



WCDPQ

Verifiable Velocity

Global liquidity, verifiable execution, and control that stays with the account holder

Spot · Perpetuals · Early-Stage Allocation

Protocol Design · Risk Architecture · Cryptoeconomic Model

WHITEPAPER · VERSION 1.0

Table of Contents



Abstract

1.0 The Retail Access Gap

2.0 System Overview

3.0 Cryptographic Assurance

4.0 Modular Topology

5.0 The Risk-Segregated Product Stack

6.0 Stablecoin Access and Settlement

7.0 AI Sentinel and Policy-Constrained Execution

8.0 Capital Democratization

9.0 WCDP: Protocol Utility and Sustainable Resource Economics

10.0 Parameter Governance

11.0 Core Architecture Team

12.0 Evolution Roadmap

13.0 Risk Factors, Disclosures and Appendices

Appendix: Cryptographic Proof System Specifications

Abstract

WCDPQ specifies an open protocol architecture for verifiable liquidity routing, high-performance settlement, and non-custodial asset custody verification, designed for mobile-first corridors. The design problem this document addresses is that a retail user has no practical way to verify the state of a venue holding their balance; the available options have been to read a report the venue commissioned or to withdraw.

The protocol answers with three mechanisms. Balances and movements are committed in a Merkle-sum liability structure with recursively aggregated zero-knowledge proofs published each epoch, so that an account holder can verify their own inclusion on a phone and an independent party can verify aggregate state without protocol-supplied software. Execution runs on a sharded matching core with deterministic sequencing and a unified clock, which makes replay possible and makes latency attributable to a subsystem rather than to a feeling. Spot, perpetual, and launch channels hold separate collateral behind ledger-enforced gates, so a failure in one cannot reach the others.

WCDP is the utility token through which network operational resources are allocated and validated, including verification staking, metered computation, and governance weighting. Network fee distributions are managed transparently via algorithmic protocol reserves. The protocol makes no representation regarding the market value or secondary liquidity of WCDP.

1.0 The Retail Access Gap

A retail participant in an emerging market today has more venues to choose from than at any point in the history of these markets, and fewer reasons to trust any of them with money they cannot afford to lose. The gap this document is organized around is not a shortage of venues. It is the distance between what a phone-based user can reach and what that screen actually lets them do, and how little of the middle they can inspect.

1.1 Three Frictions

Getting money in and out is the first obstacle, and for a user outside the major financial centers it is often the decisive one. Banking rails in much of Latin America clear quickly — a user there is accustomed to a transfer landing before they have put the phone down — but the route from a local account into a venue that prices in dollars or stablecoins is not a single hop. It crosses a payment provider, a settlement step, and a currency conversion, each with its own hours, limits, and failure modes. A venue that receives the funds by the next business day has not failed at anything in particular; it has simply inherited the pace of the corridor behind it, which the user experiences as their money being somewhere they cannot see.

The execution gap is narrower to describe and harder to close. What a professional desk has that a phone user does not is not primarily a faster connection. It is the machinery around the order: the ability to work an entry across time rather than taking the price on arrival, to set conditional exits that survive the trader going to sleep, to see how correlated their positions have become before the correlation matters, and to know where the liquidity in the book actually sits rather than where the top of it is displayed. None of these require special entitlement, and all of them are absent from the interfaces available to most retail users in these markets. The user absorbs the difference as slippage, as exits missed overnight, as a portfolio that turned out to be one position.

The third friction concerns what the user can establish on their own. A retail account holder has never had a practical way to answer basic questions about the venue holding their balance — whether the assets are where the venue says, whether their own balance is included in whatever figure is published, whether the number they see on the screen corresponds to a state that has been demonstrated rather than asserted. The answer available to them has historically been to read something the venue commissioned or to withdraw. Neither is a reasonable standard of evidence for someone checking a position at two in the morning, and the fact that institutional participants negotiate private reporting arrangements while retail has nothing is a difference in the quality of information each side acts on.

These three frictions interact. A user who cannot verify the venue will keep the balance small, which limits how much capital they will put behind a strategy that requires it. A user with no working-order tools will not hold an overnight position, which is where a directional view pays. The compound effect is a user who trades small, trades fast, and trades only what they can afford to lose — which is not a description of a population with a high risk appetite but of a population that has correctly priced the venue.



1.2 The Mobile-Native Trader

The user this system is built for does everything on a handset, and the handset is frequently on a network that drops. Their session starts in a messaging app and ends in a position, with nothing in between that resembles a research workflow. They are comfortable holding a stablecoin balance and uncomfortable leaving money in a local currency for a week. Derivative instruments are not unfamiliar to them; the second-most-common piece of trading advice in their communities concerns leverage rather than spot.

Two properties follow for the system. Requests must survive being repeated, because on a bad connection the user will press send more than once and the client will retry, and neither should produce two orders. And the interface must distinguish clearly between what is live and what is being remembered from earlier, since a price that stopped updating four seconds ago looks identical to a price that is current unless the design forces a visual difference.

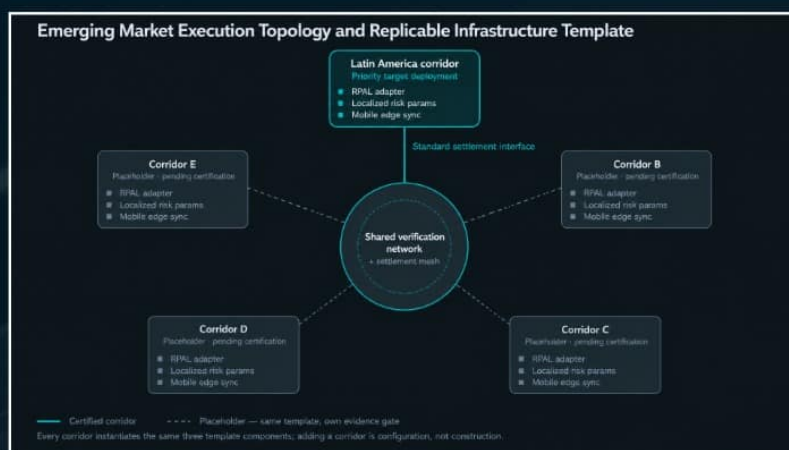
Latency expectations in this population are set by whatever messaging application they used five minutes earlier. That establishes a floor on responsiveness but not on reliability: a user will tolerate three hundred milliseconds and will not tolerate an order that hangs without resolution. The rest of this document weights reliability accordingly, which is why the degradation behavior in §4.3 reserves capacity for exits and why the client in §4.4 refuses to submit exposure-increasing orders without a server round trip.

1.3 Emerging Market Corridors as Execution Super-Nodes

High-velocity emerging market corridors serve as the operational benchmark for the protocol architecture.

Regional payment infrastructures in Latin America—characterized by instant gross settlement expectations and mobile-first retail density—provide the rigorous baseline conditions required to stress-test high-frequency cryptographic verification. Designing for environments with volatile domestic fiat dynamics and high throughput demands ensures that the underlying state-commitment mechanisms remain resilient under non-trivial network constraints.

What the protocol deploys is an infrastructure template: an adapter pattern for multi-rail connectivity, unified risk parameters, and mobile edge synchronization. The architecture is corridor-agnostic, treating local gateways as independent implementations of a standard settlement interface rather than bespoke deployments.



1.4 Design Thesis

The five invariants that follow in §2.1 are the operational content of the claim made here. In plain terms: state under the venue's control should be checkable by parties who do not work for the venue; the outcome of an order should be determined by published rules rather than by timing accidents in the software; a failure in one product should not be able to reach the collateral of another; the route between a local account and a trading balance should be built so that adding a corridor is configuration rather than construction; and the network's token should be required by real operational roles rather than offered as a loyalty instrument.

None of these is unusual as an engineering value. What is unusual is applying them where the counterparties are retail users on phones with small balances, and accepting the costs that follow — the disclosure obligations, the reduced capital efficiency of isolation, the performance left on the table by transparent proof systems, the corridor work that cannot be skipped by routing around a regulator. The rest of this document is largely a description of those costs and the mechanisms that incur them.

2.0 System Overview

This chapter situates the five layers and then states what the system is not built to defend against. The second half is the more useful one for anyone evaluating the first.

2.1 Protocol Invariants and Layered Architecture

Verifiable state. Balances, movements, and the correspondence between them are committed in a form an outside party can check. The proof system and the ledger supporting it are defined once in Ch.3.

Deterministic execution. Identical inputs produce identical orderings on every replay. Sequencing discipline, and the reasons the architecture is willing to pay for it in cross-shard latency, are in Ch.4.

Risk segregation. Spot, perpetuals, and launch participation hold separate collateral with unidirectional gates between them, enforced in the ledger. Ch.5 defines the boundaries; the isolation pool's funding is covered under §9.5.

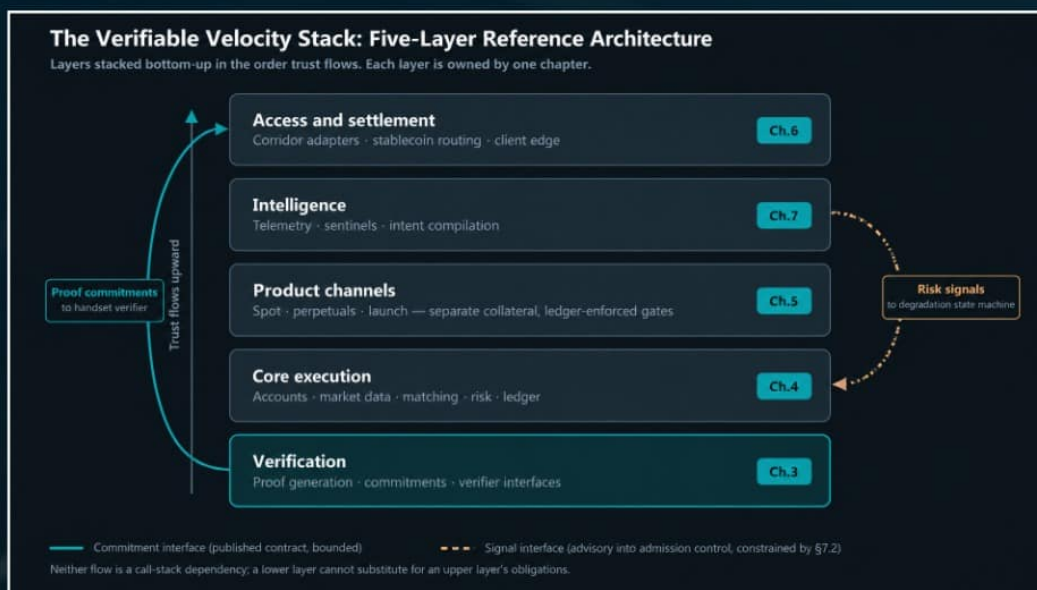
Settlement abstraction. Corridors connect through an adapter contract rather than through hardcoded integrations, which is what allows a corridor to be added without changing the platform. Ch.6.

Resource-bound token utility. WCDP is required for operational roles — validation, metered compute, allocation eligibility, governance weight — not offered as a discount. Ch.9.

The layers stack in the order the trust flows.

Layer	Function	Chapter
Verification	Proof generation, commitments, verifier interfaces	3
Core execution	Accounts, market data, matching, risk, ledger	4
Product channels	Spot, perpetuals, launch	5
Intelligence	Telemetry, sentinels, intent compilation	7
Access and settlement	Corridor adapters, stablecoin routing, client edge	6

Two cross-layer flows are worth naming because they are not obvious from the table. Risk signals from the intelligence layer travel upward into the degradation state machine described in §4.3 and into the liquidation path, which means an entity outside the matching core can influence admission decisions; the constraints on that influence are the subject of §7.2. Proof commitments travel downward from verification to the client edge, which is how a handset ends up doing verification work locally rather than reading a status page. Neither flow is a dependency in the ordinary call-stack sense; both are defined interfaces with published contracts, and each is bounded so that the lower layer cannot be leaned on as a substitute for the upper one's obligations.



2.2 Threat Model, Trust Assumptions and Non-Goals

The adversary this architecture is written against is an insider with legitimate access, not an outside attacker. External intrusion is a real risk and is treated by conventional means — key distribution, access separation, monitoring — but it is a solved category in the sense that the defenses are well understood. An operator who can move a balance, alter a fee, or suppress a report without leaving a trace detectable by someone else is the harder problem, and it is the one the verification layer addresses.

Three assumptions are made and not proven.

The first is that hash functions used in commitments retain collision resistance, and that the FRI parameters published in Appendix remain sound at the stated security level. A break here invalidates proofs rather than assets, and the migration path is a governance decision that cannot be pre-committed in detail.

The second is that external inputs are what they claim. Composite mark prices, payment confirmations, and reference data enter as attested values from named sources. The system screens them, bounds their influence, and logs their use. It does not independently establish that a confirmation from a partner gateway corresponds to settled funds in the partner's own books. Correct execution on a wrong input yields a correctly proven wrong outcome.

The third is that at least some independent parties will actually verify. Every cryptographic claim in this document is a claim about what a motivated actor *can* check at low cost. If nobody checks, the commitments are decoration. The protocol's mitigation is to make verification trivially cheap and to publish failures loudly, which raises the cost of unnoticed misbehavior without eliminating it.

What the system is not trying to do is as delineated as the list above, and shorter than a reader might expect from a document of this length.

It is not attempting to eliminate market risk, and no mechanism here changes the fact that a leveraged position can lose more than the account holds. It does not attempt to remove the need for legal entities, licenses, or jurisdiction-specific product decisions; the corporate position described in §13.2 exists precisely because cryptographic verification addresses solvency claims and not regulatory standing. It does not offer a general-purpose smart contract platform, a public chain, or a settlement layer for third-party applications. And it does not claim to be decentralized in the sense that term is used in governance literature — parameter control is a bounded, published scope, which Ch.10 specifies, and calling that community governance would be a misdescription in the flattering direction.

3.0 Cryptographic Assurance

Anyone holding a balance on a venue that custodies assets has historically had two ways to form a judgment about that venue: read a report written by a firm the venue pays, or withdraw. Neither is available to a retail user checking a position on a phone at two in the morning. This chapter describes the third option the protocol builds, and it spends comparable effort on where that option stops reaching.

3.1 What Can Be Verified

The verification layer makes three distinct claims, and they should not be confused with one another because they fail in different ways.

The first is **reserve coverage**: at each supply epoch ε , the assets the venue controls and has placed inside the proven perimeter exceed the liabilities the venue has enumerated. The second is **liability inclusion**: a specific account holder's balance is a leaf of the committed liability set, with a value the account holder can check without trusting anyone. The third is **state-transition validity**: the ledger moved from commitment C_ε to $C_{\varepsilon+1}$ by applying the published rule set, and no other transition could have produced that pair.

Coverage, inclusion, and transition validity are each verifiable by an independent party. None of them is a statement about the enterprise as a whole, and §3.5 draws that line precisely.

A signed attestation and a validity proof have different failure modes, which is the practical reason the protocol builds for the latter. An attestation records what a named professional concluded about a snapshot, and its residual risks live in sampling, engagement scope, and the recurrence interval. A proof is a machine-checkable object whose falsity is itself evidence. Frequency matters here. Solvency checked quarterly, or annually, is a statement about a state of the world that has already elapsed; solvency checked every epoch is a statement about the current one, and the gap between what is claimed and what is true stays short enough for a user to act on.

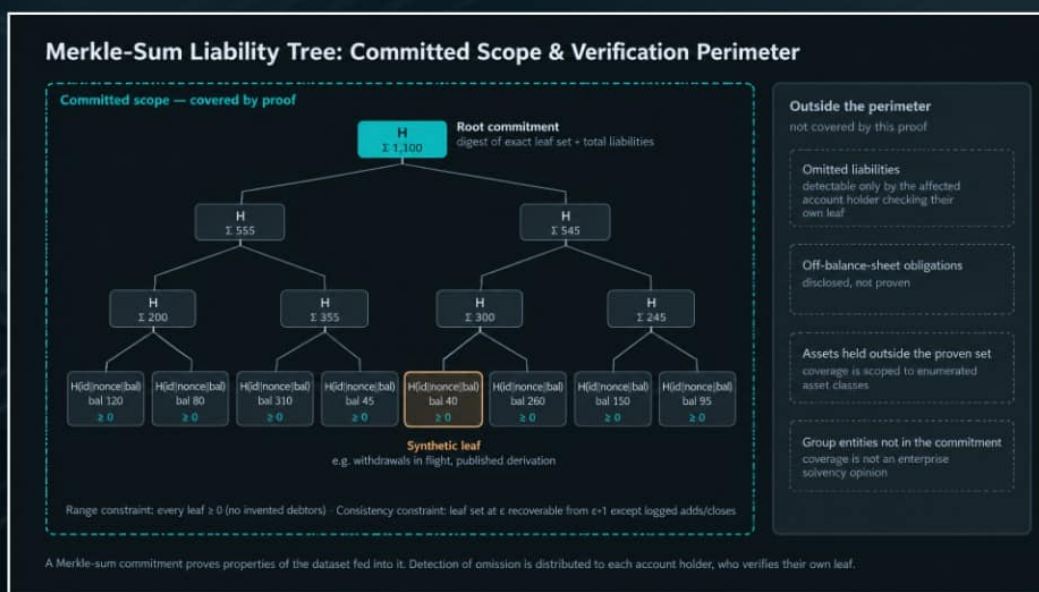
Code is Law is used in this document as a design constraint rather than a slogan. It means discretionary powers are enumerated, circuit-encoded where possible, and bounded where not. Where a rule is policy rather than arithmetic — which counterparties may onboard, how a disputed launch allocation is resolved — the protocol's commitment device is publication plus detection: the rule is written down, its invocation is logged, and any deviation from the written rule is observable by a party that was not supposed to be involved. Code constrains what can be done quietly. It does not constrain what can be done openly and badly, and a reader should keep that distinction in view through the rest of this chapter.

3.2 Merkle-Sum Liability Commitments

The liability set is committed as a Merkle-sum tree. Each leaf holds a blinded commitment $H(\text{account_id} \parallel \text{nonce} \parallel \text{balance})$ so that the tree does not publish an addressable list of customers and balances. Each internal node carries both a hash and the arithmetic sum of the balances beneath it. The root therefore publishes two things at once: a cryptographic digest binding the exact leaf set, and a total liability figure that any verifier can read off the commitment itself. A venue cannot revise the total without changing the root, and it cannot change the root without leaving evidence of the revision in the log.

Two families of constraint close the obvious attacks. Range constraints on every leaf exclude negative balances, which would otherwise let the sum be driven down by inventing a debtor. Consistency constraints between consecutive roots ensure that the previous epoch's leaf set is recoverable from the current one except for entries that have been explicitly added or closed, with those operations recorded. The residual attack is omission, and it deserves direct treatment because published implementations of this idea frequently gloss it.

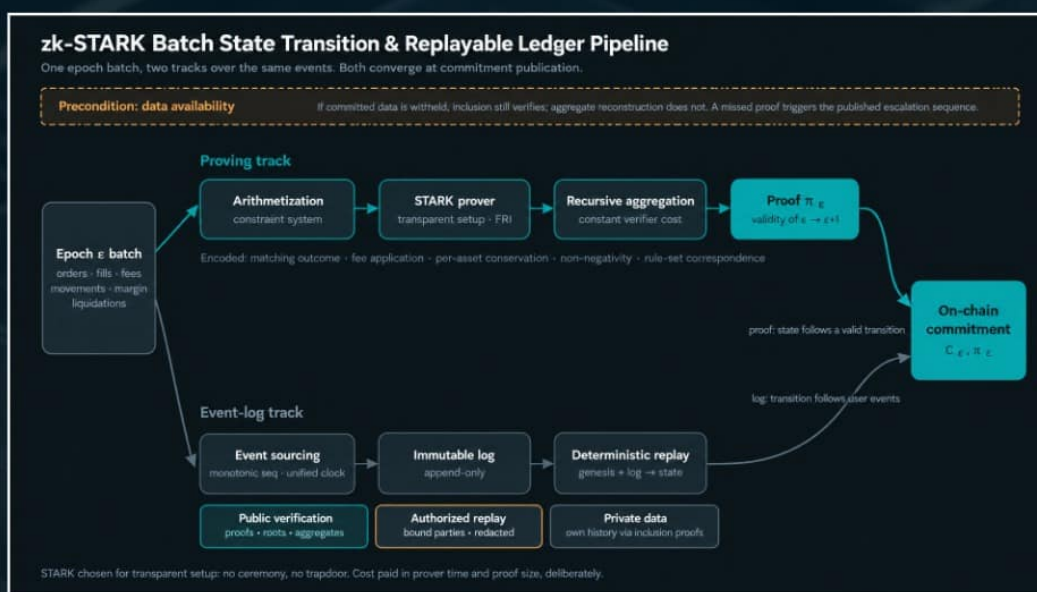
A Merkle-sum commitment is a proof about the dataset that was fed into it. If a venue simply leaves a liability out of the leaf set, the arithmetic is internally consistent, every inclusion proof verifies, and the root says nothing about the missing entry. Claims that "hidden liabilities break the commitment" are wrong, and the protocol's documentation should not make them. What the design does instead is distribute the detection burden: each account holder checks their own leaf against the committed root, so an omission is detectable by the one party with the strongest incentive to notice it. Liabilities that cannot be attributed to a named account — accrued but unsettled amounts, withdrawals in flight, and similar obligations — are included as explicitly labeled synthetic leaves with published derivation rules, or they are disclosed as outside the proven perimeter. Appendix specifies the encoding.



3.3 zk-STARK Proof-of-Solvency and the Replayable Ledger

Each epoch's transitions are arithmetized into a constraint system. The constraints encode the matching outcome for the batch, fee application, per-asset conservation, non-negativity of every affected balance, and the correspondence between the published rule set and the transition actually applied. A prover generates a proof π_e that the batch was executed correctly and that the resulting state hashes to the committed roots on both sides of the epoch. Proofs are recursively aggregated, so the verification cost a third party pays stays roughly constant as batch volume grows — the property that makes continuous verification viable rather than a monthly exercise.

The protocol uses STARKs, and the reason is not performance. Transparent setup means there is no ceremony, no participant who could have retained a trapdoor, and no trust assumption about a multi-party computation that nobody outside the room can audit. The price is paid in prover time and proof size, both of which are falling with better arithmetization and faster hashing, and both of which the protocol spends deliberately. A SNARK would give smaller proofs and a shorter verification tail, at the cost of an assumption that is invisible to the user verifying the result. Given that the entire argument of this chapter rests on what a third party can check unaided, that trade is taken in the direction of verifiability.

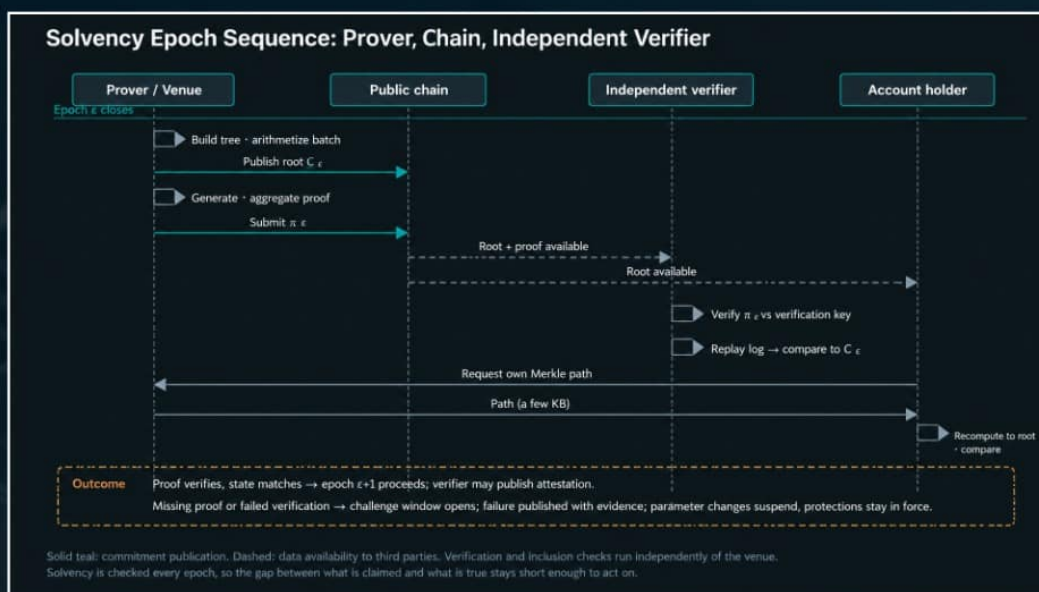


Beneath the proofs sits an event-sourced ledger. Every state change is an immutable event with a monotonic sequence number and a timestamp drawn from the unified clock of §4.2. Given the genesis state and the event log, the ledger state is reconstructible by deterministic replay. This is what makes a proof meaningful: a proof shows that the committed state follows from some valid transition, and the log shows that the transition follows from the events users actually generated.

Replay access is tiered, and the tiers are a deliberate concession rather than an oversight.

- **Public verification.** Anyone can check proof validity, read the commitment roots, and read aggregate statistics published alongside each epoch. No account-level data is exposed.
- **Authorized replay.** A party bound by an access agreement — a supervisory authority, a designated independent verifier, a settlement partner auditing a specific corridor — can reconstruct full state from the log, subject to redaction of unrelated accounts.
- **Private data.** Account holders see their own history through inclusion proofs and the SDK. Nobody else does.

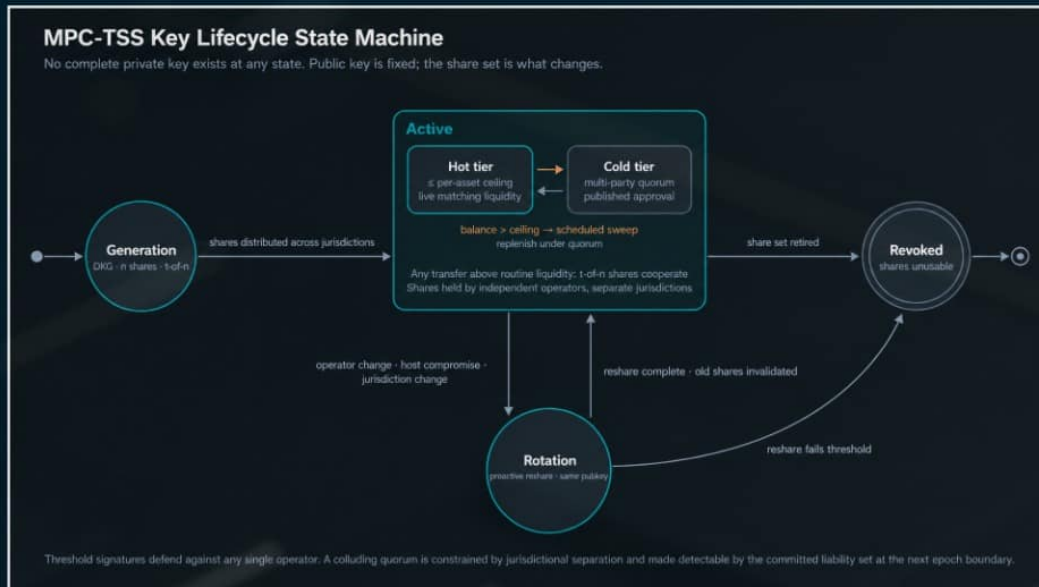
Data availability is a precondition for the first two. If committed data is withheld, an account holder can still verify their own inclusion against the published root, but third parties cannot reconstruct aggregate behavior, and the protocol's response is defined rather than improvised: a missed proof or a failed verification at an epoch boundary triggers a published escalation sequence — parameter changes suspend, existing protections remain in force, and the failure is announced with the evidence that produced it. Verification has value only if its failure is consequential.



3.4 MPC-TSS Custody and Key Lifecycle

Signing authority is distributed with threshold signature schemes. No complete private key is assembled at any point; shares are held by independent operators in separate jurisdictions, and any transfer above routine liquidity requires t-of-n shares to cooperate. The public key is fixed while the share set can be reshared proactively, which allows operators to be rotated — a departing custodian, a compromised host, a change of jurisdiction — without changing the on-chain address anyone points at.

The operational envelope is a policy layer on top of the cryptography. A hot tier holds only the liquidity that live matching and typical withdrawal flow require, with the ceiling set per asset and adjusted by governance within bounds. Balances above the ceiling sweep to cold storage on a schedule, and cold transfers require shares gathered under multi-party authorization with the approval quorum published. Key material moves through a defined lifecycle rather than sitting in one state indefinitely.

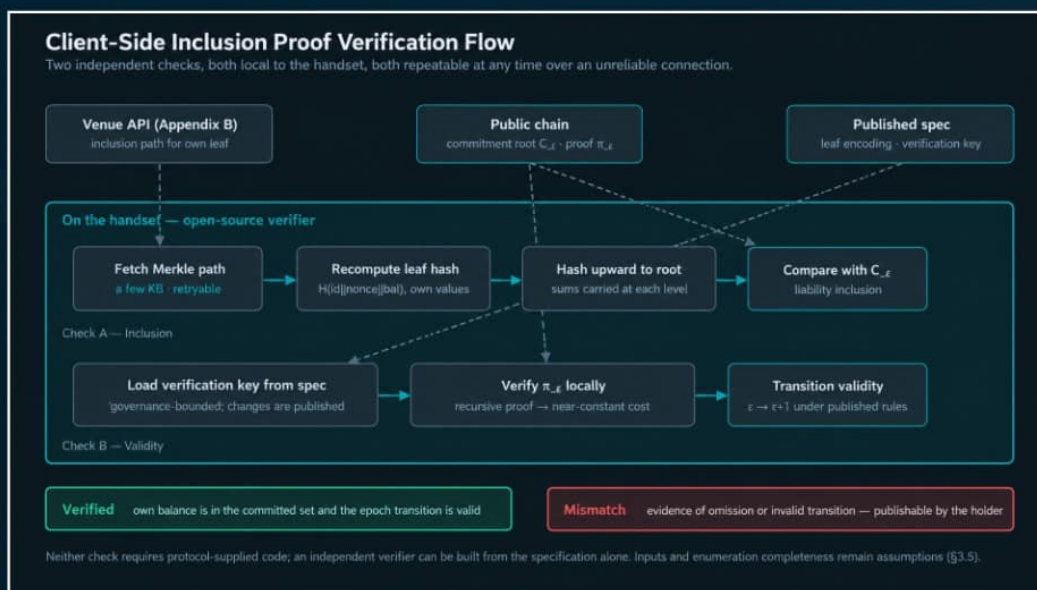


Threshold signatures protect against any single operator. They do not protect against a quorum that decides to act together, and describing the scheme as if collusion were impossible would misrepresent it. The relevant defenses at that layer are jurisdictional separation of share holders, published quorum rules that make a quietly assembled coalition harder to arrange, and the fact that the committed liability set constrains what a colluding quorum could withdraw without leaving a visible inconsistency at the next epoch boundary. That last property is a detection guarantee, not a prevention guarantee. Assets can move; the movement cannot remain unrecorded.

3.5 Client-Side Verification and the Boundaries of Assurance

An account holder verifying their own position needs a Merkle path and a commitment root — a few kilobytes, not a chain download. The mobile verifier fetches the path, recomputes the hash upward to the root using the same leaf encoding the protocol publishes, and compares against the root in the current commitment. Separately, it checks that the epoch's proof verifies against the published verification key. Both operations run locally on a phone, over an unreliable connection, and either can be repeated at any time.

Verifier implementations, the CLI, and the SDK interfaces are open source.



What the layer does not establish matters as much as what it does.

- **Coverage is scoped.** A reserve-coverage proof speaks to the asset classes inside the perimeter. Assets held elsewhere, off-balance-sheet obligations, and any entity in the group not included in the commitment are outside it. Coverage is not a solvency opinion about the enterprise.
- **Inclusion depends on enumeration.** The protocol can prove that a claimed liability sits in the tree. It cannot prove that the enumeration rule used to build the tree was complete; the defense is that each affected party can check their own leaf.
- **Validity is relative to the rules.** A transition that followed published rules is validly proven even if the rule itself is unattractive to users. The proof answers whether the protocol did what it said, and governance — Ch.10 — answers whether what it said should change.
- **Inputs are assumed, not proven.** Mark price feeds, payment gateway confirmations, and external reference data enter the system as attested inputs with named sources. Correct behavior on a wrong input produces a correctly proven bad outcome.
- **Cryptography has assumptions.** Soundness rests on hash-function collision resistance and the FRI parameters the protocol publishes. A break in those assumptions breaks the guarantee, and the migration path for such an event is a governance matter the protocol cannot pre-commit to in detail.

Verification changes which failures can be detected, by whom, and how quickly. That is the whole of the claim.

4.0 Modular Topology

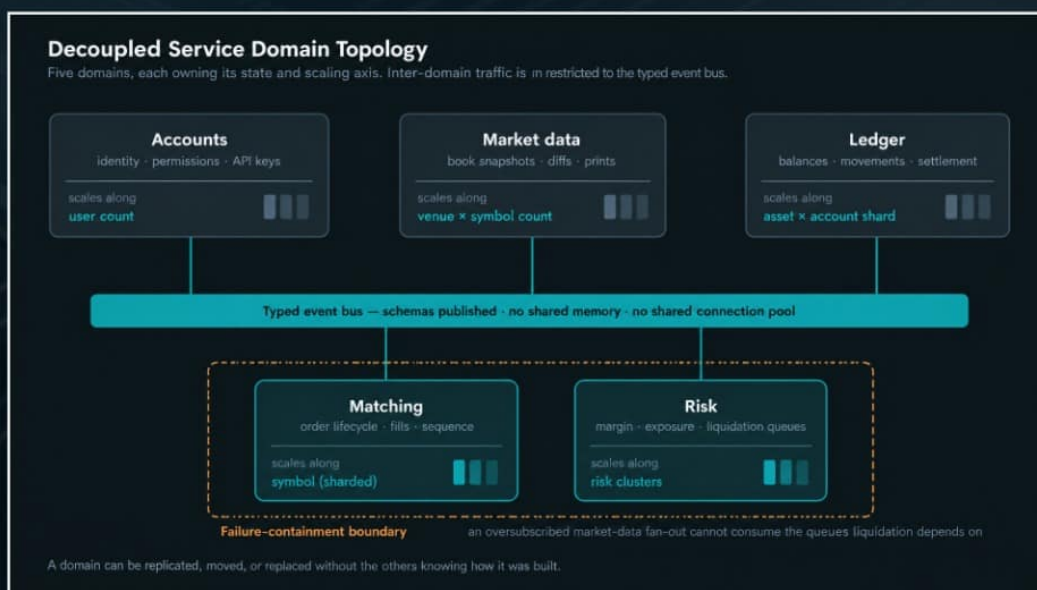
A venue that promises sub-second execution and then loses its liquidation engine during the one hour when liquidation matters has not delivered sub-second execution. The architecture in this chapter is organized around which functions must survive peak load and which are allowed to degrade, because a system that treats all traffic as equally important will, under stress, protect whatever arrives first.

4.1 Decoupled Service Domains and Deterministic Sequencing

The core is split into five domains, each owning its state and each with its own scaling axis.

Domain	State owned	Scales along
Accounts	Identity, permissions, credential state, API key surface	User count
Market data	Order book snapshots, incremental diffs, trade prints	Venue and symbol count
Matching	Order lifecycle, fills, sequence assignment	Symbol (sharded)
Risk	Margin state, exposure, liquidation queues	Risk clusters
Ledger	Balances, movements, settlement records	Asset and account shard

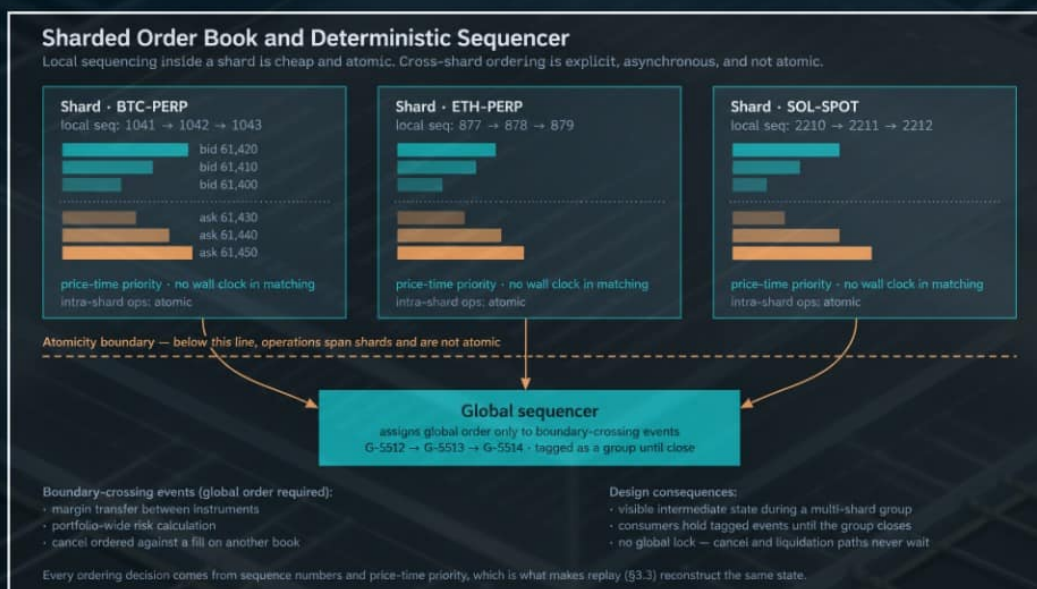
Communication runs over an event bus with typed schemas. A domain can be replicated, moved, or replaced without the others knowing how it was built, and — the part that matters operationally — a fault in one does not propagate through shared memory or a shared connection pool. An oversubscribed market-data fan-out degrades market-data latency. It cannot, by construction, consume the queues that liquidation depends on.



Order books are sharded by instrument. Each shard maintains its own price levels and assigns monotonic sequence numbers to the events it produces. A central sequencer assigns a global order only to events that cross shard boundaries — a transfer of margin between instruments, a portfolio-wide risk calculation, a cancellation that must be ordered against a fill on another book. Within a shard, sequencing is local and cheap. Across shards, it is explicit and asynchronous.

That choice imposes a boundary worth stating plainly: operations confined to one shard are atomic, and operations spanning shards are not. A cross-margin adjustment that touches three books will produce a visible intermediate state on the bus, and a client reading mid-flight can observe a portfolio that is momentarily inconsistent. The protocol's answer is to make the window short, to tag multi-shard events so consumers can hold them until the group closes, and to refuse any design that would require a global lock across books to appear atomic to a reader. Global locking restores the coupling the sharding exists to remove, and the cost lands precisely on the cancellation and liquidation paths that must never wait on anything.

Determinism is the reason this structure is compatible with the replayable ledger of §3.3. A proof attests that state moved from one commitment to the next under published rules. If two matching engines processing identical inputs could produce different orderings — through wall-clock comparison, through iteration over an unordered map, through a tie broken by thread scheduling — then replay would reconstruct a different state and the proof would be evidence of nothing. Every ordering decision in the matching path is therefore taken from the sequencer's sequence numbers and price-time priority, and the engines themselves carry no clocks that affect outcomes.

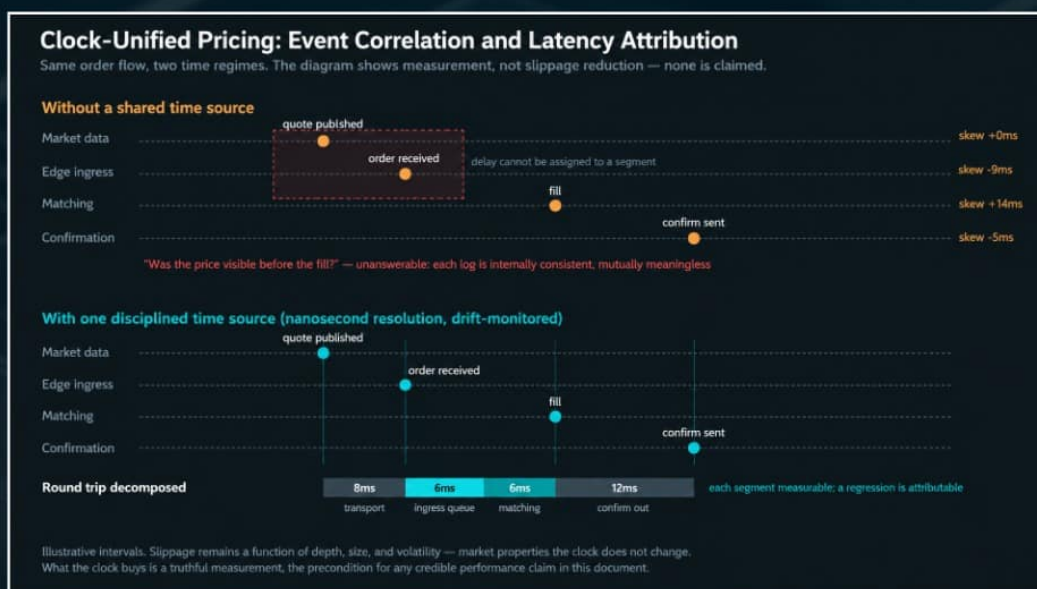


4.2 Clock-Unified Pricing and Latency Attribution

Every event entering the system carries a timestamp from a single time source, exposed to clients at nanosecond resolution. The source is disciplined against multiple external references with drift monitoring, and a divergence beyond the published tolerance raises an operational alarm rather than being silently absorbed.

The practical effect is that quotes and fills become commensurable. When a user's limit order executes at a price they believe they did not see, the question "was the price visible before the fill?" has a definite answer, because the quote publication timestamp and the fill timestamp come from the same clock and sit in the same ordered log. Without a common clock that question is unanswerable — each subsystem's timestamps are internally consistent and mutually meaningless, and the dispute reduces to which log the venue chooses to present.

Latency becomes attributable for the same reason. The interval between quote publication and fill confirmation decomposes into segments that can be measured separately: exchange-to-edge transport, edge-to-matching ingest queueing, matching computation, and outbound confirmation. Operators can see which segment moved when a client reports degradation, and clients with colocated infrastructure can measure their own side of the boundary against the same reference.



A single clock does not make execution cheap. Slippage is a function of book depth, order size relative to available liquidity, and volatility over the interval between decision and arrival. Those are market properties, and a venue that advertises a timestamp discipline as though it were a price guarantee is selling something it does not control. What the clock buys is a truthful measurement, which is the precondition for the user-side latency budgeting tools and for any credible performance claim made anywhere in this document.

4.3 Graceful Degradation Under Peak Load

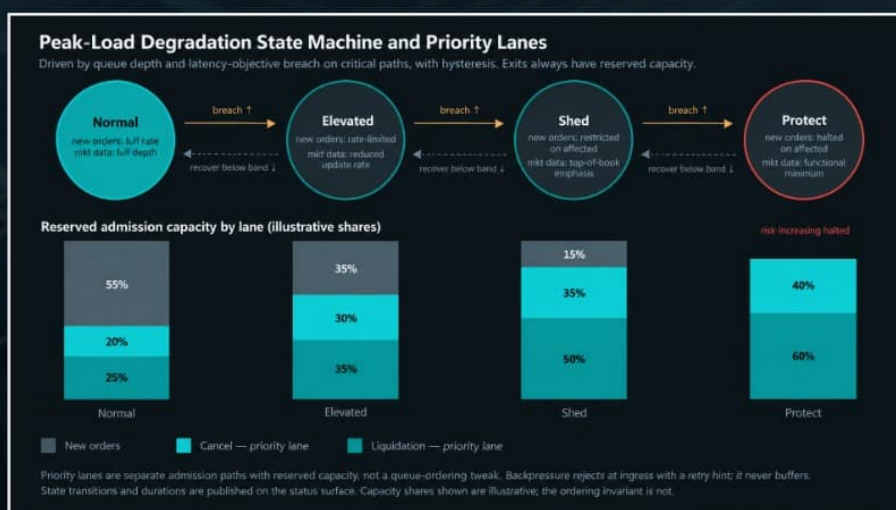
Load handling is defined as a state machine with four levels, driven by queue depth and breach of latency objectives on the critical paths, subject to hysteresis so that the system does not oscillate at a threshold.

State	Admission of new orders	Cancel	Liquidation	Market data
Normal	Full rate	Priority	Priority	Full depth
Elevated	Rate-limited by venue	Priority	Priority	Reduced update rate
Shed	Restricted on affected instruments	Priority	Priority	Top-of-book emphasis
Protect	Halted on affected instruments	Priority	Priority	Functional minimum

Priority lanes are not a queue ordering tweak. They are separate admission paths with reserved capacity, so that a flood of new orders cannot occupy the capacity a cancellation needs, and a flood of cancellations cannot delay a liquidation that other accounts' solvency depends on. Under the Protect state, the venue stops accepting risk-increasing orders on the affected instruments and keeps accepting risk-reducing ones — cancels, closes, and liquidations. A trader who cannot exit a position because the venue is busy is bearing a risk they did not agree to.

Backpressure is applied at the ingress edge rather than inside the matching path. Clients that exceed their published rate limits receive explicit rejection with a retry hint rather than being silently queued, because an order sitting in an unbounded queue is an order of unknown age, and its timestamp on arrival would misrepresent when the user made the decision. Queue depth is bounded by configuration and monitored as a first-class metric; when the bound is reached, admission control rejects rather than buffers.

State transitions and their durations are published on the status surface. A venue that quietly enters a degraded state and quietly leaves it leaves its users to infer what happened from outcomes, which is the pattern this architecture is intended to eliminate.



4.4 Mobile Edge State Synchronization

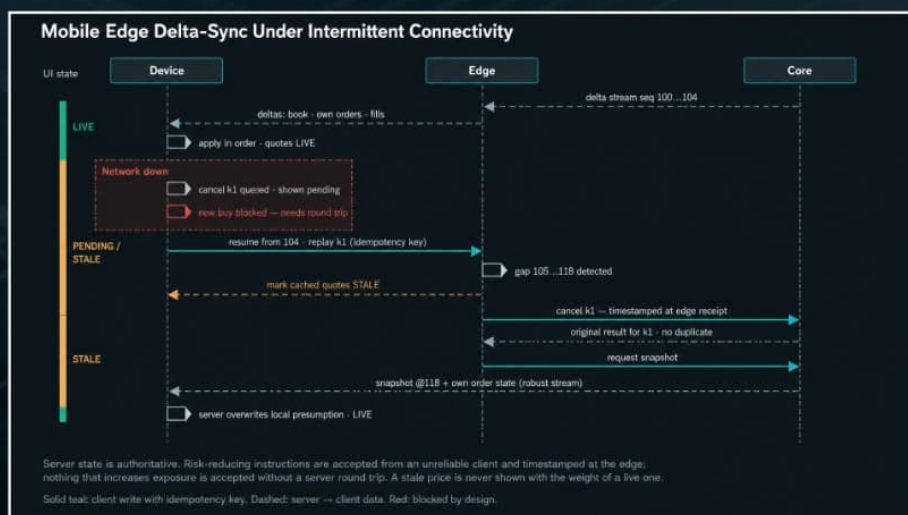
Connections from handsets are slow, intermittent, and frequently switched between networks mid-session. The synchronization model assumes all three rather than treating them as exceptions.

Clients hold a delta stream of order book updates, per-account order state, and fills, keyed by a sequence number. Each event applies to state the client already has, and any gap in the sequence invalidates the local view rather than being papered over with a patch. A client that detects a gap requests a snapshot, marks locally cached quotes as stale in the interface until the snapshot arrives, and continues to display its own order state, which is served from a separate and more robust stream. Showing a stale price with the same visual weight as a live one is the single most common way a mobile trading interface misleads its user, and the client is built to make staleness visible.

Writes use idempotency keys. A submit or cancel carries a client-generated key, and a retry across a network switch reuses it; the server returns the original result rather than creating a duplicate order. This matters most in exactly the condition that generates retries, which is a user on a weak connection watching an order they cannot confirm.

The server is authoritative on every question of state. Local optimistic display of a pending order is permitted — the interface can show an order the server has not yet acknowledged, marked as pending — but the client never treats its own state as the record. On reconnect, the client replays its pending intents against the server's actual state and discards any local presumption that the server contradicts.

Accepted from an unreliable client: risk-reducing instructions. An offline or edge-queued cancel is timestamped on receipt at the edge, not on the device, and the user sees which reference applies. Nothing that increases exposure is accepted without a server round trip, and the protocol publishes that restriction, because a market order whose price is determined on the phone is a market order whose price the user's clock and connection determined rather than the book's.



5.0 The Risk-Segregated Product Stack

Three businesses share one venue, and the failure that matters is contamination: a launch that goes badly should not reach into the trading accounts of users who never participated in it, and a perp position that gaps through its liquidation price should not touch a spot balance the user holds for other reasons. The separation is structural, implemented in the ledger rather than in policy, and it costs users something that this section states as plainly as the benefit.

5.1 Three-Channel Isolation and Capital Gates

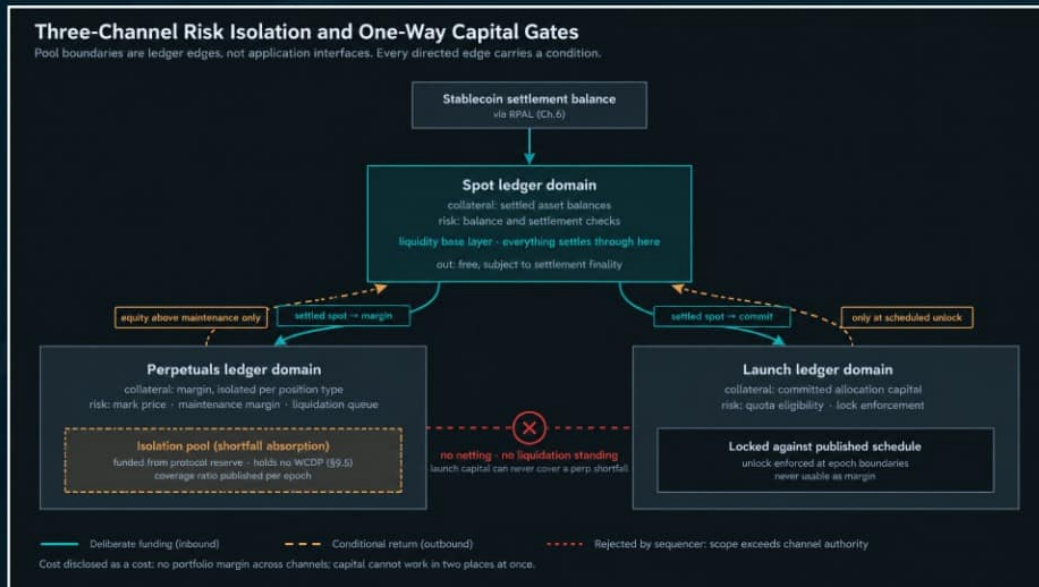
Each channel holds its own ledger domain, its own collateral pool, and its own risk computation.

Channel	Collateral	Risk engine	Cross-channel transfer
Spot	Settled asset balances	Balance and settlement checks	Free out, subject to settlement finality
Perpetuals	Margin balance, isolated per position type	Mark price, maintenance margin, liquidation queue	Inbound only from settled spot; outbound subject to maintenance requirement
Launch	Committed allocation capital	Quota eligibility and lock enforcement	Inbound from settled spot; outbound only at unlock

Gates are one-directional and condition-bearing. Spot collateral may be moved into the perpetuals account, because that is the user funding a position deliberately. Perpetual equity may be moved back out only above the maintenance requirement, and never as a mechanism to cover a liquidation shortfall. Launch commitments can return to spot at the end of their lock, and cannot be used as margin at any point — the protocol will not let a locked allocation become collateral for a leveraged position, since that converts a long-dated commitment into an instrument that can be forcibly closed at the worst moment.

The costs are real and worth naming. Capital sitting in the launch channel does not earn anything in the spot or perp channel while it waits; a user who wants the same capital working in two places cannot have it, by design. Portfolio margin across all three is off the table, which reduces capital efficiency relative to a fully fungible design and removes an entire class of correlation-driven blow-ups along with it. The protocol takes that trade and discloses it as a trade.

Isolation is enforced in the ledger, not in the matching engine's risk checks. A perp liquidation instruction cannot reference a spot balance because the instruction's asset scope does not include the spot shard, and the sequencer will reject a cross-shard event whose declared scope exceeds the originating channel's authority. Enforcement at the ledger level survives bugs in the risk engine, misconfiguration during a release, and a compromised service account. Enforcement in application code survives none of those.



5.2 Perpetuals: Mark Price, Liquidation Waterfall, and Insurance Isolation

Contract settlement runs against a mark price constructed from a composite of external reference venues, weighted by volume and filtered before use.

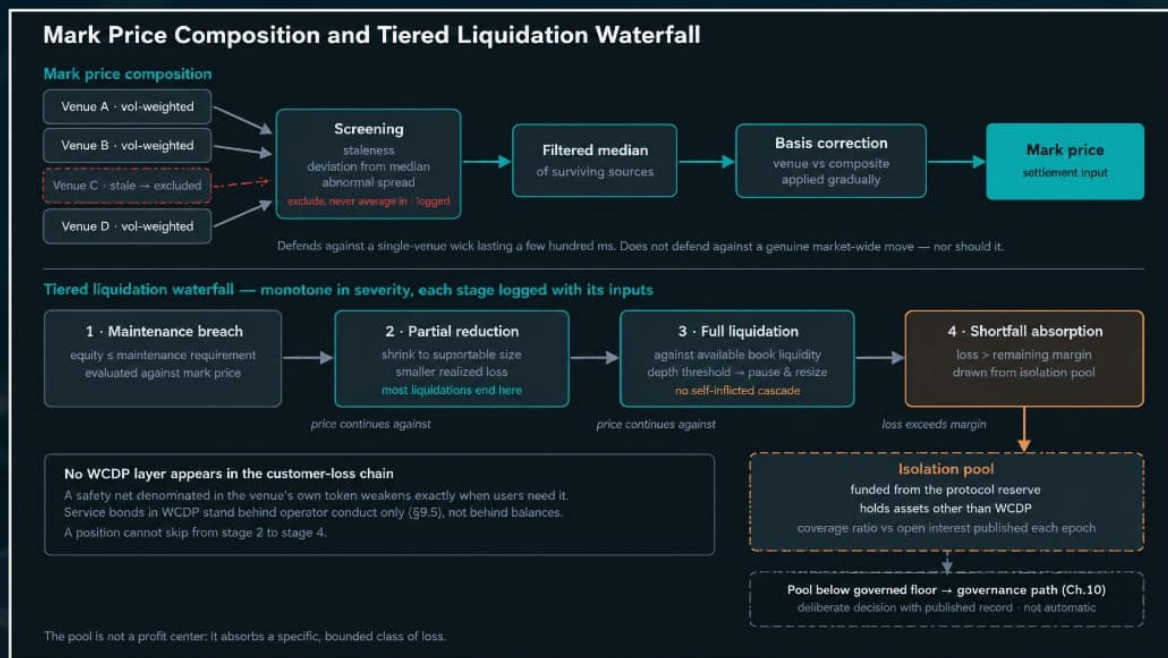
Sources are screened for staleness, deviation from the composite median, and abnormal spread relative to their own recent history. A source outside the tolerance band is excluded for the interval rather than averaged in, and the exclusion is logged with the reason. The filtered median is then corrected for the venue's own basis against the composite, applied gradually so that a persistent funding imbalance moves the mark rather than a single order moving it. The result is a price that responds to real market movement and is resistant to a print that exists only on one venue for a few hundred milliseconds. This is the whole defense against the liquidation-triggering wick, and it is a defense against manipulation that is cheap to attempt; it is not a defense against a genuine market-wide move, which will liquidate positions and should.

When a position's equity falls to the maintenance requirement, liquidation proceeds in stages rather than as a single event.

Partial reduction first. The engine reduces the position to the size the account can support at current margin, typically with a smaller realized loss than closing the whole position. Most liquidations end here, and users often do not notice beyond a position that got smaller overnight.

Full liquidation follows if the price continues against the reduced position, executed against the book at whatever liquidity exists. Here the engine will not keep pushing into a thin book: if the order would consume beyond a depth threshold, matching pauses while the sizing recomputes, which avoids the self-inflicted cascade where a liquidation order sweeps the book and triggers liquidations on the positions that provided the liquidity.

Shortfall absorption last. Where the realized loss on a full liquidation exceeds the account's remaining margin, the difference is drawn from the isolation pool. That pool is funded from the protocol reserve, and it holds assets other than WCDP — the reasoning for which is set out under §9.5. When the pool is depleted below its governed floor, the protocol's response is a governed parameter change rather than an automatic mechanism, because the alternatives available to a venue at that point — socializing losses across profitable traders, or reducing counterparty positions — are consequential enough to require a deliberate decision with a published record.



Two properties of this design deserve the attention usually given to the mark price. The liquidation sequence is monotone in severity — a position cannot skip from partial reduction to shortfall absorption, and each stage is logged with the inputs that triggered it. And the pool is not a profit center. It exists to absorb a specific, bounded class of loss; its coverage ratio is published each epoch, and a reader can compare it against outstanding open interest without taking the venue's word for adequacy.

5.3 Spot Base Layer and the Launch Firewall

Spot is the simplest channel and the one everything else settles through. Depth and spread are published per instrument, measured at several notional sizes rather than only at the top of book, since a venue with an impressive best bid and no liquidity behind it is a venue that will show users a price they cannot get. Settlement finality is explicit per asset class: what the user sees as an available balance is final, not provisional, and the difference between the two is never left to be inferred from a later correction.

The launch channel inherits none of the spot account's risk characteristics. Quota is allocated through the protocol in Ch.8, capital is committed into the launch domain, and the commitment is locked against the published schedule. Unlock is enforced by the ledger at the scheduled epoch boundaries. A project's failure to deliver, a delay in listing, or a collapse in the token's market price does not reach the user's spot balances or their margin account, and — the half of the firewall that matters more — a user's losses in the perpetuals channel cannot be satisfied from capital committed to a launch. Positions in the two domains are never netted, and no liquidation process has standing to look across the boundary.

What the firewall does not do is assess anything. It keeps a bad outcome contained to the participants who accepted it; Ch.8 addresses the disclosure and allocation machinery that determines who those participants are and on what terms.

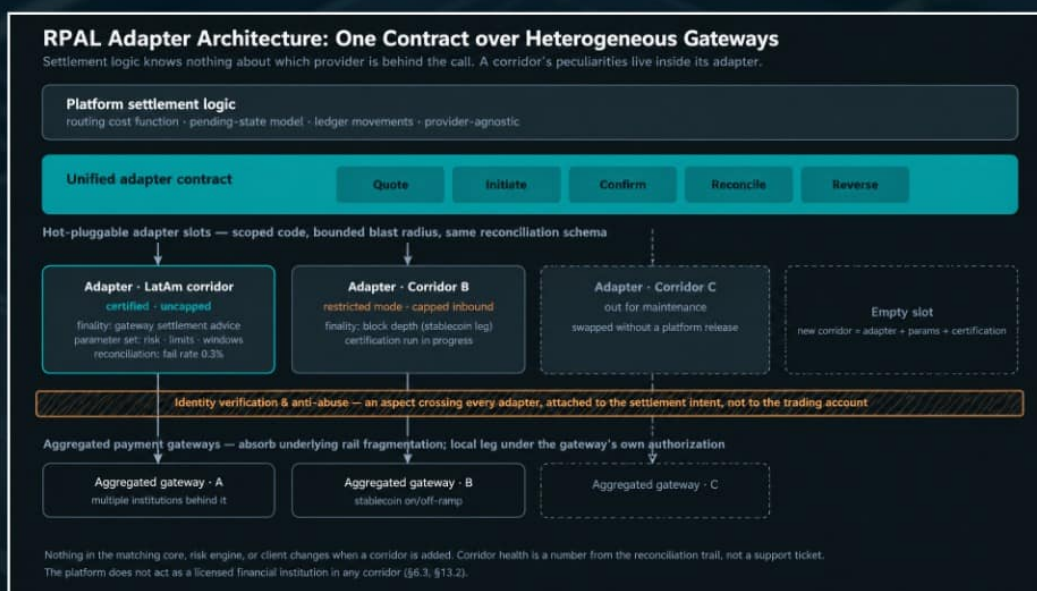
6.0 Stablecoin Access and Settlement

Money arriving from a Brazilian bank account and money arriving from a Mexican one should meet the same internal contract. The regional payment abstraction layer exists to make that true, and it exists as architecture rather than as a list of partners because the list changes and the contract does not.

6.1 The Adapter Model

Every corridor integration implements one interface: request a quote for a conversion or transfer, initiate it, confirm the outcome, reconcile against the corridor's own record, and reverse it when it fails. Above that interface sits the platform's settlement logic, which knows nothing about which provider is behind the call. Below it sits an adapter, and an adapter is a scoped piece of code with a bounded blast radius — a corridor's peculiarities live inside it and cannot leak into the routing engine.

The platform connects to leading aggregated payment gateway solutions in Latin America, which is the level of specificity the interface requires. Aggregate gateways are the right abstraction for this layer because they already absorb the fragmentation of the underlying rails: a user in one state may settle through a different institution than a user in another, and the platform has no reason to distinguish. What it needs is a stable contract, published finality semantics, and a reconciliation trail.



Two design decisions in this layer matter more than the interface shape. Adapters are hot-pluggable, so a corridor can be taken out of service for maintenance or a provider replaced without a platform release; and every adapter must produce records in the same reconciliation schema, which is what makes a corridor's health measurable. A corridor that fails reconciliation more often than its peers is visible as a number rather than as a support ticket, and it can be throttled independently.

Adding a new corridor, then, is an adapter, a parameter set, and a certification run against the reconciliation suite. Nothing in the matching core, the risk engine, or the client changes. That property is what the term super-node in §1.3 refers to at the technical level: the deployment pattern is the same each time, which is why a second corridor is a project of weeks rather than a rebuild.

6.2 Stablecoin Routing and Settlement Orchestration

The corridor between local currency and a trading balance has four stages, and the platform's settlement logic covers all four rather than only the first.

Onboarding funds. A user expresses an intent in local currency; the adapter quotes the rate, fees, and expected settlement window; the user accepts; the gateway confirms. The platform records the intent, the quote, and the confirmation as separate events, because a dispute about rate is a dispute about which of those three an intermediary altered.

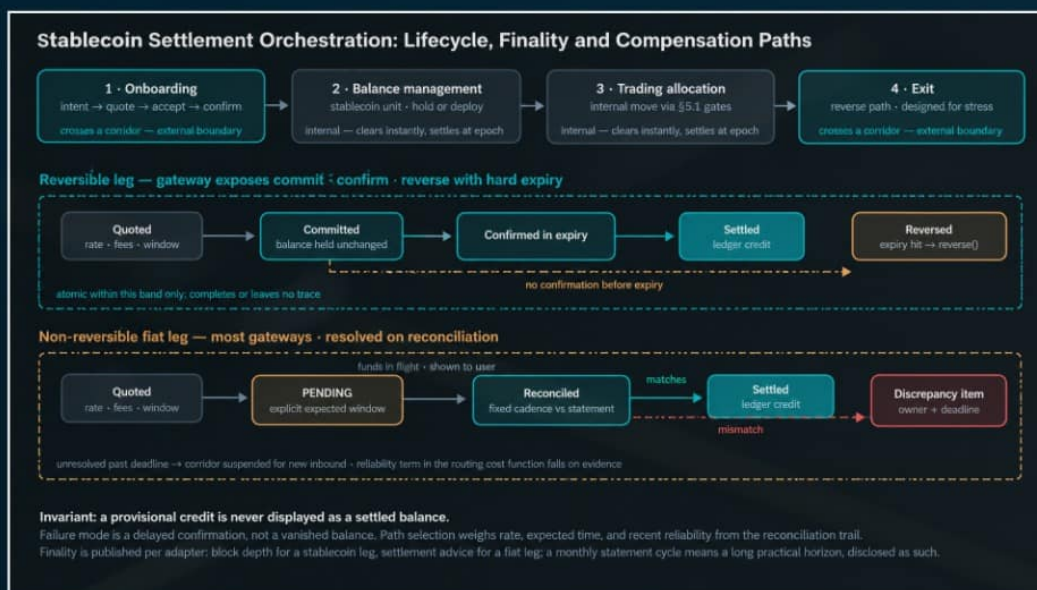
Balance management. Stablecoin balances sit in the user's account, denominated in the stablecoin unit, available for withdrawal to any corridor the platform supports or for deployment into trading. Users who hold a balance without trading are a legitimate state, not an abandoned conversion.

Trading capital allocation. Transfers from the settlement balance into the perpetuals or launch channel are internal ledger movements governed by the gates in §5.1. They do not touch a corridor, clear instantly, and settle at the epoch boundary like any other internal movement.

Exit. Withdrawal runs the routing path in reverse, with the same quoting and confirmation structure. Exit deserves more attention than entry in the design, because it is the operation users attempt under stress, and it is the one where a venue's queue behavior becomes visible.

Path selection among multiple gateways optimizes a cost function combining rate, expected settlement time, and recent reliability measured from the reconciliation trail. The reliability term is not decoration: a gateway offering the best headline rate while failing one request in fifty is more expensive than a slightly worse quote that completes, and the routing engine is built to notice the difference without an operator intervening.

Settlement across the corridor is atomic only in a defined range. Where a gateway exposes a commitment-confirm-reverse sequence with a hard expiry, the platform can hold the user's balance unchanged until confirmation lands, and reverse cleanly on expiry; within that band, the transaction either completes or leaves no trace, and the phrase is used deliberately and narrowly. Where a gateway does not expose such a sequence — most of them, for fiat legs — the platform records a pending state with an explicit expected settlement window, shows the user that the funds are in flight, and resolves the state on reconciliation. The failure mode is a delayed confirmation, not a vanished balance. What the platform refuses to do is display a provisional credit as a settled balance, because a user who trades against funds that later fail to arrive has taken a position with money that did not exist.



6.3 Finality, Reconciliation and Compliance Interfaces

Finality means different things in different corridors, and each adapter publishes its own definition rather than being flattened into a single platform-wide claim. For a stablecoin leg it is block confirmation depth. For a fiat leg it is the gateway's settlement advice, which the platform treats as final for accounting purposes and reconciles against the gateway's periodic statement. A corridor whose statement cycle is monthly has a long practical finality horizon, and the platform discloses that horizon to users rather than describing all corridors as instant.

Reconciliation runs on a fixed cadence against each gateway's records, over a window that extends past the settlement horizon. Discrepancies open an item with an owner and a deadline; unresolved items past deadline suspend the corridor's ability to accept new inbound traffic. Reconciliation output also feeds the routing cost function above, so an unreliable corridor falls out of the preferred set on evidence rather than on a decision.

Identity verification and anti-abuse controls sit as an aspect on the adapter layer. The platform collects what a corridor requires to move funds and what its own risk policy requires, and attaches the result to the settlement intent rather than to the trading account. This keeps corridor-specific requirements from hardening into the platform's user model: a requirement that applies because of one corridor's rules does not impose itself on a user in another corridor, and when a requirement changes, it changes in one place.

What this layer does not do is act as a licensed financial institution in any corridor, or offer deposit, credit, or custodial fiat services in local currency. It routes between a user's chosen gateway and the platform's own settlement balance, and the local leg is performed by the gateway under its own authorization. §13.2 states the corporate position.

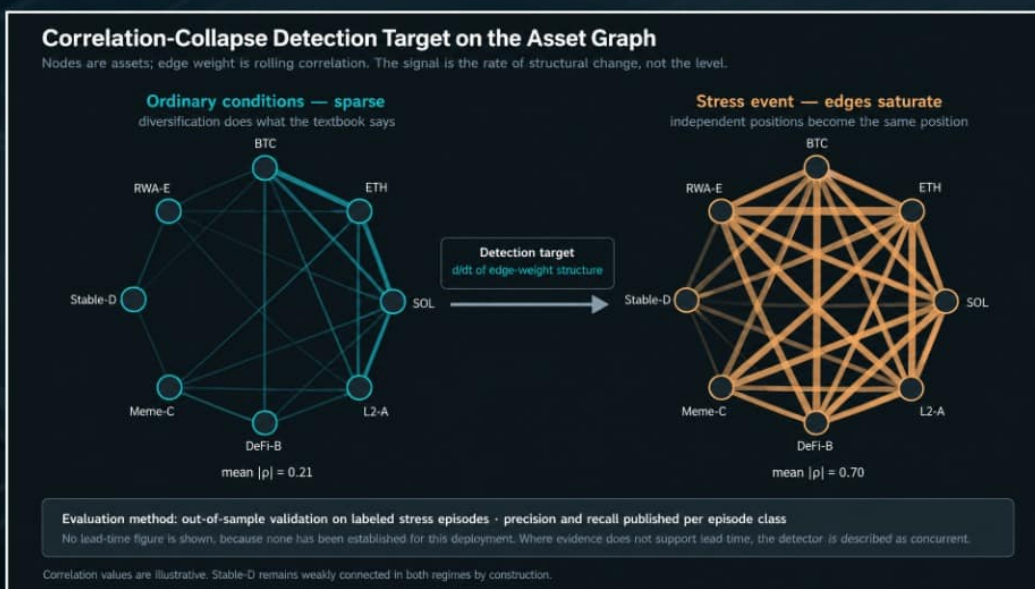
7.0 AI Sentinel and Policy-Constrained Execution

A surveillance system that predicts price is a marketing asset. A surveillance system that sees a book losing depth faster than it is losing price, and can act before the liquidation queue turns into a chain, is an operational one. This chapter is about the second category, and about what a model is permitted to do to a user's account without the user present.

7.1 Telemetry and Correlation-Collapse Detection

Every event in the core — orders, cancellations, fills, margin state changes, funding accruals, liquidation instructions, withdrawal attempts — lands on a bus the intelligence layer subscribes to. The layer reads. It does not execute orders, hold balances, or sit on the critical path between a user's instruction and the matching engine. That separation is enforced at the interface level, and it is what allows the sentinel to fail without taking the venue with it.

Position and market state is represented as a graph with assets as nodes and correlation as edge weights, computed over rolling windows across several horizons. The interesting signal is not the level of correlation but the rate at which the edge structure reorganizes. In ordinary conditions the graph is sparse: most assets have weak or no relationship, and portfolio diversification does what the textbook says. In a stress event the edges saturate — positions that were independent become the same position, and a user who believed they were hedged discovers they are doubled. Detecting that transition early enough to change admission behavior is the target.



Whether the transition is detectable ahead of the price event that usually accompanies it is an empirical question, and the answer is not assumed here. The layer maintains a labeled history of stress episodes, evaluates detection against windows the model never trained on, and publishes precision and recall per episode class. A detector with adequate precision and poor recall is useful for a throttling decision and useless for an alarm, and the published numbers tell an operator or a reader which one they have. Where the evidence supports a lead time, it is stated with the sample it was measured on; where it does not, the detector is described as concurrent and used accordingly.

7.2 Anti-Cascade Sentinels and the Action Set

Depth distribution is continuously mapped, not just the top of book. A book can show a respectable best bid and be effectively empty for a further two percent, and the depth profile's change over a rolling window is a better predictor of cascade conditions than the price level itself. Where the profile thins below a percentile band while open interest in the affected instrument concentrates on one side, the engine flags a cascade precursor: a configuration in which a single liquidation order can consume the liquidity that other positions' margins depend on.

Precursors feed the action set, which is deliberately small and gradeable.

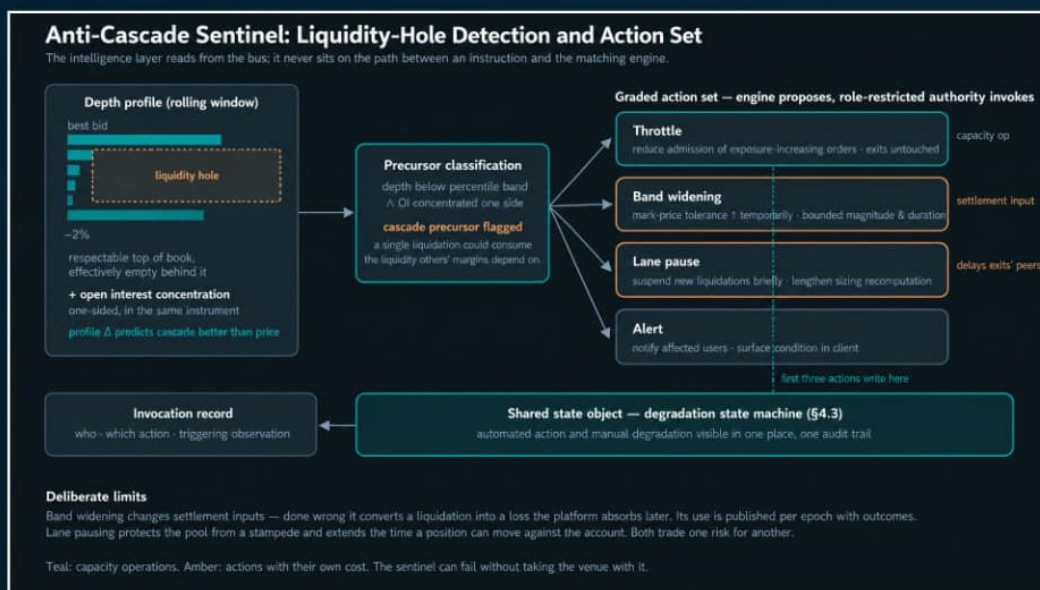
Throttle. Reduce admission rate for exposure-increasing orders on the affected instruments, leaving exits untouched.

Band widening. Expand the mark price's source tolerance temporarily and lengthen the basis correction's response, so that a thin book's prints are less likely to drive settlement in the interval.

Lane pause. Suspend new liquidation instructions on the affected instrument for a bounded interval, or lengthen the sizing recomputation described in §5.2, so that liquidations already in flight do not stampede the same liquidity.

Alert. Notify affected users with positions in the flagged configuration and surface the condition in the client.

The first three are capacity operations on the admission path of §4.3, and they share its state machine, which means an automated action and a manual degradation are visible in one place with one audit trail. Who may invoke what, and under which evidence thresholds, is a parameter question; the engine proposes, and the authority to widen a mark price band or pause a liquidation lane is restricted by role, with every invocation recorded against the observation that triggered it.

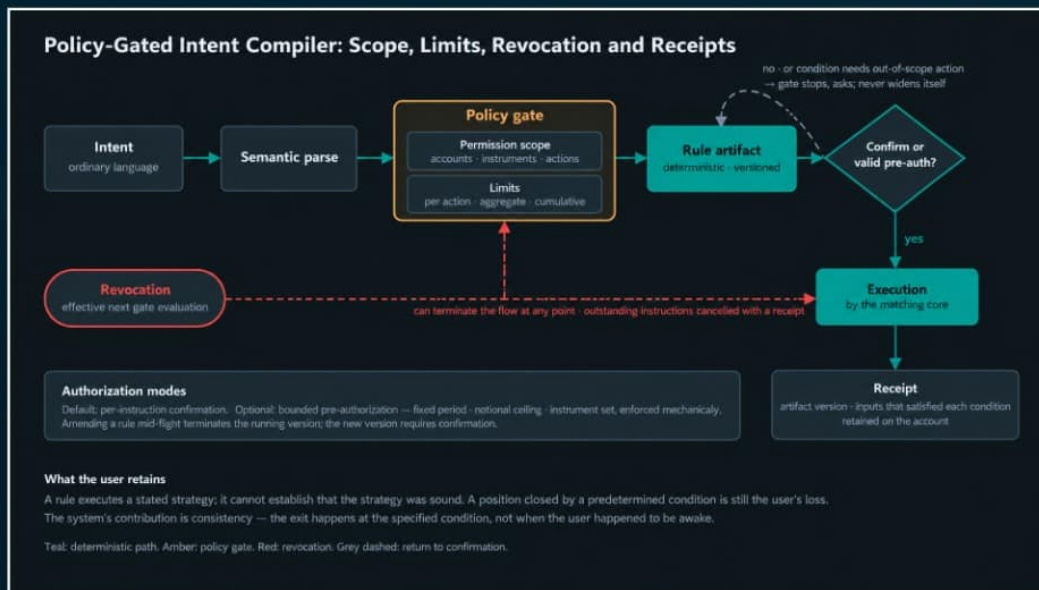


Two limits on the action set are deliberate and should be stated. Band widening changes settlement inputs, and doing it wrong converts a liquidation into a loss the platform absorbs later; it is bounded in magnitude and duration, and its use is published per epoch with outcomes. Lane pausing delays liquidations, which protects the pool from a stampede and extends the period during which a position can move further against the account. Both are damage-limiting tools with their own costs, not free improvements, and an account holder reading this should understand that under cascade conditions the platform may be trading one risk for another.

7.3 Intent Compilation Under Policy Gates

The execution assistance in this layer turns a user's instruction in ordinary language into a rule artifact the engine can execute deterministically: work an entry in five tranches over a session, exit if the correlation of the position against a stated reference exceeds a bound, rebalance to target weights on a schedule. The artifact is a rule set — parameters, conditions, bounds — that a user can read, amend, or refuse, and it is logged in a form that allows a later question about what the system did to be answered from the record rather than from recollection.

Between the user's instruction and any execution sits a policy gate with four components. **Permission scope** fixes which accounts, instruments, and actions the artifact may touch — a rebalancing rule for the spot account has no access to margin. **Limits** bound notional per action, aggregate exposure, and cumulative activity over the authorization period. **Revocation** takes effect on the next gate evaluation, and outstanding instructions not yet executed are cancelled with a receipt. **Receipts** record what was executed against which artifact version, with the inputs that satisfied each condition, and are retained on the account.



Default operation is per-instruction confirmation. A user who wants an artifact to run without them present can grant a bounded pre-authorization — a fixed period, a fixed notional ceiling, a defined instrument set — which the gate enforces mechanically rather than behaviorally. When an artifact's conditions can only be satisfied by an action outside its scope, the gate stops and requests confirmation; it does not widen its own authority. And artifacts are versioned, so amending a rule mid-flight terminates the running version and requires the confirmation step for the new one, which prevents a small parameter change from silently inheriting a broad authorization.

What the user retains through all of this is the part that cannot be delegated. A rule can execute a stated strategy; it cannot establish that the strategy was sound, and a position closed by a predetermined condition is still a loss the user incurred. The system's contribution is consistency — the exit happens at the specified condition rather than at the moment the user was awake to notice it — and consistency in execution does not convert a losing view into a winning one.

7.4 Model Governance and Human Accountability

Detectors decay. A model calibrated on one market regime registers the next one as noise until it has been retrained, and retraining is itself a risk: fit the model to the most recent stress episode and it will be exquisitely tuned to the conditions that just passed. The layer maintains a versioned registry with the training window, feature set, and performance evaluation attached to each version in production, and tracks live precision and recall against the labeled history. Detection quality falling below a threshold demotes the model to advisory-only, removing its authority to invoke an action, and that demotion is automatic and logged.

Backtests are treated as a filter, never as evidence of live performance. A detector must clear an out-of-sample bar before deployment and then continue to clear it live; the two numbers are published separately, because the gap between them is the most informative single figure about a detection system and the one most often omitted.

Accountability for outcomes sits with the account holder. The sentinel's actions operate on venue-wide risk parameters — admission rates, settlement inputs, liquidation lanes — and not on the user's strategy or the soundness of their position. A throttled entry, a widened band, or a paused liquidation changes the conditions a user trades under, and the platform discloses when these occur; none of them is a recommendation or a warranty, and none transfers a user's loss to the platform. An operator who invokes an action outside its published evidence threshold is exposed to the challenge process in §9.5 on the same terms as any other service obligation.

8.0 Capital Democratization

Early-stage allocation is where a retail user's disadvantage is most visible and least defensible. Institutional desks get calls; retail users get a page that may or may not load in time, and whatever they receive reflects how fast their connection happened to be that morning. The protocol in this chapter replaces an allocation by arrival order with an allocation by published function, and it accepts the consequence that some users will receive less than speed would have given them.

8.1 Discovery Pipeline, Disclosure and Launch Lifecycle

Projects enter through a staged pipeline, and each stage has an exit condition rather than a vote.

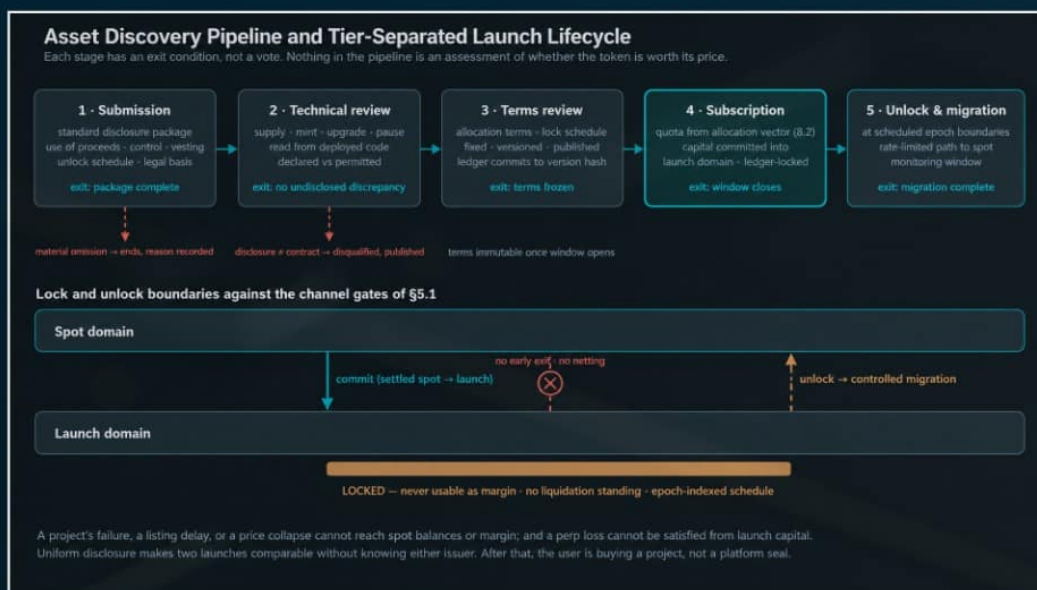
Submission. A project provides the standard disclosure package: what the token does, what the proceeds fund and on what schedule, who controls what, vesting terms for every holder class, the unlock schedule, and the legal basis on which the issuer believes it can offer the instrument. Material omissions at this stage end the process, and the reason is recorded.

Technical review. The disclosure is checked against the contract, if there is one. Token supply, mint authority, upgrade authority, pause capability, and transfer restrictions are read from the deployed code where code exists, and declared intentions are compared against what the code permits. A token whose contract grants the issuer an unbounded mint function is not disqualified automatically; the discrepancy between the disclosure and the contract is, and it is published either way.

Terms review. The allocation terms, lock schedule, and issuer obligations are fixed, versioned, and published before the subscription window opens. Once the window opens, the terms cannot change — a launch whose terms can be amended during subscription is not a launch with terms, and the ledger's commitment to the version hash makes that enforceable rather than merely promised.

Subscription. Users commit capital into the launch channel against quota eligibility. Capital is locked for the stated schedule at the ledger level.

Unlock and migration. Tokens and any returned capital become available at scheduled epoch boundaries. Migration from the launch domain into the spot domain occurs through a controlled path with rate limits and a monitoring window, so that a listing's opening liquidity conditions are observable rather than emergent from a simultaneous release.



The disclosure template is uniform across launches precisely so that a user can compare two of them without knowing either issuer. Uniformity is not a quality judgment, and nothing in the pipeline constitutes an assessment of whether the token is worth its price. What the pipeline does is make the terms legible and the enforced behavior match the declared behavior; after that, the user is buying a project and not a platform seal.

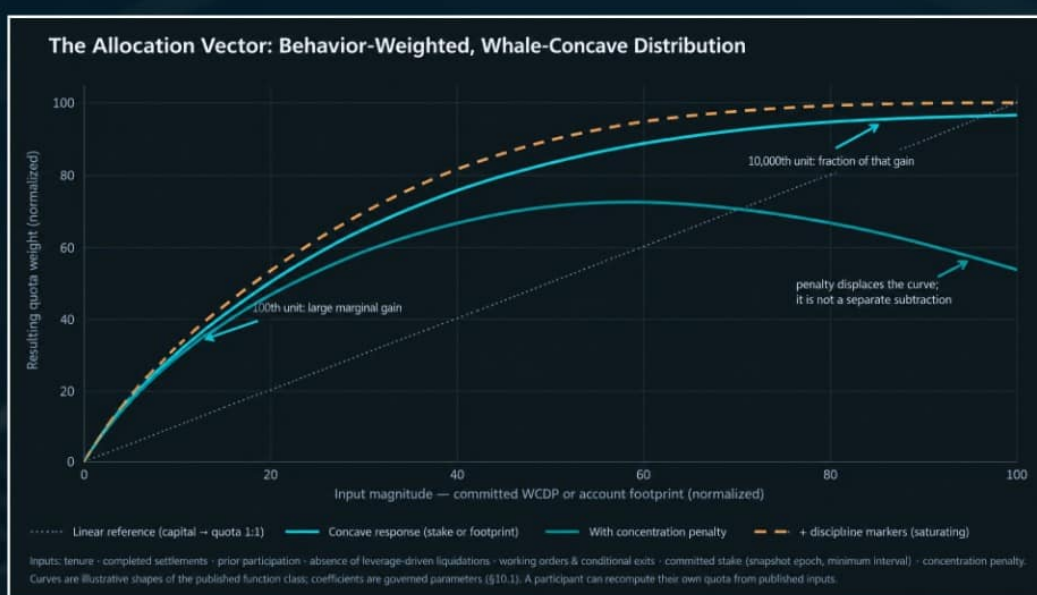
8.2 The Allocation Vector

Quota is computed from a vector with two classes of input: a behavior profile built from the user's account history, and committed WCDP weight. The allocation function is concave in both, and the concavity is the design.

Input class	Examples	Direction
Account history	Tenure, completed settlements, prior launch participation, absence of liquidations attributable to excess leverage	Increasing, saturating
Committed stake	WCDP committed before the window	Increasing, saturating
Concentration penalty	Aggregate committed capital, prior allocation size relative to the pool	Decreasing

The saturating term does most of the anti-concentration work. A user with a large historical footprint gains over a new user, and the marginal gain from the ten-thousandth unit of footprint is a fraction of the marginal gain from the hundredth. Committed stake carries the same shape: weight grows with the commitment, and the growth rate declines, so capital cannot convert linearly into allocation. Combined with the concentration penalty, a large committed balance produces a smaller share of a fixed pool than its nominal proportion, which is the mechanism by which a launch pool stays distributed rather than captured.

The commitment must precede the window by a minimum interval, and the interval is long enough that borrowing stake for a window is not viable. Weight is measured at the snapshot epoch rather than continuously, so the eligibility of a submission is fixed at a moment every participant can see in advance.



Two things this function does poorly, stated rather than discovered by participants. A disciplined long-tenured user who commits nothing may receive a quota smaller than an active trader would expect, because the committed-stake term is real and not cosmetic — the pool is a scarce resource and eligibility competes. And the behavior profile uses history on this platform, which means a user with an established record elsewhere starts at a disadvantage. That disadvantage is deliberate in the short term; the profile accrues here, and importing reputation is not supported because there is no reputation source the protocol can verify rather than trust.

Allocation outcomes, eligibility inputs, and the version of the function in force are published per window. A participant can recompute their own quota from the published inputs, and a participant who believes the computation was wrong has the same recourse as any other challenge under §9.5.

9.0 WCDP: Protocol Utility and Sustainable Resource Economics

WCDP is the unit in which the network's scarce resources are priced, allocated, and bonded. Its value case is inseparable from whether those resources are genuinely scarce and genuinely needed; a token that is not demanded by operators and users has no economics to analyze, and no supply schedule repairs that.

9.1 Service Demand and Provider Obligations

The network consumes real work in five places.

State validation. Operators verify epoch proofs, monitor commitment publication, and raise alerts when a proof is missing or a root is inconsistent. Their obligation is availability and honest reporting, and the evidence of failure is objective: an unverified epoch or a demonstrably false alert.

Proof computation. Generating recursive STARK proofs for high-volume epochs is expensive in compute. Prover operators supply that capacity under a service agreement with published throughput and latency targets, and the proof either verifies or it does not.

Intent solving. The execution layer of §7.3 requires compute to compile user intent into deterministic rule artifacts and to evaluate policy gates. Premium bandwidth — shorter compilation queues, more concurrent active intents — is a metered resource.

Allocation participation. The launch protocol of Ch.8 weighs eligible commitments when distributing quotas. Eligibility is a protocol resource, not a claim on any particular offering, and the quota a participant receives is determined by the published allocation function.

Governance. Parameter voting carries weight in proportion to committed stake and lock duration, as Ch.10 sets out.

Each of these functions has a counterparty who can fail at it and can be penalized for failing. That is where token demand originates. Operators must hold and bond WCDP to be eligible for any of the roles above; users consume metered resources or commit stake for priority and weight. Neither category is a discount, and neither is discretionary.

9.2 Four Protocol Resources

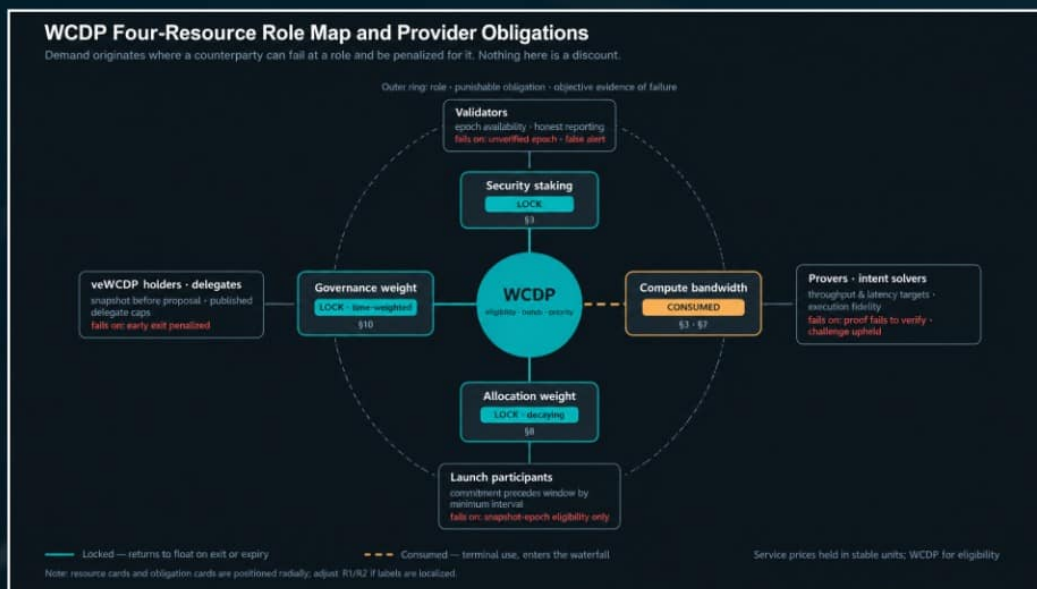
WCDP accesses four distinct resources, and the distinction between consumed and locked matters for how each behaves in the supply picture.

Security staking is a lock. Validators post WCDP as a condition of verifying epochs and raising challenges; the stake is returned on exit after an unbonding period, reduced or forfeited if the operator's work is shown to be dishonest or negligent through the challenge process of §9.5.

Compute bandwidth is consumed. Proof generation procurement and intent-compilation capacity are bought with WCDP, and the tokens paid are not returned — they are routed into the revenue waterfall of §9.4, which is a terminal use rather than a temporary lock.

Allocation weight is a lock with a decaying effect. Stake committed before a launch window carries weight through the allocation function; weight does not transfer after allocation and does not survive the window.

Governance weight is a lock with a time dimension. Longer commitments command more weight, as §10.1 specifies, and the lock is released only at expiry or through the exit path.



Pricing in a volatile unit creates friction that should be acknowledged rather than designed away by assumption. The protocol holds **service prices in stable-denominated units** and uses WCDP for eligibility, bonds, and priority. An operator bonds WCDP to qualify, then pays for metered compute at a stable price translated into WCDP at execution. This keeps the contract legible to an operator with a budget while keeping eligibility scarce. It also means the token's demand comes from the requirement to hold and commit, which is a more durable source than a fluctuating unit of account — and less elastic to short-term price movement, which cuts both ways.

9.3 Supply Architecture and Emission Logic

Total supply is fixed at 1,000,000,000 WCDP. The figures below are the genesis parameters; they are adjustable by governance only within the bounds published, and the bounds themselves are the subject of the lock described in §10.2.

Allocation	Share	Terms
Validator and prover incentives	30%	Emitted against verified work per epoch
Ecosystem and integration	18%	Adapter and corridor development, milestone-gated
Launch distribution	12%	Distributed through the Ch.8 allocation protocol
Core contributors	18%	Cliff, then linear over subsequent epochs
Protocol reserve	12%	Held in assets other than WCDP; see §9.5
Liquidity provisioning	10%	Market operations, disclosed per period

Emission is the compensation mechanism for work that fee revenue does not yet cover. The schedule is indexed by epoch, not by calendar, and decays by a governed factor each epoch toward a floor set by the incentive parameter floor. The design intent is that emission share of operator compensation falls as usage-driven revenue rises, and the protocol publishes both figures each epoch so the substitution is observable rather than asserted.

Vesting is expressed in epochs for the same reason: the relevant question for a reader is how much supply becomes liquid relative to network milestones, and calendar dates obscure that.

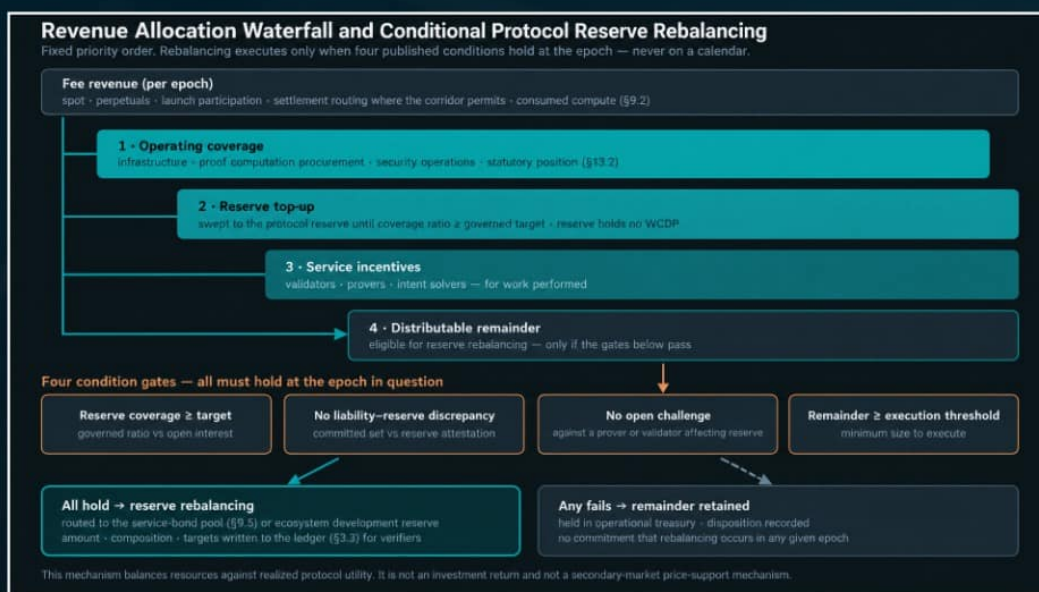
9.4 Protocol Reserve Allocation and Ecosystem Sustainability

Fee revenue — spot, perpetuums, launch participation, and settlement routing where the corridor permits it — enters a waterfall with a fixed priority order.

- 1. Operating coverage.** Infrastructure, proof computation procurement, security operations, and the cost of maintaining the statutory position described in §13.2.
- 2. Reserve top-up.** Swept to the protocol reserve until the reserve coverage ratio is at or above its governed target.
- 3. Service incentives.** Payments to validators, provers, and intent solvers for work performed.
- 4. Distributable remainder.** Whatever survives the first three steps is eligible for protocol reserve rebalancing.

Protocol reserve rebalancing is governed by deterministic rules rather than arbitrary intervention. Four published conditions must hold at the epoch in question: reserve coverage at or above target; no unresolved discrepancy between the committed liability set and the reserve attestation; no open challenge against a prover or validator whose outcome could affect the reserve; and a distributable remainder above the minimum execution threshold. When a condition fails, the remainder is retained in the operational treasury and its disposition recorded. Protocol reserve rebalancing does not execute on an arbitrary calendar schedule.

Surplus protocol allocations are routed to the service-bond pool of §9.5 or dedicated ecosystem development reserves according to governed algorithmic parameters. Both paths reinforce network capital adequacy, ensuring long-term collateral depth for high-throughput corridors. Protocol reserve rebalancing events are published directly to the ledger described in §3.3—recording amount, reserve composition, and target allocations—so that independent verifiers can reconcile network performance each epoch. This mechanism maintains resource balance against realized protocol utility; it does not constitute an investment return or secondary market price-support mechanism.



9.5 Service Bonds, Slashing, and Reserve Separation

WCDP's role in protecting users is bounded and specific. Operators post bonds denominated in WCDP that stand behind their own performance obligations: proof correctness, execution fidelity for solved intents, truthful telemetry reporting, and epoch-level availability. Slashing is triggered by verifiable evidence rather than administrative judgment — a proof that fails to verify, a challenge resolved against an operator, a demonstrated misreport against the raw event log. Governance sets the size of bonds and the severity schedule. It does not decide whether a given piece of evidence counts.

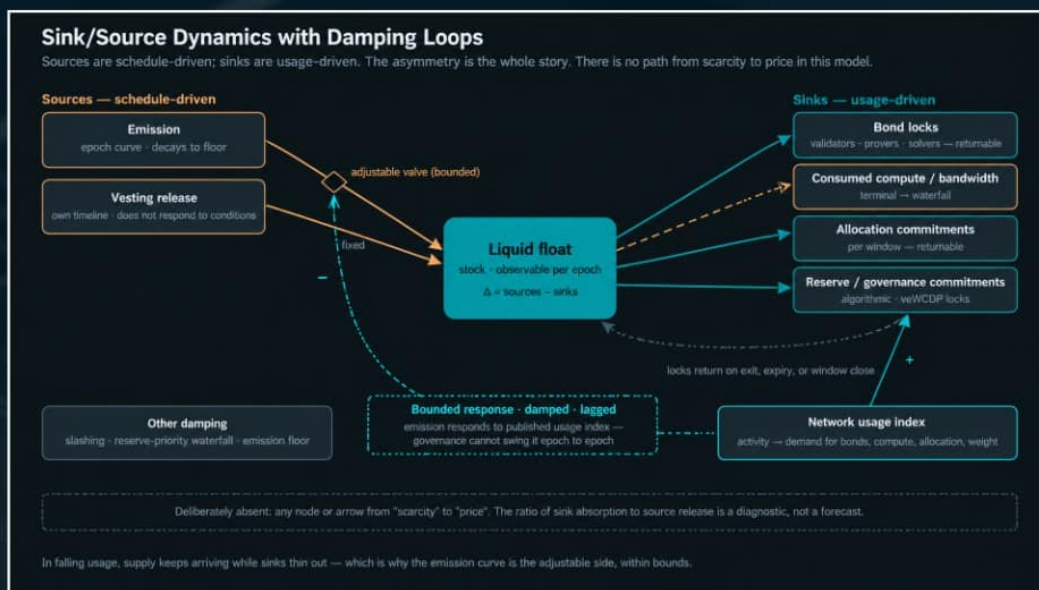
The loss-absorbing layer for customer balances sits elsewhere. The protection pool that absorbs liquidation shortfalls, described in §5.2, is funded by the protocol reserve, and the reserve is held in assets that do not include WCDP. That separation is deliberate and it is the correction to a design that is easy to write and hard to defend: a venue whose safety net is denominated in its own token has built a mechanism that weakens exactly when its users need it, because the conditions that strain a venue are the conditions that depress its token.

Slashing transfers bonded WCDP from the penalized operator to the parties harmed or to the reserve, or burns it, depending on the parameter in force. Bonded capital is at risk for operator conduct. Token holders who hold WCDP and nothing else carry exposure to the network's utility, which is a different thing from a guarantee on deposits, and the protocol's disclosure language keeps the two apart.

9.6 Sink and Source Dynamics

Supply enters through emission and through the vesting schedules of §9.3. It leaves float through four sinks: bond locks, consumed compute and bandwidth, allocation commitments, and algorithmic reserve commitments. The two sides are not symmetric in how they respond to conditions, and the asymmetry is the whole story.

Sources are largely schedule-driven. Emission follows the epoch curve regardless of usage, and vesting releases on its own timeline. Sinks are usage-driven: nobody bonds, consumes bandwidth, or commits allocation weight in the absence of activity. In a period of falling usage, supply keeps arriving while the sinks thin out, which is why the protocol treats the emission curve as the adjustable side and gives it a bounded response to a published usage index, damped and lagged so that governance cannot swing it epoch to epoch.



What this structure supplies is observability. A reader or a participant can see, epoch by epoch, whether resource demand is absorbing supply faster than emission and vesting release it. That ratio is a diagnostic. It is not a forecast, and treating a favourable reading as evidence about future token value would misread what the mechanism does.

10.0 Parameter Governance

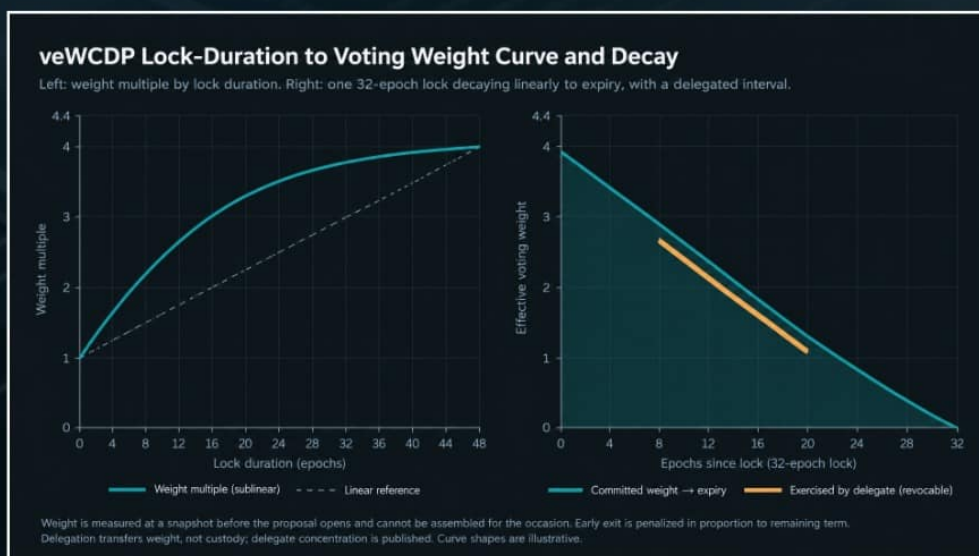
Governance in a platform of this kind has a narrow legitimate scope: deciding where the dials sit, and being unable to reach past them. Most of the argument in this chapter is about the second half, since the first is easy to grant and the constraint is what usually gets forgotten.

10.1 The veWCDP Model and Governance Scope

Voting weight derives from committed WCDP and the length of the commitment. The weight grows with the committed amount, rises with lock duration along a curve that flattens past the long end, and decays linearly as the lock approaches expiry. A voter who wants influence over a proposal passing next epoch must have committed before the proposal opened, which means weight cannot be assembled for the occasion.

Lock duration	Weight multiple	Exit terms
Short (minimum)	1×	Released at expiry
Medium	Increasing, sublinear in duration	Released at expiry, or early exit with penalty
Long	Approaching maximum multiple	Released at expiry; early exit penalized in proportion to remaining term

Delegation is supported to the extent it lets a holder without the time or expertise to review proposals transfer weight to a delegate. Delegation transfers weight, not custody, and it is revocable at any point with effect on the next proposal window. It also concentrates weight in a small number of delegates, which is a recognized weakness of the model rather than a design feature; the platform publishes the delegate distribution so that concentration is measurable, and caps on the weight a single delegate may carry are a parameter subject to the process below.



What governance may decide is enumerated, and the list is a closed one.

- Risk parameters: isolation pool target coverage, maintenance margin floors, mark price screening tolerances and their adjustment bounds, liquidation lane pause limits.
- Launch parameters: allocation function coefficients, minimum stake intervals, disclosure template fields, concentration penalty severity.
- Emission and incentive parameters, within the floors and ceilings fixed at genesis.
- Fee parameters within published bounds, and the revenue waterfall priorities' relative shares.
- Operational membership: validator and prover onboarding standards, bond sizes, slashing schedules.
- Delegate weight caps.

What it cannot decide is a shorter list and a harder one. Governance cannot move, freeze, or reallocate user balances; cannot alter a user's open position or modify a past fill; cannot amend the proof system's published verification keys or retroactively alter a commitment; cannot act on any individual account's outcome; and cannot use the protocol to extend or limit the corporate legal position. Proposals attempting any of these are rejected at the validation stage rather than defeated at a vote, so that the outcome does not depend on turnout.

10.2 Proposal Lifecycle and Emergency Guardrails

A proposal passes through stages with distinct requirements.

Draft and review. A draft enters public review with the parameter change specified as a diff against the current value, the rationale, and where applicable an impact analysis from the affected subsystem — a margin floor change without a liquidation-backtest summary does not advance. Review has a minimum duration that scales with the magnitude of the change.

Vote. Voting runs on committed weight measured at a snapshot before the review period opened. Quorum requirements are set per parameter class; changes to risk parameters of the highest class require both higher participation and a supermajority, which reduces the frequency with which a small active group can move the parameters that determine whether other users' positions survive a bad day.

Timelock. Passed proposals wait in a timelock before execution. The delay serves two purposes: users with positions exposed to a parameter change can act during the interval, and a proposal that passed on a misrepresentation can be withdrawn.

Execution and observation window. The parameter takes effect, and a bounded observation window follows in which the change can be reversed by a narrower procedure than the one that enacted it, provided the reversal is also a published parameter change. Reversal is not available for anything executed in a user's account, because nothing in scope can touch one.

Emergency authority exists and is deliberately constrained rather than denied. A small set of conditions — a detected proof-verification failure, a corridor reconciliation break beyond a severity threshold, a market condition in which the ordinary quorum process cannot complete fast enough to act — permits a defined response by a restricted group drawn from the technical committees of Ch.11, using a multisignature arrangement with a published signer set. The permitted responses are enumerated in advance: suspend admissions on affected instruments, pause a liquidation lane within the bounds of §7.2, take a corridor out of service, suspend emission. Each invocation requires a signed justification against the triggering observation, is time-bounded, and lapses automatically if not ratified through the ordinary process. Emergency authority cannot move funds, alter positions, or change the proof system.

The honest account of this arrangement is that it trades a slower and more legitimate process for unlikely, bounded, temporarily faster action by a smaller group, and that the trade favors speed during events where a delayed decision is itself a decision. Users who find the trade unattractive should weigh it against the alternative — an automatic mechanism, which is faster still and cannot exercise judgment about a situation nobody anticipated.

11.0 Core Architecture Team

The WCDPQ protocol is stewarded through an open-source development foundation model. Following industry standards for decentralized protocols, early core technical contributions are orchestrated by specialized engineering workstreams and independent technical review committees. Transition toward broad-based foundation council governance and verified institutional contributor delegations proceeds under the RFC framework outlined below.

11.1 Organizational Principles and Technical Committees

The operating model is drawn from open-source foundation practice rather than from a corporate product organization, for a reason that is technical rather than cultural. The system's expensive properties — proof soundness, deterministic replay, risk isolation — are the kind that fail silently under schedule pressure. A structure in which a lead engineer can ship a constraint change without a second party who understands the circuit reviewing it will, under deadline, produce exactly that failure.

Three things follow from that premise.

Work proceeds through published RFCs. A material change to a subsystem enters as a written proposal with a specification, a security argument where one applies, and a migration path. The RFC is public, its review is on the record, and the resulting implementation is traceable to it. This is slower than committing a fix, and the alternative has been tested extensively elsewhere.

Review authority is separated from authorship. The cryptography committee approves circuit constraint changes and proof parameter updates; the risk committee approves margin, liquidation, and mark price methodology changes together with the supporting backtests; the protocol economics committee approves emission, fee, bond, and allocation function parameters. A committee member employed on the team that authored a change can participate but counts toward no approval quorum for it.

Authority is bounded by subject matter. None of the three committees can overrule another outside its domain — the risk committee cannot relax a circuit constraint, and the cryptography committee cannot adjust a maintenance margin floor — which forces disagreement into a proposal rather than into a private escalation.

11.2 The Role Specification Matrix

Role	Core accountability	Capability baseline	Commit tee
Chief Protocol Architect	Cross-subsystem invariants; RFC arbitration; interface stability between verification, execution, and settlement layers	Distributed systems architecture at exchange or comparable scale; multi-year protocol design record	Chair, ex officio
Cryptography Lead (ZK and MPC)	Constraint system correctness; proof parameter selection; threshold signature lifecycle design	Applied cryptography; STARK or SNARK proof systems in production; threshold signature implementations; formal verification background	Cryptogra phy
Formal Verification Lead	Machine-checked proofs of critical constraints; differential and adversarial test suites; verifier soundness arguments	Formal methods; interactive theorem provers or model checkers; protocol verification against adversarial models	Cryptogra phy
Matching Engine Lead	Sequencing determinism; sharded book design; degradation state machine and priority lane behavior	Low-latency systems engineering; deterministic system design; capacity and failure-mode testing under adversarial load	Risk
Risk Systems Lead	Margin methodology; liquidation waterfall; mark price composition; isolation pool sizing and recovery procedures	Quantitative risk in derivatives venues; margin model design; liquidation and backtesting evidence standards	Risk
ML and Telemetry Lead	Detector design; out-of-sample evaluation discipline; model registry and demotion policy	Applied ML with time-series or graph methods; evaluation methodology; production monitoring of model decay	Risk
Protocol Economics Lead	Emission and fee mechanics; bond and slashing schedules; allocation function design and simulation	Mechanism design or token economics with published work; adversarial game analysis; empirical calibration discipline	Protocol econom ics
Payment Infrastructure Lead	Adapter contract; reconciliation schema; corridor certification; finality semantics per corridor	Payment systems integration; reconciliation and settlement operations; multi-jurisdiction corridor deployment	Protocol econom ics
Security and Access Lead	Key distribution; access separation; monitoring; incident response	Infrastructure security; key management; incident command in financial systems	Cryptogra phy
Governance and Open-Source Steward	RFC process integrity; proposal lifecycle and timelock operations; contributor pathways; publication obligations	Open-source foundation governance; specification process management	Ex officio, non-voting

Two rows in that table are doing work that a conventional engineering org chart would leave implicit. The formal verification role exists because cryptographic systems fail at the specification boundary more often than in the mathematics: a circuit that proves the wrong statement is sound and useless. And the governance steward is a technical function here, not an administrative one, because the timelock, snapshot, and emission-parameter machinery described in Ch.10 is code whose correctness determines whether governance promises hold.

Coverage targets are published rather than stated as intentions. Each committee maintains a minimum of independent reviewers per domain, and where a domain has fewer than the minimum — a real condition for a young protocol in a specialized field — the protocol publishes the shortfall and restricts the parameter classes that committee may approve. A committee that cannot staff review of high-consequence changes does not get to approve them by default.

11.3 Open-Source and External Contribution

Verifier implementations, the command-line verifier, client SDKs, the reconciliation schema, and the specification documents are public. The intent is that an independent party can build a verifier from the specification without using protocol-supplied code, and that the reconciliation schema is open so a corridor partner or a supervisory body can consume it with their own tooling.

The critical path is not open, and the boundary is drawn where it is for a stated reason. Matching engine internals, risk engine implementation, key management operations, and sensitive telemetry stay closed, because publishing them narrows the search space for an attacker faster than it widens the reviewer pool. What is checkable by outsiders is the property that matters for trust — whether committed state follows from published rules — and that property is checkable through the proof system and the client, both of which are public. A contributor path exists from the open components inward: verified contributions to the public layers, sustained review participation, and committee sponsorship are the route to involvement in the closed ones.

12.0 Evolution Roadmap

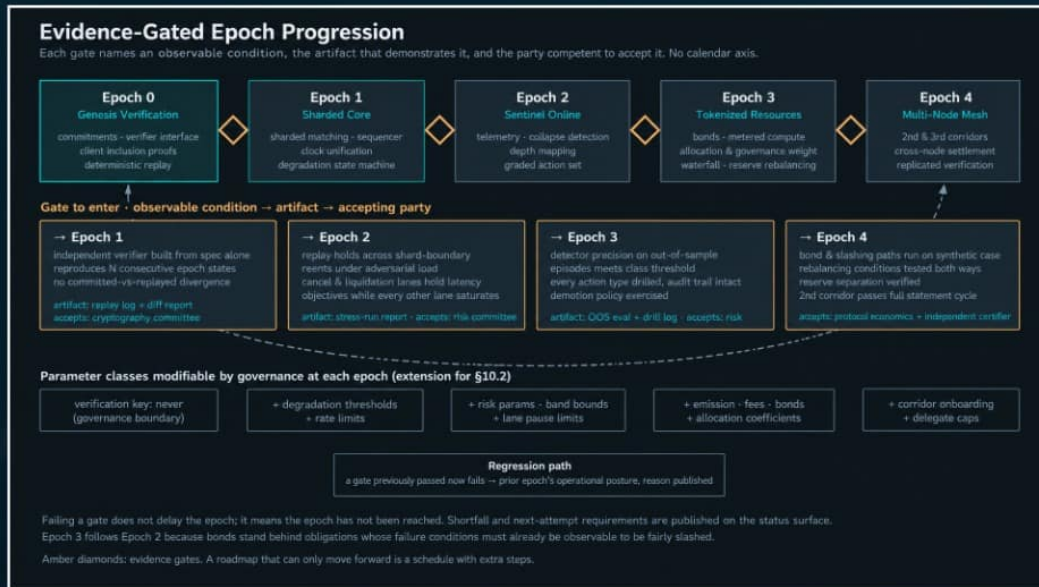
While technical delivery is strictly gated by verified cryptographic and operational milestones rather than arbitrary deadlines, the protocol aligns its development epochs with indicative sequential release windows to coordinate ecosystem auditing and integration.

12.1 Gating Philosophy and Epoch Definitions

Each epoch transition is gated by evidence rather than elapsed time. A gate names an observable condition, the artifact that demonstrates it, and the party competent to accept the demonstration. Failing a gate does not delay the epoch; it means the epoch has not been reached, and the measured shortfall is published with the next attempt's requirements.

Epoch	Scope	Gate to enter	Indicative Horizon
0 — Genesis Verification	Commitment publication, verifier interface, client-side inclusion proofs, deterministic ledger replay	Independent verifier implemented from the specification alone reproduces 30 consecutive epoch states; no unresolved divergence between committed and replayed state across the window	2026 Q3-Q4 Foundation Release
1 — Sharded Core	Sharded matching, global sequencing, clock unification, degradation state machine	Deterministic replay holds across shard-boundary events under adversarial load; cancellation and liquidation lanes maintain stated latency objectives through a stress run that saturates every other lane	Phase I Network Calibration
2 — Sentinel Online	Telemetry, correlation-collapse detection, depth mapping, graded action set	Detector performance published on out-of-sample episodes meets the precision threshold for its class; every action type invoked in a controlled drill with the audit trail intact; demotion policy exercised	Phase II Advanced Telemetry
3 — Tokenized Resources	Bonds, metered compute, allocation weight, governance weight, revenue waterfall, reserve rebalancing	Bond and slashing paths executed against a synthetic evidence case; reserve rebalancing conditions evaluated in both satisfied and unsatisfied states; reserve separation verified with the reserve auditor of record	Phase III Utility Activation
4 — Multi-Node Mesh	Second and third corridor deployments, cross-node settlement, replicated verification	Second corridor passes reconciliation certification over a full statement cycle; independent verifiers operating from a second jurisdiction confirm epoch proofs without protocol-supplied tooling	Phase IV Cross-Corridor Expansion

Epoch 0 precedes everything because a proof nobody has verified is a claim. Epoch 3 follows Epoch 2 rather than running in parallel with it because token bonds stand behind operator obligations, and an obligation whose failure conditions are not yet observable cannot be fairly slashed.



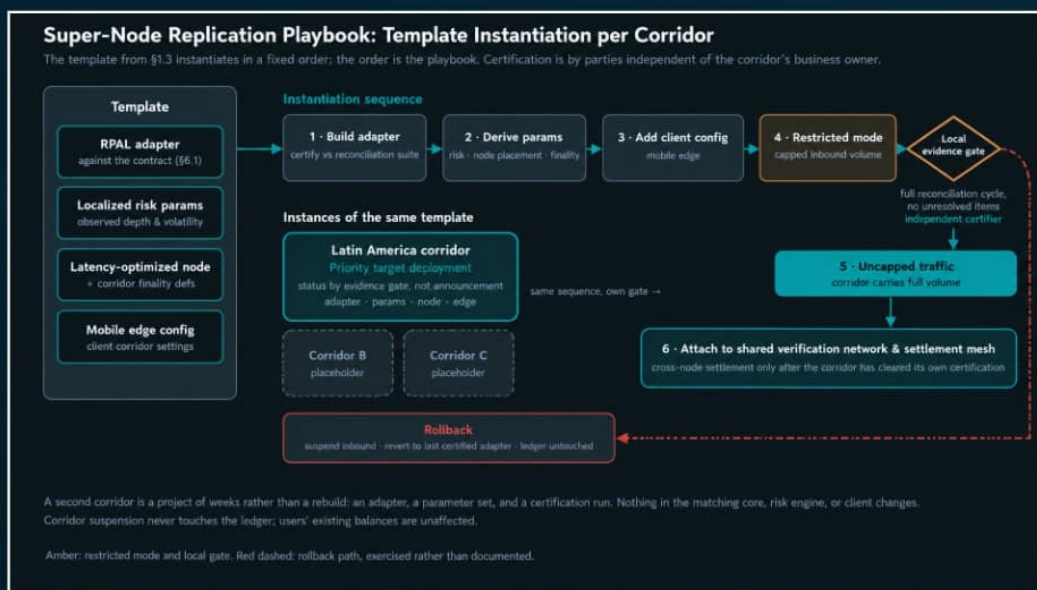
Published gate status is a standing obligation rather than a release announcement. Current state, last measured result, and remaining shortfall for the next gate are on the status surface each epoch, which means a reader can form a view about progress without relying on a summary produced for them.

12.2 The Super-Node Replication Playbook and Rollback

The template from §1.3 instantiates in a fixed order, and the order is the playbook.

An adapter is built against the RPAL contract and certified against the reconciliation suite. A parameter set is derived for the corridor: risk parameters localized against observed depth and volatility in that market, latency-optimized node placement, and corridor-specific finality definitions. The mobile client's corridor configuration is added. The corridor runs in a restricted mode with capped inbound volume until reconciliation completes a full cycle without unresolved items, and only then does it carry uncapped traffic.

Certification for a new corridor is executed by parties independent of the corridor's business owner, which keeps a market entry from being signed off by whoever negotiated it. Cross-node settlement is added only after a corridor has cleared its own certification, so that a settlement path never depends on a node still proving itself.



Both evolution axes have rollback paths, and both are exercised rather than documented. A parameter change that produces an outcome outside its forecast impact range reverses through the observation window in §10.2. A corridor that fails reconciliation past threshold is suspended for inbound traffic and reverts to its last certified adapter version, with users' existing balances unaffected because corridor suspension never touches the ledger. An epoch that regresses — a subsequent measurement failing a gate it previously passed — returns the network to the prior epoch's operational posture, with the reason published. Regression handling is the part of a conditional roadmap that decides whether the conditionality is real; a roadmap that can only move forward is a schedule with extra steps.

13.0 Risk Factors, Disclosures and Appendices

The risks below are the ones this architecture does not remove rather than the ones it addresses. A reader who has followed Ch.3 through Ch.9 has seen the mechanisms; this chapter is about what remains exposed after they work as intended.

13.1 Technical, Market and Token Risks

Constraint system defects. A bug in the circuit can make an invalid transition provable, or make a valid one unprovable. The first is the serious case: the proof verifies and says the wrong thing. Formal verification against the specification reduces this class and cannot eliminate it, since the specification itself is written by the same people who wrote the circuit. The practical defenses are the specification being public, multiple independent implementations of the verifier, and the fact that a defect becomes visible as a divergence between committed state and independently replayed state rather than as a silent loss.

Key compromise short of threshold. A single operator's shares being stolen does not permit a transfer, given a properly operating threshold scheme. An implementation fault that reconstructs more than it should, a backup containing shares in one location, or a deployment where several share holders run identical compromised images does. Geographic and operational separation of share holders is a mitigation whose effectiveness depends on it being maintained continuously, and it is a condition that erodes through ordinary convenience.

Oracle and reference data failure. Mark prices depend on external sources. A source that reports stale data without saying so, a set that becomes correlated because several venues derive from the same underlying feed, or a coordinated attempt to move a thin reference market all degrade settlement accuracy. Screening, exclusion, and bounded influence limit the damage over short intervals. They do not protect against a sustained mispricing that passes screening, and positions settled against such a price are settled.

Detector failure in both directions. A detector that fails to fire leaves the venue in its ordinary state during a cascade. One that fires spuriously throttles admission or widens price bands during ordinary trading, which costs users access and, in the band-widening case, shifts settlement inputs. Published precision and recall per episode class bound how well the failure rates are known; they do not make them small, and the correlation-collapse signal depends on historical episodes that a sufficiently novel market structure will not resemble.

Model and parameter obsolescence. Margin floors, screening tolerances, and allocation coefficients are calibrated against conditions that change. A slow drift leaves the system operating correctly against a world that no longer exists. Periodic recalibration is a governance commitment, which means its quality depends on the participation described in Ch.10 remaining genuine.

Market and liquidity risks. Extreme moves can exhaust the isolation pool, at which point the recovery path is a governed decision with the consequences described in §5.2. Stablecoin instruments used in settlement carry issuer, redemption, and peg risks that are not the platform's to control; a depeg affects the value of balances denominated in that unit regardless of the correctness of the ledger recording them. Corridors can be interrupted by changes at the gateway or in the underlying banking arrangement, stranding users' ability to move funds in or out even when the venue is functioning normally.

Token-specific risks. WCDP's utility depends on network activity. Demand for bonds, metered compute, allocation eligibility, and governance weight is derived demand, and if the network's services lose users, that demand falls regardless of the supply schedule. Vesting from the allocations in §9.3 releases supply on a timetable that does not respond to conditions. Governance parameters — emission rates, bond sizes, waterfall priorities, allocation coefficients — can be changed through the process in Ch.10 and a holder's expectation about future utility can be altered by a vote they did not participate in. Bonds are slashed against operator conduct, so a holder who stakes in a validating role carries exposure to that operator's behavior. And protocol reserve rebalancing funded from fee revenue executes strictly subject to the four sustainability conditions specified in §9.4; there is no commitment or representation that reserve distributions will occur in any given epoch.

Concentration of delegates. The veWCDP model concentrates effective governance weight over time as holders delegate. Caps on delegate weight are a parameter, and a parameter can be changed, including by the delegates it constrains.

13.2 Regulatory Boundary Statement

WCDPQ is developed and published as an open-source technical specification and software protocol stewarded by international foundation entities. The protocol developers and foundation contributors do not operate an exchange, brokerage, or custodial clearinghouse in Brazil or any specific jurisdiction.

Local currency conversion, fiat gateway settlement, and direct user onboarding are managed independently by licensed third-party regional partners operating under their respective local regulatory authorizations. Digital asset trading carries inherent capital risk, and past simulated performance does not indicate future outcomes. Nothing herein constitutes an offer of securities, collective investment schemes, or financial solicitation in any territory.

Appendix: Cryptographic Proof System Specifications

A.1 Field, hashes, domain separation

Arithmetic runs over the 64-bit Goldilocks field

$$p=2^{64}-2^{32}+1$$

with native support for 64-bit range decomposition. All in-circuit hashing uses Poseidon2 over this field, width 12, state rate 8. External digests use BLAKE3-256 over the canonical byte encoding of the same field elements, prefixed by a domain tag.

A.2 Liability commitment structure

The global commitment over assets:

$$C\varepsilon = \text{BLAKE3WCDP} - \text{EPOCH} - v1 \ \varepsilon \parallel a1 \parallel R1 \parallel \dots \parallel ak \parallel Rk$$

The blinded leaf preimage for account in asset at epoch :

$$l(u, a, \varepsilon) = \text{Poseidon2H}(u) \ \nu(u, \varepsilon) \parallel b(u, a, \varepsilon) \parallel \tau(a))$$

where the account reference is

$$H(u) = \text{BLAKE3user} - id \parallel account - salt \bmod p$$

and $\nu(u, \varepsilon)$ is the rotating nonce, the unsigned 64-bit balance in base units, the asset tag. Internal nodes carry both a digest and an arithmetic sum:

$$\Sigma(N) = \Sigma(L) + \Sigma(R)$$

$$D(N) = \text{Poseidon2D}(L) \ \Sigma(L) \parallel D(R) \parallel \Sigma(R))$$

The root publishes the pair $(D_{root}, \Sigma_{root})$, so total liabilities for an asset are readable without a tree traversal. Internal node balances and aggregate sums are represented using 64-bit limb compositions to natively prevent overflow across standard 18-decimal token representations, with field boundary checks enforced by the circuit.

A.3 Range and consistency constraints

Each balance is decomposed into four 16-bit limbs, each constrained by a lookup into a table committed at genesis:

$$b = \sum_{i=0}^3 b_i 2^{16i}, b_i \in [0, 2^{16})$$

A tree containing one out-of-range leaf fails proof generation for the epoch. There is no bypass path.

A.4 Transition circuit

Public inputs per batch:

$$(C_{\varepsilon-1}, C_{\varepsilon}, [s_{lo}, s_{hi}], h_{rules})$$

Per-batch constraint counts on the reference build:

$$2^{18} \leq c_{batch} \leq 2^{21}$$

Batches above the ceiling split. The ceiling is a prover performance parameter, not a throughput claim.

A.5 FRI configuration and cost

FRI runs at blowup factor 8 with folding factor 8, query count , and grinding difficulty bits, targeting conjectured soundness error

$$\epsilon_{sound} \leq 2^{-100}$$

Measured on the reference machine:

$$t_{verify}^{batch} < 50ms, t_{verify}^{epoch} < 400ms, 150KB \leq |\pi_{batch}| \leq 320KB$$

A circuit release that moves verification cost by more than

$$\frac{\Delta t_{verify}}{t_{verify}} > 0.25$$

requires a new reference measurement before activation.

A.6 Recursion and the epoch root

Recursion tree arity and depth:

$$a=8, \delta_{max}=3 \Rightarrow a^{\delta_{max}}=512 \text{ batches per epoch proof}$$

The epoch artifact a third party verifies is the triple . $(\pi_{\varepsilon}, C_{\varepsilon-1}, C_{\varepsilon})$

A.7 Epoch boundary

$$n_{events} = 2^{22} \text{ or } \Delta t = 86,400s$$

An epoch closes at the first of:

Both triggers are adjustable within the governed boundary parameters defined in Ch.10.