XRPlay Music on XRPL

Technical Specification

XRP-Only Streaming Settlement System

Version 0.5 — Product-Ready Consumer Draft

Jan 2026

Abstract (Board-Ready)

XRPlay Music is a consumer-scale streaming platform whose **only** settlement rail is native **XRP** on **XRPL**. No token, no IOU, no points, no yield—only sealed, deterministic manifests that convert **verified listening seconds** into **signed XRP disbursements** to rights-holders.

Why the board should care (30 s read):

- **Revenue utility:** subscriptions → metered value → **XRPL** settlement → rights-holder payouts, producing repeat settlement throughput without balance-sheet noise.
- **Aggregator bypass:** artists upload directly to XRPlay; no third-party distributor is required.
- **Risk-off execution:** historic DSP failure modes (floating balances, silent clawbacks, infinite storage, double-pay, insolvency under growth) are engineered out via **XRP**-only finality, deterministic manifests, bounded storage economics, and fail-closed signing.
- **Consumer utility:** a junk-free catalogue and **ad-free** discovery loop—structurally misaligned with ad-driven DSP incentives.
- **Audit-ready by construction:** bit-identical settlement artifacts, version-pinned rights, reconciliation delta = 0, safe-mode ≤ 60 s—no discretionary accounting.
- **Consumptive access only:** listeners purchase **XRPass** (time-limited playback access). XRPass confers **no** ownership, no transfer, no resale, no royalties, no rewards, and no economic participation. All value flows to rights-holders only.

Wallet choice, same rail:

- **Self-custody (Mode A):** the user holds private keys; XRPlay does not custody user funds.
- **Managed balance (Mode B):** platform custodies **XRP** 1:1 under custody controls; withdrawals are raw **XRP** to a user-specified destination; no fiat is held on XRPlay's books.

XRPass hard limits (fail-closed): policy-enforced caps on XRPass daily sales, per-XRPass price, and fee spend. Breaches halt outbound value within 60 s; there is no discretionary override path without incident logging and quorum approval.

Storage is an economic control surface: quotas, reserve governance, and deterministic tombstoning. "Free infinite upload" is out-of-spec.

Threat model: withstands hostile clients, bot farms, device emulators, replay crews, spoofers, concurrency abuse, stolen-account rings, regulator freezes, and 100× congestion **without** breaking determinism or double-paying.

Bottom line: a consumer-scale, audit-grade settlement workload that drives continuous **XRP** throughput on **XRPL**, while exposing **no hidden token or consumer investment surface**.

Normative language: the key words **MUST**, **SHOULD**, and **MAY** are to be interpreted as described in RFC 2119.

Contents

1 Scope, Non-Goals, and System Boundary

Purpose (non-technical). This document specifies **what XRPlay is** and, equally important, **what it deliberately is not**. The goal is to remove ambiguity for engineers, auditors, partners, and board reviewers by fixing the system boundary up front.

1.1 In Scope

XRPlay **does**:

- operate a consumer-scale music streaming service,
- meter verified listening seconds deterministically,
- resolve rights and split tables via version-pinned contracts,
- settle creator payouts exclusively in native **XRP** on **XRPL**,
- provide time-limited, consumptive playback access (XRPass) to listeners,
- retain audit-grade artifacts enabling bit-identical replay of any settlement window.

1.2 Explicit Non-Goals

XRPlay **does not**:

- issue a token, credit, point, badge, or synthetic balance,
- provide listener rewards, yield, ownership, resale, or secondary markets,
- expose per-stream economics to users,
- store or represent value off-ledger as money,
- offer financial advice, investment products, or speculative features,
- rely on discretionary reconciliation, manual payout judgment, or trust-based accounting.

1.3 System Boundary

- **On-ledger:** final settlement, custody primitives, and immutable economic truth (XRPL).
- **Off-ledger:** playback, metering, fraud detection, rights resolution, aggregation, and UX.
- **Untrusted surfaces:** all client applications.
- **Authoritative surfaces:** settlement engine, rights registry, signing service.

1.4 Interpretation

If a proposed feature:

- causes a listener-facing object to resemble money,

- introduces transferability or secondary value,
- requires discretionary payout correction,
- or weakens deterministic replay,

then it is **out of scope by definition** and rejected without further analysis.

Design axiom. XRPlay optimizes for **economic correctness over growth optics**. If correctness and convenience conflict, correctness wins.

2 Economic Model: XRP-Only Finality

Purpose (non-technical). XRPlay is engineered so that **only one thing is money**: validated, native **XRP** on **XRPL**. Everything else (metrics, counters, manifests, queues) is **non-monetary state** used to compute and prove payouts.

2.1 Economic Objects and Value States

XRPlay recognizes three value states:

Final (on-ledger).
Native **XRP** on the validated ledger, represented by confirmed XRPL outcomes.

In-transit (XRPL-native).
XRPL objects that are ultimately redeemable as **XRP** under explicit conditions, such as `PaymentChannel` claims/closures and `Escrow` releases.

Non-final (off-ledger, non-monetary).
Sealed manifests, eligibility flags, access records, playback metrics, trial/ad budgets, and internal queues. These **MUST** never be represented as funds, balances, credits, or value.

2.2 Normative Rule: No Internal Money

- XRPlay **MUST** not create any unit, point, credit, or balance that can function as money.
- XRPlay **MUST** not display any listener-facing number that implies **XRP** accrual, earnings, or redeemable value.
- Any internal accounting counters **MUST** be explicitly labeled and treated as **non-monetary** system state.
- Any attempt to map non-final state into a transferable or redeemable form is **out-of-spec**.

2.3 Deterministic Pricing and Allocation (Rights-Holder Only)

XRPlay computes payouts using a pinned pricing function and pinned Rights Bundles. The only payout recipients are verified rights-holders.

Illustrative baseline (informative). Let `DROPS_PER_SECOND` be policy-pinned and let eligible seconds aggregate per track and window. Then:

$$\texttt{gross\_drops} = \texttt{eligible\_seconds} \times \texttt{DROPS\_PER\_SECOND}.$$

Gross drops are split according to the policy-pinned platform fee and the rights-holder split table.

Destination	Drops	Share
Platform fee pool	0.10	10%
Rights-holder pool	0.90	90%

Normative constraints.

- All arithmetic **MUST** be integer or fixed-scale; floating point is forbidden.
- Rounding and residual handling **MUST** be deterministic and policy-pinned.
- Any rights ambiguity **MUST** withhold payouts rather than guess.

2.4 Payment Channels: Authorization, Not Micropayments

Payment Channels are used to support high-frequency authorization without per-stream ledger writes. XRPlay **MUST** treat channel operations as settlement primitives, not as product features.

- Clients **MUST** not rely on on-ledger per-stream payments.
- Channel updates are signed off-ledger and only claimed/closed within settlement windows.
- Any claim/closure **MUST** map to a sealed manifest and audited income attribution.

2.5 Budgets and Subsidy Modes (Fail-Closed)

XRPlay may fund listening via bounded pools (e.g., trial, ad-supported). These pools are internal controls, not user entitlements.

- Pools **MUST** have explicit daily caps (`TRIAL_DAILY_CAP_DROPS`, `AD_POOL_DAILY_CAP_DROPS`) recorded in policy pins.
- Budget exhaustion **MUST** fail closed: no negative pools, no backfilled "make-good" accounting.
- Pool-funded listening **MUST** remain subject to the same fraud and eligibility gating as paid listening.

2.6 Listener Access Is Not an Economic Instrument

Listener purchases provide time-limited access to playback and do not create a financial claim.

- Listener access records **MUST** be non-transferable and non-redeemable.
- Listener access **MUST** not generate payouts, rebates, credits, or economic participation.

- Any listener-facing artifact **MUST** be treated as a UX convenience only and **MUST** not resemble a store of value.

2.7 Fail-Closed Settlement Principle

If a condition required for correctness is missing or uncertain, value movement halts.

- Uncertain ledger outcomes **MUST** halt further signing within 60 seconds.
- Rights drift, policy-hash mismatch, or reconciliation delta **MUST** trigger safe-mode.
- Recovery **MUST** preserve the sealed manifest and resolve truth from validated XRPL state.

3 Wallet and Funding Model (Mode A / Mode B)

Purpose (non-technical). XRPlay supports two wallet modes to meet different user expectations while preserving one invariant: **value is always native XRP on XRPL**. No mode introduces internal credits, platform money, or tokenized claims.

3.1 Mode A: Self-Custody (User-Owned XRPL Account)

- The user controls private keys; XRPlay never has unilateral signing authority.
- Funding actions are performed by the user's wallet (OS wallet, hardware wallet, or compatible signer).
- XRPlay **MUST** not custody user keys, and **MUST** not present any custody-like guarantees in Mode A.
- Lost keys imply loss of access to on-ledger funds; XRPlay may suspend the application account but cannot reverse ledger state.

3.2 Mode B: Managed Balance (Custodial, XRP-Backed 1:1)

Mode B is offered only where custody operations are permitted and operationally supported.

- Customer funds are held in segregated **XRP** custody accounts under multi-sign controls.
- **Backing invariant:** liabilities **MUST** be backed 1:1 by designated on-ledger assets at defined cut-times.
- **No fiat on books:** Mode B operations **MUST** avoid holding fiat balances as a product feature; any fiat rails (if used) are handled by approved partners outside the XRPlay balance sheet.
- Withdrawals are raw **XRP** to any destination address approved by compliance and destination verification.

3.3 Mode B Proof-of-Backing (Normative)

Define a liability cut-time t_c that maps to a specific validated ledger index h_c and an internal, immutable liability snapshot.

Let:

$$L(t_c) = \sum_{u \in \mathcal{U}} \texttt{liability\_drops}(u; t_c), \quad A(h_c) = \sum_{a \in \mathcal{A}} \texttt{on\_ledger\_drops}(a; h_c)$$

Then the invariant $A(h_c) \geq L(t_c)$ **MUST** hold for every cut-time.

Breach response (normative). If $A(h_c) < L(t_c)$, XRPlay **MUST**:

- block Mode B withdrawals within 900 seconds,
- halt replenishments related to Mode B flows,
- page Treasury Ops + Security + Compliance,
- generate an evidence package (snapshot IDs, ledger index h_c, account set $\mathcal{A}$, variance Δ),
- resume only after two-person approval and documented remediation.

3.4 Funding and Settlement Paths (Both Modes)

Normative rule: no per-stream on-ledger payments. Playback and metering are off-ledger at scale; the ledger is used for settlement finality only.

- **Mode A:** users fund via direct **XRP** on-ledger actions under user keys.
- **Mode B:** users fund via approved partner rails; XRPlay credits no internal "money" units; funds are represented as backed **XRP** liabilities only.
- In both modes, any payout to rights-holders is executed only from sealed manifests and confirmed XRPL outcomes.

3.5 Display Semantics (Apple-Clean UX Rule)

To avoid confusing users (and auditors) about what is and is not money:

- The client **MUST** display **access state** (e.g., "active", "expires on", "days remaining") for listener products.
- The client **MUST** display **confirmed settlement only** for rights-holders (e.g., "settled", with XRPL tx hashes available in statements).
- The client **MUST** not display internal estimates as balances, earnings, credits, or pending **XRP**.
- Any "pending" state **MUST** be explicitly framed as non-monetary processing state (e.g., "processing verification") and **MUST** not show an amount unless it is already backed by a confirmed on-ledger object.

3.6 Wallet Mode Switching (Normative)

- Switching modes **MUST** require settling to zero for any Mode B liabilities and closing or migrating relevant channel objects.
- A mode switch **MUST** create a new immutable onboarding record (new `wallet_mode_epoch_id`) for audit continuity.
- Mode switching **MUST** not alter historical manifests, rights bundles, or prior settlement truth.

4 Playback Verification, Metering and Fraud Controls

Purpose (non-technical). Every credited second can become a payout to rights-holders. Therefore playback verification is treated as a **financial control system**, not a popularity counter. Clients are adversarial; the server is authoritative. If anything is uncertain, the system withholds value **before** settlement (no pay-then-clawback assumption).

4.1 Verified Playback Event (VPE)

A **Verified Playback Event (VPE)** is the atomic unit accepted by the Settlement Engine. Only the Streaming Core may mint a VPE. The VPE stream is append-only; corrections emit a new VPE that references the prior event via `supersedes_id`.

4.2 VPE Canonical Schema (Normative Minimum)

Each VPE **MUST** be canonically serializable and contain, at minimum:

- `vpe_id` (deterministic ID; content-addressed or derived from pinned inputs),
- `wallet_id` (pseudonymous internal ID) and `device_id`,
- `track_id`, `segment_range` (or time interval), and `credited_seconds`,
- `session_id` and `server_ts` (authoritative time),
- `ip_geo_bucket` (coarse) and `asn_id` (abuse clustering),
- `proof_bundle_hash` (hash of proof inputs),
- `risk_score` and `eligibility_status` (`ELIGIBLE|QUARANTINED|BLOCKED`),
- `supersedes_id` (optional) for corrections.

Floating-point fields are forbidden.

Determinism rule (normative). Given identical server-observed inputs and identical pinned policy versions, VPE generation **MUST** be deterministic.

4.3 Minting Rules (Fail-Closed)

- The client **MUST** never assert credited seconds; it may only request segments and report local state.
- If any required proof signal is missing, stale, inconsistent, or unverifiable, the Streaming Core **MUST** set `eligibility_status=QUARANTINED` and **MUST** not mint an `ELIGIBLE` VPE.
- A VPE **MUST** bind to a server-side session state machine (start → progress → close). Out-of-order, replayed, or duplicated progress events **MUST** be ignored or quarantined.
- Concurrency is server-enforced: `MAX_CONCURRENT`=1 per wallet unless policy explicitly allows otherwise; violations **MUST** downgrade eligibility.
- Territory eligibility **MUST** be evaluated prior to `ELIGIBLE` minting; unlicensed seconds are quarantined with reason `TERRITORY_UNLICENSED`.

4.4 Anti-Abuse Controls (Normative)

XRPlay assumes hostile automation and must defend against: bot farms, emulators, replay crews, stolen-account rings, and concurrency abuse. Required controls:

- **Rate limits:** per-account, per-device, per-IP, per-ASN, per-content.
- **Cluster detection:** device → account → payment-instrument → IP/ASN graphs; dense reuse auto-quarantines.
- **Timing coherence:** server-authoritative progression checks (segment fetch cadence, seek patterns, dwell times).
- **Device integrity signals:** where supported, use attestation; where not supported, degrade trust and tighten caps.
- **Entropy checks:** unattended playback signatures (uniform timing, emulator traits, low interaction entropy) **SHOULD** be down-weighted or quarantined.

4.5 Segment Security and Capture Resistance (Normative)

- Segments **MUST** be short-lived, per-session encrypted, and bound to a server-issued access token.
- Token reuse across sessions **MUST** be rejected.
- Repeated segment-request patterns without coherent progression **MUST** trigger `BLOCKED` or `GRAYLISTED` status.
- Content must be forensically watermarked at delivery; watermark extraction is used for piracy enforcement.

4.6 Eligibility Thresholds (Policy-Pinned)

- `QUARANTINE_THRESHOLD` (default 0.70): hold for adjudication.
- `BLOCK_THRESHOLD` (default 0.90): discard + graylist device/account.

Thresholds and reason codes are policy-pinned and recorded for replay.

4.7 Quarantine, Graylisting, and Adjustment Batches

- **Quarantine lane:** max `HOLD_HOURS` (default 48); auto-reject on timeout.
- **Graylist lane:** ineligible until re-evaluation; expiry `GRAY_MAX_HOURS` (policy-pinned).
- **Adjustment batches:** the *only* mechanism for post-window corrections; must reference the original `vpe_id` set and `batch_id`.

4.8 Settlement Eligibility Guarantee (Normative)

The Settlement Engine **MUST** accept only VPEs with `eligibility_status=ELIGIBLE`. `QUARANTINED` and `BLOCKED` VPEs are non-payable by construction. Silent retroactive edits are forbidden; any correction must be an explicit adjustment batch.

5 Rights Registry and Split Computation

Purpose (non-technical). Rights and splits are **master data**. They are reviewed, approved, time-bounded, and versioned. If ambiguity exists, XRPlay withholds value rather than guess. This section defines the data model and the fail-closed rules that make rights-holder payouts replayable and audit-grade.

5.1 Rights Bundle (Normative Definition)

Definition (required). A **Rights Bundle** is an immutable, content-addressed, signed policy object that defines the payout rule-set for a single `track_id` over a specific **time window** and **territory scope**. A payout is **out-of-spec** unless the corresponding eligible VPEs resolve to *exactly one* approved Rights Bundle version at settlement time.

What it controls (required). A Rights Bundle **MUST** unambiguously bind:

- `track_id` and `content_hashes` (audio fingerprint + delivery encode hash),
- `effective_from`, `effective_to` (validity window),
- `territory_map_hash` (licensed territories and exclusions),
- `split_table_hash` (payee shares as rational/fixed-scale),
- `payee_routing_set_hash` (approved destinations + verification state),

- `policy_pin_set` (pricing, rounding, fraud thresholds, compliance pins),
- `provenance_signatures` (Rights Ops + Finance approvals; HSM-backed).

Uniqueness and non-overlap (required). For a given `track_id`, Rights Bundles **MUST** not overlap in effective window for the same territory scope. If multiple bundles match a VPE at settlement time, the batch **MUST** enter safe-mode and withhold all affected payouts.

Versioning and activation (required). Any modification to payees, shares, territory scope, pricing pins, rounding pins, fraud pins, or compliance pins **MUST** create a new `rights_version_hash`. Activation **MUST** require four-eyes approval and a minimum cooling-off period `RIGHTS_COOL_-OFF_HOURS` (default 24 h), unless an emergency override is executed with an incident record.

Payee destination verification (required). Every payout destination **MUST** be verified prior to activation (XRPL sign-challenge or custodian attestation). A destination **MUST** not be used for signing until it has passed verification and cooling-off.

5.2 Rights Bundle Minimal Schema (Informative)

```
rights_bundle_v1 = {
  track_id: string,
  rights_version_hash: bytes32,
  effective_from: rfc3339,
  effective_to: rfc3339,
  territory_map_hash: bytes32,
  split_table_hash: bytes32,
  payee_routing_set_hash: bytes32,
  policy_pins: {
    pricing_hash: bytes32,
    rounding_hash: bytes32,
    fraud_hash: bytes32,
    compliance_hash: bytes32
  },
  content_hashes: {
    master_sha256: bytes32,
    aac_sha256: bytes32,
    fingerprint_id: string
  },
  provenance: {
    rights_ops_sig: bytes,
    finance_sig: bytes,
    hsm_attestation: bytes
  }
}
```

Manifest pinning rule (normative). Every sealed settlement manifest **MUST** include the `rights_-version_hash` for each `track_id` that contributed eligible VPE seconds, so payout computation is reproducible under the same immutable inputs.

5.3 Split Tables as Contract Constants (Normative)

Split tables define payee shares and are treated as contract constants:

- Shares **MUST** be stored as reduced rational numbers or fixed-scale decimals.
- Sum of shares **MUST** equal 1.0 exactly under the pinned arithmetic.
- Any change **MUST** produce a new `split_table_hash` and **MUST** pass four-eyes approval and HSM signature.
- Floating-point arithmetic is forbidden.

5.4 Deterministic Split Computation (Normative)

Given an eligible VPE aggregate set for a window, split computation proceeds deterministically:

1. Canonical aggregation (stable sort; lexicographic keys).
2. Gross drops = credited seconds × pinned `drops_per_second`.
3. Apply split ratios using rational/fixed-scale arithmetic and the pinned rounding rule.
4. Net by (`payee_id`, `destination`) in canonical order.
5. Generate deterministic `payout_ids` and seal the Settlement Manifest.

Replay tooling **MUST** reproduce a byte-identical payout set for the same eligible inputs and pinned policy hashes.

5.5 Safe-Mode on Rights Drift (Normative)

The system **MUST** withhold payouts and enter safe-mode for any of the following:

- manifest references a non-approved `rights_version_hash` or `split_table_hash`,
- overlapping Rights Bundle windows detected for a `track_id` under the same territory scope,
- payout destination is not verified, not allow-listed, or is inside cooling-off,
- a Rights Bundle cannot be resolved uniquely for eligible VPEs.

Actions: halt affected disbursements, quarantine impacted aggregates, page Rights + Finance + Security on-call, resume only after two-person approval and documented remediation.

6 Settlement Pipeline

Purpose (non-technical). Settlement is a daily close-out: take **eligible** listening seconds, apply **pinned** rights and pricing, compute who is owed what (to the drop), seal a tamper-evident manifest, and **then** move **XRP**. If anything is uncertain—late data, fraud signals, unclear rights, compliance holds—XRPlay **withholds value** rather than pay first and argue later.

6.1 Windows, Cut-offs, and Inputs (Normative)

- **Cadence:** default `WINDOW_HOURS`=24 (policy may define other cadences; changes are hash-pinned).
- **Cut-off:** VPE eligibility snapshot occurs at `window_close_ts`. Events arriving after `window_close_ts` + `LATE_BUFFER_MIN` **MUST** be quarantined to the next window.
- **Admitted inputs:** only canonically serializable, version-pinned objects are admitted: VPE stream segments, Rights Bundles, policy pins (pricing/rounding/fraud/compliance), and verified destination set.
- **Non-compliant inputs:** malformed or non-pinned inputs **MUST** be rejected with deterministic reason codes and **MUST** not enter the eligible set.

6.2 Eligibility Snapshot and Merkle Sealing (Normative)

- At snapshot time, the Settlement Engine **MUST** freeze the set of eligible VPEs (`eligibility_status=ELIGIBLE`).
- The engine **MUST** compute `vpe_root` (Merkle root) over the canonicalized eligible VPE IDs.
- The engine **MUST** persist the snapshot boundaries and `vpe_root` in WORM storage *before* any payout signing is permitted.

6.3 Batch Construction (Deterministic Algorithm)

The Settlement Engine executes a strict deterministic pipeline:

1. **Resolve rights:** map each eligible VPE to exactly one approved `rights_version_hash`. If any VPE cannot be resolved uniquely, affected aggregates **MUST** be withheld and the batch **MUST** enter safe-mode.
2. **Aggregate:** canonical aggregation of eligible seconds by (`track_id`, territory bucket, policy pins).
3. **Price:** compute `gross_drops` using integer/fixed-scale arithmetic only:

$$\texttt{gross\_drops} = \texttt{credited\_seconds} \times \texttt{DROPS\_PER\_SECOND}_{\texttt{pricing\_hash}}.$$

4. **Allocate:** apply split tables and pinned rounding; compute residuals deterministically.
5. **Net:** net payout lines by (`payee_id`, `destination`) in canonical order.
6. **ID assignment:** produce deterministic `batch_id` and per-line `payout_id` (domain-separated).
7. **Seal:** emit the Settlement Manifest, compute `manifest_root`, and sign it with the HSM.

Determinism guarantee (required). Identical eligible inputs + identical pinned versions **MUST** yield byte-identical manifests (including ordering, roots, and IDs) across regions, architectures, and retries.

6.4 Settlement Manifest (Normative Requirements)

A sealed manifest **MUST** include, at minimum:

- `window_id`, `window_close_ts`, and snapshot boundaries,
- `vpe_root` and eligible VPE count,
- policy pin hashes: `pricing_hash`, `rounding_hash`, `fraud_hash`, `compliance_hash`,
- referenced rights: per-`track_id` `rights_version_hash` set (or a Merkle root of the mapping),
- payout lines: (`payout_id`, `payee_id`, `destination`, `amount_drops`, `status`),
- residual policy and any deterministic sweep outputs,
- HSM signature and schema version identifier.

6.5 Execution: Funds In and Funds Out (Normative)

Inbound (listener payments). Where Payment Channels are used, inbound authorization and claims **MUST** be mapped to the settlement window via:

- `channel_id` and `channel_state_hash` recorded in the manifest,
- validated-ledger confirmation of claim/close outcomes.

Outbound (rights-holder payouts).

- The payout executor **MUST** be manifest-bound: it may sign only if (`manifest_hash`, `payout_id`) exists and the destination and amount match exactly.
- The executor **MUST** enforce caps, velocity limits, destination allow-lists, and compliance holds at the signing boundary.
- The executor **MUST** never re-sign a payout transaction once the manifest is sealed.

6.6 Idempotency and Sequence Safety (Normative)

- Each payout line item **MUST** have a deterministic idempotency key derived from `(manifest_hash || payout_id || destination || amount_drops)`.
- Each signing account **MUST** maintain durable **sequence leases** binding: `sequence`, `payout_id`, signed tx hash, and final status.
- On uncertainty, the executor **MUST** resolve status from the validated ledger and either (a) resubmit the identical signed blob or (b) hold and enter safe-mode.
- It is **forbidden** to sign a different transaction for the same payout line item.

6.7 Reconciliation and Window Close-Out (Normative)

A settlement window is `COMPLETE` only when all of the following hold:

- the manifest is sealed and HSM-signed,
- inbound channel outcomes (if applicable) are validated on-ledger,
- outbound payouts are confirmed on the validated ledger or placed into explicit `HELD` / `FAILED_FINAL` states with reason codes,
- reconciliation delta is zero (or within an explicitly booked, policy-bounded float tolerance).

Reconciliation delta (required). Define the reconciliation delta as:

$$\Delta = \texttt{manifest\_total\_drops} - \sum \texttt{confirmed\_drops} - \sum \texttt{held\_drops} - \sum \texttt{failed\_final\_drops}.$$

Δ **MUST** equal zero, or be explicitly booked under a hash-pinned float policy with documented reason codes. Any other outcome **MUST** trigger safe-mode.

6.8 Isolation Lanes: Quarantine vs. Global Halt (Normative)

- **Batch isolation:** if a subset of tracks/payees fails rights resolution, compliance, or destination verification, the system **MUST** quarantine only the affected aggregates and continue healthy payouts.
- **Global safe-mode:** the system **MUST** halt all outbound value within 60 s if any of the following occurs:
  - manifest replay mismatch,
  - reconciliation delta outside tolerance,
  - signing policy breach or anomalous signing pattern,
  - rights drift involving approved hashes,
  - validated-ledger uncertainty beyond `UNCERTAINTY_TIMEOUT_MIN`.

6.9 Adjustment Batches (Only Correction Mechanism)

- Post-settlement corrections **MUST** occur only via explicit **adjustment batches** that reference the original `window_id` and `batch_id`.
- Silent edits to historical VPEs, manifests, rights bundles, or payout lines are forbidden.
- Adjustment batches **MUST** be deterministic, sealed, signed, and reconciled using the same rules as standard batches.

7 XRP-Only Economic & Settlement Model

Plain-language overview (non-technical). XRPlay has one monetary unit: **native XRP on XRPL**. Everything else in the system is either (i) a **consumptive access state** (XRPass), or (ii) a **deterministic accounting artifact** used to compute and audit rights-holder payouts. No listener-facing state is redeemable, convertible, or representable as money.

7.1 Economic Objects (Normative)

XRPlay recognizes exactly the following economic objects:

- **XRPass (Listener access)** — time-limited playback authorization only; consumptive; non-transferable; expires.
- **VPE (Verified Playback Event)** — append-only metering unit; eligibility-gated; used only for payout computation.
- **Settlement Manifest** — sealed, deterministic payout instruction set; replayable; immutable once sealed.
- **XRPL Execution Artifacts** — `PaymentChannel`, `Payment`, `Escrow` used strictly for settlement finality.

Prohibited (out-of-spec). Any platform-issued token, points, credits, rebates, “rewards,” yield, or internal balance presented as economic value is out-of-spec.

7.2 Value States and Representations (Normative)

XRPlay distinguishes three value states:

- **Final** — validated **XRP** on-ledger.
- **In-transit** — XRPL-native objects redeemable as **XRP** (e.g., `PaymentChannel`, `Escrow`) with validated linkage to a sealed manifest.
- **Non-final** — internal manifests, queues, estimates, access states, and analytics; **MUST NOT** be described, displayed, or implied as funds.

UI rule (required). The product **MUST** not display any listener “balance,” “earnings,” “credits,” or **XRP**-derived value attributable to listening. Listener UI **MUST** display only access semantics (e.g., `access_active`, `expires_on`).

7.3 Deterministic Per-Second Allocation (Normative)

Pricing is expressed as a pinned integer rate in drops per verified second:

$$\texttt{gross\_drops} = \texttt{eligible\_seconds} \times \texttt{DROPS\_PER\_SECOND}.$$

All arithmetic **MUST** use integer or fixed-scale decimal; floating point is forbidden.

Allocation split (required). Given `PLATFORM_FEE_BPS` and a rights split table, the allocation is:

- platform fee pool receives `gross_drops` $\times$ `PLATFORM_FEE_BPS`/10,000,
- rights-holder pool receives the remainder and is then split deterministically per the pinned split table.

Residual drops are handled per the pinned residual policy (Section 17).

7.4 Revenue Sources and Plan Semantics (Normative)

XRPlay supports the following metering modes, each with a pinned budget and fail-closed behavior:

- **paid** — user-funded via `PaymentChannel` authorization.
- **trial** — platform-funded, capped by `TRIAL_DAILY_CAP_DROPS`.
- **ad-supported** — platform-funded pool, capped by `AD_POOL_DAILY_CAP_DROPS`.

Budget exhaustion (required). If a mode's budget is exhausted, the system **MUST** fail closed for that mode (no negative pools, no backfilling, no discretionary make-goods).

7.5 Settlement Truth and Finality Mapping (Normative)

A payout is considered executed only when:

- it is present in a sealed manifest (identified by `manifest_hash`),
- it is bound to a deterministic `payout_id`,
- and it maps to a validated XRPL outcome (tx hash + result code) associated to that `payout_id`.

No optimistic accounting (required). Internal records **MUST** not be treated as payment completion. Completion is defined by validated-ledger confirmation only.

7.6 Economic Risk Controls (Fail-Closed)

Disbursements **MUST** halt on any of the following:

- trial/ad pool cap breached,
- rights or split-table mismatch,
- fee ceiling exceeded,
- reconciliation delta outside pinned tolerance,
- manifest replay mismatch or signing-policy breach.

Resume discipline (required). Resume **MUST** require an evidence package and two-person approval, consistent with Sections 8 and 9.

8 Security Architecture (Keys, Signing, and Controls)

Plain-language overview (non-technical). XRPlay assumes apps, devices, and internal systems can be compromised. The design ensures **no single failure** can move funds incorrectly. High-authority keys are offline by default; routine wallets are capped; and every outbound payment is signed only if it matches a sealed, replayable manifest. If anything looks wrong, the system halts payouts first and escalates for review.

8.1 Wallet Segmentation and Exposure Controls

- **Cold Treasury** — offline by default; highest quorum; break-glass only.
- **Warm Treasury** — restricted online; replenishment-only; no direct payee disbursements.
- **Hot Settlement** — routine disbursement; strict per-transaction and per-epoch caps.
- **Mode B Custody** — customer-backed; segregated accounts; multi-sign; liabilities proven against validated ledger.

Exposure limits are enforced at the signing boundary: `HOT_MAX_DROPS`, `HOT_TX_MAX_DROPS`, velocity caps, and destination allow-lists. **Deny-by-default:** an outbound destination **MUST** be explicitly verified and allow-listed before any payment can be signed.

8.2 Signing Boundary (HSM + Policy Enforcement)

- **Key protection:** non-exportable HSM keys (custody-grade), tier-scoped key material, phishing-resistant MFA, and rotation drills at least semi-annually.
- **Policy enforcement at signer:** caps, velocity limits, allow-lists, quorum thresholds, and compliance holds are enforced *before* any signature is produced.
- **Separation of duties:** rights operations, settlement execution, and signing-policy administration **MUST** run under separate infrastructure boundaries with independent audit logs and change-control.

8.3 Manifest-Bound Signing (Required)

A settlement payout **MUST** be signed only if all conditions below hold:

1. The request references a sealed `manifest_hash`.
2. The referenced `payout_id` exists in that manifest.
3. Destination and amount **exactly** match the manifest line-item for `payout_id`.
4. The manifest is valid under the pinned schema version and signature policy.

5. The payout is not HELD by compliance, sanctions, dispute, or policy gates.
6. The request does not violate caps, velocity, or quorum requirements.

No re-sign rule (normative). If a transaction outcome is uncertain, the system **MUST** resolve it from validated-ledger state and sequence leases. It **MUST** not re-sign the payout (new tx blob) unless an explicit break-glass procedure is executed with an incident record and two-person approval.

8.4 Destination Verification and Allow-Lists

- **Verification:** each payee destination **MUST** pass a verification challenge (e.g., XRPL sign-challenge) or a custodian attestation.
- **Cooling-off:** newly verified destinations **MUST** observe a minimum cooling-off period `DEST_COOL_OFF_HOURS` (default 24 h) before first payout, unless break-glass is invoked.
- **Rotation control:** destination changes are treated as high-risk changes; repeated churn triggers compliance review and optional payout holds.

8.5 Immutable Evidence and WORM Logging

Every signing request **MUST** be logged as an immutable record containing, at minimum:

- request ID, caller identity, and timestamp,
- `manifest_hash`, `batch_id`, `payout_id`,
- destination, amount, and policy decision (`ALLOW|DENY|HOLD`) with reason codes,
- signed transaction hash / blob identifier,
- XRPL submission status and validated outcome mapping when available.

Logs are retained in tamper-evident, WORM-grade storage.

8.6 Memo and Data Minimisation at the Signing Layer

- Memo fields (if any) **MUST** contain no PII and **MUST** be limited to schema-pinned identifiers (e.g., `manifest_hash`, `batch_id`, `payout_id`).
- Memo size **MUST** be bounded (e.g., ≤ 1 kB) and deterministic by schema version.
- Any non-conforming memo structure **MUST** be rejected by the signer.

8.7 Operational Safe-Mode (Security)

Triggers (SLA $\leq$ 60 s).

- cap/velocity/allow-list breach,

- manifest-binding failure (missing `manifest_hash` / invalid `payout_id` / mismatch),
- anomalous signing patterns (volume, destinations, timing),
- suspected key compromise or policy bypass,
- sustained XRPL non-retriable failures affecting confirmation mapping.

Actions (required).

- Halt payouts and warm→hot replenishment.
- Page Treasury + Security + Compliance on-call owners.
- Produce an incident evidence package (sequence leases, last 100 signing decisions, affected manifests/batches).
- Resume only after two-person approval with documented remediation.

9 Compliance, Policy and Reporting

Plain-language overview (non-technical). Compliance gates are part of the payment path and cannot be bypassed by product, ops, or engineering. If a rule is unclear, incomplete, or violated, value movement halts *before* signing—never "send then unwind." Compliance decisions are policy-pinned, replayable, and retained under audit-grade storage.

9.1 Control Points (Normative)

C1 — Onboarding
Identity capture, jurisdiction classification, wallet-mode assignment, and risk tiering.

C2 — Pre-settlement
Territory licensing eligibility, fraud eligibility, sanctions screening freshness, and budget-eligibility gates.

C3 — Pre-signing
Final sanctions check, routing decision where required, limits gate, and hold-state enforcement.

Each control point **MUST** reference immutable policy hashes recorded in every sealed manifest.

9.2 Identity Requirements (Payees)

Rights-holders and payout recipients **MUST** have a verified identity record sufficient for payout compliance: legal name, jurisdiction, tax residency, beneficial-owner attestation where required, destination verification state, and risk tier. Listeners may remain pseudonymous except where required for Mode B or jurisdictional compliance.

9.3 Sanctions Screening and Hold States (Fail-Closed)

Outbound payments are screened against applicable sanctions lists and internal risk rules. Any hit, partial match, or inconclusive result **MUST** set `status=HELD` and block signing/submission.

Hold objects (required). Holds are first-class, immutable records with: `hold_id`, `payout_id` or `payout_family_id`, reason code, screening snapshot ID, and reviewer approvals. Release requires two-person approval and a documented rationale.

9.4 Routing Strategy and Threshold Handling

XRPlay adopts one documented routing strategy per jurisdiction and applies it consistently:

1. **Intermediary routing:** route via an approved service provider where required by local rules.
2. **Threshold-aware batching:** aggregate where legally permitted and operationally reasonable (no evasion intent; preserve attribution).
3. **Manual-release lane:** hold and release only after a complete evidence package is reviewed.

Attribution is preserved via `payout_family_id` mappings in the manifest and in payee statements.

9.5 Tax and Rights-Holder Reporting

Payee statements **MUST** include:

- gross and net amounts in drops and **XRP**, platform fees, and residual handling,
- settlement windows and `batch_id` list,
- validated XRPL transaction hashes and result codes,
- holds with reason codes and release metadata,
- adjustment batches with forward/backward linkage to original `batch_id`.

FX capture (when required). Where fiat reporting is required, FX rates are captured deterministically at `window_close_ts`, versioned, and stored in the manifest.

9.6 Data Minimisation and On-Ledger Privacy

- Memos **MUST** be ≤ 1 kB and contain only schema-pinned identifiers (`manifest_hash`, `batch_id`, `payout_id`); no PII and no rights terms.
- Rotate settlement source addresses by epoch; bounded jitter reduces linkability without breaking audit continuity.
- Off-ledger compliance data is encrypted at rest and in transit; RBAC is reviewed quarterly; secrets are prohibited in logs.

9.7 Consumer Product Guardrails (Access-Only, Non-Financial)

Consumer access products (e.g., XRPass) are consumptive by design and **MUST** not be marketed or implemented as financial instruments:

- No transferability, resale, secondary markets, or portability.
- No listener payout, revenue share, yield, "rewards," or economic participation.
- Any post-expiry state is a read-only account record (non-portable, non-redeemable) and **MUST** not carry economic value.
- Prohibited language (product + marketing): `invest`, `profit`, `yield`, `roi`, `floor`, `rarity`, `collectible`, `mint`, `holder`, or obfuscated variants.

9.8 Compliance Safe-Mode and Incident Response

Triggers.

- sanctions hit or inconclusive screening,
- routing path unavailable where required,
- anomalous destination churn, impersonation, or destination-verification failure,
- regulator-requested freeze or legal hold.

Actions (required).

- Set `status=HELD` for affected payouts and block signing.
- Produce an evidence package (screening snapshot IDs, routing decision log, affected manifests/batches).
- Resume only after written Legal/Compliance clearance under two-person approval.

10 User Dynamics (Artist / Listener)

Plain-language overview (non-technical). XRPlay sells **time-limited playback access** and nothing else. Listener-facing products are **consumptive by design**: ad-free streaming for a fixed period with **zero** financial utility. XRPlay does not provide fan identity products, wallet lists, or "proof-of-fandom" features.

10.1 Design Objectives (Normative)

AO-1 **Access-only.** XRPass confers playback access only.

AO-2 **No ownership.** No title, license assignment, or resale rights are conveyed to listeners.

AO-3 **No listener economics.** No payouts, points, rebates, revenue share, yield, rewards, or economic participation.

AO-4 **Non-transferable.** Any attempt to transfer access (account sale, credential sharing, resale markets) **MUST** revoke access and invalidate the XRPass.

AO-5 **Expiry revokes access.** At expiry, playback access ends; no post-expiry value remains.

AO-6 **No chance, no prize, no status.** No lottery odds, random rewards, weighted votes, or status mechanics tied to purchase.

10.2 XRPass Access Types

- **Standard XRPass** — unlimited issuance (policy-controlled); artist-defined price; unlocks playback access for 12 months.
- **Exclusive XRPass** — capped issuance (policy-defined); unlocks the **same** playback access for 12 months. Optional inclusion is limited to **non-economic, non-transferable** liner-note style metadata (e.g., commentary text). **Prohibited**: downloadable files, stems, gated communities, collectibles, perks, or any feature that creates ongoing utility beyond access.

10.3 Primary Sale Flow (Artist → Listener)

1. Artist selects access type and price.
2. Mandatory disclosure (hash-pinned):

   > "You are purchasing 12 months of ad-free streaming access only. This product is not transferable, creates no ownership, and cannot be used for profit."

3. Compliance checks as required by jurisdiction and payment rail.
4. Payment capture and settlement per wallet mode (Mode A or Mode B).
5. Access activates. Refunds/cancellations only where required by law and handled off-ledger.

10.4 Expiry Semantics (Normative)

- Playback keys are revoked; access ends.
- Any listener-facing XRPass artifacts **MUST** be removed from the product UX (no badge shelf, no "proof," no status marker).
- Internal records retained solely for audit, fraud prevention, dispute handling, and legal obligations are pseudonymized, are not user-queryable features, and **MUST** not be marketed or framed as user benefit.
- Repurchase later confers **no** priority, status, or advantage.

10.5 Artist Access to Listener Data (Prohibited)

- No endpoint returns listener wallets, purchaser lists, or wallet-level cohorts.
- No fan-identifying or fan-analytics products are offered to artists or labels.

10.6 Marketing Copy Restrictions (Normative)

Forbidden terms include: `invest`, `yield`, `royalty`, `governance`, `upside`, `profit`, `floor`, `rarity`, `collectible`, `mint`, `holder`, emojis, or obfuscated variants.

Required footer (auto-inserted):

> "This product provides streaming access only. It is not transferable and creates no ownership or financial entitlement."

10.7 Economic Guardrails (Access-Only)

- XRPass pricing **MUST** be policy-bounded: per-item price caps and daily sales caps.
- A kill-switch **MUST** disable XRPass offers if any cap is breached.
- Per-wallet and per-device rate limits mitigate automation and bulk abuse.

11 Media Storage & Delivery Economics

Plain-language overview (non-technical). Every uploaded minute is a **cost center** and therefore a **financial control surface**. XRPlay treats storage like collateral: uploaders must maintain a deterministic reserve that covers future footprint. When reserves fall short, the platform **fails closed**: new uploads are rejected and inactive assets are tombstoned in a deterministic order. A reserve top-up re-activates tombstoned assets without changing rights or hashes.

11.1 Codec & Bit-Rate Invariants (Normative)

- **Master ingest:** WAV 44.1 kHz / 16-bit or FLAC 44.1 kHz / 16-bit (24-bit allowed, policy-gated).
- **Delivery encode: AAC-LC**, constant **256 kbps**, 44.1 kHz, stereo.
- **Encode-once:** each track is transcoded once; compute `aac_sha256` and pin it in the Rights Bundle.
- **Serve invariant:** the CDN **MUST** serve **only** objects whose `aac_sha256` matches the pinned value.

Any encode drift ($\texttt{served\_aac\_sha256} \neq \texttt{pinned\_aac\_sha256}$) is **out-of-spec** and **MUST** halt serving for that asset.

11.2 Storage Tiers

A. **Hot cache** — tracks with eligible seconds ≥ `ACTIVE_THRESHOLD` within last `ACTIVE_WINDOW_DAYS`; served from CDN edge.

B. **Cold archive** — all other tracks; glacier-class object storage; restoration occurs on demand *only after* reserve sufficiency is re-validated.

11.3 Upload Admission & Economic Controls (Normative)

Tiered quotas are measured in **minutes** (not track count):

- `catalog_minutes`$(u) \leq$ `CATALOG_MINUTES_MAX[tier]`,
- daily ingest minutes $\leq$ `INGEST_MINUTES_PER_DAY_MAX[tier]`,
- daily uploads $\leq$ `UPLOADS_PER_DAY_MAX[tier]`,
- concurrent transcodes $\leq$ `TRANSCODE_CONCURRENCY_MAX[tier]`.

Deterministic de-duplication (required).

- `master_sha256` exact-match → soft-reject (idempotent reference; no double storage).
- audio-fingerprint similarity → review queue (`status=HELD_DUPLICATE_RISK`); not served until cleared.

11.4 Storage-Reserve Economics (Tier 0)

Tier 0 uploaders **MUST** maintain a forward reserve that covers a policy-defined horizon (default 30 days):

$$\texttt{reserve\_drops}(u) = \left(\sum_{t \in \mathcal{T}(u)} \texttt{bytes\_stored}(t) \right) \cdot \texttt{RESERVE\_DROPS\_PER\_BYTE}$$

Let `balance_drops(u)` be the uploader's storage reserve balance. If $\texttt{balance\_drops}(u) < \texttt{reserve\_drops}(u)$, the system **MUST**:

1. reject new uploads for u,
2. tombstone inactive assets for u in a deterministic order until $\texttt{balance\_drops}(u) \geq \texttt{reserve\_drops}(u)$,
3. require a reserve top-up to re-activate tombstoned assets.

Deterministic tombstone order (required). Tombstone selection is stable and replayable, ordered by:

$$(\texttt{last\_eligible\_vpe\_ts}, \texttt{bytes\_stored}, \texttt{track\_id})$$

(ascending), so two independent replays tombstone the same set.

11.5 Retention, Tombstoning, Re-activation

A tombstoned track retains metadata, hashes, and rights linkage but **is not served**. Re-activation requires:

- reserve top-up, and
- re-validation that archived object hash matches pinned `aac_sha256`.

Re-activation **MUST** not modify Rights Bundles, split tables, or historical manifests.

11.6 Abuse Scenarios & Controls (Normative)

XRPlay assumes adversarial uploaders and adversarial listeners. The following behaviors are **in-spec threats** and **MUST** be controlled.

A1 — Restore-thrashing (cold→hot churn). Attack: automated play bursts force repeated cold restores to inflate egress and retrieval costs.
Controls (required):

- per-track restore cooldown: `RESTORE_COOLDOWN_MIN` (default 60 min),
- per-uploader restore budget: `RESTORE_BUDGET_DROPS_DAY` (policy-pinned),
- restoration requires reserve sufficiency and **eligible** VPEs only (quarantined seconds do not trigger restore),
- on breach: throttle restores, keep asset cold, emit `RESTORE_THROTTLED` event.

A2 — CDN hotlinking / token replay. Attack: sharing direct media URLs to off-platform audiences to externalize bandwidth cost.
Controls (required):

- signed URLs with short TTL: `URL_TTL_SECONDS` (default 60–180),
- per-session key binding; origin checks; mandatory TLS; cache-key includes session nonce,
- suspicious referrers / ASN spikes → denylist + session revocation,
- repeated violations → account graylist and enforcement.

A3 — Partial-upload bombs / transcode pipeline abuse. Attack: repeated malformed uploads or deliberately expensive-to-parse inputs to exhaust CPU and queue slots.
Controls (required):

- staged validation: validate container + duration + codec headers *before* transcode slot allocation,
- hard per-file limits: `MAX_UPLOAD_BYTES`, `MAX_DURATION_SECONDS`,
- deterministic failure reasons (`REJECT_MALFORMED`, `REJECT_OVERSIZE`, `REJECT_DURATION`),
- failure-rate breaker: if `REJECT_RATE` exceeds threshold, disable ingest for `COOLDOWN_HOURS`.

A4 — De-duplication evasion. Attack: tiny perturbations to avoid hash matches while duplicating catalog to inflate storage footprint.
Controls (required):

- dual-keying: exact `master_sha256` + robust audio-fingerprint gate,
- near-duplicate detection → `status=HELD_DUPLICATE_RISK` and manual lane,
- repeated attempts → tier downgrade or uploader suspension.

A5 — Quota circumvention and multi-account farms. Attack: spread uploads across many accounts to bypass minutes/ingest caps.
Controls (required):

- KYC/identity binding for uploader tiers; device/payment-instrument clustering,
- per-UBO aggregate caps where required by policy,
- suspicious linkage → unified quota enforcement and audit flag.

11.7 Cost Governance (Fail-Closed)

- Daily reconciliation books CDN egress cost in drops.
- Enter safe-mode if projected CDN spend exceeds `MAX_CDN_COST_BPS` of the epoch fee pool.
- In safe-mode: throttle non-critical delivery, halt new uploads, and prioritize settlement-critical traffic.

Policy pinning (required). All quota, pricing, and tier constants are version-pinned under `storage_policy_hash`. Any change bumps the hash and the new hash is recorded in the next sealed settlement manifest.

11.8 Implementation Roadmap Additions

P0: transcode pipeline, `aac_sha256` pinning, hot/cold tiering, serve-hash enforcement.
P1: reserve service, deterministic tombstone ordering, cost-to-treasury plumbing.
P2: chaos tests (upload flood, reserve drain, hotlink attempts, restore thrash), restoration latency SLOs.
P3: replay verifier that recomputes bytes-stored ↔ reserve balance for any historical window.

12 Governance

Non-goals. XRPlay does **not** implement token governance, voting, staking, delegated committees, or "community" parameter control. Governance theater is a liability: it introduces ambiguity, delays, and ex-post rationalizations. XRPlay is governed like a payments system: **change control** over version-pinned policies with replayable evidence.

12.1 What can change (and what cannot)

Immutable by design (cannot change without a new system identity). The following are **system-identity properties**. Changing them is a new product, not an upgrade:

- **Cash-out rail:** native **XRP** on **XRPL** only.
- **No consumer economic participation:** XRPass is consumptive access only; no resale, no rewards, no royalties.

- **Deterministic settlement:** sealed manifests; bit-identical replay.
- **Fail-closed execution:** ambiguity triggers holds/safe-mode, not payouts.
- **No off-ledger "balances as money":** any internal accounting is deterministic, policy-pinned, and reconcilable to zero.

Change-controlled (allowed with strict evidence). These may change *only* via pinned policy upgrades:

- **XRPass pricing caps and limits:** `MAX_XRPass_DAILY`, `MAX_XRPass_PRICE_DROPS`, fee spend caps.
- **Quota and storage parameters:** tier minutes, ingest ceilings, reserve cost constants.
- **Anti-abuse thresholds:** rate limits, bot-score gates, restore cooldowns.
- **Operational SLO thresholds:** safe-mode triggers, congestion thresholds.

12.2 Policy pinning and upgrade mechanism (Normative)

Policy objects. All tunable constants are grouped into **policy objects**, each with a canonical serialization and SHA-256:

- `economic_policy_hash`
- `abuse_policy_hash`
- `storage_policy_hash`
- `ops_policy_hash`

Manifest binding (required). Each sealed settlement manifest **MUST** include the active policy hashes. A replay verifier **MUST** be able to recompute all settlement outputs given: (i) the manifest inputs, (ii) the pinned policy hashes, and (iii) the code version hash.

Upgrade rule (required). A policy upgrade is valid iff:

1. it produces a new policy hash,
2. the upgrade is signed by the **Policy Authority** keys (multi-sig),
3. the new hash is announced **before** the first epoch it affects,
4. the upgrade is recorded in an **Upgrade Ledger** (append-only log),
5. replays remain deterministic for historical epochs.

12.3 Roles and keys (Minimal)

Policy Authority (multi-sig). A small set of signers controls policy upgrades. Their only power is to publish **new pinned policy hashes** within the allowed change surface.

Ops Authority (multi-sig). Controls safe-mode toggles and emergency throttles. Ops Authority **MUST** not change economics; it can only **reduce risk** (halt/throttle), never increase it.

Separation of duties (required). Policy Authority and Ops Authority key sets **MUST** be disjoint. Any attempt to merge roles is **out-of-spec**.

12.4 Emergency controls (Fail-Closed)

Safe-mode is not discretion. Safe-mode triggers are deterministic (threshold-based) wherever possible. When human action is required, it is constrained:

- Safe-mode entry is always allowed.
- Safe-mode exit requires **two conditions**: (i) trigger condition cleared, and (ii) a signed incident report hash is posted to the Upgrade Ledger.

Freeze events. If regulators, courts, or platforms compel a freeze:

- new payouts **MUST** halt,
- holds are preserved with reason codes,
- manifests continue sealing with `status=FROZEN` and reconciliation delta tracked.

This preserves auditability while preventing outbound value movement.

12.5 Evidence and audit trail

Upgrade Ledger (append-only). Each entry includes:

- timestamp,
- signer set,
- old $\rightarrow$ new policy hash,
- reason code,
- incident report hash (if applicable).

Board assurance. The board does not need to trust narratives. It needs:

- a deterministic replay verifier,
- policy-hash provenance,
- separation-of-duties attestations,
- safe-mode logs with reason codes and timestamps.

13 Privacy, Observability and Production Operations

Plain-language overview (non-technical). XRPlay publishes the **minimum possible information on-ledger** to prove that payments occurred, while keeping identities, contracts, splits, and per-track economics off-ledger. Internally, XRPlay retains complete, encrypted records sufficient to reproduce every settlement end-to-end. Operationally, the platform runs with a **stop-the-line posture**: if economic correctness, policy compliance, or signing integrity is uncertain, outbound value halts automatically, owners are paged, and a complete evidence package is produced before any resumption.

Operating posture: minimise public linkability, maximise internal auditability, and fail closed whenever financial correctness is uncertain.

13.1 Privacy Posture (Normative)

- **On-ledger minimisation.** XRPL metadata (Memo fields if used) **MUST** contain only schema-pinned identifiers: `manifest_hash` (or truncated 256-bit form), `batch_id`, and `payout_id`; total memo payload $\leq$ 1 kB. Prohibited: legal names, payee identities, split ratios, rights terms, territory rules, per-track economics, or any PII.
- **Deterministic memo schema.** Memo structure is versioned and deterministic; any deviation **MUST** trigger signing denial.
- **Linkability reduction.** Settlement source addresses **SHOULD** rotate by epoch without breaking audit continuity; channel closures **SHOULD** be jittered within a bounded band (e.g., ±15 minutes); batch shaping **SHOULD** avoid closing more than `MAX_CHANNEL_CLOSES_PER_LEDGER` in one ledger.
- **Off-ledger confidentiality.** Rights, payee, and compliance data **MUST** be encrypted at rest (AES-256) and in transit (TLS 1.3). Access is RBAC-controlled with quarterly reviews. Secrets and private material are forbidden in logs.

13.2 Observability: SLIs and SLOs

13.2.1 Service-Level Indicators (Normative)

- `VPE_mint_latency_p95` $\leq$ 500 ms.
- `VPE_integrity_rate` $\geq$ 99.9%.
- `window_complete_latency` $\leq$ 6 h from `window_close_ts`.
- `reconciliation_delta_drops` = 0 (or explicitly booked, policy-bounded float).
- `signing_deny_rate` < 0.01%.

13.2.2 Service-Level Objectives (Minimum Acceptable)

- Settlement completeness: $\geq$ 99.9% of windows complete within 6 h.

- Reconciliation accuracy: $\geq 99.99\%$ of windows with zero unexplained delta.
- Signing coverage: 100% of signing requests logged with explicit `ALLOW` or `DENY`.

13.2.3 Instrumentation and Log Integrity

- Immutable event logs for: VPE minting, eligibility gating, manifest sealing, signing decisions, and XRPL confirmations.
- Mandatory correlation keys: `window_id`, `batch_id`, `manifest_hash`, `payout_id`, XRPL tx hash.
- Logs are stored in tamper-evident, WORM-grade storage; deletion or mutation is prohibited.

13.3 Production Operations and Incident Response

13.3.1 Runbooks (Required)

Documented and tested runbooks **MUST** exist for:

- key compromise (hot / warm / cold),
- payout anomaly or unexpected destination,
- XRPL congestion or fee spike,
- fraud or bot activity surge,
- compliance or sanctions hold surge.

13.3.2 Kill-Switch and Safe-Mode (Normative SLA: 60 s)

Automatic triggers include:

- reconciliation delta $\neq 0$,
- manifest replay hash mismatch,
- signing policy or cap breach,
- rights or split-table drift,
- sustained XRPL non-retriable submission failures.

Actions within 60 seconds:

- Halt all new outbound payments and channel claims.
- Freeze warm $\rightarrow$ hot wallet replenishment.
- Page Treasury, Security, and Compliance on-call owners.

13.3.3 Incident Evidence Package (Required within 4 h)

Each incident **MUST** produce a complete, immutable evidence bundle containing:

- incident ID, severity, and timestamps,
- affected `window_id`, `batch_id`, `manifest_hash`,
- submitted and validated XRPL transaction hashes (inbound and outbound),
- signing decision log excerpt (minimum last 100 entries),
- reconciliation delta report (including holds and failed-final lines),
- containment and halt timestamps,
- two-person recovery approval record.

13.3.4 Post-Incident Reconstruction (Normative)

- Recompute `vpe_root` and settlement manifest bit-identically.
- Reconcile every payout against validated XRPL state.
- Produce a checksum comparison of reconstructed vs. original manifest; any mismatch **MUST** trigger a new incident.
- Publish an internal audit note and update operational dashboards.

14 Failure Modes and Recovery

Plain-language overview (non-technical). When the real world gets messy—network congestion, uncertain transaction outcomes, partial payout failures, data pipeline issues, or security incidents—XRPlay never "guesses" with payments. If ledger state is unstable or a transaction result is unclear, payouts pause automatically, the sealed manifest stays unchanged, and immutable logs plus validated XRPL state determine exactly what did and did not happen. When something serious occurs (rights drift, reconciliation mismatch, key compromise), XRPlay enters safe-mode within 60 seconds, locks value movement, captures an evidence package, and resumes only after documented remediation and two-person approval.

Design principle: fail closed, preserve determinism, prevent double-pay, and remain replayable from immutable inputs.

14.1 Failure Classification (Normative)

F1 — Ledger conditions
Fee spikes, congestion, validator divergence, temporary network partitions.

F2 — Submission uncertainty
Signed transaction submitted, but final XRPL outcome is unknown.

F3 — Partial batch
Some payouts confirm while others fail or remain uncertain.

F4 — State drift
Reconciliation delta $\neq 0$ or manifest replay mismatch.

F5 — Security
Key compromise, policy bypass attempt, destination allow-list anomaly.

Normative: F4/F5 **MUST** trigger safe-mode.

14.2 Ledger Congestion & Fee Spikes

Fee caps (required).

- `MAX_FEE_DROPS_PER_TX` = 10 000 drops (0.01 XRP),
- `MAX_FEE_DROPS_PER_BATCH` = 100 000 drops (0.1 XRP).

Exceeding either cap **MUST** halt disbursement and set batch status `DEFERRED_SETTLEMENT`.

Congestion playbook (normative).

1. Freeze new outbound submissions; keep manifest sealed.
2. Prioritise: (a) manifest anchor (if used), (b) treasury safety moves, (c) payouts.
3. Exponential backoff with jitter; never re-sign blindly.
4. Resume when observed fee levels fall below caps and validated-ledger stability holds for `STABILITY_MIN`.

Extended deferral (required). If deferred beyond `MAX_DEFER_HOURS` (default 24), XRPlay **MUST** (i) publish a payee status update, (ii) generate a compliance evidence package, and (iii) hold all new windows if contractual timing would be breached.

14.3 Submission Uncertainty & Sequence Safety

Confirmation rule (normative). A payout is `CONFIRMED` only when a validated XRPL result is mapped to the corresponding `payout_id`.

Durable sequence leases (required). Each submitting account maintains a durable lease record: `sequence`, `payout_id`, signed tx hash, submission attempts, and final status.

Ledger-gap recovery (required).

1. Query validated ledger for current account sequence and recent tx history.
2. For leased sequences not observed as validated: resubmit the *identical* signed blob only.
3. Never sign a new transaction for a leased sequence until the prior item is resolved.

Uncertainty timeout (required). If unresolved beyond `UNCERTAINTY_TIMEOUT_MIN` (default 30 min), the system **MUST** enter safe-mode.

14.4 Partial Batch Failure

Invariants (normative).

- Sealed manifest remains immutable.
- Successful lines remain `CONFIRMED`; they are never re-sent.
- Failed lines retry with the same `payout_id`, destination, and amount.

Failure codes (normative).

`FAILED_FINAL`
Non-retriable (destination invalid, compliance hold, malformed tx). Move to exception lane.

`FAILED_RETRY`
Transient (connectivity, fee cap, insufficient hot balance). Backoff retry; no re-sign.

`FAILED_UNKNOWN`
Outcome uncertain. Freeze; run uncertainty resolution.

Retry discipline (required).

- Query validated ledger for transaction hash and/or sequence consumption.
- If not found, resubmit the identical signed blob.
- Never recompute amount or destination.
- Never re-sign without an explicit break-glass incident record and two-person approval.

Reconciliation delta (required). Define:

$$\Delta = \texttt{manifest\_total\_drops} - \sum \texttt{confirmed\_drops} - \sum \texttt{held\_drops} - \sum \texttt{failed\_final\_drops}.$$

Normative: Δ **MUST** be 0 or booked under a pinned float policy with reason codes. Otherwise, the system **MUST** enter safe-mode.

14.5 Key Compromise

Containment SLA 60 s (required).

1. Kill-switch: halt payouts and replenishments.
2. Revoke signing credentials; disable hot keys if policy allows.
3. Freeze warm → hot flows; require manual quorum for any move.

4. Snapshot last 100 signing decisions and active sequence leases.

Rotation & recovery (required).

- Rotate keys via HSM; update XRPL signer lists; log old→new mapping.
- Cold treasury remains offline; no automated path from hot compromise.
- Resume in restricted mode with `RECOVERY_DAILY_CAP` until `SECURITY_CLEARANCE`.

14.6 Data Pipeline Failures

Append-only recovery (required). VPE logs and sealed manifests **MUST** persist in WORM (or equivalent) storage.

Recovery procedure (normative).

1. Rebuild eligible set from immutable VPE log; recompute `vpe_root`.
2. Recompute manifest bit-identically; compare to stored `manifest_hash`.
3. Mismatch → safe-mode; no payouts.
4. Match → continue and reconcile to validated ledger.

14.7 Safe-Mode Summary

- Halt payouts and replenishment; preserve playback where safe.
- Preserve deterministic replay; do not mutate manifests.
- Resolve uncertainty via validated XRPL state + sequence leases.
- Resume only after two-person approval and an incident evidence package.

15 Front-End: App & Platform Experience

Plain-language overview (non-technical). The front-end is the only surface the listener and artist ever touch, yet it is explicitly **untrusted** in the threat model. It delivers purchase, access-state display, and playback UX while **never** holding private keys, **never** storing clear-audio tracks, and **never** presenting any listener-facing artefact that could be interpreted as identity, status, collectability, or "proof-of-fandom".XRPlay sells **time-limited playback access** and nothing else. After expiry, the product retains **no listener-facing objects**, shelves, badges, trophies, history tiles, or proofs. Any retained records are **internal-only**, pseudonymized, and used solely for audit, fraud, dispute, sanctions, and legal obligations.

15.1 Design Principles (Normative)

- **Untrusted by default:** all financial, rights, and compliance truth is server-authoritative and signed.
- **Access semantics only:** UI shows only `access_active`, `expires_on`, and `days_remaining`.
- **No identity / no proof surfaces:** no badges, no shelves, no collectibles, no "proof", no fan leaderboards.
- **No balances:** the UI **MUST** not display "credits", "earned XRP", "wallet yield", "rewards", or anything similar.
- **No download path:** encrypted short segments with a RAM ring buffer; no persistent clear-audio cache.

15.2 Supported Surfaces

Table 1: Front-end capability matrix (access-only)

Surface	Capabilities	Security Limitations
iOS App (Swift)	Biometric auth for wallet actions; watermark; access-state UI (`active`/`expired`)	Screen capture cannot be eliminated entirely; forensic watermark is mandatory
Android App (Kotlin)	Device integrity checks; rooted-device mitigation; recording friction where supported	Loopback capture remains possible on some devices; watermark is the primary deterrent
Web (React + TypeScript)	Fast access purchase + playback; embeddable player; accessibility-first	Desktop capture is unavoidable; watermark is the primary deterrent
PWA	Installable shell; push notifications for access expiry reminders	Lower-integrity environment; treat as lowest-trust surface

15.3 Access Purchase Flow (UI)

1. Listener taps "Activate access" → shows **fixed-period access terms** and expiry date.
2. Mandatory disclosure (hash-pinned):

   > "You are purchasing time-limited streaming access only. This product is not transferable and creates no ownership or financial entitlement."

3. Platform selects permitted path by wallet mode:
   - **Mode A (Self-custody):** OS wallet signs the required XRPL action (channel open/update) under user keys.
   - **Mode B (Managed):** Apple/Google Pay → backend acquires **XRP** → escrow/settlement object created per policy.
4. Success → access activates immediately.

5. Failure → no partial access state (no "pending access" without completed funds movement).

15.4 Access State Model (Normative)

Listener UI states are restricted to:

- `ACTIVE` — access valid until `expires_on`.
- `EXPIRED` — playback disabled; **no** residual UI artefacts.
- `HELD` — access pending only where legally required (e.g., payment dispute); no playback.

Expiry semantics (required). At expiry:

- Playback keys are revoked; playback ends.
- The app removes all access-related UI artefacts beyond `EXPIRED`.
- The app must not display purchase history as a badge, trophy, shelf, or proof construct.

15.5 Artist Dashboard

- Upload → scans (loudness + fingerprint) → rights submission and pricing rules.
- Access product configuration: price and time period only; **no** capped "collectible" framing.
- Reporting shows **aggregate XRP** settled only; no purchaser lists, no wallet lists, no cohort drilldowns.

15.6 Security UX Guards

- Watermark always-on (client + server correlation).
- Concurrency: max 1 stream per wallet unless policy permits otherwise (server enforced).
- Device/session caps: max 3 enrolled devices; suspicious churn triggers holds.

15.7 Telemetry (Privacy-Minimised)

- Privacy-first analytics (self-hosted where feasible); aggregate bucketisation; no PII.
- Metrics: stream start latency, buffer ratio, purchase conversion, crash traces.

15.8 Deployment & Kill Controls

- Web: SRI pinned; CSP locked; strict origin allow-list.
- Mobile: staged rollout with rollback.
- Feature flags can disable purchases within 60 s (no app-store review required).

15.9 Front-End Incident Surface

- Fake app clone: server enforces package/signature allow-list; unknown signature → playback denied.
- Deep-link hijack: universal links with `apple-app-site-association` / `assetlinks.json` pinning.
- Payment phishing: payment UI confined to platform-owned domains; strict allow-lists; certificate pinning where feasible.

16 Architecture (System Design)

System posture. XRPlay is engineered as an **access-only** consumer DSP whose only cash-out is **native XRP on XRPL** to verified rights-holders. Listener products are strictly **consumptive**: time-limited playback access (XRPass) with **no** ownership, transfer, resale, rewards, or financial participation (Section 10).

Trust boundaries (required). Clients are **untrusted**. All economic truth, rights truth, and payout truth are **server-authoritative** and **cryptographically sealed** in deterministic manifests. The ledger is the source of final value transfer, but never the source of metering truth.

16.1 High-Level Components

- **A. Clients (iOS/Android/Web)** — playback, XRPass purchase flow, access-state display.
- **B. Edge & Identity Gateway** — auth, device/session binding, WAF, rate limiting, bot friction, abuse scoring.
- **C. Streaming Core (Metering Plane)** — server-authoritative segment orchestration, VPE minting, eligibility gating.
- **D. Rights & Catalog Plane** — upload, fingerprinting, rights assertions, split tables, versioned Rights Bundles.
- **E. Deterministic Manifest Builder** — canonical aggregation, pricing, split allocation, manifest sealing + HSM signing.
- **F. Settlement Executor (XRPL)** — manifest-bound signing, submission, confirmation mapping, sequence leases, holds.
- **G. Internal Audit Store (Internal-only)** — WORM logs + artifacts for replay, audit, fraud, dispute, legal.
- **H. Safe-Mode & Kill Controls** — delta enforcement, cap enforcement, circuit breakers, global halt within 60 s.

16.2 Architecture Diagram (Access-Only, No Identity Products)

```
+--------------------+       +--------------------------+
| A. CLIENTS          |       | B. EDGE + IDENTITY GW     |
| iOS / Android / Web | -----> | auth, device bind, WAF    |
| playback + XRPass   |       | rate-limit, bot friction  |
+---------+----------+       +-------------+-------------+
          |                                  |
          | access state                      |
          v                                  v
+--------------------+       +--------------------------+
| C. STREAMING CORE   | -----> | E. MANIFEST BUILDER       |
| VPE mint + gating   |       | deterministic, versioned  |
| signed proof hashes |       | manifest_total_drops      |
+---------+----------+       +-------------+-------------+
          |                                  |
          | rights lookup                     | sealed artifacts
          v                                  v
+--------------------+       +--------------------------+
| D. RIGHTS + CATALOG | -----> | F. XRPL SETTLEMENT        |
| splits, licensing   |       | manifest-bound signing    |
| version-pinned      |       | confirm finality / holds  |
+---------+----------+       +-------------+-------------+
          |                                  |
          | internal-only retention           | ledger receipts
          v                                  v
+--------------------+       +--------------------------+
| G. INTERNAL AUDIT   | <----- | H. SAFE-MODE + CONTROLS   |
| STORE (INTERNAL)    |       | caps, kill-switch, delta=0|
| fraud/audit/legal  |       | halt outbound <= 60 s     |
+--------------------+       +--------------------------+
```

Figure 1: XRPlay architecture: access-only consumer product; deterministic manifests; XRPL-native **XRP** payouts; internal-only audit retention (no listener-facing identity artifacts).

16.3 Data Planes and Artifacts (Normative)

1) Usage artifacts (attested). Clients emit telemetry; the Streaming Core produces **VPEs** that are: (i) session-bound, (ii) device-bound, (iii) replay-resistant, (iv) policy-versioned, and (v) canonically serializable. Clients **MUST** never assert credited seconds.

2) Rights state (version-pinned). Rights Bundles are immutable objects that bind: content hashes, effective window, territory scope, split tables, payee routing, and policy pins (pricing/rounding/fraud/compliance). Payouts are computed only from approved, pinned versions.

3) Settlement truth artifact. For each window the Manifest Builder creates a sealed manifest:

- inputs: eligible VPE set + pinned Rights Bundles + pinned policy hashes
- outputs: deterministic payout lines in drops, `manifest_root`, and HSM signature
- property: bit-identical replay given identical inputs and versions

16.4 Settlement Boundaries (Mode A / Mode B)

Mode A (Self-custody). The user controls keys. XRPlay does not custody user funds. Any on-ledger actions requiring user authority are wallet-signed by the user.

Mode B (Managed). XRPlay custodies **XRP** 1:1 under custody controls and offers withdrawals as raw **XRP** to verified destinations. Mode B **MUST** remain **no-internal-money**: no points/credits; only backed liabilities mapped to validated ledger indices.

16.5 Internal Audit Store (Not a Product)

No identity artifacts. The audit store is internal-only. It **MUST** not expose badges, shelves, trophies, purchaser lists, wallet lists, or "proof" objects to any user-facing surface.

Purpose limitation. Retention exists solely for reconciliation, audit replay, fraud investigation, disputes/chargebacks, sanctions/legal holds, and incident postmortems.

16.6 Fail-Closed Execution Path (Required)

Ambiguity triggers holds. If any prerequisite for correctness is missing or uncertain (ledger uncertainty, rights drift, policy mismatch, destination unverified), the system **MUST** hold and **MUST** emit explicit reason codes. It **MUST** not guess.

Safe-mode coupling. If reconciliation delta violates policy, or a manifest replay mismatch occurs, Safe-Mode **MUST** halt outbound value within 60 seconds and freeze any configured purchase paths.

16.7 Primary Attack Surfaces & Controls (Summary)

- **Bot farms / emulators:** device integrity where available, behavioral scoring, ASN clustering, rate limits, watermark correlation.
- **Replay crews / concurrency:** session binding, idempotency, server-authoritative metering, `MAX_CONCURRENT` enforcement.
- **Account takeover rings:** step-up auth, destination cooling-off, withdrawal holds, velocity caps.
- **XRPL congestion / delays:** fee caps, bounded retries, uncertainty timeouts, sequence leases, no re-sign discipline.

Architectural invariant (required). No user-facing construct may outlive access expiry in a way that implies identity, collectability, status, or economic participation. All such constructs are out-of-spec by design.

17 System Scope and Design Invariants

Purpose. This section defines the non-negotiable invariants that make XRPlay audit-grade and payments-safe. If any invariant is violated, the correct response is **halt + preserve evidence**, not "patch and proceed."

17.1 Scope Invariants

I1. **XRP-only cash-out.** All rights-holder payouts **MUST** be native **XRP** on **XRPL**. No alternative rails, no tokenization, no IOUs.

I2. **No internal money.** XRPlay **MUST** not create any unit that behaves like money (credits, points, balances, rewards). Non-final state **MUST** never be displayed or implied as economic value.

I3. **Access-only listener product.** XRPass is consumptive access only: non-transferable, expires, no ownership, no resale, no rewards.

I4. **Client is untrusted.** Clients **MUST** never be authoritative for credited seconds, eligibility, rights, splits, or payout values.

I5. **Rights uniqueness.** For each eligible second, exactly one Rights Bundle version **MUST** resolve. Ambiguity **MUST** withhold payouts.

I6. **Deterministic settlement.** Identical eligible inputs + identical pinned versions **MUST** yield byte-identical manifests and payout lines.

I7. **Fail-closed execution.** Uncertainty **MUST** halt outbound value and/or place explicit holds; no optimistic accounting.

I8. **No silent edits.** Historical VPEs, manifests, rights bundles, and payout lines are immutable. Corrections occur only via explicit adjustment batches.

I9. **Double-pay impossible by construction.** Idempotency keys + sequence leases + manifest-bound signing **MUST** prevent double-pay under retries, restarts, or partitions.

I10. **Reconciliation delta discipline.** Every window **MUST** reconcile to $\Delta = 0$ (or a pinned, reason-coded float tolerance). Otherwise safe-mode.

17.2 Deterministic Ordering and Canonicalization (Required)

Canonical order. All sets used for roots, aggregation, and payout line generation **MUST** be deterministically ordered by stable keys:

(`window_id`, `track_id`, `rights_version_hash`, `payee_id`, `destination`)

Lexicographic order is required unless a different pinned comparator is specified by schema.

Serialization rule. All sealed artifacts **MUST** use one canonical serialization per schema version. Mixed serialization is out-of-spec.

17.3 Arithmetic, Rounding, and Residuals (Required)

- All computations **MUST** be integer or fixed-scale; floating point is forbidden.
- Rounding mode **MUST** be policy-pinned (`ROUNDING_MODE`).
- Residual drops **MUST** be handled deterministically via a pinned residual policy (`RESIDUAL_POLICY`).

Residual policy (normative default). If splits produce residual drops, apply a deterministic sweep:

- allocate residuals by descending fractional remainder, tie-broken by `payee_id`,
- any remaining 1-drop dust **MUST** be swept to the platform fee pool (or other pinned sink).

17.4 Safe-Mode Definition (Normative)

Definition. Safe-mode is a globally enforced state where outbound value movement is halted and new signing is denied, except for explicitly permitted containment operations (e.g., key rotation, treasury safety moves) under quorum.

Entry triggers (minimum).

- manifest replay mismatch,
- reconciliation delta outside tolerance,
- rights drift / overlap detected,
- cap/velocity breach at signer,
- ledger uncertainty beyond `UNCERTAINTY_TIMEOUT_MIN`.

Exit discipline. Exiting safe-mode **MUST** require: (i) trigger cleared, (ii) evidence package sealed, (iii) two-person approval recorded.

18 Appendices

Purpose. These appendices define the **version-pinned** constants, minimal schemas, and reason-code primitives needed to (i) reproduce settlement deterministically, (ii) audit every payout end-to-end, and (iii) fail closed with explicit evidence. Any parameter that can change *who gets paid* or *how much* **MUST** be hash-pinned and change-controlled.

Appendix A: Core Parameters (Recommended Defaults)

Parameter	Meaning	Default	Control
`WINDOW_HOURS`	Settlement cadence for manifest sealing and payout execution windows	24	Pinned
`LATE_BUFFER_MIN`	Grace period after `window_close_ts` before late VPEs are quarantined to next window	15	Pinned
`UNCERTAINTY_TIMEOUT_MIN`	Maximum time allowed for unresolved XRPL outcome before safe-mode	30	Pinned
`DROPS_PER_SECOND`	Pricing rate in drops per eligible verified second (integer)	*policy-set*	Pinned
`PLATFORM_FEE_BPS`	Platform fee applied to gross drops (basis points)	250	Pinned
`ROUNDING_MODE`	Deterministic rounding mode (no float; fixed-scale or rationals)	`BANKERS_FIXED`	Pinned
`RESIDUAL_POLICY`	Residual handling (deterministic sweep rules + sinks)	`SWEEP_FEE_POOL`	Pinned
`MAX_FEE_DROPS_PER_TX`	Maximum XRPL fee per transaction (caps fee spikes)	10 000	Pinned
`MAX_FEE_DROPS_PER_BATCH`	Maximum total XRPL fees per batch (caps fee spikes)	100 000	Pinned
`HOT_TX_MAX_DROPS`	Max amount per outbound payout tx from hot settlement wallet	*policy-set*	Pinned
`HOT_MAX_DROPS`	Max total exposure in hot settlement wallet (epoch-bound)	*policy-set*	Pinned
`DEST_COOL_OFF_HOURS`	Cooling-off window for new payee destinations before first payout	24	Pinned
`RIGHTS_COOL_OFF_HOURS`	Cooling-off window for new Rights Bundle versions before activation	24	Pinned
`HOLD_HOURS`	Max quarantine hold duration for VPEs before auto-reject (no pay)	48	Pinned
`MAX_CONCURRENT`	Max concurrent streams per wallet (server enforced)	1	Pinned
`MAX_XRPass_DAILY`	Daily XRPass sales cap (kill-switch enforced)	*policy-set*	Pinned
`MAX_XRPass_PRICE_DROPS`	Per-XRPass price cap in drops	*policy-set*	Pinned
`TRIAL_DAILY_CAP_DROPS`	Max daily drops funded by trial pool (fail-closed)	*policy-set*	Pinned
`AD_POOL_DAILY_CAP_DROPS`	Max daily drops funded by ad pool (fail-closed)	*policy-set*	Pinned
`URL_TTL_SECONDS`	Signed media URL TTL (hotlink resistance)	120	Pinned
`RESTORE_COOLDOWN_MIN`	Minimum time between cold restore attempts per track	60	Pinned
`MAX_CDN_COST_BPS`	Max daily CDN spend vs epoch fee pool before safe-mode	*policy-set*	Pinned

Parameter pinning rule (normative). All parameters above **MUST** be covered by one of the policy hashes (`economic_policy_hash`, `abuse_policy_hash`, `storage_policy_hash`, `ops_policy_hash`) and the active hashes **MUST** be included in every sealed settlement manifest.

Appendix B: Minimal Canonical Schemas (Informative)

B.1 Verified Playback Event (VPE) — Minimal Fields

```
vpe_v1 = {
  vpe_id: bytes32,
  wallet_id: bytes32,            // pseudonymous internal ID
  device_id: bytes32,
  track_id: string,
  segment_range: {start_ms:int, end_ms:int},
  credited_seconds: int,         // integer; no floats
  session_id: bytes32,
  server_ts: rfc3339,
  ip_geo_bucket: string,         // coarse
  asn_id: int,
  proof_bundle_hash: bytes32,
  risk_score: int,               // fixed-scale integer
  eligibility_status: enum{ELIGIBLE, QUARANTINED, BLOCKED},
  supersedes_id: bytes32|null
}
```

B.2 Rights Bundle — Minimal Fields

```
rights_bundle_v1 = {
  track_id: string,
  rights_version_hash: bytes32,
  effective_from: rfc3339,
  effective_to: rfc3339,
  territory_map_hash: bytes32,
  split_table_hash: bytes32,
  payee_routing_set_hash: bytes32,
  policy_pins: {
    pricing_hash: bytes32,
    rounding_hash: bytes32,
    fraud_hash: bytes32,
    compliance_hash: bytes32
  },
  content_hashes: {
    master_sha256: bytes32,
    aac_sha256: bytes32,
    fingerprint_id: string
  },
  provenance: {
    rights_ops_sig: bytes,
    finance_sig: bytes,
    hsm_attestation: bytes
  }
}
```

B.3 Settlement Manifest — Minimal Fields

```
manifest_v1 = {
  schema_version: "manifest_v1",
  window_id: bytes32,
  window_close_ts: rfc3339,
  vpe_root: bytes32,
  eligible_vpe_count: int,
  policy_hashes: {
    economic_policy_hash: bytes32,
    abuse_policy_hash: bytes32,
    storage_policy_hash: bytes32,
    ops_policy_hash: bytes32
  },
  rights_mapping_root: bytes32,      // per-track rights_version_hash mapping commitment
  payout_lines: [
    { payout_id: bytes32, payee_id: bytes32, destination: string,
      amount_drops: int, status: enum{READY, HELD, FAILED_FINAL} }
  ],
  manifest_total_drops: int,
  residual_policy: string,
  manifest_root: bytes32,
  hsm_sig: bytes
}
```

Schema rule (normative). All schema serializations **MUST** be canonical and stable. Any schema change requires a new version identifier and new pins.

Appendix C: Reason Codes (Normative)

Purpose. Reason codes are the **only** allowed explanation mechanism for quarantines, holds, failures, and safe-mode. They are deterministic, versioned, and required for audit reconstruction.

C.1 VPE Quarantine / Block Codes

- `Q_MISSING_PROOF` — required proof signal missing/stale
- `Q_CONCURRENCY` — concurrent streams exceeded policy
- `Q_TIMING_INCOHERENT` — progression/seek patterns incoherent
- `Q_DEVICE_RISK` — device integrity failure / emulator traits
- `Q_TERRITORY_UNLICENSED` — territory not licensed at time of play
- `B_BOT_FARM` — clustered automation detected (device/IP/ASN graph)
- `B_REPLAY_CREW` — token/session replay detected

C.2 Payout Hold Codes

- `H_SANCTIONS_HIT` — sanctions screening hit or partial match
- `H_ROUTING_REQUIRED` — jurisdiction requires intermediary routing not available

- `H_DEST_UNVERIFIED` — destination not verified / not allow-listed / cooling-off active
- `H_DISPUTE` — dispute or chargeback state requires hold
- `H_LEGAL_FREEZE` — regulator/court freeze request

C.3 Final Failure Codes

- `F_DEST_INVALID` — destination malformed or non-existent per verification
- `F_COMPLIANCE_DENY` — compliance denial (non-releasable)
- `F_TX_MALFORMED` — transaction construction invalid under pinned schema

C.4 Safe-Mode Trigger Codes

- `S_DELTA_MISMATCH` — reconciliation delta outside pinned tolerance
- `S_REPLAY_MISMATCH` — manifest replay hash mismatch
- `S_RIGHTS_DRIFT` — rights overlap / unresolved rights bundle for eligible seconds
- `S_SIGNER_ANOMALY` — cap/velocity breach or anomalous signing pattern
- `S_LEDGER_UNCERTAINTY` — XRPL uncertainty beyond `UNCERTAINTY_TIMEOUT_MIN`
- `S_KEY_COMPROMISE` — suspected key compromise or policy bypass attempt

Reason-code rule (normative). Any `QUARANTINED`, `HELD`, `FAILED_FINAL`, or safe-mode state **MUST** be accompanied by exactly one primary reason code and **MAY** include secondary codes. Missing reason codes are out-of-spec.

Appendix D: Idempotency Keys and Sequence Leases (Normative)

Payout idempotency (required). Each payout line item **MUST** have an idempotency key derived from:

$$\texttt{idemp} = H(\texttt{manifest\_hash} \parallel \texttt{payout\_id} \parallel \texttt{destination} \parallel \texttt{amount\_drops})$$

where H is a policy-pinned hash function (default SHA-256) and all fields are canonical-serialized. The executor **MUST** reject any request where the computed `idemp` does not match the stored value.

Sequence lease binding (required). For each submitting/signing account, a durable sequence lease record **MUST** bind: `sequence` $\leftrightarrow$ `payout_id` $\leftrightarrow$ signed tx hash $\leftrightarrow$ final status. Reusing a sequence for a different payout is forbidden.

Uncertainty handling (required). If a signed transaction's outcome is uncertain:

- the system **MUST** query validated ledger state and recent tx history,

- it **MUST** resubmit only the identical signed blob (if needed),
- it **MUST** not re-sign or alter destination/amount,
- if uncertainty persists beyond `UNCERTAINTY_TIMEOUT_MIN`, it **MUST** enter safe-mode.