Hostra Settlement Engine

Architecture & Guarantees — Technical Whitepaper

This document describes the invariants, protocols and failure semantics of the Hostra Settlement Engine — the core that moves regulated money at Hostra.

1. Design goals

The engine is designed around five properties: exactly-once settlement, deterministic recovery, an append-only ledger, lease-based ownership and first-class retry orchestration. Every property is enforced by construction, not by policy.

2. Exactly-once settlement

Every settlement intent carries a client-side idempotency key of the form SHA-256(tenant, intent_id, version). Duplicate submits within a 30-day, per-tenant, strongly consistent window collapse to a single ledger effect. An identical payload returns an identical response byte-for-byte; a differing payload with the same key returns 409 Conflict.

3. Deterministic recovery

State is reconstructed as fold(reduce, snapshot, journal[snapshot.offset:]) — a pure function over an append-only journal. Snapshots are emitted every 10^6 events. p99 recovery on a hot node is under 90 seconds. Two nodes at the same journal offset compute bit-for-bit identical state hashes.

4. Append-only ledger

Postings are content-addressed with blake3 and chained by prev_hash. A Merkle root is emitted per epoch and can be verified end-to-end by an external auditor without trust in the operator. Corrections are compensating entries; the engine never edits history in place.

5. Lease-based ownership

Each settlement partition is owned by exactly one worker via a fenced lease. TTL 8s, heartbeat 2s, fencing token monotonic. Writes with a stale token are rejected even if the previous owner is still alive. Ownership handoff across an AZ failure is p99 < 4s.

6. Retry orchestration

Retries are first-class objects. Exponential backoff with decorrelated jitter, per-tenant retry budget to prevent thundering-herd on rail outages, dead-letter routing and deterministic DLQ replay. Backoff policy is declared per counterparty and inspected in the control plane.

7. Commit protocol

POST /v1/settlement/intents with Idempotency-Key. 201 on first submit, 200 with replayed=true on any subsequent submit with the same key, 409 on a conflicting payload. Each intent produces at most one ledger effect: a debit and matching credit posting linked to the previous hash.

8. Failure semantics

Worker crash mid-commit: new owner replays journal from last committed offset, RTO < 6s, RPO 0. Primary DB failover: reconnect, verify last LSN, resume, RTO < 20s, RPO 0. AZ outage: quorum handoff, RTO < 90s, RPO 0. Full region loss: standby region promoted from streaming journal, RTO < 15m, RPO < 5s. Poisoned message: reducer error, intent quarantined to DLQ.

9. Multi-region topology

The engine runs active-active per tenant within a region and active-standby across regions. The journal streams to the standby region continuously; state at the standby is a deterministic function of the streamed offset. Failover is a control-plane action that promotes the standby to primary once its offset matches the last durable offset in the primary region.

10. Audit trail

Every posting is signed and linked. The per-epoch Merkle root is published to an internal transparency log and made available to auditors. Because state is a pure fold over the journal, any external party with access to the journal and snapshot can recompute the exact state Hostra observes.

11. Service level objectives

Durability 11 nines. Commit p99 38 ms. Recovery p99 < 90 s. RPO 0 within a region. All SLOs are measured continuously and surfaced on the control plane.