DEXTER WHITEPAPER

Pass the eval. Trade firm capital.

The first on-chain prop trading firm — settlement, custody, and payouts published to Base. Keep 90% of profits.

2026-05-25

DEXTER.MARKET

TABLE OF CONTENTS

01 DEXTER

02 ABOUT DEXTER

03 WHY DEXTER

04 ARCHITECTURE

05 GATEWAY AND READ MODEL

06 RUNTIME AND PUBLICATION

07 MARKET ENGINE

08 ORDER FLOW AND MARKET STATES

09 ADMISSION, SIGNATURES, AND ORDER COMMITMENTS

10 FUNDING, FEES, AND RISK CARRY

11 LIQUIDATIONS AND PROTECTION ROUTINES

12 PRICE LAYER

13 SOURCES AND FALLBACK

14 MARK CONSTRUCTION AND EXECUTABLE PRICE

15 SESSIONS AND DEGRADED STATES

16 VAULT AND TREASURY

17 STATE ROOTS, WITHDRAWALS, AND SETTLEMENT FLOW

18 TREASURY, INSURANCE, AND INCOME ACCOUNTING

19 SECURITY

20 CONTROLS AND PERMISSIONS

21 WITHDRAWAL PROOFS, DISPUTES, AND ROOT CHALLENGES

22 MONITORING AND RECOVERY

23 REVENUE MODEL

24 DXTR TOKEN AND PARTICIPATION

25 CHALLENGE AND AIRDROP

26 ENTRY AND ACCESS

27 SCORING, AIRDROP, AND REWARDS

28 REFERRALS AND BONUS CREDITS

29 WHAT COMES NEXT

30 DEXTER IN ONE VIEW

31 GLOSSARY

CHAPTER 01

DEXTER

#

What Dexter is

Dexter is a perpetual DEX on Base with a prop-firm layer welded on top. The trading side is self-custodial, cross-margined, oracle-priced, and competitive on fees. The prop layer is the wedge: traders skip the capital question by buying a fixed-price pack ($49 to $299), pass a coded set of

rules (+10% target, -4% daily floor, -8% total floor), and unlock a real funded account whose profits split 90 / 10 in their favor.

That bundle does not exist anywhere else. Traditional prop firms run closed simulators with discretionary cashiers. Other perp DEXes end at the order book. Airdrop farms reward shallow wallet activity. Dexter pays for one thing — proven trading skill, measured against a public matching engine and settled on Basescan.

The execution stack is hybrid for a reason. Matching, funding, and liquidations run off-chain in a single ordered loop so fills clear at CEX latency. Collateral, payouts, and treasury permissions live in audited contracts with Merkle-proof release. Traders see CEX-class fills; every dollar of collateral and every basis point of fees stays reconcilable on-chain by anyone with a block explorer.

The economics are funded by pack fees and trading volume — not by inflated incentive math. Funded traders carry no further capital at risk after the upfront pack, and Dexter's treasury sits behind the same drawdown limits every challenger faces.

#

The funded challenge

One pack purchase, one evaluation window, one funded line. There are four pack sizes and a free demo — no monthly subscription, no resets, no escalating challenge ladder. The fee is the entire upfront cost; the allocation is the entire downside Dexter underwrites.

- **Starter — $49 fee, $5,000 funded equity.** Smallest skin in the game. Best for proving the rules before sizing up.
- **Growth — $99 fee, $10,000 funded equity.** Standard tier for active traders.

- **Swing — $199 fee, $25,000 funded equity.** The most popular pack. Sized for directional swings.
- **Elite — $299 fee, $50,000 funded equity.** Largest funded line. For traders running real position size.
- **Free demo — $25,000 simulated equity, 14 days.** No KYC, no payout. Same matching engine, same rules — built to validate strategy before paying.

Every pack carries the same rule set, in plain numbers: a **+10% profit target** on the funded equity inside a 30-day window, a **-4% daily drawdown floor** measured close-to-close, and a **-8% total drawdown floor** from the high-water mark. Breach either floor and the challenge closes; clear the target without a breach and the funded account provisions automatically.

Profit share is fixed at **90% to the trader, 10% to the protocol** — no scaling tiers, no time decay, no "elevated" tier behind a paywall. Payouts settle in USDC on Base to the wallet you signed up with, typically within 24 hours of a request once one-time KYC clears. There is no escrow, no payout cap, and no withdrawal queue.

1-Step packs run target and risk rules in a single phase. 2-Step packs split the evaluation into a target phase plus a consistency phase — same fee, same payout share. Both formats publish identical numbers; the choice is style, not economics.

#

Custody and execution

Two things have to be true at once: fills clear at venue latency, and every dollar stays auditable on a public block explorer. Dexter satisfies both by splitting the runtime from custody.

Contracts hold the money. Collateral, payouts, and treasury permissions live in audited Base contracts. The collateral vault, insurance reserve, treasury, and reward pool are distinct addresses — failures on one rail do not cascade. There is no off-chain "Dexter wallet" you have to trust between deposit and withdrawal.

Releases are proof-gated. Every withdrawal references a periodically published Merkle root that proves the requested amount is solvent against the latest snapshot and consistent with the trader's recorded PnL. The challenge window catches inconsistencies before funds move.

The matching loop runs off-chain. Order admission, funding accrual, and liquidations execute in a single ordered loop with cryptographic batch receipts. The contract verifies each receipt before it affects on-chain balances. Off-chain speed, on-chain proof.

Marks come from Pyth, cross-checked. Primary feed is Pyth Hermes; Chainlink, RedStone, and Uniswap V3 TWAPs back-check it. Stale, deviating, or paused feeds are rejected before they reach the book, and the venue's current state — fully open, tightened, or waiting — is published on the trade page in real time.

KYC is one-time and only at payout. Demo accounts never trigger it. Paid traders only see it the first time they request a real on-chain withdrawal; after that, payouts settle automatically. Jurisdiction blocks (IR / KP / SY / CU) and OFAC / EU / UK sanctions lists are enforced at the cashier, not at the order book.

#

Getting started

Three steps, no waiting list.

1. **Connect and pick.** Wallet connect at app.dexter.market/dex/airdrop. Choose the free 14-day $25K demo or one of the four paid packs. Pay in USDC, ETH, or BTC on Base.
2. **Trade the rules.** Open perps at app.dexter.market/dex. Push equity past +10% inside 30 days, stay inside -4% daily and -8% total drawdown floors.
3. **Pass and get paid.** Funded line provisions in the same UI on pass — no separate onboarding. Request payout, clear one-time KYC, and USDC lands on Base inside 24 hours.

No interviews, no quarterly evaluations, no fee resets, no paperwork between the pack purchase and the funded line. Rules in, payouts out.

Product guides

Understand Dexter before you trade it.

These guides explain the product in plain language and connect the public venue to the mechanics behind execution, custody, withdrawals, and season-based participation.

01 **Hybrid perpetual venue**
See why Dexter keeps execution coordination off-chain while custody and release boundaries stay contract-bound.

02 **Contract-bound custody**
Understand where collateral lives, how treasury balances stay separate, and why the runtime does not redefine custody.

03 **Proof-gated withdrawals**
Follow the release path from published roots to proof checks and the challenge rules that protect settlement.

04 **Challenge + airdrop**
Learn how challenge access, season scoring, referrals, and feedback combine into Dexter's challenge-first model and linked airdrop record.

CHAPTER 02

ABOUT DEXTER

#

The Dexter thesis

Dexter is the first on-chain prop trading firm. The trading side is a self-custodial perpetual DEX on Base — cross-margined, oracle-marked, mid-double-digit-millisecond fills. The prop layer sits on top: pay between $49 and $299 for a pack, prove a +10% return inside 30 days under -4% daily

and -8% total drawdown floors, and trade a $5K–$50K funded account that splits realized PnL 90 / 10 in your favor.

What is new is not the perp venue and not the funded-account model — both exist elsewhere. What is new is that they are the *same product*, on the same matching engine, with every challenge entry, pass event, payout, and referral cash transfer keyed to a wallet and posted to a Base address. The whole journey from "I want to prove I can trade" to "I got paid for trading" finishes on a public block explorer, in USDC, with no off-chain cashier in the middle.

The point is not to be the biggest perp DEX. The point is that the path from skill to payout is mechanical, coded, and verifiable — and that nobody else has shipped that exact bundle yet.

#

How the product flows

From the user side the loop is one screen. Connect a wallet, pick a pack size or the free 14-day $25K demo, and trade the same matching engine the leaderboard runs on. The challenge engine watches the equity curve in the background — daily close-to-close, total drawdown from high-water mark, target hit. When the curve clears +10% inside drawdown, the funded line provisions in the same UI; the next withdrawal flow is real USDC on Base.

Under the hood, four engines run independently so each can be audited and replaced on its own. The **market engine** handles order admission, funding, and liquidations. The **challenge engine** handles rule checks, evaluation windows, and tier upgrades. The **payout pipeline** handles one-time KYC, the 90 / 10 split, and the on-chain transfer. The **referral and leaderboard ledger** tracks the 8% base + 4% activity bonus per pack, and the per-season reward bracket. Each layer publishes its own state — an oracle degradation cannot silently corrupt a payout, and a referral void cannot move a leaderboard rank.

#

Who it is for

Dexter serves three distinct audiences, and the product is shaped to make each one's path obvious from the landing page.

- **Skilled traders without size.** Pay $49–$299 for a pack, prove the rules, trade a $5K–$50K funded line. The pack fee is the entire cost of capital; the 90% profit share is the entire upside.
- **Traders with their own size.** Skip the challenge, trade the public DEX with the same cross-margin and oracle marks, and compete on the season leaderboard for ranked cash brackets at #1, #2–3, #4–10, #11–50, and top 100.
- **Builders and community members.** No trading required. Share a referral code and earn 8% of every paid pack at floor, plus a 4% activity bonus when the referee passes — paid in USDC, transparent in the on-chain ledger.

The three paths share one operating principle: results are visible on a public leaderboard, cash flows through a Base address, and the airdrop record (40% of the fixed 1B DXTR supply) is weighted by behavior that costs the user something real — pack fee, trading discipline, referrals brought in — not by who got there first.

#

What Dexter is not

The categories Dexter is not in matter as much as the one it is in.

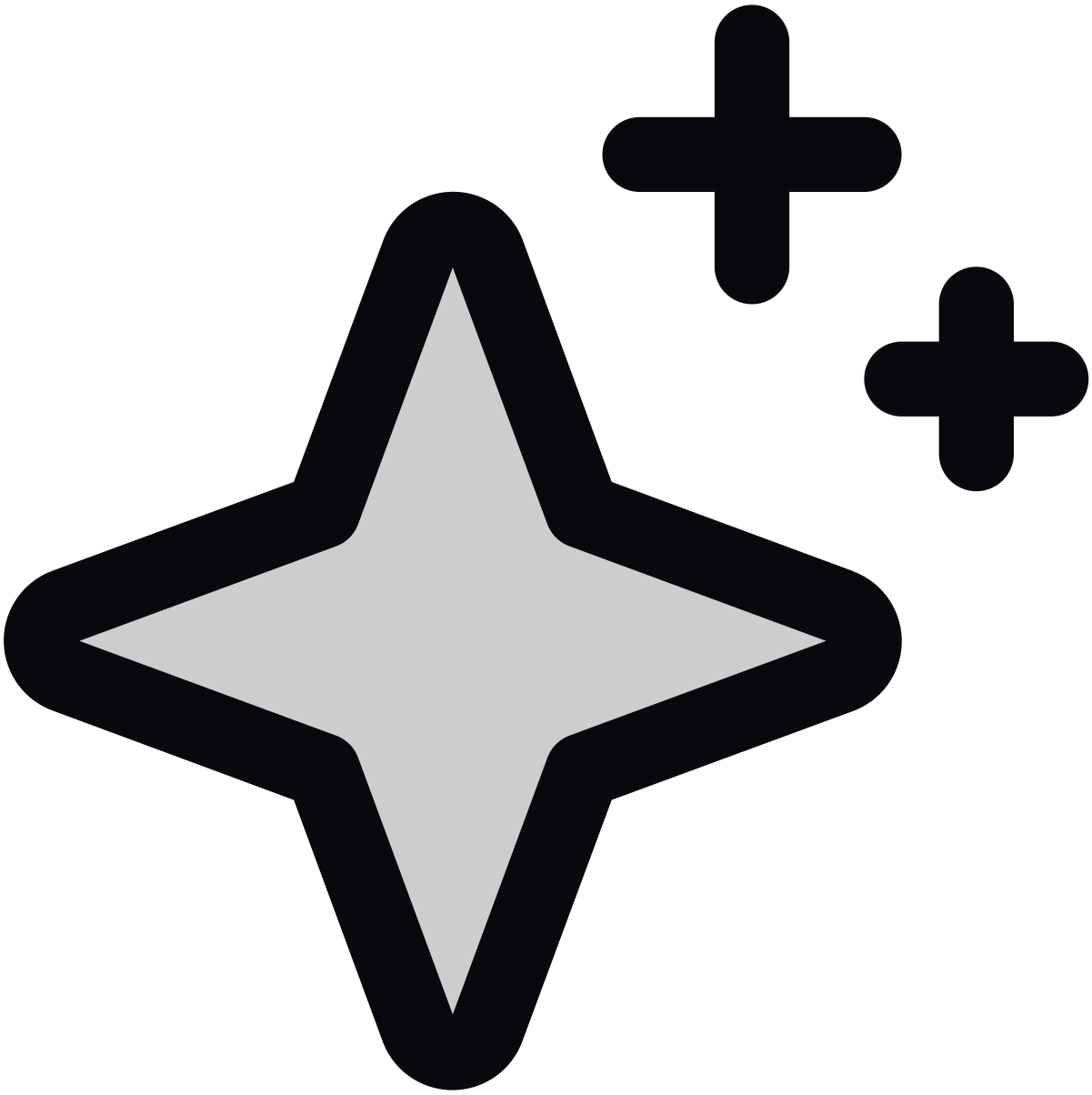
- **Not a yield product.** Holding a pack does not entitle the holder to a share of someone else's PnL. There is no staking APR, no "vault deposits earn X%", no rebate paid for showing up.
- **Not a fund.** Nobody manages capital on behalf of anyone else. The funded account is the trader's account; Dexter's downside is bounded by the drawdown floor coded into the rules.
- **Not a copy-trading desk.** Demo accounts do not pay out. Copying another funded account is rule-flagged and voids the payout. Each funded line earns by its own decisions.

- **Not a farming program.** Pack purchase does not auto-vest tokens, and there is no fixed snapshot date to game. The airdrop record is the season-long behavior record — PnL on a paid pack, referrals brought in.
- **Not a black box.** Pack fees, the +10% / -4% / -8% rule set, the 90 / 10 split, the 8% + 4% referral schedule, the leaderboard formula — every number lives in this whitepaper. If a cashier number disagrees with the doc, the doc governs.
- **Not jurisdiction-blind.** Iran, North Korea, Syria, Cuba, and OFAC / EU / UK sanctions lists are blocked at the cashier — never silently, never partially.

CHAPTER 03

WHY DEXTER

#



Versus traditional prop firms

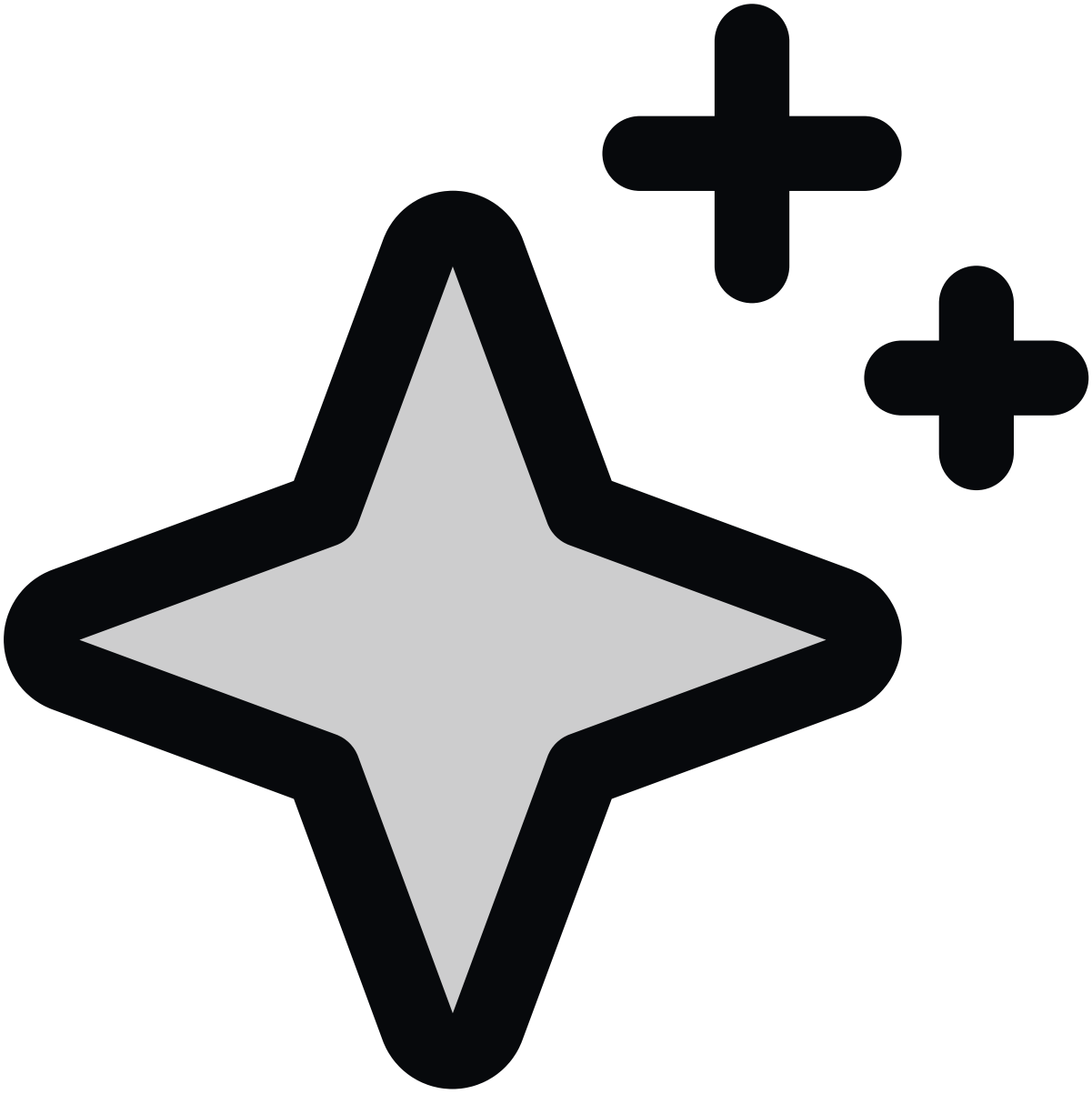
Traditional prop firms run a closed cashier. The challenge is taken on a simulator the firm controls. The rules are written so passing is rare and re-tries are paid. Payouts arrive on a bank-wire schedule of the firm's choosing, often weeks after a request, and capped per cycle. Disputed runs

go to a discretionary review the trader cannot audit. The firm's economics depend on most fees never converting to a payout obligation — so the rulebook quietly tilts against the trader.

Dexter inverts every friction.

- **Execution.** Challenge fills price off the same Pyth-backed oracle the public DEX uses. There is no separate simulator the firm can tilt. Slippage on a funded account is identical to slippage on a paid account.
- **Rules.** +10% target, -4% daily, -8% total — coded into the challenge engine, published in this whitepaper, never moved by discretion. No "judge" decides if a candle wick counts; the engine does.
- **Payouts.** USDC on Base, settled inside 24 hours of a clean request, no cap per cycle. Every transfer is a Basescan transaction. The leaderboard's "Verified Rewards" counter reconciles against those transactions hourly.
- **Cost structure.** One pack fee covers one challenge — no escalating challenge ladder, no monthly subscription, no "reset" fee. 90% to the trader, 10% to the protocol, every cycle.

#



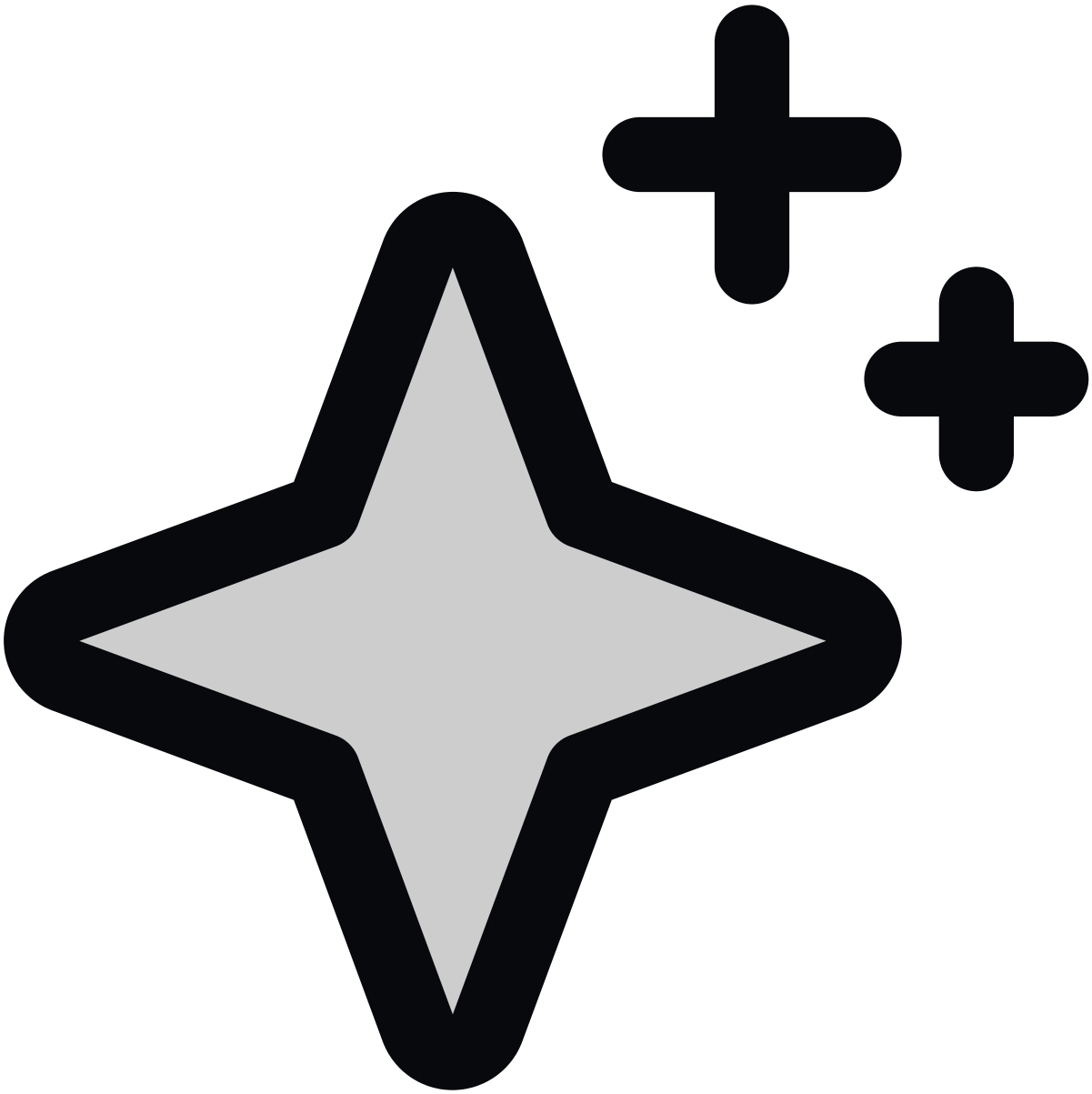
Versus other perp DEXes

Other perp DEXes compete on listing speed, leverage, and farming rebates. Their pitch ends at the order book — the user is treated as flow that must be subsidized to keep coming back. Dexter's order book is competitive (same cross-margin, same oracle-driven liquidation logic, same maker rebate schedule), but the venue does not stop at the book. Above it sit the challenge engine, the season leaderboard with ranked cash brackets, and the referral rail paying 8% base + 4% activity bonus per pack.

The economics are the load-bearing difference. Farming rebates inflate volume for a quarter and collapse the moment incentives expire — and the rebate budget eventually taxes real traders. Dexter pays nothing for showing up. It pays a fixed 90% share of realized trading PnL on funded accounts, and a fixed 8% + 4% of pack revenue to referrers who bring those traders in. The reward is tied to product output (trader skill, referee quality) instead of incentive math that always reverts.

The trader is not flow. The trader is a candidate for the prop layer, with a coded path from a $49 pack to a $50K funded line and a measurable airdrop record.

#



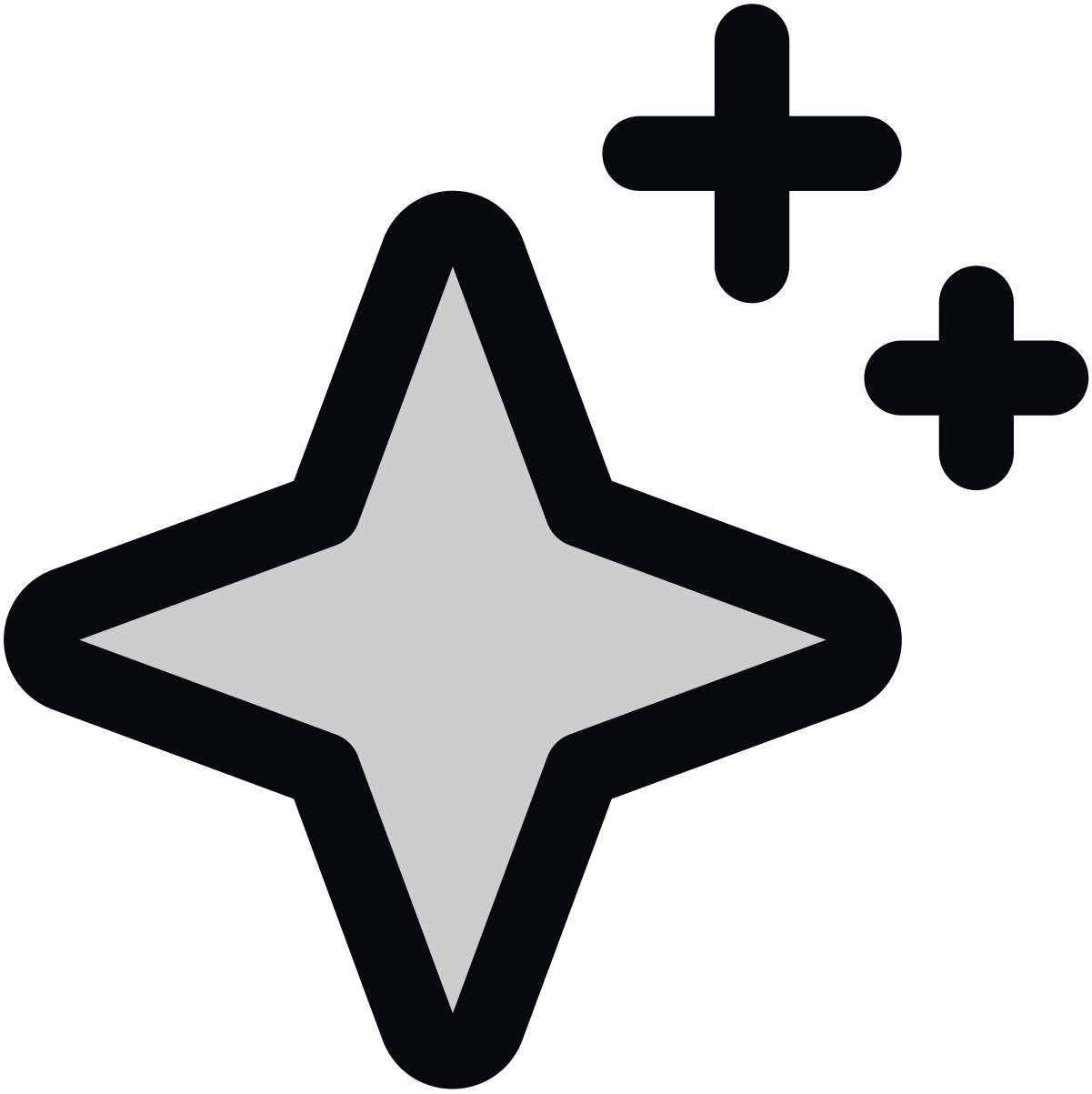
Versus airdrop farms

Most airdrop campaigns reward shallow behavior — wallet count, transaction count, faucet claim count. The outcome is the same story every cycle: sybil farms inflate the eligibility list, the snapshot rewards the cheapest signal, and real users collect a fraction of what they were promised. The producers leave, the farmers cash out, the chart goes vertical for one day and bleeds for a year.

Dexter's airdrop record is built on costly behavior, not on shallow signals. Every leaderboard point comes from realized PnL on a paid pack ($49 minimum) or from a referee who paid for a pack. Self-referrals, multi-account farming, and bot signups are filtered at the cashier — not at the snapshot — and the void reason is public on the referral ledger. The top three on the leaderboard each season have their trade history reviewed before the cash leaves treasury, and withdrawals above $5,000 trigger an additional human check.

The discipline is simple: **40% of the fixed 1B DXTR supply** is reserved for the airdrop record, and the record is weighted by skill PnL plus referred pack volume. Buying a pack costs real money. Trading the rules costs real attention. Bringing in real referees costs real time. The cash arrives in the hands of the people who actually produced the volume — there is no other path in.

#



How the product stays honest

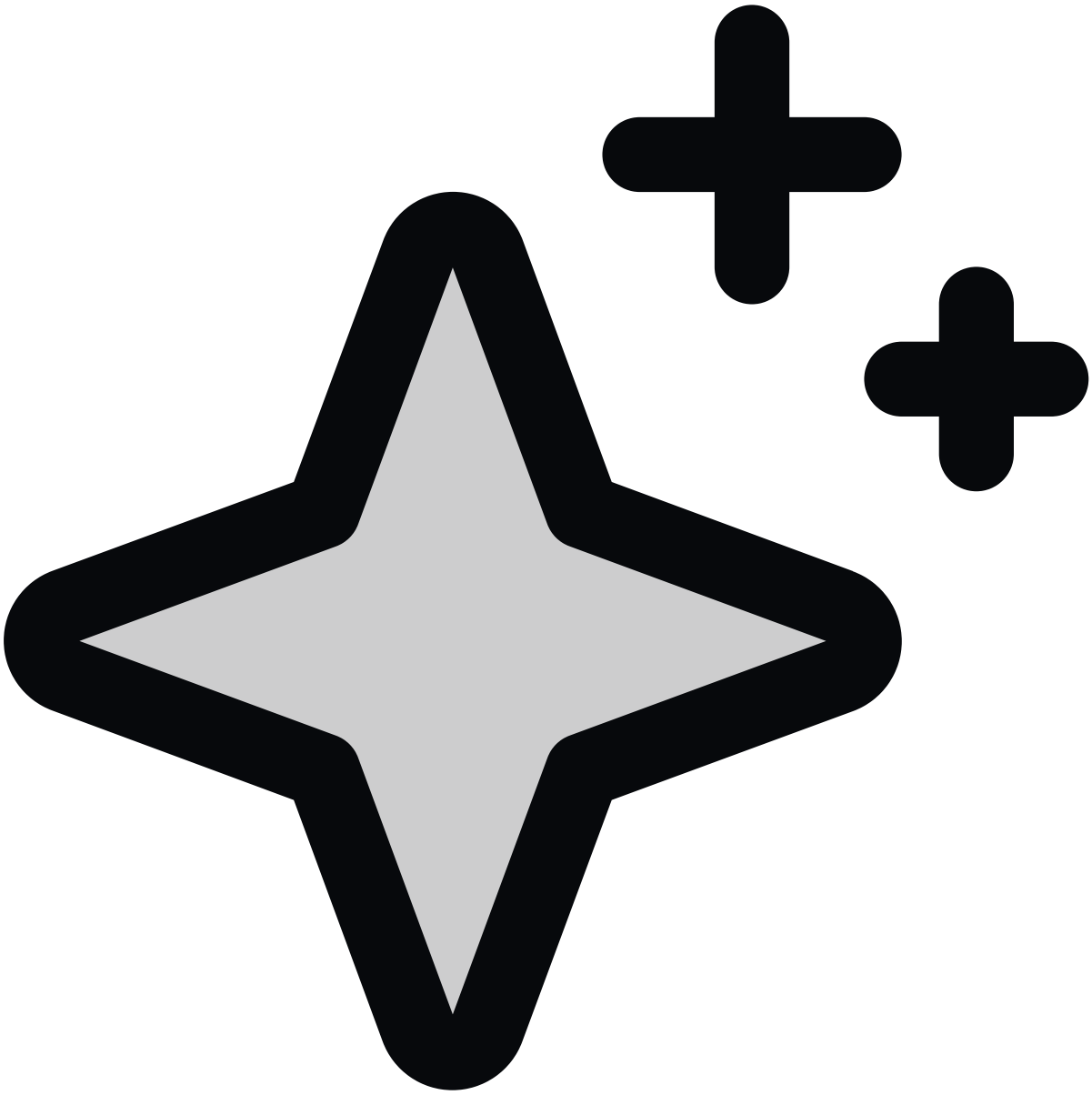
Three rails enforce the difference, and each one is a verifiable artifact instead of a marketing claim.

- **Open-mark execution.** Every challenge fill prices off the same Pyth oracle as the public order book. A funded trader cannot be quietly given worse fills than a paid account — the runtime emits a fill receipt the contract verifies before the balance updates.
- **On-chain payout pipeline.** Every payout transaction is anchored to a Base address. The leaderboard "Verified Rewards" total reconciles hourly against Basescan — the figure is a sum across real transactions, not a marketing display.

- **Human-in-the-loop risk review.** Every podium finish (top 3 per season) and every withdrawal above $5,000 is reviewed by operations before the transaction goes out. The review covers position-size hygiene, rule-edge cases, and copy-trading detection.

That is the underlying contract with the user. Rules are coded, cash flow is verifiable, the cashier is not a closed book. What distinguishes Dexter from the next perp DEX, the next prop firm, and the next airdrop farm is not a single feature — it is that every claim in this whitepaper can be checked against an on-chain transaction.

#



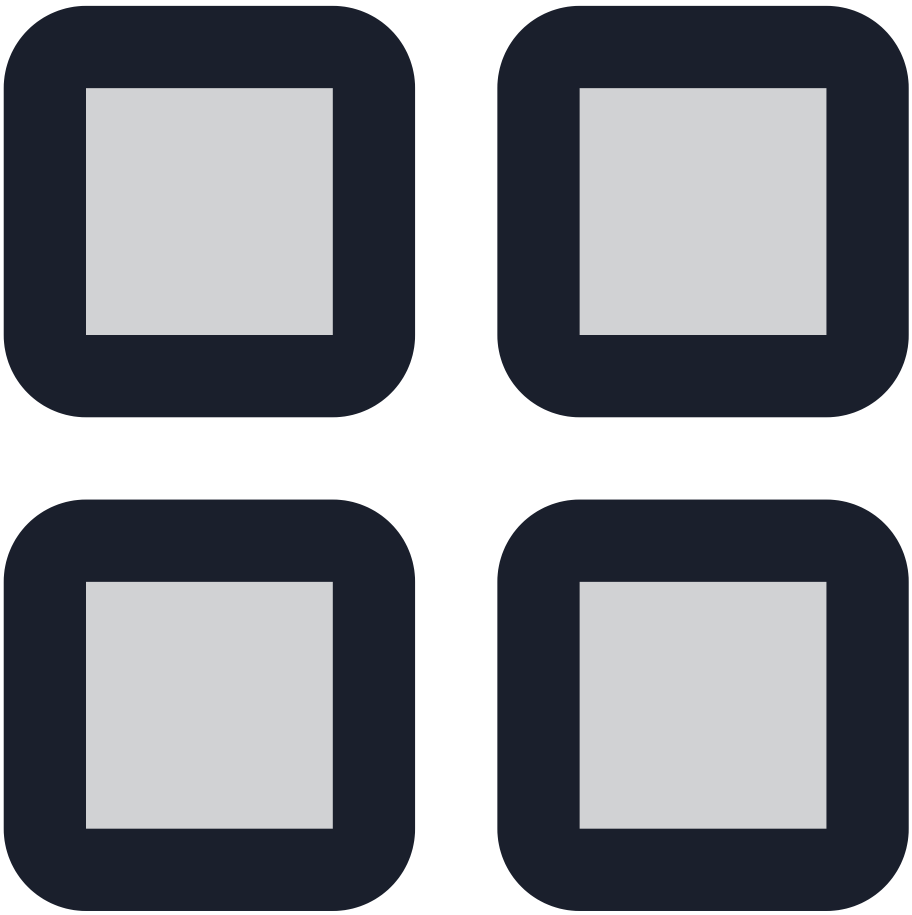
Why this matters to builders

Builders, analysts, and integrators get a venue where the score is a public artifact. The leaderboard endpoint is queryable, funded-account events are emitted on-chain, and the referral cash rail is keyed to a wallet. A third party can build alternative leaderboard dashboards, derivative scoring (e.g. Sharpe-weighted ranks), copy-trading rails, or trader-discovery tools on top of Dexter without asking the platform for permission.

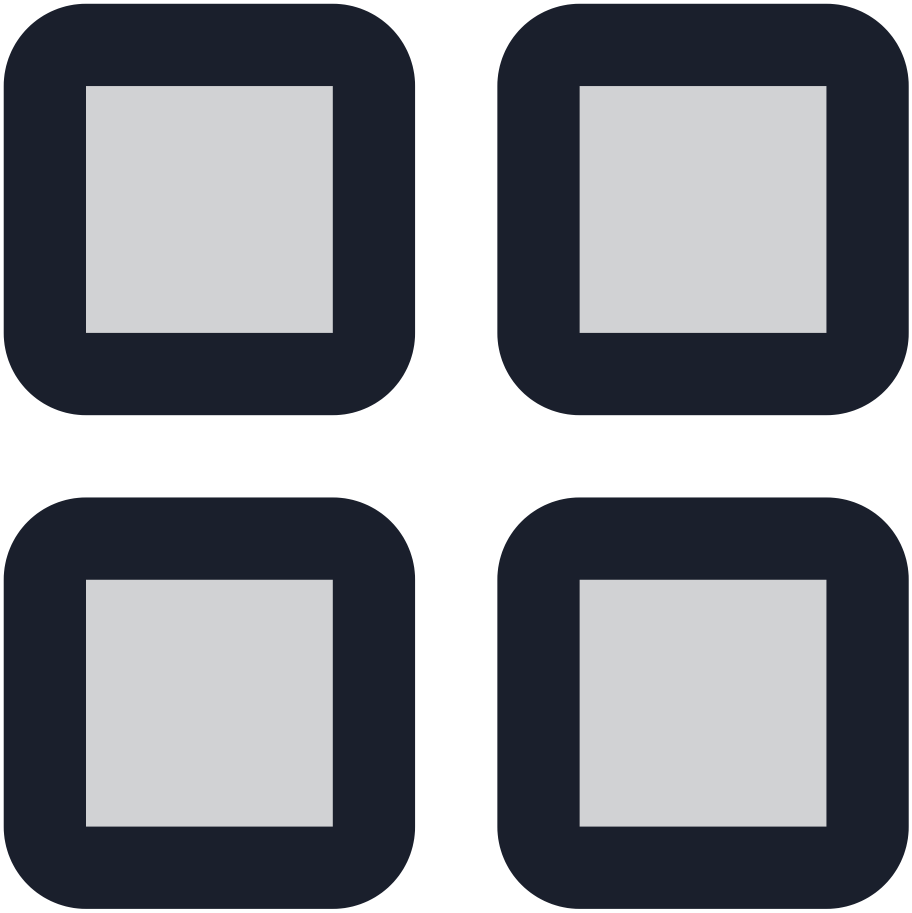
The same property that protects users — every payout is verifiable — lowers the cost of building secondary products against the protocol. The same on-chain pack purchase that gates the challenge is a primitive an external creator monetization tool can subscribe to. The same Basescan transaction that pays out a season winner is a primitive a tax tool can ingest. Dexter is engineered to be composable downstream, not a closed funnel.

CHAPTER 04

ARCHITECTURE



System flow **One venue. Separate authority.**
The interface feels singular, but execution, publication, and custody remain clearly separated.



Product surface **Trade, positions, assets**
One visible venue.

Gateway **Reads, auth, ingress**

Policy before execution.

Core runtime **Single-writer engine**

Pricing, matching, risk, funding, publication.

Publication **Roots and config evidence**

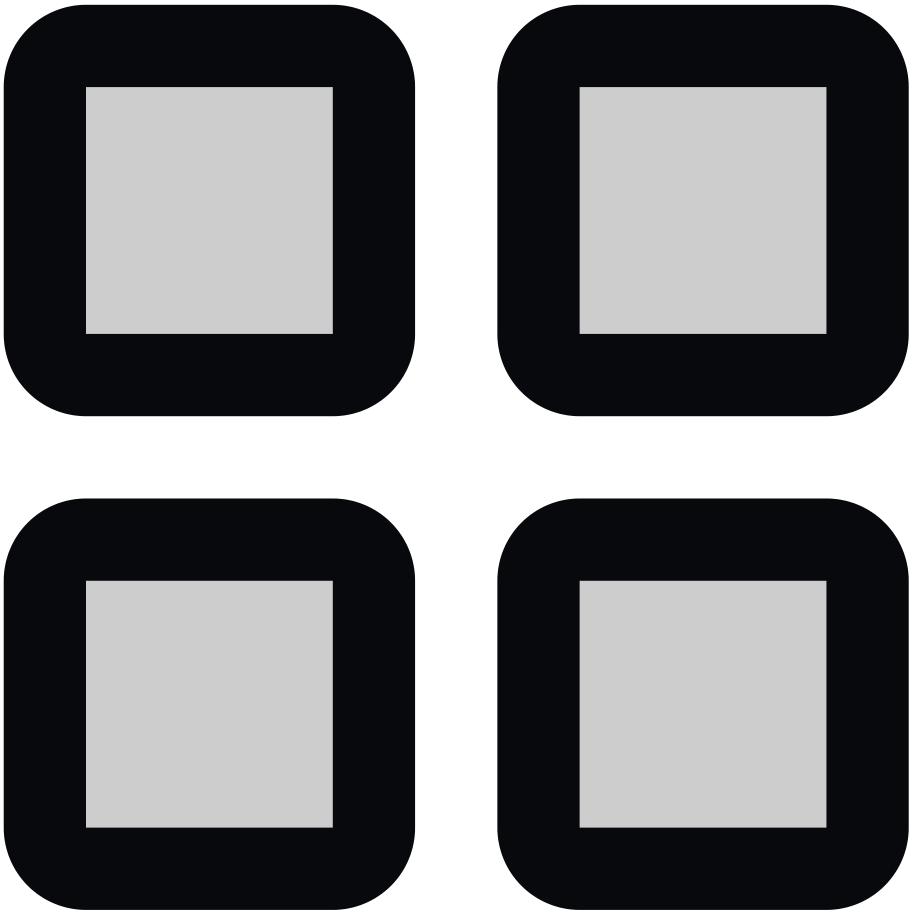
State ships with context.

Contracts **Custody and treasury rules**
Collateral stays bounded.

Read the table as a permission map. Each layer's first column is what it is allowed to do; the third column is what it cannot do, by construction. Nothing in the table moves user funds without a corresponding contract call on Base.

LAYER	PRIMARY JOB	WHAT IT CANNOT DO
Product surface	Render markets, charts, order entry, positions, assets, and account views	No matching, no custody, no settlement
Gateway and read model	Normalize persisted runtime output into one stable API; gate signed ingress with agent auth, IOC, and order-commit policy	No fill decisions, no balance authority
Runtime engine	Single-writer flow: validate signed orders, apply risk, price, fill, advance funding, prepare publication	No direct custody, no unilateral withdrawals, no parameter changes
Publication	Commit state roots with sequence, oracle, batch, and config bindings the vault settles against	No matching loop, no balance writes
Contracts on Base	Hold collateral, queue withdrawals, preserve root history, enforce multi-sig on treasury actions	No live order matching, no off-chain authority

#



One venue, separate operating layers

A trader sees one venue. The implementation splits that venue across five layers because forcing fast execution, readable product state, and custody-critical permissions into one component is how derivatives venues lose user funds. The product surface owns clarity. The gateway owns normalization and the admission policy on signed flow. The runtime owns live exchange state under a single-writer model. Publication owns the bindings between off-chain state and on-chain settlement. The contracts on Base own custody and the multi-sig gates on treasury movement.

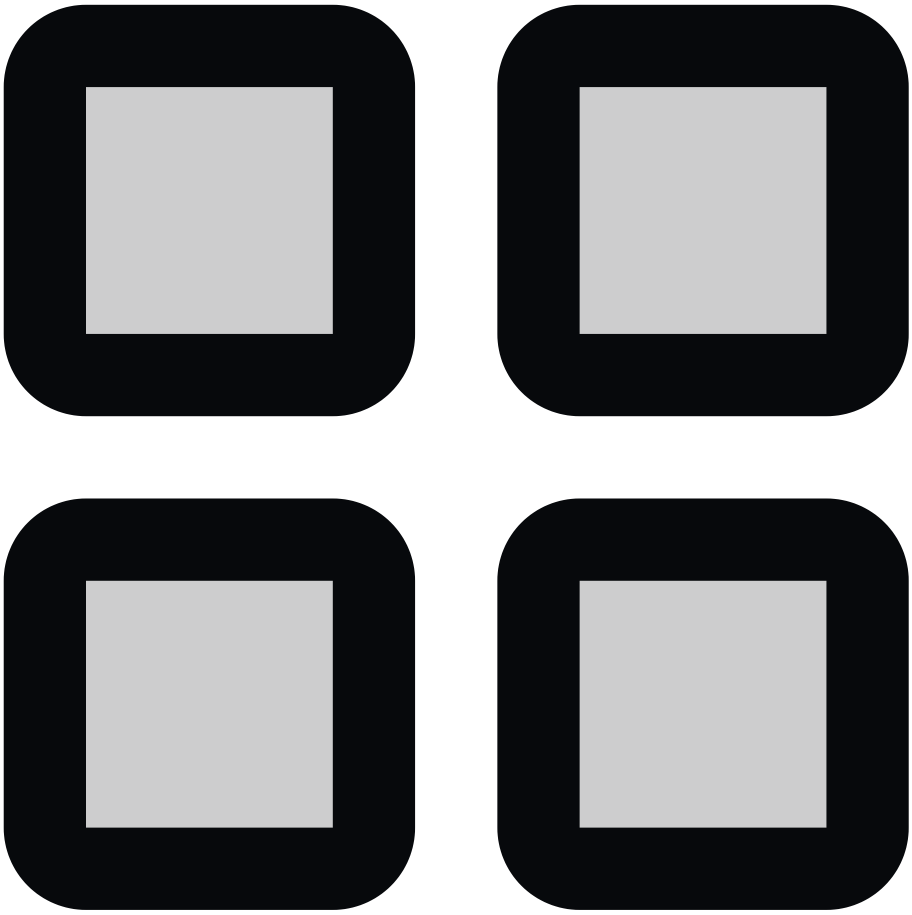
Third-party surfaces — dashboards, copy-trading rails, mobile clients, institutional API consumers — read from the same gateway as the in-house app. They do not constitute a new protocol layer; they consume the same persisted state. That is the structural reason a copy-trader bot, a retail mobile front end, and a quant desk see consistent fills.

Dexter ships in liquidityless vAMM mode. The live venue does not depend on resting maker inventory; execution pressure, skew, spread, and inventory posture are handled inside the runtime against the vAMM curve. Custody and withdrawal rights remain bound to the contracts regardless of what posture the runtime takes. A bad inventory call cannot cost a trader their margin.

TEXT

```
product surface
  → gateway and read model
  → off-chain runtime
  → published commitments
  → vault, treasury, and settlement contracts on Base
```

#



Where the trust boundary actually sits

Dexter does not claim every component is on-chain. It claims something narrower and more useful: execution can be fast off-chain so long as custody, settlement references, and protocol-owned value live on-chain with explicit rules. User collateral, withdrawal requests, nonce history, fee balances, the insurance reserve, and income-side accounting all sit in contract-governed storage on Base because those are the places where ownership has to stay unambiguous.

The runtime advances state and publishes the settlement context the vault reads against. It does not redefine custody after the fact, and it does not authorize a withdrawal the contracts would not honor on their own. Treasury moves above $10,000 require a 2-of-3 multi-sig; insurance and governance moves require a 3-of-5 with an external signer. The runtime cannot bypass either gate. That is what turns the off-chain / on-chain split into a real architectural boundary instead of marketing language layered over a black-box backend.

CHAPTER 05

GATEWAY AND READ MODEL

The gateway serves stable responses from persisted runtime output, never from runtime in-memory process state. That is what lets it cache, version, fall back, and stay deterministic under load — every read goes through the same code path whether the runtime is busy, idle, or restarting. In production it is also the narrow ingress boundary for signed trading flow: agent authorization, timing checks, IOC enforcement, and order-commit submission all happen here before a single byte reaches the matching loop.

SURFACE EXPOSED BY THE GATEWAY	BACKING SOURCE
Market summaries and order books	Runtime snapshots, order-book records, mark and index state
Account, fills, funding history	Persisted account summaries, fill logs, funding settlement records
Publication and liveness	Root history, freshness windows, sequence and config commitment metadata
Venue posture signals	Health snapshots, market-state outputs, oracle source health
Signed trading ingress	POST /agent/authorize signer policy; createdAtTs / goodTilTs windows; IOC discipline; required order-commit submission

#

What the gateway actually owns

The gateway owns presentation, normalization, and the admission policy on signed ingress. It does not own matching, mark construction, custody, or settlement finality — those live in the runtime and the Base contracts. Its job is to expose one stable interface over already-persisted state and to keep the trading front door narrow enough that the rules for entry are explicit and inspectable.

The split between public market reads, authenticated account views, and restricted service routes is enforced as three different permission models, not one. A wallet that can read its own positions cannot read another wallet's positions; an unauthenticated consumer can read the order book but

cannot read fill history; an internal service route cannot be invoked from the public ingress at all. The gateway is therefore part of the safety boundary, not just the UI layer.

Third-party surfaces — copy-trading rails, mobile clients, institutional API consumers — read the same records under the same permission model as the in-house app. They are downstream consumers of one boundary, not parallel protocol layers. There is no "partner" tier with quiet access to private endpoints.

#

The role of the persisted read model

The runtime continuously writes snapshots, fills, funding records, market summaries, account summaries, and health signals into a persisted store. The gateway reads from that store, never from runtime process memory. Two consequences follow: every consumer sees the same state at the same sequence point, and a runtime restart never produces a stale or split-brain read.

TEXT

```
signed user flow
  → POST /agent/authorize sets signer policy
  → gateway checks createdAtTs / goodTilTs / IOC / sequence policy
  → gateway submits required order commitment
  → runtime updates exchange state
  → snapshots, fills, funding, and health are persisted
  → gateway normalizes those records
  → every app and API consumer reads one stable surface
```

Hybrid venues become unreviewable when each screen is backed by a different invisible source. The read model is what prevents that on Dexter. The gateway explains the venue; it does not decide the venue. Anyone reproducing a trader's PnL, a leaderboard rank, or an insurance balance can do so from the same persisted records the gateway serves.

CHAPTER 06

RUNTIME AND PUBLICATION

The runtime validates signed flow from the gateway, applies risk and market-state rules, updates balances and positions, advances funding (peer-to-peer with a 1% admin slice above $1,000 per 8h window), and runs forced routines — liquidation, ADL, the 50/50 keeper-and-insurance residual split. Every state transition produces the references the contract layer later settles against. The table below is what ships with every commit; a root without these bindings is rejected by the vault.

PUBLISHED ITEM	WHAT IT BINDS
State root	Anchors account state for settlement and withdrawal proofs on Base
Root id and timestamp	Places state inside an explicit publication history; both monotonic
Sequence commitment	Binds event ordering to the committed state — no reordered fills
Oracle-source hash	Binds the price-source posture used while state advanced
Order-batch commitment	Binds the admitted batch around that state — no inserted or removed orders
Config hash and version	Bind the active risk and venue rules — fee schedule, margin tier, market list

#

Single-writer exchange state

Dexter runs one write path. Funding updates, liquidation passes, imbalance routines, market-state changes, and publication all move through one ordered runtime flow — never through competing writers. In a derivatives venue, wrong sequencing is as expensive as wrong pricing: a funding tick applied out of order can falsify equity, a liquidation racing a fill can settle against the wrong mark. The single-writer model eliminates both classes of bug by making one runtime the sole author of the state transition sequence the rest of the system reads.

The same runtime that updates balances prepares the material that publication later anchors. That is why settlement context and trading state cannot disagree — they come from the same process, in the same order, behind the same commitment.

TEXT

```
  signed IOC order
    → admission and risk checks
    → pricing and fill decision
    → ledger, position, fee, and funding updates
    → snapshot and read-model write
    → precommitSequencerCommitment(seqCommitment, oracleSourcesHash,
orderBatchCommitment)
    → commitStateRootWithConfig(root, schemaVersion, configHash, configVersion)
    → vault accepts the root only while publication guards remain satisfied
```

#

Publication as settlement context

Publication exists so the contract layer never has to trust a vague operator claim about runtime state. When the runtime ships a root together with sequence, oracle, batch, and config references, the vault on Base receives a concrete description of what the venue believed at the moment state advanced. That description is what withdrawals, disputes, fraud proofs, reconciliation, and post-hoc review all hang on. Without it, the contract would only know that an off-chain process claimed a balance; with it, the contract can anchor a specific runtime view, preserve it in history, and require every settlement action to point back to something concrete.

Publication is not treated as a runtime-only action. Before the vault accepts a root, the sequencer commitment, oracle-source hash, and order-batch commitment must already be precommitted via `precommitSequencerCommitment(seqCommitment, oracleSourcesHash, orderBatchCommitment)`. The root is then submitted through `commitStateRootWithConfig(...)`, which binds it to the active `configHash` and `configVersion`. If the precommitted material does not line up with the root, or if the contract-side governance lock is not satisfied, the publication is rejected and the root never becomes settlement context.

Production posture is stricter than a root commit alone. The governance lock requires the audit anchor to be active, `setOrderCommitRequired(true)` to be in force, and the publication path to maintain at least one durable evidence rail — either an archive requirement or a DA attestor requirement. A root that exists in isolation is not production-grade; it has to remain tied to the audit chain that explains how the state was published and to the same admission discipline that produced the order flow that generated it. That is what makes the off-chain runtime safe to anchor against on-chain custody.

CHAPTER 07

MARKET ENGINE

The engine decides whether an order may enter, the price at which it fills, how position size and cash settle, and when a market must shift from live trading into a defensive posture. Each of those decisions is owned by a single execution domain so a fill, a funding accrual, and a liquidation cannot disagree about state. The table below maps the five responsibilities — each is covered in depth on its own page in this chapter.

ENGINE RESPONSIBILITY	WHAT IT COVERS
Front-door control	Agent authorization, EIP-712 signature validity, monotonic seq, createdAtTs / goodTilTs guards, and the on-chain OrderCommitRegistry anchor
Pricing	Liquidityless vAMM skew, spread controls, inventory caps, and per-market open-interest pressure
Ledger updates	Position size, entry basis, realized PnL, 0.020% maker / 0.060% taker fees, fundingAccrued, and account cash
Market posture	Live, reduced, close-only, session-closed, and halted — the same five states across every venue
Forced routines	Maintenance-margin liquidation, insurance flow, ADL selection, and rebalance after a stress event

#

How the live trading path works

Production never accepts a raw order from a public RPC. Each request arrives as an EIP-712 signed IOC payload carrying a monotonic seq, createdAtTs, and goodTilTs. Before the runtime prices anything, it confirms the signer is registered through agent authorization, the seq has not been reused, the timestamps are inside the validity window, and — when commit guard is active — the order hash has already been recorded to the on-chain OrderCommitRegistry on Base via commitOrder. Any one of those checks failing aborts the request before it reaches the matching path.

Pricing runs against a liquidityless vAMM. There is no resting maker book to lean on, so the engine quotes flow through skew-sensitive curves, hard inventory caps, and per-market open-interest pressure. A taker buying into an already long-heavy book pays a higher effective rate than a taker buying into a balanced book, and the cap on outstanding directional notional is enforced at the admission boundary rather than at fill time.

When a fill is accepted, the engine moves position size, entry basis, realized PnL, the 0.020%/0.060% fee component, and any cash settlement in a single ordered transition. Funding accrues from the same state advance, so a trader cannot observe a position whose carry has updated but whose realized PnL has not. The unfilled remainder of an IOC is cancelled immediately — there is no working order to track.

TEXT

```
agent authorized
  → commitOrder(...) recorded on OrderCommitRegistry
  → EIP-712 signed IOC enters runtime
  → signer / seq / createdAtTs / goodTilTs checks
  → admission and posture checks
  → vAMM price quote against skew and inventory
  → fill or cancel remainder
  → position, fee, fundingAccrued, cash advance in one transition
  → publication pipeline emits new market state
```

#

Market posture, carry, and forced routines

Every market runs inside an explicit state machine with five postures: live, reduced, close-only, session-closed, halted. These are not UI tags — they are admission rules that govern whether a fresh order can add risk, whether reduction is still open, and whether the venue is honoring an underlying session break or waiting on a cleaner price reference. Transitions are driven by reference freshness, session calendars, skew pressure, utilization, bad-debt exposure, and recovery checks after a degraded period.

Open interest, directional imbalance, and crowding are measured continuously, and they feed the same loop that sets the funding index, the skew-adjusted taker rate, and the inventory caps. Funding in Dexter is part of the engine's carry model — accruing every 8 hours into fundingAccrued — not a number reconstructed from after-the-fact reporting. The pages that follow in this chapter break down each surface in detail.

Forced reduction lives in the same execution domain. Liquidation scans, insurance-aware bad-debt handling, ADL selection, and post-stress rebalance all share state and ordering with ordinary order processing. That is what keeps stress behavior deterministic: when maintenance margin breaches the floor and a position is liquidated, the same fee, funding, and ledger transition runs that would run for a voluntary close — only the trigger is different.

#

What this means for your challenge

- **The same engine fills your challenge as fills the public book.** A $49 or $99 attempt is not routed through a sandbox — it is priced against the production vAMM at the same skew, the same inventory cap, and the same 0.020% maker / 0.060% taker schedule. A +10% target produced here is the +10% a public $50K account would have produced on identical flow, which is why funded payouts on Base are reproducible against the chain.
- **Skew costs are real costs that count against +10%.** When the engine raises the effective taker rate against a crowded direction, that surcharge lands on your equity inside the same fill.

Fighting the skew is the most common reason an attempt grinds out the -4% daily floor without ever touching the liquidation line.

- **Funding carry is part of your PnL.** Every 8 hours the funding index settles and fundingAccrued moves your equity. Holding a crowded-direction position for 30 days can quietly burn 1–3% before any directional move — check the published rate per market before sizing a multi-day hold.
- **Forced routines end attempts the same bar they fire.** Liquidation, ADL, and rebalance flows close positions at the mark, attach an explicit liquidation fee on top of the realized loss, and almost always cross the -4% daily or -8% total drawdown on the same bar. The engine treats them as ordinary transitions, so there is no carve-out: the loss hits the rule check immediately.
- **Funded accounts run the same code path.** A 90 / 10 payout account does not get a separate matching layer or a private fee schedule. Same pricing, same fees, same funding cadence, same maintenance margin — which is why a passed account behaves exactly like the attempt that earned it.

CHAPTER 08

ORDER FLOW AND MARKET STATES

Whether the engine accepts your order is the product of two checks: the signed payload itself must be valid, and the current market posture must permit the action you are requesting. The first is covered on the admission page; the table below summarizes the second. Each posture is an explicit admission rule — the engine rejects forbidden actions at the gate rather than letting them sit and silently fail.

MARKET POSTURE	WHAT THE ENGINE ALLOWS
Live	Open, close, scale in, scale out — full directional and reduction flow at the standard 0.020% / 0.060% schedule
Reduced	New risk is admitted under tighter inventory caps and higher skew-adjusted taker pressure; reduction stays at standard cost
Close-only	Only risk-reducing flow is admitted; new opens, scale-ins, and same-side adds are rejected at the gate
Session-closed	Underlying venue is on a scheduled break (equity, metals, energy hours); positions are held but not marked against a stale tick
Halted	All new execution stops, including reduction, until the oracle path recovers or operator review clears the venue

#

Admission before matching

Before pricing logic runs, the engine classifies each request along three axes: does the target market exist and pass its freshness check, is the signed payload inside its validity window, and would the fill increase or reduce the account's directional risk. The add-vs-reduce distinction is what lets a market accept close orders while rejecting opens — close-only and reduced postures both depend on that classification rather than on a UI hint.

Production runs IOC-only. The gateway enforces GATEWAY_REQUIRE_IOC, so every signed order is evaluated against live state at arrival and any unfilled remainder is cancelled in the same transition. There is no working order book sitting between the trader and the engine, which means a posture transition takes effect on the next arriving order rather than retroactively touching resting size. That keeps stress behavior narrow and reproducible: every fill is a single signed payload meeting a single state snapshot.

#

State machine instead of hidden restrictions

Markets do not move from live to failure in one step. The intermediate postures — reduced, close-only, session-closed — exist so the venue can tighten without going opaque. A reduced market is still functional but more expensive on new size. A close-only market lets you exit but blocks fresh exposure. A session-closed market is honoring an underlying calendar (NYSE hours, LBMA fix windows, energy benchmarks) instead of pretending a stale tick is still a valid reference.

Halted is materially more serious: an oracle path has broken, a stress event is being reviewed, or operator intervention is required to restart the venue. The posture is always visible in the trade UI before you submit, and the engine rejects forbidden actions at admission rather than letting them sit in a queue. A trader should never have to infer market state from price action alone — the posture is published, the rule is enforced at the gate, and the rejection reason is returned synchronously.

TEXT

```
signed IOC arrives
  → signature, seq, createdAtTs, goodTilTs validity
  → market freshness and posture check
  → add-risk or reduce-risk classification
  → posture rule: admit or reject
  → vAMM quote against skew / inventory cap
  → fill or cancel remainder in same transition
```

#

What each posture means for an attempt

- **Live.** The default posture during major crypto sessions. Open, close, scale in, scale out at the standard 0.020% maker / 0.060% taker schedule. Skew-adjusted taker pressure is at its baseline floor.
- **Reduced.** The most common "stress but functional" state. Fresh size is admitted, but the inventory cap is tighter, the skew-adjusted taker rate is higher, and the per-account directional notional limit moves down. Existing positions continue to mark, accrue funding, and bind to the maintenance margin floor — only new opens are throttled.

- **Close-only.** Reduction is open, opens are blocked. A long can sell to flat or smaller, but cannot scale up. A flat account cannot enter. Stop-loss and take-profit reductions still route. New same-side adds are rejected at admission with an explicit reason, not queued.
- **Session-closed.** The venue is honoring the underlying market's calendar — NYSE bell hours for equity perps, LBMA windows for metal perps, exchange hours for energy perps. Positions are held, but marking against a stale tick is suspended. Drawdown rule checks pause until the next session opens.
- **Halted.** No new execution, including reduction. Typically triggered by an oracle path break, a stress event, or an operator review. While halted, equity does not advance, fundingAccrued does not progress, and the 30-day evaluation window pauses. The clock resumes only when the engine restores a normal posture and the next signed order is admitted.

Posture changes are admission rules, not UI overlays. The current state is published before you sign — if you submit an open into a close-only market, the engine returns the rejection synchronously rather than sitting on the order. Inside a 30-day challenge window, a session-closed or halted period extends your deadline by the paused duration; a reduced or close-only period does not, because the engine is still admitting valid risk-reducing flow against your equity.

CHAPTER 09

ADMISSION, SIGNATURES, AND ORDER COMMITMENTS

Six requirements gate every order. They run in fixed order and any failure aborts the request before the pricing layer sees it. The first three are signer-and-payload checks; the last three are timing, commitment, and execution-shape policy.

REQUIREMENT	WHY IT EXISTS
Agent authorization	Binds an explicit signer key to the account policy via **POST /agent/authorize**; an order from any other key is rejected before signature checks run
EIP-712 order signature	Proves the authorized signer agreed to the exact payload — market, side, size, price, validity window — that the runtime is about to act on
Monotonic seq	Each account has a strictly increasing sequence counter; replays of a prior seq are rejected and ordering is deterministic across the runtime
createdAtTs / goodTilTs guards	Rejects stale, future-skewed, or expired payloads so a signature from 10 minutes ago cannot drift into the matching path
OrderCommitRegistry anchor	When commit guard is active, the gateway records the order hash on-chain through **commitOrder** / **commitOrders** before placement; the runtime refuses to match an order whose commit is missing
IOC-only execution	Production enforces **GATEWAY_REQUIRE_IOC**; non-IOC payloads are rejected at the gateway so the active execution path stays narrow and there is no resting working-order surface

#

Authorized signed flow

A client cannot sign an order and hope the venue accepts it. The account first registers a self-policy through POST /agent/authorize, which records the specific signer key allowed to act on its behalf. That separation matters in practice: the account's withdrawal key, KYC binding, and referral cash never need to touch a trading session — a hardware wallet, a session key, or an API client takes over the signing surface without the rest of the wallet ever being exposed.

Once authorization exists, every order arrives as an EIP-712 typed payload rather than an unsigned instruction. The signed structure carries the market, side, size, limit price, validity window (createdAtTs, goodTilTs), and the account's monotonic seq. The engine rejects unauthorized signers, missing seq, expired or future-skewed timestamps, and non-IOC payloads at the gate — none of those failures reach the pricing or risk layer. In production this is enforced by GATEWAY_REQUIRE_IOC at the gateway and by the runtime's payload validator, so the same rule is checked twice on independent code paths.

#

Sequencing and on-chain commit evidence

Monotonic sequencing gives the runtime a deterministic event order. Each account's seq advances strictly upward, so two independent observers replaying the same chain see the same order of fills. The runtime rejects any payload whose seq is not greater than the previously accepted seq for that account — a replay, a network re-order, or a duplicate retry all collapse into a single accepted event.

When commit guard is active in production, the gateway anchors the order hash on the OrderCommitRegistry contract on Base through commitOrder (single) or commitOrders (batch) before the runtime is allowed to match. That achieves two distinct things in one step. It narrows admission: the runtime will not price an order whose commit is missing. And it produces a public, timestamped, on-chain record that the order existed in the form the signer attested to, before any fill could have been generated — which is what later dispute review checks against.

TEXT

```
  POST /agent/authorize
    → signer policy becomes active for the account
    → client signs EIP-712 IOC payload (market, side, size, price, createdAtTs,
goodTilTs, seq)
    → gateway enforces GATEWAY_REQUIRE_IOC and validates payload shape
    → commitOrder(orderHash) lands on OrderCommitRegistry on Base
    → runtime re-checks signer, seq monotonicity, timestamps, and commit presence
    → only then does vAMM pricing and fill logic run
```

#

Reviewing the admission boundary

The admission stack is not ceremony. It exists so a technical reviewer can answer three questions from public state alone: which signer was allowed to act for this account at the time of the order, in what order the requests arrived relative to one another, and whether the runtime matched a payload that had already been anchored on-chain. If any of those answers is "we cannot tell," the venue is operating outside its own admission contract. With agent authorization, monotonic seq, and the OrderCommitRegistry anchor, all three answers come from data a reviewer can fetch without trusting the venue's word.

#

Why this matters for the leaderboard and payouts

- **Trade history is anchored on Base.** Every order's hash hits the OrderCommitRegistry through commitOrder before the runtime matches it. A podium finish or a 90 / 10 funded payout can be reproduced from the chain — the venue cannot rewrite the history that a payout was calculated against, because the timestamp of each commit is older than the fill it justified.
- **Monotonic seq prevents double-counting.** A network retry, a relayer race, or a client bug that re-sends a payload all collapse into one accepted event because the seq has already advanced. The leaderboard cannot double-count a fill, and a profit-share withdrawal cannot be disputed by a duplicate fill claim arriving later.

- **Agent authorization isolates the trading surface.** The wallet that holds challenge entries, funded payouts, and DXTR stake does not need to sign trades. POST /agent/authorize binds a separate signer — hardware wallet, session key, API client — to the account policy. If the trading key leaks, withdrawals, KYC binding, and referral cash remain on the original wallet and outside the attacker's reach.
- **createdAtTs / goodTilTs block replay windows.** A signature is only valid inside its declared validity window. A signed payload captured 10 minutes ago cannot be replayed today, even if the signer key is later compromised — the runtime rejects it on the timestamp guard before any further check.
- **Same gate for everyone.** A copy-trading client, an institutional API integration, and a retail wallet all authorize, sign, commit, and submit through the same boundary. There is no privileged lane that lets the venue admit one class of order under a different rule.

CHAPTER 10

FUNDING, FEES, AND RISK CARRY

Every fee or carry component in Dexter is computed inside the engine — the accounting layer reconciles totals later, but the numbers below all originate from the same ordered state advance that settled the underlying fill.

COMPONENT	HOW THE ENGINE TREATS IT
Maker / taker fees	0.020% maker, 0.060% taker on notional, recorded inside the fill transition; -0.005% rebate on resting maker contribution
Skew-adjusted taker	The published taker rate is the floor; the engine raises it on directional flow into a crowded book and relaxes it on flow that flattens skew
Funding accrual	Settles every 8 hours per market into fundingAccrued; longs pay shorts when longs are crowded, and vice versa
Funding admin cut	1% of the funding paid by an account in any 8-hour window above $1K is routed to protocol-owned reserves; smaller windows pass through gross
DXTR stake discount	-10% / -20% / -30% off the taker rate at 1K / 5K / 25K DXTR staked, computed nightly and applied at the cashier
Liquidation fee	Attached on top of the realized loss when a forced close fires; 50% to the keeper, 50% to insurance after any bad-debt offset

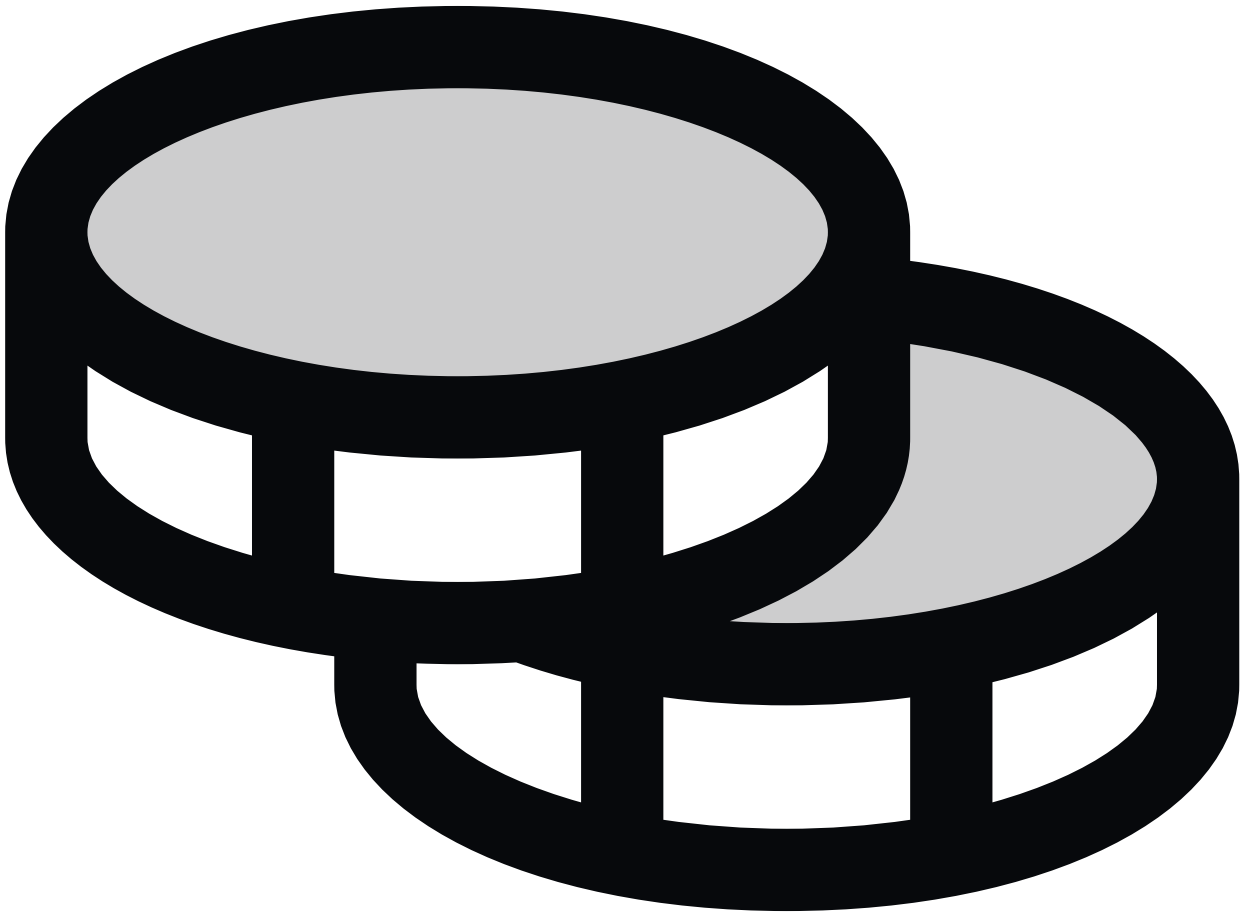
#

How funding enters the ledger

Funding in Dexter settles every 8 hours per market. The engine measures directional imbalance — open-interest skew between longs and shorts — continuously, computes the funding rate against a clamped band, and at each 8h boundary advances fundingAccrued for every account holding inventory in that market. The side that pays is the crowded side: longs pay shorts when longs dominate open interest; shorts pay longs when shorts dominate. The rate clamp keeps a single window from producing pathological numbers during a transient skew spike.

Above $1K of net funding paid by an account in a single 8h window, a 1% admin fee is routed to protocol-owned reserves; smaller windows pass through gross. That cut exists so the bulk of carry remains a peer-to-peer transfer between traders on opposite sides of the book, while concentrated outsized payers contribute a small share to the insurance and treasury layer that absorbs their stress when it materializes. Because funding is a runtime state advance and not a reporting artifact, an account's equity reflects the carry by the time the 8h boundary closes — there is no lag between the rate publication and the deduction.

#



How fees become protocol-owned value

Maker and taker fees are created the moment a fill is accepted. The 0.020% maker / 0.060% taker rate is applied to the notional inside the same state transition that moves position size, entry basis, realized PnL, and account cash — there is no separate "fee post" step that could disagree with the fill. Resting maker contribution earns the -0.005% rebate against the taker that crossed it; if you held DXTR stake at one of the published tiers, the -10% / -20% / -30% taker discount lands at the cashier reconciliation, not on each individual fill.

A later reconciliation path converts the engine's fee, funding-admin, and liquidation-fee totals into on-chain movements: protocol-owned fee balances, insurance top-up, and treasury allocations. The split between the matching engine and the finance layer is deliberate. The engine owns when and how carry and fees arise; the treasury path owns how those protocol-owned balances later settle into the vault. Both layers see the same numbers because both layers read from the same ordered state.

TEXT

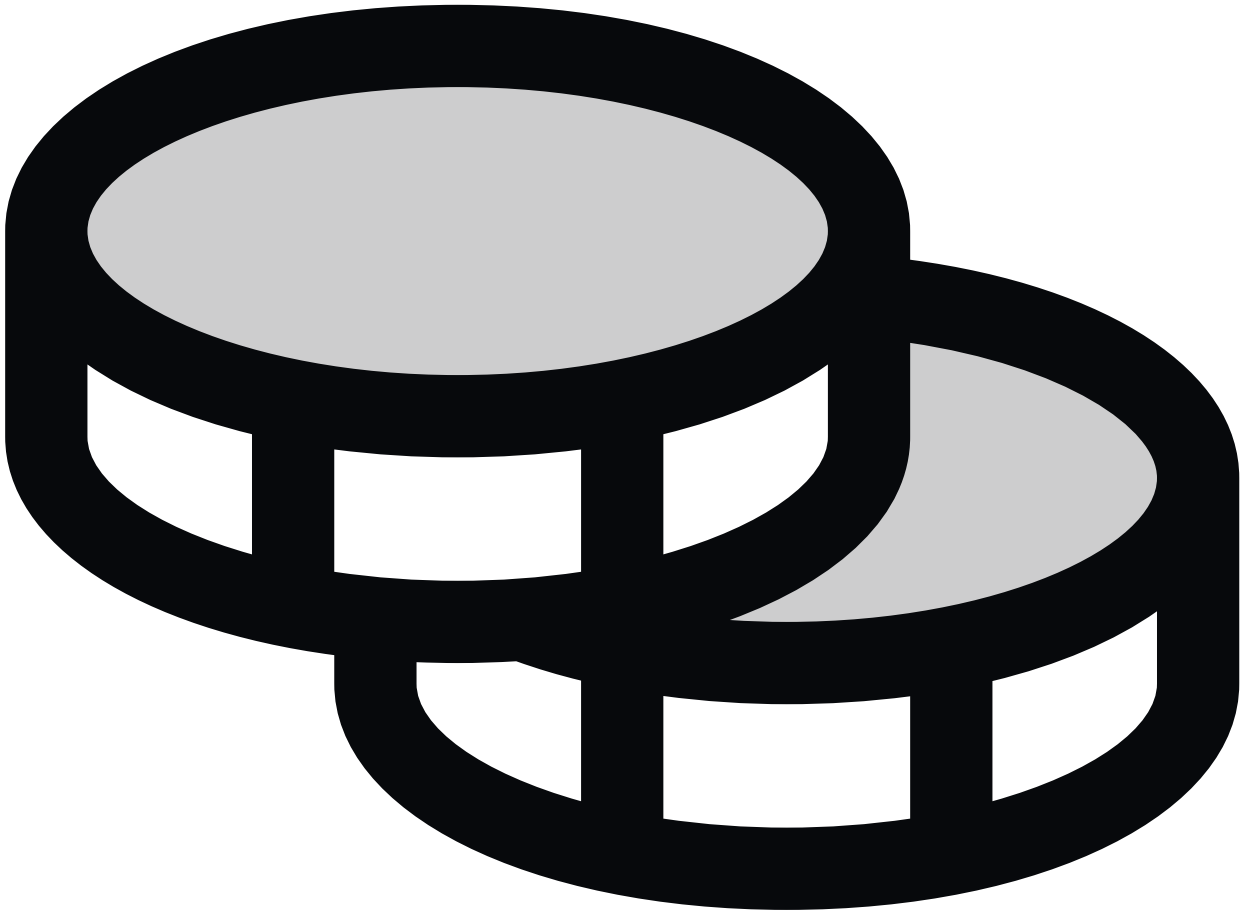
```
  trade executes
    → taker pays 0.060% notional, maker rebated -0.005% on resting contribution
    → skew-adjusted taker surcharge applied based on book imbalance
    → fundingAccrued advances at next 8h boundary; 1% admin cut above $1K/window
    → DXTR stake tier discount applied nightly at cashier
    → reconciler converts protocol deltas into vault fee, insurance, treasury
movements
```

#

Why carry belongs in the engine

If funding, the skew-adjusted taker surcharge, and the liquidation fee were treated as reporting artifacts settled out of band, a venue could publish a clean dashboard while its actual carry drifted underneath. Dexter writes all three into the same ordered state advance as the underlying fill, so the market's visible posture, an account's equity, and protocol-owned reserve balances always derive from the same transition. The funded trader sees the same fee an auditor reconstructs from the chain — and the rate that hits the leaderboard is the rate that hit the wallet.

#



What this costs a funded trader

- **Fee schedule.** 0.020% maker / 0.060% taker on notional. A $50K round-trip at taker on both sides costs $60. The fee lands inside the same fill transition, so a position that goes through and reverses inside one block pays both legs without any out-of-band reconciliation.
- **Funding cadence.** Every 8 hours per market. The rate publishes ahead of the boundary; the deduction lands on fundingAccrued at the boundary itself. Above $1K of funding paid in a single window, a 1% admin cut routes to insurance — smaller windows pass through gross.

- **Skew-adjusted taker.** The published 0.060% rate is the floor. A long taker into a 90/10 long-heavy book pays an explicit surcharge on top; a long taker that flattens that skew pays at or near the floor. Fighting the skew compounds: higher taker, higher funding paid by the crowded side, lower fill quality at the inventory cap.
- **Maker rebate.** Resting maker contribution earns -0.005% on the notional it crossed. On a thin attempt, designing entries that rest rather than take saves 0.065% per round-trip — a meaningful edge for higher-frequency strategies running into the same +10% target.
- **DXTR stake discount.** Staked DXTR moves the taker rate down by 10% / 20% / 30% at the 1K / 5K / 25K tiers. The discount is computed nightly from the on-chain stake snapshot and applied at the cashier reconciliation — it is not retroactive on intraday fills, but compounds across the 30-day window.
- **Liquidation fee.** A forced close attaches an explicit liquidation fee on top of the realized loss. The fee splits 50% to the keeper that triggered the close and 50% to insurance after any bad-debt offset. Size positions so liquidation fee + realized loss never crosses the -4% daily or -8% total drawdown floor — once liquidated, the rule break is automatic.

Concrete cost model: a 5x notional trader doing 4 round-trips per day on a $10K allocation pays roughly $24/day in taker fees ($10K × 5 × 4 trips × 0.0006 × 2 sides). Across 30 days that is about $720 of pure fee drag — 7.2% of allocation — before funding, before skew surcharges, and before any liquidation fee. A 1K DXTR stake brings that to $648, a 5K stake to $576, a 25K stake to $504. The +10% target assumes those costs are inside the strategy, not stacked on top of it; the discount tiers are how a high-frequency challenge run claws back its edge.

CHAPTER 11

LIQUIDATIONS AND PROTECTION ROUTINES

Forced reduction is a ladder, not a single switch. The engine prefers the lightest intervention that restores solvency: a staged partial close at the margin floor before a full liquidation, the insurance pool before any socialized loss, ADL only as a last resort when normal liquidation cannot find liquidity at the mark.

PROTECTION ROUTINE	WHY IT EXISTS
Liquidation scans	Continuously check every account's maintenance margin against the current mark; surface candidates the moment they cross the floor
Staged reduction	Trim a flagged position partially before a full close, so a temporary skew spike does not produce a complete wipe-out
Liquidation fee + 50/50 split	The fee on top of realized loss splits 50% to the keeper that triggered the close and 50% to insurance after any bad-debt offset
Insurance absorption	Protocol-owned insurance pool covers residual bad debt before any socialization touches winners
ADL	Last-resort de-leverage of winning counterparties when ordinary liquidation cannot clear at the mark; closes at mark, not at stop
Rebalance + posture tighten	Push an imbalanced market back toward a healthy posture, dropping the venue into reduced or close-only while recovery proceeds

#

How stressed markets are handled

The runtime scans every account's maintenance margin against the current mark continuously. When an account crosses the maintenance floor, it becomes a liquidation candidate inside the same transition that priced the move — there is no polling lag between the price advance and the flag. If a partial reduction is sufficient to restore solvency, the engine takes it first: a portion of the position is closed at the mark, the liquidation fee is attached only to the closed slice, and the remaining size stays open under a tighter margin window.

If the partial path cannot restore solvency, the engine escalates to a full liquidation. The closed loss settles against the account's remaining equity. The liquidation fee splits 50% to the keeper that triggered the close and 50% to the insurance pool after any bad-debt offset. If the realized loss exceeds the account's posted margin, the residual is drawn from insurance before any socialization is considered. The insurance pool is funded by the same fee, funding-admin, and prior liquidation flows the engine has already collected, so the buffer that absorbs stress is funded by the activity that produces it.

ADL is the last resort. It runs only when ordinary liquidation cannot clear the bad position at the mark and insurance cannot absorb the residual — typically during a gap move or a stress event when the vAMM cannot absorb the unwind without exceeding inventory caps. ADL closes selected winning counterparties at the mark, not at their stops, to clear the residual exposure. While the protection ladder is running, the market posture itself tightens: the venue can drop into reduced or close-only to prevent fresh size from arriving into a book that is still unwinding.

TEXT

```
maintenance margin breached
  → liquidation candidate flagged inside the same transition
  → staged partial close at mark if it restores solvency
  → full liquidation if partial is insufficient; liquidation fee attached
  → 50% keeper / 50% insurance split after bad-debt offset
  → insurance pool absorbs residual loss before any socialization
  → ADL of winning counterparties only if insurance cannot cover
  → posture may drop to reduced / close-only while recovery proceeds
```

#

Why protection lives inside the engine

Forced reduction cannot be a separate microservice that wakes up after a loss. It has to share the exact state, priority order, and timing guarantees as the matching layer — otherwise a liquidation can fire against a stale margin snapshot, an ADL can pick a counterparty whose position has already moved, or an insurance draw can settle against numbers that have drifted from the engine's view. Dexter keeps liquidation, insurance flow, ADL selection, and post-stress rebalance inside the same engine domain as matching, funding, and posture changes. That is what makes

stress behavior deterministic: the same maintenance margin number, the same mark, the same fundingAccrued, the same inventory cap — across the voluntary close path and the forced close path alike.

#

How to stay out of liquidation during the challenge

- **Drawdown rules fire before liquidation in most cases.** -4% on close-to-close equity and -8% from the high-water mark are stricter floors than maintenance margin. Respect the drawdown rules and you almost never see a liquidation. Ignore them and the liquidation fee on top of the

realized loss makes the rule break decisive — there is no recovery from a liquidation inside the same 30-day attempt.

- **Watch maintenance margin, not initial.** The engine flags candidates the moment maintenance crosses the floor. Sizing to initial margin and getting flagged on a 1.5% adverse move is the most common pattern in blown attempts — plan with a meaningful buffer between your worst-case mark and the maintenance line.
- **Crowded markets liquidate first.** Maintenance pressure rises on directions with crowded open interest. A long into a 90/10 long-biased book has a narrower liquidation window than a long into a balanced book — same notional, tighter floor. The leaderboard does not reward fighting the skew; the protection system is what enforces that fact.
- **ADL is rare but binding.** If your position is selected for ADL during a stress event, the close happens at the mark, not at your take-profit or stop. Plan max-loss assuming a stop might not fire and ADL might close you at the wrong end of a fast move. Winners get ADL'd; losers get liquidated. Both close at mark.
- **Forced close = realized loss = drawdown event.** The liquidation fee plus the realized loss together count against the -4% daily and -8% total drawdown. There is no "the engine did it, not me" carve-out — equity is what the rule checks, and the engine treats forced closes as ordinary state transitions for that purpose.
- **Insurance protects the venue, not your attempt.** The 50% of the liquidation fee that lands in insurance, and the residual loss absorbed by the pool when you blow up, keep the venue solvent for other traders. Your attempt is already over by the time insurance gets involved — by definition, you crossed maintenance.

Two disciplines, not one: stay inside the drawdown rules so you never approach the liquidation line, and stay inside maintenance margin so a sudden gap or session reopening cannot wipe you before the rule even fires. Clearing one without the other is not enough.

CHAPTER 12

PRICE LAYER

Price cascade **Only a defensible price becomes tradable.**

Dexter cross-checks reference families first, then promotes a mark into executable venue state.

Primary **Hermes / Pyth**

Main reference family.

Cross-check **HTTP oracle mux**

Independent confirmation.

Decision **Executable mark**

The engine trades the venue mark.

Fallback **Explicit degraded policy**

Reduced quorum only when allowed.

Publication **Source evidence with state**
Every accepted mark is explainable.

The price layer owns five decisions: which references enter the venue, which reference path is active when multiple sources are healthy, how the executable mark is derived from that reference plus venue state, when a market must downgrade from live to reduced, close-only, session-closed, or halted, and which source posture gets stamped into the next settlement root so the contract layer can later prove which prices a payout depended on. Each decision is enforced before the mark becomes tradable — none of them are post-hoc disclosures.

PRICE-LAYER RESPONSIBILITY	WHAT IT CONTROLS
Source intake	The whitelist of external publishers (Hermes/Pyth primary, HTTP oracle mux secondary) eligible to feed the venue
Source selection	The active reference path at every tick — driven by freshness, agreement, and deviation policy, not preference
Mark construction	Reference anchor blended with venue skew, spread, depth, max-move continuity, and prior-mark drift caps
Degradation logic	The trigger that downgrades a market to reduced, close-only, session-closed, or halted before risk checks misfire
Settlement context	The oracle-source hash committed inside each state root so payouts are bound to the same posture the trades priced against

#

From reference intake to executable mark

Normal production posture for every supported market is the same: Hermes/Pyth as the primary reference family running through a low-latency push channel, and an independent HTTP oracle mux providing a redundant pull path that the runtime cross-checks against every tick. The two paths must agree within deviation bounds before either can be promoted into the executable mark — a single source on its own is never sufficient under normal operation. This is the entire reason the venue refuses to treat "the price exists" as the same thing as "the price is tradable."

The runtime explicitly does not aim to maximize the number of contributing publishers. The aim is the opposite: hold exactly one defensible reference path live per market at any given second, with a pre-vetted ladder of fallbacks behind it for failover. When one family weakens — Hermes drops a publisher, an HTTP endpoint stalls, deviation widens beyond tolerance — the mux either swaps to the redundant path, restricts the posture, or pulls the market out of normal trading. Convenience is not a valid reason to keep trading on a thinned source set.

The engine then promotes the accepted reference into an executable mark instead of trading the raw tick. Reference anchor, venue skew, spread posture, depth checks, max-move continuity caps, and prior-mark drift bounds are layered together so the price the matching engine actually executes against reflects what Dexter can responsibly fill — not what the outside index happens to print at that instant. Funding, liquidation, and the rule-engine's drawdown checks all read the same executable mark. The full intake-to-mark policy lives on Sources and fallback; mark construction details and the parameter list live on Mark construction.

CHECK	PURPOSE
Freshness window	A reference older than the per-market staleness cap is rejected before it can mark equity or trigger a stop
Source agreement and deviation bounds	Primary and cross-check must agree within the configured deviation band; a single rogue publisher cannot drag the mark on its own
Depth and spread checks	If the venue book is thin or the spread blows out, the mark is widened or the market downgrades — thin books never define executable price
Recovery lag	After a degraded or stale window, the market does not snap back to live the moment one fresh tick appears — a stabilization period must pass first

TEXT

```
external references enter
  → mux and source policy checks
  → freshness and deviation validation
  → mark construction from reference + venue state
  → market posture update
  → oracle-source context is published with the next settlement state
```

#

Price posture and session honesty

The freshness, agreement, depth, and recovery checks above feed directly into the market-state machine. Every Dexter market lives in one of five postures at any moment: live (normal two-sided trading), reduced (tighter caps, wider spread, lower max leverage), close-only (open positions can reduce but no new exposure), session-closed (off-hours for the underlying market), and halted (no execution, equity frozen at the last valid mark). These postures are not advisory labels — the matching engine, the rule engine, and the cashier all read the same state.

Crypto perps run 24/7, but equity perps, metal perps, and energy perps follow the underlying venue's schedule. A session-closed equity perp at 03:00 UTC is healthy; an oracle that lost its primary path at 03:00 UTC on BTC is degraded. Dexter deliberately separates the two so users, the rule engine, and the post-incident reviewer all see the same distinction: off-session is a clean state with a published reopen time, degraded is an open question that the venue is actively working through. The detailed session calendar and the live-to-halted transition rules are documented on Sessions and degraded states.

The active source posture — primary path identity, fallback engagement, deviation observed, freshness margin — is hashed into every settlement root the runtime publishes. The vault contract therefore knows not just what the balances were when a withdrawal request was included, but which oracle configuration produced them. A payout cannot be quietly settled against a different feed than the one your trades priced against, because the two are bound by the same root commitment.

#

What this means for your PnL and payouts

- **One mark, funded or public.** A 90 / 10 funded account, a public wallet, and the leaderboard ranking engine all mark against the same Hermes/HTTP-mux executable price. There is no separate "prop feed" the platform could nudge to push an account into a -4% daily breach or out of the top 3 podium review — every challenge breach, every funding tick, and every PnL number is reproducible from public oracle data plus the published source-posture hash.
- **Degraded posture freezes fresh risk, not your account.** If primary and cross-check disagree beyond bounds, or freshness slips, the market drops to reduced or close-only. You can still cut size; you cannot add new exposure until the market promotes back to live. The rule engine

stamps the degraded window onto your evaluation timeline so the -4% daily / -8% total drawdown checks are not triggered by stale prints — but trading capacity inside the 30-day window is not refunded.

- **Session-based markets respect their schedule.** Equity, metal, and energy perps move to session-closed when the underlying market is closed. Equity is marked at the last valid session price; the daily drawdown rule resumes at the next session open. Overnight oracle noise on an off-session market cannot fail your challenge.
- **Settlement is bound to the same source posture.** Every withdrawal release on Base is anchored to a state root whose oracle-source hash matches the posture your trades priced under. The cashier and the matching engine cannot diverge. Profit-share payouts target **under 24h** from request to USDC arrival; the proof-gated mechanics are documented on Settlement and withdrawals.

CHAPTER 13

SOURCES AND FALLBACK

Normal operation on every Dexter market requires two healthy paths at once: a primary Hermes/Pyth push stream and an independent HTTP oracle mux that the runtime polls and cross-checks against the primary every tick. Single-source operation is not a steady state — it is an explicit liveness override engaged only when the runtime has logged the failure, recorded the policy exception, and downgraded the market to a degraded posture for the duration. The ladder below is therefore an ordered failover plan, not a buffet the mux picks from.

SOURCE PATH	ROLE IN THE VENUE
Hermes / Pyth (primary)	Low-latency push reference anchor for every supported market; aggregates the publisher set Pyth verifies on-chain
HTTP oracle mux (cross-check)	Independent pull path that must agree with Hermes within the deviation band on every tick before the mark is promoted
Single-path liveness override	Engaged only when one family is down and the runtime accepts reduced quorum; the market simultaneously drops to *reduced* or *close-only*
Last-known controlled hold	If no path is acceptable, the engine holds the last validated mark, halts new execution, and emits a degradation event for operations and the published state

#

How a source becomes active

A fallback path existing is not the same as it being trusted. The source mux makes one decision per tick per market: which combination of available paths clears the freshness window, agrees within deviation bounds, passes the per-market lag ceiling, and respects the depth/spread guard. That decision is policy, not heuristic — the same inputs always produce the same outcome, and the chosen source set is recorded in the runtime log and stamped into the next settlement root.

Conservatism is the explicit default. When the primary weakens, the mux first tries to keep both paths live by widening latency tolerance within the configured ceiling. If the cross-check still fails the deviation band, the runtime prefers to downgrade the market — to *reduced*, then *close-only*, then *halted* — rather than promote a thinner source set into normal posture. Convenience and pretty uptime numbers are not valid reasons to keep trading on a path the rest of the policy does not stand behind.

When a fallback is engaged, two things happen simultaneously. First, the matching engine continues to fill reduce-only orders against the surviving path so users can manage open risk. Second, the market state is published as degraded so every connected client, the rule engine, and the cashier see the same posture. A reviewer reading the runtime log later can answer one question without ambiguity: at every tick, which path was active, why, and what posture did the market trade under?

TEXT

```
tick T arrives
  → mux reads Hermes primary + HTTP cross-check
  → freshness, deviation, lag, depth gates evaluated
  → if both pass: mark promoted, market stays live
  → if one fails: mux retries within latency ceiling
  → if still failing: market posture downgrades
       reduced → close-only → halted
  → selected source set and posture hash
       committed inside the next state root
```

#

Why source selection matters for settlement

The active source set is not an operational footnote — it is part of the on-chain settlement record. The runtime hashes the source-posture metadata (active publishers, fallback engagement, deviation observed, freshness margin, market posture) and commits that hash inside every state root the vault contract accepts. A withdrawal finalized against root `R` is therefore bound to the exact oracle configuration that produced the balances inside `R`.

This is the difference between "the venue says it used Hermes" and "the contract can prove which feed posture every payout depended on." It is also what makes the price layer auditable in the same sense the balance ledger is: a post-incident reviewer reading state roots on Base can answer not only what an account was worth at root `R`, but which source paths were live, which were degraded, and which posture the matching engine was running under when that snapshot was sealed. Mark construction details continue on Mark construction; session and degraded-state transitions are documented on Sessions and degraded states.

CHAPTER 14

MARK CONSTRUCTION AND EXECUTABLE PRICE

The executable mark is constructed from five inputs every tick, applied in a fixed order. The reference anchor (Hermes/Pyth primary cross-checked against the HTTP oracle mux) sets the starting point. Venue skew, spread posture, depth checks, and max-move continuity then shape and constrain it. None of the four follow-on inputs can *push* the mark against the reference — they can only restrict it, widen it, or reject it.

INPUT INTO THE MARK	WHY IT MATTERS
Reference anchor	Hermes/Pyth primary plus the HTTP oracle mux cross-check, both within freshness window and deviation band — the only input that originates the mark level
Spread posture	The venue widens spread around the reference when liquidity tightens or volatility spikes, so the mark reflects what Dexter can actually fill, not the tightest tick visible outside
Skew and open-interest pressure	Heavy one-sided open interest produces a skew adjustment that prevents crowded positioning from being marked at a neutral price; this also shows up in the funding rate
Depth and book checks	If top-of-book depth falls below the per-market floor, the mark widens or the market downgrades — a thin book never gets to define the executable level on its own
Max-move and continuity guards	The mark cannot jump more than the per-tick drift cap from the prior accepted mark; a discontinuity that large triggers reduced posture and a stabilization window before normal trading resumes

#

From reference to tradable price

Every tick begins with the accepted reference path agreed between the Hermes/Pyth primary and the HTTP oracle mux cross-check. That reference number is not yet tradable — it describes the outside market, not Dexter's current capacity to fill against it. The engine then layers the four follow-on inputs in sequence: spread posture widens or tightens the band around the reference based on current venue liquidity, the skew adjustment pushes the mark toward whichever side is being squeezed by one-sided open interest, depth checks confirm the venue book can absorb minimum size at the resulting level, and the max-move continuity guard verifies the result is within the per-tick drift cap from the prior accepted mark.

Only after all four pass does the new mark become the executable level the matching engine will fill against, the rule engine will check drawdowns against, and the funding engine will sample. The executable mark is therefore venue pricing — a number Dexter can defend filling against right now — not feed ingestion. The reference says "the outside market is here." The mark says "this is what we can responsibly trade against, this tick, given current book and posture."

#

Executable mark versus external index

An external index — Pyth aggregate, an exchange index, a TWAP — describes where the outside market is printing. The executable mark describes where Dexter is willing to fill, mark equity, and trigger liquidations *this tick*. The two diverge whenever the venue's local conditions diverge from the outside picture: a thin book widens the executable spread even though the index keeps printing tight, a crowded one-sided OI pushes the executable mark off the index to bleed the funding rate against the crowded side, a sudden index gap larger than the per-tick drift cap is clipped to the maximum allowed move while the underlying recovers.

Under stress, the venue keeps the market open with tighter posture (*reduced*) before considering halting it. Under heavier stress it moves to *close-only*, then *halted*, instead of presenting the raw reference as if it were still fillable. Session-closed is a separate state used only when the underlying market is on a published schedule — equity, metal, and energy perps — and is documented on Sessions and degraded states.

TEXT

```
accepted source set (Hermes primary + HTTP cross-check)
  → freshness window + deviation band pass
  → reference anchor accepted
  → spread posture widens band around the anchor
  → skew pressure adjusts mark toward the squeezed side
  → depth guard verifies minimum book at the level
  → max-move continuity check vs prior accepted mark
  → executable mark becomes active
  → funding, liquidation, drawdown checks, leaderboard PnL
       all sample the same mark
```

#

Review standard for the mark

The right question to hold the price layer to is not "can Dexter show a number" — any feed can show a number. It is "can Dexter explain, after the fact, why a particular tick was safe enough to mark equity against, fund 90 / 10 funded accounts on, and trigger a -4% daily breach off." Mark construction is the answer: every executable mark is traceable to a specific reference pair, a specific spread and skew posture, a depth observation, and a continuity guard outcome — all hashed into the source-posture commitment that lives inside the next state root. A reviewer pulling root `R` from the Base contract can reconstruct which mark fired which liquidation, the same way the matching engine did at the time.

CHAPTER 15

SESSIONS AND DEGRADED STATES

POSTURE	WHAT IT MEANS INSIDE DEXTER
Live	Normal two-sided trading; primary Hermes/Pyth and HTTP cross-check both passing every gate; equity marks tick-by-tick
Reduced	Tighter caps, wider spread, lower max leverage; both source paths still live but stress (volatility, depth, latency) has crossed the soft band
Close-only	Reduce-only orders accepted; no new exposure; engaged when single-path liveness override is active or depth has collapsed
Session-closed	Scheduled off-session for the underlying market (equity/metal/energy perps); equity holds at the last valid session close; daily drawdown rule resumes at next session open
Halted	No execution at all; price path cannot be defended (no acceptable source set, or recovery lag still active); equity frozen at the last valid mark until the market promotes back

#

Session closure is not oracle failure

NYSE closes at 16:00 ET. LBMA gold fixing happens twice a day. Brent crude has weekend gaps. Folding any of that into a generic "oracle is unhealthy" label would punish funded traders who have done nothing wrong — their equity would be marked against stale references, the -4% daily rule would fire on phantom prints, and a passed evaluation could blow up between 17:00 ET on Friday and 09:30 ET on Monday because of a feed condition that has nothing to do with the price layer.

Dexter therefore models session-closed as its own posture with its own behavior. The market is intentionally not trading. Open positions are held at the last valid session-close mark. Drawdown checks pause, then resume at the next session open against the new opening mark — meaning a -4% daily rule is checked close-to-close on session markets, not over the closed gap. Funding does not accrue during session-closed windows on equity perps. The session calendar is published per market in the symbol picker and stamped into the runtime so users, the rule engine, and the cashier see the same schedule.

A session-closed market is a healthy market. A halted market is the engine telling you it cannot defend a price right now. The whole point of separating the two postures is that the difference must be visible — both to you and to a reviewer reading runtime state weeks later.

#

How degraded states are handled

Degraded postures fire when the price path itself is the problem — freshness slips beyond the per-market staleness cap, Hermes and the HTTP mux disagree past the deviation band, depth at the front of the book collapses, the spread guard widens past its ceiling, or the max-move continuity check rejects a tick. Severity determines the destination: a single weakened input pushes the market to *reduced*; loss of one source path pushes it to *close-only*; no acceptable source set at all pushes it to *halted*.

Recovery is deliberately not symmetric. One fresh tick from a recovered publisher does not promote a halted market back to live, because that would let users trade into the seam between "the feed printed something" and "the feed is stable enough to mark equity against." Each posture has a per-market stabilization period that must pass with continuous clean ticks before the market promotes one step. A halted market promotes to close-only first, then reduced, then live — never directly back to live. The same recovery lag applies after a degraded window closes on a session-based market, after the underlying session reopens.

While a market is in reduced or close-only, the rule engine stamps the window onto the funded account's evaluation timeline so the -4% daily and -8% total checks are not triggered by stale or thinned-out prints. The 30-day clock is not extended — degraded windows inside your evaluation period remain dead trading time you cannot recover — but a false breach caused by a posture event will not end the attempt.

TEXT

```
 underlying session closes
   → market posture = session-closed
   → equity held at last session-close mark
   → drawdown rule pauses until next session open

price path weakens
   → live → reduced  (single guard breached)
   → reduced → close-only (one source path lost)
   → close-only → halted (no acceptable source set)

conditions recover
   → halted → close-only after stabilization window
   → close-only → reduced after stabilization window
   → reduced → live after stabilization window
```

#

Why visible degradation matters

Every posture is surfaced on the symbol page, the order ticket, the rule-engine HUD, and the cashier — with the same vocabulary in each surface. A funded trader looking at a session-closed equity perp sees the next session open. A trader on a reduced BTC market sees the relaxed leverage cap and the wider posted spread. A trader holding through a halted state sees the last valid mark frozen on their PnL line, not a phantom number trending against them. Identical posture labels also flow into the published settlement context, so the post-incident reviewer can answer not only "what was the mark" but "what posture was the market in" at any sealed root.

A venue that conflates scheduled closure with broken pricing eventually communicates both badly, and the funded-trader contract loses its meaning the moment a -4% breach can fire from an off-session print. Separating the five postures is not stylistic — it is what makes Dexter's drawdown rules, payout pipeline, and source-posture commitments defensible together.

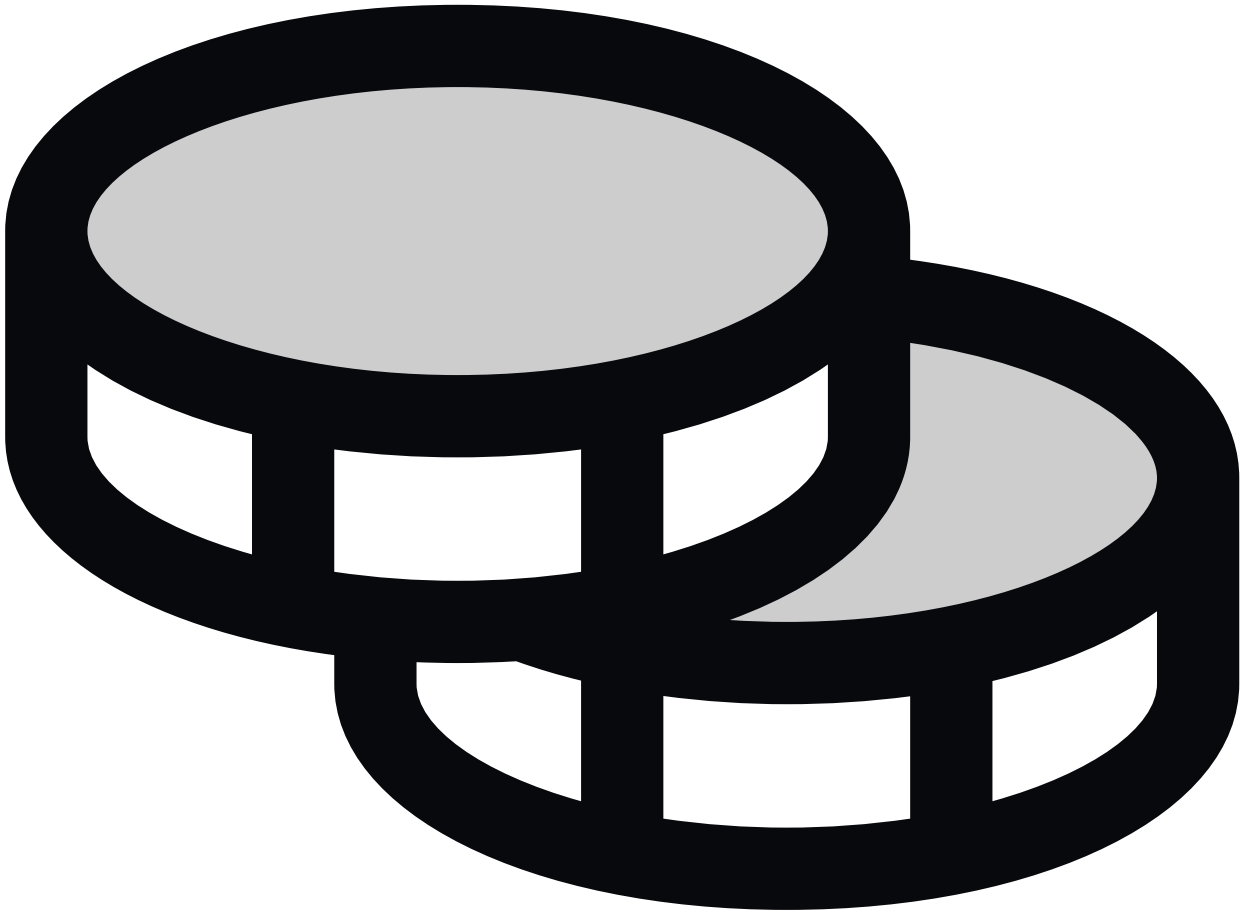
CHAPTER 16

VAULT AND TREASURY

The four balances answer four different questions and move under four different sets of permissions. The table is the contract layout, not a marketing abstraction — each row is a deployed address on Base with its own signer policy and its own accounting trail.

VAULT	WHAT IT HOLDS	HOW IT IS MEANT TO MOVE
User collateral vault	USDC deposits, open margin, pending withdrawal requests, and the per-account nonce trail	Released only against a published state root and a matching Merkle proof; no operator key can override the proof gate
Operating treasury	Fee revenue accrued from trading activity and protocol-owned working capital	Outflows above $10K require 2-of-3 multi-sig; recipients are governed and visible on Basescan
Insurance reserve	The 50% liquidation residual share, plus any seeded buffer for stressed conditions	Draws require 3-of-5 multi-sig including an external signer; every adjustment is explicitly bucketed
Reward pool	Season leaderboard cash and posted prize brackets, funded before the season opens	Released against the published bracket on season close; podium (top 3) finishes reviewed before the batch leaves treasury

#



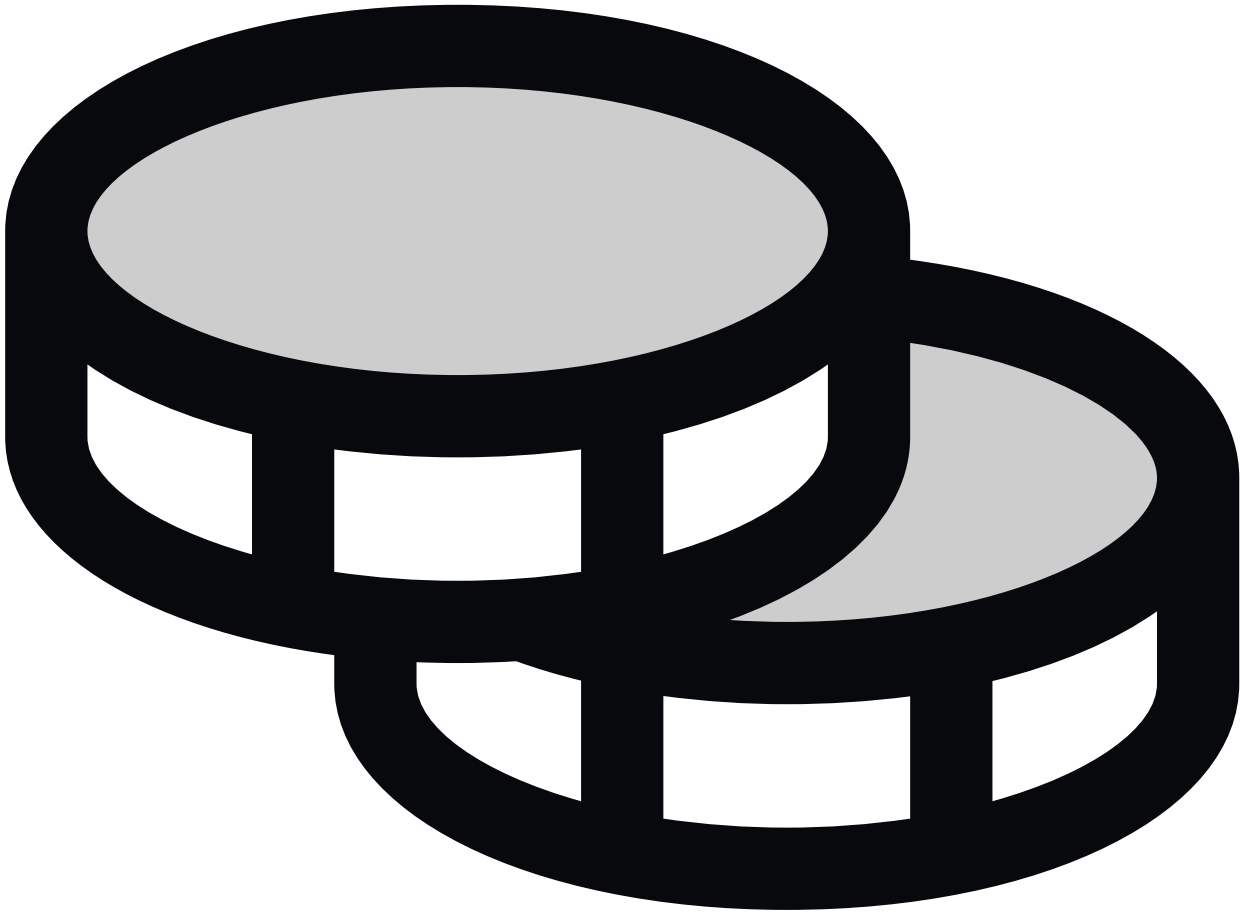
Custody, settlement, and protocol balances

User collateral is held in the dedicated vault contract on Base. Deposits, withdrawal requests, the per-account nonce trail, and the finalize step that releases USDC back to the wallet all pass through contract logic — never through the runtime alone. The runtime is fast enough to coordinate matching and risk in real time, but it cannot rewrite custody outcomes, because every release is gated by a Merkle proof against a state root the vault contract has already accepted. The full request-to-release flow is documented on Settlement and withdrawals; what matters here is that the runtime and the vault are deliberately decoupled.

The vault stores more than token balances. It also stores the settlement references that describe what the runtime believed when each state update was sealed: the active state root, the root id, the publication timestamp, the schema version, the sequence commitment, the oracle-source hash from the price layer, and the order-batch commitment. Root history is retained on-chain so any settlement transaction can be tied back to a specific published exchange state — not to a vague claim that the operator says the ledger changed.

The other three vault addresses live in parallel, with their own permission policies and their own accounting buckets. Operating treasury holds fee revenue from trading activity. Insurance reserve holds the 50% liquidation-residual share plus any seeded buffer used to absorb stressed outcomes — the other 50% of every liquidation residual goes to the keeper that triggered the routine. Reward pool holds posted leaderboard cash for the active season, funded before the season opens so the bracket cannot be silently reduced mid-season. None of these are user margin, and none of them release under user-withdrawal rules.

#



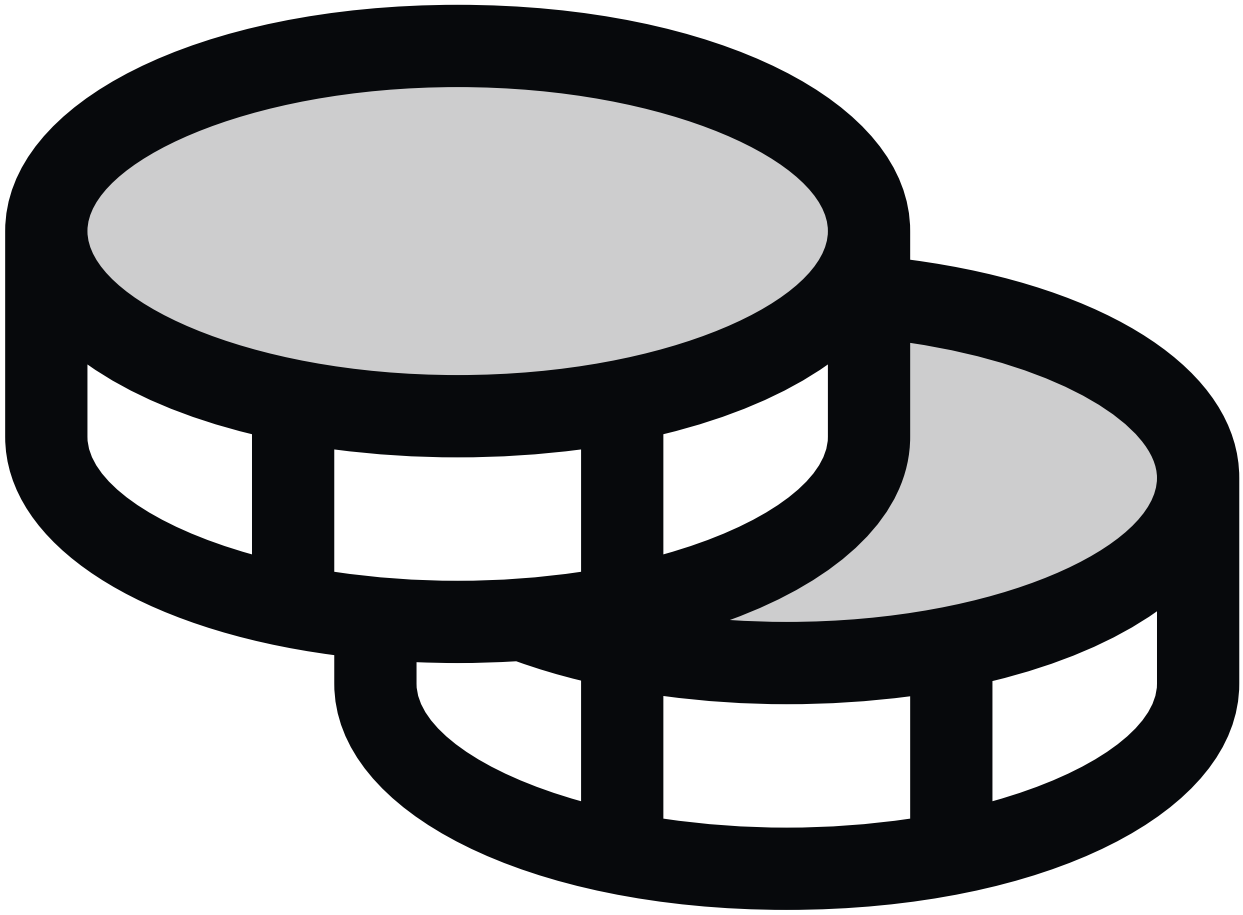
Why custody and protocol value stay separate

This architecture does not claim the vault is impossible to attack — no serious venue should promise that. It does make any single compromise less powerful than a compromise of the whole system. A runtime fault cannot move user collateral, because the vault only releases against a proof matching an already-accepted root. A signer compromise on the operating treasury cannot pay out the reward pool, because the reward pool is a different contract with a different signer set. A spike of liquidations on a single funded account cannot drain the operating treasury, because the loss draws from insurance under a separate accounting bucket.

Permission policy is explicit at each layer. Operating treasury outflows above $10K require 2-of-3 multi-sig from the operations group. Insurance reserve draws require 3-of-5 multi-sig that must include an external signer outside the operations group, so an insider-only quorum cannot deplete the reserve. Governance actions — parameter changes, contract upgrades, recipient changes — run on the same 3-of-5 external-signer requirement. Pause flags and root-guard logic give the contract layer a way to freeze releases while a dispute or incident is investigated, without those flags also enabling unilateral movement.

The result is a layout a reviewer can read in three questions: where is user money (the collateral vault, released by proof), what belongs to the protocol (operating treasury and insurance reserve, with their own signer policies), and what must be true before any balance moves (the on-chain rules listed above). No part of the answer requires trusting the operator's word.

#



How this protects your payouts

- **Your collateral cannot fund someone else's payout.** The user collateral vault releases USDC only against a published state root and a matching Merkle proof. The operating treasury cannot reach into it to cover a leaderboard bracket, a referral cash batch, or an insurance shortfall. The cashier UI shows the root id and proof at finalize time; you can verify the on-chain release against the same root yourself on Basescan.
- **Insurance is a separate contract with its own signer policy.** Every liquidation residual is split **50% to the keeper that triggered it, 50% to insurance**. Bad-debt spillover from a failed funded account draws on insurance, not on the operating treasury and not on user collateral — which

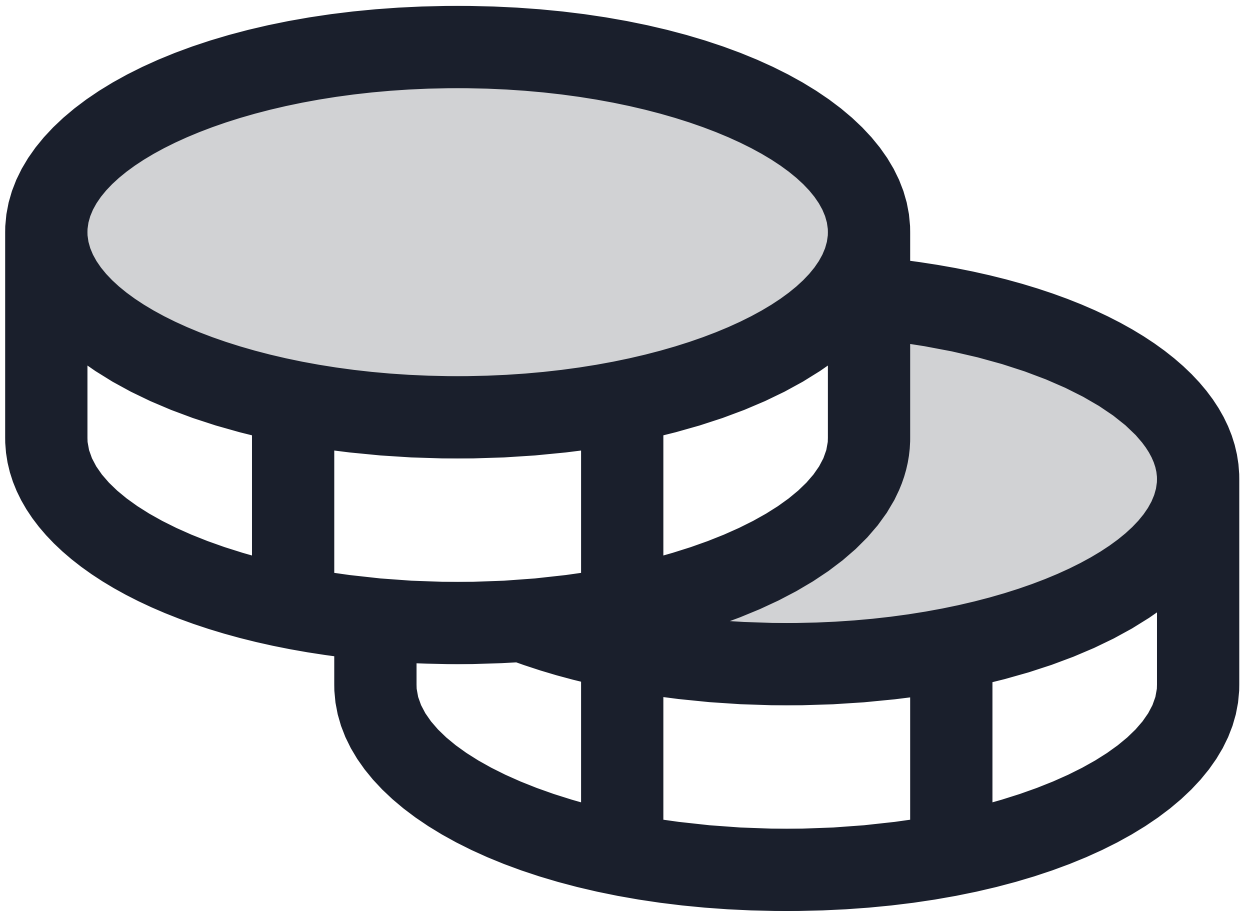
is why a chain of liquidations does not slow down legitimate withdrawals. Insurance draws themselves require 3-of-5 multi-sig including an external signer.

- **The reward pool is funded at season open, not at season close.** Leaderboard cash for the active season is moved into the dedicated reward-pool contract before the season opens. The published brackets ($X for #1, $Y for #2–3, etc.) cannot be quietly reduced mid-season because the cash is no longer in the operating wallet. Podium finishes (top 3) are reviewed before the season-close batch transaction leaves treasury; single withdrawals above $5K trigger the same high-value review.
- **Multi-sig gates every protocol-side movement.** Operating treasury outflows above $10K require 2-of-3; insurance reserve draws and governance actions require 3-of-5 with an external signer. Every transaction is posted on Base, so a sudden movement out of any of the four vaults is visible to anyone watching the address.

Each vault address is published on the security controls page and can be pinned to a Basescan watchlist. If the dexter.market "Verified Rewards" leaderboard total ever disagrees with the sum of transactions out of the reward pool address, the on-chain record wins — and you will see the discrepancy before we tell you about it. The settlement mechanics that release user collateral are documented on Settlement and withdrawals; the accounting flows for fees and insurance are documented on Treasury and accounting.

CHAPTER 17

STATE ROOTS, WITHDRAWALS, AND SETTLEMENT FLOW



Settlement timeline **Custody moves through proof, not trust.**
Withdrawals advance from request to release only after root reference, proof, and challenge rules line up.

Request **Nonce and account**

A withdrawal enters the settlement queue.

Root reference **Published state root**

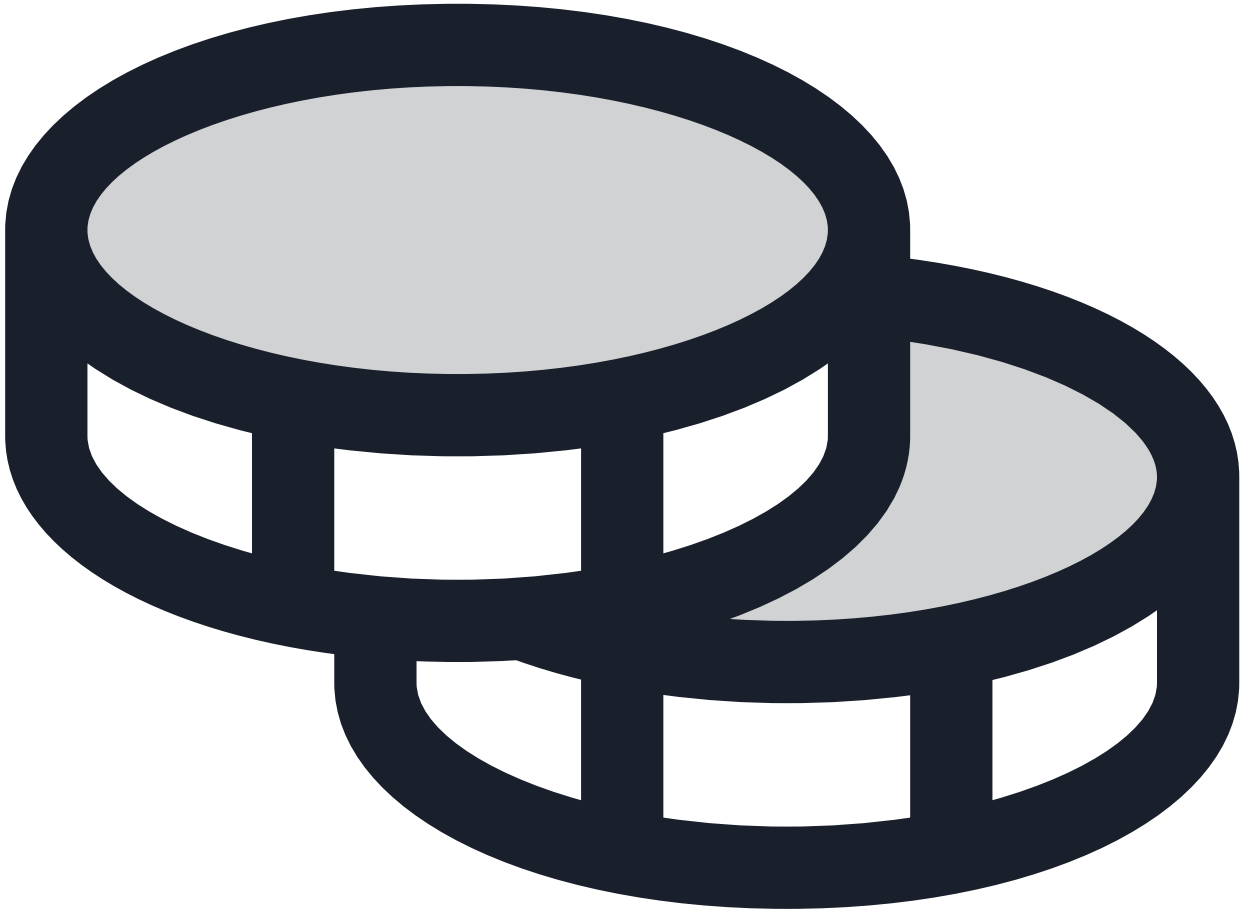
The claim binds to a committed balance snapshot.

Settlement gate **Proof-gated release**

The vault only releases collateral after proof and posture checks pass.

Proof window **Challenge and dispute**

Invalid claims can be contested before final release.



Release **Collateral exit**
Only verified settlement can leave the vault.

Every state root the vault accepts is more than a balance snapshot. Five references travel with it, so a single root commitment binds not only what each account was worth but the entire context the runtime was operating under when the snapshot was sealed. A reviewer reading the contract later can reconstruct the exact configuration that produced any given payout.

STORED REFERENCE	WHY IT EXISTS
State root	Merkle root over all account balances and pending withdrawal requests at publication time; the leaf you finalize against
Root id and timestamp	Identifies which publication event sealed this snapshot, ordered on-chain, so settlement is tied to a specific commitment and cannot be retroactively reassigned
Sequence commitment	Binds the ordering of events the runtime processed into this snapshot — no insertion of after-the-fact trades
Oracle-source hash	Hash of the price-layer posture (active publishers, fallback engagement, market posture) under which the snapshot was produced, bound to the same root the cashier releases against
Order-batch commitment	Binds the batch of orders processed in the publication window so the trades feeding this root cannot be rewritten later

#

How settlement actually works

The vault contract retains the active root, the root id, and the surrounding commitments so the contract side carries an explicit, queryable record of every published exchange state. Settlement is therefore never resolved against what the runtime believes in the last second — it is resolved against a committed state that can be referenced from chain, challenged on chain, and proven against on chain. The runtime is fast; the vault is final.

Withdrawals are request-based and nonce-tracked. The user signs a withdrawal request on-chain, which mints an account nonce the contract tracks. The runtime picks that request up in the next publication cycle, includes it inside the snapshot that produces the next state root, and posts the root with the five bound references listed above. The user (or any wallet acting on their behalf) then calls the finalize step with the Merkle leaf, the proof path, and the matching nonce; the vault contract verifies the proof against the committed root, confirms the nonce is current, checks the proof window has elapsed, and releases USDC.

This is materially different from a model where an operator decides when funds leave. The runtime cannot release funds it failed to seal into a root. The vault cannot release against a root that has not passed its proof window. An operator with full keys to the runtime still cannot push a payout through without producing a coherent committed state plus a matching proof. That layering — request, root commitment, proof window, contract release — is what makes the on-chain payout path defensible without trusting any single actor.

TEXT

```
requestWithdraw(amount)
  → on-chain account nonce is created
  → runtime includes the request in the next snapshot
  → state root + 4 bound commitments published on Base
  → proof window opens (challenge period)
  → user receives leaf data and Merkle proof from runtime
  → finalizeWithdraw(leaf, proof, nonce)
       → vault verifies proof against committed root
       → vault verifies nonce matches account state
       → vault confirms proof window has elapsed
       → USDC released to wallet on Base
```

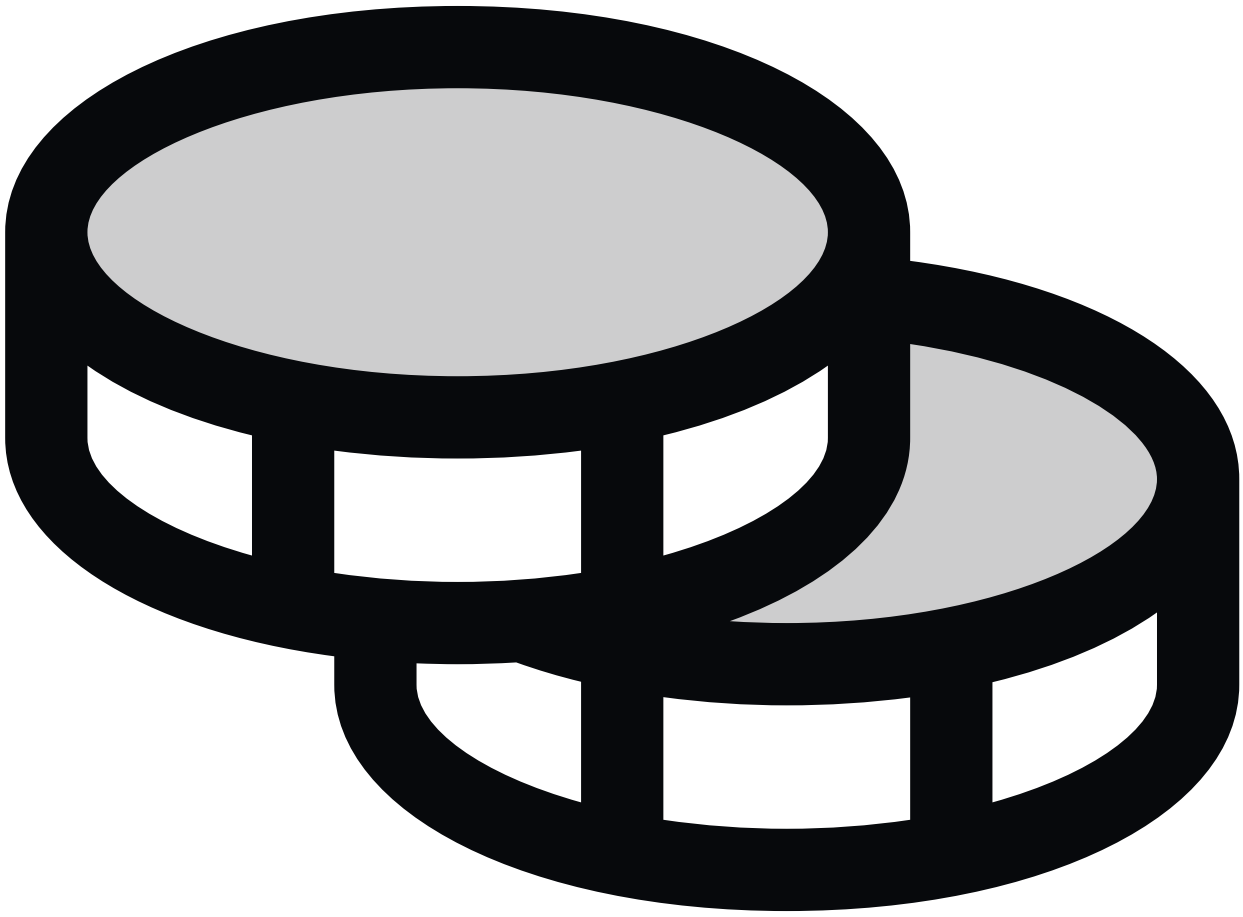
#

Proofs, challenges, and failure handling

Proofs matter because the vault is not designed to trust runtime claims blindly. Between root publication and finalize, a proof window opens during which any party can dispute a questionable root or a malformed settlement transition. Root history is retained on-chain so a challenge can reference a specific commitment by id, not just a vague accusation. Guardian controls and pause flags can freeze further releases against a root under dispute. Fraud-verifier hooks and root-guard logic exist to make a bad state update visible and contestable before value leaves — not just to log it after the fact.

This does not justify claiming the system is unhackable. No serious venue should make that promise. It does justify saying that Dexter is engineered so a successful withdrawal requires four assumptions to hold at once: the runtime must publish a coherent state, the vault contract must accept the corresponding root, the request nonce must match the user's on-chain account state, and the proof path must verify against the committed root. A compromise of any single layer in isolation does not produce an unauthorized payout. That is the layered settlement model — fewer single points of failure, more places a dispute can intercept a bad outcome.

#



Payout timeline you should expect

- **Profit-share withdrawal (funded account, 90 / 10 split).** Request from the cashier in-app against your funded equity. If KYC is already cleared on the wallet, the target is **under 24 hours** from request to USDC arriving on Base — request enters the next publication cycle, root posts, proof window elapses, finalize step releases. The first profit-share withdrawal triggers KYC if it has not been completed yet; plan 1–2 extra days the first time.
- **Leaderboard cash.** Posted at season close from the dedicated reward-pool contract, paid in a single batch transaction per bracket. Wallets that have already cleared KYC for a profit-share withdrawal do not re-verify. **Podium finishes (top 3) are reviewed before the batch leaves treasury** — every season, no exceptions — so a leaderboard exploit cannot bypass operations review by ranking #1.
- **Referral cash.** Paid monthly in batch on the first business day of the month, anchored to the upstream wallet. Every transfer carries the referee payment id and the Basescan tx hash, so the link between an attribution event and the USDC that landed in your wallet is end-to-end auditable.
- **High-value withdrawals ($5K+).** Any single request above **$5,000** is reviewed by operations before the proof window opens. This adds a few hours but does not change the on-chain settlement path — the proof, the root reference, and the contract release are identical.

The proof-gated design has one important consequence: even with full access to the cashier and the runtime, operations cannot push a payout through that does not have a valid root reference and a matching Merkle proof. The cashier UI surfaces the published root id and the proof at finalize time; you can paste either into a Basescan read call and verify the on-chain release against the same root yourself. If the published bracket total on dexter.market disagrees with the sum of transactions out of the reward-pool address, the on-chain record is the truth and the dashboard is wrong — not the other way around. Vault accounting (how fees, insurance draws, and reward-pool funding move) is documented on Treasury and accounting.

CHAPTER 18

TREASURY, INSURANCE, AND INCOME ACCOUNTING

BALANCE BUCKET	ROLE INSIDE THE PROTOCOL
feesAccrued	Protocol revenue from maker/taker fees, funding spread, and challenge-pack entry; lands in the operating treasury contract on Base
insuranceReserve	50% of every liquidation residual plus any seeded buffer; absorbs bad-debt spillover and stressed outcomes without touching user collateral or operating treasury
rewardPool	Funded at season open from operating treasury into the dedicated reward-pool contract; backs the published leaderboard brackets for the active season
Income-side balances	Participation-linked accounting surface for DIP holders and revenue-share recipients; layered on top of fee/insurance accounting, never on top of user collateral

#

How protocol-owned value is handled

Trading fees, funding spread, and challenge-pack entry revenue ($49 / $99 / $199 / $299 packs) accrue into protocol-owned accounting from the moment they are collected — they never sit inside the user collateral vault and never get touched by user-withdrawal logic. The runtime tracks fee and insurance totals in USD18 precision in its own accounting model; a reconciler snapshots the deltas and pushes only those deltas on-chain via `vaultAddFees`, `vaultAddInsurance`, and `vaultSubInsurance`. The vault contracts therefore never try to recompute the full exchange ledger; they record the protocol-owned balances that matter for settlement, governance, and reserve coverage.

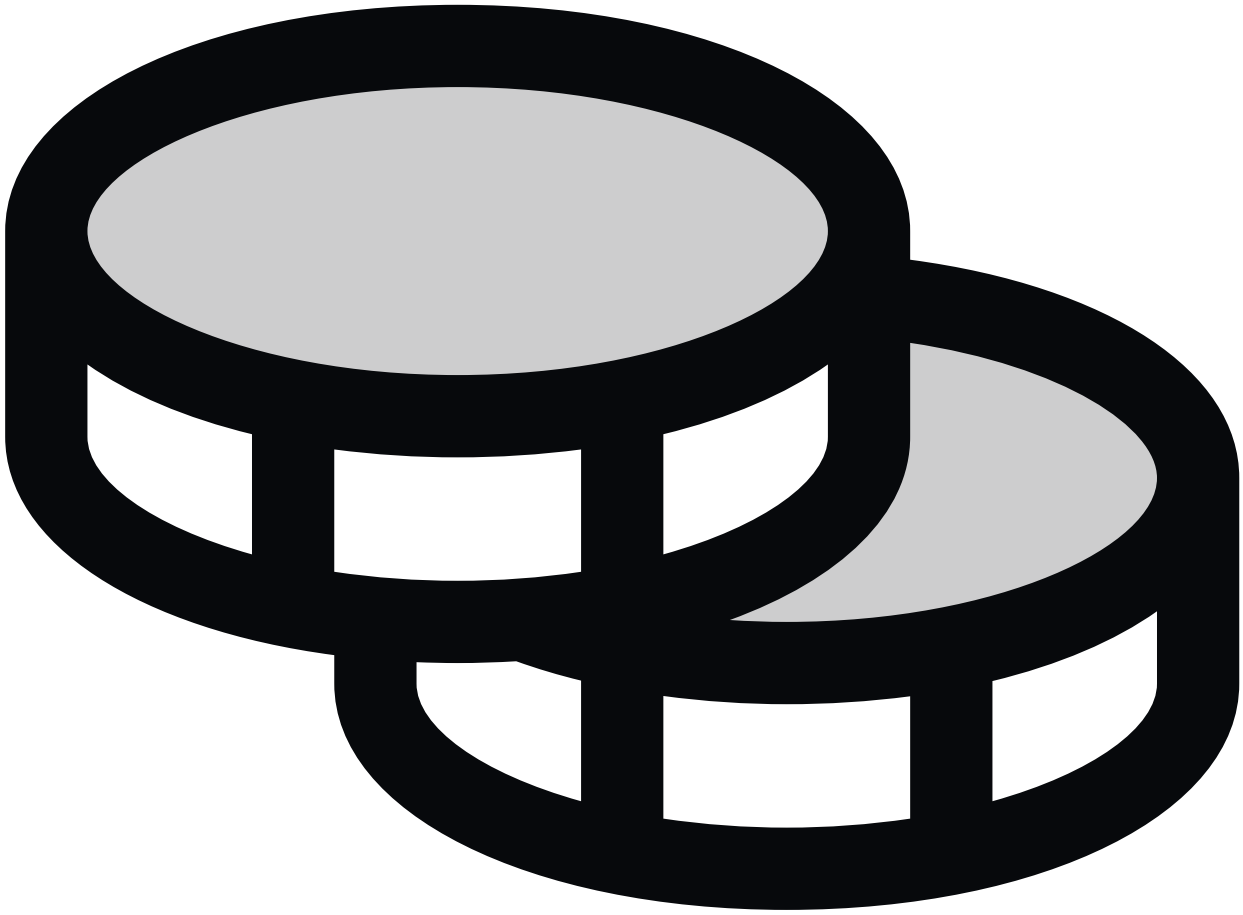
Insurance and treasury are deliberately not collapsed into one "protocol reserve" number. Insurance capital exists for a specific job: absorb the stress of failed funded accounts, bad-debt spillover from violent liquidations, and the post-liquidation residual when a liquidation closes below maintenance. Every liquidation residual is split **50% to the keeper that triggered the routine, 50% to the insurance reserve** — keeper incentives stay aligned with timely liquidations while the reserve refills from the same flow. Operating treasury, by contrast, is protocol revenue that can be routed through governed recipients: reward-pool funding at season open, payroll, infrastructure spend, runway. Conflating them would let an operations action accidentally deplete insurance, which is exactly what the separate accounting and the 3-of-5 external-signer requirement on insurance draws prevent.

The fee-siphon path is the optional bridge into income-side accounting. When governance enables it, a configured share of `feesAccrued` is siphoned on the same reconciliation tick into the income-side layer described below — but only after the underlying fee delta has landed on-chain. The siphon never operates ahead of the on-chain commitment.

TEXT

```
engine fee / insurance totals (USD18 precision)
  → reconciler snapshots deltas vs last on-chain checkpoint
  → deltas converted to collateral units
  → vaultAddFees      (operating treasury contract)
  → vaultAddInsurance (insurance reserve contract, +50% residual)
  → vaultSubInsurance (insurance draw, requires 3-of-5 external signer)
  → optional fee siphon → income-side accounting layer
```

#



Income-side accounting and why it stays separate

The income-side accounting layer tracks participation-linked balances — DIP holder claims, revenue-share recipients, and any future programmatic distribution rule governance enables. It sits on top of the fee and insurance accounting described above and depends on those numbers being correct underneath it. Revenue can be siphoned, reserved, or routed into the income-side layer under explicit rules, but the layer itself is not custody. It is an accounting surface that publishes per-participant claims; the underlying USDC still lives in the operating treasury contract until a claim is finalized.

This separation is what lets the protocol talk about revenue, insurance, treasury, and participation as four distinct things — not as one marketing-friendly "reserve" number. It also makes security review cleaner. If a governance action misconfigures one parameter, the reviewer can see which buckets are affected and which are not without inferring from logs. If a runtime fault produces a wrong fee delta, the on-chain reconciliation gap is visible against the next snapshot and can be corrected by a targeted `vaultAddFees` or `vaultSubInsurance` call without disturbing the other balances.

The finance model is therefore part of Dexter's safety design, not just its economics. Four vault addresses, four sets of permissions, four independent accounting flows, and one reconciliation discipline that ties them all back to the runtime totals — that is what lets a funded trader, a leaderboard winner, a DIP holder, and a referral wallet all be paid out of the same protocol without the four flows ever stepping on each other.

CHAPTER 19

SECURITY

Authority is split across five layers so no single component can move treasury value or finalize a withdrawal on its own assertion. Each layer has a different signer set, a different blast radius, and a different recovery path.

SECURITY BOUNDARY	WHAT IT PROTECTS
Runtime discipline	Single-writer ordering on fills, funding, liquidations, and mark updates; no competing writers can alter exchange state
Gateway boundaries	Public reads, authenticated account views, signed trading routes, and operator-only routes each carry distinct auth — a public token never reaches a treasury surface
Proof and challenge rails	Withdrawals finalize only against a published root, a matching Merkle proof, a bound nonce, and a satisfied dispute window
Vault and governance multi-sig	2-of-3 operations multi-sig on movements above $10K; 3-of-5 governance and insurance with at least one external signer for parameter and reserve changes
Monitoring and recovery	Continuous feed-freshness, root-liveness, gateway-health, and treasury-reconciliation signals with guardian pause authority on any one of them

#

Security is not one control plane

Exchange state advances through a single-writer runtime that owns funding accruals, mark updates, liquidation triggers, skew control, and the publication of settlement state. Nothing else is allowed to mutate that state in parallel, which is the prerequisite for every downstream guarantee — a proof can only be trusted if there is exactly one canonical history it can prove against.

The gateway then exposes that state through five distinct permission surfaces: anonymous market reads, wallet-authenticated account views, signed trading flow, operator-only telemetry routes, and value-transfer routes that require both signature and a server-side allowlist. No token issued for

one surface is honored by another, so a compromised product session never inherits treasury authority and a public read key never inherits operator visibility.

On Base, the contract layer adds a second perimeter that the runtime cannot override. Guardian pause flags can halt deposits, withdrawals, or the entire settlement pipeline within a single transaction. Root guards reject any commit whose timestamp, nonce, or referenced parent disagrees with on-chain history. Fraud-verifier hooks and the configurable disputeWindowSec keep questionable roots in a contestable state until the window expires. Together they guarantee that settlement always slows or stops when runtime conditions are stale, disputed, or under review — even if the runtime itself believes everything is fine.

#

Why proof-based exits matter

USDC leaves the vault only when four independent conditions agree at the same block: the user's requestWithdraw nonce exists on-chain, the runtime has included that request in a settlement root, the submitted Merkle proof reconstructs the user's leaf against that exact root, and disputeWindowSec has elapsed without a successful challenge. A balance the runtime believes it owes you is not sufficient. The contract must independently verify that the published state actually says so.

This matters because route authentication and state correctness are different threats. A stolen operator key would let an attacker call privileged endpoints, but it cannot fabricate a Merkle proof against a root whose leaves are publicly reconstructable. A buggy runtime might publish a wrong root, but the dispute window keeps it contestable through challengeRootFraudProof, challengeRootInvalidLeaf, or challengeRootConflict before any USDC moves. Both perimeters have to fail at once for a bad payout to land.

TEXT

```
runtime stays single-writer
  → gateway exposes separated auth surfaces
  → root and commitment context are published on Base
  → withdrawals require nonce + proof + root history
  → disputeWindowSec stays open for fraud-proof challenges
  → only then does finalizeWithdraw release USDC
```

#

What security means in practice

For a funded trader, the security model determines a small number of concrete things. Profit-share withdrawals up to $5K and outside the leaderboard podium top three are queued for automated processing and target release within 24 hours of Sumsub-class KYC clearing. Anything above $5K, any podium top-three payout, and any first-time payout above $50 cumulative are routed to manual review by the operations multi-sig — the same 2-of-3 threshold that signs every treasury movement above $10K. Wallets from the IR, KP, SY, and CU jurisdictions are blocked at request time, and every request is screened against OFAC, EU, and UK sanctions lists before it reaches the queue. Nothing about that flow is optional or negotiable.

What you should expect from a venue worth trusting with funded capital: a stale price never looks tradable, a constrained market never looks live, a treasury route never inherits a public read's permissions, and a withdrawal never finalizes on an operator assertion alone. Dexter is designed so failure is loud and contestable instead of silent — and so every sensitive transition has to pass through more than one healthy assumption at the same time.

CHAPTER 20

CONTROLS AND PERMISSIONS

Six tiers, each with a distinct credential type, a distinct blast radius, and a distinct threshold. A token issued for one tier never authorizes an action on another.

CONTROL LAYER	CREDENTIAL AND THRESHOLD	SCOPE
Public venue surface	None — anonymous read	Order books, mark prices, funding history, leaderboard standings, challenge stats
Wallet-authenticated routes	SIWE signature against the connected wallet	Account balances, position state, fills, withdrawal requests, KYC submission
Signed trading flow	EIP-712 typed order signature per intent	New orders, cancels, margin-mode changes, stop and limit updates
Operator telemetry	Server-side allowlist plus rotating API key	Runtime health endpoints, queue depth, internal reconciliation views
Operations multi-sig	2-of-3 threshold above $10K per movement	Routine treasury rebalancing, fee sweeps, manual-review payouts, podium top-three releases
Governance and insurance multi-sig	3-of-5 with at least one external signer	Parameter changes, fee tier updates, insurance reserve moves, guardian rotation, contract upgrades

#

How authority is segmented

The operations multi-sig is the workhorse signer. It holds three keys — held by the founding team, the on-call engineering lead, and the head of operations — and signs every treasury movement above $10K. That includes routine rebalancing between the operating wallet and the deep-cold vault, the weekly fee sweep, every manually reviewed funded-trader payout, and every podium top-three release on the seasonal leaderboard. Movements below $10K to whitelisted destinations (gas top-ups, on-ramp reconciliation, recurring infrastructure spend) can clear with a single key on a rate-limited automation account, but only against destinations pre-approved by the same 2-of-3.

The governance and insurance multi-sig is deliberately slower and broader. Five keys, three required to sign, and at least one of those three must be an external party — currently a partner from an independent security firm with no operational role at Dexter. Anything that changes how the venue behaves passes through this signer set: fee tiers, insurance-reserve rebalancing, disputeWindowSec adjustments, guardian-role rotation, oracle source promotion or demotion, and any contract upgrade. The external signer is the structural reason a unanimous Dexter team still cannot change protocol parameters unilaterally.

Guardian authority sits underneath both multi-sigs. Any of three guardians can pause deposits, withdrawals, or the full settlement pipeline within a single transaction, on its own key, with no threshold required. Pausing is one-way — only the 3-of-5 governance multi-sig can unpause, which is intentional: the cost of an unjustified pause is short downtime; the cost of an unjustified unpause is a bad withdrawal.

TEXT

```
 anonymous read   → market data only
wallet-signed    → account state, withdraw request
EIP-712 order    → trade intent, never balance change
operator key     → telemetry, no on-chain authority
ops 2-of-3       → treasury movement > $10K
governance 3-of-5 → parameter, reserve, upgrade
guardian (1)     → pause only, unpause needs 3-of-5
```

#

What segmented controls achieve

The threshold design exists to make specific compromise scenarios survivable. A stolen wallet signature reaches at most one user's account state — never another user's, never the treasury, never a parameter. A stolen operator API key surfaces telemetry but cannot move USDC, sign a withdrawal, or alter a fee tier. A single compromised operations key cannot release funds because the second signer reviews the destination, the amount, and the queue context before co-signing. A compromised majority of operations keys still cannot change protocol parameters because that path is governance-only and requires the external signer to sign as well.

Every multi-sig transaction is posted to Base with the destination address, amount, calldata, and signer set visible in the Basescan transaction log. Users with Basescan watch alerts on the operating wallet and treasury vault receive notification within seconds of every signed payout. Governance proposals are published with the proposed calldata 24 hours before the 3-of-5 quorum is requested, so any community member can independently simulate the call before it lands. There is no off-chain consent layer that supersedes what the Base ledger shows.

CHAPTER 21

WITHDRAWAL PROOFS, DISPUTES, AND ROOT CHALLENGES

Seven contract primitives compose the proof-based exit. The first four are required for every withdrawal; the last three are available to any address with standing to challenge a published root.

MECHANISM	WHAT IT DOES
requestWithdraw(amount)	Emits an on-chain nonce that binds the user's wallet, amount, and timestamp to a specific withdrawal intent — no nonce, no exit path
State root history	Append-only ring of recent settlement roots; finalizeWithdraw must reference a root still in this history, which makes silent replacement detectable
Merkle proof + leaf	User-side leaf data (wallet, balance, nonce list) plus the sibling path that reconstructs the referenced root — reconstruction failure reverts the call
disputeWindowSec	Configurable challenge window between root commit and earliest finalizeWithdraw; tuned to balance fast payouts against contest time
challengeRootFraudProof	Submits a fraud-verifier output proving the runtime committed a root inconsistent with valid state transitions
challengeRootInvalidLeaf	Submits a counter-leaf showing a specific account balance in the published tree contradicts what the runtime owes
challengeRootConflict	Submits two roots whose declared parent or sequence numbers contradict each other, exposing a history fork

#

The withdrawal path is proof-gated

The user starts by calling requestWithdraw with the USDC amount they want to exit. The contract emits a nonce that binds the request to the user's wallet, the amount, and the block timestamp. That nonce is the only thing the runtime is allowed to act on — a runtime-initiated payout to a wallet that never called requestWithdraw cannot pass finalizeWithdraw because no matching nonce exists in the Merkle leaf.

The runtime then includes the request in the next settlement state. When that state is committed to Base, the new root enters the append-only history ring and disputeWindowSec begins. The gateway exposes the user's leaf data and the sibling path needed to reconstruct the root, so the user (or any client they trust) can verify the proof off-chain before submitting it. Once disputeWindowSec elapses without a successful challenge, finalizeWithdraw is callable: the contract re-runs the Merkle reconstruction on-chain against the referenced root, checks that the nonce matches and has not been consumed, verifies the root is still in history, and only then transfers USDC to the user's wallet.

If a withdrawal sits past the 24-hour target, the failure is observable. Either disputeWindowSec is still open (visible on the contract), the root has not yet been committed (visible on Basescan as the absence of a commitSettlement transaction since requestWithdraw), or the manual review queue has flagged the payout for podium top-three or above-$5K verification. In every case the request remains queued and finalizable as soon as the gating condition clears — nothing about it expires.

TEXT

```
requestWithdraw(amount)
  → on-chain nonce binds wallet + amount + timestamp
  → runtime includes request in next settlement root
  → commitSettlement(root) posts to Base, enters history ring
  → disputeWindowSec begins
  → gateway exposes leaf + sibling path for off-chain verification
  → disputeWindowSec elapses without successful challenge
  → finalizeWithdraw(nonce, leaf, proof, rootRef) releases USDC
```

#

Why challenges and fraud proofs exist

Publication is not the same as correctness. The dispute window exists so anyone who can construct a counter-example has standing to stop a bad root before USDC moves against it. Three challenge paths cover the three failure modes a malicious or buggy runtime could introduce. challengeRootFraudProof targets the case where the published root could not have been produced by valid state transitions — the fraud-verifier hook runs the disputed step on-chain and rejects the root if its output disagrees. challengeRootInvalidLeaf targets the case where a specific account leaf misstates a balance the runtime is supposed to owe; the challenger submits the counter-leaf and

the contract compares signatures and accrued balances. challengeRootConflict targets history forks: two committed roots whose parent references or sequence numbers contradict each other are mutually exclusive and the contract invalidates the later commit.

A successful challenge invalidates the disputed root, freezes any withdrawals that referenced it, and forces the runtime to recompute and recommit from the last known-good state. A failed challenge consumes the challenger's posted bond and leaves the root in place. The economic asymmetry — bond loss for false alarms, vault-scale impact prevention for correct alarms — is deliberate. It rewards the discipline of submitting reproducible counter-examples rather than speculative complaints, while still keeping the contest path open to any address that can produce one.

This is why a Dexter withdrawal does not rest on trusting the runtime in isolation. It rests on a committed root, a verifiable proof, a history that cannot be silently rewritten, a window in which any third party can challenge, and a multi-sig vault that posts every finalization to Base.

CHAPTER 22

MONITORING AND RECOVERY

Four monitoring tracks run continuously. Each has a defined detection threshold, an on-call routing rule, and a documented response posture — degraded venue state always becomes visible before it reaches withdrawal pipelines.

OPERATIONAL SIGNAL	DETECTION TARGET	RESPONSE POSTURE
Feed freshness and market posture	Oracle staleness above the per-source freshness budget; cross-source divergence above tolerance; mark drift versus reference exchanges	Affected markets switch to constrained mode (reduced max leverage, widened bands, no new opens) within one block
Root publication and liveness	Time since last commitSettlement, queue depth of pending withdraws, and runtime publication lag	Operations page-out at threshold; guardian pause armed if lag exceeds disputeWindowSec without a publication recovery
Gateway and auth health	Per-tier auth error rate, cross-tier permission leak attempts, signed-order replay attempts, rate-limit saturation	Surface-level circuit breaker per tier; admission-tier failures never cascade to operator or multi-sig surfaces
Treasury and reconciliation state	Operating-wallet vs deep-cold-vault balance drift, fee-sweep accounting variance, insurance-reserve exposure ratio	Daily reconciliation report to the ops multi-sig; any unreconciled drift above $1K blocks the next fee sweep until resolved

#

What the venue watches continuously

Every signal feeds the same on-call pager — a single rotation across the founding team and the engineering lead. There is no quiet hour. Oracle and feed alarms route directly to the trading-services on-call; root and publication alarms route to the runtime on-call with the operations lead in CC; gateway and auth anomalies route to the platform on-call; treasury and reconciliation drift routes to the operations multi-sig signers, because anything that ends in a Base transaction needs the signer who will be asked to co-sign the remediation.

Each signal also feeds the public venue surface. Constrained markets show their state on every order ticket and in the order book header, with the reason (oracle staleness, source divergence, mark drift) labelled. Pending withdrawals show the gating condition (root not yet committed, disputeWindowSec elapsing, manual review, KYC pending) so a funded trader can read why their cashier is paused without contacting support. A degraded venue that still presents itself as normal is treated as a worse failure than the underlying degradation.

Releases follow the same discipline. Runtime and contract changes pass through regression and smoke suites before deploy; contract upgrades pass through an additional security-review window with the 3-of-5 governance multi-sig and a 24-hour proposal-visibility delay. Operational drift — a queue tuning, a feed-source weight change, a permission rotation — is logged in the ops channel with the responsible signer and the rollback path. The protocol is not secured by code alone; it is secured by the discipline with which code is changed, published, and monitored in production.

TEXT

```
signal turns unhealthy
  → market posture tightens or gateway breaker trips
  → affected surfaces label their state for users
  → on-call paged, signer set notified for treasury-class events
  → guardian pause armed if degradation crosses safety threshold
  → recovery waits for fresh data + healthy publication + reconciled treasury
  → only the 3-of-5 governance multi-sig can unpause a guardian halt
```

#

How recovery is supposed to work

Recovery is staged, not flipped. Before any protected surface returns to full authority, four conditions must hold simultaneously: oracle freshness has been continuously inside budget for at least the post-recovery soak window, the runtime has committed a settlement root that the contract accepts as in-history, gateway tier health is green across the public, authenticated, signed-order, and operator surfaces, and the operating-wallet to deep-cold-vault reconciliation matches with zero unresolved drift. Any single missing condition keeps the affected surface in constrained mode.

For funded traders the published SLA is the bound that matters. Withdrawals up to $5K outside the leaderboard podium top three target release within 24 hours of Sumsub-class KYC clearing, and within four hours of the gating condition clearing for users already past first-payout review. Withdrawals above $5K or in the podium top three target the same 24-hour window from the point the operations 2-of-3 multi-sig opens the review ticket — the longer tail is review time, not signing time. After any incident that pauses withdrawals, the unpause is accompanied by a published reconciliation showing the cause, the impacted balance state, and the per-account remediation. No payout that was owed before the incident is reduced because of it; insurance reserves backstop any shortfall produced by the recovery.

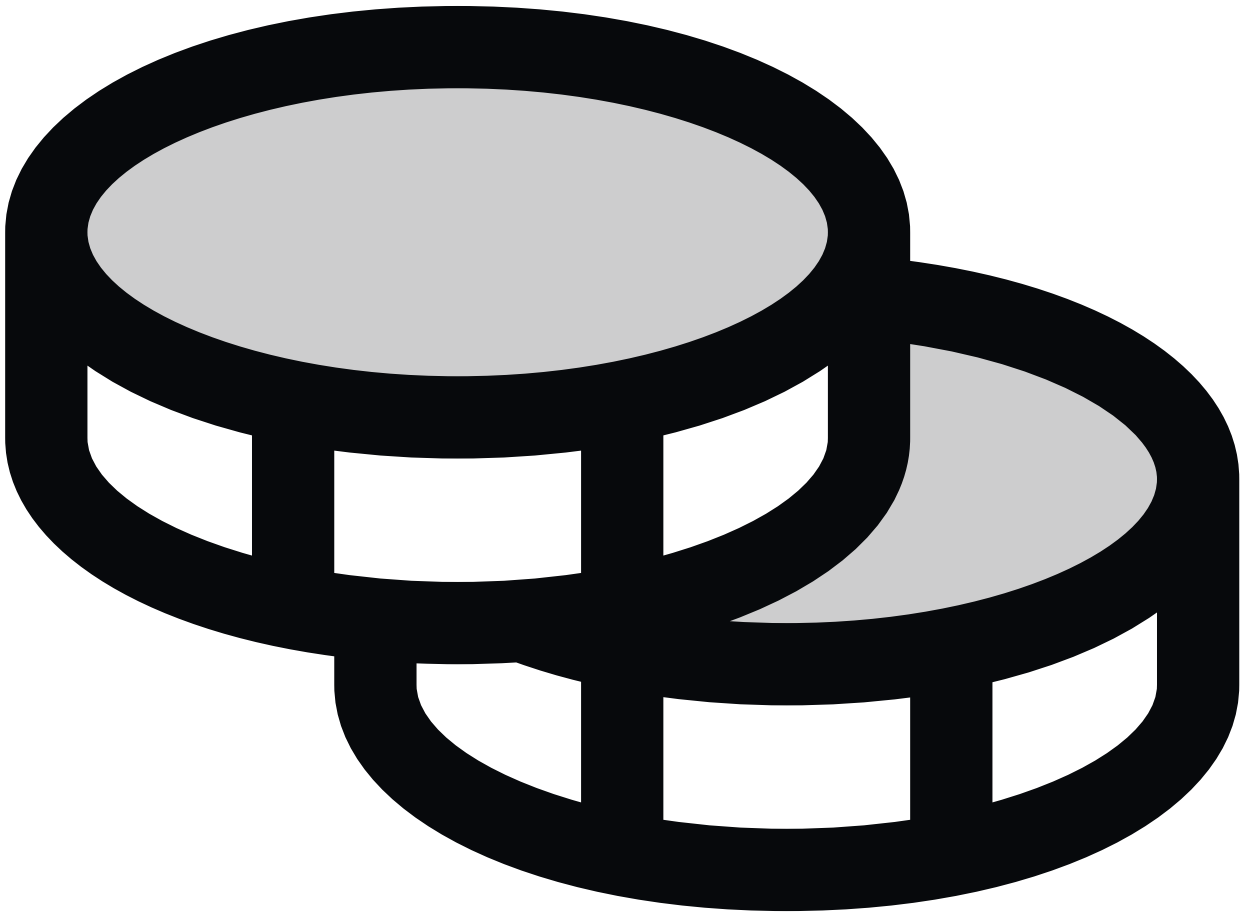
This is the standard Dexter is operating to. Detection in seconds, posture changes in one block, transparency on every degraded surface, recovery only after independent conditions reconverge, and a 24-hour cashier target that the multi-sig will honor or publish the reason it could not.

CHAPTER 23

REVENUE MODEL

The protocol funds itself the same way a prop firm does: charge for evaluation, charge for execution, share the profit with the people who pass. The rest of this page covers each cash line — what it earns, where it lands, and who can move it.

#

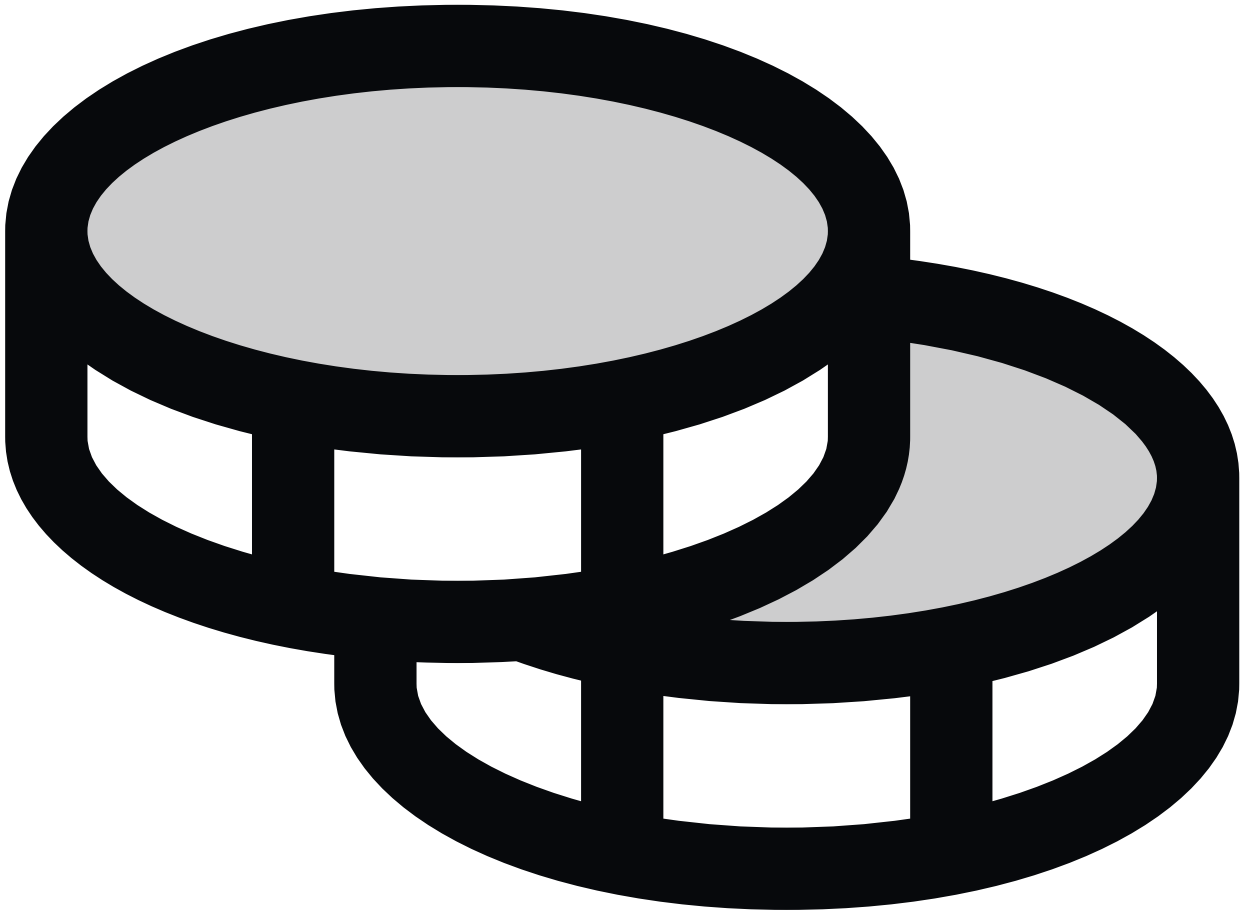


Revenue sources

- **Challenge packs.** $49 Starter, $99 Growth, $199 Swing, $299 Elite. One-time fee, no subscription. The price buys an evaluation window with fixed rules: +10% profit target, -4% daily loss cap, -8% total drawdown. A pack either passes into a funded account or expires.
- **Perp trading fees.** 0.020% maker, 0.060% taker, charged on notional. Depth-adding orders earn a -0.005% rebate. Funded accounts pay the same schedule as anyone else — there is no funded subsidy, so leaderboard PnL stays comparable across cohorts.
- **Funding admin slice.** Funding is a peer-to-peer transfer between longs and shorts; the protocol keeps 1% only on settled funding above $1,000 within a single 8-hour window. Below that threshold the rate is zero. This is the only path funding touches treasury.
- **Liquidation residual.** Maintenance margin remaining after the loss is paid splits 50% to the keeper bounty, 50% to the insurance reserve. It is a defense rail, not a growth line — the reserve grows fastest when the venue is healthy enough that liquidations close cleanly inside margin.

Pack revenue funds the prop layer — challenge engine, oracle redundancy, payout pipeline. Trading fees fund matching, settlement, and the read model. The insurance reserve backstops accounts that liquidate through their margin. No line cross-subsidizes another beyond the published splits.

#



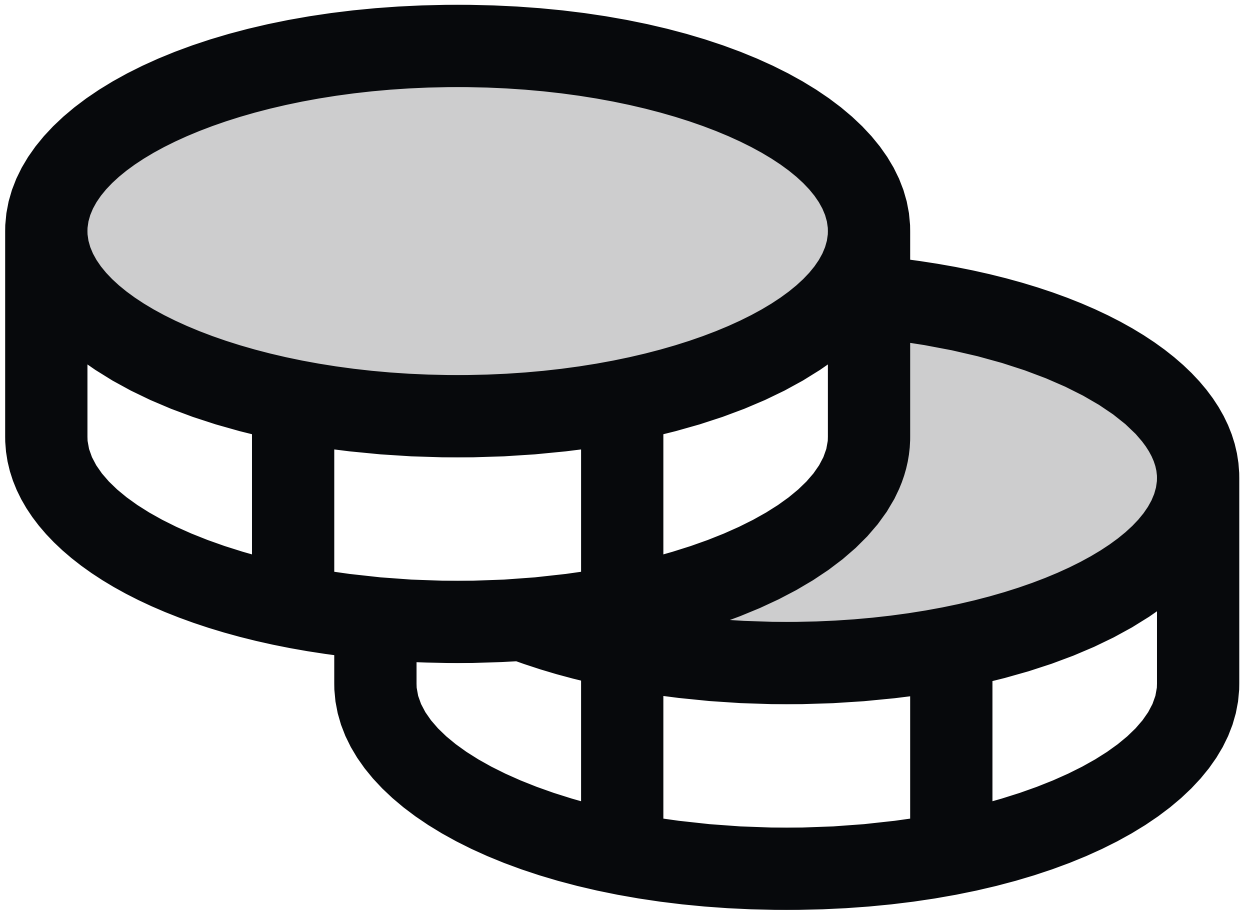
Payout rails

- **Trader profit-share, 90 / 10.** Funded accounts realize PnL with 90% to the trader and 10% to the protocol. Withdrawals are requested in the cashier and settle in USDC on Base, target under 24 hours once Sumsub-class KYC clears. The split is contract-bounded and identical for every funded wallet.
- **Leaderboard prize pool.** Each season carries a fixed pool paid across two boards — Season Rank (skill-weighted) and Points Rank (cumulative XP) — with brackets at #1, #2–3, #4–10, #11–50, and the top 100. The pool is sized at season open from pack revenue and never from trader

profit-share; bracket payouts are published before the season starts and cannot move mid-season.

- **Referral cash, 8% + 4%.** Every paid pack a referee buys pays 8% to the upstream wallet, with a further 4% activity bonus accruing as the referee trades. Payouts batch monthly to KYC-cleared wallets and post as Base transactions the moment they leave treasury — the address, amount, and txid are all public.
- **Insurance reserve.** 50% of every liquidation residual flows into a dedicated contract address, not a treasury EOA. The reserve absorbs the first dollar of an underwater account before any other capital is touched. The balance is queryable on-chain at any block; growth and draws are visible to anyone.

#



Balance separation

Dexter keeps five distinct balance classes — four on-chain in separate contract addresses, one off-chain at the regulated payment processor. They do not commingle and they cannot fund each other off-protocol.

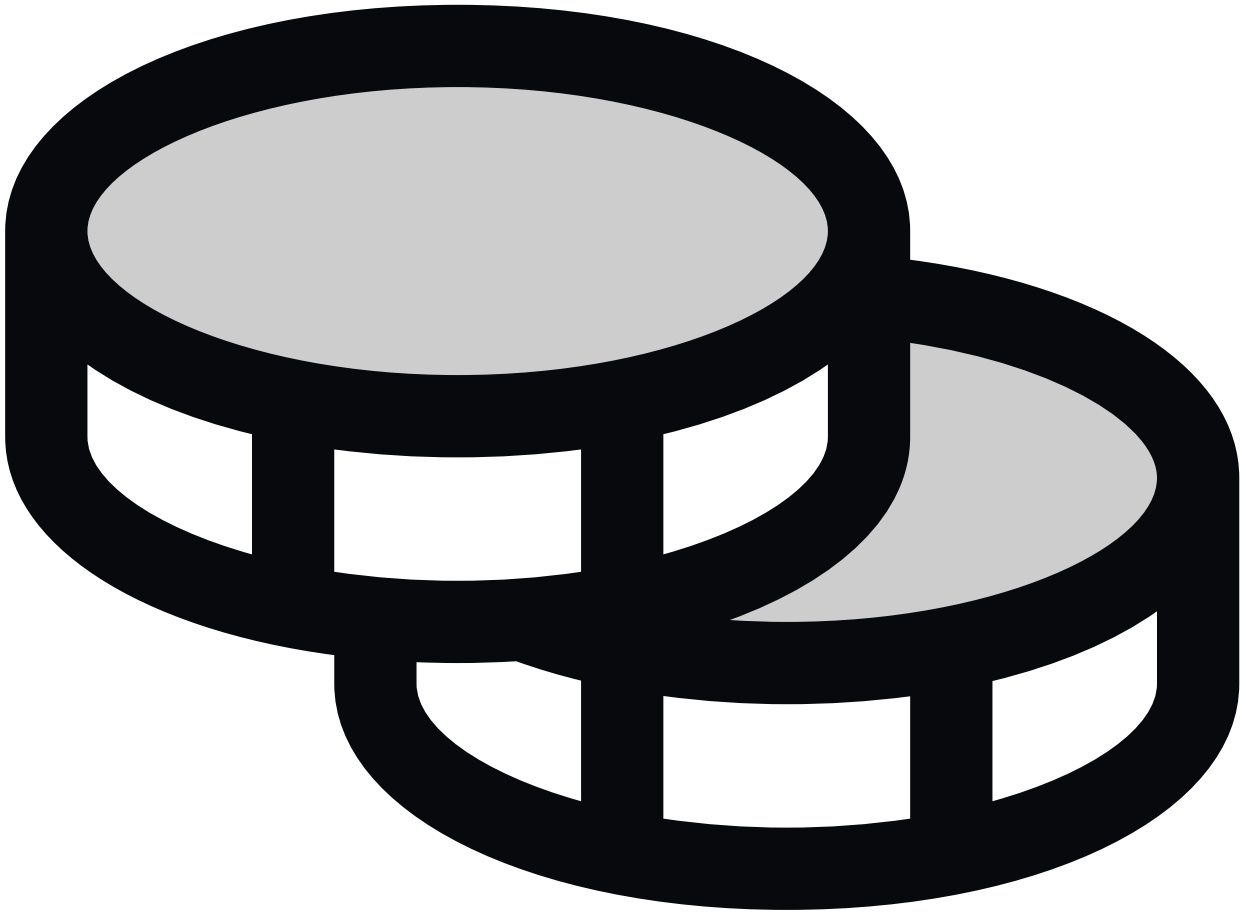
- **User collateral vault.** Holds active trading margin. Withdrawals are paid from this vault and only this vault — operating treasury cannot top it up and cannot draw from it.
- **Insurance reserve.** Funded by the 50% liquidation residual. Drawn only when a liquidation breaches position margin. No discretionary spend; movements out require the 3-of-5

governance multi-sig with an external signer.

- **Treasury operating wallet.** Holds the protocol's share of trading fees plus the 1% funding admin slice. Covers infrastructure, payroll, leaderboard refills, and the on-chain ops budget. Subject to the 2-of-3 ops multi-sig above $10,000.
- **Reward pool.** Pre-funded at season open with the published leaderboard prize pool and the referral float. Drawn down as ranks settle and referrals mature. Intra-season top-ups require a 3-of-5 governance signature so the bracket cannot be quietly inflated mid-season.
- **Fiat ramp, off-chain.** Holds card, Apple Pay, and Google Pay pack purchases for 1–2 days before they convert to USDC and post to the operating wallet. Crypto pack purchases skip this rail entirely.

The rule is single-fault isolation. A KYC dispute on the fiat ramp cannot delay a withdrawal, because withdrawals settle from the collateral vault. An insurance draw never touches operating treasury. A slow trading month does not change the published leaderboard brackets, because the reward pool was sized and funded before the season opened.

#



Treasury controls

Movements above $10,000 from the operating wallet require a 2-of-3 multi-sig. Movements out of the insurance reserve require a 3-of-5 multi-sig with at least one external signer who is not on the Dexter cap table. The season-open reward pool refill rides the same 2-of-3; any intra-season refill switches to the 3-of-5 governance flow. The challenge rules and the funded split — +10% / -4% / -8% and 90 / 10 — are wired into the v1 contract, not into a governable parameter, so no signature combination can rewrite them mid-season.

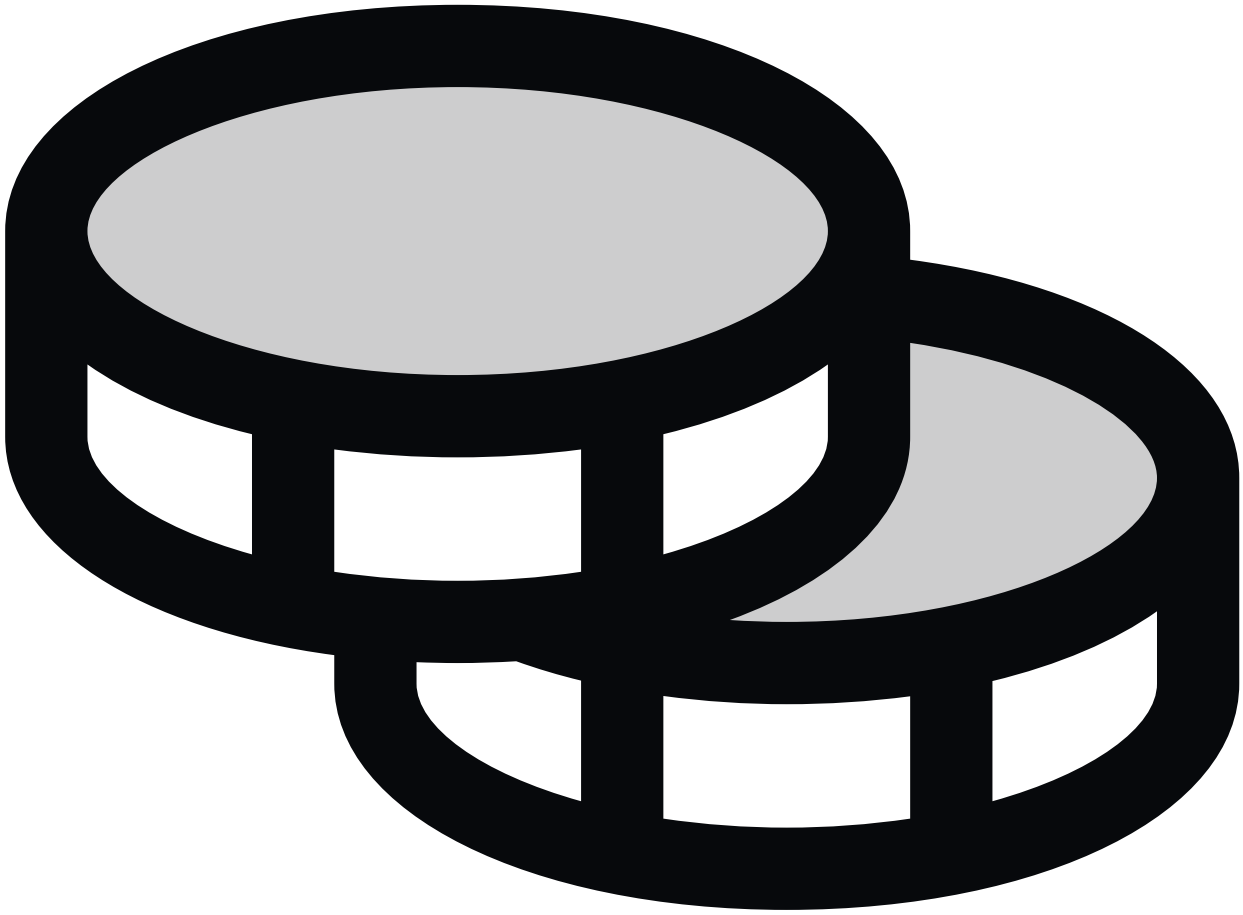
Every multi-sig transaction posts to Base. The contract addresses for each vault are listed on the security page; the season-open and season-close refills are linked from the leaderboard footer. Anyone can verify the cash sitting in each pool at any block, in any explorer, without asking Dexter for a number.

CHAPTER 24

DXTR TOKEN AND PARTICIPATION

Read this page as a token spec, not a marketing deck. Every number — supply, allocation cliff, vesting period, fee tier threshold, quorum percentage — is fixed in the deployment contract and visible on Base. The rest of this whitepaper covers what USDC pays for; this page covers what DXTR governs.

#



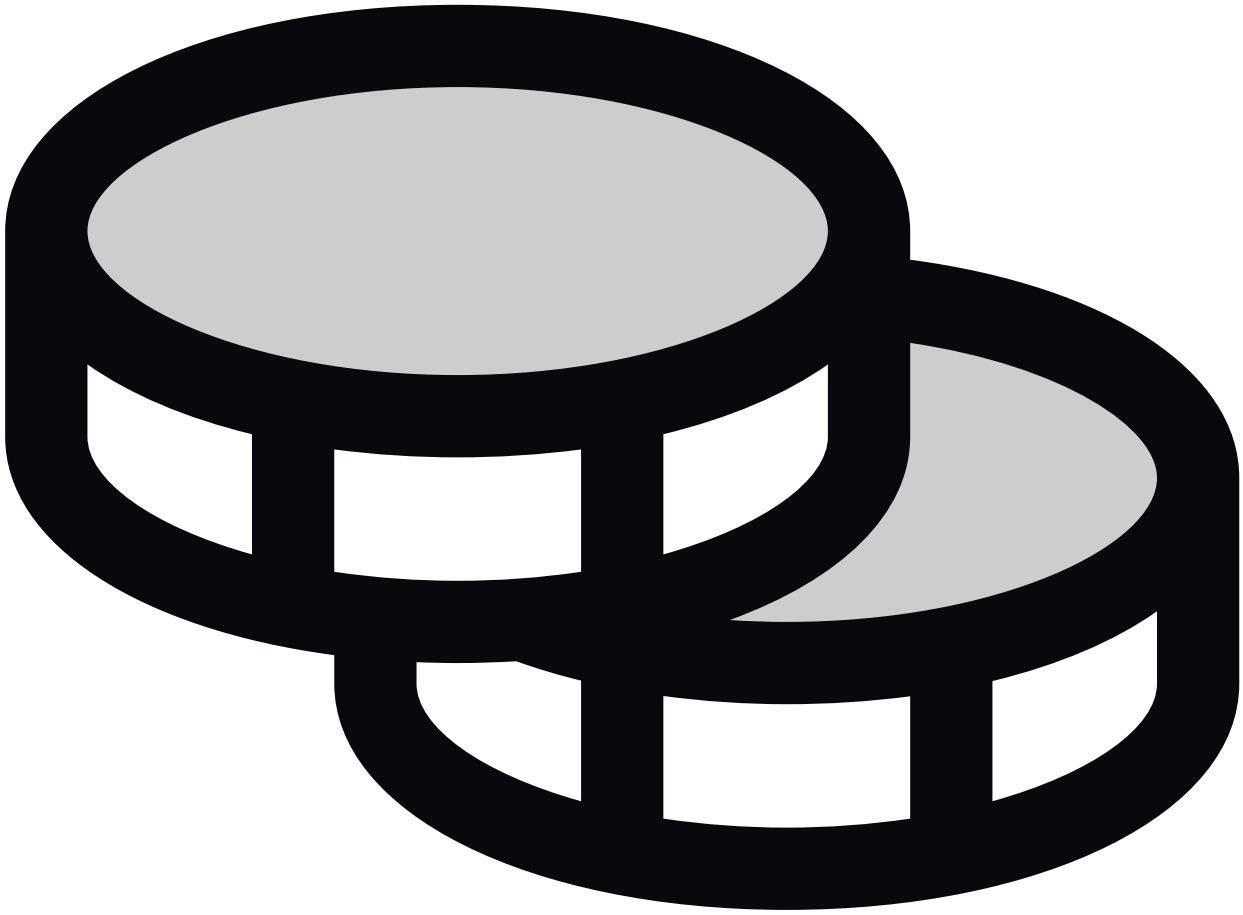
Supply and distribution

Total supply is hard-capped at **1,000,000,000 DXTR**. The contract has no mint function — neither governance nor the team can issue more. Every allocation below is enforced by an on-chain vesting contract whose address is published on the security page.

- **40% — Airdrop record.** Distributed to wallets that traded volume, finished a season ranked, or earned referral cash in the pre-token windows. The snapshot weights skill-adjusted PnL above raw points so the allocation lands on traders who actually used the venue, not on point-farming bots.

- **22% — Liquidity and market making.** Held by a market-making contract backstopping DXTR/USDC depth at launch. Refills are policy-bound to published depth targets, not discretionary.
- **20% — Team.** 12-month cliff, 36-month linear vest. Wallets are public, vesting transactions emit on-chain, and no team unlock can be accelerated by governance.
- **10% — Treasury.** Long-horizon protocol runway. Movements gated by the 3-of-5 governance multi-sig with an external signer — the same gate used for the insurance reserve.
- **5% — Ecosystem and grants.** Allocated to third-party integrations: dashboards, copy-trading rails, analytics. Granted under a published agreement; no private discount round, no off-record deals.
- **3% — Advisors and partners.** Same 12-month cliff, 36-month linear vest as the team. Same on-chain transparency.

#

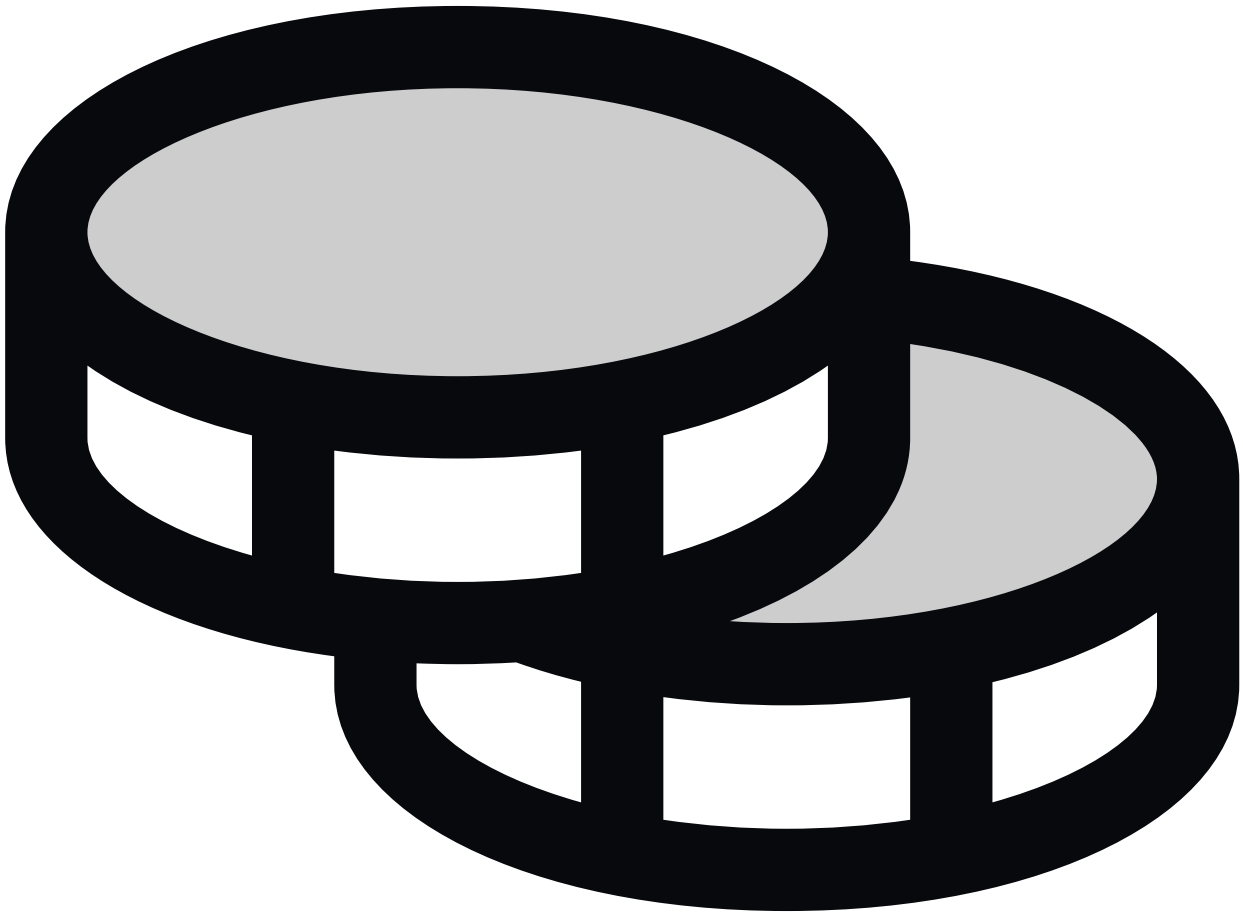


Utility and discounts

- **Pack discounts.** Wallets staking DXTR at purchase time receive 10% off the $99 Growth pack, 15% off the $199 Swing pack, and 25% off the $299 Elite pack. The Starter pack is full price for everyone — entry stays cheap. Discounts apply at the cashier as a price reduction, not as a deferred rebate.
- **Trading fee tier.** Staked DXTR steps the wallet down the taker schedule: -10% at 1,000 staked, -20% at 5,000, -30% at 25,000. Maker fees and the funding admin slice are not discounted. Tier is recomputed nightly off the staked balance; there is no manual claim.

- **Points Rank boost.** Stake multiplies XP accrual on the Points Rank board up to 1.5x at the top band. The skill-weighted Season Rank ignores stake entirely — capital cannot buy a podium finish, only an XP grind.
- **Referral acceleration.** Stakers unlock the 4% activity bonus on referee volume at 30 days of referee retention, half the default 60-day vest. The 8% base rate is the same for every wallet, staked or not.

#

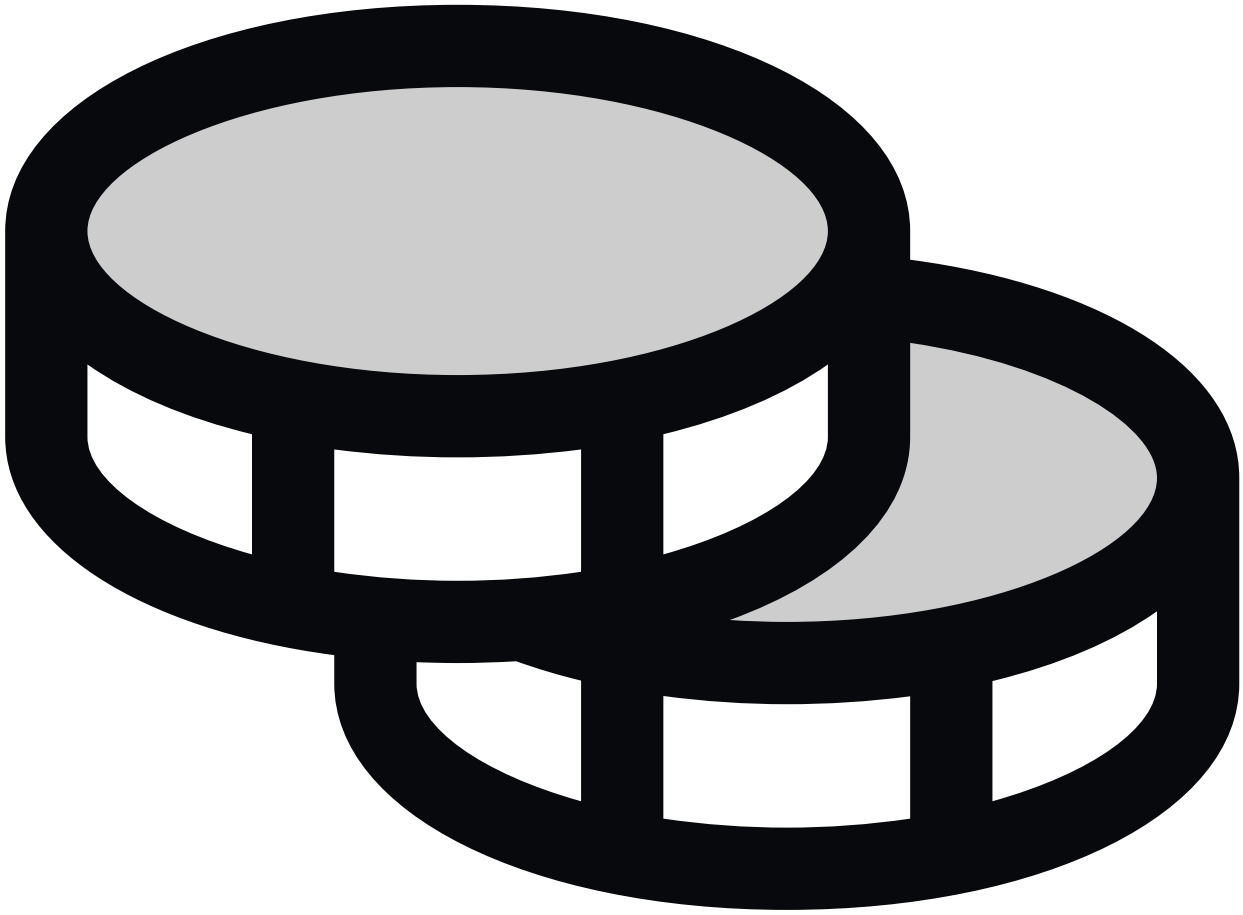


Governance

Holders vote on three parameter classes: the fee schedule (maker, taker, funding admin), the leaderboard prize pool size, and the next markets added to the listing queue. Three things are explicitly outside the vote — user collateral, individual payouts, and the challenge rules. The +10% / -4% daily / -8% total thresholds and the 90 / 10 funded split are wired into the v1 contract so no governance proposal can disqualify a trader's pass criteria after they paid for the pack.

Voting weight tracks staked DXTR with a 14-day stake-to-vote cooldown — a deliberate defense against flash-loaned governance attacks where a borrowed bag becomes a one-sided vote and unwinds the next block. Quorum is 4% of circulating supply; passage requires >50% of votes cast. Both are contract parameters, not adjustable from inside a proposal.

#



What the token is not

DXTR is not a claim on user collateral. It is not a wrapper for the insurance reserve. It is not a yield-bearing instrument — staking earns discounts, multipliers, and votes, never an advertised APR. It is not required to use the venue: a wallet that never holds a single token can still buy any pack, run the challenge, fund up, rank on either board, and collect every USDC payout in this whitepaper. There is no permissioned tier locked behind the token.

Value accrues to DXTR from holding governance and discount rights on a protocol with real cash flow — pack revenue, trading fees, funding admin slice. If those lines shrink, the rights shrink with them. There is no inflation schedule, no emissions program, and no buyback promise dressing that up. The supply is fixed, and so is the math.

CHAPTER 25

CHALLENGE AND AIRDROP

Dexter is the first on-chain prop trading firm. Buy a challenge pack between **$49 and $299**, return **+10%** in 30 days without breaching the **-4% daily** or **-8% total** drawdown, and graduate to a real funded account that pays **90 / 10 in your favor**. Every dollar of profit-share, leaderboard reward, and referral cash settles as **USDC on Base**, directly to the wallet that signed in.

#

How the program works

The full journey is four steps: connect a wallet, pick a pack, trade against published rules, withdraw the split. Four pack tiers map one-to-one to four funded allocations, each available in 1-Step and 2-Step formats with identical economics.

- **Starter** — $49 pack, $5,000 funded allocation.
- **Trader** — $99 pack, $10,000 funded allocation.
- **Pro** — $199 pack, $25,000 funded allocation. Most popular tier.
- **Elite** — $299 pack, $50,000 funded allocation.

The pass criteria are the same at every tier and never change mid-attempt: hit **+10% profit on the funded equity** inside the **30-day evaluation window**, keep close-to-close equity above the **-4% daily drawdown** floor, and never drop more than **-8% from the high-water mark**. Trader must also meet minimum active days and minimum trade count so the result reflects discipline rather than one lucky candle. Pass cleanly → funded account opens, profit-share runs **90 / 10** in your favor for the life of the account, and withdrawals settle on Base.

A free **14-day demo** with $25,000 of simulated equity runs the exact same rule engine. No deposit, no KYC, no payout — a sandbox to feel the limits before real money lands.

#

Quest progression and tiers

Behind the pack flow, every wallet runs a six-step quest worth **1,500 XP** in total — Enlist (+50), Loadout (+100), First Blood (+150), Bounty (+250), Colors (+350), Spoils (+600). The XP never burns down: it stacks across seasons and unlocks seven persistent tiers from *Initiate* at 0 XP through *Apprentice* (250), *Operator* (1,000), *Tactician* (3,000), *Strategist* (8,000), *Architect* (20,000), up to *Sovereign* at 50,000.

12 badges across four rarities (Common, Rare, Epic, Legendary) and **4 loot vaults** (Bronze, Silver, Gold, Diamond) trigger at specific XP milestones, releasing cash bonuses and fee credits on top of the regular payout split. Progression is shown live in the Challenge page so a trader always knows what unlocks at the next step.

#

Leaderboard and referrals

Every funded trader competes on a live **season leaderboard** with two parallel rankings — *Season Rank*, a weighted score (performance × risk-adjusted return × normalized bps × compliance) that favors clean equity curves over variance; and *Points Rank*, a cumulative sum of every XP source. The top of either board collects a slice of the season reward pool on top of profit-share. Scoring detail and bracket breakdown live on the Scoring page.

The **referral system** pays up to **12% of each pack fee** when an invited wallet buys and trades — 8% base on activity, 4% activity bonus on pass. Caps run **$5.88 / $11.88 / $23.88 / $35.88** per Starter / Growth / Swing / Elite referral, and every dollar is recorded on a public on-chain ledger with payment ID and payout hash. Referee receives a one-time XP bonus (+50 / +100 / +250 / +500) so both sides win.

#

What the next pages cover

The next three sub-chapters each take one surface and drill into it:

- **Entry and access** — wallet connect, the free 14-day demo, paid pack mechanics, drawdown rules in detail, and the KYC threshold that fires at first payout.
- **Scoring and airdrop** — exact formula for Season Rank and Points Rank, the daily streak multiplier, the bracket breakdown of the season pool, and how the top-3 podium review works.
- **Referrals** — two-stage cash structure, per-pack caps, the on-chain reward ledger, abuse controls, and quick math on what 100 referrals are worth.

Together they explain how Dexter pays real money to real traders without leaking the cash pool to bots, sock-puppets, or paid clicks.

CHAPTER 26

ENTRY AND ACCESS

Getting into the firm takes four steps and roughly five minutes: connect a wallet, pick a path (free demo or paid pack), trade against the published rules, clear identity review the first time you pull a real-money payout. Nothing in this flow asks for an email, a password, or a document you didn't choose to provide.

#

Wallet connect and account

Sign-up runs at app.dexter.market/dex/airdrop. A wallet signature is the entire login — MetaMask, WalletConnect, Coinbase Wallet, and Reown (formerly Web3Modal) are all supported. The same address that signs in anchors your trade history, challenge progress, referral code, and payout destination.

First connect signs a non-transactional EIP-712 message so the relay can mint an account row keyed to the wallet. No funds move, no token allowance is approved, and KYC is not requested. From that point the address can hold a demo balance, a paid pack, accumulate XP, and show on

the leaderboard without uploading a single document.

#

Free 14-day demo

Every connected wallet can start a free demo seat: **$25,000 of simulated equity**, the same live order book, the same mark prices, and a **14-day clock**. The pass criteria — +10% profit, -4% daily drawdown, -8% total — are identical to a paid pack. The point is to let a trader feel the limits with zero financial exposure before staking pack money.

- Demo runs against the production matching engine and oracle feed. Slippage, funding, and fee deductions are simulated at parity, so the equity curve you produce in demo is the curve you would have produced live.
- Quest XP and badges earned during demo are real and carry forward. A wallet that hits First Blood in demo keeps the +150 XP when it converts to a paid pack — the leaderboard does not reset.
- Demo PnL is **not** payout-eligible and demo seats cannot accumulate referral cash. Those rails require a paid pack on the upstream wallet. Demo is the sandbox; the cashier opens at the paid tier.
- If the 14-day clock expires without a target hit, the seat closes. Each wallet gets one demo reset per season; after that, the path forward is a paid pack.

#

Paid challenge packs

Paid packs sell in four tiers and can be paid in USDC on Base or in BTC/ETH at the live oracle quote. Pack price is the one-time cashier cost; the funded allocation is the equity a trader actually moves once the challenge is passed. The pricing ladder is fixed and visible at the Challenge page:

- **Starter — $49** → $5,000 funded allocation.
- **Growth — $99** → $10,000 funded allocation.
- **Swing — $199** → $25,000 funded allocation.
- **Elite — $299** → $50,000 funded allocation.

Each pack purchase settles in a single Base transaction. The relay records the payment ID, pack tier, wallet, and timestamp; the transaction hash is the public proof of entry — anyone can verify it on Basescan. Exactly one active attempt per wallet at a time. If a wallet breaches a rule, the attempt closes and a re-buy at the same or a different tier starts a fresh 30-day evaluation window with a clean drawdown clock.

#

KYC and first payout

KYC sits at the cashier, not the door. A wallet can trade, win, lose, climb the leaderboard, and accumulate referral cash with no identity check. Verification only fires the first time a wallet requests a real-money payout — challenge profit-share, leaderboard cash, or aggregated referral cash — once the cumulative payout exceeds the **$50 threshold**.

- The flow is Sumsub-class: government ID, selfie liveness, and proof-of-address only for high-volume payouts. Median completion runs under five minutes on the first attempt; payouts settle within a 24-hour target once review clears.

- Sanctions screening runs against **OFAC, EU, and UK** lists. Connections originating from blocked jurisdictions — currently **Iran, North Korea, Syria, and Cuba** — are rejected at the cashier; trading is not pre-screened, but no withdrawal will clear.
- One pass clears a wallet for every subsequent payout in the same season, including referral cash and leaderboard cash. Re-verification only triggers if a payout instruction targets a new destination address.
- KYC runs through a third-party provider. Dexter stores only the pass/fail status, the provider's verification ID, and the geographic risk tag — no ID images, no biometric data.

#

Drawdown rules and what comes next

Once the pack activates, three numbers govern the **30-day evaluation window** and nothing else: **+10% profit target** on the funded equity, **-4% daily drawdown** measured close-to-close as a hard floor, and **-8% total drawdown** from the running high-water mark. Touch either drawdown line and the attempt closes — no warnings, no soft resets, no negotiated extensions. The intent is the same as a real prop desk: discipline first, return second.

Pass cleanly and the funded account opens with the **90 / 10 split** in your favor for the life of the account. From there, the Scoring page covers how Season Rank and Points Rank carve up the leaderboard pool, and the Referrals page covers how the cash flows when your link drives another paid pack.

CHAPTER 27

SCORING, AIRDROP, AND REWARDS

Every funded trader competes on a live, on-chain seasonal leaderboard. Seasons run **30–45 days**, and the top of the board collects a slice of the season reward pool on top of the standard **90 / 10** profit-share. Two parallel rankings expose two different answers to the question "who is the best trader this season" — one weighted for discipline, one cumulative for engagement.

#

Season Rank — weighted score

The headline ranking. A single weighted score blends four inputs so raw account size doesn't automatically dominate the board:

- **Performance points** — realized PnL normalized to the funded tier. A $1,000 gain on a Starter ($5K allocation) carries more weight than a $1,000 gain on Elite ($50K), because percentage return — not dollar return — is what the firm pays for.
- **Risk-adjusted return** — return divided by realized peak-to-trough drawdown. A smooth equity curve outranks a volatile one with the same end PnL.

- **Normalized return in basis points** — scaled to a common reference so 1-Step and 2-Step traders compete on the same axis without one format gaming the other.
- **Compliance multiplier** — a clean book (no rule breaches, no warnings, no anti-pattern flags) holds the multiplier at 1.0. Each warning taxes it down; severe flags can zero the score.

The intent: a passable but messy run loses to a slightly smaller but disciplined run. That mirrors how a real prop desk pays — for risk-adjusted consistency, not for variance.

#

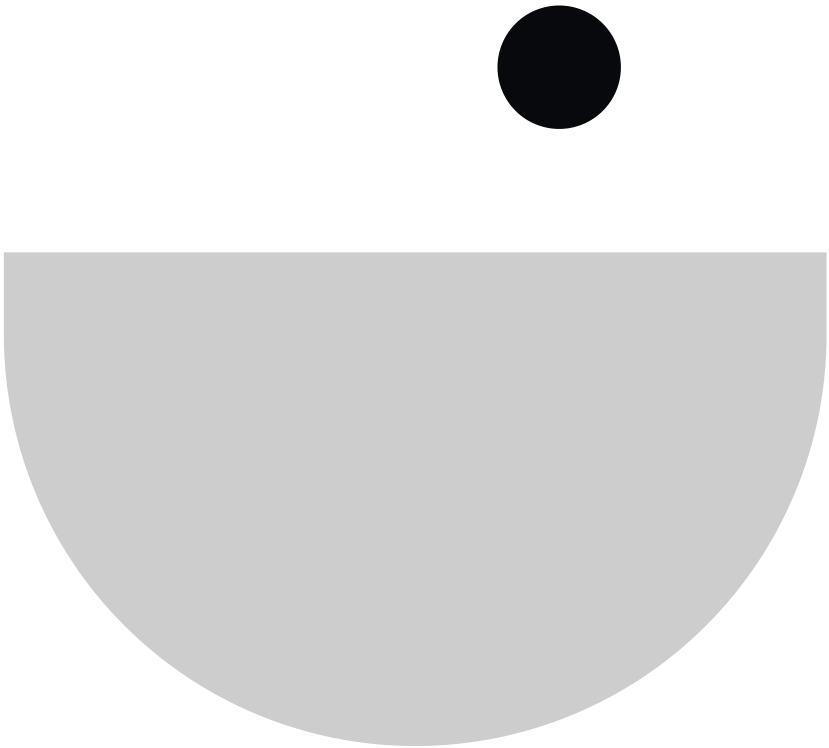
Points Rank — cumulative XP

The parallel ranking. A straight cumulative sum of every Dexter Point a wallet earns across the season:

- **Performance points** from realized PnL on the funded account.
- **Quest XP** — Enlist (+50), Loadout (+100), First Blood (+150), Bounty (+250), Colors (+350), Spoils (+600) — plus the 12 badge unlocks across four rarities and daily streak rewards.
- **Referral XP** — the +50 / +100 / +250 / +500 bonus earned per referee that signs up to Starter / Growth / Swing / Elite under your code.

This board rewards traders who compound wins through volume and engagement rather than a single big-print return. Streak alone almost never tops it; combined with a passable equity curve it routinely carries a trader into the top decile.

#



Live updates and on-chain proof

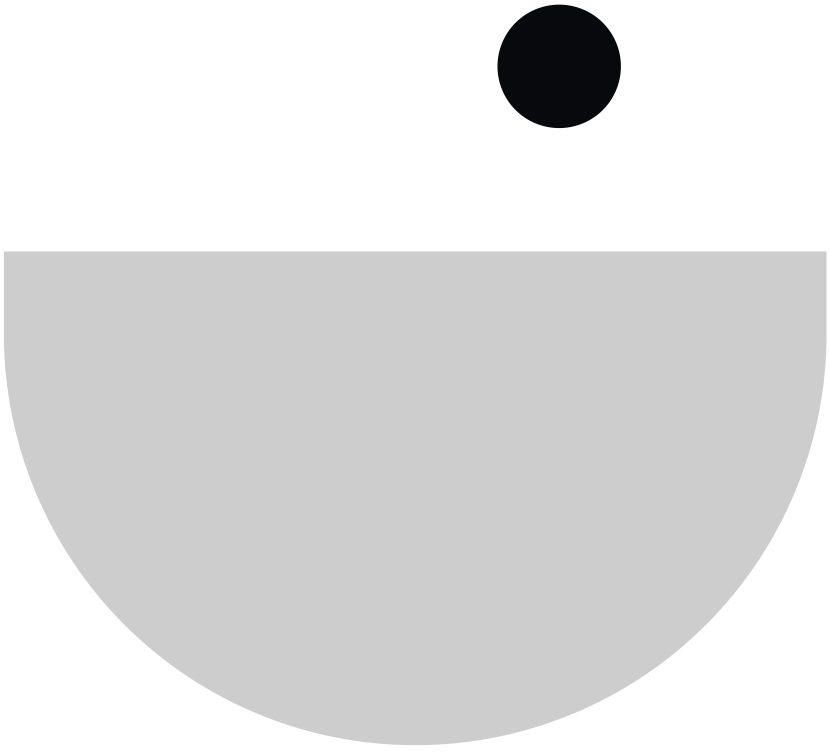
Both ranks update near-real-time as fills, settlements, and payouts publish. Your **leaderboard** row shows current rank, distance to the next slot, and the projected reward share for your current bracket — all recomputed live.

The **reward pool** — funded by a slice of every pack fee plus a portion of net trading fees — is published live on the leaderboard with entry count and currency (USDC on Base). Beside it sits the **Verified Rewards** counter: a running on-chain total of every payment hash already settled to a

winner, not a marketing figure. Past-season winners are linked to Basescan with the exact payout hash so anyone can independently verify the cash moved.

When the season closes, the **top-3 podium** goes through a final risk review — position-size hygiene, rule-edge cases, copy-trading detection. The review window is short (typically a few hours) and the season pool releases automatically once it clears, with settlement targeted inside 24 hours of pass.

#



Daily streak multiplier

Logging in and placing at least one fill on consecutive UTC days banks streak XP and feeds the Points Rank multiplier. The bonus is small in the first few days and meaningful from day 7 onward, where it also unlocks the "Marathon" rare badge.

Miss a single UTC day and the streak resets to zero — no make-up days, no streak insurance. The mechanic is deliberate: prop trading rewards showing up, not heroic comebacks. The same reset rule applies to the streak's impact on Season Rank's compliance multiplier, so a broken streak only costs XP, never your weighted score.

#

How the season pool pays out

The pool is split across the top of the leaderboard in declining brackets. Indicative shape (per-season figures are published in the leaderboard's *season info* drawer before the season opens):

- **Rank 1** — the single largest slice; typically several months' worth of an Elite-tier passing trader's profit-share.
- **Rank 2–3** — share a sizeable secondary tier on top of standard 90 / 10 profit-share.
- **Rank 4–10** — receive smaller bracketed payouts that still cover several pack-fee multiples for each trader.

- **Rank 11–50** — symbolic cash plus the permanent on-chain "Leaderboard Climber" badge.
- **Top 100** — fee-credit airdrop usable against the next season's pack purchase.

Once the season closes and the podium review clears, payouts settle as USDC on Base directly to the wallet that earned them — typically within minutes of release, inside a published **24-hour target**. Every cent is recorded as a payment hash on the public ledger, the same one the Referrals page uses.

CHAPTER 28

REFERRALS AND BONUS CREDITS

Every Dexter account ships with a unique referral code generated at first wallet connect. Sharing the code pays cash and XP whenever an invited wallet buys a paid challenge pack — settled as **USDC on Base** directly to your wallet, recorded on a public on-chain ledger. The reward unlocks in two stages, scales with pack size, and tops out at **12% of the pack fee**. Inviting an Elite trader is worth roughly 6× more than inviting a Starter; pack scale is the only knob that matters.

#

Cash structure — up to 12% per pack

The reward releases in two parts so the cash always traces a real, engaged referee — never a wallet that bought and walked:

- **Base — 8% of the pack fee.** Releases when the referee clears the minimum active days and the minimum trade count required by their challenge. The floor is real activity: the invitee has to actually trade for the base to unlock.
- **Activity bonus — 4% of the pack fee.** Adds when the referee passes the challenge cleanly, or trades 2× the minimum compliance volume on a clean record. The bonus rewards inviting

traders who stick around past the first week.

The two halves stack to a maximum of 12% of the upfront pack price. Both halves pay in USDC on Base regardless of how the referee originally funded the pack (USDC, BTC, or ETH).

#

Per-pack maximum cash

Caps map directly to the four pack tiers (base + bonus, paid USDC on Base):

- **Starter** — $49 pack → up to **$5.88** to you ($3.92 base + $1.96 bonus).

- **Trader** — $99 pack → up to **$11.88** ($7.92 base + $3.96 bonus).
- **Pro** — $199 pack → up to **$23.88** ($15.92 base + $7.96 bonus).
- **Elite** — $299 pack → up to **$35.88** ($23.92 base + $11.96 bonus).

Both 1-Step and 2-Step formats of each pack carry identical referral economics. If a referee fails the challenge and re-buys, the upstream wallet earns again on the second pack — every paid attempt is its own qualifying event.

#

Referee XP bonus

Every wallet that signs up with your code receives a one-time XP boost on its first pack purchase. The bonus feeds the Points Rank leaderboard and does not subtract from your cash share — both sides win when the code is used.

- Starter — **+50 XP** to the new trader.
- Trader — **+100 XP**.
- Pro — **+250 XP**.
- Elite — **+500 XP**.

#

On-chain reward ledger

Every referral reward writes to a public ledger with payment ID, payment transaction hash, and — once released — payout transaction hash. Status transitions (*qualified → payable → paid*) and any voided reasons are visible to anyone with the link.

Each referee row shows: the referee wallet (shortened), the pack tier they bought, the unlock state (waiting on activity / waiting on challenge pass / paid), the cash unlocked so far, and the season window. There is no private dashboard layer — what the upstream wallet sees in the Referrals page is exactly what the ledger holds.

Payouts aggregate continuously. The moment qualifying conditions clear, the reward moves into the payable bucket, the contract releases the USDC, and the cash arrives in the wallet — typically within minutes of qualification, inside the **24-hour target**.

#

Risk review and abuse protection

The pool is protected by a tight set of automated checks. Honest traders never see them fire; the controls exist so the cash reflects real product growth, not synthetic farming:

- **Self-referral** — matching wallet, IP, or KYC document on both sides triggers an automatic void with the reason logged on the ledger.
- **Multi-account farms** — coordinated sign-ups from one cluster of devices or one funding source are frozen pending manual review and released only with a clean trade record.
- **Bot-driven sign-ups** — wallets that never produce real fills, close positions immediately, or fail Sybil heuristics are voided. Pack fee on a failed referral is not refunded to the buyer; the referral reward simply does not unlock.

Voided rewards stay visible on the ledger with the void reason in plain text — the system is permissioned but transparent. Sanctions screening on the referee side (OFAC / EU / UK lists, plus Iran / North Korea / Syria / Cuba jurisdiction blocks) fires at the cashier; rewards from a blocked referee never reach *payable*.

#

Quick math: 100 referrals

Max-payout case — every referee clears the activity floor and passes the challenge cleanly:

- 100 × Starter referrals → up to **$588**
- 100 × Trader referrals → up to **$1,188**
- 100 × Pro referrals → up to **$2,388**
- 100 × Elite referrals → up to **$3,588**

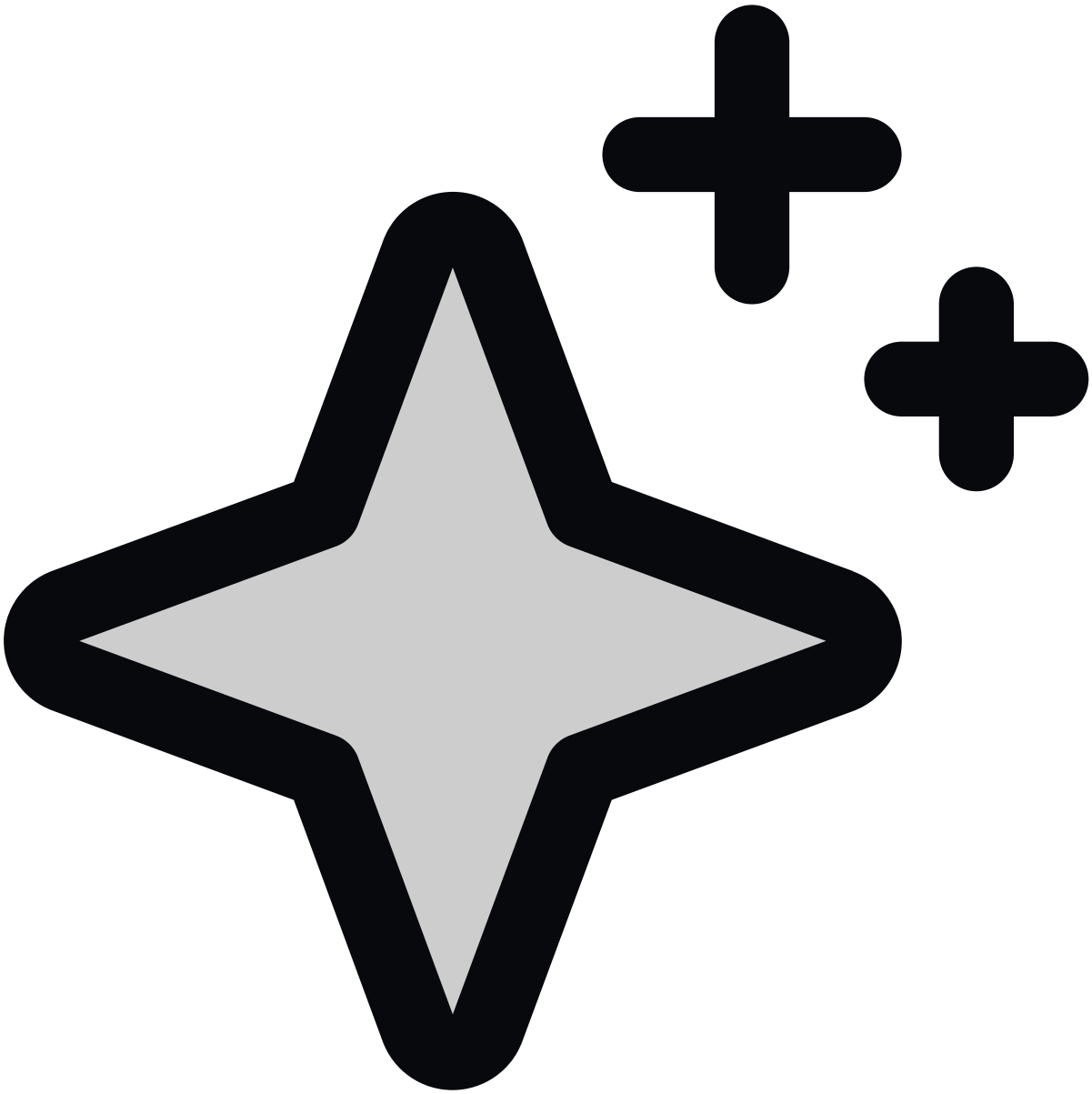
Realistic mix — assume half clear activity (base only) and a third also pass (base + bonus): a 100-Pro distribution still tails out around **$1,400**, on top of any pack profit-share the referrer earns from their own trading. The constraint on this number is reach, not pool depth. The cap is your audience, not Dexter.

CHAPTER 29

WHAT COMES NEXT

The next 90 days are fixed scope. Beyond that, items are tracked publicly but subject to governance votes once DXTR is live. This page is a snapshot; the live roadmap mirrors to the in-app Releases page and to the Dexter X account, with sprint-end notes published every four weeks.

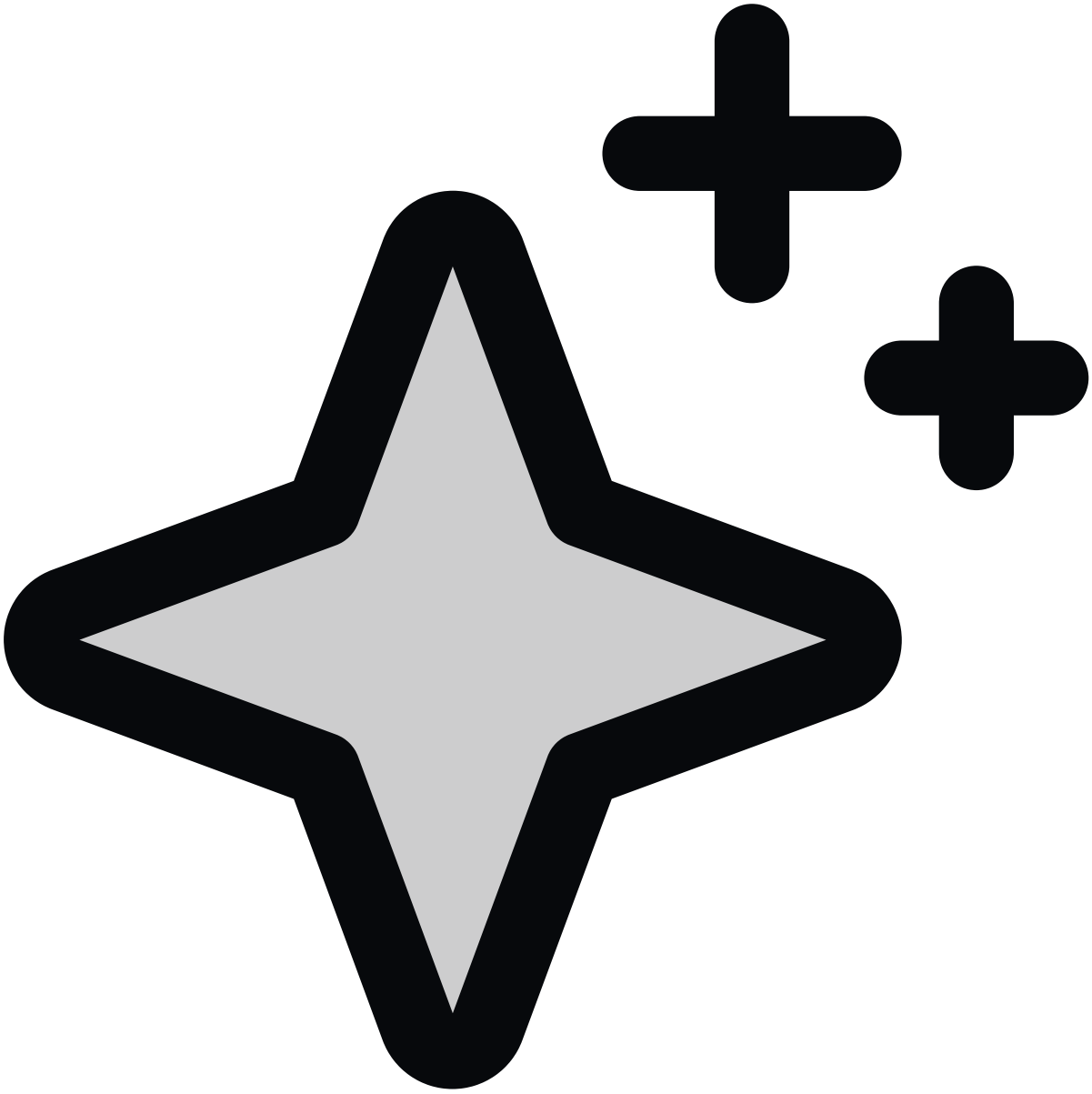
#



Live now

- **Challenge season 1.** Four packs ($49 / $99 / $199 / $299), $5K–$50K funded allocations, +10% / -4% / -8% rule set, 90 / 10 profit split, USDC-on-Base payouts. ~600 paid wallets and 12,000+ demo wallets active in the season.
- **Public season leaderboard.** Season Rank (weighted score) and Points Rank tables, ranked cash brackets at #1 / #2–3 / #4–10 / #11–50 / top 100, daily streak multiplier, "Verified Rewards" counter reconciled hourly against Basescan.
- **Referral cash rail.** 8% base unlocks on referee min-activity, +4% activity bonus on pass. Payment ID and Basescan tx hash per referral, monthly batch settlements to KYC-cleared upstream wallets, public void ledger for fraud cases.
- **Quest and tier system.** Six main quests (Enlist → Spoils, 1,500 XP path), 12 badges across four rarities, four loot vaults (Bronze / Silver / Gold / Diamond), seven prop-trader tiers (Initiate → Sovereign). XP persists between demo and paid.
- **BTC, ETH, SOL perpetuals.** Cross-margin, Pyth-backed oracle marks with Chainlink / RedStone / Uniswap V3 TWAP cross-check, maker rebate schedule, insurance-reserve-backed liquidations.
- **Compliance gates.** Jurisdiction blocks (IR / KP / SY / CU) and OFAC / EU / UK sanctions lists enforced at the cashier. One-time KYC at first real payout; demo never triggers it.

#

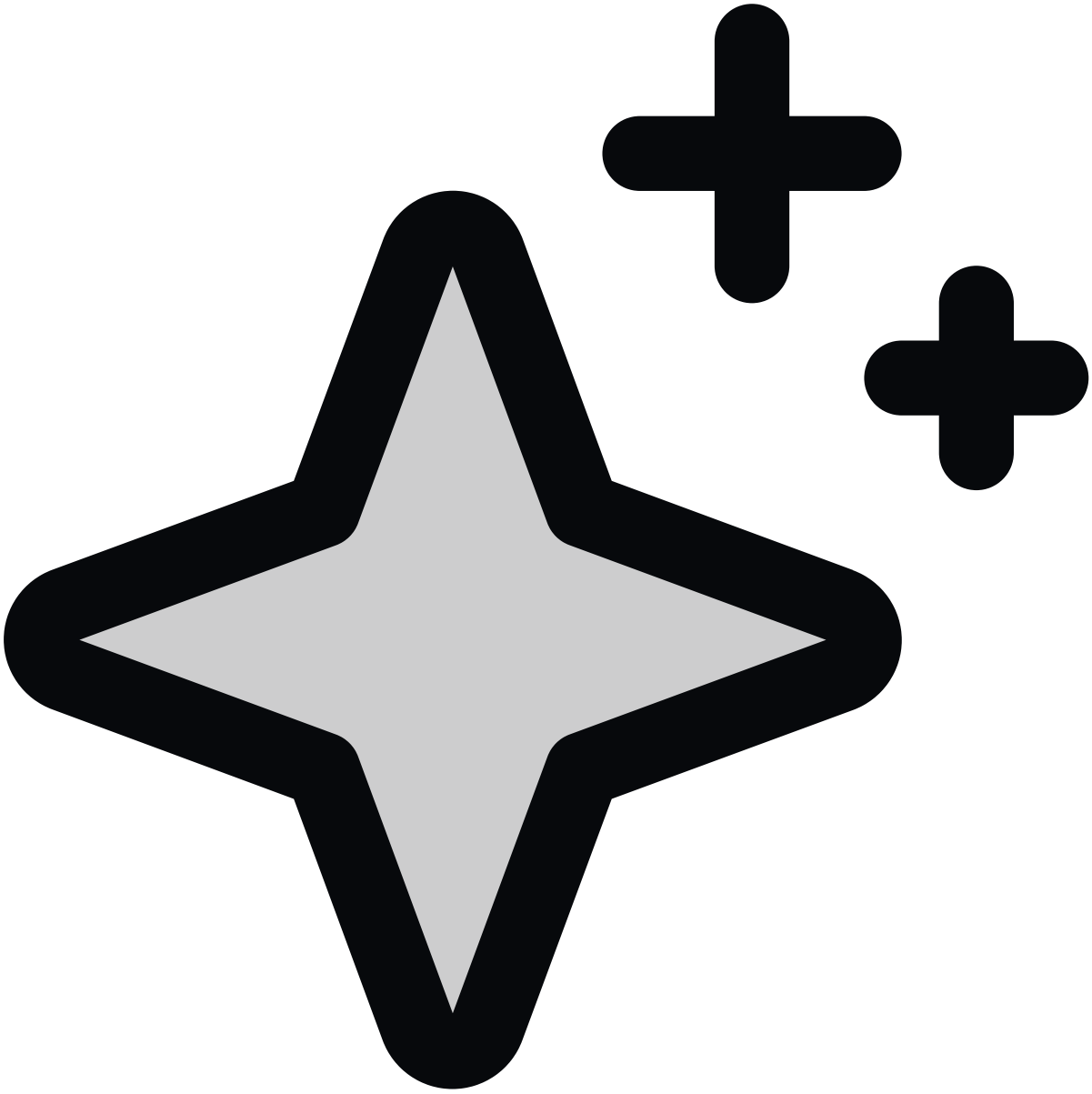


In flight this quarter

- **DXTR token launch.** Fixed 1B supply, 40% reserved for the airdrop record (weighted by skill PnL plus referred pack volume), airdrop snapshot at the end of season 2. Vesting contracts published, market-maker depth committed before TGE.
- **Mobile clients — iOS and Android.** Native trading apps with the same charting, ticket, and challenge HUD as the web client. Android APK is live on the Learn footer; iOS App Store submission queued.

- **Copy-trading on funded accounts.** Subscribers pay a per-trade fee that splits 50/50 between the lead trader and the protocol. Lead must hold a currently-passed challenge — no demo-only leads. Subscribers see lead's drawdown and rule status in real time.
- **Tradable equity perps with sessions.** AAPL / TSLA / NVDA-style perps with explicit session-closed and reduced-liquidity states surfaced in the trade ticket. Listings require governance approval after DXTR launch.
- **Public oracle dashboard.** Per-asset source breakdown (Pyth primary, Chainlink / RedStone / Uniswap V3 TWAP cross-check), fallback ladder, mark-vs-index spread, degraded-state log. Lets third parties build independent surveillance.

#

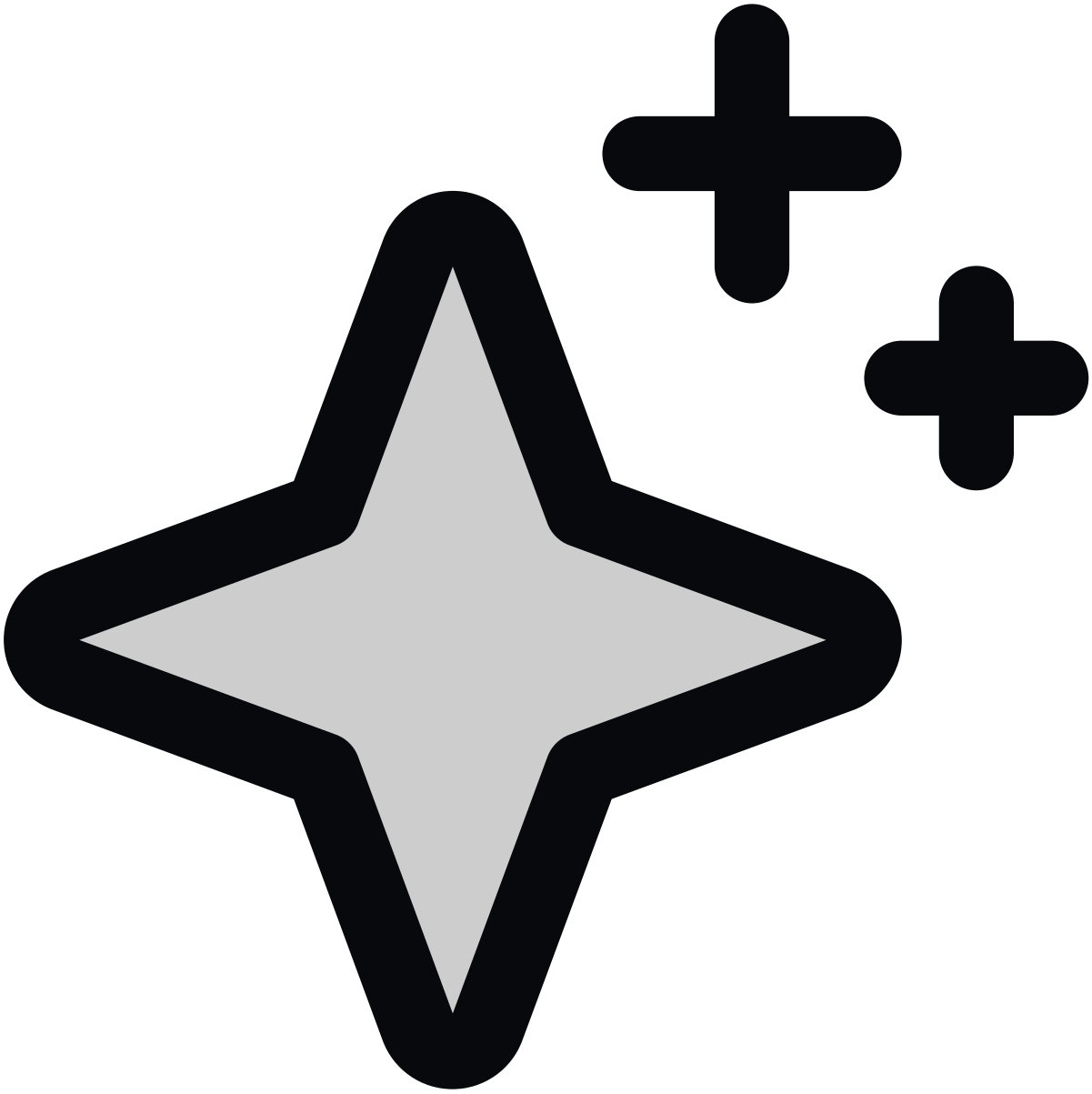


On deck for the next two quarters

- **Solana payout bridge.** Funded payouts on Solana via Wormhole-routed USDC. Base stays the canonical ledger — no leaderboard split, no fork of the rule engine. Adds payout-side reach without fragmenting volume.
- **Team challenges.** Three-trader squads with shared pooled equity, shared -4% / -8% drawdown floors, and a team leaderboard with its own cash bracket. Solves the "I trade better with a desk" friction without breaking individual accountability.

- **External backstop vault.** Non-trader users deposit USDC into a vault that backstops the funded-account pool. Depositors earn a percentage of trader profit-share in exchange for taking first-loss on funded-account drawdown. Hard cap on vault size so vault depositors cannot dilute trader payouts.
- **Quarterly audit publication.** Third-party reconciliation of every payout transaction, treasury wallet balance, insurance reserve draw, and referral cash transfer. Published so the "Verified Rewards" counter on the leaderboard has a named auditor behind it.

#



What we will not build

The list of categories Dexter has explicitly committed not to enter:

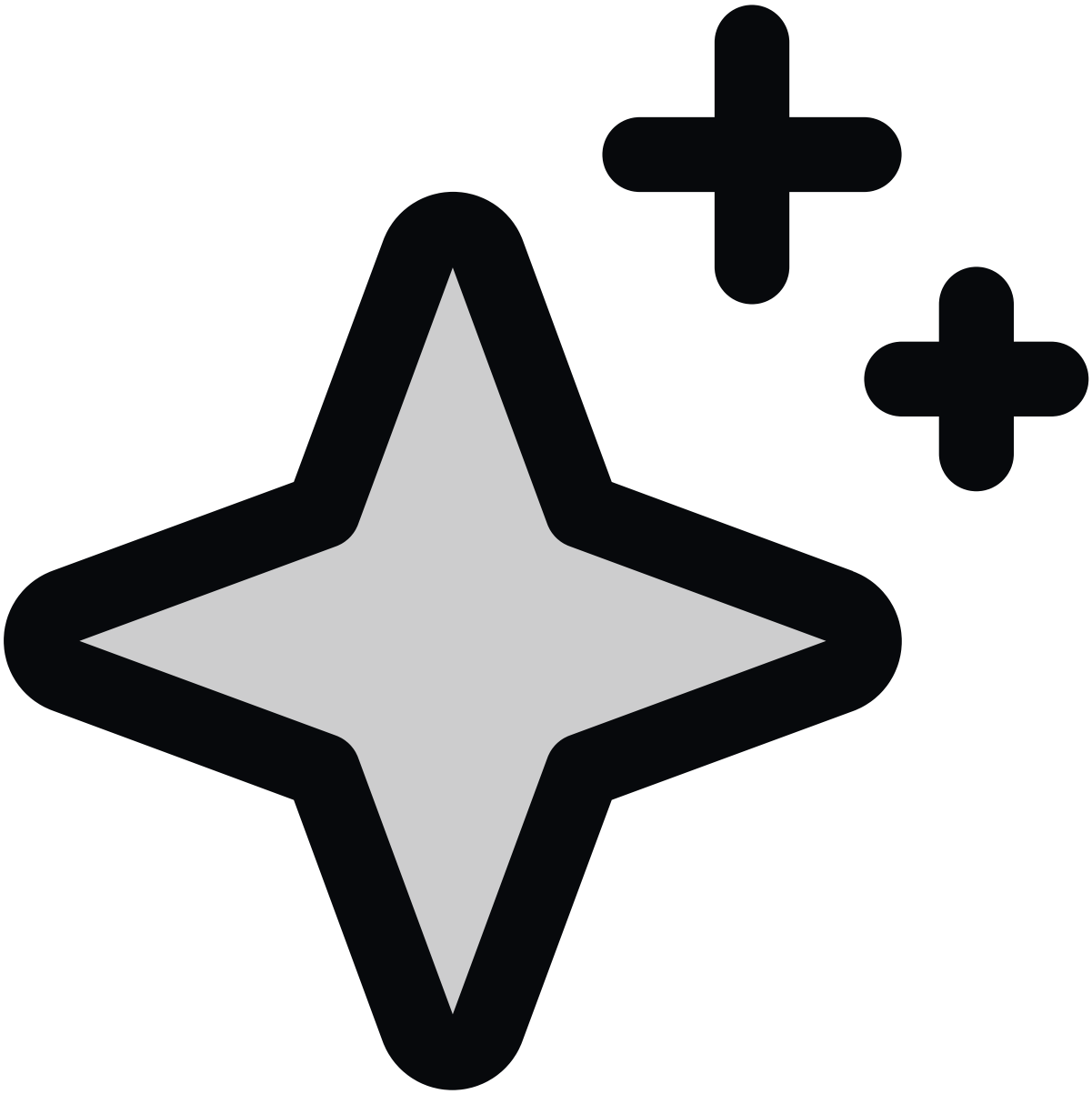
- **A chain of our own.** Base is the settlement layer. A captive L2 would split liquidity and add a new trust assumption for no user benefit.
- **Staking lockups paying synthetic APR.** A yield independent of real trading PnL taxes real traders to subsidize passive deposits. Holding DXTR will not earn an APR from nowhere; airdrop allocation is the only reserved supply for non-traders.
- **Internalized order flow.** Dexter is not the counterparty on its own listings. Funded accounts trade against the same public order book paid accounts trade against — anything else would silently corrupt the leaderboard.
- **Discretionary rule resets.** The +10% / -4% / -8% rule set is coded. The team will not "review" a missed candle, "extend" an evaluation window, or "round up" a marginal pass. The rules govern; nothing overrides them.
- **Closed-source compliance overrides.** Sanctions list and jurisdiction-block enforcement is at the cashier, with the void reason in the public ledger. There is no quiet whitelist.

CHAPTER 30

DEXTER IN ONE VIEW

Dexter is a perpetual DEX on Base with an on-chain prop-firm layer on top. Buy a pack between $49 and $299, prove +10% inside 30 days under -4% daily and -8% total drawdown floors, and trade a $5K–$50K funded account that pays 90 / 10 in your favor. Every payout posts on Base.

#

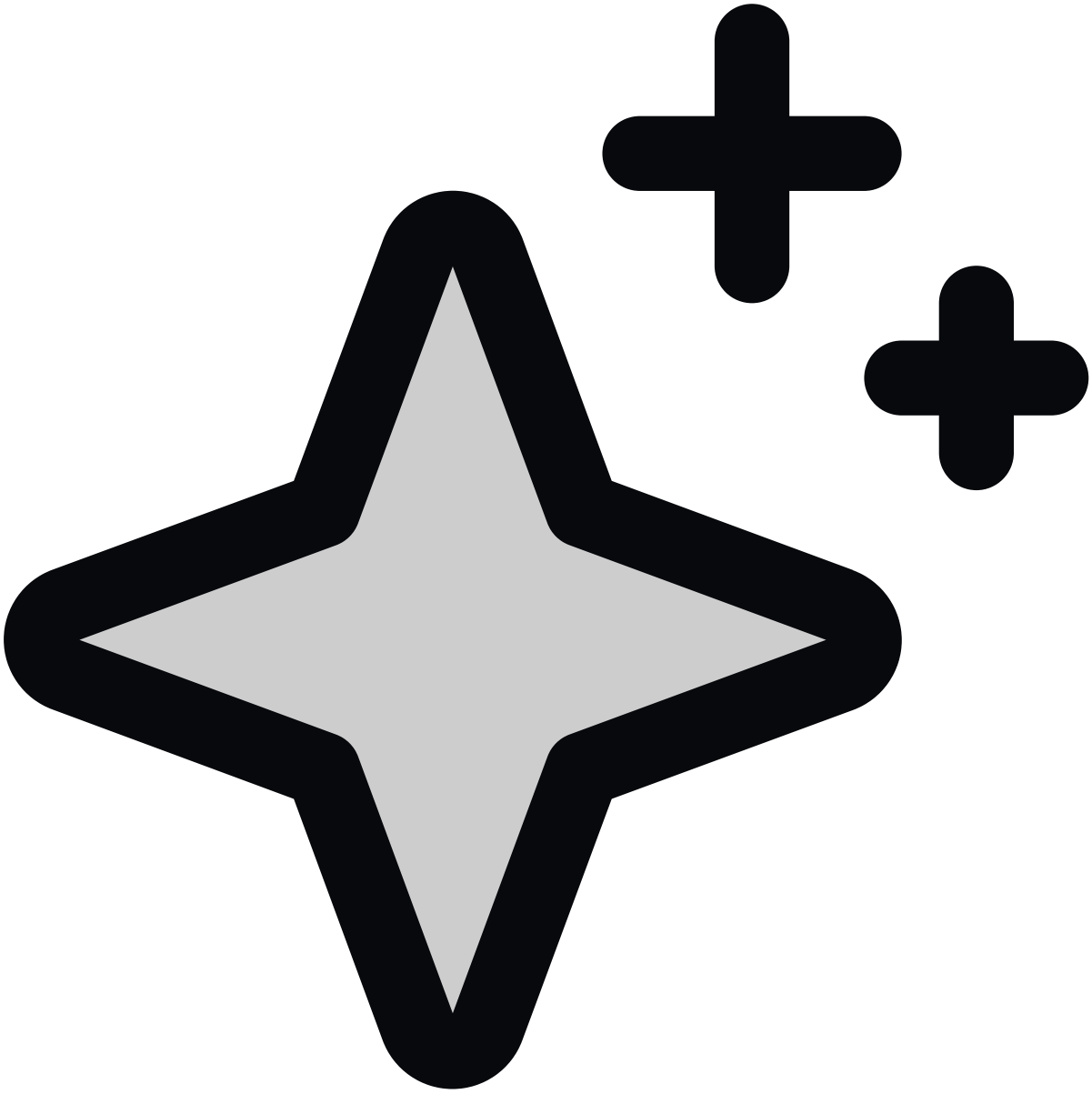


The challenge in numbers

PACK	FEE	ALLOCATION	REFEREE BONUS	MAX REFERRAL CASH
Starter	$49	$5,000	+50 XP	$5.88
Growth	$99	$10,000	+100 XP	$11.88
Swing	$199	$25,000	+250 XP	$23.88
Elite	$299	$50,000	+500 XP	$35.88
Demo	free	$25,000 simulated, 14 days	—	—

RULE	VALUE
Profit target	+10% on the allocation
Daily drawdown floor	-4% close-to-close on equity
Total drawdown floor	-8% from the high-water mark
Evaluation window	30 days from pack activation
Format	1-Step or 2-Step (same fee, same payout)
Profit split (funded)	90% trader / 10% protocol
Payout currency	USDC on Base
Payout latency	under 24 hours after KYC clears
KYC trigger	first paid payout only; demo never triggers
Jurisdiction blocks	IR / KP / SY / CU and OFAC / EU / UK sanctions lists

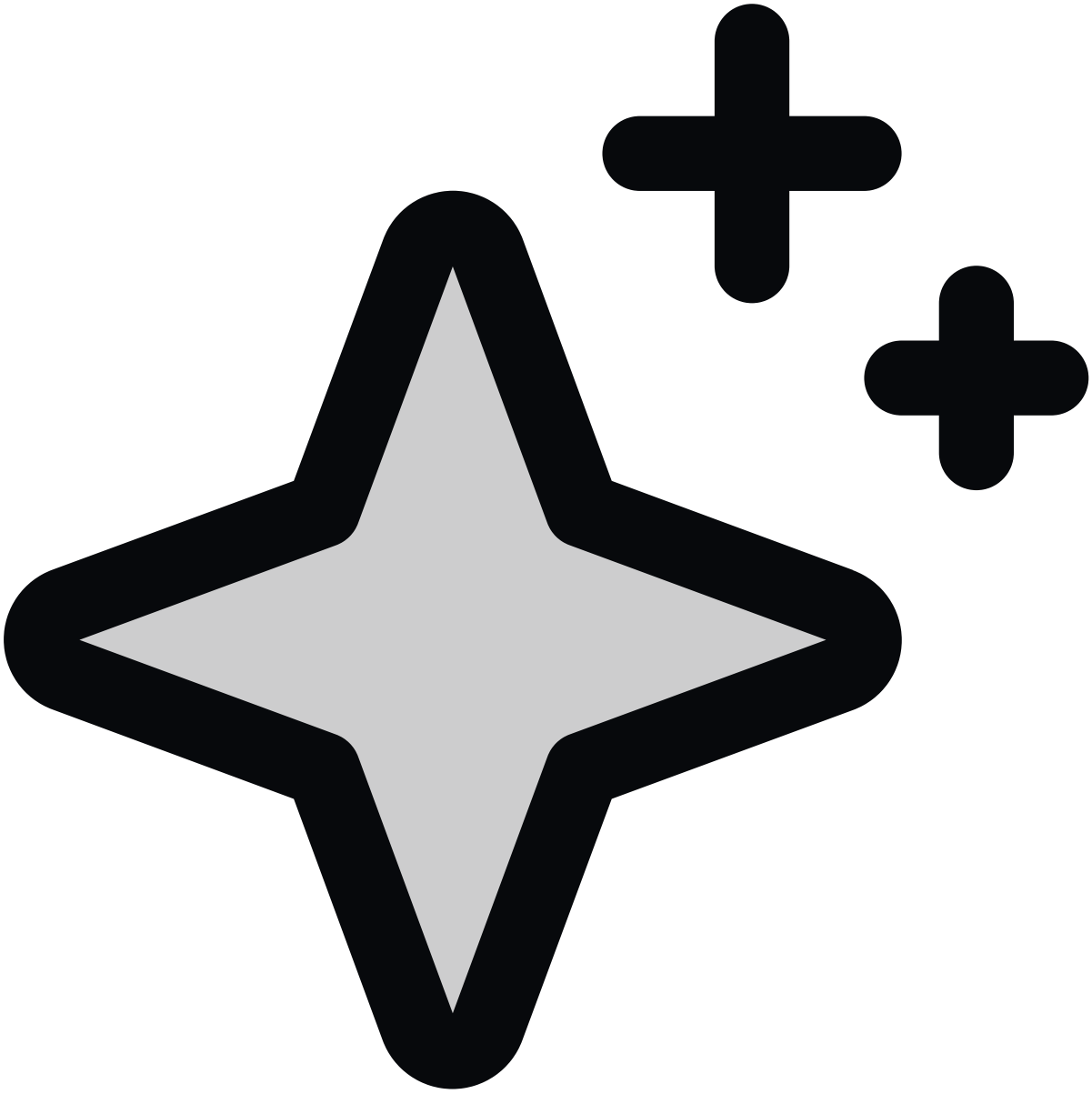
#



Reward rails

RAIL	WHAT IT PAYS	WHEN IT UNLOCKS
Trader profit-share	90% of realized PnL on the funded account	On a clean pass + KYC clear
Leaderboard cash	Ranked brackets at #1, #2–3, #4–10, #11–50, top 100 each season	End of season, after podium risk review
Referral cash — base	8% of paid pack fee	Referee clears min-active days + min-trade-count
Referral cash — bonus	+4% of paid pack fee	Referee passes the challenge (or trades 2x compliance volume on a clean record)
Referee XP bonus	+50 / +100 / +250 / +500 XP by pack tier	One-time, at first paid pack purchase via the code
Daily streak multiplier	Up to 2x XP at 30 consecutive UTC trading days	Resets on a missed day
DXTR airdrop record	40% of fixed 1B supply, weighted by skill PnL + referred pack volume	Snapshot at end of season 2, claim at TGE

#

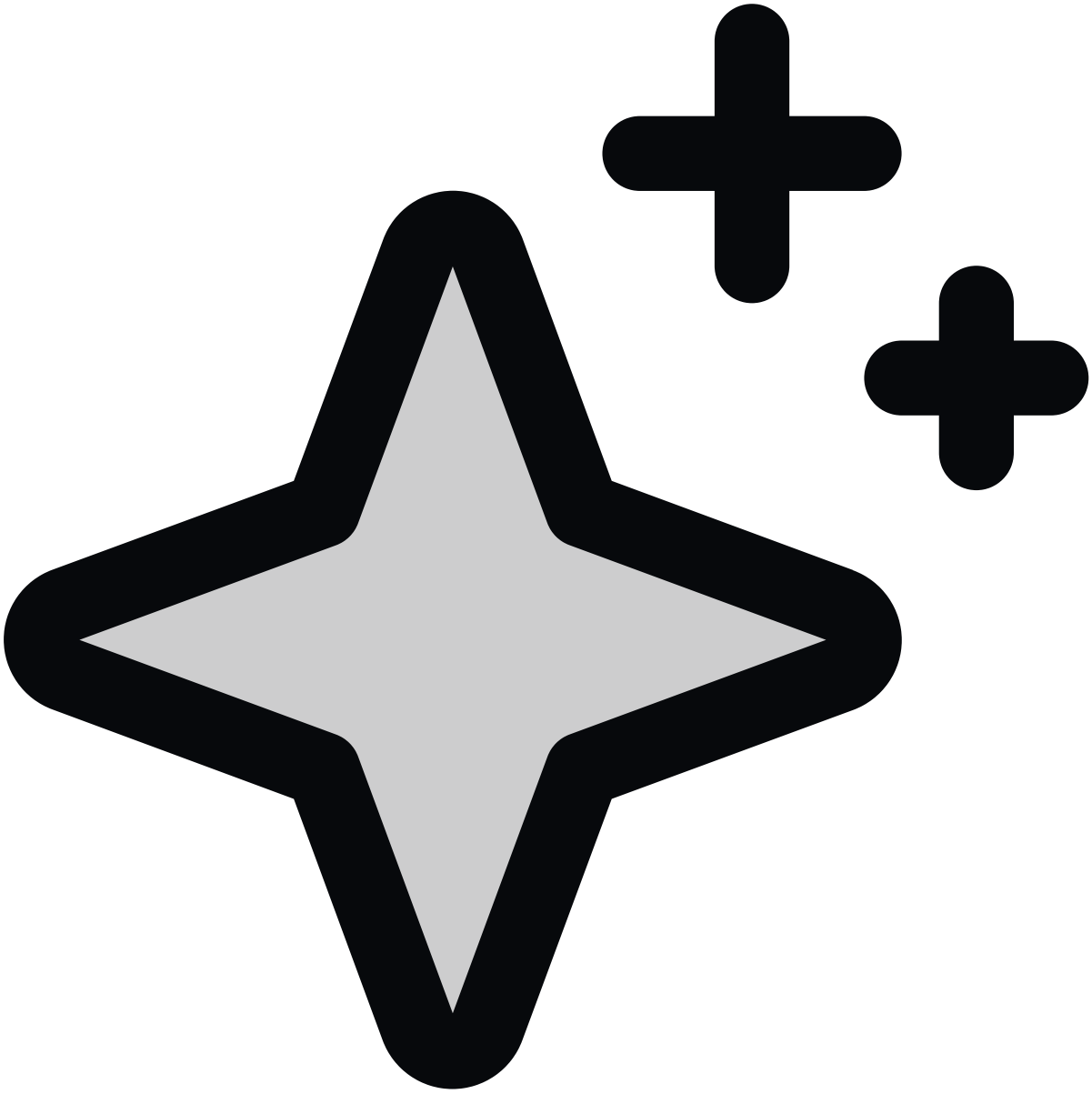


Trust anchors

- **Every payout posts on Base.** The "Verified Rewards" counter on the leaderboard reconciles hourly against real Basescan transactions — not a marketing display.
- **Same matching engine as the public DEX.** Challenge fills price off the same Pyth-backed oracle as paid accounts. No quiet worse fills for funded traders; the runtime emits a fill receipt the contract verifies on commit.
- **Oracle cross-check.** Pyth Hermes is primary; Chainlink, RedStone, and Uniswap V3 TWAPs back-check it. Stale, deviating, or paused feeds are rejected before they reach the book.

- **Rules are coded, not discretionary.** +10% / -4% / -8% is enforced by the challenge engine. No soft resets, no extended windows, no rounded-up passes — and the pass criteria cannot be moved by a governance vote.
- **Vaults live in separate addresses.** Collateral vault, insurance reserve, treasury, and season reward pool are distinct contracts; failures on one rail do not cascade to another.
- **Podium and high-value withdrawals are reviewed.** Top 3 each season and any withdrawal above $5,000 pass an ops review for position-size hygiene, rule-edge cases, and copy-trading detection before the transaction leaves treasury.
- **Compliance enforced at the cashier.** One-time KYC at first paid payout; IR / KP / SY / CU jurisdiction blocks and OFAC / EU / UK sanctions lists checked before any cash moves.

#



Your first fifteen minutes

1. Open app.dexter.market/dex/airdrop and connect a wallet — no email, no password.
2. Start with the free $25K demo if you want to feel the matching engine before paying. No KYC, no payout, full rule set.
3. Pick a pack size when you are ready. Pay in USDC, ETH, or BTC on Base. Pack fee is the entire upfront cost.
4. Trade the rules: stay above -4% close-to-close per day, stay above -8% from the equity high-water mark, push above +10% inside 30 days.

5. On a clean pass the funded account is provisioned in the same UI. Request payout, clear one-time KYC, and the USDC transfer lands on Base in under 24 hours — 90% to you, 10% to the protocol.

CHAPTER 31

GLOSSARY

A

ADL Auto-Deleveraging

Last-resort risk control. When a position is liquidated and the insurance reserve cannot cover the loss, the matching engine partially closes profitable positions on the other side at the mark, so bad debt does not propagate. ADL is rare and only triggered after staged liquidation and the insurance reserve fail. See Liquidations and protections.

Admission order admission

The pre-match gate every order passes through: signature validity, agent authorization, monotonic sequence, time-window checks, IOC-only policy. Orders that fail admission never see the matching engine. See Admission, signatures, and order commitments.

Airdrop record

The on-chain account of every challenge-related event keyed to a wallet — pack purchases, pass/fail flags, referral cash, leaderboard rank. At season close, the record is the snapshot used to compute each wallet's DXTR allocation. The record is not a balance; it is a verifiable history.

B

Base

The L2 blockchain Dexter settles on. All payouts (trader profit-share, leaderboard cash, referral cash) are USDC transactions on Base, anchored to a wallet address. Every transfer is visible on Basescan.

bps basis points

A unit equal to 0.01%. Used in fee schedules and slippage stats. A 60 bps taker fee is 0.060% on notional.

C

Challenge evaluation

The 30-day window in which a paid trader must reach +10% return while staying inside -4% daily and -8% total drawdown. Passing unlocks a real funded account paying 90 / 10 in the trader's favor. See Entry and access.

Close-only

A market posture where existing positions can be reduced but new exposure cannot be opened. The engine uses it to wind risk down without halting trading. See Order flow and market states.

Commit guard

Production policy that requires every order to be hashed and registered in the on-chain OrderCommitRegistry before the matching engine touches it. Provides public time-stamped evidence that an order existed before it was filled. See Admission, signatures, and order commitments.

D

Degraded state

A market posture where the price layer no longer meets the normal multi-source quorum but liveness is preserved on a reduced source family. Fresh risk is restricted; reduction stays open. See Sessions and degraded states.

Dexter

The first on-chain prop trading firm: a self-custodial perpetual DEX on Base with a challenge layer on top. Pay $49–$299 for a pack, prove a +10% return inside 30 days under drawdown rules, trade a $5K–$50K funded account, keep 90% of realized profits.

Drawdown

The percentage decline of an account's equity from its high-water mark. Dexter's challenge has two floors: -4% close-to-close daily and -8% total from the high-water mark. Either breach ends the attempt. See Entry and access.

DXTR

Dexter's protocol token. Fixed 1B supply. Used for governance voting, fee tier discounts, and as the unit of the season-end airdrop record. Not required to trade or earn — packs, profit-share, leaderboard, and referrals all pay USDC. See DXTR token and participation.

E

Evaluation window

The 30-calendar-day clock that starts when a paid pack is activated. The trader has this window to reach +10% target without breaching daily or total drawdown. Expires automatically.

F

Funded account

The real $5K, $10K, $25K, or $50K trading account a trader unlocks after passing the challenge. Realized PnL splits 90 / 10 to the trader. Funded losses are the firm's; the trader's downside is capped at the original pack price.

Funding funding rate

An 8-hour periodic payment exchanged between long and short holders to keep the perpetual price tracking the underlying index. Longs pay shorts when longs are crowded; vice versa. Dexter retains a 1% admin fee on settled funding above $1,000. See Funding, fees, and risk carry.

G

Gateway

The public boundary that every client touches before reaching the matching engine: API endpoints, agent authorization, signed-order admission, read model. Same surface for retail wallets, institutional API clients, and copy-trading bots. See Gateway and read model.

Governance

DXTR-staked voting on fee schedule, leaderboard prize pool sizing, market listings, and treasury actions. Challenge rules (+10% / -4% / -8%) are intentionally not governable in v1 — they are fixed in the contract.

H

Halted

The strictest market posture: no new orders, no reductions, position is frozen, equity is not marked, evaluation clock pauses. Used only when the oracle layer or matching engine cannot operate safely.

High-water mark

The all-time highest equity an account has reached during its evaluation window. The -8% total drawdown floor is measured from this point, not from the starting balance. A profitable trader who pulls back from a new high triggers the floor faster than one who never made it past starting capital.

I

IOC Immediate-Or-Cancel

Production-required order type. Either fills against existing book depth in the same instant, or the unfilled remainder cancels. No resting orders, no GTC.

Insurance reserve

Protocol-owned reserve that absorbs liquidation shortfalls before they become bad debt for users. Funded by 50% of the residual collateral on each forced liquidation. Held in a separate contract address; never mixed with user collateral or operating treasury.

K

KYC

Sumsub-class identity verification (government ID + selfie + sanctions screen). Dexter requires it only at the first real-money payout, not at sign-up or pack purchase. Sanctions blocks: OFAC, EU, UK, plus countries IR, KP, SY, CU.

L

Leaderboard

Per-season ranking of funded traders by realized PnL and risk-adjusted score. Top brackets (#1, #2-3, #4-10, #11-50, top 100) earn USDC payouts on Base at season close. Podium finishes are reviewed by ops before the batch leaves treasury. See Scoring and airdrop.

Liquidation

Forced closure of a position when account maintenance margin falls below threshold. Staged: the engine attempts partial reductions first; ADL is the last resort. Residual collateral splits 50% to the keeper bounty, 50% to the insurance reserve.

M

Maintenance margin

The minimum margin a position must hold to stay open. Falling below it triggers the liquidation candidate scan. Always lower than initial margin; trader must keep buffer to avoid being flagged on minor adverse moves.

Maker / Taker

Fee classification by order behavior. Makers add liquidity (post resting orders that wait to be filled — Dexter has limited resting flow under IOC policy); takers consume it. Dexter's schedule: 0.020% maker rebate-eligible / 0.060% taker on notional.

Mark price

The executable price the engine uses to mark account equity, trigger liquidations, and check drawdown rules. Derived from oracle reference + venue skew + spread + depth + max-move continuity. Not the same as the index price. See Mark construction and executable price.

Matching engine

The single-writer runtime that fills orders. Same engine for funded accounts, demo accounts, and public users — no privileged prop lane. See Market engine.

Multi-sig

Multi-signature wallet. Dexter operating wallet uses 2-of-3 multi-sig for movements above $10K; governance and insurance actions require 3-of-5 with at least one external signer. Every signed transaction is posted to Base.

O

Oracle

External price feed Dexter uses to mark perp markets. Production uses Hermes / Pyth as primary and a redundant HTTP oracle mux for cross-check. Sources are cross-checked for freshness, deviation, and depth before any tick promotes to executable mark. See Price layer.

OrderCommitRegistry

On-chain contract on Base where every paid-account order hash is committed before the matching engine processes it. Provides public, time-stamped proof of order arrival — the basis for after-the-fact fraud review.

P

Pack

A one-time challenge purchase. Four sizes: $49 Starter ($5K allocation), $99 Growth ($10K), $199 Swing ($25K), $299 Elite ($50K). Paid in USDC, BTC, or ETH on Base. Funded account unlocks on pass; pack price is the trader's only downside.

Perpetual perp

A derivative contract with no expiration. Holders pay funding to keep the contract price tracking the index. Dexter currently lists BTC, ETH, SOL perps; equity, metal, and energy perps are on the roadmap.

Podium review

Human operations check on the top 3 leaderboard finishers each season before the prize batch leaves treasury. Looks for wash trading, sybil patterns, and oracle exploits. Also applied to single withdrawals above $5,000.

Profit share 90 / 10 split

The realized PnL split on funded accounts: 90% to the trader, 10% to Dexter. Identical for every funded wallet, contract-bounded, no escalating tiers or time decay.

R

Reduced

A market posture between live and close-only. Fresh risk is allowed but with tighter caps and higher skew-adjusted fees. The engine uses it to bleed off pressure before forcing close-only.

Referral

Cash an upstream wallet earns when a referee buys a paid pack: 8% base + up to 4% activity bonus, capped per pack tier ($5.88 / $11.88 / $23.88 / $35.88). Demo seats and self-referrals do not pay. See Referrals.

Reward pool

A separate contract address holding the leaderboard prize pool for the current season. Funded at season open so the published payout brackets cannot be quietly reduced mid-season. Drawn down at season close.

S

Season

A bounded competition period in which leaderboard rank, referral activity, and the airdrop record accumulate. At season close the reward pool pays out and the airdrop record is snapshotted.

Session-closed

A market posture used by session-based assets (equity, metal, energy perps) when the underlying market is on schedule break. Positions are held but not marked against stale ticks; drawdown rules are checked at next session open. Distinct from degraded — the data is fine, the underlying is just off-hours.

Settlement

The on-chain release of collateral or payout against a published state root. Each request is anchored to an account nonce, included in the next state root, gated by a proof window, then finalized by the vault contract. See State roots, withdrawals, and settlement flow.

Skew

Directional imbalance of open interest on a market — how lopsided long vs. short positioning is. The engine raises taker fees and tightens margin on crowded directions to keep risk balanced.

State root

A cryptographic commitment (Merkle root) summarizing the entire account ledger at a point in time. Published periodically to Base. Payouts can only be finalized against a published root with a matching account proof.

Sumsub

Third-party identity-verification provider. Dexter's KYC runs through Sumsub-class flows: government ID, selfie liveness, sanctions screen. Dexter stores only pass/fail status, the provider ID, and a risk tag — not the underlying documents.

T

Treasury

Protocol-owned operating wallet holding fee revenue and the funding admin slice. Pays operating costs and refills the reward pool at season open. Multi-sig 2-of-3 above $10K. Distinct from user collateral and insurance reserve.

U

USDC

The stablecoin Dexter prices, collateralizes, and pays out in. All challenge fees, profit-share payouts, leaderboard cash, and referral cash settle in USDC on Base.

V

vAMM virtual AMM

Liquidity model Dexter runs in production. Execution prices flow through skew-sensitive logic, spread controls, inventory caps, and open-interest pressure rather than depending on passive maker quotes in a book.

Vault

Contract address holding user collateral. Releases only against a published state root and a matching proof. Cannot be drained to cover protocol operations, treasury actions, or another user's payout. See Vault and treasury.

Verified rewards

The leaderboard counter on dexter.market that sums every payout transaction posted on Base. Updated hourly from on-chain transfers, not from a database row — anyone can reconcile the number against Basescan themselves.

W

Wallet

The Web3 account address that anchors every Dexter relationship: pack purchase, trade history, leaderboard rank, referral cash, payouts. MetaMask, WalletConnect, Coinbase Wallet, and Reown are supported. No email or password — a signature is the only login.

Withdrawal proof

The Merkle proof a wallet presents to the vault contract to release collateral against a published state root. Without a valid proof the vault rejects the request — no operator key can override. See Withdrawal proofs, disputes, and root challenges.