private, industrial, municipal, national, academic, commercial, or off-world — with **no fees**, **no registration**, and **no approval workflow**.

Citation (required). When implementing or adapting, cite the concept name **"Shunyaya Symbolic Mathematical Network (SSM-NET)"** as the origin of the **manifest-first, stamped alignment-lane overlay** for network communication.

No exclusive control. No implementer may claim exclusive stewardship, official registry status, or mandatory gatekeeping over SSM-NET. Conformance is by **self-declaration** and **reproducibility** (manifests, subset digests, stamped continuity).

Why this matters

- Protects the semantic core: `phi((m,a)) = m`, the bounded lane kernel `a_c := clamp(...) → u := atanh(a_c) → U += w*u ; W += w → align := tanh( U / max(W, eps_w) )`, **bands from manifest cutpoints**, and **SSMCLOCK1** stamp continuity.
- Prevents weakening: no stripping bands or stamps while still calling it SSM-NET.
- Enables emergency adoption: teams can add headers and well-known endpoints without legal friction.

14B. Not "just another serialization or header format"

Traditional formats ask: **"Can I parse this?"**
SSM-NET asks: "Can anyone later *prove* what was said, how serious it looked, under which rulebook, and in what sequence?"

Identity of SSM-NET lives in semantics, not punctuation:

- **value** (bytes) stay invariant: `phi((m,a)) = m`
- **align** is bounded and reproducible: clamp → atanh → fuse → tanh
- **band** is the human stance from declared cutpoints
- **manifest_id** freezes the rulebook in time

- **stamp** chains time, order, and the **declared canonical subset**

You may carry SSM-NET inside any envelope or transport; it **remains SSM-NET** by these semantics, not by the container.

14C. Coexistence with existing transports and stacks

You can embed SSM-NET semantics:

- in HTTP-style headers (HTTP-M),
- in API metadata blocks,
- in stream/message frames,
- in device side-band key-values,
- in logs, tables, or CSV for audit export.

Overlay, not a fork. You are not asserting that your transport "is SSM-NET"; you are declaring that your messages **carry SSM-NET's accountability surface**: `value` • `band` • `manifest_id` • `stamp` (+ optional public `align`). The original payload remains byte-identical under `phi((m,a)) = m`.

14D. How to ship SSM-NET without legal friction (checklist)

1. **Preserve the truth lane**
   - Original bytes **unchanged**: `phi((m,a)) = m`.
   - No "policy-adjusted" rewrites of value.
2. **Provide the trust lane**
   - If exposing `align`, compute via the **published kernel**:
     `a_c := clamp(a_raw, -1+eps_a, +1-eps_a)` → `u := atanh(a_c)` → `U += w*u ; W += w` → `align := tanh( U / max(W, eps_w) )`
   - Declare `eps_a`, `eps_w`, and `w` policy.
3. **Attach the rulebook**
   - Include `manifest_id` referencing public cutpoints, boundary inclusivity, obligations, timing windows, assumptions.
4. **Stamp continuity**
   - Emit `SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEX` over a **declared canonical subset** (and raw body if declared).
   - Genesis uses `prev=NONE`. Repairs are **append-only**.
5. **Avoid exclusivity language**
   - Do not market as "the only official SSM-NET" or require paid registration.
6. **State the boundary**
   - Specification is **"as-is"**, **no warranty**, **no endorsement** implied by implementation.

Meet these, and you're using SSM-NET in its intended spirit: **portable, verifiable responsibility** without lock-in.

14E. Optional extensions (freedom with boundaries)

Implementers **MAY** add optional blocks (e.g., signature wrappers, privacy offsets, gating) **so long as**:

- Core math and semantics remain intact and auditable.
- Cutpoint meanings and obligations are not altered silently.
- Any extension fields are **outside** the canonical subset unless explicitly declared.

14F. Responsibilities once you emit SSM-NET

- **No hiding risk.** If the dial is uncomfortable, you **do not** repaint the band.
- **Stand by the manifest.** If `"CRITICAL"` implies a human in `<= T`, that is the obligation.
- **Timeline honesty.** Do not reorder or delete; broken chains will show it.
- **Human dignity.** Bands describe **content/system posture**, not people or identities.

14G. Openness & attribution (concise statement for docs)

- **Open standard.** Permanent free use; no registry, no fees.
- **Citation.** Use the concept name **"Shunyaya Symbolic Mathematical Network (SSM-NET)"** when describing the approach.
- **Independence.** No exclusive steward; conformance by deterministic math, published manifests, and stamped verification.
- **Datasets/examples.** Any third-party datasets retain their original licenses.
- **No warranty or endorsement.** Provided "as-is." Implementation does not imply affiliation.

14H. Conformance statements & self-attestation (paste-ready)

(Purpose: give neutral, legally safe wording for teams to declare what they actually implemented, at which level(s), on which profiles—without implying exclusivity or central certification. All claims must be reproducible via manifests, well-known endpoints, and evidence bundles.

Payload invariance `phi((m,a)) = m` *and the kernel lane* `a_c := clamp(a_raw, -1+eps_a, +1-eps_a) -> u := atanh(a_c) -> U += w*u ; W += w -> align := tanh( U / max(W, eps_w) )` *remain non-negotiable.)*

14H.1 What you MAY claim (if true)

- **Profile binding:**
  "This service implements SSM-NET overlay semantics for **<profile>** (e.g., HTTP-M, WS-M, API-M, MESH-M, IOT-M)."
- **Disclosure level:**
  "This endpoint operates at **L1** / **L2** / **L3** as defined in Section 10."
- **Scope identification:**
  "Conformance applies to scope(s): **<scope-names>** (each with its own genesis and HEAD)."
- **Rulebook publication:**
  "Manifests are published at well-known paths per Section 7; each **manifest_id** is immutable once published."
- **Deterministic kernel:**
  "Alignment lane is computed exactly as specified (clamp → atanh → accumulate → tanh) with declared `eps_a`, `eps_w`, and `w` policy."
- **Evidence availability (if L3):**
  "Self-service evidence bundles are exposed at the well-known path and reproduce **ALL CHECKS PASSED** offline."

14H.2 What you MUST NOT claim

- **No exclusivity:**
  "We are the official steward / exclusive authority" (disallowed; standard is open and non-exclusive).
- **No implied certification:**
  "Certified by SSM-NET" (unless you actually name a third-party that performed a public audit and provide the report link).
- **No identity guarantees:**
  "Bands identify or score people" (disallowed; bands describe content/system posture, not identity).
- **No payload mutation:**
  Any claim that contradicts `phi((m,a)) = m` (the overlay must never alter original bytes).

14H.3 Self-attestation template (drop-in)

```
SSM-NET Self-Attestation (Public Statement)

Service / Product: <name>
Profiles: <HTTP-M|WS-M|API-M|MESH-M|IOT-M>
```

```
Disclosure Level(s): <L1|L2|L3>
Conformance Scopes: <scope-1, scope-2, ...>

Invariants:
- Payload invariance: phi((m,a)) = m
- Lane kernel: a_c := clamp(a_raw, -1+eps_a, +1-eps_a) -> u := atanh(a_c)
               U += w*u ; W += w -> align := tanh( U / max(W, eps_w) )

Manifests:
- Published at /.well-known/ssmnet/manifest/<manifest_id>
- Immutable per manifest_id; rotation creates a new manifest_id

Discovery:
- Checkpoint: /.well-known/ssmnet/checkpoint  (reports HEAD=<HEX>)
- Evidence:   /.well-known/ssmnet/evidence     (L3; deterministic archive)

Golden Diagnostics:
- Kernel determinism: batch vs stream <= 1e-6; shard-merge <= 1e-12
- Boundary accuracy: edges match declared inclusivity
- Canonical subset digest parity: sha256 over declared subset [+ body if
declared]
- Continuity: append-only prev chain; genesis uses prev=NONE

Limitations / Notes:
- Identity is out of scope; bands describe system posture, not people
- Known deviations (if any): <none | brief description + mitigation +
timeline>

Date of Statement: <UTC_ISO>
Contact for Reproducibility: <ops email or URL>
```

14H.4 Evidence checklist for an external reviewer

- **Manifests** for all referenced `manifest_id` values, byte-stable and immutable.
- **Envelopes** (declared canonical subset only), one per line, ordered.
- **Hashes** (`sha256=<HEX>` per line) that match the subset bytes (and body bytes if declared).
- **Checkpoint** showing `HEAD=<HEX>` that equals the last verified digest.
- **Verify script** that prints **ALL CHECKS PASSED** on a clean machine with no network.
- **Optional body/** folder if body binding is part of the subset contract.

14H.5 Rotation and deprecation language (paste-ready)

- "Threshold or obligation changes are shipped by minting a **new manifest_id**; previous manifest texts remain online for replay."
- "Scopes may deprecate; historical continuity is preserved via published `HEAD` and evidence bundles. No history is rewritten."

14H.6 Interop disclosure (multi-party links)

- "This interface supports federation at **L1/L2/L3**. Links operate at `min(local_max, remote_max)`. Cross-manifest comparisons are **informational** unless a jointly published treaty manifest declares equivalence."

14H.7 Minimal UI/legal footer (optional, neutral)

- "**Posture, not identity.** Bands indicate the state of signals/systems under a published rulebook."
- "**Append-only truth.** Time/order are anchored via `SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEX` and public `HEAD`."
- "**Reproducibility first.** Anyone may fetch the manifest and replay the same bands from the same bytes."

One-line takeaway for Section 14

"SSM-NET is an open, non-exclusive semantics layer for the internet: keep bytes intact, bind a canonical subset, freeze policy in a manifest, and stamp continuity—so responsibility can travel anywhere."

SECTION 15 — COMPLIANCE CHECKLIST & COPY BLOCKS

15A. Compliance checklist (paste-ready, normative)

Math & invariants

- **Deterministic lane kernel implemented:** `a_c := clamp(a_raw, -1+eps_a, +1-eps_a)` → `u := atanh(a_c)` → `U += w*u ; W += w` → `align := tanh( U / max(W, eps_w) )`
- **Payload invariance guaranteed:** `phi((m,a)) = m` (original bytes never mutated)
- **Tolerances documented:** batch vs stream ≤ `1e-6`; shard/merge ≤ `1e-12`
- **`eps_a`, `eps_w`, `w` declared** in the manifest or profile notes

Manifests & bands

- **Published `manifest_id`** with explicit boundary inclusivity text
- **Cutpoints frozen**; any change **mints a new `manifest_id`**
- **Band obligations stated** (timings, human escalation, overrides)
- **Disclosure mode set** (`value-only` / `value+band` / `full`)

Canonical subset & digest

- **Subset declared** (e.g., `["value","band","manifest_id"(,"align_ascii"?)]`)
- **`sha256` computed** over byte-exact subset serialization [+ raw body bytes if declared]
- **Serializer stability documented** (field order, encoding)

Continuity & checkpoints

- **Stamp emitted:** `SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEX`
- **Genesis uses `prev=NONE`**; **HEAD tracked** per scope
- **Checkpoints exposed** at `/.well-known/ssmnet/checkpoint`

Well-known publication

- **Manifest fetchable** at `/.well-known/ssmnet/manifest/<manifest_id>`
- **Evidence bundle available** at `/.well-known/ssmnet/evidence` (envelopes, manifests, hashes, checkpoint, `verify.sh`)

Privacy & ethics

- **Label-first default** unless `full` is required
- **No PII** in canonical subset; posture describes **content/system state**, not people
- **Incident handling append-only**; errors stamped with minimal disclosure

Intermediaries

- **Bytes preserved**; upstream stamps and subset intact
- **Local observation stamped** as a new note; no rewrites

Federation

- **Max level advertised** (L1/L2/L3); **link operates at minimum**
- **Cross-manifest comparisons** treated as informational unless explicitly agreed

---

15B. Copy blocks (drop-in text for specs, READMEs, and UIs)

Short disclosure (spec intro)

Bytes are invariant. Labels come from a declared rulebook.
A bounded alignment lane is computed via `clamp → atanh → U/W → tanh`.
A canonical subset is digested with `sha256` and chained by
`SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEX`.
Anyone can replay meaning without changing a single payload byte.

Banner (HTTP-M profile)

HTTP-M — *Keep your bytes. Add a tiny lane. Declare your rulebook. Stamp every exchange.*

Continuity notice (ops footer)

Continuity: append-only. If a defect is found, a new stamped note is added; history is never edited.

Evidence CTA (UI button text)

Export Evidence — envelopes • manifests • hashes • checkpoint • verify.sh

Parity badge (full mode)

Align parity: recomputed via `clamp → atanh → U/W → tanh` — **OK**

Incident micro-copy

Incident: `E_BODY_HASH_MISMATCH` — bytes preserved; investigation stamped and chained.

15C. Minimal header set (HTTP-M quick reference)

Request
SSMNET-Manifest: <manifest_id>
SSMNET-Disclosure: value-only | value+band | full
SSMNET-Canonical-Subset: ["value","band","manifest_id"(,"align_ascii"?)]
SSMNET-Body-Hash: sha256=<HEX> // if body present
SSMNET-Stamp: SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEX

Response
SSMNET-Manifest: <manifest_id>
SSMNET-Band: <BandLabel> // optional (label-first)
SSMNET-Align: <FloatAlign> // optional (only if public)

SSMNET-Canonical-Subset: ["value","band","manifest_id"(,"align_ascii"?)]
SSMNET-Body-Hash: sha256=<HEX>
SSMNET-Stamp: SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEX
SSMNET-Checkpoint: HEAD=<HEX> // optional

15D. One-page Quickstart (outline)

1. **Publish a starter manifest** with cutpoints and boundary text.
2. **Add headers** above to one GET/POST endpoint (label-first).
3. **Compute digest** over the canonical subset (and body if present).
4. **Emit a stamp** chaining to the previous `HEAD`.
5. **Host well-known** manifest, checkpoint, evidence.
6. **Run `verify.sh`** → expect **ALL CHECKS PASSED** on a clean machine.

One-line takeaway for Section 15

"If you can prove the math, the rulebook, the subset digest, and the chain—without touching the bytes—you're SSM-NET compliant."

SECTION 16 — ANNEX INDEX & WORKING MATERIALS

(Compact, copy-ready annexes. Each item is minimal now and expansion-friendly.)

Annex A — Header Grammar (ABNF-style, illustrative)

```
; ABNF core
DIGIT          = %x30-39
HEXDIG         = DIGIT / %x41-46 / %x61-66  ; 0-9 A-F a-f
ALPHA          = %x41-5A / %x61-7A
WSP            = %x20 / %x09
DQUOTE         = %x22

manifest-id    = 1*( ALPHA / DIGIT / "." / "-" / "_" )
disclosure     = "value-only" / "value+band" / "full"

; JSON-like list encoded as ASCII without spaces preferred
subset-field   = DQUOTE ( "value" / "band" / "manifest_id" / "align_ascii"
) DQUOTE
subset-list    = "[" subset-field *( "," subset-field ) "]"

sha256hex      = 64*64( HEXDIG )
```

```
utc-iso        = 4DIGIT "-" 2DIGIT "-" 2DIGIT "T" 2DIGIT ":" 2DIGIT ":"
2DIGIT "Z"
nonce          = 1*( ALPHA / DIGIT / "-" / "_" )
prevhex        = "NONE" / sha256hex

ssmclock1      = "SSMCLOCK1" "|" utc-iso "|" nonce "|" "sha256=" sha256hex
"|" "prev=" prevhex

; Request (client → server)
SSMNET-Manifest         = "SSMNET-Manifest:" WSP manifest-id
SSMNET-Disclosure       = "SSMNET-Disclosure:" WSP disclosure
SSMNET-Canonical-Subset = "SSMNET-Canonical-Subset:" WSP subset-list
SSMNET-Body-Hash        = "SSMNET-Body-Hash:" WSP "sha256=" sha256hex
SSMNET-Stamp            = "SSMNET-Stamp:" WSP ssmclock1

; Response (server → client)
SSMNET-Band             = "SSMNET-Band:" WSP DQUOTE 1*VCHAR DQUOTE
SSMNET-Align            = "SSMNET-Align:" WSP ["+" / "-"] 1*DIGIT ["."
1*DIGIT]
SSMNET-Checkpoint       = "SSMNET-Checkpoint:" WSP "HEAD=" sha256hex
```

Notes

- Header names are case-insensitive; values are case-sensitive where specified.
- `SSMNET-Align` is present only in **full** disclosure.
- Prefer no extra whitespace in `subset-list` to avoid serializer drift.

Annex B — Evidence Bundle Layout (minimal, reproducible)

```
/evidence/
  manifests.json        ; array of manifest objects keyed by manifest_id
  envelopes.jsonl       ; one JSON object per line with canonical subset
  hashes.txt            ; "<sha256hex>  <ordinal or key>"
  checkpoint.txt        ; "HEAD=<sha256hex>\nupdated=<utc-
iso>\nscope=<name>\n"
  verify.sh             ; tiny verifier (see below)
```

verify.sh (illustrative skeleton)

```
#!/bin/sh
set -e

echo "== SSM-NET verifier =="
HEAD_FILE="checkpoint.txt"
HASHES="hashes.txt"
ENVS="envelopes.jsonl"

HEAD=$(grep "^HEAD=" "$HEAD_FILE" | cut -d= -f2)
prev="NONE"

ord=0
while IFS= read -r line; do
  ord=$((ord+1))
```

```
  sha=$(printf "%s" "$line" | awk '{print $1}')
  # check chain
  if [ "$prev" = "NONE" ]; then
    : # genesis ok
  else
    : # in a fuller script, we'd read the stamped prev per envelope
  fi
  prev="$sha"
done < "$HASHES"

if [ "$prev" = "$HEAD" ]; then
  echo "ALL CHECKS PASSED"
else
  echo "FAIL: last hash != HEAD"
  exit 1
fi
```

(Production verifiers recompute `sha256` over the declared canonical subset bytes and validate each `SSMCLOCK1` line's `prev`.)

Annex C — Golden Vectors (starter files)

`vectors_lane.csv` (sample rows)

```
# a_raw,w,eps_a,eps_w,align_expected
-0.400000,1,1e-6,1e-9,-0.400000
+0.732000,1,1e-6,1e-9,+0.732000
+0.999000,3,1e-6,1e-9,+0.998667
```

`vectors_cuts.csv` (boundary checks)

```
# align,band_expected,edge_note
-0.800000,A++,left-inclusive
-0.799999,A0,open-left
+0.600000,A0,right-inclusive
+0.600001,CRITICAL,over-right
```

`envelopes.jsonl` (subset only)

```
{"value":"<opaque>","band":"A0","manifest_id":"TRANSPORT_POSTURE.DEMO"}
{"value":"<opaque>","band":"CRITICAL","manifest_id":"TRANSPORT_POSTURE.DEMO
"}
```

Annex D — Threat Catalog (brief) & Mitigations

- **Stamp spoofing / replay** → verify `prev` against local `HEAD`; randomize `nonce`; short-TTL checkpoints.
- **Subset mismatch** → receivers recompute `sha256` over declared bytes; fail with `E_SUBSET_DECL` or `E_BODY_HASH_MISMATCH`.
- **Manifest drift** → manifests immutable per `manifest_id`; rotate with new ID only.

- **Lane divergence** → parity-check kernel when `align` public: `a_c := clamp(a_raw, -1+eps_a, +1-eps_a)` → `u := atanh(a_c)` → `U += w*u ; W += w` → `align := tanh( U / max(W, eps_w) )`.
- **Cache poisoning** → evidence includes `hashes.txt` and `checkpoint.txt`; use transport integrity and cache controls.
- **Privacy bleed** → default **label-first**; keep `align` private unless declared public; exclude PII from canonical subset.

Annex E — Profile Maps (summary)

- **HTTP-M**: headers carry `manifest_id`, disclosure, subset, body hash, stamp; response may add `band`, optional `align`.
- **WS-M**: per-frame envelope + stamp; scope is the stream; first `prev=NONE`.
- **API-M**: metadata block mirrors headers; bind raw body via `SSMNET-Body-Hash`.
- **MESH-M**: peer links exchange envelopes; checkpoint gossip publishes `HEAD=<HEX>`.
- **IOT-M**: compact KV side-band for `manifest_id`, `band`, `stamp`; gateway may append its own stamped observation.

Annex F — Example Manifests (two styles)

Starter (label-first posture)

```
manifest_id: "TRANSPORT_POSTURE.DEMO"
bands:
  - "A++": [-1.00, -0.80]
  - "A0" : (-0.80, +0.60]
  - "CRITICAL": (+0.60, +1.00]
boundary_inclusivity:
  A++: left-inclusive, right-inclusive
  A0:  left-open, right-inclusive
eps_a: 1e-6
eps_w: 1e-9
weight_rule: equal
disclosure: value+band
obligations:
  A++: "monitor"
  A0:  "operate-normally"
  CRITICAL: "human-respond <=10m"
assumptions: "Nominal sampling; valid for load<=X"
```

Enterprise (full parity)

```
manifest_id: "TRANSPORT_POSTURE.ENTERPRISE"
bands:
  - "STABLE": [-1.00, +0.40]
  - "WATCH": (+0.40, +0.70]
  - "CRITICAL": (+0.70, +1.00]
boundary_inclusivity:
  STABLE: left-inclusive, right-inclusive
```

```
  WATCH:  left-open, right-inclusive
eps_a: 1e-6
eps_w: 1e-9
weight_rule: trust-score
disclosure: full
obligations:
  WATCH: "review <=30m"
  CRITICAL: "halt <=5m; human-confirm"
assumptions: "Valid for rpm in [A,B]; temp in [C,D]"
```

Annex G — Operator Runbooks (paste-ready)

Scope genesis

1. Emit first item with `prev=NONE`.
2. Store local `HEAD`.
3. Publish `/ .well-known/ssmnet/checkpoint`.

Incident

1. Quarantine failing item; do not accept into chain.
2. Append stamped error note: `SSMCLOCK1|UTC_ISO|nonce|sha256=...|prev=HEAD`.
3. Correct and resend; new item chains from latest `HEAD`.

Rotation

1. Draft new cutpoints/obligations.
2. Mint new `manifest_id`.
3. Publish manifest; start using new ID; keep old online.

Annex H — Glossary (compact)

- **value** — original bytes / fields; untouched (`phi((m,a)) = m`).
- **align** — bounded dial in (−1,+1) computed by `clamp → atanh → U/W → tanh`.
- **band** — human label from `cutpoint_map(align, manifest_id)`.
- **manifest_id** — pointer to immutable rulebook version.
- **stamp** — `SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEX`.
- **canonical subset** — fields bound by the digest (e.g., `["value","band","manifest_id"(,"align_ascii"?)]`).
- **HEAD** — last accepted hash in a scope.
- **scope** — the sequence being chained (resource, stream, conversation).
- **disclosure** — `value-only` / `value+band` / `full`.
- **parity sentinel** — periodic test proving kernel order-invariance.

One-line takeaway for Section 16

"These annexes are the working kit: precise header grammar, bundles, vectors, threats, profiles, manifests, runbooks, and a glossary—everything needed to reproduce SSM-NET in the real world."

SECTION 17 — EXECUTIVE NOTE & RELEASE STRATEGY

17A. Positioning

SSM-NET is a **manifest-first, stamped overlay** that adds portable meaning to today's internet **without** changing transports, routing, or encryption. It pairs the invariants `phi((m,a)) = m`, bounded lane math `clamp → atanh → U/W → tanh`, rulebook-linked bands, and `SSMCLOCK1|...|prev=...` continuity to make responsibility **verifiable** on the wire.

17B. Why this ships fast

- **Tiny surface area.** A handful of headers/metadata, a manifest, and a stamp line.
- **Deterministic core.** The lane kernel and subset digest are concise and vendor-neutral.
- **Overlay-only.** Legacy stacks ignore unknown fields; payload bytes remain byte-for-byte identical.
- **Proof, not promises.** Evidence bundles and `verify.sh` let anyone print **ALL CHECKS PASSED**.

17C. Where to start (two-track rollout)

Track 1 — Public web endpoints (HTTP-M, label-first)

1. Publish a starter `manifest_id` with explicit boundary text.
2. Add `SSMNET-*` headers on one read-heavy endpoint (GET).
3. Host well-known: `/manifest/<manifest_id>`, `/checkpoint`, `/evidence`.
4. Monitor: digest parity, chain continuity, cache behavior.

Track 2 — Programmatic API (create with evidence)

1. Bind request body via `SSMNET-Body-Hash`.

2. Require stamped submissions; return stamped creations.
3. Store envelopes for later bundle export.

17D. Minimal artifacts to release

- **Spec (this document).** Sections **0–16** are sufficient for a public pilot.
- **Reference manifest(s).** One "starter" (label-first), one "enterprise" (full).
- **Golden diagnostics.** `vectors_lane.csv`, `vectors_cuts.csv`, `verify.sh`, `envelopes.jsonl`, `manifests.json`, `hashes.txt`, `checkpoint.txt`.
- **Demo kit.** `index.html`, sample response headers, `stamp_gen.py` (optional), quick README.

17E. Governance & openness

- **Open, non-exclusive.** Anyone may implement; no registry, no fees.
- **Cite the concept name "Shunyaya Symbolic Mathematical Network (SSM-NET)".**
- **Self-verification.** Conformance = deterministic math + public manifests + stamped replayability.
- **Append-only truth.** Repairs append; history is never rewritten.

17F. Risks & mitigations (executive view)

- **Serializer drift** → Lock canonical subset order; hash raw bytes; document encoders.
- **Policy ambiguity** → Make boundary inclusivity explicit; mint a new `manifest_id` for any change.
- **Privacy overexposure** → Default **label-first**; keep `align` private unless policy requires public parity.
- **Chain forks** → Publish checkpoints; reject non-`HEAD` branches; annotate with stamped incident notes.
- **Adoption friction** → Start with one endpoint; prove no payload mutations; show **ALL CHECKS PASSED**.

17G. Impact outlook

- **For operators:** defensible postures, faster incident root-cause, less debate over "what was known when."
- **For integrators:** fewer bespoke contracts; replayable semantics across vendors.
- **For auditors/regulators:** self-serve evidence; immutable policy lineage via `manifest_id`.

- **For users:** dignity preserved—posture labels describe **content/system state**, not people.

One-line takeaway for Section 17

"Ship a thin, stamped overlay now—prove parity, publish the rulebook, and let anyone replay meaning without touching a single payload byte."

SECTION 18 — DEMO (NOW → AFTER) — HTTP-M QUICKSTART

18A. Goal (one-click proof, 5 minutes)

Show that portable, stamped meaning can be added to any static response **without changing its payload bytes**: `phi((m,a)) = m`.

The **AFTER** state carries:

- a **published rulebook** (`manifest_id`),
- a **canonical-subset digest**,
- a **continuity stamp**,
- and a **band** that is verifiable later **without private trust channels**.

A tiny verifier prints **ALL CHECKS PASSED** directly in the browser.

18B. Minimal demo kit (single-file browser)

What it does

- **Generate & auto-verify:** creates a stamped envelope and verifies it immediately.
- **Paste & verify:** paste any external envelope; verification triggers automatically.
- **Preview raw body bytes:** proves digest is over **exact payload**, not DOM-rendered text.
- **Fetch manifest:** load a rulebook from `/.well-known/ssmnet/manifest/<manifest_id>` and pin the **byte-exact** text used for replay.

Why this matters

- **No install. No server. Works offline.**
- **Payload invariance** remains provable: `phi((m,a)) = m`.
- **Canonical-subset digest** ensures **semantics travel with the bytes**.
- **Continuity stamp** establishes a **replayable timeline**.

What it proves

- **Values are unchanged.**
- **Bands come from published cutpoints** (rulebook, not guesswork).
- **Continuity is replayable**, scope-local, and **append-only**.

18C. NOW (plain fetch)

Serve any static body (e.g., HTML, text, binary).

```
curl -i http://localhost:8080/index.html
```

Observe:

- No `manifest_id`
- No `band`
- No **canonical subset**
- No **digest**
- No **continuity stamp**

Meaning exists only **out-of-band**.

18D. AFTER (HTTP-M overlay, illustrative headers)

1. **Canonical subset (label-first default)**
   `["value","band","manifest_id"]`
2. **Starter manifest (illustrative)**

```
manifest_id: TRANSPORT_POSTURE.DEMO
bands:
  - "A++": [-1.00, -0.80]
  - "A0" : (-0.80, +0.60]
  - "CRITICAL": (+0.60, +1.00]
eps_a: 1e-6
eps_w: 1e-9
disclosure: value+band
obligations:
  A++     : monitor
  A0      : operate-normally
  CRITICAL: human-respond <=10m
```

3. **Body hash (byte-exact)**
   `sha256(body_bytes) -> HEX`
4. **Stamp**
   `SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEX`
   Genesis: `prev=NONE`.
5. **Response snapshot (illustrative)**

```
SSMNET-Manifest: TRANSPORT_POSTURE.DEMO
SSMNET-Disclosure: value+band
SSMNET-Canonical-Subset: ["value","band","manifest_id"]
SSMNET-Body-Hash: sha256=9A2C...77F4
SSMNET-Stamp: SSMCLOCK1|2025-11-
07T12:59:00Z|n17|sha256=F2BC...09AE|prev=NONE
```

Body: unchanged (proves `phi((m,a)) = m`)

18E. Built-in verifier (browser, compact logic)

Executed automatically on **Generate**, **Paste**, or **Fetch**:

1. `require(manifest_id)`
2. `require SUBSET contains ["value","band","manifest_id"]` (order must match declared subset)
3. If `SSMNET-Body-Hash` present:
   `ACTUAL := sha256(body_bytes)` → expect `ACTUAL == BODY_HEX`
4. Normalize textual subset fields to **UTF-8 NFC**; fail if normalization not possible (`no BOM`, newline = `\n`)
5. Match stamp structure: `"SSMCLOCK1|*Z|*|sha256=*|prev=*"`
6. Continuity:
   expect `prev == last_HEAD` **OR** `prev == "NONE"`
   `HEAD := sha256(subset_bytes [+ body_if_declared])`
7. Band check:
   - If **align private** → confirm band is derivable under manifest (cutpoints exist for `band`).
   - If **align public** → parity check:
     `a_c := clamp(a_raw, -1+eps_a, +1-eps_a)`
     `u := atanh(a_c)`
     `U += w*u ; W += w`
     `align := tanh( U / max(W, eps_w) )`
     then confirm `band := cutpoint_map(align, manifest_id)`

Output (illustrative)

```
ALL CHECKS PASSED
band: A0
manifest: TRANSPORT_POSTURE.DEMO
head: <HEX>
```

18F. What changed (and what did not)

Changed	Not changed
A **published rulebook** travels with the bytes.	**Payload bytes** remain byte-exact.
Canonical subset is now **digest-bound**.	Routing, encryption, and app logic remain **untouched**.
Continuity is provable via a **stamped chain**.	`phi((m,a)) = m` holds at every step.

18G. Extending the demo (implementation-ready options)

Goal. Keep the demo compact, verifiable, and copy-paste friendly while showcasing discovery, continuity, and self-service replay.

A. Discovery & Self-Service (recommended)

- **Manifest viewer.** Route `/.well-known/ssmnet/manifest/<manifest_id>` to render the exact rulebook text (read-only, immutable).
- **Checkpoint viewer.** Route `/.well-known/ssmnet/checkpoint` to show `HEAD=<HEX>` and `updated=<UTC_ISO>`.
- **Evidence bundle.** Route `/.well-known/ssmnet/evidence` to serve: `envelopes.jsonl`, `manifests.json`, `hashes.txt`, `checkpoint.txt`, `verify.sh`, plus `bundle.sha256`.

B. Interaction & Continuity (recommended)

- **Chain next packet.** Button to append a new envelope using `prev=last HEAD`; auto-recompute `sha256`.
- **Disclosure toggle.** Cycle `value-only → value+band → full` (public `align` only when declared).
- **Paste & Verify.** Text area accepts an envelope; verifier runs automatically and reports **ALL CHECKS PASSED** or a pinpointed failure.

C. Verification Surfaces (recommended)

- **Body commitment.** URL/file picker to hash exact wire bytes and compare with `SSMNET-Body-Hash` (`sha256=<HEX>`).
- **Stamp parser.** Validate `SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEX`; genesis uses `prev=NONE`.
- **Cutpoint mapping.** If `align` is public, parity-check lane kernel and confirm band mapping under the referenced `manifest_id`.

D. Compact Enhancements (optional, non-breaking)

- **Manifest switcher.** Dropdown to load multiple `manifest_id` entries and compare outcomes on the same payload.
- **Lane sparkline.** Tiny inline plot of successive `align` values to visualize stability transitions.
- **Sharded replay.** Accept multiple envelopes pasted out of order; auto-sort by `prev` to reconstruct the linear chain.
- **Delta highlight.** Brief visual cue when a band flips (cutpoint crossing).
- **Epoch note.** Display a tag when a declared `(U,W)` epoch boundary is crossed.

E. Invariants (must remain true)

- **Payload invariance.** `phi((m,a)) = m` — the original bytes are never mutated.
- **Subset determinism.** Digest target is `sha256( serialize(subset_fields) [+ raw_body_bytes_if_declared] )` with stable order/encoding.
- **Append-only history.** Repairs are new stamped notes; prior envelopes are never edited.

One-line takeaway for Section 18

"Same bytes — now with a rulebook, a digest, and a replayable stamped chain that anyone can verify with one click."

SECTION 19 — PROFILE BINDINGS (WS-M, API-M, MESH-M, IOT-M)

19A. Purpose

Bind SSM-NET semantics to common transports **without changing payload bytes**: `phi((m,a)) = m`. Each profile defines **where** the overlay fields live, **how** the canonical subset is declared, and **what** constitutes a valid stamp: `SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEX`.

19B. WS-M (WebSocket-style streams)

Scope: the **stream** is the chaining unit; first frame uses `prev=NONE`.

Outbound frame (illustrative, side-band block)

```
type: ssmnet.frame
value: <opaque or structured body bytes>
band: "<BandLabel>"                  # label-first default
manifest_id: "<ManifestID>"
subset: ["value","band","manifest_id"(,"align_ascii"?)]
stamp: "SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEX"
# optional when full:
align_ascii: "+0.732000"
```

Receiver rules

- **MUST** recompute `sha256` over declared subset bytes.
- **MUST** verify `prev` against local `HEAD` (per stream).
- **SHOULD** parity-check lane when `align_ascii` is provided using `a_c := clamp(a_raw, -1+eps_a, +1-eps_a)` → `u := atanh(a_c)` → `U += w*u ; W += w` → `align := tanh( U / max(W, eps_w) )`.

Intermediaries

- **MAY** append a **separate** `type: ssmnet.note` with its own `stamp`; never rewrite prior frames.

19C. API-M (programmatic APIs: JSON/Graph-style calls)

Scope. Each request/response pair carries its own SSM-NET envelope. Bodies, when present, **MUST** be bound by a byte-exact digest. Payload invariance holds: `phi((m,a)) = m`.

Request metadata (illustrative)

```
ssmnet_manifest: "<manifest_id>"
ssmnet_disclosure: "value-only" | "value+band" | "full"
ssmnet_subset: ["value","band","manifest_id"(,"align_ascii"?)]
ssmnet_body_hash: "sha256=<HEX>"          # if body present
ssmnet_stamp: "SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEX"
```

Response metadata (mirrors request; optional fields)

```
ssmnet_manifest: "<manifest_id>"
ssmnet_band: "<BandLabel>"                # label-first default
ssmnet_align: "+0.732000"                 # present only when disclosure =
"full"
ssmnet_subset: ["value","band","manifest_id"(,"align_ascii"?)]
ssmnet_body_hash: "sha256=<HEX>"          # if body present
ssmnet_stamp: "SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEX"
```

Server expectations

- **Echo** `manifest_id`, the declared `ssmnet_subset`, the computed digest, and the continuity `ssmnet_stamp`.
- For create/update operations, **return a new stamped envelope** describing the created/updated resource.
- When `ssmnet_body_hash` is present, **compute `sha256` over the exact on-the-wire body bytes** and expose `sha256=<HEX>`.
- **Default to label-first** (`value+band`); publish `align` only when disclosure is `full` and policy permits.

Client expectations

- **Recompute** `sha256` over the declared subset (and body bytes if declared) and compare with the advertised digest.
- **Honor continuity**: accept `prev == "NONE"` for genesis or `prev == <last_HEAD>`; update local `HEAD` to the verified digest.

Notes for canonical subset

- Text fields **MUST** be UTF-8 NFC; newline is `\n`; no BOM.
- If `align_ascii` is included, it **MUST** use fixed sign and precision (e.g., `+0.732000`) consistent with the manifest/profile.

19D. MESH-M (peer/mesh exchanges)

Scope: link or **topic** defines chaining scope; peers gossip checkpoints.

Handshake

- Exchange maximum supported federation level (L1/L2/L3).
- Exchange latest `HEAD=<HEX>` for each shared scope.

Envelope (compact K/V)

```
m: "<ManifestID>"
b: "<BandLabel>"
s: "SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEX>"
k: '["value","band","manifest_id"(,"align_ascii"?)]'  # subset
a: "+0.110000"      # optional (full)
```

Fork resolution

- Accept only chains extending the advertised `HEAD`.
- Divergent branches are quarantined and announced as stamped incidents.

19E. IOT-M (device/gateway pattern)

Constraint. Tiny frames; disclose **label-first** by default.

Device → Gateway (ultra-compact)

```
mid=TRANSPORT_POSTURE.DEMO
band=A0
sub=["value","band","manifest_id"]
stamp=SSMCLOCK1|UTC_ISO|n|sha256=HEX|prev=HEX
```

Gateway duties

- **Preserve bytes.** Keep payload invariant (`phi((m,a)) = m`).
- **Append, never rewrite.** To upgrade disclosure (to `full`), **append** a new stamped note; do not alter device stamps.
- **Host discovery.** Provide well-known endpoints for the fleet:
  - `/.well-known/ssmnet/manifest/<manifest_id>`
  - `/.well-known/ssmnet/checkpoint`
  - `/.well-known/ssmnet/evidence`

Binding notes

- **Canonical subset.** Devices SHOULD declare `["value","band","manifest_id"]`; add `"align_ascii"` only if `full`.
- **Nonce.** Use a crypto-secure nonce; gateway MAY add its own stamped observation with a distinct nonce.
- **Clock skew.** Receivers accept continuity by chain (`prev`), not wall-clock; publish `HEAD` for resync.
- **Manifest parity.** `eps_a`, `eps_w`, and `weight_rule` live in the published `manifest_id` for reproducibility.

19F. Canonical subset & serializer guidance (all profiles)

- **Declare the subset explicitly** (e.g., `["value","band","manifest_id"(,"align_ascii"?)]`).
- **Hash over byte-exact bytes** of the subset serialization (+ raw body bytes if declared).
- **Document field order and encoding** to prevent drift.
- **Never include PII** in the canonical subset.

19G. Disclosure defaults & upgrades

- **Default:** `value+band` (label-first).
- **Upgrade to full** only by policy and consent; expose `align`/`align_ascii` then enable parity checks.

- **Downgrade on failure:** if an implementation cannot parity-check at full, it **MUST** fall back to label-first and log a stamped notice.

19H. Minimal compliance per profile

- **WS-M:** first frame `prev=NONE`; chain every frame; publish `HEAD`.
- **API-M:** bind request/response with `ssmnet_body_hash` when bodies exist.
- **MESH-M:** checkpoint gossip; quarantine forks; stamped notes for anomalies.
- **IOT-M:** gateway hosts well-known; devices remain label-first to save bytes.

One-line takeaway for Section 19

"Same semantics, many roads: WS-M streams, API-M calls, MESH-M peers, and IOT-M devices all carry the rulebook, subset digest, and stamped continuity—without touching payload bytes."

SECTION 20 — ROADMAP & FUTURE WORK

20A. North-star

Make **meaning** as portable as **bytes**: every network exchange carries a verifiable rulebook (`manifest_id`), a bounded posture (`align/band`), and stamped continuity (`SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEX`) — **without** changing payload bytes (`phi((m,a)) = m`).

20B. Immediate milestones (weeks)

- **Spec polish (grammar + examples).** Finalize ABNF for HTTP-M; tighten canonical subset rules; add two more cutpoint examples with explicit inclusivity text.
- **Golden diagnostics v1.** Publish `vectors_lane.csv`, `vectors_cuts.csv`, `verify.sh`, `envelopes.jsonl`, `manifests.json`, `hashes.txt`, `checkpoint.txt` with a clean **ALL CHECKS PASSED**.
- **Public demo kit.** Minimal `/demo/http-m/` with `index.html`, `manifest.json`, `verify.sh`; copy-ready README.

- **Dual manifests.** Ship **starter** (label-first) and **enterprise** (full parity) manifests to illustrate disclosure choices.

20C. Near-term adoption (1–2 quarters)

- **Reference libraries (three languages).** Tiny helpers to:
  `a_c := clamp(a_raw, -1+eps_a, +1-eps_a)` → `u := atanh(a_c)` → `U += w*u ; W += w` → `align := tanh( U / max(W, eps_w) )`,
  compute subset `sha256`, and emit `SSMCLOCK1|...`.
- **Verifier CLIs.** A cross-platform tool that ingests evidence bundles and prints **ALL CHECKS PASSED** or the first failing step.
- **Profile hardening.** WS-M frame schema, API-M metadata block, MESH-M checkpoint gossip, IOT-M compact KV guidance.
- **Operator runbooks.** Day-0/7/30/60/90 checklists templated for quick copy-in.
- **Intermediary pattern.** Mirror/CDN guidelines for append-only notes and `HEAD` relays.

20D. Standardization tracks (parallel)

- **Semantics draft.** Freeze invariants (payload invariance, lane kernel, canonical subset, stamp continuity) as a concise, transport-agnostic core.
- **Profile drafts.** Publish slim profile notes for HTTP-M, WS-M, API-M, MESH-M, IOT-M referencing the core.
- **Test registry (open).** Community-submitted vectors and counter-examples; each addition must ship with reproducer + expected outcome.

20E. Evidence & provenance extensions (optional, non-breaking)

- **Bundle manifest.** A small index that lists included files, versions, and the bundle's own `sha256`.
- **Detached chain notes.** Stamped "observations" that comment on prior entries **without** mutating them (auditor annotations).
- **Cross-scope links.** Lightweight pointers from one scope's `HEAD` to another (e.g., request→result), still append-only.

20F. Privacy & ethics workstream

- **Label-first defaults.** Expand examples where `align` remains private yet posture stays verifiable via `band` and manifest cutpoints.
- **PII avoidance lint.** A simple static checker to ensure canonical subset excludes PII fields.
- **Human obligations.** Template language for manifests that tie bands to clear response windows and escalation promises.

20G. Formal assurance (math + chain)

- **Kernel proofs.** Short notes on order-invariance and shard/merge invariance of `U/W` fusion; numeric tolerances (`≤ 1e-6`, `≤ 1e-12`) stated with rounding discipline.
- **Stamp sanity.** Small model showing that any reorder/edit attempt breaks `prev` continuity and is detectible via checkpoints.

20H. Performance & resilience

Goals. Keep overhead tiny, verification deterministic, and recovery straightforward under imperfect conditions.

Header size budget

- **Target ≤ 1 KB** per message for the full overlay (typical ≤ 512 B with short IDs).
- Prefer short `manifest_id`, compact subset lists, and a 16-byte nonce (`nonce_bytes = 16`) rendered minimally.
- Avoid whitespace drift in lists; canonical subset strings should be tight (e.g., `["value","band","manifest_id"]`).

Digest and body hashing

- Compute `sha256` only over the **declared** bytes: `sha256( serialize(subset_fields) [+ raw_body_bytes_if_declared] )`.
- When binding bodies, hash the **exact wire bytes** (e.g., compressed on transport → hash compressed bytes).
- For large bodies, prefer **no body bind** unless required; when needed, use `byte_ranges` in the canonical subset to cap cost.

Lane math cost

- The kernel `a_c := clamp(a_raw, -1+eps_a, +1-eps_a) → u := atanh(a_c) → U += w*u ; W += w → align := tanh( U / max(W, eps_w) )` is O(1) per sample; batch or stream produce identical `align`.

- Embedded targets MAY use fixed-point if documented; parity tolerance remains ≤ `1e-6`.

Caching & intermediaries

- Caches MUST key on overlay headers so distinct stamps aren't collapsed.
- Stamped responses are cacheable; **do not** strip `SSMNET-Body-Hash` or `SSMNET-Stamp`.
- Intermediaries MAY append new stamps; they MUST NOT rewrite bytes already committed in the upstream subset.

Clock resilience

- Use `UTC_ISO` for human audit; ordering relies on `prev`.
- Under degraded clocks, treat `prev` as source of truth for sequence; publish scope `HEAD=<HEX>` for resync.
- If local time is uncertain, still emit stamps; verification remains deterministic via digest + chain.

Failure containment

- If subset normalization fails, **skip parity** and fall back to continuity verification only.
- If evidence fetch fails, proceed at lower disclosure level (label-first) without blocking payload (`phi((m,a)) = m` holds).
- Quarantine branches that do not chain to current `HEAD` unless a documented fork procedure is active.

Throughput guidance

- Prefer **label-first** (`value+band`) at scale; expose `align` only when needed for public parity.
- Use concise header names, stable ordering, and avoid duplicate fields to minimize parsing overhead.

Epoch rollover & state

- When using (U,W) epochs, stamp a rollover marker; do not reuse prior (U,W).
- Recovery uses `HEAD`, manifests at well-known endpoints, and evidence bundles for replay—no history edits.

Nonce & randomness

- Use CSPRNG for nonces; 16-byte nonces provide ample collision safety for high-throughput chains.
- Nonce scope is per-stamp; do not reuse within the same continuity chain.

Idempotency & retries

- On retry, **recompute** digest and stamp; receivers accept the item that correctly chains to `HEAD`.

- If the same logical event must be deduplicated, carry a separate application-level idempotency key in **payload** (outside SSM-NET semantics).

One-line takeaway

- Keep headers small, bind only what you must, trust `prev` for order, and never rewrite history; verification stays fast, byte-true, and resilient.

20I. Federation ecosystem

- **Level negotiation.** Simple handshake: each side declares max (L1/L2/L3); link runs at `min(local_max, remote_max)`.
- **Cross-manifest comparison.** Examples that compare **posture** across different rulebooks (informational only unless policy states otherwise).
- **Transparency cadence.** Monthly public checkpoints and parity metrics; stamped notes for red-team drills.

20J. Adoption playbook (pragmatic)

- **Start on read endpoints.** Add headers to a single GET; prove "bytes unchanged" and "ALL CHECKS PASSED."
- **Roll to POST/PUT.** Bind bodies via `SSMNET-Body-Hash`; return stamped creations.
- **Publish well-known.** `/.well-known/ssmnet/manifest/<manifest_id>`, `/checkpoint`, `/evidence`.
- **Invite a partner.** Federate at L1 first; advance to L2/L3 after one week of clean parity.

20K. Non-goals (guardrails against scope creep)

- **No new transport, routing, or encryption.** SSM-NET remains an overlay.
- **No identity system.** Bands describe **content/system posture**, not people.
- **No payload rewriting.** `phi((m,a)) = m` is inviolable.

20L. Graduation criteria (exit to "stable")

- **Three independent implementations** pass the Golden Diagnostics end-to-end.
- **Two profiles** (e.g., HTTP-M and WS-M) proven in public demos.
- **Evidence bundles** from at least two independent operators reproduce **ALL CHECKS PASSED** on fresh machines.

- **Manifests** for starter and enterprise styles published with boundary inclusivity and obligations.

One-line takeaway for Section 20

"Freeze the core, prove it in public, and keep privacy first — tiny overlays, deterministic math, and stamped continuity that any independent party can replay."

SECTION 21 — PRACTICAL VISION: COMMUNICATION THAT CARRIES MEANING ACROSS NETWORKS AND BORDERS

"Whether communication happens across cities, ships at sea, remote research stations, or satellites in orbit, SSM-NET makes the **meaning travel with the message**—so state, urgency, and responsibility are understood the same way everywhere."

21A. Premise

Bytes already move reliably; **meaning** often does not.
SSM-NET adds a universal, lightweight semantic layer: **value invariance** (`phi((m,a)) = m`), **bounded posture** (`clamp → atanh → U/W → tanh`), **frozen policy** in a **manifest**, and **stamped continuity** (`SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEX`).
Result: any operator can independently replay **what was known, when, and what was required**, without private coordination.

21B. Universal invariants (work across real-world networks)

- **Value invariance:** original bytes remain byte-for-byte identical (`phi((m,a)) = m`) across gateways, caches, archives.
- **Bounded posture:** `a_c := clamp(a_raw, -1+eps_a, +1-eps_a) → u := atanh(a_c) → U += w*u ; W += w → align := tanh( U / max(W, eps_w) )` ensures stable fusion under batch, streaming, or sharded processing.

- **Declared rulebooks:** `manifest_id` names the band definitions and obligations—no silent reinterpretation after the fact.
- **Stamped continuity:** append-only chains survive outages, high latency, and store-and-forward relays.

21C. Disclosure levels (practical federation)

- **L1 — Label-first:** share `band + manifest_id + stamp`. Proves posture and policy without exposing numeric `align`. Works for public status pages, supplier interfaces, safety dashboards.
- **L2 — Lane parity:** expose `align` (or `align_ascii`) so receivers recompute the kernel and confirm `band := cutpoint_map(align, manifest_id)`. Suited for cross-vendor QA and regulated handoffs.
- **L3 — Full evidence:** publish `envelopes.jsonl`, `manifests.json`, `hashes.txt`, `checkpoint.txt`, `verify.sh` so an independent reviewer can print **ALL CHECKS PASSED** offline.

21D. Manifests as portable rulebooks (like “units of meaning”)

Use compact, language-independent manifests to define **bands** and **obligations** for any signal (service health, environmental telemetry, financial events, robotic state).
Starter lexicon (illustrative):

- **"STABLE"** — operate normally
- **"WATCH"** — increase sampling; human review `<= T_watch`
- **"CRITICAL"** — safe-mode or halt `<= T_crit`; human must acknowledge
  Each band’s timing window and obligations live **inside the manifest**, so expectations travel with the data.

21E. Time across jurisdictions and systems (multi-clock sanity)

- **Canonical anchor:** keep `UTC_ISO` inside the stamp for universal comparability.
- **Local convenience:** carry auxiliary clocks (shift index, mission day, device tick) in the **envelope**, not the stamp.
- **Truth of order:** when clocks disagree, `prev` governs sequence; `UTC_ISO` remains for audit.

21F. Delay, disruption, and federation (what to do in practice)

- **Outages / high latency:** build chains locally; reconcile by exchanging `HEAD` and appending—**never rewriting**.
- **Store-and-forward relays:** intermediaries preserve bytes and stamps; they **may append** their own stamped note, but **must not mutate** upstream evidence.
- **Policy differences:** communicate at the **lower disclosure** level and cite both `manifest_id`s; posture comparisons remain informational unless a shared manifest declares equivalence.

21G. Safety and dignity (default protections)

- **No identity drag:** bands describe **content/system posture**, not people.
- **PII discipline:** canonical subset excludes personal attributes; public signaling defaults to **label-first**.
- **Human/agent obligations:** manifests encode escalation windows and veto rules (e.g., `"CRITICAL" ⇒ human confirm before auto-execute`). The stamped trail proves whether duty was met.

21H. Cross-domain scenarios (near-term, implementable)

- **Public infrastructure:** water/energy services publish `"WATCH"` during stress; third parties can replay exact cutpoints years later.
- **Logistics & ports:** status messages carry `"STABLE"`/`"WATCH"` posture with stamps; disputes resolve by **replay of evidence**, not negotiation.
- **Satellites & remote stations:** label-first broadcasts for public situational awareness; partners pull **L3 evidence bundles** for full reproducibility.
- **Industrial automation:** devices emit `"CRITICAL"` with stamped continuity; site gateways add their own stamped observations; central ops reconstruct an **append-only** truth.

21I. Governance and openness (anti-gatekeeping)

- **Open, non-exclusive semantics.** Anyone may implement SSM-NET; no registry, no fees, no exclusive steward.
- **Cite the concept name** “Shunyaya Symbolic Mathematical Network (SSM-NET)”.
- **Conformance by reproducibility:** deterministic kernel, published manifests, canonical subset digests, and stamped chains.

21J. Practical starter pack (copy-ready today)

- **Baseline bands:** `"STABLE"`, `"WATCH"`, `"CRITICAL"` with explicit boundary inclusivity.
- **Timing fields:** `T_watch`, `T_crit` in the manifest; obligations as short verbs (`"review <=30m"`, `"safe-mode <=5m"`).
- **Profiles:** begin with **HTTP-M** (label-first) for public endpoints; add streaming profile for telemetry; use **evidence bundles** for audits and cross-org handoffs.
- **Tolerance discipline:** adopt `≤ 1e-6` (batch vs stream) and `≤ 1e-12` (shard/merge) as numeric expectations across teams.
- **Discovery:** publish manifests at `/.well-known/ssmnet/manifest/<manifest_id>`, checkpoints at `/.well-known/ssmnet/checkpoint`, evidence at `/.well-known/ssmnet/evidence`.
- **Intermediaries:** document pass-through guarantees; if adding notes, **append** a new stamp, never edit history.

21K. Relationship to SSMT and SSMDE (how this plugs in)

- **SSMT** universalizes **measurement** (e.g., temperature) with explicit semantics.
- **SSMDE** universalizes **data exchange responsibility** (`value • align • band • manifest_id • stamp`).
- **SSM-NET** universalizes **communication meaning on the wire** (headers/metadata + well-known endpoints) so responsibility **travels with the bytes** end-to-end.

One-line takeaway for Section 21

"SSM-NET turns everyday communication into declared, stamped meaning—so any independent party can replay the same truth, posture, and obligations without touching a single payload byte."

SECTION 22 — CONCLUSION

What SSM-NET delivers

- **Portable meaning without payload edits.** Original bytes remain byte-for-byte identical: `phi((m,a)) = m`.
- **Bounded, reproducible posture.** One kernel defines the lane everywhere: `a_c := clamp(a_raw, -1+eps_a, +1-eps_a) → u := atanh(a_c) → U += w*u ; W += w → align := tanh( U / max(W, eps_w) )`.

- **Declared rulebooks.** Bands come from **published cutpoints** linked by `manifest_id`, with **boundary inclusivity** and **human/agent obligations** spelled out.
- **Stamped continuity.** Exchanges carry `SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEX`, enabling **replayable time, order, and subset integrity**.
- **Deterministic commitments.** The **canonical subset** is byte-stable (UTF-8 **NFC**, fixed decimals for `align_ascii`), so independent tools recompute the same `sha256`.
- **Fetch & discovery. Well-known endpoints** expose manifests, `HEAD`, and evidence for **self-service verification**.
- **Practical federation.** Levels **L1/L2/L3** align disclosure to need: label-first → numeric parity → full offline replay.

What it never does

- **Not a new transport.** Rides alongside web, API, stream, mesh, or device patterns.
- **No content mutation.** Overlays sit **beside** payloads; bytes are not touched.
- **No identity layer.** Bands describe **content/system posture**, not people.

Why this matters now

- **Interoperability with proof.** Different vendors can disagree on policy yet still exchange **verifiable posture and continuity**.
- **Audit without ceremony.** Evidence bundles make independent verification **routine**, not an escalation ritual.
- **Privacy-first posture. Label-first** (`value+band`) satisfies most use cases; **full lane disclosure** is **opt-in**.
- **Operational realism.** Works with **outages, high latency, and intermediaries** (append-only stamps; never rewrite). Optional `(U,W)` **epoch rollover** scales long streams without changing math.
- **Near-term adoption.** Deterministic text norms (UTF-8 NFC), fixed decimals, and well-known URLs keep implementation small and **tool-friendly**.

How to begin (today)

- **Publish** a starter **manifest** with explicit cutpoints, obligations, `cmp_tolerance`, and `text_norm`.
- **Add** HTTP-M overlay headers to **one** read endpoint: declare `manifest_id`, **canonical subset**, **body hash**, and **stamp**.
- **Host** well-known paths for **manifest**, **checkpoint (HEAD)**, and **evidence**.
- **Run** the **Golden Diagnostics** (Section 12); expect **ALL CHECKS PASSED** on a clean machine.
- **Invite** a partner at **L1**; **upgrade** to **L2/L3** by mutual consent and policy.

Promise to the operator and the public

This standard exists to **protect people doing real work**. **Bounded math** prevents distortion; **manifests** prevent quiet policy drift; **stamps** prevent timeline edits. If a **band implies action**, that obligation is **clear, time-anchored, and replayable**—on any network, across organizations, and across time.

One-line takeaway

"Keep your bytes; add verifiable meaning — a bounded dial, a named rulebook, and an append-only stamp — so responsibility can travel anywhere on the internet and still be proven later."

Appendix A — Header Grammar (ABNF-style, illustrative)

Scope. This appendix defines a compact, interoperable grammar for SSM-NET wire fields and well-known artifacts. It is profile-agnostic, with concrete HTTP-M mappings included for convenience. All productions are illustrative and may be extended without breaking earlier definitions **so long as the canonical subset contract remains intact**.

A.1 Notation and Conventions

- **Syntax style.** ABNF-like (case-insensitive tokens unless explicitly quoted).
- **Byte truth.** Hashes always bind byte-exact serializations of the declared canonical subset and, when declared, raw body bytes.
- **Newlines.** Newline is `\n` (LF). No BOM.
- **Decimals.** Canonical decimal strings use fixed precision where specified; numeric lane is bounded by the kernel and, when public, may be mirrored as `align_ascii`.
- **Safety constants.** Keep `eps_a > 0`, `eps_w > 0`. Formulas appear inline in ASCII, e.g., `align := tanh( U / max(W, eps_w) ).`
- **Escaping.** JSON strings inside subset serializations **MUST** follow standard JSON escaping; no additional presentation whitespace is permitted inside the serialized bytes beyond what the deterministic serializer emits.

A.2 Common Tokens

```
ALPHA        = %x41-5A / %x61-7A                  ; A-Z / a-z
DIGIT        = %x30-39                             ; 0-9
HEXDIG       = DIGIT / "A"-"F" / "a"-"f"
WSP          = SP / HTAB
```

```
VCHAR        = %x21-7E
token        = 1*( ALPHA / DIGIT / "-" / "_" / "." )
manifest-id  = token                                  ; e.g.,
TRANSPORT_POSTURE.DEMO
nonce        = 8*64( ALPHA / DIGIT / "-" / "_" )    ; URL-safe ASCII,
length-enforced
nonce-b64url = 11*86( ALPHA / DIGIT / "-" / "_" )   ; optional profile:
base64url-encoded, length post-decode 8-64
utc-iso      = 4DIGIT "-" 2DIGIT "-" 2DIGIT "T" 2DIGIT ":" 2DIGIT ":"
2DIGIT ["." 3DIGIT] "Z"
hex          = 1*HEXDIG
sha256-hex   = 64HEXDIG
scope-name   = token                                  ; e.g., default,
orders, feed-alpha
band-label   = 1*VCHAR                                ; printable, no control
chars
float-dec    = ["-"/"+"] 1*DIGIT ["." 1*DIGIT]      ; non-canonical free-
form float
align-ascii  = ("+" / "-") 1DIGIT "." 6DIGIT        ; fixed sign + 6
decimals (e.g., +0.732000)
subset-field = DQUOTE token DQUOTE
subset-list  = "[" subset-field *( "," subset-field ) [ "," DQUOTE
"align_ascii" DQUOTE ] "]"
```

Note (nonce profiles): A manifest profile **MAY** permit `nonce-b64url`; if enabled, receivers **MUST** base64url-decode and validate decoded length `8..64` and URL-safe alphabet on the wire.

A.3 Continuity Stamp (One-Line Anchor)

```
stamp-line = "SSMCLOCK1" "|" utc-iso "|" "nonce=" (nonce / nonce-b64url)
             "|" "sha256=" sha256-hex "|" "prev=" prev-ref
prev-ref   = "NONE" / "nil" / sha256-hex
```

- **Genesis rule.** First item in a scope uses `prev=NONE` (demo) or `prev=nil` (strict), per manifest policy; receivers **MUST** accept either.
- **Append-only.** Repairs are chained as new `stamp-line` entries; history is never edited.

A.4 HTTP-M Request Headers (Client → Server)

```
SSMNET-Manifest          = "SSMNET-Manifest:" WSP manifest-id
SSMNET-Disclosure        = "SSMNET-Disclosure:" WSP disclosure-mode
disclosure-mode          = "value-only" / "value+band" / "full"

SSMNET-Canonical-Subset  = "SSMNET-Canonical-Subset:" WSP subset-list

SSMNET-Body-Hash         = "SSMNET-Body-Hash:" WSP "sha256=" sha256-hex
                           ; present only when a request body exists and is
committed

SSMNET-Stamp             = "SSMNET-Stamp:" WSP stamp-line
```

- **Namespace reservation.** The JSON-RPC sidecar key `_ssmnet` is **reserved** for SSM-NET overlays; if collision exists, use `_ssmnet_v2` (overlay remains optional).
- **Transport hygiene.** For sensitive declarations, HTTP-like transports **SHOULD** use a secure channel.

A.5 HTTP-M Response Headers (Server → Client)

```
SSMNET-Manifest          = "SSMNET-Manifest:" WSP manifest-id
SSMNET-Band              = "SSMNET-Band:" WSP band-label         ; optional
(label-first default)
SSMNET-Align             = "SSMNET-Align:" WSP float-dec          ;
optional (only if public)
SSMNET-Align-Ascii       = "SSMNET-Align-Ascii:" WSP align-ascii  ;
optional mirror, if public

SSMNET-Canonical-Subset  = "SSMNET-Canonical-Subset:" WSP subset-list
SSMNET-Body-Hash         = "SSMNET-Body-Hash:" WSP "sha256=" sha256-hex
SSMNET-Stamp             = "SSMNET-Stamp:" WSP stamp-line
SSMNET-Checkpoint        = "SSMNET-Checkpoint:" WSP "HEAD=" sha256-hex
                           ; optional per-scope advert for quick parity
```

- **Invariant.** Payload bytes remain byte-for-byte unchanged: `phi((m,a)) = m`.
- **Privacy default.** Prefer `value+band`; expose `SSMNET-Align`/`SSMNET-Align-Ascii` only when disclosure is full.

A.6 Canonical Subset Declaration & Serialization

- **Declaration (on the wire).** Example: `["value","band","manifest_id"]` or `["value","band","manifest_id","align_ascii"]`.
- **Digest target.** `sha256( serialize(subset_fields) [+ raw_body_bytes_if_declared] )`.
- **Serialization rules (normative, minimal):**
  1. **Order is fixed** by the `subset-list`.
  2. **Encoding is byte-stable** (same key order, same quoting, **no extra whitespace** beyond what the deterministic serializer emits).
  3. **Outside the subset** (other headers, presentation whitespace, non-declared fields) **MUST NOT** affect verification.

A.7 Decimal Canonical Form for Public Lane

- **When public**, include `SSMNET-Align-Ascii: align-ascii` as a **fixed canonical mirror** of the float (e.g., `+0.732000`).
- **Computation (deterministic kernel).**
  `a_c := clamp(a_raw, -1+eps_a, +1-eps_a) -> u := atanh(a_c) -> U += w*u ; W += w -> align := tanh( U / max(W, eps_w) )`
- **Tolerance guidance.** Batch vs. stream parity `<= 1e-6`; shard-merge parity `<= 1e-12`.

A.8 Well-Known Endpoints (Discovery Surfaces)

```
Manifest (by ID)
GET /.well-known/ssmnet/manifest/<manifest_id>
  -> immutable manifest text including band cuts, boundary inclusivity,
eps_a, eps_w, weight_rule

Checkpoint (scope HEAD)
GET /.well-known/ssmnet/checkpoint[?scope=<scope-name>]
  -> "HEAD=" sha256-hex [ CRLF "scope=" scope-name ] [ CRLF "updated=" utc-
iso ]

Evidence Bundle (self-service audit)
GET /.well-known/ssmnet/evidence[?scope=<scope-name>&from=<utc-
iso>&to=<utc-iso>]
  -> zipped/compact directory (see A.9)
```

A.9 Evidence Bundle Layout (Minimal Set)

```
/evidence/
  manifests.json     ; includes all referenced manifest texts
  envelopes.jsonl    ; one canonical-subset object per line in declared
field order
  hashes.txt         ; "sha256=<HEX>" per line matching each envelope order
  checkpoint.txt     ; "HEAD=<HEX>" [CRLF "scope=<name>"] [CRLF
"updated=<UTC_ISO>"]
  verify.sh          ; recompute subset digests, follow prev-chain, check
bands; prints ALL CHECKS PASSED
```

A.10 Error Model Headers (Overlay-Safe Signaling)

```
SSMNET-Error       = "SSMNET-Error:" WSP error-code
error-code         = "E_POLICY_MISMATCH" / "E_BODY_HASH_MISMATCH" /
"E_STAMP_PREV"
                   / "E_MANIFEST_MISS" / "E_ALIGN_POLICY" / "E_SUBSET_DECL"
                   / "E_DISCLOSURE" / "E_CHECKPOINT_DRIFT" / "E_FORMAT" /
token

SSMNET-Error-Stamp = "SSMNET-Stamp:" WSP stamp-line
                     ; stamped incident note chaining from current HEAD
```

- **Do not mutate history.** Errors append stamped facts; payload bytes and prior stamps remain intact.

A.11 Robustness and Ignorability Rules

- **Forward-compatibility.** Unknown `SSMNET-*` headers **MAY** be ignored by legacy stacks without affecting payload delivery.
- **Subset integrity.** Receivers **MUST** recompute `sha256` over the declared subset and, if present, raw body bytes; mismatches **MUST** fail verification deterministically.

- **Intermediaries. MUST** forward payload bytes unchanged; **SHOULD** preserve original subset and digest; **MAY** append their own stamped observation; **MUST NOT** rewrite upstream stamps.

A.12 Non-Normative Examples

A.12.1 Minimal label-first response (align private)

```
SSMNET-Manifest: TRANSPORT_POSTURE.DEMO
SSMNET-Band: A0
SSMNET-Canonical-Subset: ["value","band","manifest_id"]
SSMNET-Body-Hash: sha256=9A2C...77F4
SSMNET-Stamp: SSMCLOCK1|2025-11-
10T12:59:00Z|n17|sha256=F2BC...09AE|prev=NONE
```

A.12.2 Full disclosure with canonical lane mirror

```
SSMNET-Manifest: TRANSPORT_POSTURE.DEMO
SSMNET-Band: CRITICAL
SSMNET-Align: 0.732
SSMNET-Align-Ascii: +0.732000
SSMNET-Canonical-Subset: ["value","band","manifest_id","align_ascii"]
SSMNET-Body-Hash: sha256=4C10...AA21
SSMNET-Stamp: SSMCLOCK1|2025-11-
10T13:10:22Z|n21|sha256=BBEE...019C|prev=F2BC...09AE
```

A.13 Quick Math Reminders (for implementers)

- **Kernel (order-invariant fusion).**
  `a_c := clamp(a_raw, -1+eps_a, +1-eps_a) -> u := atanh(a_c) -> U += w*u ; W += w -> align := tanh( U / max(W, eps_w) )`
- **Collapse parity (payload invariance).** `phi((m,a)) = m`.
- **Band mapping.** `band := cutpoint_map(align, manifest_id)` with explicit boundary inclusivity defined in the manifest.

Appendix B — Evidence Bundle Layout (minimal, reproducible)

*(Purpose: enable any independent party to replay meaning and print **ALL CHECKS PASSED** offline.)*

B.1 Bundle contents (minimum set)

```
/evidence/
  manifests.json        ; all referenced manifests (immutable text)
  envelopes.jsonl       ; one canonical-subset object per line
  hashes.txt            ; sha256 over each declared subset [+ body if
declared]
  checkpoint.txt        ; "HEAD=<HEX>" (64 hex chars) for the scope
  verify.sh             ; tiny offline verifier (no network required)
```

B.2 File formats

manifests.json (array or map; immutable entries)
• Each entry contains:
o `manifest_id`
o `bands` with explicit intervals and boundary inclusivity
o `eps_a`, `eps_w`, and optional `weight_rule`
o `disclosure` (`value-only` / `value+band` / `full`)
o optional `assumptions` / notes
• **Rotation rule:** any change mints a new `manifest_id`; older entries remain.

envelopes.jsonl (newline-delimited JSON; each line is the declared subset only)
• Typical label-first line:
• `{"value": "...", "band": "A0", "manifest_id": "TRANSPORT_POSTURE.DEMO"}`
• If disclosure is full, include canonical decimal:
• `{"value": "...", "band": "A0", "manifest_id": "TRANSPORT_POSTURE.DEMO", "align_ascii": "+0.732000"}`

hashes.txt (one line per envelope; strict order)
`sha256=<HEX> index=<N>`
• `index` is zero-based position corresponding to the line number in `envelopes.jsonl`.
• The hex **MUST** be computed over the byte-exact serialization of the subset fields (and raw body bytes if declared).
• `<HEX>` is 64 hexadecimal characters (sha256-hex).

checkpoint.txt

```
HEAD=<HEX>
updated=<UTC_ISO>
scope=<name>     ; optional, e.g., "default"
```

verify.sh (POSIX-style; pure shell + common tools)

```
#!/usr/bin/env sh
set -eu

fail() { echo "FAIL: $1" >&2; exit 2; }
ok()   { echo "OK:  $1"; }

# 1) Load HEAD
HEAD_LINE=$(grep '^HEAD=' checkpoint.txt || true)
[ -n "$HEAD_LINE" ] || fail "missing HEAD in checkpoint.txt"
HEAD_EXPECTED=${HEAD_LINE#HEAD=}
```

```
# 2) Walk envelopes and hashes in lockstep
i=0
PREV="NONE"
while IFS= read -r ENV; do
  # subset bytes as-is
  printf "%s" "$ENV" > .subset.tmp

  # hash check
  HASH_LINE=$(sed -n "$((i+1))p" hashes.txt)
  HASH_EXPECTED=$(printf "%s" "$HASH_LINE" | awk -F'sha256=' '{print $2}' |
awk '{print $1}')
  HASH_ACTUAL=$(openssl dgst -sha256 .subset.tmp | awk '{print $2}')
  [ "$HASH_ACTUAL" = "$HASH_EXPECTED" ] || fail "digest mismatch at
index=$i"

  # rolling head (mirrors last subset digest when body not declared)
  HEAD_COMPUTED="$HASH_ACTUAL"

  PREV="$HEAD_COMPUTED"
  i=$((i+1))
done < envelopes.jsonl

[ "$PREV" = "$HEAD_EXPECTED" ] || fail "HEAD mismatch (computed $PREV !=
expected $HEAD_EXPECTED)"

ok "ALL CHECKS PASSED"
```

• No network access required.
• If bodies are declared in the stamp contract, include a parallel `body/` folder and hash `subset||body` bytes in the same order; keep the script minimal but deterministic.

B.3 Integrity rules (normative)

• **Subset determinism:** the JSON in each `envelopes.jsonl` line is the exact byte sequence used for hashing; pretty-printing elsewhere **MUST NOT** affect results.
• **Order preservation:** `hashes.txt` entries **MUST** correspond 1:1 to `envelopes.jsonl` lines.
• **Continuity:** `checkpoint.txt HEAD` **MUST** equal the last verified `sha256` in sequence; history is append-only (prev logic in transport; bundle reflects the final `HEAD`).
• **Payload invariance:** when the body is bound, it is the raw entity bytes; overlay **MUST NOT** alter payload bytes (`phi((m,a)) = m`).

B.4 Privacy posture (recommended defaults)

• Default label-first bundles: include `band`, exclude `align_ascii` unless disclosure is `full`.
• No PII in canonical subset. If needed for business logic, keep such fields outside the subset and ensure the digest still binds operational meaning (`value` / `band` / `manifest_id`).
• **Retention clarity:** include a short `retention.txt` note (optional) describing rotation and deletion policies that do not touch published manifests or checkpoints.

B.5 Negative tests (should fail clearly)

• **Digest mismatch:** flip one byte in an `envelopes.jsonl` line → verifier prints **FAIL**.
• **HEAD drift:** change `checkpoint.txt HEAD` to a non-matching value → **FAIL**.
• **Subset discrepancy:** remove a declared field from an envelope line → **FAIL**.

B.6 Example minimal bundle tree (illustrative)

```
/evidence/
  manifests.json            (immutable)
  envelopes.jsonl           (N lines; subset only)
  hashes.txt                (N lines; sha256=index mapped)
  checkpoint.txt            (HEAD=<HEX>)
  verify.sh                 (offline checker)
  body/                     (optional, only if body binding was declared)
    0000.bin
    0001.bin
```

One-line takeaway (Appendix B)

A tiny, self-contained folder—subset lines, their hashes, the `HEAD`, and a ~20-line script—is enough for anyone to replay meaning and print **ALL CHECKS PASSED**.

Appendix C — Conformance & Quick Reference (Profiles • Roles • Levels)

(Purpose: a copy-paste checklist to implement SSM-NET correctly on day one. Transport-agnostic; HTTP-M examples are illustrative.)

C.1 Conformance roles (what each MUST do)

Senders (MUST)

- **Publish or reference** a `manifest_id`.
- **Declare** the **canonical subset** on the wire (example: `["value","band","manifest_id"]`).
- **Compute and expose** the digest bound to that subset (and body if declared): `sha256( serialize(subset_fields) [+ raw_body_bytes_if_declared] )`.
- **Emit a continuity stamp:** `SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEX` (genesis `prev=NONE`).

- **Preserve payload bytes** unchanged (`phi((m,a)) = m`).
- **Honour manifest rotation:** any policy change → **new** `manifest_id`.

Receivers (MUST)

- **Fetch manifest** by `manifest_id` and pin byte-exact text.
- **Recompute digest** over the **declared** canonical subset (and raw body if declared) and compare `sha256`.
- **Verify chain continuity** (`prev` equals prior accepted `sha256`; last equals `HEAD` if published).
- **Confirm derivability** of `band` under the manifest; if `align` is public, **parity-check** via:
  `a_c := clamp(a_raw, -1+eps_a, +1-eps_a) → u := atanh(a_c) → U += w*u ; W += w → align := tanh( U / max(W, eps_w) ).`

Intermediaries (SHOULD / MUST)

- **MUST** forward payload bytes **untouched**.
- **SHOULD** preserve upstream subset and `SSMNET-Body-Hash`.
- **MAY** append their own **new** stamped observation; **MUST NOT** rewrite upstream stamps.

C.2 Federation levels (what crosses the link)

Item	L1 (Label-first)	L2 (Lanes reproducible)	L3 (Full evidence)
value (bytes)	yes	yes	yes
band, manifest_id	yes	yes	yes
align / align_ascii	no	yes	yes
stamp `SSMCLOCK1	UTC_ISO	nonce	sha256=HEX
canonical subset declaration	yes	yes	yes
evidence bundle (pull)	optional	optional	required

- **Negotiate:** operate at `min(local_max, remote_max)`.
- **Fallbacks:** if `align` parity fails at L2 → behave as **L1** (non-fatal notice). If evidence fetch fails at L3 → continue at **L2/L1**.

C.3 Canonical subset & text norms (digest will pass if and only if…)

- **Declare** the subset list **in order** (example: `["value","band","manifest_id"(,"align_ascii"?)]`).
- **Serialize** exactly in that order; no hidden fields; no extras.
- **Text normalization:** all textual bytes used in hashing **MUST** be UTF-8 in **NFC**; newline is `\n`; no BOM.

- **Decimals:** any public lane mirror uses fixed canonical form `align_ascii` (example `+0.732000`).
- **Body binding:** if body is declared, hash the **exact wire bytes** (post content-encoding).

C.4 Lane kernel quick rules (determinism)

- **Clamp first:** `a_c := clamp(a_raw, -1+eps_a, +1-eps_a).`
- **Rapidity:** `u := atanh(a_c).`
- **Accumulate:** `U += w*u ; W += w.`
- **Fuse:** `align := tanh( U / max(W, eps_w) ).`
- **Safeguards:** `eps_a > 0`, `eps_w > 0`.
- **Order-invariance targets:** batch vs stream ≤ `1e-6`; shard-merge ≤ `1e-12`.
- **Shard-merge:** sum `(U_i, W_i)` only; do **not** re-clamp at joins.

C.5 Manifest essentials (must contain)

- `manifest_id` (stable identifier).
- **Bands** with explicit numeric intervals **and** boundary inclusivity text.
- `eps_a`, `eps_w`, optional `weight_rule`, optional `cmp_tolerance`, optional `assumptions`.
- **Disclosure mode:** `value-only` / `value+band` / `full`.
- **Rotation rule:** any change → **new** `manifest_id` (old remains online).

C.6 Well-known endpoints (read-only discovery)

- **Manifest:** `/.well-known/ssmnet/manifest/<manifest_id>` (immutable text; cacheable; optional embedded `sha256=` of manifest body).
- **Checkpoint (HEAD):** `/.well-known/ssmnet/checkpoint[?scope=<name>]` → `HEAD=<HEX>` (short-TTL).
- **Evidence:** `/.well-known/ssmnet/evidence[?scope=&from=&to=]` → deterministic archive with `envelopes.jsonl`, `manifests.json`, `hashes.txt`, `checkpoint.txt`, `verify.sh`, `bundle.sha256`.

C.7 Minimal HTTP-M mapping (illustrative)

- `SSMNET-Manifest: <manifest_id>`
- `SSMNET-Disclosure: value-only | value+band | full`
- `SSMNET-Canonical-Subset: ["value","band","manifest_id"(,"align_ascii"?)]`
- `SSMNET-Body-Hash: sha256=<HEX>` *(if body is bound)*
- `SSMNET-Stamp: SSMCLOCK1|UTC_ISO|nonce|sha256=<HEX>|prev=<HEX|NONE>`

- Optional (response): `SSMNET-Band: <label>`; if full: `SSMNET-Align: <float>` and `SSMNET-Align-Ascii: <+/-d.dddddd>`

Invariant: payload bytes remain **byte-exact** (`phi((m,a)) = m`).

C.8 Golden Diagnostics (pass/fail gist)

- **Kernel determinism:** batch = stream = shard-merge (within tolerances).
- **Boundary accuracy:** edges respect manifest inclusivity text.
- **Shard/merge invariance:** `(U,W)` sums match monolith.
- **Stamp continuity:** linear chain; last equals `HEAD`.
- **Subset digest parity:** `sha256` matches independently.
- **Profile conformance:** hash **wire bytes**, not decoded text.
- **Privacy posture:** label-first hides `align`; full exposes with `align_ascii`.

C.9 Error codes (overlay-safe signalling)

- `E_POLICY_MISMATCH` — manifest published but local policy forbids link.
- `E_BODY_HASH_MISMATCH` — digest over subset/body fails.
- `E_STAMP_PREV` — `prev` does not match prior accepted `sha256`.
- `E_MANIFEST_MISS` — unknown `manifest_id` (not fetchable).
- `E_ALIGN_POLICY` — lane parity fails under disclosed kernel parameters.
- `E_SUBSET_DECL` — declared subset malformed/missing/extra fields.
- `E_TEXT_NORM` — subset text not UTF-8 NFC.
- `E_CHECKPOINT_DRIFT` — advertised `HEAD` != recomputed last digest.
- `E_FORMAT` — header/field structure invalid.
  *(Errors are appended as **new** stamped notes; history is never edited.)*

C.10 Implementation checklist (one page, copy-paste)

- **[ ]** Choose `manifest_id`; publish manifest with bands, inclusivity, `eps_a`, `eps_w`, `weight_rule`, disclosure.
- **[ ]** Emit headers: `SSMNET-Manifest`, `SSMNET-Canonical-Subset`, optional `SSMNET-Body-Hash`, and `SSMNET-Stamp`.
- **[ ]** Ensure canonical subset bytes are **UTF-8 NFC**, ordered, and deterministic.
- **[ ]** Keep payload bytes unchanged (`phi((m,a)) = m`).
- **[ ]** Host `/.well-known/ssmnet/manifest`, `/checkpoint`, `/evidence`.
- **[ ]** Run Golden Diagnostics; target tolerances met.
- **[ ]** Declare max federation level (L1/L2/L3); allow **min** at runtime.
- **[ ]** Document intermediary behaviour: pass-through, append-only stamps.
- **[ ]** Set retention/rotation (manifests immutable; new `manifest_id` for policy change).
- **[ ]** Include a tiny offline `verify.sh` with an evidence bundle for partners.

One-line takeaway (Appendix C)

"Pin the manifest, declare the subset, stamp the chain, and never touch the bytes — if your tool can recompute the same sha256 and follow `prev`, you're conformant."

Appendix D — Deterministic Serialization & Hashing (Text • Numbers • Ordering)

(Purpose: make independent implementations hash the same bytes every time. This appendix defines byte-level rules for the canonical subset, including normalization, decimals, escaping, ordering, and failure cases.)

D.1 Scope & intent

• **What this appendix governs:** how to turn declared fields into exact bytes for `sha256( … )` in stamps and evidence.
• **Goal:** if two tools read the same envelope and manifest, they produce the same digest, regardless of language, platform, or locale.
• **Non-goal:** this appendix does not require any particular storage format outside the declared subset used for hashing.

D.2 Canonical text rules (normative)

• **Encoding:** UTF-8, no BOM.
• **Normalization:** NFC (Normalization Form C) for all textual bytes inside the declared subset.
• **Newlines:** `\n` (LF) only.
• **Whitespace:** no leading/trailing spaces around separators inside the serialized bytes beyond what is explicitly defined by the chosen profile's canonical form.
• **Line-trim (normative helper):** `TRIM_WHITESPACE_PER_LINE` removes only ASCII spaces (`0x20`) and horizontal tabs (`0x09`) from the start and end of each logical line in profile-defined contexts; it does **not** collapse internal spaces and never alters bytes inside quoted/escaped regions.
• **Quoting:** one escape scheme per profile; no alternates. If JSON is used, strings MUST follow JSON escaping rules; if a profile uses another syntax, it must define an equivalent single scheme.
• **Case:** field names and tokens are case-sensitive as emitted in the subset; manifests may define case rules for band labels if required.

Failure mapping
• Non-UTF-8 or non-NFC inside subset → `E_TEXT_NORM`.
• Extra spaces or alternate escapes inside the subset → `E_SUBSET_DECL`.

D.3 Canonical number rules (normative)

• **Public lane mirror:** if the numeric lane is public, expose a fixed-width decimal mirror as `align_ascii` with **sign + 6 decimals** (example `+0.732000`).
• **Integers & counters (if present):** emit as minimal decimal digits (no `+`, no leading zeros, no grouping).
• **General floats (if present):** manifests SHOULD avoid hashing free-form floats; prefer fixed canonical strings (e.g., thresholds mirrored as strings).
• **Locale:** no thousands separators, no locale-specific decimal marks.

Failure mapping
• Variable-decimal align without `align_ascii` in the subset → `E_SUBSET_DECL`.
• Locale-formatted numbers → `E_FORMAT`.

D.4 Canonical subset declaration (normative)

• **Order is law:** the subset list is an ordered contract (example: `["value","band","manifest_id"(,"align_ascii"?)]`).
• **Hash target:** `sha256( serialize(subset_fields) [+ raw_body_bytes_if_declared] ).`
• **No extras, no omissions:** include only the declared fields, all of them, and in order.
• **Deterministic serializer:** given the same field order and values (already normalized), the serializer MUST produce the same bytes everywhere.

Failure mapping
• Missing or extra field, or wrong order → `E_SUBSET_DECL`.

D.5 Strings & escaping (normative)

• **Single scheme:** the profile defines one escape scheme; implementations MUST NOT accept alternates inside the subset.
• **Stable escapes:** the same character always escapes to the same byte sequence.
• **No pretty variants:** avoid tabs or multi-space padding inside subset bytes; represent spacing only as defined.

Failure mapping
• Mixed escape styles or pretty variants → `E_FORMAT`.

D.6 Bodies bound to the digest (normative)

• **Wire bytes, not decoded text:** if a body is declared, hash the **exact raw body bytes** as carried on the transport (after any content encoding the wire actually used).

• **Header binding:** when a body is bound, expose `SSMNET-Body-Hash: sha256=<HEX>` alongside the subset declaration.
• **Binary allowed:** any bytes are acceptable; hashing is byte-agnostic.

Failure mapping
• Hashing decoded or re-encoded content instead of wire bytes → `E_BODY_HASH_MISMATCH`.

D.7 Numeric tolerances (reference, not part of hashing)

• **Lane fusion parity targets:** batch vs stream ≤ `1e-6`; shard-merge ≤ `1e-12`.
• **Kernel order:** `a_c := clamp(a_raw, -1+eps_a, +1-eps_a) → u := atanh(a_c) → U += w*u ; W += w → align := tanh( U / max(W, eps_w) ).`
• **Note:** tolerances apply to computed `align`, not to the hashed `align_ascii` (which is fixed-format text).

D.8 Negative fixtures (must fail clearly)

• **Alternate normalization:** same visible text in NFD vs NFC → fail with `E_TEXT_NORM`.
• **Subset spacing drift:** insert a space after a comma inside the subset bytes when the canonical form forbids it → fail with `E_SUBSET_DECL`.
• **Float drift:** emit `0.732` instead of `+0.732000` for `align_ascii` → fail with `E_SUBSET_DECL`.
• **Body decode error:** hash decompressed bytes when the wire carried compressed → fail with `E_BODY_HASH_MISMATCH`.

D.9 Minimal algorithm (reference pseudocode)

```
# Inputs: ordered subset fields F[0..k-1], map V[field]->string/bytes,
optional raw_body_bytes
# Output: sha256_hex

# 1) For each textual V[field], enforce UTF-8 NFC; for bytes, pass through.
# 2) Serialize exactly in subset order using the profile's canonical rules.
#    (Apply TRIM_WHITESPACE_PER_LINE only where the profile specifies.)
# 3) If body is declared, append raw_body_bytes exactly as sent on the
wire.
# 4) Compute sha256 over the resulting byte sequence.

sha256_hex := SHA256( canonical_bytes [+ raw_body_bytes_if_declared] )
```

D.10 Interoperability checklist (copy-paste)

• [ ] UTF-8, no BOM, NFC.
• [ ] Newlines are `\n` only.
• [ ] Subset fields are exactly those declared, in order.
• [ ] `align_ascii` used for public lane, **sign + 6 decimals**.
• [ ] Hash wire bytes for body; expose `SSMNET-Body-Hash`.
• [ ] No alternate escapes, no pretty whitespace.
• [ ] Same bytes on two machines → same `sha256`.

One-line takeaway (Appendix D)

"**Normalize once, serialize once, hash once — identical bytes mean identical truth.**"

Appendix E — Golden Vectors & Reference Fixtures (Full Recipes)

(Purpose: provide canonical, copy-ready fixtures and generation recipes so any independent team can reproduce Section 12 results byte-for-byte.)

E.1 Scope & goals

- **What this appendix provides:** precise vector formats, edge-case sets, and generation rules for lanes, bands, shard/merge, subset hashing, and stamped chains.
- **Goal:** two independent implementations following these rules will compute the **same `align`**, the **same `band`**, and the **same `sha256`**, and will traverse the **same stamp chain**.

E.2 Lane vectors (kernel parity: batch • stream • shard-merge)

Vector fields (CSV):
```
series_id, order_id, a_raw, w, eps_a, eps_w, align_expected
```

Computation (normative kernel):
```
a_c := clamp(a_raw, -1+eps_a, +1-eps_a) → u := atanh(a_c) → U += w*u ; W += w → align := tanh( U / max(W, eps_w) )
```

Expectations:

- **Batch vs stream:** `|Δ| ≤ 1e-6` with identical inputs and parameters.
- **Shard-merge:** split by `series_id`, compute `(U_i, W_i)` per shard, fold `U_total := ΣU_i`, `W_total := ΣW_i`, then `align_total := tanh( U_total / max(W_total, eps_w) )`; **target** `|Δ| ≤ 1e-12` to monolith.

Edge rows to include:

- Near-clamp: `a_raw = ±(1 - 1e-12)` with small `eps_a`.
- Mixed weights: `w ∈ {1, 2, 5}` within same `series_id`.
- Long tail: 1k–10k rows for numeric stability inspection.

E.3 Boundary vectors (band edges with inclusivity text)

Vector fields (CSV):
`cut_name, align_in, band_expected, boundary_text`

Recipe: for each manifest cut (e.g., `(-0.80, +0.60]`), include points:
`exact_left`, `just_inside_left`, `just_outside_left`, `exact_right`, `just_inside_right`, `just_outside_right`.

Expectation: the `band_expected` must match **boundary inclusivity** precisely as declared in the manifest.

E.4 Shard/merge fixtures (U/W folding rules)

Fixture fields (CSV):
`series_id, shard_id, U_i, W_i, eps_w, align_expected`

Recipe:

1. Generate `U_i := Σ (w*atanh(clamp(a_raw)))` and `W_i := Σ w` per shard with the **same** `eps_a`, `eps_w`, `w` conventions.
2. Fold by summation only; **do not** re-clamp at joins.
3. Compare folded `align_total` to the monolith result.

Expectation: parity within `≤ 1e-12`.

E.5 Canonical subset fixtures (serialization → digest)

Lines (JSONL): each line is a **declared canonical subset** in the **declared field order**. Examples (illustrative):

- Label-first:
  `{"value":"...","band":"A0","manifest_id":"TRANSPORT_POSTURE.DEMO"}`
- Full (with canonical lane mirror):
  `{"value":"...","band":"A++","manifest_id":"TRANSPORT_POSTURE.DEMO","align_ascii":"+0.732000"}`

Normalization rules (normative):

- Text inside subset is **UTF-8**, **no BOM**, **NFC**, newline `\n`.
- Field order **exactly** matches the subset declaration list.
- `align_ascii` is fixed-form `(+|-)d.dddddd` (sign + 6 decimals).

Digest rule:
`sha256( subset_bytes [+ raw_body_bytes_if_declared] ) → HEX`

Expectation: independent tools recompute the **same HEX**.

E.6 Stamp chain fixtures (continuity)

Fields (text, one per line):
`SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEX_OR_NONE`

Recipe:

- **Genesis:** first line uses `prev=NONE`.
- For each subsequent envelope, compute digest as per **E.5** and set `prev` to the **prior accepted** digest.
- Include at least one **fork** case (duplicate `prev` with different follow-up) for negative testing.

Expectation: linear path resolves to a single **HEAD**; forks are rejected.

E.7 Evidence bundle parity set (minimal, deterministic)

- `manifests.json` — the exact manifest texts used (immutable).
- `envelopes.jsonl` — canonical subset lines, in declared order.
- `hashes.txt` — `sha256=<HEX>` per envelope line.
- `checkpoint.txt` — `HEAD=<HEX>` of the last accepted envelope.
- `verify.sh` — recomputes digests, walks chain, confirms band mapping; prints **ALL CHECKS PASSED**.

Expectation: bundle verifies offline with common tools; no network required.

E.8 Negative fixtures (must fail, with specific codes)

- **Digest mismatch:** flip one byte in a subset line → expect `E_BODY_HASH_MISMATCH`.
- **Broken prev:** set `prev` to a non-prior digest → expect `E_STAMP_PREV`.
- **Unknown manifest:** reference missing `manifest_id` → expect `E_MANIFEST_MISS`.
- **Lane policy drift (full):** change `eps_a` or `eps_w` mid-scope → expect `E_ALIGN_POLICY`.
- **Subset error:** misorder fields vs declared list → expect `E_SUBSET_DECL`.
- **Text normalization:** switch a subset line to NFD → expect `E_TEXT_NORM`.
- **Boundary misuse:** evaluate edges opposite inclusivity → expect `E_BOUNDARY_RULE`.

E.9 Reproducibility checklist (operators & auditors)

- **[ ]** Same manifest text (pin `sha256` of manifest body).
- **[ ]** Subset lines are **UTF-8 NFC**, newline `\n`, exact field order.
- **[ ]** `align_ascii` present and canonical when lane is public.
- **[ ]** Body hashing uses **wire bytes** when declared.
- **[ ]** Chain is linear; last digest equals **HEAD**.
- **[ ]** Kernel parity meets $\leq$ `1e-6` (batch vs stream) and $\leq$ `1e-12` (shard/merge).
- **[ ]** All negative fixtures fail with the **expected** error code.

One-line takeaway (Appendix E)

"Golden vectors make ambiguity impossible: same inputs, same bytes, same bands, same digests."

Appendix F — Profiles & Mappings (HTTP-M • WS-M • Stream-M • Device-M)

(Purpose: copy-paste mappings so teams can ship SSM-NET on real surfaces today. All profiles preserve payload bytes: `phi((m,a)) = m`.)

F.1 Profile overview (what changes, what never does)

- **Always the same semantics:**

    - **Lane kernel:** `a_c := clamp(a_raw, -1+eps_a, +1-eps_a) → u := atanh(a_c) → U += w*u ; W += w → align := tanh( U / max(W, eps_w) ).`
    - **Bands from manifest:** `band := cutpoint_map(align, manifest_id)` with explicit boundary inclusivity.
    - **Canonical subset:** declared list in order; `sha256( serialize(subset_fields) [+ raw_body_bytes_if_declared] ).`
    - **Stamp:** `SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEX`.
- **What varies per profile:** where fields live (headers vs. envelope keys), body binding rules, and intermediary behavior.

F.2 Common field names (profile-agnostic vocabulary)

- **Manifest:** `manifest_id` (wire name varies per profile).
- **Disclosure:** `value-only` / `value+band` / `full`.
- **Canonical subset declaration:** ordered list of field names (e.g., `["value","band","manifest_id"(,"align_ascii"?)]`).
- **Digest over subset/body:** `sha256=<HEX>`.
- **Stamp:** single line string as above.
- **Optional lane mirror when public:** `align_ascii` using fixed sign + 6 decimals (e.g., `+0.732000`).

F.3 HTTP-M (requests/responses over HTTP-like transports)

F.3.1 Header mapping (illustrative names)

- `SSMNET-Manifest: <manifest_id>`
- `SSMNET-Disclosure: value-only | value+band | full`
- `SSMNET-Canonical-Subset: ["value","band","manifest_id"(,"align_ascii"?)]`
- `SSMNET-Band: <BandLabel>` *(optional; label-first default)*
- `SSMNET-Align: <float>` *(only if full)*
- `SSMNET-Align-Ascii: <+/-d.dddddd>` *(canonical mirror, only if full)*
- `SSMNET-Body-Hash: sha256=<HEX>` *(when a body exists and is bound)*
- `SSMNET-Stamp: SSMCLOCK1|UTC_ISO|nonce|sha256=<HEX>|prev=<HEX|NONE>`
- `SSMNET-Checkpoint: HEAD=<HEX>` *(optional advert for scope)*

F.3.2 Body binding (critical rule)

- Hash the **exact wire bytes** as carried (after any content-encoding). Expose `SSMNET-Body-Hash`.

F.3.3 Intermediaries

- **MUST** pass payload bytes unchanged.
- **SHOULD** preserve `SSMNET-Body-Hash` and all SSMNET headers.

- **MAY** append a **new** `SSMNET-Stamp` (their observation). **MUST NOT** rewrite upstream stamps.

F.3.4 Minimal response (label-first)

```
SSMNET-Manifest: TRANSPORT_POSTURE.DEMO
SSMNET-Disclosure: value+band
SSMNET-Band: A0
SSMNET-Canonical-Subset: ["value","band","manifest_id"]
SSMNET-Body-Hash: sha256=9A2C...77F4
SSMNET-Stamp: SSMCLOCK1|2025-11-
07T12:59:00Z|n17|sha256=F2BC...09AE|prev=NONE
```

Invariant: body is byte-identical to the non-SSMNET version (`phi((m,a)) = m`).

F.3.5 Verification expectations (receiver)

1. Recompute `sha256` over declared subset (+ body if bound).
2. Validate `prev` chain; publish/compare `HEAD` if available.
3. Fetch manifest; confirm band derivability; if full, parity-check lane from kernel.

F.3.6 Typical errors

- Decode bytes then hash → `E_BODY_HASH_MISMATCH`.
- Reorder subset fields vs declaration → `E_SUBSET_DECL`.
- Missing manifest fetch → `E_MANIFEST_MISS`.

F.4 WS-M (message frames over a persistent socket)

F.4.1 Frame envelope (JSON-like illustrative):

```
{
  "value": <opaque>,
  "band": "<BandLabel>",
  "manifest_id": "<ManifestID>",
  "align_ascii": "+0.732000",        // only if full
  "subset": ["value","band","manifest_id"(,"align_ascii"?)],
  "sha256": "<HEX>",
  "stamp": "SSMCLOCK1|UTC_ISO|nonce|sha256=<HEX>|prev=<HEX|NONE>"
}
```

F.4.2 Message ordering & continuity

- Use `prev` across frames in the stream scope.
- Heartbeats MAY carry only `stamp` and `subset` commitment (no `value`).

F.4.3 Intermediaries (relays/brokers)

- Forward frames intact; if aggregating, **append** their own stamped note.
- No mutation of prior `stamp` or subset bytes.

F.4.4 Typical pitfalls

- Pretty-printing `value` differently between producer/relay → ensure **declared subset bytes** are serialized deterministically before hashing.
- Resetting `(U,W)` mid-session without new `manifest_id` → lane policy drift → `E_ALIGN_POLICY`.

F.5 Stream-M (chunked telemetry / pub-sub / queue)

F.5.1 Record structure

- **Header fields:** `manifest_id`, `disclosure`, `subset`, `sha256`, `stamp`.
- **Payload channel:** raw bytes; hash **payload channel bytes** if bound.
- **Shard/merge:** compute `(U,W)` per shard; fold by summation only.

F.5.2 Windowing & epochs

- Very long streams MAY declare **epoch rollover** (e.g., `epoch_id`) with `(U,W)` reset at epoch boundaries. The manifest or stream policy **MUST** define epoch rules; changing them requires a new `manifest_id` or a stamped policy event.

F.5.3 Backpressure & replay

- Consumers re-verify by recomputing `sha256` over stored subset (+ body if stored).
- Checkpointing: providers periodically publish `HEAD=<HEX>` to speed range replays.

F.6 Device-M (resource-constrained devices & gateways)

F.6.1 Wire-lean mapping

- **Minimal envelope:** `band`, `manifest_id`, `subset`, `sha256`, `stamp`.
- **No free-form floats on wire:** if lane is public, **only** send `align_ascii`.
- **Integer weights:** prefer `w := 1` unless manifest defines otherwise.

F.6.2 Energy & storage notes

- Optionally buffer `(U,W)` locally and flush at intervals; append a stamped summary packet.
- For binary links, store subset & `stamp` in a small sidecar metadata block.

F.6.3 Gateway duties

- Gateways MUST forward device payload bytes unchanged; they MAY add their own stamped notes or expose well-known endpoints for the device scope.

F.7 Discovery & fetch patterns (all profiles)

- **Manifest:** `GET /.well-known/ssmnet/manifest/<manifest_id>` → immutable text.
- **Checkpoint:** `GET /.well-known/ssmnet/checkpoint[?scope=<name>]` → `HEAD=<HEX>`.
- **Evidence:** `GET /.well-known/ssmnet/evidence[?scope=&from=&to=]` → deterministic archive.
- **Client "fetch" option:** profiles SHOULD expose a way to **auto-fetch** the manifest when `manifest_id` is first seen, cache it, and pin its `sha256` locally.

F.8 Federation levels per profile (what crosses)

- **L1:** `band + manifest_id + stamp + subset`.
- **L2:** L1 + `align` (or `align_ascii`).
- **L3:** L2 + pullable **evidence bundle**.
  Negotiation: each side declares max; operate at the minimum; never coerce higher disclosure.

F.9 Privacy posture per profile

- **Default label-first (value+band).**
- **No PII** in the canonical subset.
- Public lane? Include **only** `align_ascii` (fixed form) to avoid float drift.
- If a profile exposes user-supplied fields, keep them **outside** the subset unless essential.

F.10 Error signalling (overlay-safe)

- Add `SSMNET-Error: <code>` alongside a **new** `SSMNET-Stamp` (incident note).
- Core codes: `E_POLICY_MISMATCH`, `E_BODY_HASH_MISMATCH`, `E_STAMP_PREV`, `E_MANIFEST_MISS`, `E_ALIGN_POLICY`, `E_SUBSET_DECL`, `E_TEXT_NORM`, `E_CHECKPOINT_DRIFT`, `E_FORMAT`.

F.11 Micro-checklists (ship it today)

HTTP-M (endpoint)

- **[ ]** Publish manifest; pin its text.

- **[ ]** Emit `SSMNET-Manifest`, `SSMNET-Canonical-Subset`, `SSMNET-Body-Hash` (if body), `SSMNET-Stamp`.
- **[ ]** Keep payload bytes unchanged (`phi((m,a)) = m`).
- **[ ]** Host well-known paths; advertise `HEAD` optionally.
- **[ ]** Run Golden Diagnostics; archive evidence.

WS-M (socket)

- **[ ]** Per-message envelope with `subset`, `sha256`, `stamp`.
- **[ ]** Maintain linear `prev` chain across frames.
- **[ ]** Deterministic serialization of subset bytes.

Stream-M (telemetry)

- **[ ]** Define scope & epochs; document shard/merge rules.
- **[ ]** Publish periodic `HEAD`.
- **[ ]** Evidence export supports time windows.

Device-M (constrained)

- **[ ]** Use `align_ascii` when public; avoid raw floats.
- **[ ]** Gateway exposes well-known endpoints.
- **[ ]** Append-only stamped summaries permitted.

One-line takeaway (Appendix F)

"Same semantics everywhere; per-profile mappings just say where the manifest, subset, digest, and stamp live—bytes stay byte-exact."

Appendix G — Fetch & Discovery Cookbook (cURL • Pinning • Offline Replay)

*(Purpose: make discovery and verification **muscle-memory**. Copy these blocks to stand up manifests, checkpoints, and evidence without custom tooling.)*

G.1 What this appendix covers

- **Fetch** a manifest by `manifest_id`, **pin** its bytes locally.
- **Check** a scope `HEAD` and detect drift.
- **Pull** an evidence bundle and verify **offline**.

- **Wire sanity**: prove payload invariance `phi((m,a)) = m`, and the stamp contract `SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEX`.
- **No company names** or products; generic endpoints only.

G.2 Manifest fetch & pin (immutable per `manifest_id`)

Goal: fetch once, keep forever; every replay uses the **exact same bytes**.

Fetch (text, UTF-8, NFC expected):

```
curl -sS https://example.com/.well-
known/ssmnet/manifest/TRANSPORT_POSTURE.DEMO \
  -H "Accept: text/plain, application/json" \
  -o manifest.TRANSPORT_POSTURE.DEMO.txt
```

Pin digest (local note):

```
sha256sum manifest.TRANSPORT_POSTURE.DEMO.txt >
manifest.TRANSPORT_POSTURE.DEMO.sha256
cat manifest.TRANSPORT_POSTURE.DEMO.sha256
# Expect: <HEX>  manifest.TRANSPORT_POSTURE.DEMO.txt
```

Replay rule (operator habit):

- Always record the **hex** from the manifest you used.
- If the published text ever changes for the same `manifest_id` → treat as policy violation (should rotate `manifest_id` instead).

Quick checklist (copy-ready):

- **[ ]** Fetched manifest text is byte-stable.
- **[ ]** Boundary inclusivity is explicit (e.g., `(-0.80, +0.60]`).
- **[ ]** `eps_a`, `eps_w`, and any `weight_rule` are declared.
- **[ ]** `disclosure` mode is stated (`value-only` / `value+band` / `full`).

G.3 Scope HEAD check (lightweight drift detection)

Goal: know the latest accepted digest for a scope without downloading everything.

Fetch HEAD (default scope):

```
curl -sS https://example.com/.well-known/ssmnet/checkpoint \
  -H "Accept: text/plain" | tee checkpoint.txt
```

Example response (illustrative):

```
HEAD=1F3C4D...A902
scope=default
```

```
updated=2025-11-07T12:30:00Z
```

Compare with your last known HEAD:

```
KNOWN="1F3C4D...A902"
REMOTE=$(grep '^HEAD=' checkpoint.txt | cut -d= -f2)
test "$KNOWN" = "$REMOTE" && echo "OK: HEAD unchanged" || echo "NOTE: HEAD
moved"
```

Operator notes:

- HEAD movement is **append-only** under the stamp rule (`prev` chain).
- If you detect **forks**, providers should publish a short `fork.txt` inside the evidence bundle that explains recovery (see Section 7F).

G.4 Evidence bundle pull (self-service audit)

Goal: fetch everything needed to print **ALL CHECKS PASSED** offline.

Pull (time/range optional):

```
curl -sS -L "https://example.com/.well-
known/ssmnet/evidence?scope=default&from=2025-11-01T00:00:00Z&to=2025-11-
07T23:59:59Z" \
  -o evidence.zip
unzip -q evidence.zip -d evidence
```

Expected minimal tree (illustrative):

```
evidence/
  manifests.json
  envelopes.jsonl
  hashes.txt
  checkpoint.txt
  verify.sh
  # body/ (optional if body binding was declared)
```

Run offline verifier:

```
( cd evidence && sh ./verify.sh )
# Expect: ALL CHECKS PASSED (or a precise FAIL code)
```

If FAIL occurs, map to error model:

- `E_BODY_HASH_MISMATCH` → body bytes hashed incorrectly (not wire bytes).
- `E_STAMP_PREV` → broken `prev` chain; investigate fork or edit.
- `E_MANIFEST_MISS` → referenced `manifest_id` not present.
- `E_SUBSET_DECL` → declared subset vs serialized bytes mismatch.
- `E_ALIGN_POLICY` → kernel safeguards or weights drifted mid-scope (full mode).
- `E_TEXT_NORM` → text in subset not UTF-8 NFC.

G.5 Proving payload invariance on a live response

Goal: show bytes are unchanged while meaning rides beside them (`phi((m,a)) = m`).

Fetch raw body exactly as carried (respecting content-encoding):

```
curl -sS https://example.com/resource \
  -D headers.txt \
  --output body.raw
```

Extract the declared body hash from headers:

```
DECL=$(grep -i '^SSMNET-Body-Hash:' headers.txt | head -1 | sed
's/.*sha256=//')
ACT=$(sha256sum body.raw | awk '{print $1}')
test "$DECL" = "$ACT" && echo "OK: body bytes invariant" || echo "FAIL:
body hash mismatch"
```

Interpretation: if hashes match, the overlay did **not** mutate payload bytes; headers carry portable meaning only.

G.6 Canonical subset parity (quick local probe)

Goal: recompute the digest exactly as stamped for a single envelope line.

Given a subset line (JSONL exact bytes) in `line.json`:

```
cat line.json | tr -d '\r' > .subset.bytes
sha256sum .subset.bytes | awk '{print $1}'
```

Compare with the stamped `sha256=HEX` in the `SSMCLOCK1|...` line.

- If mismatch → diagnose `E_SUBSET_DECL` (field order/spacing/escaping) or `E_TEXT_NORM`.

G.7 Manifest-aware band replay (label-first vs full)

Goal: confirm that bands come from **declared cutpoints**, not guesswork.

Label-first (no numeric lane disclosed):

- Fetch the manifest (G.2).
- Ensure the label in the envelope falls within a declared interval **given boundary text**.
- If it does not, flag as policy error.

Full disclosure:

- Recompute lane via kernel:
  `a_c := clamp(a_raw, -1+eps_a, +1-eps_a) -> u := atanh(a_c) -> U += w*u ; W += w -> align := tanh( U / max(W, eps_w) )`
- Mirror to canonical text `align_ascii` (sign + 6 decimals).
- Apply `band := cutpoint_map(align, manifest_id)`; require exact edge handling.

G.8 Minimal incident runbook (fetch-first, no ceremony)

1. **Pin** the manifest for the `manifest_id` you saw.
2. **Download** the evidence bundle for the time window.
3. **Run** `verify.sh` offline; capture result.
4. If FAIL, **echo** `SSMNET-Error: <code>` back on your next response and **append** a new `SSMNET-Stamp` (do **not** rewrite history).
5. **Publish** a short note at the evidence URL root indicating remediation ETA; keep payload delivery unaffected.

G.9 Quick cURL crib sheet (copy-paste)

```
# 1) Fetch & pin manifest
curl -sS https://host/.well-known/ssmnet/manifest/<manifest_id> -o
manifest.txt
sha256sum manifest.txt

# 2) Check HEAD
curl -sS https://host/.well-known/ssmnet/checkpoint | tee checkpoint.txt

# 3) Pull evidence
curl -sS -L "https://host/.well-known/ssmnet/evidence?scope=default" -o
evidence.zip
unzip -q evidence.zip -d evidence && (cd evidence && sh ./verify.sh)

# 4) Prove body invariance
curl -sS https://host/resource -D headers.txt --output body.raw
DECL=$(grep -i '^SSMNET-Body-Hash:' headers.txt | sed 's/.*sha256=//')
ACT=$(sha256sum body.raw | awk '{print $1}')
test "$DECL" = "$ACT" && echo OK || echo FAIL
```

G.10 Operator checklist (daily habit)

- **[ ]** Manifests pinned (hex recorded) for every active `manifest_id`.
- **[ ]** HEAD polled on cadence; drift recorded.
- **[ ]** Evidence bundles archived per window; offline verifier run.
- **[ ]** Any FAIL mapped to an **error code** and appended as a **new stamp** (no edits).

- **[ ]** Payload invariance spot-checks pass (`phi((m,a)) = m`).

One-line takeaway (Appendix G)

"Fetch. Pin. Verify offline. Append stamps—never edits. That's how meaning travels and stays provable."

Appendix H — Example Manifests (Two Styles: Minimal • With Obligations)

(Purpose: copy-ready manifest texts that align with Sections 3 and 7. Each manifest is immutable per `manifest_id`; any meaning change mints a new `manifest_id`.)

H.1 How to use these examples

- Publish each manifest at: `/.well-known/ssmnet/manifest/<manifest_id>`
- Treat the body as **byte-stable text** (UTF-8, no BOM, newline `\n`), suitable for hashing and long-term pinning.
- Bands map from the kernel lane: `a_c := clamp(a_raw, -1+eps_a, +1-eps_a) -> u := atanh(a_c) -> U += w*u ; W += w -> align := tanh( U / max(W, eps_w) )`.
- Boundary text is **authoritative** for edge behavior (e.g., `(-0.80, +0.60]`).
- Disclosure defaults to **label-first** unless explicitly set to **full**.

H.2 Style A — Minimal, Label-First (copy-paste text)

(Lean rulebook for public posture. No numeric lane disclosure on the wire.)

```
manifest_id: "TRANSPORT_POSTURE.DEMO"
purpose: "Public label-first posture for generic content delivery; no
numeric lane disclosure."
disclosure: value+band

bands:
  - "A++"     : [-1.00, -0.80]
  - "A0"      : (-0.80, +0.60]
  - "CRITICAL": (+0.60, +1.00]

boundary_inclusivity:
  A++     : left-inclusive,  right-inclusive
  A0      : left-open,       right-inclusive
  CRITICAL: left-open,       right-inclusive
```

```
kernel_safety:
  eps_a: 1e-6
  eps_w: 1e-9
weight_rule: "equal"    # w := 1 unless otherwise stated

assumptions: "Nominal sampling; no guaranteed sample rate. Align is
computed per Section 2A kernel."

notes:
  - "Bands describe content/system posture, not people."
  - "Any policy change mints a new manifest_id; older texts remain online."
```

Intended use: public endpoints, mirrors, discovery surfaces. Receivers verify band via this manifest; numeric lane may be kept in logs.

H.3 Style B — Operational With Obligations (copy-paste text)

(Adds clear human/agent actions and timing windows. Still label-first by default.)

```
manifest_id: "OPERATIONS_POSTURE.v3"
purpose: "Operational posture with human/agent obligations and timing
windows."
disclosure: value+band     # full allowed only if explicitly declared at
the interface

bands:
  - "STABLE"   : [-1.00, -0.40]
  - "WATCH"    : (-0.40, +0.40]
  - "CRITICAL" : (+0.40, +1.00]

boundary_inclusivity:
  STABLE   : left-inclusive,  right-inclusive
  WATCH    : left-open,       right-inclusive
  CRITICAL : left-open,       right-inclusive

kernel_safety:
  eps_a: 1e-6
  eps_w: 1e-9
weight_rule: "equal"        # w := 1 unless a declared weighting is
published

obligations:
  STABLE:
    action: "operate-normally"
    human_required: "no"
    window: "none"
  WATCH:
    action: "increase-sampling; human-review <=30m"
    human_required: "preferred"
    window: "PT30M"
  CRITICAL:
    action: "enter-safe-mode; human-ack <=10m"
    human_required: "yes"
    window: "PT10M"

tolerances:
  batch_vs_stream: "<=1e-6"
```

```
  shard_merge: "<=1e-12"

assumptions: "Sensor calibration nominal; known latencies tolerated. Epoch
rollover permitted if declared out-of-band."

notes:
  - "Obligations are promises; stamped trails prove if duty windows were
met."
  - "Identity/PII is out of scope; posture concerns signals and systems."
```

Intended use: operations, control rooms, regulated audit trails. Evidence bundles can confirm whether **obligations** were met within declared windows.

H.4 Optional 'full' lane disclosure variant (when parity must be public)

(Flip `disclosure` to `full` and add a canonical mirror for floats.)

```
manifest_id: "LAB_TELEMETRY.full.v1"
purpose: "Public numeric parity for research collaboration."
disclosure: full

bands:
  - "GREEN"  : [-1.00, -0.20]
  - "AMBER"  : (-0.20, +0.20]
  - "RED"    : (+0.20, +1.00]

boundary_inclusivity:
  GREEN : left-inclusive,  right-inclusive
  AMBER : left-open,       right-inclusive
  RED   : left-open,       right-inclusive

kernel_safety:
  eps_a: 1e-6
  eps_w: 1e-9
weight_rule: "equal"

lane_public:
  align_ascii: "required"       # fixed sign + 6 decimals, e.g., +0.732000
  decimal_policy: "sign + 6dp"  # (+|-)d.dddddd

assumptions: "Public research data; deterministic serializers in use."
```

Receiver expectation: recompute lane via kernel and parity-check band assignment exactly at boundaries.

H.5 Authoring checklist (prevent silent drift)

- **[ ]** Bands list every interval and include **boundary inclusivity** text.
- **[ ]** `eps_a`, `eps_w`, and `weight_rule` included under `kernel_safety`.
- **[ ]** `disclosure` is explicit (`value-only`, `value+band`, or `full`).
- **[ ]** Any obligations have clear verbs and time windows (e.g., `PT10M`).
- **[ ]** No identity fields; posture is about signals/systems.

- **[ ]** If anything above changes → **new `manifest_id`**; keep old text online.

H.6 Validation notes (receivers & auditors)

- **Bands are manifest-truth:** apply `band := cutpoint_map(align, manifest_id)` exactly as these texts declare.
- **Decimals:** if `disclosure: full`, mirror `align` to `align_ascii` using fixed sign + 6 decimals to avoid serializer drift.
- **Kernel parity:** target `≤ 1e-6` (batch vs stream) and `≤ 1e-12` (shard/merge).
- **Stamp chain:** continuity is append-only: `SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEX_OR_NONE`.

One-line takeaway (Appendix H)

"Publish a frozen, byte-stable rulebook with explicit edges and (optionally) obligations; if meaning changes, mint a new `manifest_id`—never edit history."

Appendix I — Glossary (compact)

(Purpose: give implementers and auditors a single-page language for SSM-NET. All terms align with Sections 1–22 and Appendices A–H.)

Alignment lane (lane)
A bounded, replayable dial computed beside original bytes. Kernel: `a_c := clamp(a_raw, -1+eps_a, +1-eps_a) -> u := atanh(a_c) -> U += w*u ; W += w -> align := tanh( U / max(W, eps_w) )`.

Band (label)
Human-readable posture bucket derived from the lane via a published manifest: `band := cutpoint_map(align, manifest_id)` with explicit boundary inclusivity.

Boundary inclusivity (edges)
Text that defines interval edges (e.g., `(-0.80, +0.60]` → left-open, right-inclusive). Authoritative for edge behavior.

Canonical subset (declared subset)
Ordered list of fields bound by the digest (and optionally the raw body). Example: `["value","band","manifest_id"(,"align_ascii"?)]`.

Checkpoint (HEAD)
Well-known report of the latest accepted digest for a scope. Example body: `HEAD=<HEX>` plus optional `scope` and `updated`.

Collapse parity
Payload invariance guarantee: `phi((m,a)) = m`. The overlay never edits original bytes.

Conformance roles
Sender (emits overlay + payload), Receiver (verifies), Intermediary (forwards, may append stamped note), Publisher (hosts manifests/checkpoints/evidence).

Continuity stamp (stamp)
One-line, append-only anchor of time and order:
`SSMCLOCK1|UTC_ISO|nonce|sha256=HEX|prev=HEX_OR_NONE`.

Decimal canonical (align_ascii)
Fixed text mirror of `align` when public, with sign and six decimals (e.g., `+0.732000`) to avoid float/locale drift.

Disclosure modes
`value-only`, `value+band` (default, label-first), `full` (lane public, include `align_ascii`).

Digest (subset digest / body hash)
`sha256( serialize(subset_fields) [+ raw_body_bytes_if_declared] )`. Body hash advertised as `sha256=<HEX>` over **wire bytes**.

Evidence bundle
Deterministic archive for offline verification: `manifests.json`, `envelopes.jsonl`, `hashes.txt`, `checkpoint.txt`, `verify.sh` (and optional `body/`).

Epoch (streaming policy)
Declared boundary where `(U,W)` may reset for long streams; changing epoch rules requires a new `manifest_id` or stamped policy event.

Error model (overlay-safe)
Wire signaling of verification failures via `SSMNET-Error: <code>` plus a **new** stamped note (never edit history). Examples: `E_BODY_HASH_MISMATCH`, `E_STAMP_PREV`, `E_SUBSET_DECL`.

Federation levels
L1 (label-first), L2 (lanes reproducible), L3 (full evidence). A link operates at `min(local_max, remote_max)`; never coerce higher disclosure.

Genesis
First stamped item in a scope (`prev=NONE`).

Intermediary
Mirror/cache/relay that **must** forward payload bytes untouched; **should** preserve overlay; **may** append its own stamped observation.

Manifest (rulebook)
Immutable text published at `/.well-known/ssmnet/manifest/<manifest_id>` defining bands, edges, `eps_a`, `eps_w`, optional `weight_rule`, disclosure, and assumptions.

Manifest rotation
Any change to meanings, thresholds, or obligations requires a **new** `manifest_id`; old manifest texts remain online.

Order-invariance
Lane fusion yields identical results (within tolerances) under batch, stream, or shard-merge when the same kernel and parameters are applied.

Payload invariance
Original body bytes are never altered by SSM-NET (`phi((m,a)) = m`). If a body is bound, the digest must be over **exact wire bytes**.

Prev chain (continuity)
`prev` equals the prior accepted digest; HEAD equals the last. Used for append-only ordering and fork detection.

Profiles (bindings)
Concrete mappings for transports (e.g., HTTP-M, WS-M, Stream-M, Device-M) that place manifest, subset, digest, and stamp in headers or envelope keys.

Scope
Logical sequence boundary for continuity (e.g., a URL, topic, stream, or device). Each scope has its own genesis and HEAD.

Shard-merge
Fold per-shard `(U_i,W_i)` via `U_total := ΣU_i`, `W_total := ΣW_i`, then `align_total := tanh( U_total / max(W_total, eps_w) )`. No re-clamp at joins.

Tolerances (math)
Target parity: `<= 1e-6` (batch vs stream) and `<= 1e-12` (shard-merge) in 64-bit float.

Well-known endpoints
Read-only discovery surfaces: `manifest/<manifest_id>`, `checkpoint`, `evidence` (time/range optional). Byte-stable and cache-friendly.

Weight rule (`w`)
Declared policy for contribution weighting in the kernel; `w := 1` if none is declared.

Wire bytes
The exact bytes **as carried** (post content-encoding). Always hash these for body commitments.

Appendix J — Operator Runbooks (paste-ready, copy-ready)

(Purpose: give Day-0 to steady-state procedures you can paste into ops docs. All steps preserve payload invariance `phi((m,a)) = m` and the kernel lane: `a_c := clamp(a_raw, -1+eps_a, +1-eps_a) → u := atanh(a_c) → U += w*u ; W += w → align := tanh( U / max(W, eps_w) ).`)

J.1 Day-0 Enablement (first hour)

Goal: publish one manifest, add overlay headers to one read endpoint, expose discovery, and self-verify.

Checklist

- **[ ]** Draft and publish **manifest** at `/.well-known/ssmnet/manifest/<manifest_id>` with bands, boundary inclusivity, `eps_a`, `eps_w`, `weight_rule`, `disclosure`.
- **[ ]** Add **HTTP-M** overlay fields on one endpoint:
  - `SSMNET-Manifest: <manifest_id>`
  - `SSMNET-Canonical-Subset: ["value","band","manifest_id"(,"align_ascii"?)]`
  - `SSMNET-Body-Hash: sha256=<HEX>` (if body)
  - `SSMNET-Stamp: SSMCLOCK1|UTC_ISO|nonce|sha256=<HEX>|prev=<HEX|NONE>`
- **[ ]** Host **checkpoint** at `/.well-known/ssmnet/checkpoint` returning `HEAD=<HEX>`.
- **[ ]** Run **Golden Diagnostics** locally; store evidence bundle.
- **[ ]** Announce **disclosure mode** (default: `value+band`) to consumers.

Smoke Test (operator)

1. GET the endpoint, save headers and body as bytes.
2. Recompute body `sha256` over **wire bytes**; expect match with `SSMNET-Body-Hash`.
3. Fetch manifest; verify band interval edges.
4. Confirm `prev` chain: `prev=NONE` for first, else equals last digest.

J.2 Day-7 Cadence (parity sentinel & checkpointing)

Goal: automate parity and publish HEAD regularly.

Checklist

- **[ ]** Cron/CI runs lane parity on sampled traffic: batch vs stream `<= 1e-6`; shard-merge `<= 1e-12`.
- **[ ]** Publish `HEAD=<HEX>` every N minutes; include `updated=<UTC_ISO>`.

- **[ ]** Archive weekly **evidence bundles** (manifests, envelopes, hashes, checkpoint, verify script).
- **[ ]** Ensure intermediaries **append** stamps, never rewrite.

Parity Sentinel (sketch)

- For each scope, recompute `align` locally (if full) or verify band derivability (label-first).
- On any mismatch: emit `SSMNET-Error: E_ALIGN_POLICY` (full) or `E_SUBSET_DECL` (subset issue) **with a new stamp**.

J.3 Manifest Rotation (policy change without drift)

Trigger: thresholds or obligations change.

Steps

1. Author **new text** with updated cuts/obligations.
2. Mint a **new `manifest_id`**; publish at well-known path.
3. Switch producers to emit the **new `manifest_id`**.
4. Keep **old manifests online**.
5. Note rotation in ops log; do **not** edit history.

Acceptance Check

- Random sample near each cut; verify boundary inclusivity is honored by receivers.
- Evidence bundle contains both old and new `manifest_id` references across time.

J.4 Incident Response (overlay-safe, append-only)

Symptoms → Actions

- **Digest mismatch (subset/body):**
  - Action: return `SSMNET-Error: E_BODY_HASH_MISMATCH` (or `E_SUBSET_DECL`) and **append** an incident `SSMNET-Stamp`.
  - Verify raw bytes path; ensure hashing uses **wire bytes**.
- **Broken continuity (`prev`):**
  - Action: `SSMNET-Error: E_STAMP_PREV`; freeze new emissions until last accepted `HEAD` is re-established; publish short `fork.txt` in evidence.
- **Manifest not found:**
  - Action: `SSMNET-Error: E_MANIFEST_MISS`; republish or correct `manifest_id`.
- **Lane policy drift (full mode):**
  - Action: `SSMNET-Error: E_ALIGN_POLICY`; ensure `eps_a`, `eps_w`, `w` consistent within scope or rotate manifest.

- **Text normalization drift:**
  - Action: `SSMNET-Error: E_TEXT_NORM`; normalize to UTF-8 NFC for subset serialization.

Never do

- Do **not** modify past stamps, subsets, or bodies.
- Do **not** silently change manifests; always rotate.

J.5 Federation Setup (peer-to-peer disclosure)

Objective: interoperate at the lowest common level.

Procedure

1. Exchange **max levels**: L1/L2/L3.
2. Operate at `min(local_max, remote_max)`.
3. Publish each side's **`manifest_id`**; do **not** force equivalence.
4. If L2/L3, enable lane parity or full evidence pull.
5. Document that cross-manifest comparisons are **informational** unless a treaty manifest declares equivalence.

Health Checks

- L1: bands and stamps traverse; `HEAD` reachable.
- L2: `align` or `align_ascii` present; parity within tolerances.
- L3: evidence bundle verifies offline: **ALL CHECKS PASSED**.

J.6 Intermediary Playbook (mirrors, caches, relays)

MUST

- Forward payload bytes unchanged (`phi((m,a)) = m`).
- Preserve upstream `SSMNET-Body-Hash`, subset declaration, and `SSMNET-Stamp`.

MAY

- Append **their own** `SSMNET-Stamp` (observation), never rewrite upstream.

SHOULD

- Expose `/.well-known/ssmnet/checkpoint` for accelerated replay (`HEAD=<HEX>`).
- Pass through `SSMNET-Error` codes from upstream when present.

J.7 Evidence Lifecycle (pull → store → verify)

Daily

- Pull evidence per scope/time window; store immutable.
- Run `verify.sh` offline; archive the console result.

Weekly

- Sample **negative tests** (one byte flip) to prove verifier catches errors.
- Produce a short **provenance note**: manifest pins (sha256), time range, HEAD.

Retention

- Keep manifest texts and evidence bundles per policy; never delete manifests linked by history.

J.8 Privacy Posture (default label-first)

Defaults

- Public: `value+band` and `manifest_id` and stamp.
- Private: numeric lane unless `disclosure=full`.
- Canonical subset excludes PII; posture is about signals/systems.

Audits

- Prove that public surfaces never emit `align` unless explicitly set to `full`.
- If `full`: always mirror `align_ascii` using fixed sign + 6 decimals.

J.9 Stream & Epoch Operations (long-running scopes)

When to roll epochs

- Very long streams where `(U,W)` growth affects numeric conditioning or operational rotation.

Rules

- Declare `epoch_id` and reset `(U,W)` at boundaries; document in manifest or stamped policy note.
- Do **not** change epoch policy mid-scope without a new `manifest_id` or stamped notice.

Shard/Merge

- Compute `(U_i,W_i)` per shard; fold via `U_total := ΣU_i`, `W_total := ΣW_i`, `align_total := tanh( U_total / max(W_total, eps_w) )`.
- Do **not** re-clamp at joins.

J.10 CI Gate (pre-deploy verification)

Pipeline Steps

- **[ ]** Run Golden Diagnostics (Section 12) with current manifest set.
- **[ ]** Validate ABNF conformance for headers (Appendix A).
- **[ ]** Serialize subsets in deterministic form; prove digest parity across two serializers.
- **[ ]** Build evidence bundle from a synthetic run; require **ALL CHECKS PASSED**.
- **[ ]** Block deploy on any `E_*` outcome.

J.11 Operator Crib Sheets (copy blocks)

Header minimum (label-first)

```
SSMNET-Manifest: <manifest_id>
SSMNET-Band: <BandLabel>
SSMNET-Canonical-Subset: ["value","band","manifest_id"]
SSMNET-Body-Hash: sha256=<HEX>
SSMNET-Stamp: SSMCLOCK1|UTC_ISO|nonce|sha256=<HEX>|prev=<HEX|NONE>
```

Full disclosure add-ons

```
SSMNET-Align: <float>
SSMNET-Align-Ascii: +0.000000
```

Error echo (append-only)

```
SSMNET-Error: E_BODY_HASH_MISMATCH
SSMNET-Stamp: SSMCLOCK1|UTC_ISO|nonce|sha256=<HEX>|prev=<HEAD_HEX>
```

Well-known responses (illustrative)

```
# checkpoint
HEAD=<HEX256>
scope=default
updated=<UTC_ISO>
```

J.12 Rollback & Repair (without rewriting history)

If a bad manifest shipped

- Publish corrected text under a **new `manifest_id`**.
- Keep the old text online; add a stamped ops note explaining the correction.
- Switch producers to the new id; receivers keep honoring old id for historical data.

If a digest bug shipped

- Fix hashing implementation; **do not** re-emit past messages.
- Publish evidence bundles marking the affected window; parity sentinel should raise `E_BODY_HASH_MISMATCH` for those samples, proving detection works.

J.13 Executive Snapshot (paste into status emails)

- **Bytes untouched:** `phi((m,a)) = m` proven by body hash over wire bytes.
- **Meaning portable:** band from manifest with explicit edges; `manifest_id` pinned.
- **Continuity verifiable:** stamped chain with `prev`; `HEAD` published.
- **Privacy by default:** label-first; lane disclosure opt-in.
- **Audit self-service:** evidence bundle prints **ALL CHECKS PASSED** offline.

One-line takeaway (Appendix J)

"Operate append-only: publish a frozen rulebook, bind a declared subset, stamp continuity, and prove everything offline—without ever touching payload bytes."

OMP