TICKER/YARD

Markets, routed.

PROTOCOL WHITE PAPER

Asset Mobility and Protocol Participation

Neutral routing, controlled representations, and user-operated protocol work

AUTHOR	@jeronxd3
VERSION	v1.1
STATUS	V1.1 design candidate; prepared for review
PREPARED DATE	9 August 2026

Neutral routes. Portable operators. Funded work.

Abstract

Blockchain liquidity is distributed across many execution environments, but moving an asset between them still forces users and applications to reason about bridges, representations, gas, liquidity, route quality, and incompatible security assumptions. TickerYard presents that complexity through one cross-chain quote and execution interface.

Phase I discovers eligible paths from interoperability and liquidity providers, validates them against the user's exact asset intent, and ranks them under a disclosed policy. Its first specialized vertical is tokenized equities, explored through certificate-mirror and fungible wrapping models. Phase II proposes a canonical-vault and representation network through which an eligible asset may be locked on its origin network and represented across supported destinations.

The contribution is a shared control and accounting layer for route selection, backing custody, representation issuance, remote transfer, and redemption. Every outstanding representation, unused issuance authorization, and pending redemption is intended to remain covered by eligible canonical backing. External routes retain their providers' trust assumptions; native TickerYard rails introduce vault, registry, messaging, adapter, operator, and governance risks.

The Yardkeepers Participation Network adds a separate user-operated coordination layer. Anvil creates the sole canonical $YARD against a fixed-genesis collection of 3,333 Yardkeeper NFTs. Phase 1 routes all NFTs through the Anvil AMM with no whitelist or direct allocation; the factory assigns 95% of supply to AMM capacity, 5% to bounded NFT-collateralized loans, and 0% to its optional reward bucket. Any activation tier may participate in Anvil's separate owner-wallet fee stream, but operator eligibility begins only at Tier 4 and requires a separate zero-value enrollment of a local, user-funded runner. Every NFT owns a canonical token-bound account.

TickerYard is the proposed network's first protocol client. Its initial canary attributes one narrow certificate-completion action through an endpoint-specific adapter while the underlying endpoint remains permissionless. Paid production begins at Level 0 with a refundable base bond and no operating history; quality-adjusted assigned work, elapsed time, reliability, and client-specific evidence may unlock higher levels, while raw notional volume does not. A future external client may opt in only by authorizing an isolated adapter, defining protocol-specific qualification and assignment, and pre-funding its maximum job liabilities. Yardkeeper ownership does not create universal execution authority, guarantee jobs or earnings, or inherit a client protocol's native NFT, token, or holder-revenue rights. Work compensation follows objective receipts; client-native holder programs remain controlled by that client. A mature TickerYard-only equal-seat epoch exists only after positive settled distributable revenue, reserve clearance, exact funding, audit, and legal approval. $YARD itself receives no TickerYard protocol funds.

TickerYard remains a partner-integration prototype, not an unrestricted public protocol. The generalized asset network is proposed. Keeper runner, collection, account, registry, executor, and seeder source now provide implementation evidence, but there is no public deployment, independent audit, populated signed manifest, final 3,333-piece art release, or legally cleared value flow.

Keywords: cross-chain routing; asset mobility; tokenized equities; protocol participation; local operators; verifiable work; interoperability.

Document status and scope

V1.1 extends the design with a protocol-neutral participation layer; it does not announce an external integration. Status labels and the canonical product specifications control. It is not an audit, legal opinion, offering document, MiCA filing, investment recommendation, job promise, or return promise.

Contents

1 Scope, status, and contribution

TickerYard's long-term thesis is that an application should be able to request a supported asset on a supported destination network without selecting the underlying bridge or representation mechanism. The system should determine which valid path can satisfy that request and expose the path's cost, expected time, and trust assumptions before execution.

That thesis spans three materially different maturity levels. They must not be read as a single deployed system.

TickerYard's intended contribution has three parts:

1. **A route abstraction.** The user expresses an exact source and destination asset intent. TickerYard discovers and compares eligible external or native paths without silently changing that intent.
2. **An asset abstraction.** When no acceptable canonical or issuer-supported path exists, a proposed TickerYard rail can lock eligible backing and issue a controlled representation, subject to explicit accounting and trust boundaries.
3. **A participation abstraction.** A protocol client defines one objective action, isolates it behind an exact adapter, and funds its liabilities; a separately qualified NFT-authorized local runner can perform it while public or client-controlled fallback remains available. TickerYard is the first proposed client, not the universal owner of future client jobs.

The protocol is deliberately narrower than the slogan "any asset, any chain." Assets and networks are supported only when they pass technical, liquidity, operational, issuer, and legal admission policies.

Table 1. Product and protocol status at the v1.1 preparation snapshot

Layer	Status	Evidence and next release gate
Cross-chain routing UI and API	**Partner prototype**	**Current:** TickerYard-owned quote boundary, multiple upstream route sources, route validation, wallet execution, and activity tracking for the supported graph. Browser execution remains narrower than discovery. **Next:** production credentials, broader execution, durable canonical tracking, operating controls, and public release evidence.
Tokenized-equity rail	**Reference integration**	**Current:** reviewed certificate-mirror and fungible-wrapper design for Robinhood Chain and Arbitrum using CCIP messages and application verification. **Next:** contract remediation, audit, full lifecycle evidence, separated roles, recovery, issuer and legal approval, and a signed deployment manifest.
Yardkeepers participation network	**Pre-deployment implementation**	**Current:** locally tested runner and production-shaped contract source, including signed-release verification, exact runner acceptance, endpoint-specific submission, restart recovery, and a separately gated checkpointed TickerYard revenue distributor; local and Robinhood-fork contract compatibility tests; focused revenue-distributor unit/fuzz tests; 288-composite internal art-expansion pilot. TickerYard is the first proposed protocol client. Nothing is publicly deployed or audited, the production service is not enabled in the default desktop build, no revenue source or distributor authority is configured, no external client adapter or allowlist is approved, and the 3,333-piece art/rarity/rights/provenance release is not frozen. **Next:** signed manifest and art freeze, production desktop composition, legal clearance, public testnet, independent audit, signed desktop release, then one capped TickerYard-client job with current-owner Anvil Tier 4, zero-value Keeper Enrollment, user-funded gas, and zero TickerYard work reward.
Universal Ticker assets	**Protocol proposal**	**Current:** the vault, registry, representation, accounting, and adapter model defined in this paper. **Next:** a formal state machine, implementation, testing, audits, governance, operators, chain adapters, legal perimeter, and staged deployment.

2 Problem and design goals

An asset can be liquid on one network while its users or applications operate on another. Existing solutions generally fall into four categories:

- **Canonical or issuer-supported bridges**, which often provide the clearest representation but cover only selected assets and networks.
- **Third-party bridges and liquidity networks**, whose coverage, execution model, price, latency, and security assumptions differ.
- **Issuer-integrated omnichain standards**, which can preserve a unified supply but require deployments, configuration, and ongoing participation by the token project or an authorized adapter operator.
- **Third-party wrapped assets**, which can expand access but introduce separate custody, backing, liquidity, governance, and redemption assumptions.

The result is not only a discovery problem. It is also an intent, accounting, and risk-comparison problem. Two routes that end with tokens sharing a ticker may deliver different contracts, rights, liquidity, or redemption paths.

2.1 Design goals

TickerYard is designed around seven goals:

1. **Exact intent.** A route must preserve the requested canonical asset and permitted destination representation.
2. **Disclosed optimization.** "Best" means best among eligible routes under a visible user or application policy, not an unexplained global ranking.
3. **Solvency before reach.** New networks and assets must not weaken backing or replay-prevention invariants.
4. **Route-specific risk.** External and native routes expose their own dependencies rather than inheriting a generic "secure" label.
5. **Recoverable operation.** Pauses, expiry, retries, and bounded recovery are protocol requirements, not only interface states.
6. **Progressive decentralization with explicit authority.** Every role that can affect custody, issuance, routing, or recovery must be enumerated and observable.
7. **Client isolation.** Retail participation must never become universal execution authority; every protocol client retains its own adapter, qualification, assignment, funding, pause, and fallback boundary.

2.2 Non-goals

TickerYard does not make every asset legally transferable, remove issuer controls, eliminate cross-chain finality, guarantee liquidity or execution time, or make third-party bridge risk disappear. It does not treat tickers as identity, and it does not promise that every chain pair will have a direct native lane. The participation layer is not an arbitrary-code marketplace, a guarantee that an NFT will receive work, a promise that a home operator can win latency-sensitive races, or an entitlement to a client protocol's fees or holder program.

2.3 Publication and legal perimeter

This document is a technical protocol design paper. It is not, by that label alone, the regulatory crypto-asset white paper that may be required for a public $YARD offer or admission to trading in the European Union. The repository contains no evidence of a completed classification opinion, identified offeror and home Member State, CNMV notification, machine-readable regulatory filing, authorized service-provider perimeter, or legally cleared public offer.

If MiCA Title II applies, the notification package must reach the competent home-state authority at least 20 working days before publication, and the public document and marketing must follow the applicable notification and publication controls [11, 12]. A series of 3,333 tokens is not outside MiCA merely because each token is called an NFT; classification follows the final rights and economic substance. Tokenized securities, transferable revenue rights, market operation, custody, routing, promotions, and services can introduce separate MiFID, CASP, consumer, tax, sanctions, and national-law questions.

Calling the NFT an operating seat does not make paid work or income certain. Any public communication must distinguish present software access from unapproved future client opportunities, compensation for completed work from passive holder rights, and protocol-funded liabilities from TickerYard revenue. A transferable pay-to-work narrative, undisclosed operator costs, or advertised dual-protocol yield would require separate legal, consumer, tax, and accounting review.

Legal release boundary

V1.1 may be published only as non-offer protocol design material. No public $YARD or Yardkeepers offer, Anvil market activation, admission to trading, revenue distribution, operator-opportunity campaign, or unrestricted tokenized-security route is authorized by this paper. Qualified counsel, the exact legal entities and jurisdictions, partner/issuer rights, applicable filings and waiting periods, and any required authorized service providers remain launch gates.

Yardkeepers public phase boundary

Phase 1 is the NFT + Anvil launch. The collection contract first performs the fixed-genesis mint of 3,333 Yardkeepers into its distribution vault because Anvil requires an existing collection address. Public distribution is nevertheless Anvil-only: direct release is closed, Anvil creates the sole canonical $YARD and market, and all 3,333 NFTs are seeded to the Anvil AMM before opening, with no whitelist or reserved allocation. The two launch LP positions are permanently locked; their exact composition and ranges belong to the separate launch communication and signed manifest. Disclosed activation, borrowing, fee-route, and owner-wallet reward mechanics may open, but they grant no Keeper runner authority. **Phase 1.5 is the separately signed capped TickerYard-client canary.** Any deployed runner registry remains paused until that release explicitly opens zero-value Keeper Enrollment, the signed local runner, one bounded TickerYard job, and work receipts. Phase 1 activation is therefore a market state, not a live Keeper seat, and Phase 1.5 is not evidence of an external client integration.

3 Terminology and system assumptions

Table 2. Core terminology

Term	Meaning in this paper
Canonical asset	The specific asset whose economic or protocol rights are being represented, identified by an origin namespace and canonical identifier rather than a ticker.
Origin network	The network on which eligible backing is locked or otherwise controlled. It may differ from a venue where a similar asset trades.
Representation	A contract or ledger entry on a supported destination network whose issuance and redemption are tied to the canonical asset.
Route	A complete, executable path from a source asset-on-network to an allowed destination asset-on-network.
Lane	A configured directional relationship between two protocol endpoints for a particular message and asset policy.
Adapter	Network- or asset-specific logic that normalizes custody, identifiers, decimals, finality, and transfer behavior behind a common interface.
Finality policy	The source confirmation rule that must be satisfied before a cross-chain authorization is accepted.
Issuance credit	A unique, amount-bounded authorization created by a finalized deposit or remote burn and consumed exactly once by destination issuance.
Redemption obligation	A unique, amount-bounded obligation created by a finalized representation burn and consumed by canonical release.

Participation terminology is deliberately separate from asset backing:

- **Keeper NFT.** A fixed-genesis portable participation seat whose current ownership epoch can authorize one local runner. It is eligibility infrastructure, not a universal credential, employment promise, or client-revenue right. Its canonical token-bound account may receive source-specific funded payments under separately approved rules. No TickerYard payout exists in the MVP.
- **Keeper account.** A deterministic ERC-6551-compatible account controlled by the current NFT owner. Assets left in it travel with the NFT; an ordinary balance is withdrawable and is not permanent backing.
- **Ownership epoch.** A monotonic NFT state that changes on transfer and immediately invalidates the prior activation and runner.
- **Keeper Enrollment.** The zero-value runner-binding step available only to a current owner whose NFT is at Anvil Tier 4 (onchain index 4) for the current ownership epoch. It is not a second activation or burn.
- **Runner.** A low-value EOA generated and stored on the NFT holder's computer. It holds only user-funded native gas and can call one narrow executor.
- **Protocol client.** TickerYard or a future external protocol that defines a bounded job, authorizes an isolated adapter, specifies qualification and assignment, funds maximum liabilities, and retains its own pause and fallback authority.
- **Client adapter.** A protocol-specific capability wrapper pinned to an exact chain, target, selector, code identity, job schema, value limit, and postcondition. One client adapter grants no authority over another client.

- **Job manifest.** A signed, versioned commitment to the client, adapter, capability, funding, assignment, gas policy, expiry, fallback, and trust roots used by the desktop runner.
- **Operator qualification.** Current-owner and current-runner evidence for one client capability. Transferable NFT provenance may inform review, but competence, reliability, and higher-trust permissions are re-evaluated after an ownership-epoch change.
- **Operator level.** A current-ownership-epoch quality tier derived from pre-assigned completed work, elapsed time, reliability, challenge history, and capability evidence. It is an input to client admission, not universal authorization.
- **Operator bond.** A refundable, separately accounted commitment required only for funded production participation or a higher-risk capability. It does not buy reputation or fund rewards and can be slashed only under predeclared objective rules with bounded exposure and appeal.
- **$YARD.** The fixed-supply Anvil collection token. Its exact supply and per-NFT quote belong to the separate launch-economics communication and signed deployment manifest. It supports Anvil activation, bounded NFT-collateralized borrowing, later objective bonds, pairing, and buyback demand. It has no TickerYard protocol-revenue right and is never canonical backing.
- **Work receipt.** An append-only event proving that the authorized runner completed the exact client, adapter, job, and objective postcondition.
- **Capability lane.** A separately qualified, client- and endpoint-specific class of Keeper work. Tier 4 opens eligibility to qualify but does not itself prove technical competence, independence, price quality, capital, or client approval.
- **TickerYard revenue epoch.** A TickerYard-source-policy-bound, fully funded settlement-asset pot with an immutable equal-seat eligibility root, claim period, canonical-TBA recipient, and same-asset rollover rule. It is not a default component of external client work and is zero when no fresh distributable TickerYard revenue is recognized.
- **Sponsor Train.** A fully funded campaign for verified user, conductor, keeper, or integration outcomes; its pass-through budgets are not protocol revenue.

TickerYard assumes that supported networks offer sufficiently deterministic transaction and event verification for their configured finality policy; that adapters can identify the actual amount received or destroyed; and that an asset's transfer, freeze, rebase, fee, and administrative behavior is understood before listing. Assets that violate those assumptions require a dedicated adapter or are not eligible.

4 Protocol overview

The user-facing abstraction is a quote-and-execution request. Behind it, TickerYard separates a data plane that discovers and executes routes from a control plane that defines what may be routed.

4.1 Core components

- **Intent validator.** Resolves asset identities and rejects routes that substitute an unapproved contract, network, recipient, or representation.

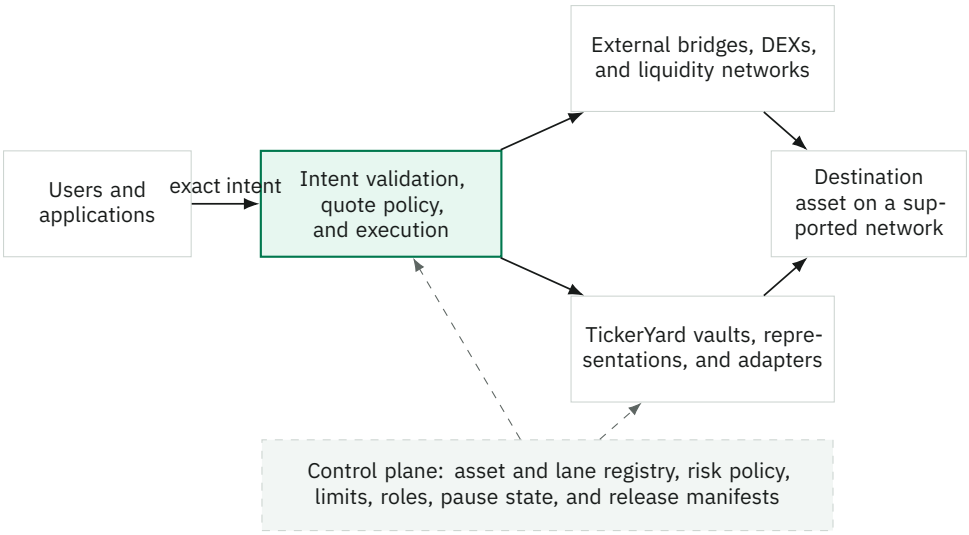

Figure 1. TickerYard separates route selection from the providers or native rails that settle a route. The control plane constrains both.

- **Quote and policy engine.** Queries capable providers, normalizes responses, applies hard eligibility constraints, and ranks the remaining routes.
- **Execution adapters.** Construct the exact approval, swap, bridge, deposit, burn, mint, or release actions required by the selected path.
- **Activity observer.** Tracks source confirmation, provider or message progress, destination execution, and terminal failure or recovery states.
- **Asset and lane registry.** Maps canonical assets to representations and records endpoints, code identities, peers, limits, finality, roles, and enablement.
- **Native asset rail.** A proposed set of canonical vaults, issuance and redemption endpoints, replay protection, message verification, and chain adapters.
- **Yardkeepers participation.** A fixed NFT collection, pinned Anvil Tier-4 eligibility adapter, token-free local-runner registry, low-value runner, signed job manifest, and isolated client executor that can attribute narrow work without becoming a custody or liveness monopoly. TickerYard supplies the first proposed client capability; later clients remain separately authorized.

No off-chain quote response is itself authority to mint or release assets. Native settlement authority must be derived from finalized on-chain state and accepted through audited on-chain verification rules. Likewise, open-source keeper software or a hosted API is not authorization: recognized work requires the current NFT owner, ownership epoch, Anvil Tier 4, zero-value Keeper Enrollment, runner, signed client manifest, immutable client adapter, assignment, and objective postcondition to agree onchain.

5 Yardkeepers protocol participation network

The proposed Yardkeepers Participation Network is a distribution, qualification, and execution layer through which individuals can perform narrowly authorized work for protocol clients from their own computers. It is not one universal keeper contract controlling every client. Each client exposes only a separately reviewed capability, retains its target and pause authority, chooses its qualification and assignment policy,

and funds the maximum liabilities attached to its jobs.

The network separates portable participation infrastructure from client-specific permission. A Yardkeeper can carry a current ownership epoch, runner binding, public work provenance, and supported desktop tooling across integrations. The client still decides whether that evidence is sufficient for a particular capability. A transfer preserves public NFT provenance but invalidates the prior runner and requires current-owner requalification; higher-trust permissions and reliability claims cannot be sold merely by transferring the NFT.

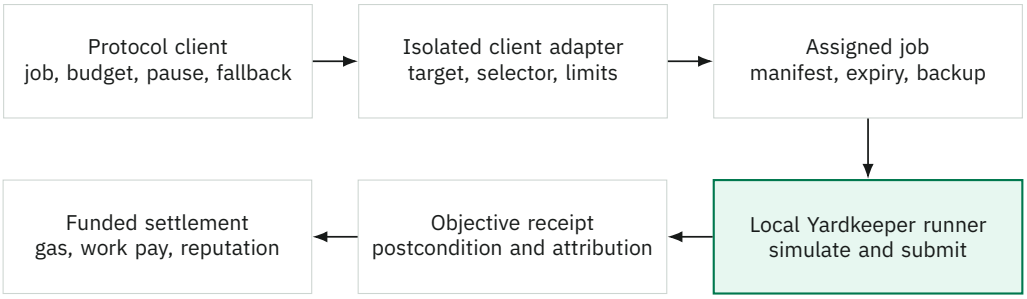

Figure 2. A protocol client authorizes and funds one isolated capability. Yardkeepers supplies qualified local operators; objective receipts settle work without granting cross-client authority.

5.1 TickerYard as the first protocol client

TickerYard provides the first proposed demand-side job and the first adapter implementation. The capped canary asks an enrolled current-owner runner to call the pinned certificate-vault `execute(messageId)` function only after source, message, attestation, delay, ownership, runner, target-code, and postcondition checks pass. The vault endpoint remains directly permissionless, so Yardkeeper authorization controls recognized attribution rather than bridge liveness.

Using TickerYard as the first client permits the runner, signed-manifest, assignment, simulation, restart, receipt, and fallback boundaries to be tested without presenting an external protocol as committed. It does not prove independent demand. The canary therefore pays no TickerYard work reward; a later funded TickerYard job must still demonstrate positive unit economics and exact maximum-liability funding.

5.2 Retail operator path

Participation should progress from low-risk observation to separately qualified automation. The desktop must show authority, expected cost, funding source, assignment, failure behavior, and maximum exposure before a user enables a valuable capability.

No mode guarantees a job, profit, minimum utilization, or reimbursement unless the assigned job's funded terms say so. Low-latency races, liquidations, unconstrained solver activity, custody, and arbitrary target calls are outside the retail starting boundary.

5.3 Bonded progression and client access

The zero-value TickerYard canary remains bond-free because it is test evidence rather than paid production work. When funded work opens, every newly enrolled current ownership epoch starts at Level 0 with no operational track record. Entry

Table 3. Progressive retail operator modes

Mode	Operator interaction	Authority and evidence boundary
Observer	Monitor eligible public state and receive alerts	Read-only; no runner transaction, job claim, or compensation
Assisted	Review one simulation, cost, assignment, and exact call before approving	One quoted action under a short-lived manifest; no unattended authority
Local automated	Enable one signed client capability on a low-balance runner	Exact target, selector, value, gas, expiry, pause, and postcondition; every success produces a receipt
Advanced	Qualify for higher-value or latency-sensitive client work	Client-specific history, capital, bond, independence, or uptime requirements; never inferred from NFT ownership alone

may require a separately refundable base $YARD bond. The bond demonstrates bounded commitment and supports objective remedies; it does not buy reputation, finance rewards, guarantee assignments, or convert token value into competence.

Table 4. Proposed operator progression

Level	Minimum evidence	Candidate access boundary
0 — Starter	Current enrollment, accepted runner, base bond, and no unresolved prior-epoch liability	TickerYard and specifically approved low-exposure lanes; assisted or tightly capped local automation
1 — Proven	Minimum active time and quality-adjusted assigned completions without unresolved challenge	Low-risk external client lanes that explicitly accept Level 1; bounded automation and newcomer allocation
2 — Established	Sustained success, multiple job types or clients, reliable primary/backup behavior, and any capability bond	Moderate-volume or more frequent client work under higher caps and client-specific qualification
3 — Advanced	Long-duration current-epoch record, operational redundancy, incident performance, and any required identity, capital, or independence evidence	High-volume or higher-trust lanes that expressly accept Level 3; not presumed suitable for every retail operator

Reputation is quality-adjusted work evidence, not trading volume. For ownership

epoch e, a candidate score can take the form

$$R_e = \sum_{j \in \mathcal{J}_e} \min(w_j, w_{\max})\, q_j - P_e, \tag{1}$$

where $\mathcal{J}_e$ contains only pre-assigned, funded, finalized jobs; w_j is a difficulty and risk weight committed before assignment; $q_j \in [0, 1]$ measures objective completion quality such as correct postcondition, timeliness, and operator-caused failure; and P_e contains finalized penalties. Per-client, per-capability, and per-period contribution caps prevent one sponsor or a stream of trivial jobs from manufacturing a tier. Notional transaction value, token balance, fees paid, self-dealing, unassigned calls, reverted spam, and related-party wash activity contribute no reputation.

Promotion also requires minimum elapsed time, a challenge-free window, and the evidence declared for the target level. A network level is a portable recommendation to a client, not an automatic allowlist. Each client maps its own lanes to minimum levels and may require additional review. Missing an assignment may lower reliability and activate a backup; bond slashing is reserved for an objectively proven, predeclared violation with evidence, review, appeal, and a bounded maximum.

The score and operational level bind to the current NFT ownership epoch. A same-owner runner rotation can preserve them after a verified handoff, but an NFT transfer closes the old epoch and the new owner starts at Level 0. Lifetime NFT work provenance may remain visible for art and historical context, yet it cannot transfer a prior operator's client access. Any future mechanism that lets one human move current reputation across seats requires a separately verified identity and anti-Sybil policy.

5.4 External client admission and economic separation

A future external protocol may participate only after it intentionally authorizes the integration. The minimum admission sequence is:

1. define a bounded action with an objective on-chain postcondition and a public, client-operated, or independently operated fallback;
2. deploy or approve a client-isolated adapter with pinned chain, target, selector, code identity, value and rate limits, assignment rules, and pause authority;
3. pre-fund maximum gas, retry, work-payment, refund, and challenge liabilities in the agreed settlement asset;
4. publish a signed manifest and reviewed desktop capability that fails closed on code, role, funding, state, or trust-root drift; and
5. complete technical, operational, economic, legal, and incident-response review before any valuable mainnet job opens.

Assignment must prevent a retail gas race. A primary operator, deterministic rotation, reserved newcomer allocation, and time-bounded backups are preferred to an unbounded first-transaction-wins contest. If open execution is required, non-winning operators must not be induced to spend unreimbursed gas under an advertised earnings opportunity.

Client-native tokens, NFTs, protocol fees, and holder programs remain under that client's contracts and terms. Allowlisting a Yardkeeper adapter creates no automatic right to those balances. A person may independently hold both a Yardkeeper and a client credential. Each right and payment remains separately sourced, accounted, disclosed, and legally assessed. Better operator coverage may improve liveness and

support client usage; it does not guarantee client fee volume, holder distributions, or Yardkeeper income.

The architecture deliberately avoids one cross-protocol super-adapter. A defect or revocation in one client lane must not authorize calls, consume funds, alter reputation, or pause work for another client except through an explicit network-wide emergency release decision.

6 Phase I: cross-chain routing and execution

Phase I turns bridge selection into a constrained routing problem. A request contains:

- source network, canonical source asset, and amount;
- destination network and permitted destination representation;
- recipient;
- slippage or minimum-output policy where swaps are involved;
- preferences such as output, time, or a permitted risk tier.

6.1 Eligibility before ranking

Let $\mathcal{R}$ be all discovered route candidates. TickerYard first constructs the eligible set

$$\mathcal{R}_{\text{valid}} = \left\{ r \in \mathcal{R} \;\middle|\; \begin{array}{l} \text{asset identity and destination contract match the request,} \\ \text{endpoints and dependencies pass current health policy,} \\ \text{liquidity, amount, finality, and lane limits are satisfied,} \\ \text{and the route fits the selected risk policy} \end{array} \right\}. \tag{2}$$

Only then does a policy choose

$$r^* = \arg\max_{r \in \mathcal{R}_{\text{valid}}} U(\text{protected output},\ \text{estimated time},\ \text{reliability};\ \text{preferences}). \tag{3}$$

Security is not hidden inside one unexplained score. A route must meet a minimum risk policy, and its remaining dependencies are disclosed separately. Quotes are time-bound estimates; network gas, liquidity, provider state, and execution price can change before submission.

At the repository snapshot supporting v1.1, the prototype's deterministic order is protected minimum output first, then estimated completion time, then stable route identity. Richer user-weighted policies are proposed behavior, not a current claim of security or historical-reliability scoring.

6.2 Execution lifecycle

1. **Discover.** Query only sources capable of satisfying the exact request.
2. **Normalize and validate.** Parse each response into a common model, verify contract and chain identities, and reject conflicts or incomplete actions.

3. **Rank and disclose.** Present protected output, expected costs, estimated time, required approvals, provider provenance, and material trust assumptions.
4. **Preflight.** Revalidate balances, allowance, recipient, chain, route health, quote freshness, and the exact transaction immediately before signature.
5. **Execute.** The wallet submits the source actions. TickerYard does not treat source confirmation as cross-chain completion.
6. **Observe.** Track provider or message identifiers until destination finality, explicit failure, expiry, or recovery.

An external route remains an interaction with its underlying bridge, liquidity, messaging, and contract system. TickerYard can improve discovery, validation, and evidence, but cannot replace those dependencies with its own assurance.

Current composed stock journeys are assisted and resumable, not atomic or gasless. A user may sign separate acquire, approval, wrap, and delivery actions; the reference native rail requires source-network gas and a quoted CCIP fee, while a later destination execution must also be funded. A completed early step does not guarantee completion or refund of the remaining cross-domain journey.

6.3 Failure behavior

A production route adapter must define at least these states: quoted, awaiting approval, submitted, source-confirmed, in-flight, destination-executed, failed, expired, recoverable, refund-submitted, and refunded. Recovery must never depend solely on a browser tab remaining open. Each provider integration must document who can retry or refund, the maximum wait before intervention, and the evidence required to distinguish delay from terminal failure.

The routing threat model also includes stale or malicious provider calldata, wrong transaction targets or approval spenders, recipient substitution, unlimited allowances, quote expiry, slippage and MEV, RPC divergence, frontend or configuration compromise, and partial provider failure. Production adapters must allowlist permitted targets and selectors, bind recipient and asset intent, minimize approvals, re-quote and simulate immediately before signing, and reject any response that changes the requested destination.

7 Initial asset vertical: tokenized equities

Tokenized equities are a useful but demanding test of the asset-mobility thesis. They combine cross-chain engineering with issuer controls, market structure, corporate actions, identity restrictions, and jurisdiction-specific rights. A token or certificate representation is not automatically equivalent to direct registered ownership of the referenced security.

7.1 Two reference models

1. **Certificate mirror.** A specific certificate NFT is escrowed on its origin network. A unique remote mirror records the corresponding position. Returning the mirror burns it before the original certificate can be released. The mirror is not independently redeemable with an issuer; its protocol redemption path is the return-and-release operation.

2. **Fungible stock wrapper.** Eligible fungible stock tokens are escrowed on the origin network and a one-to-one remote representation is issued, net of any disclosed in-kind fee. Remote burning authorizes release of the corresponding eligible backing, subject to lane accounting and issuer transfer rules.

In the reviewed reference architecture, the underlying stock asset remains on Robinhood Chain while remote representations are created on Arbitrum. Chainlink CCIP transports application messages; the application contracts control escrow, mint, burn, and release [1, 2]. The design also adds peer verification, two application-level attestations, and an execution delay. These layers create additional dependencies and do not constitute an independent audit.

The reviewed contract version has a material control weakness: a privileged owner can disable and replace a peer without the intended replacement delay, can replace both application attesters, and can reduce the execution delay; the reviewed owner and guardian roles are also concentrated. The production design must remove the peer re-enable bypass, enforce minimum delays for safety-increasing changes, separate roles across independently controlled multisignatures or equivalent controls, and cap exposure before any broader launch.

> **Current boundary**
> The reference integration is suitable for partner testing, not an unrestricted public launch. Known release work includes contract remediation, independent audit, separated production roles, complete outbound-and-return lifecycle evidence, user recovery, rate limits, issuer and legal approval, and a signed manifest of every production endpoint and authority.

7.2 Rights and restrictions

Before a tokenized-equity asset can be supported, its registry entry and user disclosure must identify:

- the issuer, custodian, and contract administrator, if any;
- the legal and economic rights of the canonical asset and each representation;
- who is eligible to acquire, transfer, redeem, or hold it;
- transfer restrictions, sanctions controls, freezes, administrative burns, and upgrade powers;
- treatment of dividends, splits, mergers, voting, delistings, and market closures;
- redemption venue, timing, fees, minimums, and insolvency treatment; and
- the jurisdictions in which the route may be offered.

TickerYard must not describe a mirror or wrapper as the underlying equity unless the issuer's legal documentation establishes that equivalence. A technically solvent wrapper can still fail to deliver expected economic rights or remain transferable under an issuer action.

8 Phase II: the Ticker asset network

Phase II generalizes the lock-and-represent pattern into a proposed network for eligible assets that were not originally designed to move across all destination networks.

Consider AAVE on Ethereum as an illustrative asset, not a launch commitment:

1. The user deposits AAVE into an approved canonical vault on Ethereum.
2. After the configured finality policy, the deposit creates a unique issuance credit.
3. A destination endpoint consumes that credit exactly once and issues the approved Ticker representation on the requested network.
4. A remote transfer burns the representation on its source network, creates a new destination-specific issuance credit, and issues the equivalent amount on the next network.
5. Redemption burns the remote representation, creates a redemption obligation, and releases the corresponding AAVE only on the canonical vault's network.

Figure 3 summarizes the three conservation paths. Every destination issuance consumes a finalized, amount-bounded credit; every canonical release consumes a finalized redemption obligation.

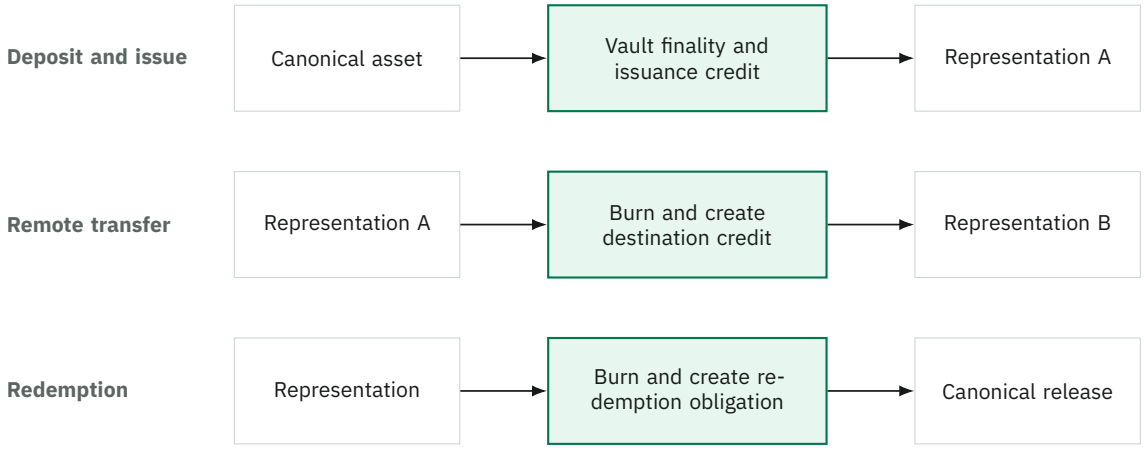

Figure 3. Proposed native-rail lifecycle. Deposit and issuance, remote transfer, and canonical redemption are separate replay-protected state transitions.

The illustrative label "TY-AAVE" is a display convention, not an identity. Wallets and applications must resolve the representation through the asset registry.

8.1 Canonical asset identifiers

Tickers are neither unique nor immutable. A canonical asset identifier should be derived from a versioned tuple such as

$$\text{AssetId} = H(\text{namespace},\ \text{networkReference},\ \text{canonicalIdentifier},\ \text{assetStandard},\ \text{registryVersion}). \tag{4}$$

For an EVM ERC-20 [5], the network reference is the chain ID and the canonical identifier is the contract address. A non-EVM adapter substitutes that ecosystem's unambiguous network and asset identifiers. The registry also records decimals, transfer behavior, issuer controls, canonical vault, approved representations, and code identities.

8.2 Required protocol components

Table 5. Proposed native-rail components and authority boundaries

Component	Required behavior	Must not do
Canonical vault	Measure eligible deposits, segregate backing, create redemption releases, and expose reconciliable balances.	Issue remote assets, execute arbitrary calls, or spend backing as treasury capital.
Asset registry	Bind asset IDs to canonical and remote contracts, adapters, code identities, finality, limits, and status.	Treat a ticker, operator submission, or nonzero peer as proof of identity.
Message verifier	Accept finalized, authenticated protocol events and create unique credits or obligations.	Mint or release without replay-protected, amount-bounded state.
Representation endpoint	Consume issuance credits, mint and burn approved representations, and expose global accounting inputs.	Change the canonical asset or select an arbitrary vault.
Chain adapter	Normalize identifiers, decimals, received amounts, finality, and transaction semantics.	Assume ERC-20 behavior for incompatible assets.
Observer and reconciler	Independently compare backing, supply, pending state, roles, limits, and message progress.	Hold unilateral custody or make a failed reconciliation look healthy.

Native assets, rebasing tokens, fee-on-transfer tokens, restricted real-world assets, and assets from non-EVM networks require explicit adapter and accounting policies. The phrase “wrap anything” therefore means “support new asset classes through reviewed adapters,” not permissionless listing of arbitrary contracts.

9 Accounting and state invariants

For canonical asset a at time t, define:

- $B_a(t)$: eligible, reconciled backing held by the canonical vault;
- $I_a(t)$: total outstanding Ticker representations across all networks;
- $C_a(t)$: total unconsumed issuance credits created by finalized deposits or remote burns; and
- $R_a(t)$: total pending redemption obligations created by finalized representation burns.

The core solvency invariant is

$$I_a(t) + C_a(t) + R_a(t) \leq B_a(t). \tag{5}$$

This formulation covers in-flight state. A deposit increases B_a and C_a; destination issuance decreases C_a and increases I_a. A remote transfer decreases I_a on the source

and creates C_a for the destination; issuance reverses those changes. A redemption burn decreases I_a and creates R_a; canonical release decreases both R_a and B_a.

Every transition is keyed by a unique message or operation identifier and can be consumed once. A failed or expired credit cannot be both refunded and issued. Dust, fees, quarantined deposits, or non-eligible balances are accounted for separately and cannot be counted as backing.

Each destination also has a lane limit $L_{a,n}$:

$$I_{a,n}(t) + C_{a,n}(t) \leq L_{a,n}(t). \tag{6}$$

Lane limits bound exposure but do not replace the global invariant. Production deployments require on-chain invariant tests, cross-chain state-machine tests, independent reconciliation, and alerts that fail closed when data sources disagree.

Table 6. Conservation effects of native-rail transitions

Transition	ΔB	ΔI	$\Delta C / \Delta R$	Precondition
Finalized deposit	$+x$	0	$\Delta C = +x$	Actual eligible amount received and final
Destination issue	0	$+x$	$\Delta C = -x$	Matching unused issuance credit
Remote burn	0	$-x$	$\Delta C = +x$	Finalized burn and destination limit
Redemption burn	0	$-x$	$\Delta R = +x$	Finalized burn addressed to canonical vault
Canonical release	$-x$	0	$\Delta R = -x$	Matching unused redemption obligation

10 Security and trust model

Cross-chain systems compose multiple security domains. TickerYard must name them per route.

10.1 Required controls

Before a native rail can hold user value, its design must include:

- source finality policies that account for reorganizations and chain-specific failure modes;
- domain-separated messages, exact peer identities, nonces, replay protection, and one-time credit consumption;
- amount and rate limits at asset, lane, epoch, and operation levels;
- immediate emergency disable but delayed, independently reviewed re-enable or peer replacement;
- separated custody, pause, configuration, attestation, and recovery roles, preferably through hardware-backed multisignature or equivalent controls;

Table 7. Route-level trust boundaries

Route type	Primary dependencies	TickerYard responsibility
Canonical or issuer path	Issuer contracts, administrators, messaging, and destination deployment	Validate identity and status; disclose issuer and route controls
External bridge or liquidity route	Provider contracts, liquidity, relayers or validators, pricing, recovery, and destination asset	Validate exact intent and executable response; preserve provenance; monitor completion
TickerYard native rail	Vaults, registry, representations, message verification, adapters, roles, limits, observers, and backing	Specify, implement, audit, operate, reconcile, and disclose the full stack
Composite route	Every swap, bridge, token, approval, and intermediary above	Validate the entire action graph and protected outcome, not only the first step

- minimum execution delays where they create a meaningful intervention window, without presenting delay alone as security;
- explicit expiry, retry, cancel, refund, and user-exit state machines that cannot produce double satisfaction;
- continuous reconciliation of backing, issued supply, pending credits and obligations, lane exposure, roles, and code identities;
- versioned deployments, public manifests, independent audits, bug bounties, incident runbooks, and capped launches; and
- fail-closed behavior when providers, RPCs, observers, or accounting views disagree.

Participation adapters add a separate control set. Each client lane must pin its target, selector, code identity, value and rate limits, funding source, assignment, expiry, postcondition, and pause authority. The local runner must use a low-balance key that cannot control the NFT, its token-bound account, user principal, another client adapter, or protocol administration. A client job must be reviewed as potentially hostile input: arbitrary code download, mutable call data, concealed approvals, delegate calls, unrestricted batching, and silent manifest updates fail closed. Work attribution must bind the client, capability, job, ownership epoch, runner, and objective result, while cross-client reputation remains advisory rather than execution authority. Reputation inputs must come only from approved, assigned, funded jobs under precommitted weights and contribution caps. Operator bonds remain segregated from rewards and treasury assets, with explicit cooldown, pending-liability, slash, appeal, and return rules.

10.2 Threats and limitations

Material threats include smart-contract defects; compromised or concentrated administrators; forged, replayed, or prematurely accepted messages; chain reorganizations or halts; oracle or price manipulation in composite routes; insufficient liquidity; representation depegs; issuer freezes or upgrades; adapter mistakes; decimal loss; denial of service; censored redemptions; keeper-key compromise; malicious client jobs; cross-client authorization leakage; stale manifests or malicious desktop

updates; duplicate execution and gas races; assignment griefing; reputation farming through trivial, self-funded, related-party, or high-notional jobs; purchasable reputation after NFT transfer; arbitrary or confiscatory slashing; Sybil concentration; unfunded job or campaign promises; misleading pay-to-work or earnings claims; circular token incentives; and insolvency or legal failure of a custodian, issuer, client, payer, or economic module.

Audits reduce implementation risk but do not prove economic rights, eliminate operator risk, guarantee liveness, or make a third-party provider safe. A route that cannot be described with an actionable failure and recovery model is not eligible for production.

11 Interoperability and related approaches

TickerYard is intended to compose with existing standards and providers, not force every asset into a new representation.

OFT documentation describes debit and credit operations using burn/mint or lock/mint and burn/unlock patterns while preserving supply invariants [3, 4]. TickerYard adopts the need for explicit debit/credit accounting but is not itself an OFT and must not imply endorsement or shared security. Unit describes itself as an asset-tokenization layer for deposits to and withdrawals from Hyperliquid [6]; the comparison here concerns the abstraction pattern, not protocol equivalence.

Table 8. Positioning relative to adjacent approaches

Approach	Primary strength	TickerYard relationship
Issuer-supported or canonical bridge	Direct issuer alignment and recognized representation	Prefer and route through it when it meets policy
Bridge or liquidity aggregator	Broad discovery and route competition	Phase I owns the intent, validation, ranking, execution, and evidence boundary while using capable sources
LayerZero OFT and adapters [3, 4]	Standard debit/credit model for unified cross-chain fungible supply	Complementary; an issuer or authorized adapter operator still deploys and configures the mesh
Chainlink CCIP [1]	Cross-chain messaging and token-transfer infrastructure	One possible messaging dependency; application-level custody and authorization remain explicit
Unit [6]	Asset tokenization and deposits/withdrawals centered on Hyperliquid	Evidence that assets can be abstracted into a destination ecosystem; TickerYard proposes a multi-destination network
TickerYard native rail	Shared registry, backing, representation, and routing across approved destinations	Used only when a superior approved path does not already exist

Automation networks validate parts of the participation thesis without establishing TickerYard's implementation or demand. Keep3r documents a registry in which projects publish externally funded jobs and keepers choose work under job-specific requirements [8]. Gelato documents dedicated sender proxies that a target contract

can allowlist for a bounded automation relationship [9]. Olas presents a desktop path through which individuals run agents from a laptop [10]. Yardkeepers combines different elements: an owner-controlled local runner, an NFT ownership epoch, signed manifests, protocol-isolated exact capabilities, objective receipts, and client-funded work. Those differences require independent security and market validation.

Table 9. Positioning relative to adjacent operator approaches

Approach	Participation pattern	Yardkeepers distinction
Managed automation service	Protocol buys execution from a managed node or sender network	Yardkeeper operators keep local keys and qualify individually; client adapters remain isolated
Keep3r [8]	Jobs register work and pay qualifying external keepers	Curated retail desktop, NFT ownership epochs, signed manifests, exact client adapters, and explicit assignment
Olas operator tooling [10]	Individuals run packaged agents through a beginner-oriented desktop path	Narrow protocol-authorized on-chain calls and objective per-job receipts rather than general agent goals
Yardkeepers	Portable participation seat across separately approved client lanes	TickerYard is the first proposed client; no external client, job volume, or earnings are assumed

12 Fees and economic boundaries

After the 3,333-NFT collection exists, Anvil creates its sole fixed-supply token and mints the full amount into escrow without later mint authority or a creator allocation. The exact token supply and per-NFT quote are intentionally reserved for a separate launch-economics communication and the signed deployment manifest; this paper does not use denomination as a valuation claim. The factory policy assigns 95% of supply to AMM capacity, 5% to loans, and zero to its optional reward bucket.

The collection performs its immutable genesis mint into the distribution vault, but every public acquisition goes through Anvil. Before public opening, the generic direct-release path is disabled irreversibly and the purpose-limited seeder transfers all 3,333 NFTs to the pinned AMM in committed FIFO batches of at most twenty. It cannot redirect inventory, buy, borrow, activate, or make arbitrary calls. There is no whitelist, reserved allocation, contributor allocation, or discretionary later tranche. After seeding, the AMM holds all 3,333 NFTs, while the loan bucket remains initially unused.

The Yardkeepers contract fixes an ERC-2981 secondary-sale royalty of 575 basis points (5.75%) to a manifest-pinned royalty-receiver Safe and exposes no royalty setter. ERC-2981 only reports the receiver and amount; a marketplace must choose to honor it. This is therefore a possible NFT secondary-sale receipt, not an Anvil fee, LP fee, bridge fee, passive holder entitlement, or guaranteed revenue stream.

The offline release verifier replays Yardkeepers role history through a signed finalized anchor. The separate control observer then revalidates that anchor and supplies only its bounded later role-event tail, so it is a bounded online attester rather

than a trustless historical proof. A compromised observer key could forge historical absence before its signed snapshot. Before enabling an action, the browser independently verifies the manifest-pinned signature, 120-second freshness, finalized block/hash, post-snapshot role-event tail, and direct current expected members and reward-token set; those checks do not recreate an independent proof of earlier absence. The observer runs with a distinct key and isolated service from the offline release authority. Its signed validity window is at most 93 days. Key separation, expiry/attestation monitoring, incident response, and signed rotation remain external production operations gates.

For each gross full-collection seeding payout, 10% enters Anvil's soft-staking fee stream, 0.5% goes to the Anvil protocol recipient, and 89.5% becomes net seeder inventory. TickerYard sends all net inventory into two Uniswap V3 launch positions: 75% to YARD/WETH and 25% to YARD/STONKBROKER. Exact launch parameters belong to the separate launch communication and signed manifest. The launch vault atomically transfers both completed position NFTs to the pinned Stonk V3 locker under permanent fee mode 2, retains both lock-receipt NFTs, and exposes no principal-exit or migration surface. Permissionless collection routes 80% of LP fees to the pinned project LP-fee Safe and 20% to the pinned Stonk Safety Deposit; its documented 90/10 downstream rule produces the gross 80/18/2 project, community-leg, and Stonk-protocol benchmark. The 18% treatment is asset-specific: non-WETH assets may enter permissionless activated-broker claim rounds, while ETH/WETH is flushed to StockBooster rather than directly settled to broker token-bound accounts. Collection, round opening, broker clock-in, and WETH flushing are transaction-triggered rather than autonomous. External codehash, recipient, receipt-NFT, or fee-exemption drift fails closed. Ranges, fee tiers, lock transaction, lock-token IDs, receipt and fee-right recipients, and out-of-range behavior remain signed-manifest fields, while signed Stonk terms, approved direct-broker treatment for every fee asset, and audit remain open deployment gates.

Borrowing is available but deliberately bounded to 5% of supply. One NFT secures one factory-defined principal for a fixed 365-day term at 15% APY. Full-term interest is withheld up front, so the borrower initially receives 85% of principal before the live oracle-priced Robinhood ETH creation fee; the stated interest is therefore about 17.65% of the tokens actually delivered. A borrower repays principal plus any late fee; the grace period is seven days, the cap is five active loans per wallet, and the liquidation incentive is 2.5%. Exact principal and aggregate capacity belong to the separate launch communication. After grace expiry, liquidation moves the principal accounting from the loan bucket to the AMM bucket, sends the NFT to AMM inventory, and pays the incentive from remaining loan capacity. The signed release verifier requires the fully seeded AMM to retain enough conservative capacity for the maximum loan principal; availability is not a promise of demand, liquidity, or borrower suitability.

The Keeper NFT is a fixed-genesis participation seat with one deterministic token-bound account. The owner activates through Anvil, whose five tier amounts route 95% to a dead address and 5% to the external Anvil treasury. That transfer makes tokens economically unavailable but does not reduce ERC-20 `totalSupply()`. Base through Tier 3 may receive only their Anvil-defined owner-wallet fee share. The highest tier—Tier 4, onchain index 4, at 240% of the fixed quote and 3.33x Anvil weight—is the sole network-level eligibility gate. A separate Keeper runner registry verifies that current-owner Tier 4 state and binds an accepted local runner through zero-value Keeper Enrollment without another token charge or burn. Transfer invalidates market activation and runner authorization; the buyer must reach Tier 4 and enroll for the new ownership epoch, while same-owner runner rotation is free. Enrollment opens

eligibility to seek client-specific qualification; it does not whitelist the seat into an external protocol. Anvil weight affects only Anvil's external stream, while capability, assignment, and work allocation depend on separately evidenced current-owner and current-runner qualification. The user separately owns and funds the runner's native ETH, and the runner has no authority over the NFT account.

The zero factory-reward bucket is distinct from Anvil's AMM-fee-funded soft-staking stream. In the reviewed Anvil contract snapshot, that fee stream remains economically active when factory reward basis points are zero: when a nonzero fee pool is streamed, every currently active token weight accrues its pro-rata Anvil share; a zero pool produces zero new accrual. Claims pay the current activating owner wallet, not the NFT account; transfer causes activation and pending-stream loss under the Anvil settlement rules; and weights have no warm-up, minimum active cohort, per-NFT cap, or wallet cap. These are accepted and disclosed Anvil market constraints, not TickerYard Keeper-work rewards. Future keeper jobs and Sponsor Trains must escrow their maximum user, conductor, keeper, gas, race, refund, and prize liabilities before opening. Gas reimbursement returns to the runner that spent gas; a successful-work bounty pays the NFT account. Only an earned TickerYard service fee is revenue.

12.1 Participation service portfolio and payment paths

Tier 4 plus Keeper Enrollment is the common operating-seat gate, not a universal competence certificate. A seat qualifies separately for each typed capability, and current operator evidence remains distinguishable from transferable NFT history. The initial commercial portfolio is deliberately bounded:

Table 10. Externally funded Keeper service lanes

Lane	External buyer	Objective Keeper service and payment form
Cross-chain completion and rescue	Bridge, issuer, or integration partner	Finalize, retry, recover, or reconcile one destination action; capacity retainer plus gas and completion bounty
Conditional intent automation	User, wallet, or fintech partner	Trigger a signed DCA, limit, recurring conversion, or bounded rebalance; subscription or per completion
Sponsor verification	Sponsor, issuer, or ecosystem	Verify finalized use, retention, or a deterministic milestone, then release or refund escrow; campaign verification budget
Watchtower and safety	Protocol or treasury	Prove a defined stale attestation, backing mismatch, admin change, or oracle fault and invoke only a bounded response; availability retainer plus incident bounty
Treasury and issuer operations	Partner treasury or tokenized-asset issuer	Perform an allowlisted sweep, epoch rollover, distribution, or redemption-window action; integration fee, retainer, and per action
Solver or filler	External intent order flow	Settle through an established reactor using operator capital; operator execution economics, not guaranteed NFT income

Protocol clients procure those lanes through a participation-service agreement: adapter and integration fee, pre-funded capacity retainer, maximum gas and retry li-

ability, successful-work bounty, optional incident reserve, and TickerYard service fee. Gas reimbursement and operator service pay go to the runner. A disclosed capped seat fee or successful-work bounty may go to the NFT account. Only TickerYard's earned service fee is revenue; the remaining balances stay segregated liabilities until earned, refunded, or released.

A retainer purchases a finite measured roster, not every activated NFT. Proven operators may receive more assignments, while a published newcomer allocation can provide access to real low-limit work. Heartbeats, quotes, activation, or NFT ownership alone do not create TickerYard compensation. Every valuable lane requires a typed primary assignment, backup, objective postcondition, and public or independently operated fallback. Price-sensitive work uses an established solver or reactor rail: the NFT authorizes participation but does not prove a fair quote, independent infrastructure, identity, liquidity, or future uptime.

A client protocol's own NFT, token, treasury, fee converter, work tip, or holder distribution remains outside the Yardkeepers accounting boundary unless signed terms and an audited adapter say otherwise. An external client's native holder program may continue independently while Yardkeeper operators receive only the assigned work payment defined for their completed actions. Allowlisting a Yardkeeper adapter does not duplicate, redirect, or dilute those native rights. A dual holder may qualify under both systems, but the resulting rights remain separate rather than becoming one promised yield.

For any production route, TickerYard should disclose separately:

- source and destination network gas;
- provider, liquidity, and swap fees;
- TickerYard service or protocol fees, if any;
- expected price impact and protected minimum output;
- which costs are estimates and which are contract-enforced; and
- whether fees are taken from user principal, paid separately, or accrued in kind.

External revenue means settled consideration from a non-project counterparty for routing, integration, job, campaign, API, automation, analytics, or white-label service. User principal, backing, pass-through gas, runner balances, campaign or job escrows, bonds, Anvil activation sinks and fee streams, NFT/token sale proceeds, emissions, and token or LP mark-to-market value are not external revenue.

Any activated-StonkBroker or Stonk protocol share is a source-specific contractual claim, not an inferred entitlement over every TickerYard business line. The discussed 20.00% and separate 2.23% bridge terms apply only if a signed agreement defines the revenue base, scope, beneficiary, checkpoint, payout asset, and legal treatment.

The local `BridgeRevenueRouterV1` source enforces the cumulative 20.00% activated-broker, 2.23% Stonk-protocol, and 77.77% TickerYard bridge waterfall for one immutable fee source and immutable recipients, without an owner, upgrade, sweep, or alternate-recipient path. It is undeployed. The reviewed legacy wrapped-stock vault remains incompatible because its owner can choose an arbitrary fee-withdrawal recipient. Production therefore still requires a patched, audited source that makes the router its only fee destination, plus signed terms and checkpointed beneficiary evidence.

Special Projects market fees are distinct. Any Anvil-treasury, StockBooster, lock-receipt, Safety Deposit, activated-StonkBroker, or Stonk protocol share applies to YARD only when the exact deployed contracts and signed terms define it. Those

same receipts are not charged again under the separate illustrative bridge percentages.

Every TickerYard covered source proposed for the separate revenue-epoch module must publish a policy commitment fixing the revenue base, direct costs, senior claims, Keeper share basis points, payout asset, and accounting period. The conditional rule is exact: when positive settled distributable TickerYard revenue is recognized and reserve gates pass, the declared Keeper percentage is the fresh allocation; otherwise it is zero. This guarantees neither volume nor revenue. Where TickerYard controls an onchain fee recipient, the covered share should route directly into the revenue module. A partner-controlled or offchain source remains dependent on signed terms and reconciled reporting because a distributor cannot discover concealed or undeclared receipts.

The cross-protocol participation network does not use this TickerYard revenue module by default and creates no automatic claim on an external client's revenue. After six reconciled TickerYard external-revenue epochs, twelve months of non-token cash runway, fully covered senior liabilities, security reserves, audit, and legal clearance, a separately deployed activated-Keeper program may allocate at most 15% of TickerYard residual revenue to eligible NFT accounts. Eligibility requires current-epoch Anvil Tier 4 and Keeper Enrollment. Anvil's all-tier market stream continues to use Anvil weight; the TickerYard allocation gives every eligible Keeper seat one equal weight and never imports the Tier 4 3.33x multiplier. The general epoch requires one warm-up epoch and at least fifty eligible NFTs, caps one token ID at 2%, and allocates zero when revenue or reserve headroom is zero.

`KeeperRevenueDistributorV1` is now implemented as production-shaped, undeployed source. A proposal commits the revenue-source ID and policy hash, declared distributable revenue, share basis points, payout asset, finalized snapshot block and hash, complete sorted roster-manifest hash, entitlement root, and eligible count. A collection-bound administrator, proposer, funder, and challenger are distinct contract authorities; rotation cannot merge those roles or remove the last member of one. The manifest-pinned review delay must be 2–30 days and the claim period 30–365 days. A challenge permanently blocks funding of that proposal, immediately releases committed rollover, and requires cancellation and replacement. Exact fresh funding plus any same-asset rollover precedes claims.

Anyone may relay a valid claim, but the recipient is fixed to `account(tokenId)`. A bitmap permits one claim per token and epoch. Transfer-tax payout assets fail closed; failed exact delivery preserves the claim and liability; and expired unclaimed funds plus integer dust become disclosed same-asset rollover. A pre-transfer entitlement continues to the deterministic TBA that travels with the NFT. The contract enforces funded arithmetic, delivery, uniqueness, and per-asset solvency. It does not prove that external revenue reporting or a Merkle roster is complete. Direct fee adapters, an independently reproduced sorted roster, published proofs, and challenger review are therefore release gates. Once that correct roster is funded, every checkpointed eligible NFT has its deterministic equal-seat claim.

`$YARD` held inside the NFT account is ordinary withdrawable bearer inventory and never increases activation or reward weight. An optional post-launch starter parcel must be market-purchased from a disclosed budget and remain immaterial relative to Base activation. The fixed NFT quote is funded by Anvil escrow, not by assets inside the token-bound account. The reviewed Anvil AMM does not itself enforce clean-account intake. The guarded adapter permits existing, deployed, or dusted canonical token-bound accounts only with explicit bearer-state acknowledgement and protects only observable mid-call drift across the adapter, AMM, and recipient calls; arbitrary

assets at a deterministic account remain non-enumerable. Optional protocol-funded starter parcels remain disabled pending an audited pricing and clean-seat policy.

Backing, pending user obligations, pass-through network fees, runner gas, job and campaign escrow, partner liabilities, NFT reward assets, token-bound-account inventory, Keeper-market reserves, protocol revenue, security reserves, POL, and treasury assets must remain logically and physically distinguishable. Neither $YARD, a liquidity position, an NFT, nor a treasury asset may be counted as backing for a canonical asset unless it is itself the exact eligible backing defined by that asset's registry policy. Token, NFT, or economic governance never controls routing rank, custody, release, peers, attesters, safety delays, listings, or exits.

Economic constitution

External capital enters before TickerYard rewards leave. The NFT has no TickerYard work payout in the MVP; Anvil's separate owner-wallet stream follows its disclosed all-tier weight rules; client work is paid only from client-funded liabilities; and $YARD holders receive no TickerYard protocol funds. External client holder programs remain client-native and create no Yardkeeper entitlement. A TickerYard covered source with positive settled distributable revenue must deposit its published Keeper share before claims open, and a correct funded roster gives every checkpointed eligible NFT its deterministic claim. Zero distributable revenue creates zero fresh allocation. Activation demand, token trading, and an unapproved client opportunity are not evidence of product demand.

13 Roadmap and release gates

The phases below are evidence gates rather than date promises.

Table 11. Proposed delivery sequence

Stage	Product state	Required evidence and exposure boundary
0. Partner prototype	Routing UI/API and reference stock integration	Reproducible local tests, exact configuration, honest route status, and no production claim. **Exposure:** no unrestricted public custody.
1A. Production routing	Selected external routes	Provider adapters, protected-output tests, credential isolation, durable status, recovery runbooks, monitoring, and legal review. **Exposure:** route- and amount-capped.
1B. Initial asset rail	Audited certificate and fungible stock lanes	Patched contracts, independent audit, separated roles, signed manifests, lifecycle evidence, recovery, limits, and issuer approval. **Exposure:** capped assets, networks, and value.
2A. Canonical-vault pilot	One low-complexity asset and limited destinations	Formal state machine, invariant and fuzz testing, two independent reconciliations, audit, bounty, and incident exercise. **Exposure:** conservative lane and global caps.
2B. Ticker asset network	Multiple approved assets and adapters	Adapter-specific audits, governance and operations evidence, sustained reconciliation, and a transparent risk registry. **Exposure:** incremental admission only.

The Yardkeepers Participation Network follows an independent evidence track:

The public product labels do not replace the evidence stages below. Phase 1 contains the collection and Stage M market launch. Stages A and B prepare the separate Phase 1.5 TickerYard-client release; Stage C opens that capped canary only after a signed registry-unpause authorization. External client admission is a later evidence stage, not a consequence of the TickerYard canary.

- **M — Phase 1 Anvil market launch.** Mint the fixed collection into its vault, permanently close direct NFT release, create the fixed-supply factory-issued $YARD with 95/5/0 buckets, seed all 3,333 NFTs through Anvil, create and permanently lock the two launch positions, disclose borrowing and owner-wallet rewards, prove the same-token partner path, complete legal publication controls, accept the external admin graph, and keep the runner registry paused.
- **A — Phase 1.5 local preparation.** Mock or test-value NFT and Anvil activation, no user value, and complete install, key, simulation, pause, restart, and recovery tests.
- **B — Phase 1.5 public-testnet preparation.** Prove fixed supply, ownership epochs, pinned Anvil Tier-4 reads, zero-value Keeper Enrollment, runner acceptance, transfer invalidation, registry pause/unpause, typed execution, and work receipts.
- **C — Phase 1.5 TickerYard-client mainnet canary.** Separately authorize registry unpause, one audited immutable TickerYard job, current-owner Anvil Tier 4, token-free Keeper Enrollment, user-funded ETH, signed desktop build, low caps, zero TickerYard work reward, and manual fallback.

- **D — funded TickerYard jobs and Level 0.** Maximum-liability escrow, refundable base bond, current-epoch Level 0 entry, precommitted quality scoring, deterministic primary and backup assignment, objective completion, bounded challenge/slash/refund accounting, settlement-asset pay, and positive operator unit economics.
- **E — first external client adapter and higher levels.** Written client authorization, isolated target and selector, client mapping of Levels 1–3 to its lanes, funded job liability, assignment without a retail gas race, signed manifest, audit, incident exercise, and explicit separation from client-native holder economics.
- **F — Sponsor Trains.** Pre-funded deterministic outcomes, retention and anti-wash rules, conductor attribution, refunds, and sponsor reporting.
- **G — mature TickerYard token economy.** Reconciled source-specific TickerYard external revenue, reserve and runway gates, bounded POL/buybacks, direct fee adapters or reconciled reporting, independently reproduced complete rosters, and an audited, deployed, and legally cleared `KeeperRevenueDistributorV1`.
- **H — Anvil market operations.** Post-launch inventory and loan monitoring, clean-seat policy, concentration evidence, out-of-range and fee-settlement handling, and reviewed exits.
- **I — higher-trust work.** Objective bonds, evidence-based slashing, appeals, capability-specific qualification, and additional settlement, intent, verification, watchtower, treasury, or solver executors under typed limits and fallback.

At the repository snapshot supporting v1.1, the production-shaped contract graph and separate local-only Stage A artifacts provide implementation evidence for distinct parts of the planned system. The legacy Solidity fixtures live in the isolated `stage-a/` Foundry root beneath `contracts/keeper` and are excluded from production compilation. `apps/keeperd` keeps the zero-value desktop simulation separate and now also contains a signed-manifest-gated production service for runner acceptance, exact endpoint submission, and deterministic restart recovery; the default desktop facade leaves that service disabled until reviewed RPC, state, fee-policy, and trust-root inputs exist. Focused unit and fuzz tests cover the separate TickerYard revenue distributor's funding, cap, claim, transfer, rollover, replay, restricted-asset, and conservation boundaries. Local unit tests and reverted Robinhood-fork compatibility tests cover the remaining production contract and runner seams without broadcasting to a public network. These sources do not prove a complete revenue statement or roster, deployed distributor, public runner lifecycle, public-testnet Stage B, mainnet Stage C, or any external client adapter, allowlist, funded job, code-sharing arrangement, or partnership. The 288-composite internal expansion pilot is not the final collection: art, rarity, rights, content-addressed metadata, populated signed deployment manifest, independently reproduced external dependencies, audit, and legal record remain open gates.

No stage exits on interface completeness alone. Each value-bearing stage requires verified deployed bytecode and configuration, reconciled backing and supply, end-to-end transactions in every supported direction, alerting, an incident runbook, documented user recovery, legal approval for the offered assets and jurisdictions, and no unresolved critical or high-severity audit finding.

14 Open design questions

The proposed protocol still requires decisions that should be resolved in a formal specification:

- Which verification model and independent operators authorize native-rail credits on each network?
- How are heterogeneous finality, chain reorganizations, and non-EVM light-client limitations normalized?
- Which role can quarantine backing or a representation after an issuer freeze, and how can users exit safely?
- How are fees, rebases, corporate actions, decimal dust, and forced token migrations reflected without violating Equation 5?
- What is the smallest governance surface consistent with rapid emergency disable and delayed re-enable?
- When should TickerYard retire its own representation in favor of a later issuer-supported canonical path?
- How are economically equivalent but legally different asset representations shown without confusing users?
- Which signed operating policy should monitor the fully seeded Anvil inventory, loan utilization, liquidations, fee settlement, activation concentration, and clean-seat risk after launch?
- Which exact Anvil factory, market, token, escrow, liquidity, collection, runner registry, executor, endpoint, external-admin, and signed-desktop manifests satisfy the $YARD and keeper policies?
- Which ERC-6551 registry, account implementation, codehash, salt, execution policy, and asset display make each Keeper account safe and discoverable?
- Which funded TickerYard job follows the zero-value canary, which settlement asset pays it, and how is primary and backup assignment enforced before material bounties?
- Which first external client intentionally authorizes an isolated adapter, and which code, target, selector, qualification, funding, pause, fallback, and incident terms define that lane?
- Which job classes are genuinely suitable for home operators rather than professional latency, capital, or uptime infrastructure?
- Which operator history travels as public NFT provenance, which current-owner qualifications reset on transfer, and how can clients reproduce the resulting eligibility decision?
- Which exact minimum time, capped difficulty weights, quality measures, client-diversity rules, challenge window, decay, and promotion authority govern Levels 0–3 without letting wealth, notional volume, trivial jobs, or related parties manufacture reputation?
- What base and capability bonds are proportionate to retail risk, which objective violations permit a bounded slash, who reviews and hears appeals, and when can an operator exit and recover the bond?
- Can one economic operator control multiple seats at higher levels, and which privacy-preserving identity or anti-Sybil evidence may a client require?
- How does the job board display expected payment, maximum gas, non-winning

race risk, assignment, refund, tax, and jurisdictional restrictions without advertising guaranteed earnings?

- Which revenue sources are subject to activated-StonkBroker or Stonk protocol claims under signed terms, and which remain separate business lines?
- Which direct revenue adapters or reconciled statements, independently reproduced roster builder, proposer/funder/challenger Safes, payout-asset allowlist, exact review/claim periods, audit, legal record, and launch date authorize the implemented mature Keeper distributor?
- What post-launch budget, if any, buys an optional starter $YARD parcel without mispricing loaded accounts in Anvil?
- In which jurisdictions can Keeper NFT operating rights and distributions, Anvil activation sinks and owner-wallet claims, token-bound assets, Sponsor Trains, Anvil pairing, and buybacks be offered, and what disclosures or exclusions are required?

These are protocol boundaries, not implementation details to be hidden behind a user interface.

15 Conclusion

TickerYard begins with a practical question: what valid route can move this exact asset from its current network to the requested destination under the user's policy?

Phase I answers that question through a unified routing, validation, execution, and evidence layer. Tokenized equities provide an exacting first vertical because they expose both the usefulness and the limits of cross-chain representations. Phase II proposes a canonical-vault and representation network for eligible assets that lack an acceptable existing path.

The Yardkeepers Participation Network adds a complementary question: how can an individual qualify for and complete exact useful protocol work from a locally controlled runner, with isolated authority, objective receipts, funded liabilities, and a client-approved fallback? Its NFT represents a portable participation seat and bearer account, while every client capability remains separately earned, assigned, bounded, and revocable. TickerYard is the first proposed client and supplies the first exact certificate-completion adapter; that starting point proves neither external adoption nor paid demand.

Anvil $YARD represents market activation, pairing, commitment, and scarcity without a TickerYard token-holder revenue right. Any active Anvil tier may receive its external tier-weighted market share; only current-owner Tier 4 plus zero-value Keeper Enrollment opens network-level eligibility. External clients retain their own tokens, NFTs, holder programs, target contracts, and economic policies. They fund job capacity before operator payments leave, while allowlisting a Yardkeeper adapter creates no automatic client-revenue claim. A mature TickerYard covered source creates equal-seat NFT claims only after positive settled distributable TickerYard revenue, reserve clearance, exact epoch funding, audit, and legal approval.

The end state is not a world without chains or trust. It is a system in which chain, route, representation, custody, work, and economic choices are made explicit, constrained by accounting and solvency invariants, and presented through one coherent interface.

Route markets. Open verifiable protocol work.

A Minimum route disclosure

Every executable route should expose or link to a machine-readable record containing:

- canonical source and destination asset IDs and contract identifiers;
- every provider, protocol, swap, bridge, vault, and representation involved;
- protected minimum output, fees, gas estimates, quote expiry, and expected timing;
- source and destination finality assumptions;
- administrators, pausability, upgradeability, rate limits, and recovery authority where known;
- message, transaction, and destination evidence identifiers; and
- TickerYard service fee, affiliate or sponsor attribution, campaign eligibility, reward source, and whether any amount is a pass-through liability; and
- terminal-state and refund procedures.

References

[1] Chainlink, "CCIP Architecture," *Chainlink Documentation*. Accessed 8 August 2026. https://docs.chain.link/ccip/concepts/architecture

[2] Chainlink, "Send Arbitrary Data," *Chainlink Documentation*. Accessed 8 August 2026. https://docs.chain.link/ccip/tutorials/evm/send-arbitrary-data

[3] LayerZero, "Value Transfer Implementations," *LayerZero V2 Documentation*. Accessed 8 August 2026. https://docs.layerzero.network/v2/concepts/value-transfer-implementations

[4] LayerZero, "LayerZero V2 OFT Quickstart," *LayerZero Documentation*. Accessed 8 August 2026. https://docs.layerzero.network/v2/developers/evm/oft/quickstart

[5] F. Vogelsteller and V. Buterin, "ERC-20: Token Standard," *Ethereum Improvement Proposals*, EIP-20, 2015. https://eips.ethereum.org/EIPS/eip-20

[6] Unit, "About Unit," *Unit Documentation*. Accessed 8 August 2026. https://docs.hyperunit.xyz/

[7] StonkBrokers, "NFT Collection, Anvil NFT AMM, and Activation & Distributions," *Protocol Documentation*. Accessed 8 August 2026. https://www.stonkbrokers.cash/docs

[8] Keep3r Network, "Introduction, Keepers, and Jobs," *Keep3r V2 Documentation*. Accessed 9 August 2026. https://docs.keep3r.network/

[9] Gelato, "Dedicated msg.sender," *Web3 Functions Documentation*. Accessed 9 August 2026. https://docs.gelato.cloud/web3-functions/security-considerations/dedicated-msg-sender

[10] Olas, “Operate: Run AI Agents,” *Olas Operator Documentation*. Accessed 9 August 2026. https://olas.network/operate

[11] European Parliament and Council, “Regulation (EU) 2023/1114 on Markets in Crypto-assets,” *Official Journal of the European Union*. Consolidated text accessed 9 August 2026. https://eur-lex.europa.eu/eli/reg/2023/1114/oj

[12] Comisión Nacional del Mercado de Valores, “MiCA: Nueva regulación de criptoactivos.” Accessed 9 August 2026. https://www.cnmv.es/Portal/mica/regulacion-criptoactivos?lang=es