VERIT GLOBAL LABS

www.veritglobal.com

Tech. Blueprint

Contents:

1. Interface patterns & recommendations (with ranking, decision tree, anti-patterns)
2. Architecture: **12 core** modules + **4 context** boxes, ASCII diagram
3. Data model & repositories (per module + cross-cutting contracts)
4. How to run (K8s, scale units, DR/multi-region, SLOs/alerts, security)
5. Implementation plan & checklists (MVP → GA)
6. Failure modes & runbooks
7. Glossary & appendices (reason codes, payload schemas, API sketch)

---

1) Interface patterns & recommendations

1.1 The core invariant (applies to every interface)

- Compute payouts **deterministically**, seal a **transcript** and **output digest**, then **release funds only** if **replay digest == transcript digest** *and* an **acceptance bundle** (ACK/CT/optional SPV) meets **freshness** and **quorum**.

1.2 Options (8)

#	Interface pattern	Best for	Pros	Cons
1	**One-call Authorization Gate (REST/gRPC)** + tiny verifiers	Quick overlay, minimal change	Simple; reason-coded holds; cryptographic audit	Add 2–3 verifier webhooks; pass IDs through rails
2	**Batch flat-file (Tier-0) + attestations**	Weekly/biweekly; low lift	Fastest start; no streaming	Windowed latency; later upgrade to streaming

#	Interface pattern	Best for	Pros	Cons
3	**ERP-native adaptor (e.g., NetSuite)**	Finance-led, strict audit	Tight audit trail; clean GL mapping	Some scripting/config; chunk at volume
4	**SDK-first (Stripe-style)**	Developer-led builds	Great DX; feature flags	Manage keys/versions; still need attestations
5	**Event streaming (Kafka/Pub/Sub)**	High volume, near real-time	Smooth scale; predictable close	More platform work than files
6	**PSP plugin + SPV receipts**	Keep existing rails	Airtight reconciliation	Wire receipt pickup within freshness
7	**On-/cross-chain SPV adaptor**	On-ledger touchpoints	Deterministic acceptance across domains	Finality/headers to monitor
8	**Ops copilot (service/agent)**	Large ops teams	Fewer escalations; auto-nudges	Support layer, not primary integration

Default ranking (typical buyer): 1 → 6 → 2 → 5 → 3 → 4 → 8 → 7

1.3 Decision tree (text)

```
Start → Need value in weeks (Y) → Choose 1 + 2; add 6 later
                |
                N → Tight SLA or heavy scale (Y) → 1 + 5 (+6)
                |
                N → Finance-led & ERP-centric (Y) → 3 + 1 (+6)
                |
                N → Product/dev-led (Y) → 4 + 1 (+6)
                |
                Crypto/on-ledger (Y) → add 7 to acceptance matrix
```

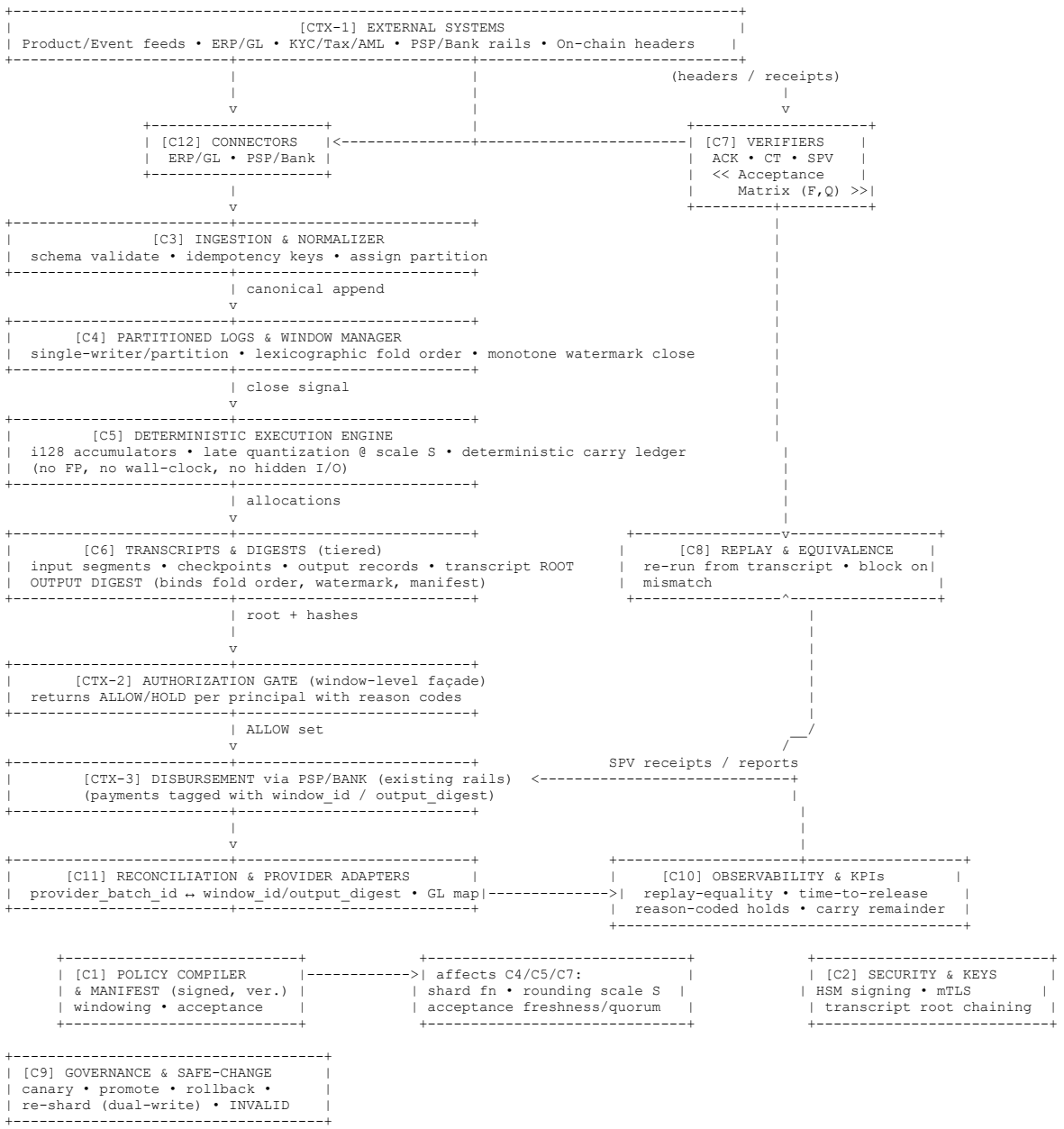
1.4 Anti-patterns to avoid

- Becoming the **payer of record** (replace rails) → prefer overlay + SPV.
- **Per-payment** synchronous gating → gate **per window**.
- Re-implementing compute in ERP → keep ERP as **system of record**, not allocation engine.
- Skipping evidence (no ACK/CT/SPV) → defeats the purpose.
- Dropping determinism guardrails (unordered data, FP math, wall-clock) → breaks replay.

---

2) Architecture (12 core + 4 context)

2.1 Big picture ASCII (copy-safe)



2.2 Module deep-dives (Interfaces • Process • Repositories • Flows)

For each module, “Gets from → Sends to” shows dependencies.

[C1] Policy Compiler & Manifest

Interfaces: propose/promote/rollback; get active manifest per window.

Process: compile policy → IR; fix window cadence/timezone; shard function/version; rounding scale S; acceptance F/Q; checkpoint cadence; sign/version.
Repositories: Git/config; Manifest table (OLTP); artifact bucket.
Flows: **Gets** Governance [C9], Security [C2] → **Sends** params to [C4][C5][C7].

[C2] Security & Keys
Interfaces: KMS/HSM, cert issuance, signing API.
Process: mTLS everywhere; sign transcript roots/manifests; rotate keys; audit.
Repositories: KMS/HSM; key metadata (OLTP).
Flows: **Gets** Governance [C9] → **Sends** signatures to [C6] and [C1].

[C3] Ingestion & Normalizer
Interfaces: Event API; file intake; streams.
Process: schema validate; derive/validate `window_id`; idempotency; assign `partition_id`; canonical timestamps.
Repositories: raw/normalized lake; topics; idempotency index.
Flows: **Gets** External [CTX-1], mappings [C12] → **Sends** canonical events to [C4].

[C4] Partitioned Logs & Window Manager
Interfaces: append per (tenant, window, partition); watermark queries.
Process: single-writer; lexicographic fold; monotone watermark to close.
Repositories: compacted logs (open); window state (OLTP); archived logs.
Flows: **Gets** [C3], rules [C1] → **Sends** close + ordered stream to [C5].

[C5] Deterministic Execution Engine
Interfaces: window close trigger; replay versioning.
Process: i128 accumulation; late quantization; deterministic carry; no FP/wall-clock/hidden I/O.
Repositories: worker KV; checkpoints; execution index.
Flows: **Gets** [C4], [C1] → **Sends** allocation records to [C6].

[C6] Transcripts & Digests (tiered)
Interfaces: get transcript/output digest/segment list.
Process: content-addressed segments (inputs/checkpoints/outputs); compute `output_digest` + trailer (fold order, watermark, manifest hash); sign/publish `transcript_root`.
Repositories: object store tree; transcript index (OLTP).
Flows: **Gets** from [C5], signatures [C2] → **Sends** digest/root to [C7][C8][CTX-2][C11].

[C7] Verifiers (+ Acceptance Matrix)
Interfaces: proof intake: **ACK** (finance), **CT** (tax/KYC/rights), **SPV** (provider receipts or

on-chain headers); acceptance query.
Process: validate; enforce freshness F & quorum Q; reason-coded outcomes; bind acceptance to transcript/output_digest.
Repositories: acceptance (OLTP); proofs (object store); cache for not-expired.
Flows: **Gets** digest/root [C6], external proofs [CTX-1/3] → **Sends** acceptance to [CTX-2]; snapshot to [C10].

[C8] Replay & Equivalence
Interfaces: start replay; get equality/mismatch.
Process: reproduce from transcripts only; block on mismatch; store diffs.
Repositories: job log (OLTP); replay logs (object store).
Flows: **Gets** [C6] → **Sends** result to [CTX-2], [C9], [C10].

[C9] Governance & Safe-Change
Interfaces: propose/promote/rollback; canary cohorts; re-shard dual-write.
Process: N-window success before promotion; rollback on violations; INVALID + checkpoint recovery.
Repositories: governance ledger (append-only, signed); manifest snapshots.
Flows: **Gets** [C7][C8] → **Sends** activations to [C1]; rules to [C4][C6]; notifications to [C10].

[C10] Observability & KPIs
Interfaces: metrics/logs/traces; dashboards/alerts.
Process: time-to-release; replay-equality; reason-codes; carry remainder; watermark lag; proof freshness.
Repositories: metrics TSDB; logs; traces; BI warehouse.
Flows: **Gets** [C4–C8], [CTX-3] → **Sends** alerts; feeds [C9].

[C11] Reconciliation & Provider Adapters
Interfaces: provider report intake; normalize; post SPV; ERP export mappers.
Process: map `provider_batch_id` ↔ `{window_id, output_digest}`; reconcile totals; raise diffs as reasons; feed SPV back to Verifiers.
Repositories: raw drop-zone; normalized receipts (OLTP); recon dashboards.
Flows: **Gets** from [CTX-3] → **Sends** SPV to [C7]; GL maps to [C12]; metrics to [C10].

[C12] Connector Layer to ERP/GL/PSP
Interfaces: field mappings; webhook inbox; export jobs; optional ERP `eligible_to_pay` flag.
Process: stamp `window_id`, `output_digest`, `transcript_url`, `provider_batch_id` on bills/payments; filter ALLOW for pay-runs.
Repositories: connector state (OLTP); payload archives.

Flows: **Gets** decisions [CTX-2], recon [C11] → **Sends** to External [CTX-1]; optional echoes to [C3].

[CTX-1] External Systems
Interfaces: file drops, streams, read-only DB views.
Process: land/validate/promote; supply KYC/Tax proofs; return provider receipts.
Repositories: staging buckets/queues.
Flows: **Sends** events to [C3]; proofs to [C7]; receipts to [C11].

[CTX-2] Authorization Gate (window-level façade)
Interfaces: authorize `window_id` → ALLOW/HOLD with reasons per principal.
Process: require (replay digest == transcript digest) ∧ (acceptance F/Q met).
Repositories: authorization log (OLTP).
Flows: **Gets** digest [C6], acceptance [C7], replay [C8] → **Sends** ALLOW to [CTX-3]; eligible refs to [C12].

[CTX-3] Disbursement via PSP/Bank
Interfaces: originate batches; tag (`window_id`, `output_digest`); return `provider_batch_id` and reports.
Process: pay ALLOW; keep batch/report artifacts for SPV.
Repositories: payout batch registry (OLTP); raw provider reports.
Flows: **Gets** ALLOW [CTX-2] → **Sends** metadata to [C11]; receipts to [C7] via [C11].

[CTX-4] Acceptance Matrix (Ops View)
Interfaces: read-only per-window matrix (kinds, freshness, quorum, expiry, status).
Process: materialized view from [C7]; no business logic.
Repositories: materialized view + cache.
Flows: **Gets** from [C7]; **Sends** none.

---

3) Data model & repositories

3.1 Cross-cutting IDs

- `window_id`, `tenant_id`, `partition_id`, `policy_version`, `output_digest`, `provider_batch_id`, `transcript_root`.

3.2 Canonical contracts (sketch)

- **EventRecord (C3)**: `{tenant_id, window_id, event_id, ts_logical, bucket_id, principal_id, currency, amount_minor, payload_digest, provider_id}`
- **AllocationRecord + Trailer (C6)**: `allocations[]` + `{watermark, fold_order_desc, policy_manifest_hash}`
- **PayoutHeader (C6/C7)**: `{window_id, policy_version, output_digest, acceptance_requirements{F,Q,kinds}}`
- **AcceptanceRecord (C7)**: `{window_id, kind ∈ {ACK,CT,SPV}, subject_id, status, quorum, expires_at, reason_code, signature}`
- **AuthDecision (CTX-2)**: `{window_id, principal_id, output_digest, decision ∈ {ALLOW,HOLD}, reason_code}`

3.3 Repositories by module (authoritative store)

- **C1** Git + Manifest (OLTP); artifacts in object store
- **C2** KMS/HSM; key metadata (OLTP)
- **C3** Object store (raw/normalized), Topics, Idempotency (OLTP)
- **C4** Topics (open), Window state (OLTP), Archived logs
- **C5** Worker KV + Checkpoints (object store)
- **C6** Object store tree + Transcript index (OLTP)
- **C7** Acceptance (OLTP) + Proofs (object store)
- **C8** Replay jobs (OLTP) + logs (object store)
- **C9** Governance ledger (OLTP, append-only) + snapshots
- **C10** Metrics/Logs/Traces backends + Warehouse
- **C11** Raw receipts (object store) + Normalized receipts (OLTP)
- **C12** Connector state (OLTP) + archives
- **CTX-2** Auth log (OLTP); **CTX-3** Payout registry (OLTP); **CTX-1** staging buckets; **CTX-4** materialized view

3.4 Security & retention

- **Transcripts** immutable; 7–10 years; WORM/immutability.
- **Proofs** expire at `expires_at`, but **decisions**/audit retained long-term.
- Keep **PII out of transcripts**; store sensitive CT/KYC evidence separately with KMS keys.

---

4) How to run (platform & scale)

4.1 Deployment model

- **Default:** Multi-AZ SaaS single region → Active-passive DR → Active-active (tenant-pinned windows).
- **Alt:** Private cloud/on-prem: same shapes, self-hosted services.

4.2 Kubernetes layout

- **Namespaces per env** (dev/stage/prod). HPA on [C3][C4][C5][C7].
- API Gateway/Ingress + mTLS + NetworkPolicies.
- Secrets via KMS; short-lived tokens (SPIFFE/SPIRE optional).

4.3 State & queues

- **Kafka/PubSub** for [C3][C4]; 3→9 brokers, ISR across AZs.
- **Postgres HA** for [C1][C7][C9][C12][CTX-2]; shard later (Citus/Hyperscale).
- **Object store** for [C6][C11]; cross-region replication + Object Lock.
- **Redis** optional for caches/webhook dedupe.
- **Observability**: Prometheus + ELK/OpenSearch + Jaeger/Tempo; Warehouse for BI.

4.4 Scale units & autoscaling

- Partition count `K` by window (target 2–5M events/partition); one **engine worker per (partition, window)**.
- Autoscale triggers: **queue depth** (C3), **watermark lag/open windows** (C4), **active partitions & CPU** (C5), **proof backlog & expiry proximity** (C7).
- Backpressure: 429 beyond tenant quota; pause promotion if `time-to-release` SLO degrades.

4.5 Multi-region

- **Phase 1:** single region multi-AZ.
- **Phase 2:** active-passive DR (RPO ≤ 5m, RTO ≤ 30m).
- **Phase 3:** active-active via tenant pinning or independent windows; reconcile at [C11].

4.6 Boot sequence

1. Provision platform (Kafka, Postgres(HA), Object store, Redis, Observability).
2. Bring up control plane [C1][C2][C9]; publish **Manifest v1**.
3. Start data plane [C3→C6]; run canary tenant with synthetic load.

4. Wire [C7] with dummy ACK/CT; run [C8] replay; enforce **digest equality = 100%** for canary.
5. Enable [CTX-2] Gate; tag rails; confirm SPV loop [C11]⇄[C7].
6. Roll pilots under governance canary; promote after N clean windows.

4.7 SLOs & alerts (minimum)

- **Window close p95** (C4): ≤ 15 min; warn 20; page 30.
- **Transcript seal** (C6): ≤ 10 min after close.
- **Replay equality** (C8): 100% on canary; any mismatch pages.
- **Gate correctness** (CTX-2): no ALLOW unless (digest match ∧ acceptance met).
- **Proof freshness** (C7): alert T-15 min; quorum breach pages.
- **Recon completeness** (C11): provider receipt within freshness window.

4.8 Security defaults

- mTLS everywhere; sign manifests, transcript roots, gate responses.
- Dual-control on governance; audit every signing event.
- No PII in transcripts; separate encrypted bucket for CT/KYC evidence.
- Strict RBAC + NetworkPolicies; Kafka ACLs + Postgres RLS.

---

5) Implementation plan & checklists

5.1 0–30 days (MVP)

- Stand up platform; implement [C3][C4][C5][C6]; basic [C7] with ACK/CT only; [CTX-2] gate; [C11] minimal SPV intake; [C10] metrics.
- Deliver **Interface #1** + **#2**; tag rails with ids.

5.2 30–60 days (Pilot)

- Add **Interface #6** SPV loop; harden [C7] reason codes; enable [C8] replay; Observability SLO dashboards; governance canary.

5.3 60–90 days (GA)

- Add streaming (#5) if needed; ERP adaptor (#3) for finance-led teams; SDKs (#4) for dev-led teams; on-chain SPV (#7) where applicable; ops copilot (#8) as a support layer.

Readiness checklist (excerpt)

- Determinism tests (unordered/FP/wall-clock bans)
- Replay = compute on canary windows
- Transcript bucket: immutability + replication
- Gate idempotency on `{window_id, output_digest}`
- Proof freshness alerts; quorum dashboards
- Recon round-trip verified against provider

---

6) Failure modes & runbooks

- **Digest mismatch (C8)** → **HOLD** at gate; attach replay diff; likely causes: non-canonical input, shard drift, accidental FP; action: freeze promotion; re-ingest/repair; replay; resume.
- **Stale proofs (C7)** → **HOLD**; ops copilot nudges owners; action: obtain fresh ACK/CT/SPV; gate re-authorizes deterministically.
- **Watermark stall (C4)** → slow/late partition; action: inspect lag; apply grace policy; re-balance partitions.
- **Recon variance (C11)** → mismatch totals; action: re-normalize provider file; open reason code; block delta cohort.

---

7) Glossary (short)

- **ACK**: Finance approval (e.g., reserves OK).
- **CT**: Compliance/TAX/KYC/rights attestation.
- **SPV**: Simplified Proof of Validity (provider receipts or on-chain headers/inclusion proofs).
- **Transcript**: Content-addressed, tiered record set (inputs/checkpoints/outputs) sufficient to replay and verify equivalence.
- **Output digest**: Hash over canonical allocation records + trailer (watermark, fold order, manifest hash).
- **Watermark**: Monotone condition that closes a window across partitions.

Appendices

A) Reason-code catalog (starter)

- `DIGEST_MISMATCH`, `STALE_PROOF`, `INSUFFICIENT_QUORUM`, `MISSING_ACK`, `MISSING_CT`, `MISSING_SPV`, `PROVIDER_TOTALS_MISMATCH`, `WATERMARK_STALL`, `POLICY_VERSION_CONFLICT`.

B) Payload sketches (JSON)

Authorize (CTX-2)

```
{
  "window_id": "W-2025-09-12-1",
  "output_digest": "hex...",
  "decisions": [
    {"principal_id": "P123", "decision": "ALLOW"},
    {"principal_id": "P124", "decision": "HOLD", "reason_code":
"STALE_PROOF"}
  ]
}
```

Acceptance (C7)

```
{
  "window_id": "W-2025-09-12-1",
  "kind": "CT",
  "subject_id": "P123",
  "status": "VALID",
  "quorum": 2,
  "expires_at": "2025-09-13T00:00:00Z",
  "signature": "base64..."
}
```

C) API surface (sketch)

- `/ingest/events:POST` (C3)
- `/window/{id}/authorize:POST` (CTX-2)
- `/audit/transcript/{window}:GET` • `/audit/output-digest/{window}:GET` (C6)
- `/acceptance/{window}:GET, /acceptance/proof:POST` (C7)
- `/replay/{window}:POST` (C8)
- `/policy/propose|activate|rollback` (C1)
- `/governance/decision:POST` (C9)