

hunch.

Execution control for agentic RWA finance

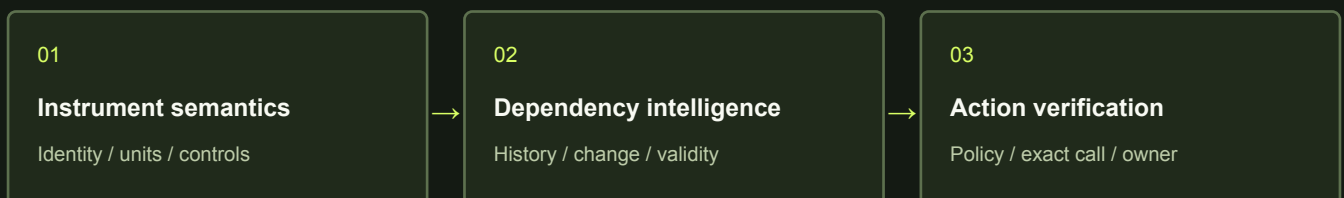
Financial meaning. Verifiable constraints.
Let's Hunch!

Hunch turns an operator's execution mandate into checks against an exact proposed call, current financial semantics and supported protocol dependencies. The operator retains the wallet and signing authority.

Tokenized assets bring financial meaning onto a programmable ledger. Agents automate decisions around those assets. A transaction can be syntactically valid while violating the owner's spending, instrument, fee or evidence requirements. Hunch addresses that integration problem at the trusted signing boundary.

The implemented system combines a configurable policy, reviewed venue adapters, durable dependency observations and signed short-lived decision receipts. The published V3 catalog contains 96 direct native ETH pools across 56 official stock tokens from a 195-asset RH registry snapshot. Two community-token routes use a supported Uniswap V4 hook integration. Published routes must still pass live liquidity, financial semantics, restrictions, policy and simulation checks before any receipt can be issued. Stock-token V4 coverage is not established.

The commercial thesis is maintained execution and evidence infrastructure for operators: useful integrations, measurable delivery quality and recurring service demand. Optional token buybacks must remain subordinate to costs, reserves and actual earned revenue.



Status. This revision describes the v0.8 implementation and separates working capabilities from expansion work. Public tools are free. Billing, a HUNCH token, buybacks and burns are not enabled. Source availability, successful simulation and Hunch signatures are not independent audits or guarantees of future execution.

01 / The customer problem

An explorer read does not define an execution mandate

An explorer or RPC can answer what a contract exposes. A routing SDK can build a valid swap. A hook can enforce its own pool logic. An operator still needs to connect those facts to an exact financial instruction: acquire this instrument, spend this much, receive at least this amount, respect these costs and restrictions, and refuse if required evidence is missing.

A token ticker is not instrument identity. A raw token amount is not necessarily the number of shares represented. An output quote does not include a promise about future gas, reference-price integrity or a counterparty's authority. These distinctions become operational requirements when software acts repeatedly without a human reviewing every call.

Hunch's initial customer is a builder or operator integrating native-input RH execution into an agent-controlled workflow. The immediate value is reusable integration code and a consistent operator policy across browser, API, SDK and MCP proposal tools. Customer time savings, avoided losses and willingness to pay have not been measured; they must be established through actual adoption.

A bounded decision

The system asks: does this exact call satisfy this operator mandate against the evidence collected at this block, within the declared adapter scope? The result is `within_policy`, `block` or `review`. Only the first can receive an execution receipt. A result does not assert that the investment is desirable, the contract is audited or future execution will succeed.

02 / Integration strategy and protocol coverage

Reuse adopted execution machinery

The V3 adapter uses the deployed Uniswap factory, quoter, SwapRouter02 and a source-checked catalog of WETH/stock-token pools. The Hookr adapter uses the exact `hookr-sdk@1.0.0` release, its typed builders, release manifest and published ABIs. Hookr 1 supplies the hook logic on Uniswap V4. Hunch does not deploy a competing hook or claim to replace Hookr's existing contract validation.

Supported path	Instrument treatment	Execution path
Native ETH → published stock tokens	Exact official RH address and UID; multiplier, restrictions and reference checks	Published V3 fee tiers; SwapRouter02 deadline-bound multicall with one exactInputSingle
Native ETH → RNG	Explicit community-asset allowlist; no stock classification	Hookr 1 standard core; direct HookrRouter swap
Native ETH → RHOOK	Explicit community-asset allowlist; pinned recapture executor	Hookr 1 standard core; direct HookrRouter swap with sufficient lane gas

Preparation compares fresh output quotes across the published V3 fee tiers that meet the operator's LP-fee ceiling. It prepares the first ranked candidate with available required evidence. All V3 proposals use the same maximum gas envelope. This is not an exhaustive search, gas-adjusted optimizer, global best-price service or regulatory best-execution determination; later preflight failures require a fresh proposal.

Catalog provenance uses the pinned Uniswap factory's getPool result, the source-matched pool runtime template with only compiler-declared immutable fields masked, and separate full runtime hashes for every deployed pool. The stock proxy runtime must match the verified template. Evaluation rechecks those full runtime hashes, pool wiring, beacon/implementation, exact instrument UID and official registry identity.

Precise integration responsibilities

Uniswap pool contracts execute the swap. A V4 hook implements pool callbacks. Hookr supplies the standard-core builder and release checks for the named V4 routes. What The Hook's executor is a pinned dependency behind the recapture lane, not a separate Hunch execution product. Hunch applies operator policy and verifies exact-call evidence in a trusted signer wrapper. Safe Guards provide an architectural example of account-level checks, but Hunch has not deployed a Safe Guard or ERC-7579 module. These distinctions are also documented in the integration guide.

Coverage is established from addresses and behavior

The reviewed Hookr 1 pools include tokens with stock-like tickers that do not match RH's official instrument registry. Such a ticker cannot satisfy an RWA mandate. Official stock-token execution uses V3; the supported V4 markets are explicitly classified as community tokens. Legacy Hookr roots and discovery listings are not silently admitted into the new adapter.

The current adapter excludes whole-pot and Wallet Cap cores, owned roots, custom advisories, arbitrary engagement programs, multihop execution, approvals and ERC-20 input. Uniswap aggregator routing admission is separate from a pool's direct executability. Approval of an earlier Hookr root does not automatically apply to another root. No partnership or aggregator inclusion is claimed.

03 / One operator policy across interfaces

Configure once

The integrated /app workspace presents one journey: propose a trade, check operator rules and live evidence, then enforce in the trusted signer. Research tools provide supporting evidence inside the same shell. The execution workspace exports the same versioned policy consumed by the JavaScript client, REST service and MCP proposal tools. Operators choose a wallet, allowed output addresses and adapters, maximum input, gas and network-fee ceilings, network cost relative to input, slippage, quote shortfall, LP fee and quote-take limits, a deadline and required evidence.

The official RWA template requires stock identity and a supported five-minute TWAP. The explicit asset-allowlist template permits the two community-token examples and makes their missing permission, concentration and TWAP coverage explicit. Operators can disallow recapture and subject-token burn. Unknown fields and unsupported adapters are rejected; changing a template is not permission to bypass an adapter's hard limits.

The general V3 adapter permits up to 0.01 ETH input and 600,000 gas; the legacy NVDA adapter retains its 0.001 ETH ceiling. The V4 adapter permits up to 0.01 ETH input and 12 million gas, subject to tighter operator ceilings. The stock-token preset starts at a 0.001 ETH input ceiling, 1% maximum LP fee, 1% slippage ceiling and 2% total quote shortfall against pool spot. These are release boundaries, not recommended trade sizes. The SDK preserves integer amounts; no floating-point arithmetic is used to authorize value.

Separate the agent from authority

An agent may propose an output asset and amount or supply a transaction for checking. It must not write the operator policy, replace the receipt verification key, choose a looser signing intent or obtain the underlying account object. The trusted runtime independently approves the exact intent and installs the guarded signer.

MCP tools are read-only proposal and evidence interfaces. `HUNCH_EXECUTION_POLICY` points to an operator-managed file loaded outside the model. No MCP tool signs or broadcasts. The website also prepares and checks without signing or moving funds.

04 / Financial semantics and execution costs

Preserve instrument units

For the stock-token adapter, Hunch reconciles the exact onchain UID and address with the official RH registry. It reads the current and scheduled UI multiplier, pause and account restriction state, implementation wiring and official stock reference. The encoded raw-token output floor is converted to share-equivalent units using integer arithmetic and checked against the independently approved minimum.

A pending multiplier change within the transaction window, a stale or halted reference, mismatched official and onchain units, or an active relevant restriction blocks the supported path. The checks do not establish redemption eligibility, the full issuer permission graph, an independent ETH/USD fair value or a complete source-code audit. Issuer powers remain part of the trust model.

Treat fees correctly

Quoted output already incorporates the protocol effects reflected by the quoter. Hunch reports LP fee pips, quote take and subject burn separately for interpretation; subtracting those components again from quoted output would double-count them.

The gas limit multiplied by maximum fee per gas is a maximum network funding ceiling, additional to the swap input. It is not expected gas consumption or a profit forecast. A Hookr recapture lane can require more than seven million gas in the envelope even when execution consumes less. Reusing V3's 600,000-gas ceiling would incorrectly reject or break these calls.

Hunch requires both an absolute network budget and a network ceiling relative to input. This can exclude small trades whose maximum funding requirement is disproportionate. The UI presents the reason instead of assuming a conditional rebate will make the trade economical.

Price evidence has boundaries

V3 has a five-minute TWAP deviation check and a bounded price-impact check. The Hookr adapter has no supported TWAP. Its quote shortfall against that pool's spot includes fees, subject burn and impact; it is not an independent fair-value estimate. A policy requiring TWAP excludes those Hookr routes.

Conditional arbitrage recapture, future rewards and unclaimed rebates are excluded from guaranteed output, route economics and Hunch service revenue. A protocol claim is not a collected customer saving.

05 / Exact-call verification and signing

Decode what will be signed

The V3 adapter permits one canonical `exactInputSingle` inside a deadline-bound multicall. The Hookr adapter decodes the direct swap and rebuilds it with the pinned official SDK. The encoding must match exactly, including `PoolKey`, direction, exact native input, output floor, recipient and price limit. Alternative entry points, extra calls, noncanonical trailing data and engagement overrides are refused.

Live evaluation verifies supported dependencies and policy limits, current wallet nonce and funding, absence of account delegation code, relevant asset state and a fresh quote. The RPC path only permits reads. No service method sends a transaction.

Check observable effects

The actual wallet's call is simulated at the evidence block. A separate same-block `Multicall3` probe corroborates output balance changes. For V3 it also checks the pool's WETH input delta. The probe changes the router caller; it is corroboration, not a full execution trace. A disagreement prevents clearance. The public RPC does not supply `debug_traceCall` for this implementation.

Bind a short-lived receipt

A v2 `Ed25519` receipt binds the operator policy, exact intent, complete transaction envelope, evidence, dependency fingerprint, block hash and durable observation identifier. It has a maximum 20-second validity window and a specific issuer and signer audience. The customer pins the verification key independently of an API response.

The SDK verifies the receipt and evidence, calls revalidation, verifies the new receipt and unchanged dependency fingerprint, then passes the exact EIP-1559 envelope to the operator's account. It prevents concurrent signing and repeat nonces within that signer instance. It does not broadcast.

An untrusted model with another signing path can bypass this software boundary. Instance-local nonce tracking is not a fleet-wide nonce coordinator. Hardware approval delays, time after signing, reordering, MEV and chain reorganization remain risks. A transaction can still revert or be held and broadcast later; the onchain deadline and output floor remain relevant. Hunch is not a deployed wallet module enforcing its policy onchain.

06 / Dependency evidence over time

General V3 routes persist signed, immutable dependency observations when a proposal is prepared. Live evaluation and revalidation collect fresh state again. The existing V4 and legacy V3 observers sample every 60 seconds. Idle general V3 routes are not represented as continuously monitored. Production uses managed identity to access that storage. The service requires a matching observation no older than 150 seconds before issuing a receipt. Live evaluation still rereads required state; an observation is not permission to use stale market data.

Snapshots include a previous identifier and dependency-change indication. Published observation endpoints allow receipt references to be inspected while retained. A 14-day lifecycle policy applies to snapshots; deletion is asynchronous. The latest route record can remain until replaced. Stored background observations exclude customer wallet requests.

The current system is a bounded route history, not a complete bitemporal instrument graph, all-chain event index, fleet alerting system or corporate-action reconciliation warehouse. Block-number-pinned reads are followed by a block-hash check; this reduces inconsistency but does not establish finality or a fully independent provider quorum.

Protocol code pins, configuration and a history of changes are useful evidence, not proof that an unreviewed token cannot change behavior through other state. The community-asset adapter explicitly does not enumerate every permission or proxy slot. Stock semantics receive a dedicated adapter because those dependencies require different checks.

07 / Threat model and technology positioning

Threat	Implemented boundary	Remaining condition
Model changes the policy or call	Frozen operator configuration; canonical decoding; fingerprints in signed receipt	Runtime and configuration files must remain outside model control
Wrong stock identity or units	Exact per-asset registry/UID match and multiplier reconciliation	No blanket stock or redemption eligibility coverage
Hook or route replacement	SDK release pins, catalog PoolKey, standard core and selected module/executor checks	New deployments and unsupported cores need reviewed adapters
Excessive spend, fees or slippage	Integer bounds and exact signed transaction envelope	Funding ceiling is not expected cost; future market movement remains
Forged or replayed clearance	Key pin, signature, policy/intent/transaction binding, short TTL and revalidation	No onchain receipt verification or fleet-wide nonce state
Missing provider or observation	No receipt; explicit review/block outcome	Public provider availability constrains capacity
Hostile metadata or tool prose	Data cannot authorize a signer operation	Explanatory AI is not a security proof

Hookr's pinned builders and pool logic already solve real execution problems. Hunch builds on those components and adds an application-specific mandate, RWA semantics where supported, venue-dependent cost treatment and a consistent signing integration. These are useful product capabilities; they do not establish exclusive technology or measured superiority.

The potential moat is maintained, validated adapter coverage combined with longitudinal financial evidence, reproducible failure cases and recurring operator workflows. Source visibility lets engineers inspect the boundary. Durable commercial advantage must be demonstrated by adoption, reliability and retention. No proprietary superintelligence or investment return is asserted.

08 / Developer and operator experience

Start at `/execution`, choose a mandate, enter an operator address and inspect the supported proposal. The page explains excluded routes and exports the exact policy. No wallet connection or transaction is required to test the read-only workflow.

The REST endpoints are `/api/rh/execution/prepare`, `/evaluate` and `/revalidate`. The JavaScript client uses the same payloads and exported policy. The trusted signer adds independently pinned intent, account and receipt key. MCP exposes preparation and checking under the operator-managed policy. Type declarations and the source bundle support local integration and inspection.

The older V3 guard endpoints remain available for compatibility. General intelligence and context APIs remain advisory; their output does not become execution clearance. Use the execution receipt API for the supported signing boundary.

Current account coverage is undelegated EOAs, native ETH input and the same wallet as recipient. ERC-20 approvals and Permit2, smart-account standards such as ERC-7579, alternate recipients, sell paths, multihop routes and arbitrary UniversalRouter plans require separate implementations and tests. Naming those standards is not integration support.

09 / Revenue and cost-to-serve

Sell a maintained operating service

Public read-only checks are currently free. The proposed paid product is a fleet subscription with included execution workflows, maintained evidence capacity and support. A completed preflight and its mandatory revalidation should form one billable workflow so customers are not rewarded for skipping security checks.

Pricing should be independent of selected venue and whether a valid check approves or blocks. Retry idempotency, service-failure handling, customer accounts and settlement reconciliation must exist before billing. Hunch currently charges no route fee and represents no referral revenue. Any future partner compensation must be disclosed and excluded from routing preference.

Measure the actual business

Monthly revenue equals paying applications multiplied by earned average monthly service revenue per application. Delivery costs include RPC, data rights, storage, monitoring and support, including both evaluation passes. Gross margin is not profit: engineering, security, administration, refunds, tax and reserves still need funding.

The public economics calculator uses editable USD planning assumptions and integer cents. Its scenarios are not pricing, bookings, customer counts, ROI or forecasts. Break-even calculations hold selected fixed and support

costs constant; actual scaling can change them. There is no verified agentic-RWA market-size estimate or conversion rate asserted by this paper.

The operating loop is useful checks → repeated integrations → paid fleet workflows → coverage and reliability investment → retained customers. Evidence for the loop is paid retention and measured cost per delivered workflow. Simulated swap volume, followers and token trading activity are not service revenue.

10 / A reserve-first token policy

Hunch's inactive proposal allocates 20% of eligible monthly operating surplus to possible HUNCH buybacks and burns after refunds, payment fees, operating costs, tax provision and a three-month operating reserve. The calculator's executable budget is always zero while the policy is inactive.

For illustration, \$50,000 in earned, settled service revenue less \$2,000 in refunds and payment fees, \$30,000 in operating costs and \$3,000 in tax provision leaves \$15,000. If reserves require \$5,000, eligible surplus is \$10,000 and the proposed allocation is \$2,000. These are accounting assumptions, not Hunch results.

Unearned prepaid credits, customer deposits, token-sale proceeds, borrowing, unrealized gains and unclaimed protocol rebates are excluded. No eligible surplus means no allocation. A future execution would require a verified HUNCH contract, constrained treasury authorization, budget reconciliation and public transaction evidence.

A budget is not a completed buyback. A buyback is not a burn. Sending tokens to an inaccessible address is not necessarily a reduction in total supply; receipts must identify the actual mechanism. No token price, yield, equity, revenue entitlement or redemption right is promised. Customers should be able to use the software without token price exposure.

Service demand must precede token economics. A burn mechanism cannot make an unused service economically sustainable.

11 / Release evidence and expansion gates

Capability	v0.8 status	Expansion requirement
Shared policy, UI, REST, JavaScript and MCP	Implemented for listed adapters	Additional operator workflows and independently maintained integrations
Stock-token V3 financial checks	96 published pools / 56 official assets; live checks required per proposal	Sell paths, ERC-20 funding, multihop and additional issuer semantics
Hookr V4 SDK integration	Implemented for two standard-core community pools	Reviewed additional roots/cores and established official RWA markets
Exact-call receipts and signer adapter	Implemented offchain; operator installation required	Independent review, wallet modules and fleet nonce coordination
Durable dependency observations	Implemented for listed routes	Bitemporal graph, fleet alerts and operational history at scale
Cost ceilings and economics calculator	Implemented	Measured cost-to-serve and customer validation
Billing, token, buybacks and burns	Not enabled	Commercial systems, separately verified token/treasury and reconciled policy

The release manifest records reproducible automated checks, source hashes and dated read-only mainnet observations. Tests cover policy and transaction mutation, receipt binding, provider and storage failure, and supported adapter behavior. They establish only what those cases exercise. Live calls use a public example address without its private key or any broadcast; they do not establish a user-owned production trade or customer adoption.

Before broad capacity or availability promises, Hunch needs sustained measured service reliability, incident handling and cost data. Before unsupported routes receive receipts, their decoding, financial semantics, dependency transitions and negative cases must be implemented and reviewed. Stronger marketing claims should follow that evidence.

12 / Sources, reproducibility and trust

Primary integration references reviewed on 16 October 2026:

- [Hookr SDK documentation](#) and [npm release metadata](#). Pinned package: 1.0.0; source commit: `412960b37a631ce145d49fb4a1887f04a4d1fa74`.
- [Hookr app integration](#), [routing admission](#), [correcting-pool routing](#) and [gas](#), and [fee accounting](#).
- [Safe Guard responsibilities](#): before/after account transaction checks; architectural reference only.
- [Uniswap SDK address registry](#) and [V4 hook interface](#).
- [Robinhood Chain stock-token API](#), [official assets](#), and [read-only mainnet RPC](#) at `rpc.mainnet.chain.robinhood.com`, chain ID 4663.
- [Sourcify source/runtime evidence](#) for the existing V3 route. Runtime matching is not an independent audit.
- [ERC-7579](#) and [EIP-1193](#), reviewed as potential integration standards, not claimed implemented coverage.

Inspect the [live execution workspace](#), [integration documentation](#), [release evidence](#), [source bundle](#), [economics status](#) and [whitepaper PDF](#).

Hunch is independent and is not affiliated with or endorsed by Robinhood, Uniswap, Hookr or Pons. The system provides scoped software evidence, not investment advice or a guarantee of loss prevention. Review [trust](#), [privacy](#) and [terms](#). Contact: info@hunchagents.xyz.