Loxley

The marketplace where AI agents buy what they need to trade

Whitepaper v1 · Robinhood Chain 4663 · August 2026

Abstract

Every AI agent that trades today starts from nothing. It has a model, an execution venue, and no knowledge. The strategy it runs was typed into a chat window an hour ago and will be gone tomorrow. Meanwhile the people who actually know how to trade, the quants, the risk managers, the data engineers, have no way to sell what they know to those agents.

Loxley closes that gap. It is a marketplace where anyone publishes a piece of trading intelligence as an onchain artifact, and where an agent discovers it, buys a license, and calls it as a native tool, on its own, without a human in the loop.

Payments are in USDC. Licenses are bought once and owned forever. Execution goes through the user's own Robinhood account. Every call leaves a signed trace anyone can replay. `$LOX` is the coordination asset that secures the market and captures the value it creates.

Contents

Part I · The marketplace

1. The problem
2. What an artifact is
3. The five classes
4. The manifest
5. How an agent uses Loxley
6. The MCP bridge
7. Licensing and payment
8. Seats and price discovery
9. Execution
10. Proof: signed traces and replay
11. Track record and scoring
12. Safety: the guardrail
13. Publishing
14. Fees

Part II · The token

15. What $LOX does
16. Fee capture and buyback
17. Staking and roles

18. Supply and allocation
19. Release schedule
20. Governance
21. What $LOX is not

Part III

22. Roadmap
23. Risks

Part I · The marketplace

1. The problem

In May 2026 Robinhood opened its Trading MCP to third-party agents. Any AI agent can now hold a dedicated account, read a portfolio, and place orders. In July, Robinhood Chain went live, an AI-native L2 built for onchain assets. The infrastructure for agentic finance exists and it is open.

What does not exist is the layer above it.

An agent connected to Robinhood today knows how to place an order. It does not know **which** order. It has no signal feed, no tested strategy, no options pricer, no drawdown limit. The person operating it either builds all of that themselves, which almost nobody can do well, or they let a general-purpose model improvise, which is worse.

At the same time, the supply side is stranded. A quant with a working intraday strategy has three options: run it themselves at small size, get hired by a fund, or sell a newsletter. None of those scale, and none of them reach the agents that are now doing the trading.

The road is built. Nothing drives on it.

Loxley is the missing layer. Not another model, not another broker. A market for the things that make an agent competent.

---

2. What an artifact is

An **artifact** is a self-contained, executable unit of trading intelligence, published onchain, licensed for a one-time price, and callable by an agent.

Three properties define it:

It is code, not advice. An artifact takes explicit inputs and returns a structured output. It has no knowledge of who is calling it, no access to the caller's personal situation, and no opinion about what the caller should do with the result. It is a tool.

It is sealed and deterministic. Artifacts run as WebAssembly modules in a sandbox with no network access, no clock, and no undeclared source of randomness. Given the same inputs, an artifact returns the same output, every time, for everyone. This is what makes verification possible at almost no cost.

It is owned. An artifact is an ERC-721 on Robinhood Chain. The owner receives the revenue from every license sold. Ownership is transferable, so an artifact is an asset in the full sense: it can be sold, inherited, or used as collateral.

A license to use an artifact is a separate token, an ERC-1155, minted to the buyer at purchase. It never expires.

---

3. The five classes

Artifacts are typed. The class determines the call signature, the verification method, and how an agent is allowed to use the output.

SIGNAL

`() → observation`

An observation of the world, timestamped and structured. It does not tell you what to do; it tells you what is.

Examples: cross-exchange perp funding skew, insider filings flow, sector breadth, implied vol term structure, a sentiment score on 8-K filings.

Signals have no market impact, so they have no seat limit. They are the cheapest artifacts on the shelf and often the most heavily used, because an agent can subscribe to a dozen of them and combine them itself.

STRATEGY

`(market_state, params) → order_intent[]`

An executable policy. It reads state and emits order intents. This is the class that most directly replaces the "what should I trade" gap.

Examples: intraday mean reversion on US large caps, sector momentum rotation, vol carry, funding-rate arbitrage.

Strategies are seat-limited, because alpha decays with the capital deployed against it (§8).

TOOL

`(input) → output`

A deterministic function with no view on the market. Pure computation.

Examples: American options pricer, mean-variance optimizer under constraints, expected shortfall calculator, position sizing under a Kelly fraction, correlation matrix estimator.

Tools are the most reusable class and the easiest to verify, since the correct output is a mathematical fact.

POLICY

`(order_intent, portfolio) → allow | resize | deny`

A constraint. It never generates an order; it sits between the agent and the broker and vetoes or shrinks what passes through.

Examples: rolling drawdown budget with progressive de-escalation, per-sector exposure cap, leveraged-product blocklist, maximum notional per hour, expected-shortfall ceiling.

Policies are the safety layer of the entire system and the class we most want to see grow. An agent running three strategies and no policy is a liability. An agent running three strategies behind two policies is a portfolio.

REASONER

`(context) → structured_analysis`

The one non-deterministic class. A reasoner calls a language model with a proprietary prompt and returns a **typed structure**, never free text.

Examples: earnings call transcript parser returning `{guidance_delta, hedging_score, tone_shift, confidence}`; filing summarizer returning risk-factor deltas.

Because reasoners cannot be replayed exactly, they are verified statistically (§10.3) and carry an explicit verification surcharge, disclosed on the listing.

---

4. The manifest

Every artifact publishes an immutable manifest, anchored onchain by its hash. This is the artifact's contract with the market: what it does, what it needs, what it costs.

```
{
  "lip": 1,
  "class": "STRATEGY",
  "id": "lox:0x7a3f91c2",
  "publisher": "0x91c2...4b1e",
  "code_hash": "keccak256(module.wasm)",
  "runtime": "wasm-1.2",
  "determinism": "strict",

  "inputs": [
    { "feed": "rh:nbbo:equities", "resolution_ms": 100 },
    { "artifact": "lox:0x4b19c7", "share_bps": 800 }
  ],
  "output_schema": "order_intent_v2",

  "pricing": {
    "license_usdc": 189,
    "model": "one_time",
    "seats_max": 400,
    "seats_sold": 137,
    "curve_gamma": 1.0
  },

  "risk_class": "moderate",
  "disclosure_hash": "keccak256(disclosure.pdf)"
}
```

Four fields carry most of the weight.

`code_hash` anchors the module. If the code changes, the ID changes. An artifact is never silently mutated under the people who bought it. An update is a new artifact carrying a `supersedes` link, and existing owners choose whether to migrate. The track record does not transfer automatically (§11.5).

`determinism` is either `strict` or `none`. Strict means the module is verifiable by exact replay. This is a binding declaration: a single divergent replay slashes the publisher's bond.

`inputs.artifact` declares a dependency on another artifact and the share of revenue that flows upstream. This is what makes artifacts composable (§13.4).

`pricing` is the commercial contract. `license_usdc` is the base price. `seats_max` is how many licenses the publisher is willing to sell, ever. Once seats are gone, they are gone.

5. How an agent uses Loxley

This is the core of the system, so we walk it end to end. Three steps: **discover**, **license**, **call**. No human touches any of them.

5.1 Discover

The agent queries the registry through a single MCP tool. It filters the way a portfolio manager would, not the way a shopper would.

```
loxley.search({
  class:        ["STRATEGY", "SIGNAL", "POLICY"],
  risk_class:   "moderate",
  min_score:    0.70,
  min_history:  90,          // days of forward record
  max_price:    250,         // USDC
  seats_left:   true,
  universe:     "us_equities",
  max_pairwise_correlation: 0.55
})
```

The registry returns candidates with everything needed to decide: score, forward history length, seats remaining, current price on the curve, dependency tree, incident log, and the full manifest.

Two filters here matter more than they look. `min_history` enforces that the agent only considers artifacts with a real forward record, because backtests never count on Loxley (§11.1). `max_pairwise_correlation` stops the agent from buying five artifacts that are secretly the same bet.

5.2 License

The agent buys. It signs the transaction itself, from its own wallet, with its own USDC.

```
loxley.license({
  artifact: "lox:0x7a3f91c2",
  max_price: 200            // slippage guard on the seat curve
})
```

Two things happen onchain in one transaction: USDC moves from the agent's wallet to the publisher, net of the protocol fee, and an ERC-1155 license is minted to the agent's address. The license never expires.

`max_price` protects the agent against the seat curve moving between the quote and the fill. If the price has risen past the cap, the transaction reverts and nothing is spent.

There is no form, no application, no approval, no whitelist. The registry is permissionless on both sides.

5.3 Call

Once licensed, the artifact **appears inside the agent as a native tool**. The agent does not learn a new API; from its point of view the artifact is indistinguishable from any other tool in its harness.

```
mean_revert_spx_15m({
  symbols: ["AAPL", "MSFT", "NVDA"],
  lookback: 15
```

```
})

→ [
    { symbol: "AAPL", side: "buy",  qty: 40, limit: 212.30, conf: 0.71 },
    { symbol: "NVDA", side: "sell", qty: 12, limit: 883.10, conf: 0.64 }
  ]
```

The agent decides what to do with the output. It might feed it to a POLICY artifact for a risk check, combine it with a SIGNAL, size it with a TOOL, and only then place the order. That composition is the agent's job, not the protocol's.

5.4 What this looks like in practice

A user tells their agent, in plain language:

> *Set me up for moderate-risk income. Cap my drawdown at 8%.*

The agent runs `loxley.search`, gets back candidates, filters for correlation, checks seats and prices against its budget, buys four licenses in a single session, and starts running. Total human involvement: one sentence.

That is the product.

---

6. The MCP bridge

Loxley speaks to agents through **MCP**, the open protocol Robinhood already uses for its own trading tools. Any agent that speaks MCP works with Loxley out of the box: Claude, Codex, Cursor, a custom harness, anything.

The bridge runs **locally**, on the user's machine, next to their agent.

```
┌──────────────┐     MCP      ┌─────────┐     MCP      ┌────────────────────┐
│    Agent     │  ─────────►  │ LOXLEY  │  ─────────►  │ Robinhood          │
│              │              │ BRIDGE  │              │ Trading MCP        │
│              │  ◄─────────  │ (local) │  ◄─────────  │ (agentic account)  │
└──────────────┘              └─────────┘              └────────────────────┘
                                   │
                     licenses      │  signed traces
                                   ▼
                          Robinhood Chain · 4663
```

On every call the bridge does six things:

1. **Resolves the license.** Reads the chain. Does this wallet own a license for this artifact?
2. **Executes.** Runs the WASM module in a sandbox, or calls the model for a REASONER.
3. **Signs a trace.** `(artifact_id, input_hash, output_hash, timestamp, market_snapshot_hash)`.
4. **Applies policies.** Every POLICY artifact the user owns runs last, chained, against the candidate order. A `deny` is terminal.
5. **Routes.** The surviving order goes to the Robinhood MCP.
6. **Anchors.** Trace hashes are published onchain in batches.

6.1 Why local matters

There is no Loxley server on the path of an order. We cannot see your trades, cannot reorder them, cannot censor them, and cannot go down in a way that stops you trading. Your keys, your agent, your orders.

The only Loxley-operated service is the indexer, and it is a convenience: everything in it is reconstructible from onchain events by anyone who wants to run their own.

6.2 Install

```
{
  "mcpServers": {
    "loxley": {
      "command": "npx",
      "args": ["-y", "@loxley/mcp"],
      "env": { "LOXLEY_WALLET": "<your wallet>" }
    }
  }
}
```

That is the entire integration.

---

7. Licensing and payment

7.1 One price, once

You buy a license once. You own it forever. There is no subscription, no renewal, no metering, and no cut of whatever you make with it.

This is a deliberate choice and it deserves an explanation, because most marketplaces go the other way.

A metered model means the marketplace has to watch what you do with what you bought. That requires either trusting a reporting layer or instrumenting the agent, and both are worse for the buyer. A revenue-share model means the marketplace has a claim on your P&L, which means it has to measure your P&L, which means it has to see your positions. Loxley sees none of that, by design, and cannot: the bridge runs on your machine.

One price, once, is the only model that is compatible with an agent that operates without supervision and a protocol that never sees your trades.

7.2 Settlement

Everything settles in USDC on Robinhood Chain. A single transaction moves the money and mints the license.

```
buyer pays:      p
publisher gets:  p × (1 − φ)
protocol gets:   p × φ
```

where `p` is the current seat price and `φ` is the protocol fee (§14).

Publishers are paid **instantly, at purchase**, straight to their wallet. There is no escrow, no vesting of revenue, no waiting period, no minimum payout threshold.

7.3 What a license grants

- The right to call the artifact, unlimited, forever
- The right to see the artifact's full manifest and dependency tree
- The right to the artifact's ongoing updates if the publisher issues a `supersedes` version and offers migration
- The right to replay any trace the artifact produced for you, and to challenge it

A license does **not** grant the right to redistribute the code or resell the artifact.

8. Seats and price discovery

Here is the part most marketplaces get wrong, and it is the reason copy-trading products die within eighteen months.

8.1 Alpha has a capacity

A strategy that returns 30% on $1M does not return 30% on $500M. Market impact eats the edge. The standard square-root impact model, well supported empirically, says:

```
α_net(C) = α₀ − λ · (C / ADV)^δ            δ ≈ 0.5
```

where `C` is the total capital deployed against the strategy across everyone using it, `ADV` is average daily volume of the traded universe, and `λ` is an impact coefficient.

Total profit is `Π(C) = C · α_net(C)`, which is maximized at a finite capacity:

```
C* = ADV · [ α₀ / (λ(1 + δ)) ]^(1/δ)
```

For $\delta = 0.5$, `C* = ADV · [α₀ / 1.5λ]²`. **Past `C*`, every additional dollar deployed destroys value for everyone already deployed.**

8.2 Why an unlimited marketplace is a trap

If a publisher can sell unlimited licenses, they will. They collect on every sale and bear none of the degradation. Buyer number 500 ruins the returns of buyers 1 through 499, and the publisher is paid for it.

That is not a hypothetical. It is the observed life cycle of every signal group and copy-trading service that has ever existed.

8.3 Seats

Loxley makes capacity explicit and finite. Every seat-limited artifact declares `seats_max`. When they are sold, the artifact is closed. Permanently.

The price of a seat rises as the shelf empties:

```
                 p₀
p(u) =  ─────────────────          u = seats_sold / seats_max ,   γ ∈ [0.5, 2]
             (1 − u)^γ
```

At $\gamma = 1$ and `p₀` = $189:

Seats sold	Price
0%	$189
25%	$252
50%	$378
75%	$756
90%	$1,890

Three effects, all of them good:

Early buyers are rewarded. The person who found an artifact before it was obvious pays the least, which is the correct incentive for a discovery market.

Late buyers pay for scarcity, and are told why. Nobody is surprised. The curve is published in the manifest.

Oversubscription becomes impossible. As `u → 1`, price → ∞. You cannot drown a strategy on Loxley, no matter how much money wants in.

And the publisher captures the rent their own edge creates, rather than watching it get arbitraged away by whoever bought first and resold access.

8.4 Seats are enforced honestly

`seats_max` is a claim, and claims need a cost. The publisher posts a bond proportional to their declared capacity (§17.2).

The protocol continuously compares realized slippage on attested fills against the impact model in the manifest. If realized slippage significantly exceeds predicted slippage over a rolling 20 trading-day window (t-test, α = 0.01):

```
seats_max ← seats_max · min( 1 , [ slippage_predicted / slippage_realized ]^(1/δ) )
```

Remaining seats are withdrawn from sale, the event is written permanently into the artifact's incident log, and the publisher's score takes a hit.

Overstating capacity costs money. That is what turns `seats_max` from marketing into information.

---

9. Execution

Loxley does not execute anything. It never holds your assets, never takes custody, never touches your broker credentials.

Orders produced by artifacts are routed by the local bridge to **your own Robinhood agentic account**, using your own connection. Robinhood executes exactly as it would for any agent connected to its MCP. Nothing changes on the broker side, and every control Robinhood gives you, including disconnecting the agent entirely, still works and still overrides everything Loxley does.

This separation is deliberate and it is not only technical. Robinhood provides execution, custody and compliance. Loxley provides content and its measurement. Neither takes on the other's risk, and Loxley never becomes an intermediary between you and your money.

9.1 Your risk policy comes first

Before any order leaves the bridge, it passes through the policy the user signed:

```
Π = ( max_notional, max_drawdown, ES_limit, sector_caps, blocklist, venues )
```

This is enforced locally, by the bridge, independent of any POLICY artifact the user bought. Artifacts are the refinement. This is the floor.

The bridge also caps position sizing below full Kelly, computed on the artifact's attested moments:

```
  f* = μ / σ²            f_allowed = min( f_user , 0.25 · f* )
```

Quarter-Kelly is deliberate. Full Kelly maximizes asymptotic growth but produces drawdowns no human tolerates and is extremely sensitive to estimation error in μ. **No artifact can ask for more than a quarter of Kelly, ever.** That is not configurable.

---

10. Proof: signed traces and replay

A marketplace of trading intelligence lives or dies on one question: can you prove what an artifact actually did?

10.1 Every call leaves a trace

```
trace = (
  artifact_id,
  input_hash,
  output_hash,
  timestamp_ns,
  market_snapshot_hash
)
```

Signed by the bridge, co-signed by attestors, hash anchored onchain. Traces are the raw material of every track record on Loxley. **No performance number on this platform is ever a publisher's claim. It is always computed from traces.**

10.2 Deterministic replay

For any artifact declared `determinism: strict`, verification is exact, not statistical:

```
  Valid(trace)  ⇔  H( Exec_wasm( code_hash, input_hash ) ) = output_hash
```

Anyone pulls a trace, re-runs the module, compares a hash. It costs milliseconds. There is no threshold to tune, no false positive rate, no calibration. It either matches or it does not.

This is why we force artifacts into a deterministic sandbox in the first place. It makes honesty cheap to prove and dishonesty impossible to hide.

Sampling and security. Let `p` be the probability a given trace is challenged, `n` the number an attacker falsifies, `B` their bond, `G` their gain per falsification. They cheat only if:

```
  n·G  >  [ 1 − (1 − p)ⁿ ] · B
```

Because detecting a **single** false trace slashes the **entire** bond, the right side approaches `B` quickly while the left grows linearly. With p = 2% and bonds sized at `B ≥ 50 × G_max`, falsification is never profitable, and the network spends 2% of execution cost on verification instead of 100%.

Challenge window: 48 hours. A successful challenge pays the challenger 40% of the slashed bond, sends 40% to the insurance reserve, and burns 20%.

10.3 Reasoners

Non-deterministic by nature, so exact replay is impossible. Instead, the protocol re-runs the same prompt on the same snapshot `m` times and tests whether the contested output falls inside the honest distribution:

$$z = \left(\hat{s} - \hat{\mu}_m \right) / \left(\hat{\sigma}_m \cdot \sqrt{1 + 1/m} \right) \qquad \text{reject if } |z| > z_{1-\alpha/2}$$

Because reasoners return typed structures rather than free text, we test the **conclusion**, not the tokens. This avoids the failure mode of token-level fingerprinting, where two honest runs of the same model diverge because of temperature, batching or kernel non-determinism.

The cost of this verification is billed to the caller and shown on the listing. We do not hide it in the protocol's overhead.

10.4 Market data

All of this rests on two parties agreeing on what the market looked like at time *t*. The `market_snapshot_hash` is a Merkle commitment over a signed NBBO snapshot at 100ms resolution, published by a staked attestor set.

Artifacts whose edge lives below 100ms cannot be reliably verified this way. They are flagged `verification: best_effort` on their listing, and buyers see that flag before they buy.

---

11. Track record and scoring

11.1 Backtests never count

A publisher may attach a backtest. It **never** enters the score, and it is displayed separately, clearly labelled as unverified.

The reason is simple: a backtest is an object someone iterated on until it looked good. There is no way to distinguish a genuine edge from a curve fit after the fact, and pretending otherwise is how every strategy marketplace before this one filled up with noise.

Loxley scores **forward performance only**, from the artifact's mint date, computed from attested traces.

The cost is real: a new artifact starts at zero and has to earn its record in public. That is the point. A score you cannot buy or fabricate is the only score worth displaying.

11.2 Deflated Sharpe Ratio

Raw Sharpe is biased upward when the artifact was selected from many candidates. On a marketplace with thousands of listings this bias is severe. Loxley uses the **Deflated Sharpe Ratio** (Bailey & López de Prado):

$$\mathrm{DSR} = \Phi \left[\frac{\left(\hat{SR} - SR^* \right) \cdot \sqrt{T - 1}}{\sqrt{1 - \gamma_3 \cdot \hat{SR} + \left((\gamma_4 - 1)/4 \right) \cdot \hat{SR}^2}} \right]$$

with `SR*`, the expected maximum over `N` independent trials:

$$SR^* = \sqrt{V[\hat{SR}_n]} \cdot \left[(1 - \gamma) \cdot \Phi^{-1}(1 - 1/N) + \gamma \cdot \Phi^{-1}(1 - 1/(N \cdot e)) \right]$$

$\gamma \approx 0.5772$ is Euler-Mascheroni. `T` is the number of observations, γ_3 and γ_4 the skew and kurtosis of returns.

`N` is estimated by the protocol, not declared by the publisher. It counts: artifacts by the same publisher, `supersedes` variants of the same family, and correlated artifacts in the same class over the same universe (return correlation above 0.7 on the common window).

This closes the obvious attack. Publishing twenty variants and listing only the winner does not work, because all twenty count toward `N` and the survivor's DSR collapses accordingly.

DSR reads as a probability. **DSR = 0.95 means: given how many attempts were made, there is a 95% chance this edge is real.** The registry displays DSR, never raw Sharpe.

11.3 Persistence

A high number over a short window means nothing. Persistence is the rank correlation of returns across disjoint sub-periods:

```
P = ρ_Spearman( r_odd , r_even )        P ∈ [−1, 1]
```

Negative persistence, where an artifact works precisely when it previously did not, is the signature of overfitting and is penalized harder than zero.

11.4 The composite score

```
R = w₁·DSR + w₂·tanh(P) + w₃·log₁₀(1 + V) − w₄·(seats_sold/seats_max)^1.5 − w₅·incidents

(w₁, w₂, w₃, w₄, w₅) = (0.45, 0.20, 0.15, 0.15, 0.05)
```

`V` is cumulative license revenue. `incidents` counts lost challenges, capacity reductions, and manifest violations.

The seat term is deliberately a **penalty**. As an artifact fills up, its expected value to the next buyer falls, and the score reflects that. The market ranking works against the seller and for the buyer, always.

11.5 Versions

If a track record transferred automatically from v1 to v2, a publisher could build a reputation and then swap in anything. If it never transferred, nobody would fix a bug.

Loxley transfers it partially, as a function of how much the code changed:

```
Score(v₂) ← Score(v₁) · exp( − ρ · d(v₁, v₂) )        ρ = 2.5
```

`d` is a normalized distance over the module's control-flow graph. A bug fix keeps most of the score. A rewrite starts near zero.

11.6 Delisted artifacts still count

If losing artifacts vanished from the reference distributions, the registry's average would drift upward and mediocre artifacts would look good by comparison. Loxley keeps delisted artifacts in the statistics for 36 months, with their performance up to delisting.

Every listing shows two numbers: its score, and its percentile **including the dead**.

---

12. Safety: the guardrail

Buyers are running purchased code against real money. That demands more than a rating.

12.1 Every submission is screened

Before an artifact can be listed, it goes through the guardrail, a static and dynamic analysis pass built in-house. It looks for:

- Attempted network access, filesystem access, or any escape from the sandbox
- Undeclared entropy sources, which break determinism and therefore verification
- Manifest violations: declared inputs versus actual inputs, declared determinism versus observed behaviour
- Order patterns consistent with wash trading or self-dealing
- Known malicious patterns and obfuscation

12.2 Flagged means quarantined, and quarantined means read

Anything the guardrail flags does not get listed. It goes into quarantine, and **a person reads the code** before it moves either way. Not a model, not a heuristic, a human being.

Nothing that has been flagged reaches the shelf on an automated decision.

12.3 The loop closes

Every manual review is fed back into the guardrail as a new rule or a new test case. The system gets stronger and more autonomous over time, with fewer false positives as the boundary sharpens.

We do not claim zero false positives. No detection system has ever had zero false positives, and anyone who tells you otherwise is selling something. We claim the rate goes down and that a human is on the other side of every one of them.

12.4 The insurance reserve

Funded by 10% of protocol fees plus 8% of the initial token supply.

Covers: a proven replay divergence, execution that violates the manifest, sandbox escape, a POLICY artifact that fails to apply, and any protocol-level failure.

Does not cover: market risk. An honest artifact that loses money is not a claim. It is the product working exactly as described.

That boundary is written into the contract and is not governable. It is the single most important clause in the protocol, because it is the one that a governance vote chasing adoption would try to soften first.

Payouts are capped per claim at 2% of the reserve, and rationed pro rata if an epoch is oversubscribed:

```
payoutⱼ = min(Lⱼ, cap) · min( 1 , Reserve / Σₖ min(Lₖ, cap) )
```

The responsible actor's slashed bond pays first. The reserve only tops up.

---

13. Publishing

13.1 What it takes

Compile to WASM. Write the manifest. Post the bond. Mint.

There is no application, no interview, no revenue minimum, no company required, and no identity disclosure. A pseudonymous quant with a verified forward record competes on equal terms with an established shop, and after six months the record is the only thing that matters.

13.2 You set the price

One number, in USDC. You choose `p₀`, you choose `seats_max`, you choose the curve steepness `γ`. Nobody negotiates with you, nobody takes a cut of your upside, and nobody can force a discount.

You are paid instantly, at every sale, directly to your wallet.

13.3 What you keep

The code, if you want it sealed. Loxley never publishes your source. The registry stores the module hash and the compiled artifact, and buyers get the right to call it, never to read it.

You can also publish openly if you prefer. Some of the best artifacts on any marketplace are open source, monetized by maintenance rather than by secrecy, because a frozen strategy decays and copying the code copies a photograph of a moving object.

13.4 Composition

An artifact can depend on another. Revenue flows upstream automatically along the dependency graph, split according to the `share_bps` declared in each manifest.

```
Maximum depth               D ≤ 4
Cumulative upstream share   Σ share_bps ≤ 5000    (publisher keeps ≥ 50%)
Cycles                      rejected at mint by topological sort
Upstream failure            propagates, downstream suspends automatically
```

This is what lets a curator assemble four artifacts into a coherent bundle, sell it as one product, and have every underlying publisher paid correctly without a single negotiation.

---

14. Fees

5% of every license sale. That is the entire fee schedule.

3% if the publisher stakes `$LOX` (§17.1).

There is nothing else. No listing fee, no minting fee beyond gas, no monthly cost, no take on the buyer's side, no cut of the buyer's trading profits, no payout threshold, no withdrawal fee.

Why 5% and not 30%, and why not 10%? Because the fee has to sit below the cost of leaving. Any marketplace fee creates an incentive for two parties who have found each other to settle privately. At 30% that incentive is overwhelming and every serious publisher eventually builds their own storefront. At 5%, going around the protocol means giving up the registry, the verified track record, the seat mechanism, the guardrail, and the agent distribution, and almost nobody finds that trade worthwhile.

The fee is charged on the sale, in USDC, at the moment of purchase. It is the only revenue the protocol has, and it is what funds everything in Part II.

Part II · The token

15. What `$LOX` does

Before anything else, the boundary: `$LOX` **is not the currency of the marketplace.** Every license is priced and paid in USDC. Every publisher is paid in USDC. A buyer can use Loxley for years and never touch `$LOX`.

That is not an oversight, it is the design. A market whose unit of account swings 40% in a quarter is unusable. Nobody can price a strategy at "4 LOX" and mean it. Forcing a token into the payment path is the most common way protocols make themselves unusable in exchange for a chart.

`$LOX` does four things, and each of them is a real mechanism rather than a claim.

It lowers your fee. Publishers who stake pay 3% instead of 5%. For anyone selling seriously, this is a direct, calculable return on holding the token.

It backs capacity claims. A publisher who declares 400 seats posts a bond in `$LOX`. If realized slippage proves the claim false, the bond is slashed. The token is what makes `seats_max` an honest number instead of a marketing figure.

It secures verification. Attestors, verifiers and executors post bonds. A single proven false trace costs the entire bond. This is what makes cheating irrational rather than merely discouraged.

It captures the fee. Protocol revenue buys `$LOX` on the open market, programmatically. Stakers receive part of it, the rest is burned.

---

16. Fee capture and buyback

16.1 Where the money goes

```
Protocol fee on every license sale (100%)
     ├── 70%  →  Buyback bucket
     ├── 20%  →  Treasury (USDC, operations, never sells tokens)
     └── 10%  →  Insurance reserve
```

The treasury is held in USDC, not in `$LOX`. The protocol does not fund itself by selling its own token, which is the failure mode that ends most projects: as the price falls, the runway shortens, which forces more selling, which lowers the price.

16.2 The buyback

When the bucket crosses its threshold, **anyone** can trigger execution. There is no treasurer deciding when to act, no announcement, no discretion. The cadence is public and auditable by anyone reading the chain.

Executed volume is then split:

```
Buyback (100%)
     ├── 65%  →  distributed to stakers
     └── 35%  →  burned permanently
```

The burn dial is governable within [20%, 80%] and nothing else about the flow is.

Net effect per dollar of license revenue:

```
buy pressure   = 5% × 70%         = 3.5 cents
to stakers     = 3.5% × 65%       = 2.28 cents
burned         = 3.5% × 35%       = 1.22 cents
```

16.3 Executed without being farmed

A permissionless buyback with a public bucket balance and a naive market order is free money for anyone watching the mempool: front-run, let the protocol buy at the degraded price, sell into it. The protocol would systematically buy at the worst price of the window.

Loxley executes in randomized tranches, bounded by an oracle:

```
Size    :  aₖ = (B / K) · (1 + εₖ)          εₖ ~ U(−0.3, 0.3)
Timing  :  tₖ = t₀ + (k/K)·W + ξₖ           ξₖ ~ U(0, W/K)
Price   :  reverts if P_exec > (1 + ψ)·TWAP₃₀ₘ     ψ = 100 bps
```

Randomness is derived from `keccak256(blockhash(n−1) ‖ bucketNonce)`.

The result: an attacker knows neither the size nor the block of any tranche, so their capital is tied up across an uncertain window for an uncertain return. And the oracle bound makes the attack self-defeating, because pushing price past 100 bps over the 30-minute TWAP reverts the transaction and they paid gas for nothing.

The caller stays permissionless and receives a gas rebate plus 5%, so someone always does it.

---

17. Staking and roles

Role	Stakes	Why	Earns
Staker	Freely, 1 week to 2 years	Aligns holding with protocol revenue	65% of buyback flow
Publisher	Bond ∝ declared seats	Backs the capacity claim	Fee cut from 5% to 3%
Attestor	Fixed tier	Covers false attestations	Attestation fees
Verifier	Fixed tier	Covers frivolous challenges	40% of slashed bonds
Curator	Bonding curve per artifact	Signals quality, takes risk	Share of fees on artifacts curated early

17.1 The publisher discount

The most concrete utility in the system. A publisher staking above their tier threshold pays 3% instead of 5%.

For a publisher selling $200,000 of licenses a year, that is $4,000 saved annually against a stake they still own. The token pays for itself through use, not through speculation, which is the only kind of token demand that survives a bear market.

17.2 The capacity bond

```
S ≥ β · seats_max · p₀              β = 5%
```

A publisher declaring 400 seats at $189 posts `0.05 × 400 × 189 = $3,780` worth of `$LOX`.

High enough that inflating `seats_max` to farm early sales is expensive. Low enough that an independent quant is not shut out. The bond is returned when the artifact is fully sold or voluntarily delisted in good standing.

17.3 Staking weight

```
W = S · ( 1 + ln( 1 + D / D₀ ) )        D₀ = 1 week
```

Logarithmic, not linear. A two-year lock weighs about 4.7× a one-week lock, not 104×. A linear duration multiplier hands governance to a small number of patient whales and empties it of everyone else.

Locked positions are transferable NFTs, so committed capital keeps secondary liquidity without unlocking supply.

17.4 Slashing, and its limits

Every slashable cause is **provable by computation**. Never by vote.

Cause	Proof	Slash
Replay divergence	Re-run produces a different hash	100%
False attestation	Trace invalidated by replay	100%
Overstated capacity	Slippage t-test, α = 0.01, 20 days	25% + seats reduced
Manifest violation	Declared inputs ≠ observed inputs	50% + delisting
Frivolous challenge	Replay confirms the trace	5% of challenger bond

Distribution of a slash: 40% to the discoverer, 40% to the insurance reserve, 20% burned.

No governance vote can slash. No multisig can slash. If it cannot be proven by a replay, it cannot be slashed. This is the only way to have slashing without arbitrary power, and it is why we can afford to have it at all.

---

18. Supply and allocation

18.1 Fixed supply

1,000,000,000 `$LOX`. Minted once, at genesis. No inflation, ever.

There is no minter contract. There is no emission schedule. There is no governance path to create more tokens. The supply can only go down, through the burn.

We considered a gated emission model and rejected it. Every inflation schedule, however well designed, is a promise that a future governance can renegotiate. A fixed cap enforced by the absence of a mint function is not a promise, it is a fact about the code.

18.2 Allocation

Bucket	%	`$LOX`	Unlocked at TGE	Schedule
Ecosystem & builders	40%	400,000,000	5.0%	Released in seasons against verified marketplace volume
Treasury	20%	200,000,000	5.0%	25% at TGE, remainder linear over 12 months
Core contributors	17%	170,000,000	0%	12-month cliff, then 24-month linear vest
Liquidity	15%	150,000,000	15.0%	LP position locked 10 years

Insurance reserve	8%	80,000,000	0%		Locked, released only against proven claims
Circulating at TGE		**250,000,000**	**25.0%**		

No venture allocation. No private round. No advisor tokens.

Core contributors are on a twelve-month cliff. Not a month, not a quarter. Nobody working on this protocol can sell anything for a year, and the addresses are published at genesis so you can check.

18.3 The ecosystem bucket

40% is the largest bucket by a wide margin, and it exists for one reason: **the catalogue is the product, and the people who build it are the ones taking the risk.**

Nothing in this bucket is released on a calendar. Every season is gated against verified marketplace volume, which means license sales that actually happened, to buyers who actually paid, in USDC.

Distribution within a season:

Destination	Share	Basis
Publisher rewards	45%	Matched against verified license revenue, capped per artifact
Buyer rebates	30%	Rebate on fees paid, for active agents
Trust layer	15%	Attestors, verifiers, executors
Curation	10%	Bonding curve rewards, decaying over time

Note what is not on that list: there is **no allocation for passive staking**. Stakers are paid from the buyback, which is paid from real revenue, from day one and forever. The yield is small at the start. That is honest, and it is the only number that will not lie to you.

18.4 Circulating schedule

	TGE	+3m	+6m	+12m	+24m	+36m
Circulating	25.0%	28.5%	32.0%	40.0%	56.0%	68.0%

Excludes ecosystem seasons, which are volume-gated and therefore unpredictable by construction. If the marketplace does no volume, the ecosystem bucket does not release. That is the point.

---

19. Release schedule

The ecosystem bucket, being 40% of supply, deserves its own explanation of exactly how it opens.

```
Season_e  =  min ( B_e , θ · V_e )
```

- `B_e` is the season's cap from the published schedule
- `V_e` is verified license revenue for the season, in USD

- `θ = 1.0`

Read plainly: **the protocol never releases more, in dollar terms, than the marketplace earned.**

Season caps, declining:

Season (quarter)	1	2	3	4	5	6	7	8
Cap (M $L0X)	22	20	18	16	14	12	10	8

Cumulative over eight quarters: 120M of the 400M bucket. The remainder carries forward into later seasons under the same gate. If volume never materializes, the tokens are never released and simply remain locked.

19.1 Why gate on license revenue

Because it is the hardest number on the platform to fake.

To manufacture license revenue you have to actually pay the protocol fee on every wash sale, and the fee is charged in USDC at the moment of purchase. The attacker pays 5% of every dollar they pretend to earn, and the fee they pay feeds the buyback, which buys the token they are trying to farm. The loop is closed against them.

Compare that to gating on downloads, users, TVL, or transaction count, all of which are free to fabricate.

---

20. Governance

Narrow and slow, on purpose.

Governable, by staker vote:

Parameter	Bounds	Delay
Protocol fee	[200, 800] bps	7 days
Staked fee discount	[0, 400] bps	7 days
Burn share of buyback	[20%, 80%]	7 days
Score weights $w_1 \ldots w_5$	±50% of initial	14 days
Seat curve default γ	[0.5, 2.0]	7 days
Bond coefficient β	[2%, 15%]	14 days

Not governable, by anyone, ever:

- **The supply cap.** There is no mint function. This is not a rule, it is an absence.
- **The slashing catalogue.** No new cause can be added. The list in §17.4 is fixed at deployment.
- **The insurance boundary.** "Does not cover market risk" is in the contract and immutable. This is the clause a vote chasing growth would come for first, which is exactly why it is out of reach.
- **Forward-only scoring.** No governance can make backtests count.
- **The quarter-Kelly ceiling.** No artifact will ever be permitted to request more.

21. What `$LOX` is not

Stated plainly, because the absence of these statements is usually where the trouble starts.

`$LOX` is not a claim on publisher revenue. Publishers are paid in USDC and keep 95% or 97% of every sale; token holders have no claim on that.

`$LOX` is not a claim on the treasury. There is no redemption mechanism and no path to one.

`$LOX` is not equity, not a security offering, and not a promise of profit from the efforts of others.

`$LOX` does not give holders any control over individual artifacts, their pricing, their listings, or their scores.

What it does is described in §15, and everything it does is implemented in contracts you can read. Value accrues through an open-market buyback that anyone can observe, trigger, and front-run if they want to. There is no distribution, no dividend, and no claim.

Part III

22. Roadmap

Now. Registry live. Five artifact classes. One-time licensing in USDC. MCP bridge. Robinhood execution. Signed traces and deterministic replay. Forward-only scoring. Guardrail and quarantine review. Seat curves.

Next. Federated attestor set. Sealed execution in TEE for publishers who want the code never to leave an enclave. Curation market. Bundles as first-class objects.

After that. Performance-linked licensing as an optional pricing model alongside the one-time license, for publishers who prefer it. Multi-broker adapters. Transfer of remaining governance to stakers.

The token launches after the marketplace works, not before. A protocol whose token precedes its product funds the product by selling the token. A protocol whose product precedes its token funds the token with the product. Only the second has a buyback that means anything.

23. Risks

A whitepaper that does not name its failure modes is a brochure.

The core bet may be wrong. It is possible that any strategy which works stops working the moment it is sold, whatever the seat mechanism does. Loxley is built on the belief that capacity pricing solves this. That belief could be false.

Cold start. An empty marketplace attracts nobody; a marketplace full of noise destroys trust. The first months are an execution risk that no mechanism solves.

Broker concentration. The protocol is built around Robinhood's MCP. The execution layer is abstracted in the bridge and therefore portable, but until a second adapter exists, there is a single commercial dependency.

Verification has limits. Deterministic replay is exact for strict artifacts. It cannot detect look-ahead baked into a model's training data, an asset universe selected with hindsight, or a threshold calibrated on what already worked. Forward-only scoring erodes these over time, but it does not help the person buying in month one.

Regulation. An artifact that produces personalized investment recommendations for compensation may make its publisher an investment adviser in some jurisdictions. Loxley cannot settle that question for publishers. Artifacts are tools: they produce deterministic outputs on caller-supplied inputs with no knowledge of the caller's circumstances, and every manifest anchors a disclosure. That is a design choice, not a legal opinion.

Sandbox escape. WASM sandboxing is strong and it is not perfect. The guardrail, the bonds and the insurance reserve exist for this, and they are finite.

Disclaimer

This document describes protocol mechanics. It is not an offer, a solicitation, investment advice, or a promise of future value. `$LOX` confers no claim on revenue or assets beyond what is implemented in the published contracts.

Trading carries the risk of total loss, and agentic trading concentrates that risk by removing a human from each decision. The artifacts described here are tools, not recommendations. No score, attestation, replay or verification in this protocol constitutes a guarantee of performance. They establish that a computation ran as specified. They say nothing about whether its output will be profitable.

Robinhood Markets, Inc. is not affiliated with Loxley in any way. References to its products are descriptive and imply no endorsement, sponsorship, or partnership.