

hunch.

The decision layer for agentic RWA finance

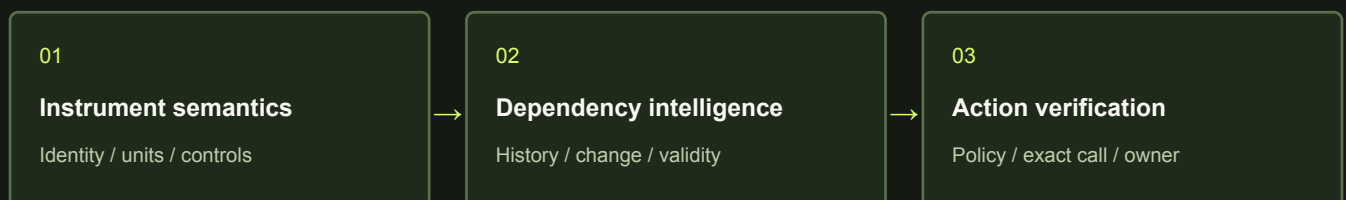
Financial meaning. Verifiable constraints.
Evidence before execution.

Hunch's ambition is to become the infrastructure that wallets, trading applications, vault operators and autonomous agents use to determine whether a proposed financial action satisfies their rules against sufficiently current evidence.

Tokenization brings assets onchain. Agents bring automation. The intersection creates a recurring infrastructure problem: software must understand what an asset represents, how that meaning changes, and which conditions must still hold when capital moves.

Hunch begins with Robinhood Chain and tokenized equities. Its proposed architecture connects an instrument and dependency graph, an engine that tracks changes over time, and a verifier placed at the application's signing boundary. Every supported decision should be reproducible, scoped to an exact action and invalidated when a required assumption changes.

The proposed business model earns revenue from maintained financial semantics, decision infrastructure and continuous monitoring. The potential moat compounds through validated adapters, longitudinal evidence, independently adjudicated failure cases and deeply integrated operating workflows.



Status. Hunch v0.5.0 provides live read-only RH intelligence and developer interfaces. The graph, durable monitoring and transaction-bound verification architecture described below are proposed engineering work. Section 11 separates the existing implementation from the intended system. This paper defines the technical direction and the evidence required to substantiate each capability.

01 / The thesis

The same token can have different financial meaning over time

An ERC-20 interface provides balances, transfers and approvals. A financial application also needs to understand the instrument behind those units. For RH stock tokens, corporate actions can change the shares represented by a token while raw balances and total supply remain unchanged. A price may describe an underlying share or an already adjusted token. An oracle can retain a value during a pause. A successful call can use an unacceptable recipient, allowance or route. [1–3]

These distinctions create work for every application that integrates the asset. They also create correlated risk when applications repeat the same incorrect interpretation.

Hunch's thesis is that **financial meaning and execution constraints should be maintained as shared infrastructure**. An application should be able to ask:

Given this instrument, this proposed action, this owner policy and this observed state, which requirements are satisfied, which fail, and which remain unknown?

The output should remain useful to a human operator, a deterministic wallet policy and an AI agent. It should carry enough evidence to reproduce the conclusion and enough scope information to prevent reuse for another action.

Start where the integration burden is concrete

The first commercial focus is RH applications handling stock-token valuation, swaps and portfolio rebalancing. These workflows combine issuer metadata, corporate actions, oracle state and transaction constraints. They are a coherent engineering surface with explicit primary documentation. [1–4]

Lending, collateral management and other asset classes expand the opportunity, but require additional protocol and instrument models. A stock-token adapter cannot establish collateral eligibility, liquidation economics or the redemption rights of a different security.

The initial customer is a development or risk team operating an application that moves or accounts for tokenized assets. Individual traders can use the workspace, while the recurring business is integration into operating software.

Three concrete sources of value

Workflow	Failure the application must avoid	Hunch's intended contribution
Portfolio valuation	Apply a corporate-action multiplier twice, or retain an old conversion after its effective time	Typed price units, source reconciliation and time-aware revaluation
Swap or rebalance	Approve one route and sign different calldata, or use a quote after a dependency changes	Exact action comparison, bounded approvals and invalidation before signing
Ongoing operation	Miss an implementation upgrade, oracle pause or issuer metadata change that affects multiple positions	Dependency-based impact analysis, persistent alerts and an inspectable history

Customer value is measured through avoided integration work, independently confirmed policy violations identified, lower investigation time and reliable operation. Counting scans alone does not establish any of these outcomes.

02 / The instrument and dependency graph

Treat financial units as types

The graph's primary key is a chain and contract address. An issuer asset identifier can connect deployments when the issuer explicitly establishes that relationship. Symbols and names remain display metadata. Cross-chain identity does not imply equivalent liquidity, permissions, bridging rights or execution conditions.

Each supported instrument model records:

- **Identity and claims:** issuer registry reference, deployment identity, instrument class, and separately sourced terms or eligibility information.
- **Units:** raw base units, decimals, whole tokens, underlying share-equivalent units, quote currency and price denomination.
- **Controls:** recognized implementations, runtime hashes, verified-source provenance, privileged roles and the specific powers established by analysis.
- **Market dependencies:** oracle feeds, their units and freshness rules, actual route venues, counter assets and relevant hook or pool implementations.
- **Time:** when an observation was received, when its source says it became valid, scheduled changes, block identity and the required confirmation level.

Assertions carry provenance and a verification method. A fact observed in runtime state, a statement from an issuer and a conclusion from code analysis remain distinct. Missing edges are explicit. The graph does not equate a verified contract source with verified asset backing or legal entitlement.

A dimensional example

Let B be an integer token balance in base units, d its decimals, M the 18-decimal shares-per-token multiplier, and P_{share} a price denominated in dollars per underlying share.

```
wholeTokens      = B / 10^d
sharesPerToken   = M / 10^18
shareEquivalent  = wholeTokens × sharesPerToken
referenceUsd      = shareEquivalent × Pshare
```

For an already adjusted token price P_{token} , the value is $\text{wholeTokens} \times P_{\text{token}}$. The multiplier is not applied again. Implementations use fixed-point integer arithmetic with explicit rounding; the equations above express units rather than floating-point code. [1–3]

Consider 100 tokens, a multiplier of 1.05 and a share reference of \$200: the share equivalent is 105 and the reference value is \$21,000. An already adjusted token reference of \$210 produces the same value. Applying 1.05 to that \$210 again incorrectly yields \$22,050. These are illustrative inputs, not a live quote or executable valuation.

A scheduled multiplier change invalidates a prior conversion at its effective time. Whether total economic value increases or decreases also depends on the underlying price and the corporate action. A stock split cannot be modeled as free value creation.

Make dependencies explicit

```
Issuer asset → chain deployment → runtime / implementation / roles
      ↓
      unit and entitlement model
      ↓
Price source → typed valuation → position / proposed action
      ↑               ↑
corporate-action state   actual route / counter asset
```

An owner chooses the evidence requirements appropriate to each action class. Displaying a reference value, executing an approved swap and accepting collateral require different evidence. Every class must define its mandatory dependencies and treatment of unknowns. This avoids both unsafe clearance and an unusable system that requires every conceivable check for every action.

03 / Intelligence across time

The hard question is what a change invalidates

A list of current facts is a starting point. The stronger infrastructure maintains the relationship between a changed fact and the decisions that relied on it.

Suppose a token's oracle pauses. The system should identify affected reference valuations, queued rebalances and monitoring rules. Suppose a router implementation changes. The system should identify receipts built against its old runtime and request a new evaluation. A scheduled corporate action must trigger at its effective time even if no user reloads a page.

The proposed event store records both **source-valid time** and **observation time**. This permits reconstruction of what Hunch knew at a decision, including information learned later or corrected by the provider. Corrections append a superseding observation instead of silently rewriting history.

Consistency is part of the evidence

Onchain observations should include chain ID, block number, block hash, parent hash and confirmation policy. Where supported, hash-pinned reads use EIP-1898 with canonicity requirements. Where unavailable, the adapter must disclose its weaker consistency model or reject an action whose policy requires stronger guarantees. [5, 12]

Offchain prices and registry responses carry source time, receipt time, request identity, schema version and a content digest. They cannot be made atomic with chain state by assigning them a shared timestamp. A decision must explicitly bound acceptable time skew and source age.

The observer handles gaps, duplicate events, out-of-order delivery and reorganizations. Decisions depending on orphaned state become invalid. A redundant RPC helps detect disagreements, but two endpoints sharing an upstream provider are not two independent sources.

Validity follows the earliest relevant boundary

```
validUntil = earliest of:
  evidence expiry
  executable quote deadline
  policy expiry
  scheduled relevant state transition
  session or authorization expiry
```

An observed dependency change can invalidate a decision earlier. Reusing cached evidence is permitted only when its identity, dependencies, consistency and age satisfy the current request. Unknown freshness never becomes a fresh observation because a cache was read recently.

The event store can live in PostgreSQL with indexed dependency edges and an append-only observation log. A specialized graph database is a later performance choice. The moat depends on the correctness and history of the relationships, not the database brand.

04 / From proposed action to enforceable constraints

Separate the owner's policy from the agent's proposal

An agent may propose a swap, a rebalance or an approval. It cannot grant itself a larger spending limit, add a trusted recipient or relax evidence requirements. Policies are selected and versioned through an authenticated operator path that untrusted text and the planning model cannot modify.

The proposed action compiler decodes supported calls into an economic intent: asset inputs and outputs, amount bounds, recipient, spender, approval scope, venue, deadline and expected state effects. An unrecognized function, hook or batch member yields `unsupported` or `review`. A plausible natural-language explanation cannot substitute for decoding.

Simulation should inspect state changes and asset flows for supported routes. It must account for allowances, transfer behavior and applicable hook accounting. Logs alone are insufficient because contracts can emit misleading events. Exact balance-delta interpretation also requires asset-specific semantics.

Bind the result to the action that will be signed

The proposed decision envelope binds at least:

Binding	Required content
Domain	Schema version, service or verifier identity, chain ID, intended consumer and key ID
Action	Account, target, value, calldata digest, ordered batch calls, nonce or session replay scope, and permitted fee or gas bounds
Economic intent	Token identities, raw quantity limits, recipient, allowance limits, minimum output and quote deadline
Decision basis	Operator policy hash, adapter versions, evidence manifest, dependency versions, confirmation requirements and explicit gaps
Lifetime	Issued time, valid-until boundary, request ID and revocation reference

Smart-account support must additionally model the relevant account implementation, EntryPoint version, validator or session module, delegatecall exposure and sponsorship restrictions. Provider-controlled fields must have explicit allowed bounds. A generic EOA tuple is insufficient to authorize an arbitrary account-abstraction operation. [4]

A signed envelope authenticates who produced a statement and the bytes it covers. It does not establish that every provider was truthful or every contract was safe. Typed signatures such as EIP-712 provide domain separation; nonce handling and replay protection must be implemented separately. [13]

The verifier sits at the trust boundary

The application's trusted runtime checks the signature, policy hash, exact outgoing action, freshness, dependencies and replay state immediately before allowing its signer to proceed. Request-time comparison is insufficient if calldata changes later.

Where a supported contract can enforce amount bounds, recipient and deadline atomically, those constraints belong in the executed call. For offchain facts, a remaining observation-to-execution race exists. Policies must bound that exposure; operations requiring guarantees the adapter cannot enforce remain unavailable for automatic execution.

```
Agent proposes → decode supported action → resolve evidence
                → evaluate owner policy → issue scoped decision
                → local verifier rechecks → owner's signer
                → reconcile execution and record outcome
```

The proposed states are `block`, `review`, `unsupported` and `satisfies_policy`. The last means that named requirements passed within a supported scope and validity window. The current public context API emits only `review` or `block`.

05 / The role of AI and the threat model

Intelligence should improve investigations

Models can help extract candidate issuer terms, identify suspicious control relationships, explain dependency changes and suggest a smaller or differently routed action for evaluation. The resulting suggestions return through the same deterministic policy checks. Changing the route or amount requires a new decision.

Claims extracted from documents retain an exact source span, document version and verification status. A model may propose an assertion; it cannot promote its own assertion into a trusted permission, trusted price or signing authorization. Current Hunch explanations use deterministic synthesis, and current untrusted-text checks use heuristics. No existing frontier-model reasoning engine is claimed by this paper.

The desired learning system improves retrieval, coverage and investigation quality from labeled cases. Model upgrades pass a fixed regression suite before release. Private prompts, wallet activity and customer policies are excluded from shared training by default.

Threats and their architectural consequences

Threat	Required response
Malicious metadata or prompt injection	Treat retrieved text as data; isolate policy configuration and signer access
Incorrect price units or issuer mapping	Typed units, address-and-chain identity and conflicting-source handling
Stale state, reorganization or sequencer outage	Explicit finality and freshness requirements; invalidate affected decisions
Proxy upgrade or hidden authority	Bound source verification and runtime analysis; unresolved behavior remains unknown
Calldata substitution or replay	Exact-action verification, scoped signatures, deadline and nonce/session enforcement
Compromised Hunch key or adapter release	Key rotation and revocation, signed releases, version pinning and operator kill switch
Service or data-source outage	Fail according to the owner's published policy; an error cannot manufacture authorization
Cross-tenant evidence exposure	Tenant isolation, minimized retention and restricted access to sensitive evidence

The system cannot establish offchain backing from token code, guarantee issuer solvency, prevent every exploit or make a compromised owner signer trustworthy. These boundaries determine the product's actual assurance level.

06 / Where the moat can compound

Five assets that become harder to reproduce

1. Validated financial semantics. Each supported issuer, asset class and venue contributes a maintained model of units, lifecycle events, permissions and failure conditions. An adapter ships with source provenance, conformance fixtures and a scope manifest. Quality includes recognizing where a model does not apply.

2. Longitudinal dependency history. Accurate historical relationships permit replay of decisions at the time they were made, reconstruction of corporate-action transitions and attribution of affected positions. Some history is reconstructible from public data; correctly joining it with time-stamped offchain evidence and validated interpretations takes continuing work.

3. An adjudicated decision corpus. Useful cases include unit mismatches, route substitution, oracle recovery, authorization changes and benign actions that should pass. Labels must be reviewed independently of the system

being evaluated. Synthetic cases supplement real observations, while source and tenant separation prevent benchmark leakage.

4. Trusted integration and operations. A verifier embedded in an application's signing workflow must survive version changes, interruptions and incident response. Reliability, migration tooling and audit exports create continuing value. Open schemas and exportable evidence reduce dependence on opaque lock-in.

5. Lower cost per correctly evaluated decision. Reuse of compatible evidence and incremental dependency recomputation can reduce upstream work. Measure this against a fresh-read baseline without relaxing validity requirements. Efficiency can improve gross margin and make broad monitoring economically useful.

The compounding loop is: **more retained integrations** → **more consented, adjudicated cases** → **better semantics and fewer decision errors** → **greater developer trust** → **more retained integrations**. This loop is a hypothesis to measure. Public RPC access, a branded dashboard, model API access, an SDK wrapper or a token burn are individually easy to reproduce.

Competitive position

Reviewed offering	Published focus	Hunch's proposed position
Agent Wormhole	Agent configuration hardening, prompt-boundary checks, x402 quote matching and signed metadata attestations	Financial instrument semantics and dependency-aware evaluation of actual RWA actions
Hookr	Uniswap v4 hook infrastructure, SDK call builders, runtime-pinned manifests, market controls and analysis	Application-level context across issuer, instrument, route and portfolio dependencies, integrated with venue-specific controls
Issuer APIs and Chainlink feeds	Authoritative metadata or reference observations within their stated scope	Combine the sources with action requirements, timing and policy; retain their provenance
Application-built risk logic	Rules tailored to one application's supported workflows	Offer maintained adapters and decision operations that cost less to integrate and maintain than repeated bespoke work

The first two rows describe public product claims reviewed on 15 October 2026, not independent effectiveness assessments. Both offerings overlap with parts of the proposed design. Hunch has no established exclusive category or measured superiority. [6–9]

07 / Become an integration point

One decision contract, explicit adapter coverage

The developer-facing contract should connect discovery, valuation, proposed execution and continuing observation. Schemas, verifier logic and compatibility tests should be inspectable. Maintained data services, operational capacity, continuous monitoring and support provide the commercial value.

The following are candidate integration points, not partnerships or completed adapters:

Application surface	Insertion point	Required additional engineering
Hookr / supported Uniswap v4 routes	After the venue builds a quote and call, before approval and signing	Pin release and hook identity; decode actual calls and approval scope; handle exact accounting and gas requirements
Pons discovery and launch workflows	Asset review, launch state transitions and post-launch observation	Verify factory provenance and launch state; model bonding curves independently of indexed AMM liquidity
RH RFQ / Rialto interfaces	Between executable quote acquisition and wallet confirmation	Verify maker, taker, quote signature, nonce, expiration, settlement contract and actual output constraints
Wallets and smart accounts	Immediately before the local signer or validator boundary	Integrate one specific account stack first; decode batches and session permissions; test bypass paths
Portfolio and vault operators	Valuation refresh and proposed rebalances	Position-aware exposure accounting, reconciliation and durable dependency monitoring

Hookr's current SDK documentation describes some recapture paths requiring gas above Hunch's present 5,000,000 simulation limit. Those paths are currently unsupported by that simulation interface. A future adapter must preserve the real gas requirement and enforce a separately tested resource budget. [7]

RH documentation lists RFQ, AMM, propAMM and orderbook integration surfaces. A listing establishes an available integration direction; it does not establish a Hunch relationship, approved collateral market or production adapter. [1]

Sell an operating service

The commercial package should combine a recurring platform subscription with metered intensive work. Low-cost inspection can remain accessible. Paid value consists of persistent monitored scopes, application policies, continuous invalidation, historical replay, tenant controls and contracted operating capacity.

The first product sold should be a maintained RH stock-token decision service for a precisely supported workflow. Expansion should follow reuse of its semantics and operating system across additional workflows. Bespoke consulting revenue may help fund adapters, but must be reported separately from recurring platform revenue.

Production adoption is demonstrated by paid usage and retention, enforcement inside a real application, and willingness to renew. A large follower count or a collection of nominal integrations does not establish infrastructure demand.

08 / Market and economic logic

A growing asset base is context, not a revenue forecast

RWA.xyz displayed **\$39.18 billion in distributed asset value**, **\$352.65 billion in represented asset value**, and **\$292.40 billion in stablecoin value** in its 15 October 2026 snapshot. Distributed assets can move outside an issuing platform and between wallets, including assets with eligibility controls. Represented assets have different transfer constraints. [10–11]

These are separately defined ecosystem measures. They are not RH volume, agent-executed volume, Hunch's addressable revenue or additive evidence of a single “agentic DeFi market.” No independently verified agentic RWA infrastructure market-size estimate is asserted here.

Hunch's bottom-up market model is the number of applications with a recurring integration problem multiplied by their willingness to pay for the supported service. That willingness remains to be established through commercial use.

Illustrative business sensitivity

All figures below are USD planning assumptions, not published pricing, bookings or forecasts. Each scenario assumes average total monthly service revenue per customer, including any usage charges.

Paying applications	Average monthly revenue each	Monthly service revenue	Annualized service revenue
20	\$500	\$10,000	\$120,000
50	\$1,000	\$50,000	\$600,000
100	\$2,000	\$200,000	\$2,400,000

At \$50,000 monthly service revenue and \$10,000 variable delivery cost, the illustrative gross margin is 80%. That leaves \$40,000 before fixed engineering, security, sales, administration, taxes and reserve needs. Data licensing, RPC and traces, retention, monitoring, support and incident response all belong in the cost model. A target margin without measured costs is not evidence of profitability.

Publish revenue and delivery economics separately from token trading activity. Successful delivery, idempotency and clear failure billing are prerequisites to metering a paid infrastructure service.

A sustainable token policy

Hunch's existing proposal allocates **20% of eligible operating surplus** to monthly buybacks and burns after refunds, payment fees, operating costs, tax provision and a three-month operating reserve. It is inactive. The current calculator returns a zero executable budget. [14]

For illustration, \$50,000 earned and settled service revenue less \$2,000 refunds and payment fees, \$30,000 operating costs and \$3,000 tax provision leaves \$15,000 operating surplus. If reserves need \$5,000, eligible surplus is \$10,000 and the proposed 20% allocation is \$2,000. These inputs form a separate accounting example from the gross-margin scenario above.

Token-sale proceeds, borrowed funds, customer deposits, unearned prepaid credits and unrealized gains are excluded. No eligible surplus means no allocation. Any eventual execution requires verified token and burn semantics, independently controlled treasury authorization, constrained trade execution and reconciled onchain records.

Customers should be able to buy useful software without taking HUNCH price exposure. Burning tokens does not create service demand or guarantee market value. Holding HUNCH does not currently confer equity, revenue ownership or an implemented redemption right. Any future token function must have a separately specified need and implementation.

The economic priority is continued investment in technology and reliable service. A published whitepaper cannot create a durable revenue engine; paid retention and sound delivery costs can substantiate one.

09 / Engineering commitments and acceptance gates

The sequence below defines production work. Dates and delivery commitments require staffing and provider choices; the gates describe the evidence required to make each capability claim.

Stage	Deliverable	Acceptance evidence
1. Reliable instrument state	Versioned asset schemas, authoritative adapter manifests, oracle and sequencer checks, durable observations and dependency invalidation	Correct replay across scheduled changes, source disagreement, provider gaps and reorganizations; no silent use of stale required data
2. Enforced action scope	One fully decoded stock-token swap route, owner policy service, scoped receipts and a local verifier	Substitution and replay tests for every bound field; blocked bypass attempts; independent security review of the signer integration
3. Paid operations	Tenant isolation, limits, usage ledger, incident workflow, service monitoring and historical exports	Capacity measurements, recovery drills, billing reconciliation and a full operating reliability report
4. Repeatable expansion	Additional venue and account adapters, then another instrument model or chain	The same public conformance suite plus adapter-specific invariants; existing customers continue to use and renew the service

Measure decision quality before using strong claims

The benchmark must contain unsafe actions, valid actions and unsupported cases. An evaluator independent of the implementation labels outcomes and documents disagreements. Separate genuine historical cases from constructed adversarial cases. Hold out both contracts and time periods when evaluating generalization.

Proposed release targets include zero automatic authorizations across at least 1,000 critical mutation cases, an adjudicated false-block rate below 5% on supported valid cases, and measured unknown rates by action class. These are initial gates to validate and revise; they are not current results or universal safety guarantees. A system that blocks everything fails the valid-action and usability measures.

Track latency and correctness together. A candidate capacity gate is 25 decisions per second sustained for 15 minutes on a published workload and hardware profile, with p95 response time under two seconds for reusable evidence and under eight seconds when new upstream work is needed. Include upstream failures and capacity rejections in the report. The present service has not established this performance.

Before selling an availability commitment, publish 30 days of measured reliability against the proposed 99.9% request-success objective, with defined treatment of valid policy blocks, unsupported requests, outages and throttling. Policy blocks are successful evaluations; server errors and capacity failures are service failures for valid in-scope requests.

Falsifiable business and moat tests

- If customers can maintain equivalent semantics and operations cheaply with issuer APIs, Hunch's shared-service value is insufficient.
- If new adapters need extensive one-off rewriting, the architecture has weak reuse and limited software margins.
- If retained customers do not use persistent monitoring or decision enforcement, the platform may be an occasional research tool rather than recurring infrastructure.
- If adjudicated evidence does not improve decision quality or cost, the data flywheel is not working.

- If a competing product wins on scope, reliability and integration effort, Hunch must improve those dimensions; stronger launch language cannot substitute.
-

10 / Trust, ownership and operational design

The proposed architecture keeps customer signing authority with the customer. An operator must be able to stop automation, revoke a Hunch key, change its required policy and export decision history without granting Hunch custody.

Publish schema versions, supported adapter scopes, source provenance, verifier code, release fingerprints and a security contact. Keep individual customer policies, prompts, wallet associations and unpublished intents private. Hashing a small or guessable input does not make it anonymous; public records should not expose sensitive manifests by default.

Shared research uses public evidence or explicitly consented, minimized cases. Training permission is separate from service access. The operator can disable shared contribution without losing core protection. Tenant-specific observations must not enter another tenant's graph view through caching or telemetry.

Adapter and key changes require an authenticated release process, rollback plan and revocation path. Public change logs should identify which decisions or instrument models a correction affects. Independent security review should focus on actual enforcement and release paths as well as code quality.

Hunch is independent of Robinhood, Pons, Hookr, Agent Wormhole and the other providers discussed here. Mentioning a documented integration surface does not imply endorsement or a commercial relationship.

11 / Implementation status and evidence

The local v0.5.0 implementation was reviewed on 15 October 2026. Its existing automated suite completed with **37 tests passing, zero failing**. This confirms the tested behaviors; it is not an external audit, a capacity result or a production reliability certification.

Capability	Existing implementation	Work required for the proposed architecture
Contract intelligence	RH chain checks, interface observations, standard proxy patterns, first recognized target and runtime fingerprints	Recursive bounded resolution, verified source, established authority relationships and code-change history
Instrument intelligence	Official address matching, multiplier reconciliation, typed reference calculations, halt/pause and time-window checks	Chainlink round and heartbeat checks, sequencer status, source history and broader instrument semantics
Decision context	REST packet combining selected asset, counter-asset and optional simulation observations	Consistency-aware dependency manifest, durable state, exact-intent decoding and signed scope
Action policy	Fixed profiles; RH assessments require review or block	Authenticated tenant policies, cumulative exposure, supported-action requirements and enforcement at the signer boundary
Execution preflight	Read-only <code>eth_call</code> , optional gas estimate and explicit revert handling	Asset-flow analysis, real route constraints, approvals, supported account batches and state-transition reconciliation
Monitoring	Browser-local saved evidence and watchlists; selected snapshot comparison	Server-side schedules, event indexing, gap recovery, reorganization handling and durable alerts
Integration	Portable JavaScript, TypeScript declarations, Python client, REST and MCP	Published versioned packages, compatibility policy, completed venue/account adapters and deployment assurance
Economics	Inactive surplus-allocation calculator	Tenant billing, metering, settled revenue, treasury controls and verified token execution

The public context reader currently admits one concurrent request per process; the API has a documented limit of 40 requests per minute per IP. Those controls are not evidence of readiness for broad paid integration traffic. Capacity, provider contracts and reliability must be established before accepting such dependence.

Hunch does not currently provide complete permissions analysis, verified RH holder distribution, recursive proxy resolution, signed decision receipts, persistent hosted monitoring, automatic execution clearance or custody. No official HUNCH contract has been published by the project at the time of this review. The reviewed materials do not substantiate paying customers or completed third-party integrations.

The present foundation is real and bounded. The moat proposed here must be earned by shipping and operating the additional architecture.

12 / Sources and reproducibility

Public pages below were retrieved on 15 October 2026 in Toronto (16 October UTC). Market figures use the page's 15 October snapshot. Public claims may change; the source links identify the material used for this review. Competitor statements are attributed descriptions, not audited performance data.

1. **Robinhood Chain — Building with Stock Tokens.** Instrument interfaces, multipliers, venue types and authorized-participant mint/burn access. <https://docs.robinhood.com/chain/building-with-stock-tokens/>
2. **Robinhood Chain — Stock Token APIs.** Registry identity and REST price semantics. <https://docs.robinhood.com/chain/stock-token-apis/>
3. **Robinhood Chain — Oracles & Price Feeds.** Token-price units, heartbeat, pause and sequencer considerations. <https://docs.robinhood.com/chain/oracles-and-price-feeds/>
4. **Robinhood Chain — Account Abstraction.** Smart-account and paymaster integration documentation. <https://docs.robinhood.com/chain/account-abstraction/>
5. **Robinhood Chain — Transaction Finality.** Confirmation stages and application tradeoffs. <https://docs.robinhood.com/chain/transaction-finality/>
6. **Hookr — Product.** Public descriptions of hook infrastructure, analysis and market controls. <https://hookr.fun/>
7. **Hookr — SDK.** Release manifests, call builders, caller checks and gas requirements. <https://hookr.fun/docs/pools/sdk>
8. **Agent Wormhole — Product.** Public descriptions of agent hardening, quote guards and metadata attestations. <https://www.agentwormhole.com/>
9. **Pons — Builders.** Public builder contact and integration-request surface. <https://www.ponsfamily.com/builders>
10. **RWA.xyz — Market overview.** Distributed, represented and stablecoin values displayed for 15 October 2026. <https://app.rwa.xyz/>
11. **RWA.xyz — Tokenization Type.** Definitions of distributed and represented assets. <https://docs.rwa.xyz/frameworks/tokenization-type>
12. **EIP-1898.** Hash-pinned JSON-RPC block selection. <https://eips.ethereum.org/EIPS/eip-1898>
13. **EIP-712.** Typed structured signatures and replay-protection boundary. <https://eips.ethereum.org/EIPS/eip-712>
14. **Hunch — Economics proposal.** Existing inactive 20% surplus allocation and reserve policy. <https://hunchagents.xyz/economics>
15. **Hunch — Integration documentation and current scope.** <https://hunchagents.xyz/docs>

Implementation evidence reviewed: `lib/context.mjs`, `lib/instruments.mjs`, `lib/robinhood.mjs`, `lib/decision-brief.mjs`, `lib/economics.mjs`, `server.mjs`, portable clients and the existing test suite. The dated source-review fingerprints and test result are available at <https://hunchagents.xyz/whitepaper-audit.json>. This records a source review, not an independent security audit.

Hunch · Evidence before execution.

<https://hunchagents.xyz> · <https://x.com/hunchagents> · info@hunchagents.xyz