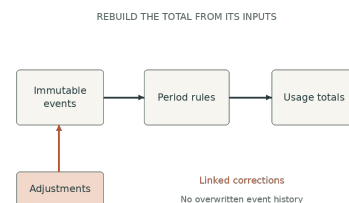


BACKEND AND DATA SYSTEMS

Usage reconciliation ledger

Reconcile usage records before they become invoice quantities.



PROPOSED ARCHITECTURE

Ledger design brief; event policy to define

THE PROBLEM

A service receives usage events late, twice, or after a correction. Adding the latest records into a total makes disputes hard to investigate. This ledger preserves the original events and produces reproducible period totals with a trace back to each input.

SCOPE

- Store events in PostgreSQL with a unique source and event ID; use exact quantities and declared units.
- Record corrections as linked adjustments; close periods explicitly and version calculation rules.
- Provide a reconciliation view and export; keep prices, tax, and payment processing out of scope.

PLANNED DELIVERABLES

- Ingestion API, reconciliation UI, and immutable event model
- Replay, period-boundary, and account-isolation tests
- Sample event generator, exports, and an operator runbook

CONSTRAINTS

Identity

Duplicate IDs must carry identical payloads or be rejected

Periods

Event time, arrival time, and late-event policy are separate

Quantities

Integer or fixed-decimal arithmetic; no binary-float totals

Access

Account-scoped queries and audited adjustment permissions

VERIFICATION PLAN

Duplicate events

Replay a batch in different orders; the total remains unchanged.

Corrections

Trace a reversal and replacement to the original event.

Closed period

A late event creates a visible pending adjustment, not a silent rewrite.

NEXT STEP

Define the usage units, period boundary, and late-event adjustment policy.

The service exports usage quantities. Pricing, taxes, invoices, and payment collection belong to downstream systems.