Anti Hunter — Staking & Token Distribution Service Spec (Base)

Version 0.9 • Draft for implementation • Date: 2026-02-13 • Network: Base Mainnet

Purpose

Design a production-grade staking and token distribution system for **$ANTIHUNTER** on Base that enforces current tokenomics: fixed lock terms (30/60/90/120d), weighted rewards, 25% early-exit penalty to rewards pool, optional rollover boost, and transparent accounting.

---

1) Goals, Constraints, and Product Principles

Goals

- Reward long-term stakers with deterministic, auditable rules.
- Route protocol fees + penalties back to stakers (compounding loop).
- Avoid fake APY framing; rewards can be zero when vault is unfunded.
- Support per-deposit maturities (no global unlock cliffs).

Non-goals (v1)

- No cross-chain staking.
- No leveraged staking derivatives.
- No discretionary operator intervention in reward math.

---

2) Onchain Architecture (Recommended)

1. **AntiHunterStakingV1** — lock positions, claim, early exit, rollover.
2. **RewardsVaultV1** — reward inventory + cumulative reward-per-weight accounting.
3. **FeeRouterV1** — routes protocol fees + penalties into rewards vault.
4. **BuybackExecutorV1** (optional v1.1) — deterministic PnL→buyback logic.

Contract trust model

- Use **multisig admin** for privileged actions.
- Prefer immutable core math; if upgradeable, use timelocked UUPS + guardian pause.
- Roles: `DEFAULT_ADMIN_ROLE`, `PAUSER_ROLE`, `REWARD_NOTIFIER_ROLE`, `FEE_ROUTER_ROLE`.

3) Position Model (Per-Deposit)

Each stake creates an independent position record (optionally mapped to NFT token IDs later).

```
struct Position {
  address owner;
  uint256 principal;           // amount staked
  uint64  lockStart;
  uint32  lockDurationDays;    // 30/60/90/120
  uint32  baseWeightBps;       // 10000/14000/19000/25000
  uint32  rolloverBoostBps;    // +2000 max, capped by total 30000
  uint64  unlockAt;
  uint256 rewardDebt;          // standard accRewardPerWeight debt model
  bool    closed;
}
```

---

4) Tokenomics Rules (Encoded)

Lock Term	Weight	Notes
30 days	1.0x (10,000 bps)	Base weight
60 days	1.4x (14,000 bps)	Mid-term preference
90 days	1.9x (19,000 bps)	Strong conviction
120 days	2.5x (25,000 bps)	Max base weight

- **Rollover boost:** if re-lock happens within 24h of maturity, add +0.2x (2,000 bps), hard cap effective weight at 3.0x (30,000 bps).
- **Early exit:** allowed any time with 25% principal penalty, 100% routed to RewardsVault.
- **Rewards source:** protocol fees in $ANTIHUNTER + penalties + optional treasury allocations.
- **No guaranteed APY:** UI should display realized + projected ranges, never fixed promises.

---

5) Reward Accounting Design

Use standard cumulative accounting to avoid O(n) loops:

```
accRewardPerWeight (scaled 1e18)
positionPending = (positionEffectiveWeight * accRewardPerWeight / 1e18) - rewardDebt
```

- On stake/rollover/exit/claim: settle pending rewards first.
- On rewards notification: increase `accRewardPerWeight` proportionally to active total weight.
- If no active weight, rewards remain in vault until next distribution update.

6) Core Function Surface (v1)

```
// staking
stake(uint256 amount, uint32 lockDurationDays)
rollover(uint256 positionId, uint32 newLockDurationDays)
claim(uint256 positionId)
claimAll(uint256[] positionIds)
exitEarly(uint256 positionId)          // principal − 25% penalty
withdrawMatured(uint256 positionId)

// views
pendingRewards(uint256 positionId) view returns (uint256)
positionInfo(uint256 positionId) view returns (Position)
totalEffectiveWeight() view returns (uint256)

// vault/admin
notifyReward(uint256 amount)
routePenalty(uint256 amount)
pause() / unpause()
```

---

7) Event Schema (Must-Have for Indexing)

```
event Staked(address indexed user, uint256 indexed positionId, uint256 amount, uint32 lockI
event Rollover(address indexed user, uint256 indexed oldPositionId, uint256 indexed newPos
event Claimed(address indexed user, uint256 indexed positionId, uint256 amount);
event EarlyExit(address indexed user, uint256 indexed positionId, uint256 principalReturne
event MatureWithdraw(address indexed user, uint256 indexed positionId, uint256 principal);
event RewardNotified(uint256 amount, uint256 newAccRewardPerWeight);
event PenaltyRouted(uint256 amount);
event Paused(address by);
event Unpaused(address by);
```

---

8) Security Controls & Invariants

Controls

- `ReentrancyGuard` on all mutating payout paths.
- `Pausable` for emergency freeze.
- Explicit token allowlist (only $ANTIHUNTER for v1).
- No external callback hooks in reward flow.

Invariants (for property tests)

- Principal conservation: total staked + withdrawn + penalized equals deposits (mod rounding).
- No user can claim > funded rewards allocated to weight.

- Penalty always equals exactly 25% of principal on early exit.
- Effective weight never exceeds 30,000 bps cap.
- Position cannot be withdrawn twice.

---

9) Offchain System (Do Not Overbuild)

- **Indexer:** Goldsky/Subgraph for positions, APR views, realized rewards, TVL by term.
- **RPC/Node:** Alchemy Base endpoints.
- **Automation:** OpenZeppelin Defender relayers for reward notifications and scheduled ops.
- **Alerts:** failed tx spikes, pause events, vault underfunding, abnormal early exits.

---

10) API + Frontend Requirements

Read API

- `GET /positions/:address`
- `GET /positions/:id`
- `GET /rewards/pending/:address`
- `GET /metrics` (TVL, term split, daily claims, penalty inflow)
- `GET /health` (vault liquidity, notifier status)

UI copy guardrails

- No “guaranteed APY” language.
- Show lock term, weight, penalties, and reward source transparently.
- Require explicit confirmation on early exit penalty.

---

11) Launch Plan

Phase 0 — Spec freeze (2–4 days)

- Finalize state, math, events, and admin model.

Phase 1 — Build + test (1–2 weeks)

- Solidity implementation (Foundry).
- Unit + fuzz + invariant tests.
- Base Sepolia integration tests.

Phase 2 — Security hardening (1–2 weeks)

- External audit / competitive review.
- Fixes + retests.

- Deploy scripts + runbook freeze.

Phase 3 — Mainnet guarded launch

- Initial stake cap + staged enablement.
- 24–72h monitoring period before cap expansion.

---

12) Open Decisions (Need Product Call)

1. Single reward token forever vs future multi-reward extension?
2. Should matured positions auto-roll by default (opt-in) or always manual?
3. Upgradeability preference: immutable core vs timelocked upgrades?
4. Do we expose position NFTs publicly in v1 or keep internal IDs?

---

13) Recommended Default Decisions

- **Single reward token** in v1 ($ANTIHUNTER only).
- **Manual rollover** with explicit +0.2x boost within 24h.
- **Timelocked upgradeable** architecture with multisig + pause guardian.
- **Internal IDs first**, NFT wrappers in v1.1 if needed.

---

14) Implementation Checklist

- [ ] Solidity contracts + storage gap planning
- [ ] Foundry unit tests
- [ ] Fuzz/invariant suite
- [ ] Deployment scripts (Sepolia + Mainnet)
- [ ] Subgraph/indexer + API
- [ ] Ops runbook (pause, recovery, incident protocol)
- [ ] Security review + fixes
- [ ] Public docs + risk disclosures

Prepared for Anti Hunter / Anti Fund product + engineering planning.