Risk Assessment – Pendle PTs

Anthias Labs

1 August 2025

Abstract

Pendle's Principal Tokens package fixed-rate, zero-coupon bond economics into ERC-20s, letting DeFi protocols lock in predictable yield while retaining on-chain composability. This report explains every layer that supports PTs on Pendle V2, from the SY wrapping standard (EIP-5115) through the logistic-curve AMM and permissionless order book, mapping their collective risk surface for money-markets that wish to onboard PT collateral.

A focussed case study on kHYPE, the largest HyperEVM liquid-staking token, shows how these controls translate to asset-specific recommendations. We explain kHYPE's auto compounding yield mechanics, staking–unstaking flows and validator oracle design, then quantify the incremental risks introduced when kHYPE is wrapped into PT-kHYPE. Resulting parameter recommendations include an exponential PT price-feed oracle configuration, LTVs inside correlated-asset markets, weekly re-calibration and conservative supply caps that scale with on-chain liquidity.

In essence, the framework validates that PT collateral can be onboarded safely provided that oracle cadence, liquidity depth and governance monitoring keep pace with each asset's life-cycle.

1 Protocol Overview

Pendle exists because DeFi's notion of "programmable money" ran into a bottleneck: every yield-bearing token has its own separate interface for deposits, withdrawals, reward accounting, and pricing. This led to fragmentation and made it expensive for wallets, money-markets and treasuries to support more than a few assets. It also meant that yield was attached to the principal, so there was a gap for products which let users hedge, leverage, or trade interest-rate views in a capital-efficient way.

Pendle tries to solve both of the issues at the same time. It wraps any yield-bearing token into a Standardised Yield (SY) format, instantly normalising every deposit into the same API, and then splits that SY into a Principal Token (PT) and a Yield Token (YT). Once yield and principal are unbundled they can be bought, sold, hedged, or used as collateral. A purpose-built AMM along with an order-book provides the liquidity needed for those trades. In TradFi PTs can be correlated with zero coupon bonds and YTs can be considered as the detached coupon payments.

At a systems level Pendle consists of four broad layers:

- The Yield-wrapping layer is where SY contracts live. Each contract acts as a vault around the underlying protocol, exposes a consistent deposit/redeem surface, and tracks both exchange-rate growth and extra reward tokens.
- The Tokenisation layer is the engine that mints PT and YT for every SY-maturity pair and burns them back into SY either before or after expiry.
- The Trading and Liquidity layer consists of the V2 AMM, which holds PT and SY reserves, concentrates liquidity so price impact stays low, and uses a flash-swap to route YT trades through the same pool. A cross-chain order book for limit orders quoted in implied-rate terms is also present along with a built-in oracle that records a cumulative log-rate value each block.
- Finally, the Governance & Incentive layer is powered by vePENDLE locks on Ethereum: lockers vote every Thursday on which pools should receive the next week of emissions, Gauges embedded in each AMM stream those emissions (plus underlying rewards) to LPs with a 40 / 60 formula, and merkle proofs mirror both locks and votes to every supported chain.

In effect Pendle turns every yield source into programmable blocks and supplies the marketplace, price oracle, and incentive rail which makes them work together in a smooth manner. This lets Pendle create a composable, interest-aware DeFi stack.

2 High Level Protocol Mechanics

In this section we look at components involved in the user workflow from a high level. It starts with the user depositing a yield-bearing token into the protocol, follows each transformation step, and shows how the AMM keeps the resulting assets liquid. After that we discuss how

Pendle exposes the Oracles to price these derivatives along with the remaining components like orderbook, router, vePENDLE and gauges.

2.1 Standardised Yield (SY)

Every position in Pendle begins as a deposit into an SY contract. SY is an ERC-20 wrapper that sits on top of a yield-bearing token such as stETH. By wrapping, it gives every source of yield the same surface:

- **deposit** and **redeem** move value in and out.
- **exchangeRate** shows how much the wrapped shares are now worth in the accounting asset.
- **claimRewards** pulls any incentive tokens the underlying protocol emits.
- **getTokensIn** and **getTokensOut** list which tokens can be used when minting or redeeming.

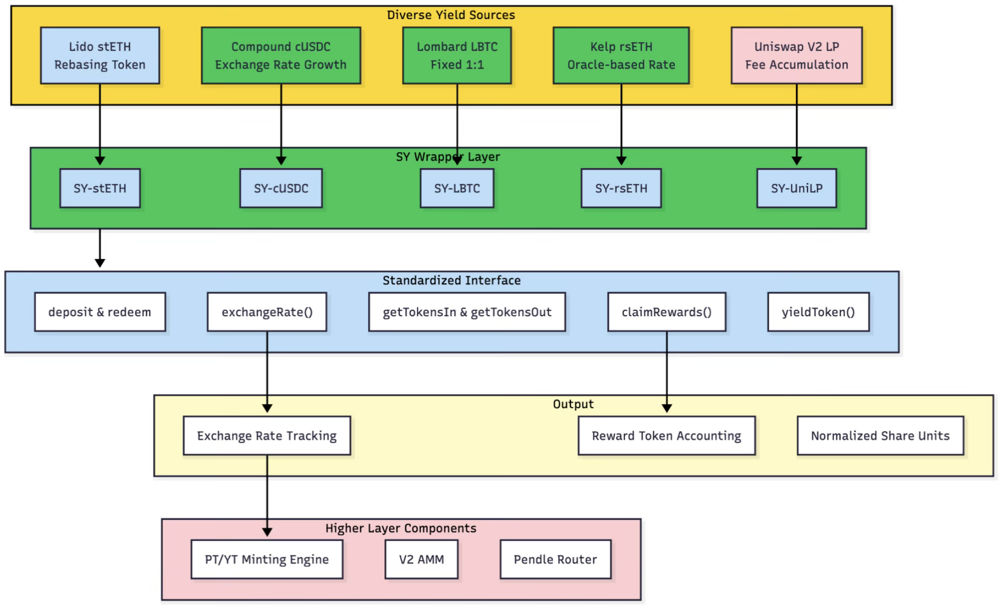

Figure 1: SY wrapper flow diagram

Because each SY follows this interface, later contracts never need to learn the details of the underlying protocol. SY stores the yield history in a simple index, so interest compounds automatically. If the yield source distributes extra tokens for example GLP paying ETH, the contract tracks a reward index for each token and lets the user claim on demand. One unit of the underlying asset always comes back out as one unit of SY, regardless of price drift inside the wrapper.

2.2 Principal Token (PT)

The principal side of an SY position is PT. One PT promises to redeem one accounting asset (e.g. stETH in the wstETH market, USDC in the aUSDC market) after the series reaches its maturity date. Before maturity PT trades at a discount, the size of which converts directly into a fixed annual yield:

$$\text{Fixed APY} = \left(\tfrac{\text{Asset}}{\text{PT price}}\right)^{365/\text{days to expiry}} - 1$$

The discount closes as time passes, so a buy-and-hold investor locks in that rate without price risk. At expiry PT is redeemed and if the underlying token is a rebasing asset the contract handles the conversion so redemptions are exact.

2.3 Yield Token (YT)

The yield that PT gives up is represented by YT. A holder accrues the underlying yield continuously and can claim at any moment. As soon as the series matures YT stops receiving yield and its trading price falls to zero. Market price is guided by the implied APY, the forward rate investors infer from the current PT-to-YT ratio:

$$\text{Implied APY} = \left[1 + \left(\tfrac{\text{YT price}}{\text{PT price}}\right)\right]^{365/\text{days to expiry}} - 1$$

If the future realised yield ends up higher than that number, the YT buyer profits; if it runs lower the buyer loses. This structure lets traders act on interest rates directly, without needing to use leverage.

2.4 V2 AMM

The AMM holds PT and SY reserves for a single maturity. Instead of the constant-product curve it uses the same logistic shape that powers Notional's fixed-rate pools:

$$\text{ER} \;=\; \frac{\ln\bigl(p/(1-p)\bigr)}{\text{rateScalar}} + \text{rateAnchor}$$

Here p is the PT share of total pool value. The scalar shrinks as the maturity date approaches, automatically concentrating liquidity where trades take place and pushing the PT price toward the accounting asset. Because both sides of the pool are tightly correlated, impermanent loss is negligible and disappears entirely at expiry.

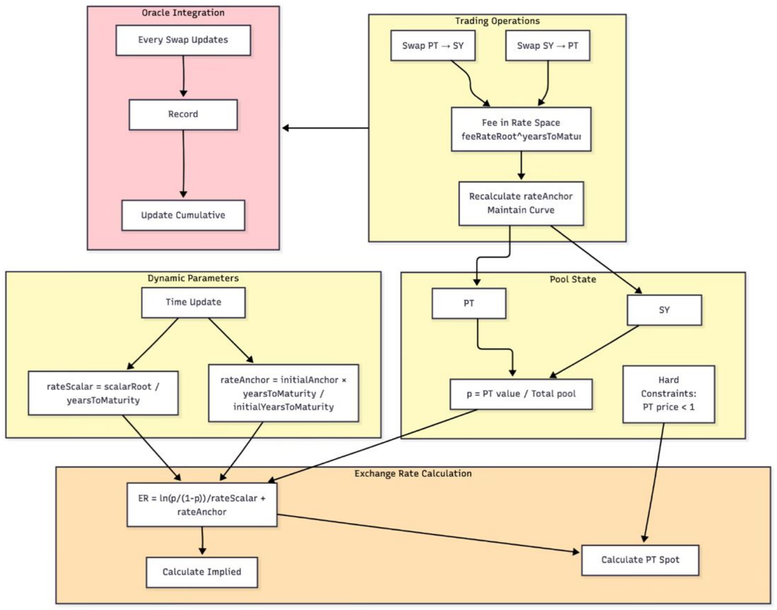

Figure 2: V2 AMM exchange-rate flow chart

Trades pay a fee by moving the exchange rate, a fixed root factor in yield space. The same pool can quote YT trades; the router uses a flash-swap: mint PT-plus-YT from SY, hand the YT to the buyer, then sell the PT back into the pool. Liquidity providers earn:

- Underlying SY yield.
- Fixed PT yield.
- Explicit and implicit swap fees.
- PENDLE emissions from the pool's embedded Gauge.

Each swap also writes a timestamped cumulative log-rate value into an observation ring. Integrators can ask for any time-weighted average price up to nine days long with one call, removing oracle risk without extra infrastructure.

2.5 Oracle layer (PT)

Every market records the cumulative natural log of its implied rate at the end of each swap:

```
Observation {
    blockTimestamp,                     // 32-bit
    lnImpliedRateCumulative,            // 216-bit
    initialized                         // 8-bit
}
```

Because `ln(rate)` turns multiplication into addition, the difference of two observations divided by the time delta yields the geometric-mean implied rate for any window up to

65_536 blocks (nine days on Ethereum). The AMM itself stores the ring; no external keeper is needed.

Pendle ships two wrappers:

- PendleOracle: pure-view contract that returns PT/SY or PT/asset TWAP.
- PendleChainlinkOracle: exposes the same value through `latestRoundData()` so protocols already wired for Chainlink can adopt PT pricing without code changes. A `WithQuote` variant multiplies by an external asset/USD feed.

Before querying, an integrator should expand `observationsCardinalityNext` once and wait at least the TWAP period so the buffer is primed.

2.6 Router

The `PendleRouter` (currently V4, upgradeable proxy) chains arbitrary steps like swapping, adding liquidity, minting PT-YT, zapping from any ERC-20 into a single transaction. Four data bundles drive it:

- TokenInput: (tokenIn, netTokenIn, tokenMintSy, pendleSwap, swapData)
- TokenOutput: (tokenOut, minTokenOut, tokenRedeemSy, pendleSwap, swapData)
- ApproxParams: (guessMin, guessMax, guessOffchain, maxIteration, eps) for on-chain binary searches when an exact-in amount must resolve to an unknown PT quantity.
- LimitOrderData: for including off-chain maker orders in the execution path.

Direct on-chain generation works with the help of helper libraries which expose `createTokenInputSimple` but it costs more gas because the router must binary-search blindly. Using the hosted SDK is cheaper because it pre-computes the PT range, folds in routes for arbitrary ERC-20 zaps, and matches any resting limit orders before the swap hits the pool.

2.7 Order Book and Limit Orders

Pendle adds a permissionless order-book layer on top of the AMM so makers can quote PT, YT, or SY at fixed implied rates without providing continuous liquidity. Makers sign an `Order` struct:

```
{
  salt, expiry, nonce,
  orderType  // (SY_FOR_PT | PT_FOR_SY | SY_FOR_YT | YT_FOR_SY),
  token      // in or out depending on side
  YT         // defines the series
  maker, receiver,
  makingAmount,
  lnImpliedRate,
  failSafeRate
}
```

Signatures stay off-chain until a taker bundles one or more orders into `fill(params[], receiver, maxTaking, optData, callback)`. During settlement the router can call back into an arbitrary contract, letting arbitrageurs flash-swap against the AMM or other orders before they deliver the owed tokens. The price field is always `lnImpliedRate` so the order-book and AMM have the same context; `failSafeRate` guards against front-running on the SY `<->` asset conversion path. Makers cancel either order-by-order (`cancelSingle`) or globally by incrementing `nonce`.

2.8 vePENDLE, Voting, Gauges

Locking PENDLE on Ethereum mints vePENDLE that decays linearly to zero at the chosen unlock date (max 2 years). A weekly cycle begins every Thursday 00:00 UTC:

1. Snapshot: `VotingController` records each address's ve balance.
2. Voting: addresses assign weight to any combination of markets until the epoch ends.
3. Tally: `GaugeController` converts each pool's share of total votes into a per-second PENDLE emission (`actualSpeed`) for the coming week and broadcasts the numbers to every chain. Unused emissions from the previous week roll forward automatically.

Reward flow inside a market passes through an embedded `PendleGauge` that also receives the SY-level reward tokens. Distribution uses the 40 / 60 boost:

$$\text{activeBalance}_u = \min\Big(LP_u,\, 0.4\, LP_u + 0.6\,\text{totalLP}\left(\tfrac{\text{vePENDLE}_u}{\text{vePENDLE}_{\text{total}}}\right)\Big)$$

Rewards accumulate continuously, any holder can call `redeemRewards()` to claim without unstaking first. A merkle relayer can publish user lines and the global supply to each L2 so boosts and votes remain in sync across all chains.

2.9 Liquidations and helper sellers

Protocols that accept PT or LP as collateral can tap two reference contracts:

- **BoringPtSeller**: before maturity sells PT for SY via the AMM; after maturity redeems PT $\rightarrow$ SY directly.
- **BoringLpSeller**: removes LP liquidity single-sided to SY or unwraps PT+SY then converts.

Both end in SY form, letting the caller redeem to any allowed output token (per `getTokensOut`) and settle the debt in a familiar asset.

3 Deep Dive with focus on PTs

Given that we now have a basic understanding of how Pendle works at a high level, we will focus on detailed analysis of the components relevant to PTs. This will let us assess upcoming PT assets which launch on Hyperliquid in a concrete fashion. It will also help us make sure that protocols listing these PT assets have complete awareness about the associated risks so that they can set their protocol parameters accordingly.

3.1 SY

SY's abstraction eliminates protocol specific integrations and lets every higher-level Pendle contract treat diverse assets the same way. Pendle also introduced a new token standard for SY tokens, i.e. EIP-5115. Where ERC-4626 unified classic vaults, EIP-5115 encompasses any mechanism that fits the Generic Yield Generating Pool (GYGP) model. It does so by requiring every wrapper to speak a strict superset of ERC-20 while keeping mint-and-burn logic flexible enough to mirror the underlying asset.

3.1.1 GYGP recap

A GYGP tracks four parts:

- **Asset**: the accounting unit that measures pool value.
- **Shares**: ERC-20 units that convey ownership (the SY supply).
- **Reward tokens**: zero or more auxiliary tokens the pool earns.
- **Exchange rate**: Assets per Share, TotalAsset/TotalShares, expected to rise as yield accrues.

State only transitions in three cases: a user deposits or withdraws assets, the pool's asset balance changes as yield arrives, or the pool receives reward tokens. Framing every yield source this way lets one wrapper spec cover Compound cTokens, Lido's wstETH, Uniswap V2 LP tokens, Kintetiq's kHYPE, Stargate deposits, and all the rest.

3.1.2 EIP-5115 changes over ERC-20

- `deposit(receiver, tokenIn, amount, minShares, useInternalBalance)`: Converts `tokenIn` to the pool's Asset if needed, updates `TotalAsset`, mints shares, and transfers them to `receiver`. Reverts if fewer than `minShares` would be created. Supports payable ETH deposits and the "pull-from-receiver" ERC-20 pattern.
- `redeem(receiver, shares, tokenOut, minToken, burnFromInternalBalance)`: Burns shares, withdraws the proportional Asset, converts to `tokenOut`, and transfers it to `receiver`. Mirrors the internal-balance toggle of `deposit`.
- `exchangeRate()`: Syncs and returns the latest Assets-per-Share, scaled to 1e18. It must exclude any protocol-level fees siphoned off inside the wrapper, ensuring PT redemption maths remain exact.
- `getTokensIn() / getTokensOut()`: Enumerate all mintable inputs and redeemable outputs. Integrators can route deposits and withdrawals without chain-specific hard-coding.
- `yieldToken()`: Returns the address of the underlying yield-bearing token if the SY is a wrapper, or `0x0` if the SY is native.

- `previewDeposit` / `previewRedeem`: Pure helpers that quote the resulting shares or output tokens without state changes, bounded so a real call can only improve the user's result.

3.1.3 Exchange-rate

- For rebasing assets (wstETH, sUSDe) `exchangeRate` pulls directly from the yield token.
- For fixed-rate wrappers (LBTC at 1:1) it is constant.
- For LRTs without a canonical token on the current chain (rsETH on Arbitrum) Pendle updates a chain-agnostic oracle signed by off-chain relayers.

3.1.4 Watermark-rate

For interest-bearing tokens (IBTs) whose exchange-rate can decline (e.g., restaking with slashing risk), Pendle records a *Watermark Rate* which is defined as: the highest `exchangeRate()` ever observed for the series' SY. When `exchangeRate` falls below Watermark PT redeems less than one accounting asset at maturity, proportional to the fall and YT stops earning yield until `exchangeRate` climbs back above Watermark.

This mechanism prevents the protocol from minting value out of thin air during prolonged negative-yield periods and caps losses to the real decline in the underlying. Interfaces expose Watermark alongside the current rate so protocols and investors can price the downside explicitly.

3.2 PT & YT Mechanics

3.2.1 Principal Tokens (PT)

A PT is an ERC-20 that represents the right to redeem a fixed amount of the accounting asset on or after a preset maturity date. That fixed amount is always 1, but expressed in SY terms it equals $1/\text{exchangeRate}_{\text{at\_expiry}}$.

At the moment of split each PT is worth less than one accounting asset because the holder gives up future yield. The discount is what produces a fixed return:

$$\text{Fixed APY } = \left(\frac{1}{\text{PT price}}\right)^{\frac{365}{d}} - 1, \qquad d = \text{days to maturity}$$

As time passes two things happen. First, `exchangeRate` continues to rise inside SY, so the same PT promises more SY-units when expressed that way; second, the discount naturally decays as the redemption date draws nearer. Together they push the market price upward in a nearly deterministic way. Immediately after maturity any PT can be submitted alone to `redeemPT`, which burns it and transfers the exact accounting asset share to the caller.

Pendle's AMM enforces $\text{PT\_share} \leq 0.96$ of pool value. Substituting $p = 0.96$ into the swap curve yields a hard minimum PT price and therefore a hard maximum implied yield. Those bounds slide upward as `rateScalar` shrinks, guaranteeing that a PT cannot be pushed to zero even in stressed markets.

Investors can purchase PT to lock in a guaranteed yield, treasuries may use it to improve staking revenue, and money-markets treat it as zero-coupon collateral. Because redemption is a pure SY transfer, PT has no re-entrancy path back into the AMM, simplifying collateral risk analysis.

3.2.2 Yield Tokens (YT)

YT is entitled to every increment in `exchangeRate` plus any reward tokens distributed to the underlying protocol, from the split date until the same maturity. The contract tracks two numbers per holder:

- **scaledPrincipal**: their YT balance expressed in SY units.
- **claimableInterest**: SY that has accumulated but not yet withdrawn.

When `exchangeRate` jumps from E_0 to E_1 the new interest is

$$\Delta SY \;=\; \text{scaledPrincipal}\left(\tfrac{1}{E_0} - \tfrac{1}{E_1}\right)$$

and is added to `claimableInterest`. A call to `redeemDueInterestAndRewards` transfers the SY (and any reward tokens) and zeroes the slot. At maturity the stream stops, `scaledPrincipal` is wiped, and YT trading price converges to zero.

Market participants quote YT in implied APY:

$$\text{Implied APY} \;=\; \left[1 + \left(\tfrac{\text{YT price}}{\text{PT price}}\right)\right]^{\frac{365}{d}} - 1$$

If that number sits below the expected realised yield, buying YT is profitable; if it sits above, selling YT (or buying PT) captures the premium. Instead of a separate pool, the router trades YT through the PT/SY liquidity. To buy YT, it borrows additional SY from the pool, mints PT + YT, sends YT to the trader, then returns the borrowed SY by immediately selling the PT back into the pool.

3.3 V2 AMM

Pendle's liquidity engine is an AMM that trades one Principal Token against its matching SY share. Because the two assets are tightly correlated, and because PT drifts toward the accounting asset as the series approaches expiry, the pool needs a curve that adapts over time and keeps capital where trades take place. Pendle adopts the logistic design first introduced by Notional Finance, then overlays additional guardrails so as to manage it according to its own risk framework.

3.3.1 Curve geometry and parameters

Each pool is characterised by two numbers set at bootstrap:

- **scalarRoot**: controls how narrow the liquidity band is. A larger value compresses liquidity, raises the floor price of PT, and narrows the yield range that traders can move through before hitting steep slippage.
- **initialAnchor**: chooses where that band sits. Liquidity is centred on the implied yield represented by 1/initialAnchor.

These constants flow into the live parameters every swap:

$$\begin{aligned} \text{rateScalar}(t) &= \frac{\text{scalarRoot}}{\text{yearsToMaturity}} \\ \text{rateAnchor}(t) &= \text{initialAnchor}\,\frac{\text{yearsToMaturity}}{\text{initialYearsToMaturity}} \end{aligned}$$

When time t reduces, `rateScalar` grows and `rateAnchor` falls, so liquidity progressively shifts toward parity. With no trades the PT price therefore rises in a smooth, predictable arc.

At any instant, let p be the PT fraction of total pool value. The exchange-rate that maps PT to SY is

$$\text{ER} \;=\; \frac{\ln\bigl(p/(1-p)\bigr)}{\text{rateScalar}(t)} + \text{rateAnchor}(t)$$

From ER one can compute a running implied yield and the spot price of PT. Two hard constraints are always enforced:

- PT price must remain below the accounting asset price (always less than 1).
- PT's share of pool value is capped at 96 %. Substituting $p = 0.96$ into the curve yields a deterministic minimum PT price and therefore a maximum implied APR that the market can quote.

3.3.2 Fee model and continuity

A swap pays its fee in rate space. The contract multiplies the fee-free exchange rate by $\text{feeRateRoot}^{\text{yearsToMaturity}}$ (positive exponent when buying PT, negative when selling). Because the adjustment happens on the x-axis of the curve rather than the y-axis, slippage remains symmetric whether a trader moves from PT to SY or vice-versa. Before the trade finalises, the algorithm recalculates `rateAnchor` so that the post-trade curve still passes through the old implied rate. Due to this, the "continuity rule" prevents arbitrageurs from harvesting a gap created by their own transactions.

3.3.3 Liquidity provision and impermanent loss

Adding liquidity means depositing PT and SY in the current pool ratio and receiving LP tokens.

Impermanent loss is very low for three reasons:

1. **Price convergence.** Every PT must equal one accounting asset at maturity, so the PT/SY price ratio collapses toward 1 and any divergence experienced earlier disappears for LPs who keep their position until expiry.
2. **Correlated reserves.** PT and SY respond to the same underlying yield moves, reducing day-to-day divergence compared with unrelated token pairs.
3. **Adaptive curve.** As `rateScalar` grows the band narrows, concentrating depth where trades happen and limiting the reserve imbalance created by large swings.

An LP therefore holds a time-weighted mix of implied APR (via PT) and realised underlying APR (via SY). If implied yield is higher than the realised yield, holding PT alone can outperform liquidity provision; if the opposite holds, SY will be superior. Many pools spend long periods in which the blended profile of the LP token beats either leg held in isolation, particularly when swap volume and gauge incentives are added to the equation.

3.4 Orderbook

Pendle pairs its AMM with an Orderbook so that participants can post limit orders and offers at exact implied yields. Where the AMM provides continuous depth, the order book supplies price precision, supplements liquidity when the curve moves out of range, and gives passive market-makers a tool for quoting.

3.4.1 Order anatomy

A limit order is an off-chain signed message that encodes:

- a salt for uniqueness, an expiry timestamp, and a nonce for blanket cancellation;
- the trade type: `SY for PT`, `PT for SY`, `SY for YT`, or `YT for SY`;
- the input or output token address, the YT series reference, and the maker and receiver addresses;
- `makingAmount`, the quantity the maker is willing to trade;
- `lnImpliedRate`, the natural logarithm of the implied APY being quoted, scaled by 1e18;
- `failSafeRate`, a worst-case SY-to-asset conversion rate that must hold when the order is settled.

By quoting price as the log of the implied rate, makers speak in the same context that the AMM uses internally. That alignment ensures the book and the pool move the market's forward yield in step rather than in conflict.

3.4.2 Posting and maintenance

Creating an order costs no gas. The signed payload is simply uploaded to a public relay or liquidity server. Cancelling is on-chain and can be done in two ways: calling `cancelSingle` with the original order fields, or incrementing the maker's global nonce to revoke every older order at once. Gas for maintenance is therefore paid only when necessary, an important feature for high-fee chains like ETH mainnet.

3.4.3 Filling orders

A taker wishing to execute bundles the chosen order payloads into the `fill` function along with:

- `maxTaking`, the highest amount of input tokens the taker is willing to transfer;
- a receiver address for the output tokens;
- optional opaque data for the callback.

During settlement the contract:

1. Verifies each signature and expiry.
2. Checks that `lnImpliedRate` is still favourable after accounting for the pool's current TWAP.
3. Transfers or mints the required tokens through the Pendle router, respecting the `failSafeRate` guard.
4. Pays the maker and receiver and records the fill so the remaining amount can no longer be reused.

The order book charges no maker fee. A taker pays the same effective fee as an AMM swap because the router routes any residual between the two sources to minimize price impact.

3.4.4 Callback mechanism and arbitrage

The `fill` call can include a target contract and calldata. Immediately after the order tokens are provisionally transferred, the limit-order router invokes `limitRouterCallback(actualMaking, actualTaking, totalFee, data)`. Arbitrageurs use this hook to:

- flash-swap against the PT-SY pool to cover the taking side,
- route through other Pendle order books,
- hedge on external derivatives venues.

When the callback returns, the owed tokens must already reside at the router, otherwise the entire transaction reverts. Arbitrage is therefore gas-sponsor neutral and risk free for the maker and taker.

3.4.5 Interaction with the AMM

Before the router reaches the AMM it first consumes any limit orders that improve the trader's price. Only the unfilled remainder is swapped on the curve. This sequencing deepens liquidity exactly where makers want it and smooths the transition between discrete quotes and continuous depth. An on-chain arbitrage bot maintained by the protocol watches both venues; if the AMM and the book drift it nudges them back into alignment.

3.5 Oracle implementation (PT)

Pendle combines the oracle into the AMM itself. Every swap updates a cumulative number and anyone can later turn two observations into a manipulation-resistant TWAP. The method is completely on-chain, covers the full life of the pool, and requires no keeper.

3.5.1 Methodology

PT and SY appreciate together. What traders/protocols actually care about is the implied forward yield. Using price directly would bias the average if yield jumps strongly early in the window, because prices compound multiplicatively. The natural log removes that bias:

$$\begin{aligned}\texttt{lnImpliedRate} &= \ln\bigl(1 + \text{impliedYield}\bigr) \\ &= \ln\bigl(\tfrac{\text{PT\_value}}{\text{PT\_value+SY\_value}}\bigr) \times \text{yearsToMaturity}\end{aligned}$$

Additive changes in `lnImpliedRate` map one-for-one with additive changes in compounded return. Averaging the log therefore gives the geometric mean of the implied rate.

Each pool keeps a ring buffer of

```
Observation {
    uint32   blockTimestamp;
    uint216  lnImpliedRateCumulative;
    bool     initialized;
}
```

`lnImpliedRateCumulative` is simply the previous cumulative value plus `lnImpliedRate_now × seconds`.

The buffer size grows only when an integrator calls `increaseObservationsCardinalityNext`. Recommended cardinalities are 85 for a 15-minute window on Ethereum, 900 for the same window on Arbitrum and about 520 for the same window on HyperEVM. Once set, the pool costs no further gas to maintain.

Note: A safe rule of thumb for calculating the cardinality is:

$$\text{cardinalityRequired} = \frac{\text{windowSeconds}}{\text{avgBlockTime}} + \text{headroom}$$

where:

- **windowSeconds** is target TWAP span (900 s for the 15-min preset)
- **avgBlockTime** is mean block interval for the chain, in seconds
- **headroom** is extra slots (10–20 %) so occasional slow blocks do not exhaust the buffer

3.5.2 Deriving the oracle price

Suppose an integrator wants a thirty-minute TWAP as of block T. They call `consult(market, secondsAgo = 1800)`; the helper contract:

1. Locates the Observation recorded immediately before $T - 1800$ s (linear interpolation if necessary).
2. Reads the latest Observation written at time T.
3. Computes:

$$\text{lnMean} = \frac{\text{cum}_T - \text{cum}_{T-1800}}{1800}, \qquad \text{impliedRate} = e^{\text{lnMean}}$$

4. Converts the implied rate into a PT price:

$$\text{PT\_to\_asset} = \text{impliedRate}^{\text{timeToMaturity}/1\text{ year}}$$

`timeToMaturity` is derived from the series expiry and `block.timestamp`; the exponent collapses to 1 at maturity so the formula automatically returns par.

3.5.3 Manipulation resistance

Because the oracle reads the entire pool reserve after every swap, a would-be attacker must move the AMM price for the full duration of the window to budge the TWAP. The cost scales linearly with time and quadratically with depth, quickly negating any profit. A shorter window lowers gas costs for integrators; a longer one raises the attack cost. Protocols are free to choose any number up to nine days.

The additional guard-rails in the curve itself (PT share $\leq 96\,\%$, PT price < 1) cap the magnitude of any flash move, further reducing incentive.

3.5.4 Mathematics behind the oracle

Let

$$p(t) = \text{PT value share at time } t, \qquad y(t) = \text{years to maturity at time } t$$

Then:

$$\begin{aligned}\texttt{lnImpliedRate}(t) &= \frac{\ln\big(p/(1-p)\big)}{\text{scalarRoot}/y} + \text{rateAnchor}\\ &= \ln\big(p/(1-p)\big) \times \frac{y}{\text{scalarRoot}} + \text{rateAnchor}\end{aligned}$$

The oracle records:

$$L(t) = \int \texttt{lnImpliedRate}(\tau)\, d\tau$$

For a window $[t_0, t_1]$:

$$\begin{aligned}\text{Mean lnRate} &= \frac{L(t_1) - L(t_0)}{t_1 - t_0} \\ \text{impliedRate} &= e^{\text{Mean lnRate}} \\ \text{PT / asset} &= \text{impliedRate}^{\,y_{\text{now}}} \\ \text{asset / PT} &= \frac{1}{\text{PT / asset}}\end{aligned}$$

All operations are integer math on 256-bit words preventing any sort of overflow errors.

4 Key Risks associated with PT assets

In this section we are going to discuss the relevant types of risks associated with PT assets. We are keeping in mind that PT assets are going to be used as collateral on money markets and hence those protocols should be aware of the risks associated with these assets.

4.1 Smart Contract Risks

Pendle's core SY wrappers, PT–YT splitting, AMM, order book have been live since 2022 and audited repeatedly. Critical contracts are immutable and emergency pause is scoped to SY wrappers only. Yet no protocol can be deemed unbreakable, hence like any other blockchain protocol Pendle also encompasses smart contract risks.

4.1.1 Audits

Pendle has gotten multiple security audits from reputed auditing firms and high-ranking solo auditors. Their latest audits were done by Chainsecurity and Spearbit. Looking at the audit reports we could see that Pendle has never had a critical bug in all of their reports combined. Some of the earlier audits had a couple highs which were acknowledged and resolved by the team; in their latest reports there was only one high in Spearbit's and 0 highs in Chainsecurity's audit report. This gives us even more confidence in Pendle's smart contracts and we appreciate their high standards for smart contract security.

4.2 Underlying Asset Risk

PT assets share all the risks which the underlying asset possesses: this includes volatility risk, price manipulation risk, risks emerging from yield volatility and negative yields, etc. These risks are highly dependent on what sort of underlying asset is being used for creating a PT asset and we highly recommend protocols doing their own due diligence on these assets before onboarding their PT counterparts.

5 Case Study: kHYPE on Pendle

Here we present a case study for onboarding kHYPE's PT counterparts to a lending protocol. This will include an explainer on kHYPE and its underlying risks along with the risks associated with onboarding PT assets which have kHYPE as their underlying asset.

5.1 Understanding kHYPE

kHYPE is a LST protocol based on HyperEVM. Users deposit HYPE into Kinetiq's smart contracts, the protocol then mints kHYPE to their wallet based on the exchange rate at the time of minting. The contract batches everyone's deposits and, via Hyperliquid's new `CoreWriter` interface, delegates that HYPE to a rotating set of high-performing validators.

Rewards compound automatically on the L1, so the HYPE backing each kHYPE grows every minute, while the token balance stays fixed. The result is a single asset that earns native staking yield yet remains fully liquid; users can swap, lend and use kHYPE as collateral without waiting through the seven-day unbond period.

5.2 Yield accrual mechanism

kHYPE's yield accrual is powered by Hyperliquid's native auto-compounding staking rewards. When validators earn HYPE for securing the network, those rewards are instantly restaked on the core chain, no manual claiming or reinvestment required. Kinetiq's `StakingAccountant` tracks the total HYPE under management (including all accrued rewards) against the fixed supply of kHYPE. As a result, the exchange rate (HYPE per kHYPE) continuously rises.

5.3 Staking and Unstaking mechanism

5.3.1 Staking flow

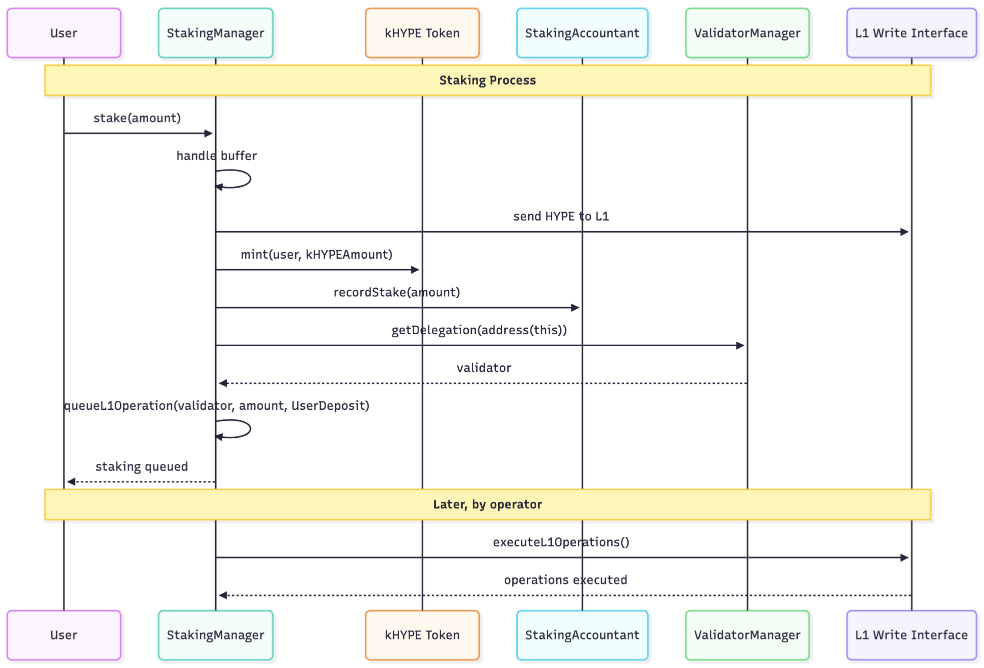

Figure 3: kHYPE staking flow diagram

The above figure demonstrates Kinetiq's Deposit Flow. The user's HYPE deposit is processed by the `StakingManager`, which mints kHYPE to the user and records the stake with the Accountant. The `ValidatorManager` suggests a validator and the `StakingManager` queues a delegation operation.

An off-chain operator later calls `executeL1Operations` to send the HYPE to the validator via Hyperliquid's `CoreWriter` interface. This design allows instant issuance of kHYPE while the actual delegation is handled asynchronously, ensuring a seamless user experience. There is a minimum limit of 5 HYPE which users can stake on kHYPE.

5.3.2 Unstaking flow

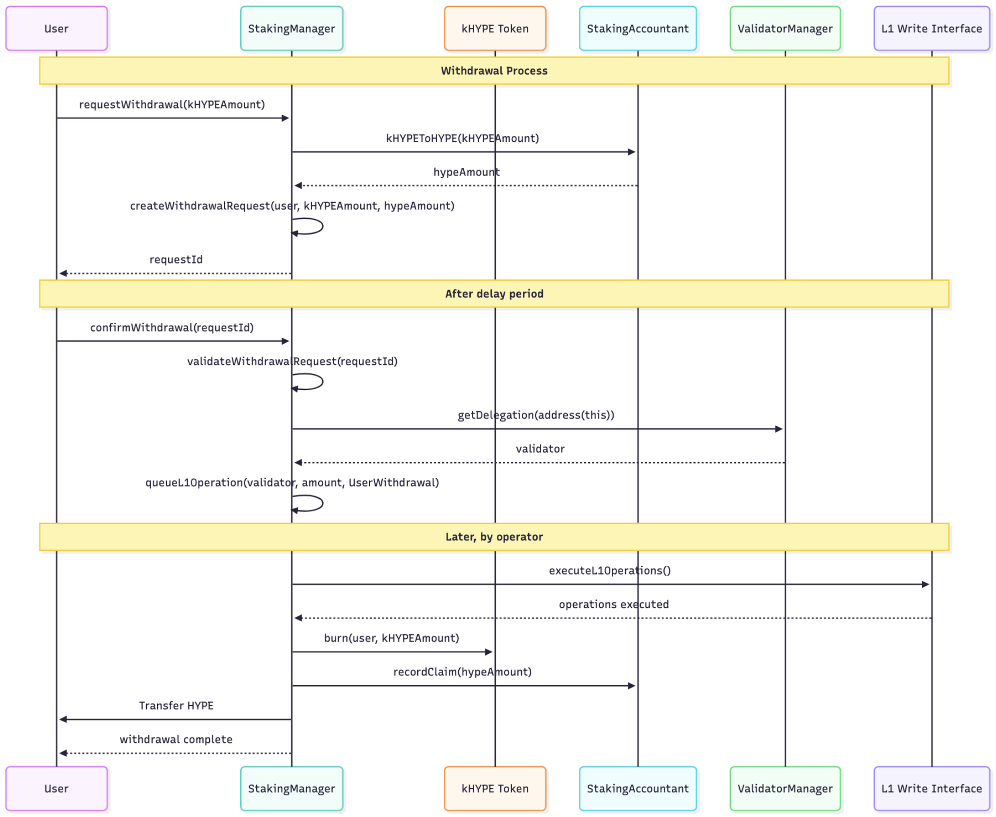

Figure 4: kHYPE unstaking flow diagram

In this figure we observe Kinetiq's Withdrawal Flow. The user first submits a withdrawal request for a given kHYPE amount. The `StakingManager` computes the equivalent HYPE and logs the request (returning a `requestId`). After the required delay (unbonding period), the user calls to confirm the withdrawal. The `StakingManager` then validates the request and asks the `ValidatorManager` which validator to withdraw from.

A withdrawal operation is queued and later executed by the operator via `CoreWriter`. Once the undelegation is executed on Hyperliquid, kHYPE is burned and the HYPE is released to the user. This process enforces the 7-day exit period while allowing the user to trade or hedge with their kHYPE during the waiting time. Unstaking via the Kinetiq dApp assesses a 0.10% fee on the user's withdrawal amount.

5.4 Staking on Hyperliquid vs Ethereum

Hyperliquid uses a delegated Proof-of-Stake model (HyperBFT) where HYPE token holders delegate to validators and earn rewards. This differs from Ethereum's staking in several ways:

- **Delegation vs Self-Validation**: In Hyperliquid, any HYPE holder can delegate tokens to professional validators, whereas Ethereum requires users to run a validator node or use pooled solutions (there is no native delegation).
- **Activation and Unbonding**: Staking activation on Hyperliquid is immediate (no queue). Unstaking, however, incurs an unstaking period of 7 days before tokens can be withdrawn. By contrast, Ethereum's exit process involves a minimum ∼1 day withdrawal period (and potentially a much longer queue under high exit demand).
- **Reward Distribution**: Hyperliquid staking rewards are auto-compounded and accrue continuously on the validator (rewards every minute, auto-restaked). Users' staked HYPE effectively grows within the protocol without manual intervention. On Ethereum, rewards accumulate and must be withdrawn (or re-delegated) manually; they are not auto-compounded by default.
- **Slashing**: Uniquely, Hyperliquid currently does not enforce slashing penalties for validator misbehavior. Validators can be jailed (temporarily disabled from block production) but not slashed. This differs with Ethereum, where validators can be slashed for serious faults (e.g. double-signing) resulting in loss of stake. The absence of slashing on Hyperliquid means stakers don't face direct loss of principal from validator faults, a significant reduction in risk.

5.5 Associated Risks

5.5.1 Market & Volatility Risk

kHYPE inherits the price volatility of its underlying asset, HYPE. HYPE has shown significant price movement historically. There have been multiple very sharp drops, for example the recent drop that happened during the escalation of Iran–Israel war, from 16 June to 22 June HYPE dropped almost 30 %.

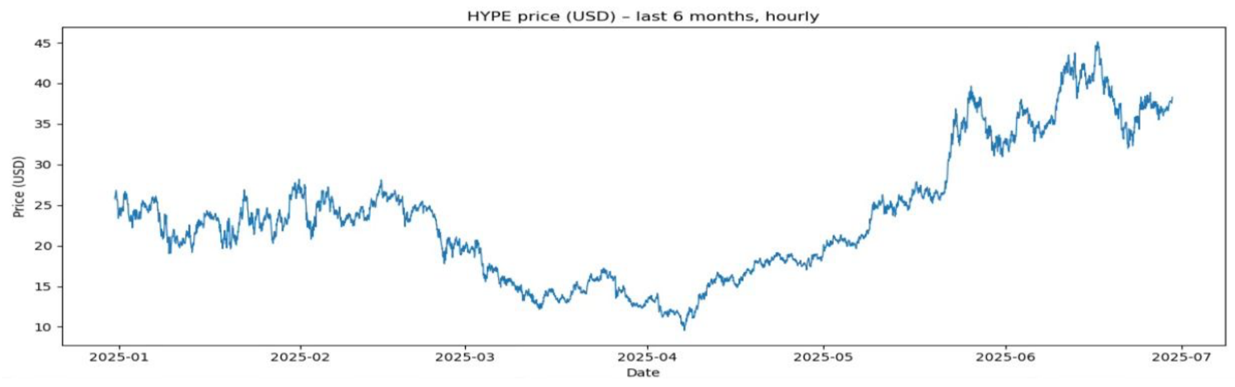

Figure 5: HYPE price (USD) – last 6 months, hourly

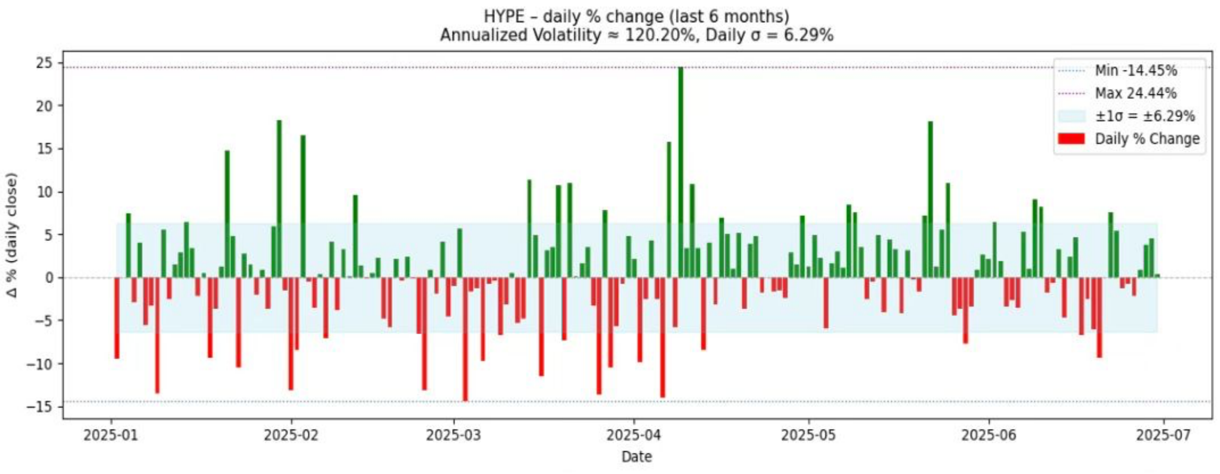

Figure 6: HYPE daily % change (last 6 months)

5.5.2 Depeg Risk

Depeg here refers to the risk that kHYPE's market price deviates from the underlying HYPE value (1 kHYPE should ideally trade for > 1 HYPE). Several factors can cause a depeg:

- **Illiquidity**: If kHYPE trading markets are thin, even modest sell pressure can drive its price down relative to HYPE. An attacker or panicked seller could push kHYPE price down on DEXs by dumping a large amount, especially if buyers are scarce. This liquidity risk is highest in the early days of kHYPE and improves as adoption grows.
- **Mass Unstaking Demand**: If a large portion of kHYPE holders simultaneously want out (due to some reason), kHYPE could trade at a discount. Those who don't want to wait 7 days might accept, say 0.95 HYPE per kHYPE. We saw analogous scenarios with stETH during the 2022 crypto credit crisis, stETH traded up to $\sim 5\%$ below ETH because many needed immediate exit.
- **Staking Reward Dynamics**: The staking APY on Hyperliquid is relatively modest ($\sim$ 2.1% annual at current stake levels). This means the "yield incentive" to hold kHYPE vs just HYPE is not huge. With 2 % APY, a week's worth of yield is $\sim 0.04\%$. That's roughly the maximum premium rational buyers might pay for kHYPE (since one-week unbond wait costs $\sim 0.04\%$ in missed yield). It also means that if kHYPE trades even 1 % below HYPE, that's equivalent to $\sim$ 6 months of yield, a significant deviation.

5.5.3 Smart Contract Risk

Using Kinetiq means trusting the security of its smart contracts. Any bug or exploit in the kHYPE contract or related code could potentially put user funds at risk. This is a general risk with DeFi protocols. A contract exploit could affect liquidity pools or the accounting of staked funds. Kinetiq's contracts have gone through multiple audits, but users should be aware of this technical risk whenever locking assets into a protocol.

Audits kHYPE went through several audits and public audit contests:

Auditor	**Dates (2025)**	**Key findings & fixes**
Pashov Audit Group[1]	Jan – Feb	22 issues (8 High, 6 Med, 8 Low). Highs included unsafe type-casts that could lock funds (H-05) and an early exchange-rate mis-mint bug. All of them were resolved.
Zenith Security[2]	Feb – March	High: 2; Medium: 6; Low: 3. All of the issues have been resolved.
Spearbit[3]	19–29 March	High: 2; Medium: 7; Low: 5; Informational: 16; Gas Optimizations: 8. 27 findings were fixed and 11 were acknowledged.
Code4rena audit contest[4]	8 – 17 Apr	High: 3; Medium: 5; Low/QoL: 17.

5.5.4 Stakehub Oracle Manipulation Risk

Kinetiq's Oracle System Risks: Kinetiq's use of an `OracleManager` and external adapters introduces novel oracle-related risks:

- **Malicious or Failed Oracle Adapter**: If one of the authorized oracle providers (say an off-chain server run by Kinetiq or a partner) is compromised, it could feed incorrect data. However, Kinetiq mitigates this by requiring multiple oracle reports and setting a minimum threshold (quorum). They also allow disabling a faulty oracle quickly via `setOracleActive(address,false)`[5].

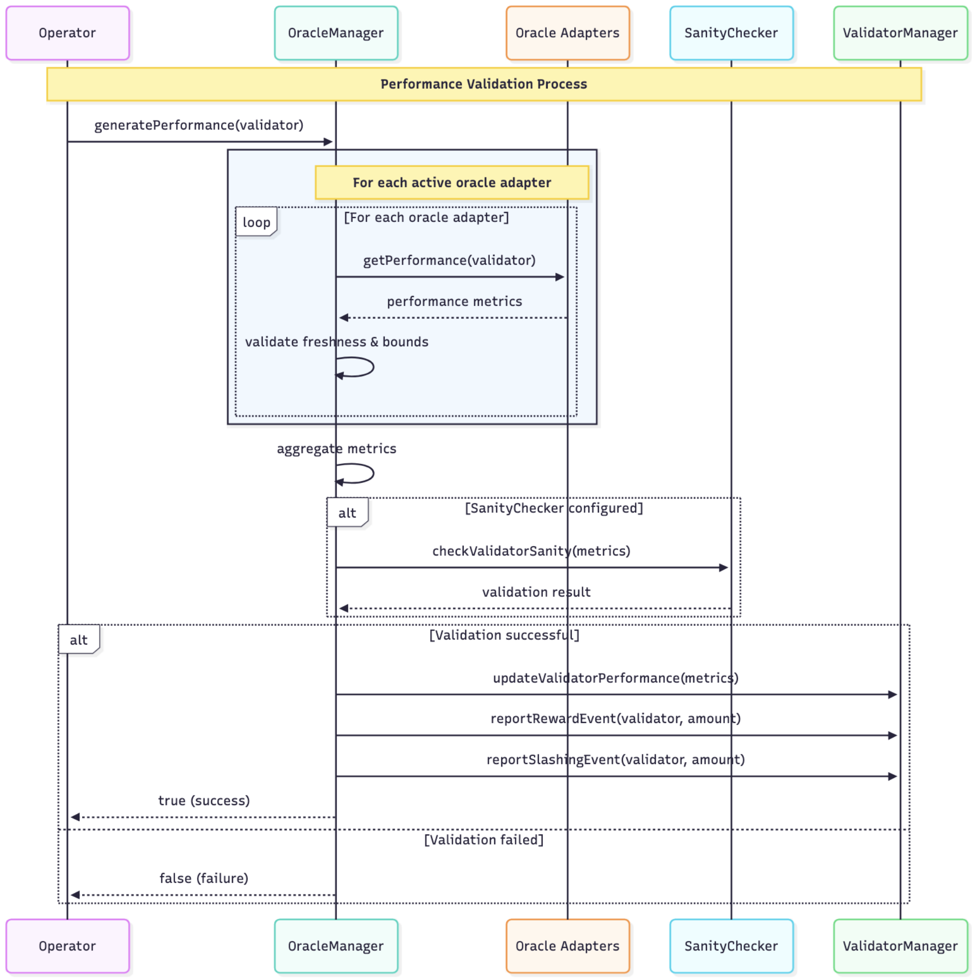
Operator
OracleManager
Oracle Adapters
SanityChecker
ValidatorManager
Performance Validation Process
generatePerformance(validator)
For each active oracle adapter
loop
[For each oracle adapter]
getPerformance(validator)
performance metrics
validate freshness & bounds
aggregate metrics
alt
[SanityChecker configured]
checkValidatorSanity(metrics)
validation result
alt
[Validation successful]
updateValidatorPerformance(metrics)
reportRewardEvent(validator, amount)
reportSlashingEvent(validator, amount)
true (success)
[Validation failed]
false (failure)
Operator
OracleManager
Oracle Adapters
SanityChecker
ValidatorManager

Figure 7: Kinetiq's Oracle Reporting Flow

This diagram illustrates Kinetiq's Oracle Reporting Flow. Kinetiq utilizes multiple independent Oracle Adapters to gather validator performance data, with an on-chain `OracleManager` aggregating the metrics and a `SanityChecker` validating the results. This multi-source oracle design helps protect against faulty or malicious data feeds, ensuring that stake allocations and reward distributions are based on reliable information.

5.5.5 Slashing/Validator Risk

In proof-of-stake networks, validators can be slashed for malicious behavior or extended downtime. Hyperliquid to date has no automatic slashing implemented; under-performing validators are instead jailed (temporarily removed) without slashing. This greatly mitigates the slashing risk at present. Kinetiq's StakeHub is designed to minimize this risk by delegating only to high-performance, reputable validators.

5.5.6 Network/Protocol Risk

Since kHYPE is tightly linked to the Hyperliquid network, any systemic issues with Hyperliquid could affect kHYPE holders. For instance, if Hyperliquid's staking APR drops significantly, the rewards for LST holders drop as well. Conversely, if Hyperliquid governance changes staking parameters or if a large portion of HYPE is suddenly unstaked, it could impact yields or redemption times. Additionally, if Hyperliquid's core consensus had an incident or needed a halt, kHYPE would be directly impacted (though this risk is quite low given Hyperliquid's robust design). Essentially, kHYPE inherits the risks of the underlying PoS system, including any future changes to slashing, reward rates, or lock-up rules.

5.6 kHYPE as a PT asset

kHYPE presents a very strong case for itself to be listed on Pendle as an asset. It satisfies all four basic requirements of being a GYGP asset and is currently the largest and fastest-growing LST on HyperEVM. kHYPE has a fixed and predictable yield source with negligible yield volatility and no chances of a negative yield. It has ample liquidity and is in high demand among users; naturally it makes sense for Pendle to incorporate kHYPE after it launches on HyperEVM.

Among all the above-listed risks a PT-kHYPE will also contain risks associated with PT assets, for example the security risks associated with Pendle and the liquidity risks associated with the PT asset itself. Keeping these factors in mind we give parameter recommendations for a lending protocol to list PT-kHYPE.

When setting parameters like LTVs for a PT asset we need to keep in consideration that as the asset matures it gets less risky, as PT starts to converge into the underlying asset. Hence the recommendations for these parameters are to be updated frequently in order to keep them aligned with market dynamics. We recommend updating these parameters every 7 days in order to avoid stale parameters which no longer correctly reflect the asset's risk profile.

5.6.1 Oracle configuration

We recommend using the standard Linear Discount Oracle which is used by other lending protocols such as Morpho as it reflects the correct pricing properties of kHYPE as PT collateral. We recommend using max of yield range i.e. 21.5% as the discount rate for the oracle pricing.

This linear discount oracle feed will be in PT-kHYPE/HYPE terms. In order to price in the risks associated with kHYPE we recommend using a multiplier of HYPE/kHYPE market price feed from Redstone(`0x3519B2f175D22a4dFA0595c291fEfe0945F0656d`).

Afterwards, in order to convert this feed into USD terms for protocol use we recommend using Redstone's kHYPE/USD market price feed(`0xF2448DC04B1d3f1767D6f7C03da8a3933bdDD697`).

5.6.2 LTV recommendations

We recommend using PT-kHYPE in correlated-asset pools or *e-mode* (Aave terminology). This insulates the system from HYPE's volatility. Examples include the HYPE vault in Felix's Vanilla Markets and any frontier HYPE vault launched later for riskier exposure.

The recommended LTV is 86%, in line with industry standards for correlated PT assets and with our LTV guidance for early-stage LSTs on HyperEVM.

5.6.3 Risk containment

To ensure that PT assets don't lead to overexposure in core vaults we recommend supply caps calibrated so PT assets remain liquidatable even during downturn events or exploits in the underlying protocol.

For the launch we recommend Felix to use a conservative cap of $5M. After the market risk preference is understood by the initial positions a better suited asset cap can be implemented to allow for more deposits and capital efficiency.

[1] `https://github.com/code-423n4/2025-04-kinetiq/blob/main/audits/kinetiq-pashov.pdf`

[2] `https://github.com/code-423n4/2025-04-kinetiq/blob/main/audits/kinetiq-zenith.pdf`

[3] `https://cantina.xyz/portfolio/3086102a-674e-4c82-9f1a-9c66dc13fdc2`

[4] `https://code4rena.com/reports/2025-04-kinetiq`

[5] `https://github.com/code-423n4/2025-04-kinetiq/blob/7f29c917c09341672e73be2f7917edf920ea2adb/src/OracleManager.sol#L114C1-L118C6`