

Axient: Canonical Protocol-Graph Composition for Leveraged Event Markets: Single State Authority, Atomic Composition, Durable Sagas, and Exactly-Once Recovery

Maksym Nechepurenko

Research Department of Devnull FZCO, Dubai, UAE

maksym@devnull.ae ORCID 0000-0002-9515-8841

30 September 2026 | Version r0.6.3

Abstract

A modular leveraged event-market protocol can contain individually correct contracts for risk approval, positions, debt, settlement evidence, credit pools, junior backstops, reserves, liquidation, and governance while still lacking one authoritative financial execution path. This paper develops a canonical protocol graph in which every financial domain has one storage authority and composition components coordinate transitions without recreating balances, debt, positions, evidence receipts, queues, reserves, pause state, or terminal scenario state. The architecture formalizes atomic position origination, a settlement-confirmed debt saga, trader residual release, an ordered reserve-backstop-Senior loss waterfall, loss-participating withdrawal queues, capability-conditioned execution, protocol-wide pause and timelocked recovery, deterministic deployment manifests, canonical state projection, and exactly-once reconstruction after a chain-confirmed checkpoint. We prove composition-level authority uniqueness, atomic closure, loss conservation and priority, receipt-idempotent settlement, legal no-go safety, capability-transition safety, and no-financial-reexecution recovery under explicit assumptions. The implementation is evaluated through twelve parent Financial Interaction Assertions (FIA-001–FIA-012), including mandatory FIA-011A/B capability subcases. The registered local cohort contains twelve uniquely correlated scenario records and 84 retained `ACTUAL_EVIDENCE` layer positions, two clean materializations with one byte-identical file index, a deterministic strict archive accepted by a separate verifier, and a selected ordered replay bound to that archive. The result connects the formal protocol graph to a reproducible evidence chain while preserving explicit boundaries between the observed engineering result and future external-operation hypotheses.

Keywords: event markets; protocol composition; smart contracts; on-chain credit; single source of truth; atomicity; durable sagas; exactly-once recovery; loss waterfall; canonical state.

1 Introduction

Leveraged event-market positions combine a time-bounded trading problem with a credit problem. A trader supplies collateral, a credit provider finances additional outcome exposure, an execution venue determines what can be bought or sold, and an oracle or verification gateway later fixes the event payout. Earlier Axient work separates leverage maturity from claim maturity and develops debt-free finality: leverage is extinguished before the claim becomes non-tradable, leaving a fully funded residual claim for later resolution [10]. The companion credit paper introduces Senior Credit LPs, Liquidation Backstop Providers (LBPs), reserves, queues, and a deterministic loss waterfall [9]. The empirical programme then distinguishes

formal certificate bounds from statistical validation and defines a controlled internal-alpha methodology [11].

Those results do not, by themselves, answer an implementation question: when the system is decomposed into many contracts and services, which object is the financial source of truth? A test-only bridge can reproduce a desired terminal state while maintaining its own queue, pause flag, evidence registry, or scenario state. A reference waterfall can be correct algebraically yet become a second capital ledger if it is connected alongside real vault balances. An application service can expose plausible projections without proving that the intended canonical selectors were called. The result may be internally consistent but architecturally ambiguous.

This paper studies the composition problem. Its central requirement is:

Every financial domain has exactly one canonical storage authority. Composition code may coordinate canonical modules, but it may not own a second copy of balances, debt, positions, evidence receipts, queues, reserves, pause state, or terminal financial state.

The requirement is stronger than modularity. Modularity says that responsibilities are separated; canonical composition says that their states and transitions jointly form one financial machine. It also differs from a generic orchestrator. The orchestrator may schedule, correlate, retry, and project, but it cannot settle debt by rewriting its own database. The authority boundary is on-chain and manifest-bound.

Contributions.

- (C1) A composition-level storage-authority rule that applies to every nonempty subset of financial domains, not only to each component in isolation.
- (C2) Storage-free composition controllers for position origination, ordered loss allocation, and protocol-wide emergency pause.
- (C3) Capability-conditioned execution authority, including stale-capability rejection before admission and fail-closed book-epoch transition handling while debt remains positive.
- (C4) An authority context binding each financial mutation to a legal-entity role, custody class, and permitted-use class.
- (C5) A distinction between atomic EVM transitions and durable, idempotent service sagas.
- (C6) A canonical settlement lifecycle in which typed evidence is the idempotency boundary and debt reduction precedes residual or collateral release.
- (C7) A single executable loss waterfall over reserve, LBP, and Senior modules without a duplicate capital ledger.
- (C8) A deterministic deployment manifest that binds contract addresses, code identities, ABIs, storage layouts, roles, selectors, pause targets, adapter identity, and application schemas.
- (C9) A canonical state projection derived from ABI views and ordered canonical events rather than fixture values.
- (C10) An effect-level recovery criterion based on a chain-confirmed checkpoint and replay without a duplicate committed financial effect.
- (C11) A twelve-scenario FIA evaluation surface and a strict separation between the completed cohort-bound evidence result and successor external-operation hypotheses.

The architecture overview, ordered waterfall, and emergency-control topology are shown in Figures 1 to 3.

2 System model and evidence scope

2.1 Financial domains

Let $\mathcal{D}$ be the finite set of financial domains and $\mathcal{M}$ the set of deployed modules. The relevant domains are

$$\mathcal{D} = \{\text{risk, position, debt, evidence, senior, lbp, reserve, liquidation, governance}\}.$$

The global financial state is the product

$$X = \prod_{d \in \mathcal{D}} X_d,$$

where X_d is the state space of domain d .

Definition 2.1 (Canonical authority map). *A canonical authority map is a function*

$$A : \mathcal{D} \rightarrow \mathcal{M}$$

that assigns exactly one storage-owning module to each financial domain.

Definition 2.2 (Cross-cutting pause state). *Let $\mathcal{T}$ be the nonempty manifest-frozen set of pausable modules. Its authoritative pause state is the indexed vector $P_{\mathcal{T}} = (p_m)_{m \in \mathcal{T}}$, where each component is owned by its target module. The statement that all targets are paused is distinct from the statement that at least one is paused. An action is governed by its registered target and selector policy, not by an unqualified aggregate flag. Missing or stale required state does not establish permission or global safety. Pause is a cross-cutting control over financial domains, not another financial domain with one storage owner. The coordinator may fan out commands but owns no component of $P_{\mathcal{T}}$.*

Definition 2.3 (Authority context). *An authority context is a tuple*

$$\lambda = (e, c, p),$$

where e is the legally responsible entity class, c the custody class governing the asset or claim, and p the permitted-use class. A module address is technically reachable but not financially authorized unless the manifest binds the relevant domain to an admissible authority context λ .

Definition 2.4 (Venue capability state). *At time t , the venue capability state is*

$$\kappa_t = (a_t, b_t, \rho_t),$$

where a_t is the versioned capability vector, b_t the book epoch, and ρ_t the pause/reactivation mode. Capability continuity for a command means that every capability used by its certificate remains valid over the registered execution interval, or that a registered risk-reducing fallback is triggered when continuity fails.

The registered local composition uses the financial-domain mapping in Table 1. Pause is evaluated separately through the manifest-frozen indexed vector in the cross-cutting pause predicate.

The entity, custody, and permitted-use classes associated with the map are summarized in Table 2.

A displayed balance or economically available unit is not automatically eligible capital. If the entity, custody, or permitted-use gate fails, the command receives a typed `LEGAL_NO_GO` result and produces zero financial mutation.

Table 1: Canonical storage-authority map.

Domain	Sole canonical authority
Risk approvals and 2× limits	<code>RiskManager</code>
Position lifecycle and locked collateral	<code>PositionManagerCore</code>
Principal, accrued interest, matched/settled capacity	<code>DebtManagerCore</code>
Evidence, replay, conflict, recovery	<code>SettlementEvidenceRegistry</code>
Senior assets, shares, losses, withdrawal queue	<code>CreditPool</code>
LBP assets, shares, encumbrance, impairment, queue	<code>LBPVault</code>
Segregated reserve custody	<code>ReserveVault</code>
Liquidation request and execution evidence	<code>LiquidationManager</code>
Governance operations	<code>GovernanceTimelock</code>

2.2 Controllers and services

A controller coordinates calls among canonical modules. A service performs durable orchestration, evidence ingestion, indexing, and presentation. Neither is permitted to become financial authority.

Definition 2.5 (Storage-free composition controller). *A controller C is storage-free with respect to financial state if its mutable storage contains no balance, share supply, debt, position, evidence receipt, withdrawal queue, reserve account, pause flag, or terminal financial state. Immutable module addresses and non-authoritative correlation events are allowed.*

Definition 2.6 (Durable financial saga). *A durable financial saga is a sequence of manifest-authorized canonical commands whose progress is recorded in an append-only inbox/outbox/journal. Retries read canonical state and may repeat idempotent commands, but no service-owned balance or debt state determines financial completion.*

2.3 Evidence boundary

The implementation programme is intentionally local-only: Anvil chain ID 31337, loopback services, emulated execution, test assets, and a hard 2× leverage ceiling. No production network, public testnet, remote RPC, real assets, external venue writes, public liquidity, live credit, DNS, or Vercel production promotion belongs to the evaluated scope. The evidence therefore concerns canonical composition under controlled local conditions, not production readiness. The four principal composition boundaries are summarized in Figure 1.

3 Canonical transition semantics and command admissibility

3.1 Global transition system

Let $x \in X$ be canonical financial state, m a sealed deployment manifest, r a role, and u a typed command. A command transition is a partial function

$$T_u : X \times M \times R \rightharpoonup X \times O,$$

Table 2: Authority context bound to canonical domains. Exact entity names are deployment-specific; the classes are part of the manifest identity.

Domain class	Entity role	Custody class	Permitted-use condition
Position/debt	protocol or li-censed/intermediated operator	collateral-locked or contract-controlled	leverage origination and debt service permitted
Venue execution	venue-authorized trader, delegate, or clearing agent	venue account or approved contract account	order entry, cancellation, and settlement permitted
Claim custody	claim custodian or contract-controlled outcome account	segregated claim custody	financed claim may be held, redeemed, and subject to lien/control
Credit/LBP/reserve	credit pool, backstop vehicle, or reserve owner	segregated protocol capital	loss absorption and withdrawal policy permitted
Liquidation	authorized liquidator/default manager	execution-only authority	registered risk-reducing transition permitted

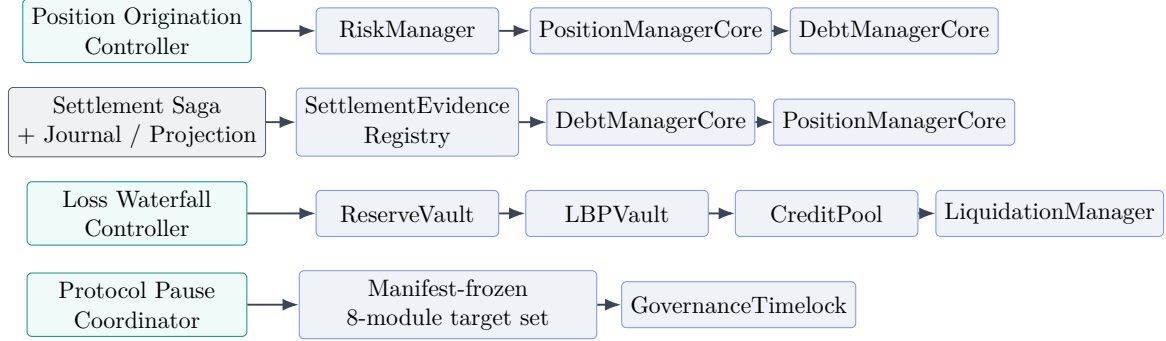


Figure 1: Canonical protocol graph. Each row shows one composition boundary; financial state remains in the domain modules and the controllers do not become a second ledger.

where O is the set of receipts and canonical events. Capability, legal-authority, and target-owned pause context are fixed evaluation parameters, suppressed in this notation; they are not inferred from service projections. The function is undefined when any required authorization or precondition fails.

The command is *financially admissible* only if six predicates hold:

$$G_M(u, m) = 1 \quad \text{manifest address and selector authorization,} \quad (1)$$

$$G_R(u, r) = 1 \quad \text{role and governance authorization,} \quad (2)$$

$$G_P(u, x) = 1 \quad \text{protocol preconditions in canonical state,} \quad (3)$$

$$G_K(u, x, \kappa) = 1 \quad \text{venue-capability freshness and continuity,} \quad (4)$$

$$G_L(u, x, \lambda) = 1 \quad \text{entity, custody, and permitted-use eligibility,} \quad (5)$$

$$G_B(u, x) = 1 \quad \text{economic and environment boundaries.} \quad (6)$$

The execution predicate is

$$G(u, x, m, r, \kappa, \lambda) = G_M G_R G_P G_K G_L G_B.$$

A failed predicate produces either an EVM revert or a typed service rejection whose evidence explicitly proves zero financial mutation. Economic availability cannot compensate for $G_L = 0$,

and a previously valid certificate cannot compensate for a capability or book-epoch transition that makes $G_K = 0$.

Definition 3.1 (Canonical command). *A canonical command is a typed command whose target address and selector are manifest-authorized, whose role is permitted, whose venue-capability and legal-authority contexts are valid for the required interval, and whose financial effect is determined exclusively by the canonical storage authorities.*

Proposition 3.2 (No off-manifest equivalence). *Two commands with the same user-facing label but different target addresses, selectors, role policy, or deployment-manifest identities are different canonical commands. Evidence for one does not prove the other.*

This distinction is important for upgrades and testing. A mock, a legacy bridge, and the canonical position manager may all expose a method called “open position,” but their state effects and authority differ. The manifest prevents interface similarity from being treated as execution equivalence.

Proposition 3.3 (Legal no-go safety). *If $G_L = 0$ and the command dispatcher and canonical modules enforce the manifest-bound legal context, then displayed collateral or credit capacity cannot originate a position, increase debt, or enlarge risk. The rejected command contributes zero financial mutation. Equality of the complete pre- and post-state additionally presumes an isolated execution without unrelated concurrent transitions.*

Proposition 3.4 (Capability-transition safety). *Suppose admission requires fresh capability state κ_{t_0} and the active position remains bound to book epoch b_{t_0} . If capability is stale before admission, the command is rejected with zero mutation. If $b_t \neq b_{t_0}$ while debt $D_t > 0$, risk-increasing authority is revoked and the system must enter a registered fail-closed or risk-reducing path before any further exposure increase.*

3.2 Financial state and projection state

Let x denote canonical state and y denote service/application projection state. The system evolves as

$$x_{t+1} = T_u(x_t), \quad y_{t+1} = P(J(x_{0:t+1})),$$

where J is the append-only journal of canonical observations and P is a deterministic materializer. There is deliberately no transition of the form $x_{t+1} = F(y_t)$ for ordinary projection state. A service can propose a command based on y_t , but the contract preconditions and canonical state decide whether it commits.

Proposition 3.5 (Projection rebuildability). *If the journal retains the manifest identity, command correlation, transaction and event identifiers, and all inputs required by P , then deleting y does not destroy financial state and y can be rebuilt without a financial transaction.*

3.3 Module interface map

Table 3 records the intended local composition surface. The public evidence companion records the FIA taxonomy, canonical result identities, and publication-safe registries used by the evaluated path; this article supplies the interface detail needed to establish the composition properties.

Table 3: Canonical module interface and authority map.

Module	Principal mutations	Canonical events / observations	Composition role
RiskManager	approve, consume, release risk capacity	approval, consumption, rejection reason, limits	sole admission authority
Position Manager Core	open, transition, release collateral	position opened, state changed, collateral released	position/collateral authority
Debt Manager Core	create debt, record match, apply settlement, residual accounting	debt created, match recorded, settlement applied	debt authority
Settlement Evidence Registry	configure adapter, submit, classify replay/conflict, record recovery	admitted, replay, exception, recovery	evidence authority
CreditPool	Senior deposit, queue/service withdrawal, apply approved loss	shares, queue, service, loss events	Senior capital authority
LBPVault	deposit, encumber, impair, queue/service	shares, encumbrance, impairment, queue	junior capital authority
ReserveVault	fund and allocate approved loss	funding and reserve allocation	reserve custody authority
Liquidation Manager	request, record execution, record allocation hash	request, execution, and allocation evidence	liquidation evidence authority
Governance Timelock	queue and execute authorized operations	operation queued/executed	governed recovery authority
Position Origination Controller	consume approval, open position, create debt atomically	correlation metadata only	zero-storage coordinator
Loss Waterfall Controller	call canonical loss selectors atomically	allocation amounts and digest	zero-storage coordinator
Protocol Pause Coordinator	pause frozen target set atomically	fan-out correlation only	zero-state coordinator

3.4 Cross-module invariants

The graph must preserve the following cross-module invariants:

- (I1) every open financed position has one bound debt record and a consumed approval;
- (I2) collateral cannot be released while canonical debt remains positive;
- (I3) debt cannot decrease without admitted settlement evidence;
- (I4) one evidence receipt cannot create two financial deltas;
- (I5) trader residual is released only after debt and admitted fees;
- (I6) reserve, LBP, and Senior loss application is ordered and conservative;

- (I7) queued shares remain in the relevant share supply until service and burn;
- (I8) a pause coordinator cannot report a state different from its targets;
- (I9) recovery cannot create a duplicate committed financial effect in the same operation domain;
- (I10) no service projection or controller becomes a financial ledger;
- (I11) capability freshness and book-epoch continuity condition every execution authority;
- (I12) legal entity, custody class, and permitted use are manifest-bound and can force a zero-mutation `LEGAL_NO_GO`.

These invariants form the implementation-level specialization of the accounting properties in Papers I and II rather than a replacement for them. Invariant (I8) is the cross-cutting counterpart of financial-domain authority uniqueness: the coordinator cannot substitute for, contradict, or aggregate away the target-owned pause vector $P_{\mathcal{T}}$.

4 Single authority and compositional closure

Definition 4.1 (Composition-level no-secondary-ledger property). *For every nonempty subset $S \subseteq \mathcal{D}$ and every finite composition Γ of controllers, adapters, services, and test utilities, the authoritative state on S is the projection of the states stored by $\{A(d) : d \in S\}$. No mutable state owned by Γ may determine, replace, or override an authoritative read or write on S independently of the canonical authorities; merely consulting and then overriding them is not sufficient.*

Assumption 4.2 (Manifest-complete call surface). *Every mutating command is directed to an address and selector present in a sealed deployment manifest. The manifest identifies all canonical authorities, authority contexts, venue-capability requirements, and allowed storage-free controllers.*

Theorem 4.3 (Financial-domain subset compositional authority uniqueness). *Suppose the authority map A is total and single-valued, every financial transition writes only to $A(d)$ for each affected domain d , and every component composition satisfies the composition-level no-secondary-ledger property. Then for every nonempty $S \subseteq \mathcal{D}$, the composed system has at most one authoritative representation of the joint financial state $X_S = \prod_{d \in S} X_d$.*

Proof. Assume a nonempty subset S has two authoritative joint representations. One is the canonical product of the states stored by $\{A(d) : d \in S\}$. The other must be determined by mutable state in some composition Γ or by a different storage-owning module. Because the alternative can determine or override an authoritative read or write on S independently of the complete canonical product, it violates the composition-level no-secondary-ledger property. The argument covers both a second per-domain ledger and a synthetic subset ledger that combines several domains. Correlation metadata, append-only observations, and rebuildable projections do not contradict the theorem because they are explicitly non-authoritative. $\square$

Proposition 4.4 (Pause-vector authority and coordinator non-authority). *Let $\mathcal{T}$ be the nonempty manifest-frozen pause target set. If each p_m is stored only by target module m , the coordinator owns no pause state, and fan-out reverts when any target call fails, then the authoritative pause state is exactly the indexed vector $P_{\mathcal{T}} = (p_m)_{m \in \mathcal{T}}$. A coordinator event or projection cannot establish a pause state different from its targets.*

Proof. Each component is owned by its target module and the coordinator stores none of them. Atomic fan-out prevents a committed strict subset in the successful path. Therefore invariant (I8) reduces pause verification to the target-owned indexed vector and ordered target events. $\square$

Algorithm 1 Atomic canonical origination

Require: manifest-authorized modules; valid approval a ; position request q

- 1: verify q references the same pool, market, account, leverage tier, and collateral as a
- 2: call `RiskManager.consumeApproval( $a$ )`
- 3: call `PositionManagerCore.openPosition( $q$ )`
- 4: call `DebtManagerCore.createDebt( $q$ )`
- 5: emit non-authoritative correlation metadata

Ensure: either all canonical calls commit or the EVM reverts the complete transaction

Corollary 4.5 (Projection non-authority). *A service, API, browser, or audit projection can be stale or unavailable without changing canonical financial state. Conversely, changing a projection alone cannot create debt repayment, collateral release, share issuance, impairment, or a target-owned pause transition.*

The result provides a testable architecture criterion. A bridge that stores its own scenario state, queue shares, pause flag, or evidence manager violates the criterion even if its terminal hashes equal a fixture. Such a bridge may remain useful as a harness, but it cannot be promoted to the canonical graph.

5 Atomic origination

Position origination consumes risk capacity, locks trader collateral, and creates debt. A partial completion is unsafe: an approval could be consumed without a position, a position could exist without debt, or debt could exist without collateral.

Let a be an approval, p a position, and d a debt record. The intended transition is

$$T_{\text{orig}} : (a, p, d) \mapsto (a', p', d'),$$

where a' is consumed, p' is open with locked collateral, and d' contains the principal and authorized terms.

Theorem 5.1 (Atomic origination closure). *If the three canonical calls in Algorithm 1 execute in one EVM transaction, each successful call realizes its specified effect, and every failure is propagated so that the complete transaction reverts, then its committed post-state cannot contain exactly one or two of the three effects: approval consumption, position opening, and debt creation.*

Proof. EVM transaction semantics apply state updates atomically [18]. If any call reverts and the revert is not swallowed, all prior state changes in the transaction are reverted. Therefore the only reachable outcomes are the complete pre-state or the complete committed post-state. $\square$

The result concerns the committed transaction boundary, not an arbitrary intermediate callback or an external venue action. Caught exceptions and unchecked low-level failure returns do not satisfy its assumptions [14]. The controller owns immutable addresses only. It does not cache approval state, position state, or debt. The manifest therefore becomes part of the correctness boundary: calling a different address or selector is a different system, not the same origination under another configuration.

Algorithm 2 Durable canonical settlement saga: admission is not application

Require: command c , evidence e , sealed manifest M ; receipt/application uniqueness

- 1: append command and its stable idempotency key to the durable inbox
- 2: submit or reconcile e at `SettlementEvidenceRegistry`
- 3: **if** receipt is conflicting, inadmissible, or its canonical status is uncertain **then**
- 4: retain exception/reconciliation evidence
- 5: **return** without a new financial mutation
- 6: **end if**
- 7: read the canonical application status for the same settlement domain and payload
- 8: **if** the settlement effect is already committed **then**
- 9: do not reapply it; read canonical debt and position
- 10: **else if** receipt has a valid admission and application is reliably not yet committed **then**
- 11: call the atomically guarded `DebtManagerCore.applyAdmittedSettlement`
- 12: **else**
- 13: **return** for reconciliation; do not invent a new key or assume application
- 14: **end if**
- 15: confirm the application outcome, or return for reconciliation
- 16: reconcile any still-required permitted `PositionManagerCore` transition
- 17: persist the confirmed application checkpoint; rebuild missing projections
- 18: evaluate residual/collateral release separately under its own canonical eligibility guard

6 Settlement evidence and durable debt finalization

6.1 Why settlement is not one transaction

External execution and settlement can arrive asynchronously. An emulator or venue adapter reports match and settlement evidence; the evidence registry classifies a receipt; the debt manager applies admitted value; the position manager transitions the lifecycle; and residual/collateral release follows only after canonical getters prove finality. Compressing all steps into one transaction is neither always possible nor desirable. The appropriate abstraction is a durable saga [5, 7].

For a unique evidence identifier e , let

$$R(e) \in \{\text{new, admitted, replay, conflict, recovered}\}$$

be the registry state. Let $\Delta D(e)$ be the debt reduction applied from e .

Assumption 6.1 (Receipt and application uniqueness). *A settlement observation is bound to a stable logical settlement domain and payload. Reusing the domain cannot admit conflicting content or authorize another application by changing only a transport identifier. Evidence admission and financial application are distinct facts. In the debt authority, checking the applied status, recording application, and mutating debt form one atomic guarded transition for that domain. The result below is conditional on this application guard; admission uniqueness alone is insufficient.*

The durable settlement sequence is formalized in Algorithm 2. The algorithm distinguishes the existing registry status from the debt manager’s existing application status; it introduces no new module or receipt type.

Theorem 6.2 (Receipt-idempotent debt application). *Under receipt and application uniqueness, the same admitted settlement can produce at most one committed debt application. A*

retry after its successful application produces zero additional debt reduction. An admitted but unapplied receipt remains eligible for its first guarded application; registry replay alone does not establish debt repayment.

Proof. The first successful application atomically records the applied state with the debt mutation. Every concurrent or later call in the same settlement domain therefore either supplies that first application or encounters the already-applied guard and produces no duplicate effect. If only receipt admission committed, the applied guard has not yet been consumed. Treating that state as completed would omit the first application rather than prove idempotence. $\square$

Corollary 6.3 (Conflict safety). *If conflicting evidence cannot enter the admitted branch, a conflict cannot reduce debt, release residual, or release collateral.*

6.2 Residual and collateral release

Settlement-confirmed value follows the debt-first rule from Paper II [9]. Let $y \geq 0$ be admitted settled cash, $D \geq 0$ the outstanding debt due under the registered repayment rule (including accrued interest classified as debt), and $F \geq 0$ separately admitted fees. Amounts in D are not counted again in F . Then

$$y^D = \min\{y, D\}, \quad y^F = \min\{[y - y^D]^+, F\}, \quad y^T = [y - y^D - y^F]^+.$$

The service cannot infer debt clearance from a local projection. Release is allowed only if canonical getters prove zero principal, zero accrued interest, no pending residual precondition, and a final debt-free lifecycle state.

7 Canonical loss waterfall

A liquidation request and an actual allocation are different operations. The liquidation manager records the lifecycle and execution evidence; the canonical reserve, LBP, and Senior modules own the capital balances.

Let the recognized shortfall be $\ell \geq 0$ and the ordered capacities be

$$(c_0, c_R, c_J, c_G, c_S),$$

where the capacities are nonnegative and non-overlapping: c_0 is position-specific equity or buffer, c_R segregated reserve, c_J eligible LBP capital, c_G a configured global reserve, and c_S approved Senior loss capacity. The shortfall ℓ is measured before these listed layers are applied; a buffer already netted from the shortfall is not counted again. In the initial local deployment, absent c_0 and c_G are explicit zero layers.

Define recursively

$$a_k = \min \left\{ c_k, \left[\ell - \sum_{j < k} a_j \right]^+ \right\}.$$

Proposition 7.1 (Unique ordered allocation). *The vector $a = (a_0, a_R, a_J, a_G, a_S)$ is unique, satisfies $0 \leq a_k \leq c_k$, preserves priority, and obeys*

$$\sum_k a_k + \ell_{\text{res}} = \ell,$$

where $\ell_{\text{res}} \geq 0$ is unabsorbed loss.

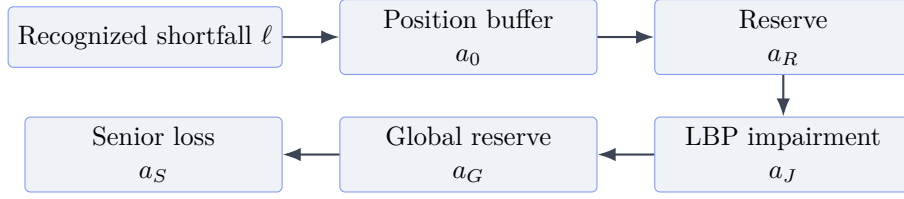


Figure 2: Canonical ordered waterfall. Zero or absent layers remain explicit in the deployment manifest rather than being silently omitted.

Proof. The recursion uniquely determines each a_k from prior allocations. The minimum enforces capacity, the positive part enforces non-negativity, and later layers receive positive loss only after earlier capacities are exhausted. Conservation follows by definition of the residual. $\square$

Theorem 7.2 (Atomic multi-ledger waterfall). *Suppose a storage-free controller computes a through a pure, versioned allocation library and invokes canonical reserve, LBP, and Senior loss selectors in one EVM transaction. Assume that every successful canonical call applies exactly its prescribed component of a ; every mismatch, insufficient authorization, or required-call failure propagates to a revert of the root transaction; and no later action in that transaction can cancel or distort an allocation component already applied. Then the committed post-state either reflects the entire ordered allocation a or no layer mutation. The controller cannot retain loss capital.*

The root-revert condition is essential: a controller that catches or ignores a failed child call does not satisfy the theorem’s assumptions. Under the stated conditions, EVM transaction atomicity rolls back every earlier layer mutation whenever any required component cannot be applied; on success, the exact-component and non-distortion conditions preserve the ordered allocation computed in Proposition 7.1.

The explicit layer ordering and zero-layer visibility are illustrated in Figure 2.

The distinction between eligible loss-absorbing junior capital J^{abs} and withdrawal-service headroom J^{free} is load-bearing. The waterfall may use J^{abs} , not an operational liquidity proxy. A unit of LBP capital cannot simultaneously count as protection capacity, free withdrawal liquidity, and another certificate allocation.

8 Loss-participating withdrawal queues

A withdrawal request is denominated in shares. Until actual service and burn, queued shares remain in the canonical share supply and participate in subsequent impairment. This property prevents an already recognized loss from being transferred to slower providers by a first-exit request [9].

Let z_i be shares owned by provider i , $q_i \leq z_i$ queued shares, and V provider NAV. The queued claim at time t is not a fixed cash amount; it is

$$W_{i,t} = q_i \pi_t, \quad \pi_t = \frac{V_t}{Z_t},$$

for $Z_t > 0$, subject to holdbacks, encumbrance, and available cash at service. A loss before service changes V_t and therefore changes the queued claim. Capital still reserved to absorb loss is not automatically available for withdrawal.

Proposition 8.1 (Queue participation). *If queued shares remain included in Z_t until service and burn, and losses reduce canonical NAV before service, then requesting withdrawal cannot exempt the queued shares from that loss.*

The accepted FIA-09 record exercises the actual `Senior CreditPool` queue, not a bridge-owned mapping. The analogous `LBPVault` queue remains a successor evidence item rather than part of the accepted record. The evidence must show ordering, queue head, share supply, loss application, service, and burn from canonical views and events.

9 Protocol-wide pause and governed recovery

The protocol has multiple modules with separate pause states. A local bridge-level flag is not sufficient: risk-increasing calls could remain available on an unpaused canonical module.

Let $\mathcal{T}$ be the manifest-frozen target set. Each mutating selector is classified as

`RISK_INCREASING`, `RISK_REDUCING`, `RECOVERY_ONLY`, `ADMIN_ONLY`.

The guardian may atomically pause risk-increasing operations across $\mathcal{T}$. The coordinator stores no pause flag; the authoritative state remains in each target module. Unpause requires a queued and executed `GovernanceTimelock` operation.

The execution policy also distinguishes two order treatments. Under `PERSIST_THROUGH_PAUSE`, an already-submitted order may remain only when its persistence is demonstrably risk-reducing under the paused capability state. Under `CANCEL_ON_PAUSE`, an order that could increase risk is cancelled, but such an order cannot support a certificate whose hard-flat horizon spans the pause. A pause is not a book reset: reactivation with a new book epoch and post-close lock are separate capability transitions.

Proposition 9.1 (Pause-policy certificate consistency). *A hard-flat or reduce-only certificate may rely on an order across a pause only under `PERSIST_THROUGH_PAUSE` and only if the order remains risk-reducing in the registered pause mode. If policy requires `CANCEL_ON_PAUSE`, the order contributes zero certified capacity beyond the pause boundary.*

Theorem 9.2 (Atomic pause fan-out). *If the coordinator calls every target in one EVM transaction and reverts when any target fails, then the transaction cannot commit a strict subset of the target pauses.*

Proposition 9.3 (Recovery-path availability under selective pause). *If the selector policy preserves every transition needed by the registered risk-reducing path while blocking new risk, a pause does not by itself remove that path’s callable selectors. Completion additionally requires admissible evidence, sufficient applicable capacity, canonical availability/finality, and eventual scheduling and retries. Selector availability alone does not prove eventual closure.*

The emergency control and delayed recovery sequence are summarized in Figure 3.

10 Exactly-once recovery

Distributed systems cannot guarantee exactly-once message delivery in the abstract. The relevant property is narrower: a financial transition that has already committed on-chain must not commit a second time when a worker restarts, while the off-chain projections must remain rebuildable.

Let τ be a transaction and e its evidence receipt. Let C denote a durable chain-confirmed checkpoint recorded before projection update. The recovery sequence is

$$\begin{aligned} \tau \text{ confirmed} &\rightarrow C \text{ persisted} \rightarrow \text{worker crash} \rightarrow \text{new worker replay} \\ &\rightarrow \text{replay/no-op} \rightarrow \text{projection rebuild.} \end{aligned}$$

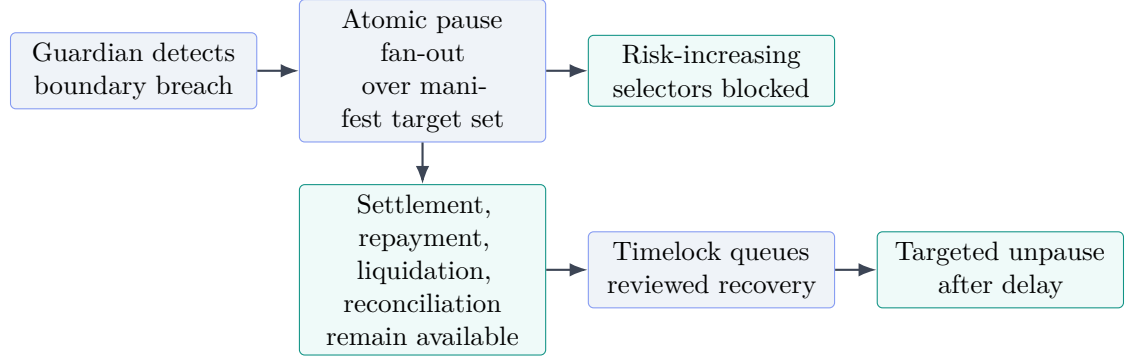


Figure 3: Emergency pause and governed recovery. The coordinator fans out commands but does not own the canonical pause state.

Definition 10.1 (Financial exactly-once recovery). *For a stable operation domain, key, and payload binding, recovery is financially exactly-once when there is at most one successful committed financial effect, retries are state-inert or read that effect, and derived projections are rebuildable from retained canonical observations. For an operation already known to have succeeded, the effect count is exactly one. Transaction submissions, reverted or no-op attempts, acknowledgements, and distinct settlement/release operations are not duplicate financial effects.*

Theorem 10.2 (Checkpointed no-reexecution). *Assume the confirmed checkpoint binds the manifest, stable operation identity, payload, committed application, receipt, and evidence identifier; retries query canonical application state; and the financial selector enforces the atomic application guard. A restart after that checkpoint cannot commit a duplicate financial effect for the same operation.*

Proof. The checkpoint identifies an already-committed application. Canonical lookup and the same-key guard either return that result or reject/no-op a repeated application. Such attempts can have their own transaction identifiers but cannot create another committed financial delta. Projection rebuilding does not consume another financial operation. $\square$

The theorem is explicitly post-checkpoint. It does not, by itself, establish recovery in every earlier crash window. In particular, an admitted receipt is not an applied settlement, and an unavailable or unfinalized lookup is not reliable evidence of absence. The at-most-once safety argument is conditional on the canonical guard even when a service checkpoint is missing; no additional crash-window execution is reported here. Eventual application requires separate liveness assumptions. A local ledger also cannot make a non-atomic external-venue effect exactly-once merely by retrying a local command.

The retained FIA-010 fixture has the narrower observation of one canonical financial transaction, zero duplicate financial delta, old/new process identities, a durable checkpoint, and rebuilt projections. That historical observation is preserved, not elevated into a universal bound on transaction submissions. Settlement application (FIA-004) and eligible residual release (FIA-005) remain distinct valid operations.

The public verifier has a narrower scope than a runtime replay: the inspected companion checks selected fields and published-file integrity, while withheld runtime identities remain attested. A source inspection at the separately pinned September-review snapshot confirms this distinction [3]; it is not a rerun of the private protocol.

11 Deterministic deployment manifest

The graph is identified by more than source code. Addresses, bytecode, ABIs, storage layouts, role assignments, pause targets, selectors, adapter identity, API schema, and generated-client identity collectively determine the system under test.

Definition 11.1 (Canonical deployment manifest). *A canonical deployment manifest M is a canonicalized record containing at least:*

- (i) *chain and execution-mode identity;*
- (ii) *workspace commit or tag;*
- (iii) *every deployed canonical module and controller address;*
- (iv) *creation and deployed bytecode hashes;*
- (v) *ABI and storage-layout hashes;*
- (vi) *role assignments and selector policy;*
- (vii) *pause target set;*
- (viii) *emulator adapter identity;*
- (ix) *OpenAPI, schema, and generated-client hashes;*
- (x) *a separately stored manifest digest, excluded from the manifest body that it hashes.*

The manifest body excludes its own digest and downstream receipts or package digests. Canonicalization operates on that body, not a self-referential wrapper. The canonical serialization must be fixed, for example RFC 8785 JSON canonicalization [13]; the retained release codec and test vectors, not mathematical tuple notation alone, determine historical bytes. No existing manifest or digest is recomputed or replaced by this clarification. The full stack fails closed if runtime addresses, bytecode, role, selector, schema, or adapter identity differs from M . This resembles supply-chain provenance systems in which a result is meaningful only under a sealed build and dependency identity [12, 15, 16].

Proposition 11.2 (Manifest-bound command safety). *If every service command checks that its target address and selector are authorized by M , then a command directed to a stale deployment, unregistered controller, forbidden selector, or legacy bridge is rejected before financial execution.*

12 Canonical state projection

A final hash is useful only if its source state is well defined. Fixture expectations are permitted only as simulator outputs; they cannot be copied into an actual layer.

For scenario s and terminal block b , define the canonical projection

$$\Pi(s, b, M) = (M, \{V_m(b)\}_{m \in \mathcal{M}_c}, \mathcal{E}(s, b)),$$

where $V_m(b)$ is the set of authorized ABI views from canonical module m , and $\mathcal{E}(s, b)$ is the ordered event list from manifest addresses, sorted by

(block number, transaction index, log index).

The state hash is

$$h_{\text{contract}} = H(\text{JCS}(\Pi(s, b, M))).$$

Proposition 12.1 (Fixture exclusion). *If Π is derived from authenticated, retained canonical observations at the registered boundary, and its actual-state inputs are not copied directly or indirectly from the simulator expectation, agreement of their declared semantic projections is an observed comparison. Excluding only the expected digest is insufficient: copying the expected state and rehashing it would preserve the same defect.*

The remaining layers—journal, service, API, browser, and audit reconstruction—must retain their actual inputs and root ancestry. Agreement is tested on the registered common semantic projection, not on equality of heterogeneous raw artifact digests. Derivation remains necessary even when normalized values agree. Independent runtime participation would require an additional claim-specific corroboration gate; that stronger gate was not evaluated for the registered local cohort and is `NOT_EVALUATED`, not a failure of its accepted local profile.

13 FIA evaluation surface

The twelve Financial Interaction Assertions form the intended end-to-end surface. Table 4 states the canonical questions, not final results.

Table 4: FIA scenarios and canonical proof obligations.

FIA	Scenario	Canonical proof obligation
01	successful admission	risk approval, atomic position/debt origination, manifest-bound receipt, projection chain
02	exact 2× admission	discrete leverage policy, exact calldata, same atomic graph
03	risk rejection	typed rejection and zero financial mutation
04	partial then final settlement	admitted partial and final evidence, bounded debt application, lifecycle closure
05	residual and collateral release	release only after FIA-04 establishes debt-clearance eligibility
06	liquidation to hard-flat	authorized liquidation request/execution and hard-flat transition
07	liquidation and waterfall	request/execution, reserve-LBP-Senior ordering, conservation
08	evidence conflict	canonical exception/quarantine and zero debt mutation
09	withdrawal queue	real pool/vault queue and loss participation until service/burn
10	exactly-once recovery	chain checkpoint, process restart, one financial transaction, rebuild
11A	stale capability before admission	VenueRegistry/RiskManager typed rejection, causal freshness check, and zero mutation
11B	book-epoch change with $D_t > 0$	epoch transition, risk-increase revocation, fail-closed/risk-reducing path, and retained causal evidence
12	pause and recovery	frozen targets, guardian fan-out, pause-policy semantics, timelocked recovery/unpause

Each FIA is complete only when the role-specific UI or typed test client issues a command, the API and durable saga bind the command to canonical selectors and receipts, canonical state is observed, and the journal, service, API, browser, and audit layers retain registered derivations. The parent denominator remains twelve. FIA-011 is conjunctive: it cannot pass unless both FIA-011A and FIA-011B have retained raw and normalized evidence, causal checks, negative controls, replay, and explicit report rows.

14 Implementation mapping and verification obligations

14.1 FIA-to-module touch matrix

Table 5 states the minimum module participation required for each scenario. A final evidence package must show actual manifest-authorized calls and events, not only a test fixture that describes them.

Table 5: Minimum canonical module coverage by FIA.

FIA	Required authorities	Financial effect	Evidence emphasis
01	Risk, Position, Debt, Evidence	admitted $1\times$ position and debt within the $2\times$ ceiling	atomicity and correlation
02	same as FIA-01	exact discrete $2\times$ policy	exact command and calldata identity
03	Risk and capability inputs	no mutation	typed rejection and before/after equality
04	Evidence, Debt, Position	partial then final admitted settlement and debt/lifecycle closure	bounded application and distinct evidence identities
05	Debt, Position	one-time trader residual and collateral release after debt-clearance eligibility	ordered release preconditions
06	Liquidation, Position, Evidence, Debt	authorized liquidation to hard-flat	request/execution identity and close-horizon transition
07	Liquidation, Reserve, LBP, Credit	ordered canonical loss allocation	conservation and allocation hash
08	Evidence, Debt	exception/quarantine, zero debt delta	conflict/replay domain
09	CreditPool	Senior withdrawal request remains loss-participating until service/burn	queue ordering and NAV; LBP queue is successor evidence
10	Evidence, Debt, service persistence	one transaction, rebuilt projections	checkpoint and process identities
11A	VenueRegistry, RiskManager	no mutation	stale capability, typed cause, and freshness provenance
11B	VenueRegistry, RiskManager, Position, Debt	no risk increase; fail-closed/risk-reducing transition	book epoch, positive-debt causal chain, and retained transition evidence
12	all frozen pause targets, Timelock	pause and governed recovery	atomic fan-out, pause-policy mode, and delayed unpause

14.2 Verification layers

For every scenario, the registered local programme retains seven evidence layers with registered derivations; claims of independent runtime participation would additionally require the proposed multi-root corroboration gate, which was not evaluated for this cohort:

1. simulator expectation;
2. canonical contract state from ABI views and ordered canonical events;
3. append-only journal;
4. service materialization;
5. authenticated API response;
6. browser projection after a real role control or typed browser client action;
7. Audit Explorer reconstruction.

The expected simulator hash is not an input to layers 2–7. The contract state collector accepts only addresses in the deployment manifest. Journal ordering references block, transaction, and log coordinates rather than receipt-file order.

Definition 14.1 (Canonical E2E completion). *A scenario is canonically complete only if its typed command enters through the approved role boundary, every required financial effect occurs in the canonical graph, all seven layers are retained and correlated, mandatory negative controls pass, and no forbidden authority participates.*

14.3 Differential verification

The pure allocation library, reference waterfall, exact simulator, and canonical contract transition can be compared without granting the reference model financial authority. The comparison is

$$\text{normalize}(x_s^{\text{sim}}) = \text{normalize}(x_s^{\text{contract}}),$$

for registered scenario s . Disagreement may reveal a contract defect, reference-model defect, or underspecified normalization. It is reported rather than averaged away.

14.4 Failure classification

The final report should distinguish:

- **FULL_E2E**: complete canonical graph and evidence;
- **READ_ONLY_E2E**: full read/reconstruction without required financial command;
- **CANONICAL_REPLAY_ONLY**: deterministic contract replay without full application path;
- **BOUNDARY_BLOCKED**: a declared environment, data-rights, or authority boundary prevents completion;
- **FAILED**: the intended transition or evidence property is contradicted.

These labels describe distinct profiles, not a requirement to satisfy every conceivable production gate. The accepted local result is a pass within its registered gate set. A missing required gate is incomplete inside that scope; a broader untested claim is **NOT_EVALUATED**. An expected typed rejection can be a local pass without a committed financial transaction. No label in an immutable source schema is rewritten by this explanatory distinction. A scenario cannot be promoted by renaming a harness or copying projection hashes.

15 Proof obligations for the final implementation

The final source and evidence review should discharge the following obligations.

O1: controller storage exclusion. Storage layout and source review show that new controllers contain immutable module addresses and non-authoritative metadata only.

O2: manifest completeness. Every canonical module, controller, role, selector, adapter, pause target, ABI, bytecode, and schema identity used in a scenario is present in the sealed manifest.

O3: authority reachability. Every financial mutation in the scenario occurs in the declared authority module; no legacy bridge or test evidence manager participates.

O4: waterfall conservation. Before/after vault values and events satisfy the registered allocation and conservation equation.

O5: queue persistence. Queued shares remain in supply and participate in impairment until canonical service and burn.

O6: replay safety. Duplicate and conflicting evidence create no unauthorized financial delta.

O7: recovery exactly-once. A confirmed application checkpoint survives worker termination; a fresh worker reconstructs state without a duplicate committed effect. The one-transaction count remains the narrower retained FIA-010 fixture result.

O8: pause coverage. The target set is complete, fan-out atomic, risk-reducing selectors remain available where specified, and unpause requires the timelock.

O9: derivation topology and corroboration. Each evidence layer retains its inputs, derivation, and root ancestry; a fixture expectation is never copied into an actual layer. A claim of independent runtime participation requires a separately specified multi-root corroboration predicate rather than pairwise relabeling; that proposed predicate was not evaluated for this cohort.

O10: reproducibility. Two clean complete packages and the Phase 1–17 replay meet their declared raw/normalized identity policies and survive independent review.

O11: capability continuity. FIA-011A and FIA-011B each retain the exact capability/book-epoch transition, causal checks, negative controls, replay result, and explicit report coverage; the parent FIA-011 remains non-passing unless both subcases pass.

O12: legal authority binding. Every risk-increasing command binds the canonical module to an eligible entity role, custody class, and permitted-use class; the additive `LEGAL_NO_GO` case proves zero mutation when displayed economic capacity is legally ineligible.

A Additional proof details

A.1 Composition of storage-free controllers

Let C_1 and C_2 be storage-free controllers over disjoint or overlapping canonical domains. Their sequential composition $C_2 \circ C_1$ does not itself create a second ledger because neither controller stores authoritative financial state. However, correctness still requires that the composed selector order satisfy the protocol invariants. Storage freedom is necessary for authority uniqueness but not sufficient for semantic correctness.

Remark (Why an omnibus controller is discouraged). A large controller can remain storage-free and still become operationally opaque, hard to pause, and difficult to audit. The architecture therefore prefers narrow controllers aligned with one atomicity boundary. The rule is not only “no storage” but also “one reviewed reason for atomic composition.”

A.2 Saga safety versus atomic safety

Atomicity prevents partial state within one transaction. A saga permits intermediate canonical states. Its safety therefore depends on registered intermediate states, idempotency, compensation or recovery paths, and a durable progress record. In the settlement saga, an admitted evidence receipt and a chain-confirmed debt application may exist before the browser updates. That is acceptable because financial authority is already canonical and the projection is rebuildable. It would not be acceptable for the browser or service database to release collateral before canonical debt clearance.

A.3 Exactly-once financial effect

The exactly-once property can be restated as a linearizability condition around the chain-confirmed transaction: the financial command linearizes at the committed canonical transaction, while later retries linearize as reads/no-ops of the same completed command. The service may process the message more than once; the financial effect occurs once.

A.4 Explicit zero layers

Representing an absent position buffer or global reserve as an explicit zero layer prevents two ambiguities: whether the layer was intentionally disabled or accidentally omitted, and whether a later configuration can insert the layer without changing allocation semantics. The manifest therefore records both presence and configured capacity.

B Registered local evidence result

Registered local evidence result. One accepted cohort contains twelve uniquely correlated FIA records and 84 retained `ACTUAL_EVIDENCE` layer positions. Two clean materializations matched at the registered byte/file-index boundary; a deterministic strict archive was accepted by a separate verifier; and a selected ordered replay is bound to that archive.

The evidence companion documents the registered local cohort [2]. Its retained aggregate records support the counts and derivation classes evaluated here; confidential operational identifiers remain outside the distributed evidence boundary.

The completed evidence milestones and their retained identities are summarized in Table 6.

The all-12 ledger contains one uniquely correlated record for each FIA and seven retained `ACTUAL_EVIDENCE` positions per record. Two clean materializations share one byte-identical canonical file index, and a separate verifier accepted the registered deterministic archive. A selected ordered replay is content-bound to that archive. These findings are reported at the registered-local level; confidential materialization, package, and replay identifiers are outside the distributed evidence boundary.

FIA-011 remains one parent assertion in the denominator of twelve and retains both mandatory subcases: stale capability before admission and book-epoch change after position opening while debt remains positive. The retained A/B envelope and focused adapter tests prevent a generic FIA-03 rejection or a one-receipt placeholder from satisfying the parent.

B.1 Claim typing

The observed result is the completed cohort-bound evidence chain described above. Its interpretation is that the canonical protocol graph can preserve distinct financial assertions

Table 6: Registered local evidence milestones.

Milestone	Observed result	Evidence description
Source-owned FIA evidence	five non-omnibus adapter groups; registered local evidence reports completed coverage	publication-safe cohort record
All-12 cohort	12 records, 7 retained layers each, 84 positions	registered local cohort
Clean materialization	two fresh trees; one byte-identical file index	registered materialization comparison
Strict archive	separate verifier accepted the registered archive	registered local package
Ordered replay	selected replay is bound to the registered archive	registered local replay

through common manifest, materialization, packaging, and replay identities. Public-network operation, external-venue execution, real custody, terminal oracle settlement, and production governance are successor evidence programmes rather than conclusions inferred from this cohort.

C Security analysis

C.1 Threat model

The relevant adversary may:

- direct a service command to a stale or unauthorized address;
- substitute a selector with the same user-facing meaning but different state effects;
- replay or conflict settlement evidence;
- mutate one layer of the evidence package while preserving others;
- restart workers between chain confirmation and projection update;
- attempt partial pause, premature unpaue, or risk increase during recovery;
- exploit a second ledger in a controller, service, or test bridge;
- count the same reserve or LBP capital in multiple certificates or withdrawal headroom.

The architecture addresses these risks through unique authority, manifest binding, atomic controllers, receipt idempotency, canonical queues, explicit capital eligibility, and fail-closed evidence promotion. It does not solve oracle truth, venue insolvency, stablecoin failure, key compromise, legal priority, or undiscovered contract vulnerabilities.

C.2 Upgradeability and code identity

The public evidence companion records the tested source and manifest identities but does not convert that snapshot into a general immutability claim. A production release must still identify immutable, proxy-based, and replaceable modules and specify the authorized manifest transition under upgrades.

C.3 Economic non-duplication

The current stateful integrated waterfall remains a reference oracle rather than canonical capital storage. Differential tests may compare the pure allocation result with the reference model, but the production-path controller must mutate only the real reserve, LBP, and Senior modules. This distinction is essential: mathematically equal outputs do not justify two authoritative balance sheets.

D Relation to prior work

The no-secondary-ledger principle specializes transaction-processing and linearizability ideas to modular smart-contract finance [5, 6]. Ethereum provides atomic transaction semantics but does not automatically provide a correct multi-contract architecture [18]. DeFi lending systems demonstrate on-chain pool shares, utilization, liquidation, and reserve factors [1, 8, 17]; event-margin protocols add time-bounded execution, external settlement evidence, and debt-free finality. Financial-network models motivate explicit loss ordering and isolation [4]. The present contribution is the combination of single-domain authority, storage-free composition, durable evidence sagas, manifest-bound deployment, and independently reconstructible state across a leveraged event-market graph.

E Reproducibility and evidence availability

The public evidence companion contains sanitized FIA definitions, schemas, checksums, provenance records, and an offline verifier for published structure and metadata. Raw runtime payloads and operating-system paths are excluded. The verifier recomputes repository-file checksums and checks selected required fields, counts, ordering, canonical identifiers, and internal consistency; payload-derived identities remain attested because their raw roots are not public.

F Limitations

The paper establishes a formal composition architecture and an observed cohort-bound evidence result. It does not estimate production failure rates, external venue compatibility, real-asset custody, human usability, regulatory compliance, or economic profitability. Those are separate empirical and operational questions.

G Conclusion

A leveraged event-market protocol is not a collection of contracts; it is a single financial state machine whose authority must remain unambiguous across modules, services, and recovery. The architecture developed here assigns one storage owner to every financial domain, confines composition logic to storage-free controllers, uses atomic transitions only where partial completion is unsafe, uses durable idempotent sagas where evidence arrives asynchronously, and binds the complete system to a deterministic manifest. The resulting design makes it possible to state exact proof obligations for origination, settlement, liquidation, queues, pause, and recovery. The registered local evidence chain shows how the twelve FIA assertions, clean materialization, strict packaging, and ordered replay can be bound without turning controllers or services into a second financial ledger.

Funding

No external funding was received for this research.

Competing Interests

A related pending UAE patent application is disclosed in the Patent Disclosure section.

Data and Code Availability

The article reports a registered local engineering-evidence cohort. The retained public evidence companion supports recomputation of its published structure and metadata; it is not an external dataset deposit, a production record, or a live-venue data source.

JEL: C63, G13, G14, G23.

AI Disclosure

OpenAI ChatGPT and Codex were used for editorial and technical assistance during manuscript preparation. The author made all substantive research decisions, reviewed the final manuscript, and assumes full responsibility for its contents.

Patent Disclosure

Certain technical aspects described in this article are included in a UAE patent application filed on 30 September 2026 (Application No. MOE-Patents-PTA-0003399-20260930). The application is pending; no granted patent is claimed.

References

- [1] Aave. Aave V3 Technical Paper. Technical report, Aave, 2022. Official technical paper; archived citation record, accessed 12 August 2026.
- [2] AxientLab. Axient FIA Public: Reproducibility and Evidence Companion. Public GitHub repository, 2026. URL <https://github.com/AxientLab/axient-fia-public/tree/9a473fbd49cbd4f95c64bb0bd090bbdaeac3f442>. Public evidence companion; accessed August 2026.
- [3] AxientLab. Axient FIA Public: Verification Scope and Offline Verifier. Public research companion, source inspection, 2026. URL https://github.com/AxientLab/axient-fia-public/blob/a0f0c510e527942a42f5cac6a34bc06c12d23f04/VERIFICATION_SCOPE.md. Public companion commit a0f0c510e527942a42f5cac6a34bc06c12d23f04; inspected 30 September 2026.
- [4] Larry Eisenberg and Thomas H. Noe. Systemic risk in financial systems. *Management Science*, 47(2):236–249, 2001. doi: 10.1287/mnsc.47.2.236.9835. URL <https://doi.org/10.1287/mnsc.47.2.236.9835>.
- [5] Jim Gray and Andreas Reuter. *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann, 1993.

- [6] Maurice P. Herlihy and Jeannette M. Wing. Linearizability: A correctness condition for concurrent objects. *ACM Transactions on Programming Languages and Systems*, 12(3): 463–492, 1990. doi: 10.1145/78969.78972. URL <https://doi.org/10.1145/78969.78972>.
- [7] Martin Kleppmann. *Designing Data-Intensive Applications*. O’Reilly Media, 2017.
- [8] Robert Leshner and Geoffrey Hayes. Compound: The Money Market Protocol. Technical report, Compound Labs, February 2019. URL <https://compound.finance/documents/Compound.Whitepaper.pdf>. Version 1.0; accessed 12 August 2026.
- [9] Maksym Nechepurenko. Axient: On-Chain Credit and Loss Allocation for Leveraged Event Markets: A Venue-Agnostic Protocol for Traders, Credit Providers, Market Makers, and Liquidation Backstops. *SSRN Electronic Journal*, 2026. doi: 10.48550/arXiv.2608.00647. URL <https://ssrn.com/abstract=7216081>. SSRN abstract 7216081; arXiv:2608.00647.
- [10] Maksym Nechepurenko. Axient: Debt-Free Finality for Leveraged Binary Event Markets. *SSRN Electronic Journal*, 2026. doi: 10.48550/arXiv.2608.00631. URL <https://ssrn.com/abstract=7216080>. SSRN abstract 7216080; arXiv:2608.00631.
- [11] Maksym Nechepurenko. Axient: Empirical Calibration of Venue-Agnostic Event-Margin Protocols: A Prospectively Frozen Analysis Plan, Deterministic Internal-Alpha Technical Pilot, and Venue-Emulator Methodology. *SSRN Electronic Journal*, July 2026. doi: 10.2139/ssrn.7216083. URL <https://ssrn.com/abstract=7216083>. SSRN abstract 7216083; 30 July 2026.
- [12] Open Source Security Foundation. Supply-chain Levels for Software Artifacts (SLSA) Specification, Version 1.0, 2023. URL <https://slsa.dev/spec/v1.0/>. Final specification, 19 April 2023; accessed 12 August 2026.
- [13] Anders Rundgren, Bret Jordan, and Samuel Erdtman. JSON Canonicalization Scheme (JCS). Technical Report RFC 8785, Internet Engineering Task Force, 2020. URL <https://doi.org/10.17487/RFC8785>.
- [14] Solidity Contributors. Expressions and Control Structures: Error Handling, 2026. URL <https://docs.soliditylang.org/en/latest/control-structures.html>. Official documentation; inspected 30 September 2026.
- [15] Murugiah Souppaya, Karen Scarfone, and Donna Dodson. Secure Software Development Framework (SSDF) Version 1.1. Technical Report NIST SP 800-218, National Institute of Standards and Technology, 2022. URL <https://doi.org/10.6028/NIST.SP.800-218>.
- [16] Santiago Torres-Arias, Hammad Afzali, Trishank Karthik Kuppusamy, Reza Curtmola, and Justin Cappos. in-toto: Providing farm-to-table guarantees for bits and bytes. In *28th USENIX Security Symposium*, pages 1393–1410, 2019. URL <https://www.usenix.org/conference/usenixsecurity19/presentation/torres-arias>. Accessed 12 August 2026.
- [17] Sam M. Werner, Daniel Perez, Lewis Gudgeon, Aariah Klages-Mundt, Dominik Harz, and William J. Knottenbelt. SoK: Decentralized Finance (DeFi). In *Proceedings of the 4th ACM Conference on Advances in Financial Technologies*, 2022. doi: 10.1145/3558535.3559780. URL <https://doi.org/10.1145/3558535.3559780>.
- [18] Gavin Wood. Ethereum: A Secure Decentralised Generalised Transaction Ledger, 2014. URL <https://ethereum.github.io/yellowpaper/paper.pdf>. Ethereum Yellow Paper, living specification; accessed 12 August 2026.