WHITEPAPER · V1

PYRAMID.RIP

97% of launches die. Read the graves.

ABSTRACT

pyramid.rip is a launchpad that starts from the death rate. Every token launched through the Pons V2 factory on Robinhood Chain is indexed here — alive, abandoned or graduated — read live from the chain, with no backend. Connect a wallet and the same scan finds the creator fees your dead launches earned and never claimed. Launch a new one and the defaults come from what survived, with the base rate for your configuration printed above the signing prompt. $PYR accrues nothing on day one: no presale, no allocation, and the only mechanism that can ever route value to it burns what routed launches actually pay in creator fees. What is measured, what is built, and what is not promised, in that order.

VERSION	CHAIN	VENUE
1.0	Robinhood Chain · 4663	Pons V2

01

The base rate — what 7,685 launches on this factory actually did

Start with the denominator. PonsV2LaunchFactory, at 0x7eD598BcEf8bd9Edd8C97A195C6d13f40801EC7e, has minted 7,685 curves in the window we indexed, and Robinhood Chain adds between 2,000 and 5,500 more a day. About 97% of them never took a real bid. Not underperformed. Deployed, the reserve never moved past the deployer's own buy, and the contract is still there answering calls nobody makes.

Graduation is the only unambiguous survival event on this factory. A curve that reaches its threshold closes, and its liquidity migrates into a Uniswap v4 position with no withdrawal path. Everything short of that is a token still sitting on a curve, whatever its chart looked like for an afternoon. Across every denominator measured, the best graduation rate on this factory is 1.65%, and two of the most-used denominators have never produced a single graduation.

Three more facts out of the same scan. 43% of launches pass creatorTaxBps = 0, so even the ones that trade pay their deployer nothing. Roughly 28% of all launches come from five addresses. And a substantial balance of creator fees sits unclaimed in the Pons escrow, earned by launches whose deployers never came back for it.

The windows differ, and we would rather say so than smooth it over: 7,685 is this factory's launch log, while the per-pair rates are measured across each pair's full history. Neither requires trusting us. An eth_getLogs over the factory, then a Multicall3 batch of curve reads, reproduces both.

That is the number this site is built on. Every chapter after it is either a consequence of the base rate or a wager against it.

> Deployed, the reserve never moved past the deployer's own buy, and the contract is still there answering calls nobody makes.

02

The machinery we did not build — Pons V2, the launch, the graduation, the locked position

pyramid.rip does not deploy tokens. Pons does. Every launch on this site is a direct call to the factory — no wrapper of ours, no proxy, no custody step between your wallet and the contract. We wrote the form and the index. The machinery underneath is somebody else's, and it already carries every grave in this document.

The call is launchToken(params, launchConfigId, pairToken, snipeTaxExemptions), payable with launchFee(), currently 0.0005 ETH. params carries the metadata plus creatorFeeRecipient, creatorTaxBps, buybackEnabled and expectedEconomics — a hash returned by previewLaunchEconomics(launchConfigId, pairToken) and handed back with the transaction, so if the terms move between the quote and the block it reverts instead of launching on terms you never read. pairToken sets the denominator: native ETH, or one of the chain's tokenized equities.

The curve is readable at any block through getReserves(), realQuoteReserve() and readyToGraduate(). At 4.2 ETH of real reserve — graduationThreshold, emitted in TokenLaunched — it closes, and the liquidity mints as a Uniswap v4 position held by the locker. There is no withdrawal path. The creator has no more access to it than a stranger does. That locker is Pons's contract, not ours: read locker() off the factory and verify the absence of a withdraw yourself, because if it is wrong we cannot fix it.

A dev buy goes through PonsV2LaunchAndBuy.launchAndBuy, which puts the deploy and the first purchase in one transaction, so the opening block is not a sniper's. The launcher is exempt from the snipe tax by construction. The 32 slots in the snipeTaxExemptions array belong to the launcher, to fill or leave empty, and pyramid.rip never writes an address of its own into that array. Chapter 9 is about why that sentence is load-bearing.

The factory has an owner who can call setLaunchEnabled and setLaunchFee. If launching is turned off or the fee moves, this site moves with it. We read that contract. We are not a party to it.

> We wrote the form and the index; everything in the launch path is somebody else's contract, and if it is wrong we cannot fix it.

03

What a dead token leaves behind

A token on Pons is never deleted. The curve stays deployed and still answers calls. The only thing that changes is that nobody calls it. So every launch off this factory leaves the same record, and the record is readable forever.

Deployer — the address in the launch event. Five addresses produce more than a quarter of everything here, so the deployer is the first thing that tells you whether you are looking at a person or a loop.

Description — the free-text string passed at launch and stored on chain. That is the epitaph. It is unverified: the chain records what was written, not whether it was true. Most are two words.

pairToken — what the curve was priced in. Native ETH for most of them, a tokenized equity for the rest, and the choice decides more than it looks like it decides.

Curve reserves — current, and the high-water mark reached. Since graduation is a fixed threshold, the high-water mark is the honest measure of how close a token got. For nearly all of them, the answer is that it never got anywhere.

Last trade — the block of the most recent buy or sell in the curve's own logs. The time since that block is how long the thing has been dead.

creatorTaxBps and creatorFeeRecipient — what the creator charged, and where it went. At zero, nothing accrued and there is nothing to claim, permanently. Above zero, it accrued into the Pons escrow and waited.

All of it is a read. pyramid.rip batches these through Multicall3 and stores nothing. It cannot claim for you: the claim goes from your address. If this site disappears, the graves do not.

> The code runs forever, the curve still answers, nobody calls it.

04

The Necropolis — indexing every launch with no backend and no database

There is no backend. There is no database. The graveyard is assembled in your browser from three round trips to an RPC endpoint you can swap for your own node.

One: eth_getLogs against the factory, filtered to TokenLaunched. Each log already carries token, curve, deployer, pairToken and graduationThreshold — the occupant, the plot, the gravedigger, the currency, and the depth required. Blocks here are roughly 100ms, so the default 120,000-block window is about three hours of burials, which at the observed rate is 250 to 700 launches.

Two: one Multicall3 aggregate3 to 0xcA11bde05977b3631167028862bE2a173976CA11, asking every token in the window for description(). That is the epitaph. It was written by the deployer at launch, it is verified by nobody including us, and we quote it exactly as found.

Three: a second aggregate3 for name(), symbol(), logo(), and on the curve realQuoteReserve() and readyToGraduate() — what was actually paid in, and whether the threshold was crossed. Alive, abandoned and graduated fall out of those two values plus the block of the last trade.

The batching is the whole trick. Reading 150 descriptions one call at a time takes the better part of a minute on a public RPC. Batched, that leg is about 250ms and the feed loads in roughly two seconds instead of fifteen.

The limits belong in the same breath as the mechanism. The index is only as honest as the RPC that answers it, so the endpoint is configurable and every row links to the raw call. History deeper than the log window needs an archival node and a paginated scan — slower, not blocked. And a token that never traded has no last trade. An empty grave still counts as a grave.

The same scan feeds the board: today’s body count, graduation rate by denominator, and the wallets with the most fees still in the ground.

> The index is only as honest as the RPC that answers it, which is why the endpoint is yours to change and every row links to the raw call.

05

The Excavation — how creator fees got stranded, and how to claim yours

Creator fees on Pons are pull, not push. A curve accrues creatorTaxBps to whatever address was written into creatorFeeRecipient at launch, and it stays in escrow until someone signs for it. Nothing arrives. Nothing notifies you. Across the launches off this factory, roughly 95 ETH has never been signed for.

Four things stranded it. Most launches set the tax to zero and earned nothing, so the whole backlog sits under the minority that charged. Almost nothing takes a real bid, and the handful that did usually traded after the creator had stopped looking. creatorFeeRecipient is fixed at launch, and bots point it at hot wallets whose keys are in some cases gone, which makes that share unreachable by anyone, permanently. And no interface ever showed the rest, because front ends index launches that are alive, and a dead token has no page.

The dig is mechanical. Connect a wallet, and the site filters TokenLaunched on its indexed deployer topic in one eth_getLogs against the factory, pulls the curve address out of each log, and batches the fee reads through Multicall3 — the same round trips the necropolis already makes. Each curve you deployed comes back with the amount still owed to its recorded recipient. The claim is a transaction you sign against that curve's escrow, and you should read the calldata before you sign it. pyramid.rip never holds the funds, takes no cut, and holds no key that could move them. Where creatorFeeRecipient is not an address you control, the row still renders and the button is dead.

Say the limit out loud: this is a backlog, not an income. Once your wallet is cleared, it stays cleared.

> Nobody took your fees. They were left where you stopped looking.

06

Burial rites — the four parameters that move the odds

Four fields in the launchToken params tuple move the odds. None of them make a token live.

creatorTaxBps, a uint16 in that tuple, is the tax you take on every trade of your own curve, paid to creatorFeeRecipient. Zero is the most common setting on this factory, and it is almost always the result of not deciding rather than of deciding. The burial form defaults to a non-zero value. The trade-off is real and it is charged to your buyers: a visible tax deters them, and it should. But at zero, a token that trades and then dies pays you exactly nothing, forever, and that is the outcome most deployers have already chosen without noticing.

pairToken is the address the curve is denominated in — the zero address for native ETH, or any tokenized equity on this chain. The form orders that field by measured graduation rate and prints the sample size beside each option. Chapter 7 is the table.

A dev buy goes through PonsV2LaunchAndBuy.launchAndBuy, which creates the curve and executes the first buy in one transaction. The snipe tax makes the opening seconds punitive for anyone who is not the launcher; atomicity, not goodwill, is what keeps that first block out of a bot’s hands. The form wires a dev buy for the ETH pair only.

snipeTaxExemptions is an address[] capped at 32. The form passes your own address and leaves the other 31 empty. They are yours to fill or to waste. Staking $PYR does not put anyone in that array. There is no list, no ranking, and no privileged buyer.

> Zero creator tax is almost never a decision; it is the shape of not making one.

07

Choosing a pair — graduation rates by denominator

pairToken decides who is allowed to buy you. A curve priced in ETH takes bids from anyone holding the chain's gas token. A curve priced in TSLA takes bids only from wallets that already hold tokenized TSLA and are willing to spend it on a memecoin instead of on the equity they bought it for. Every equity denominator is a filter applied to your buyer set before the first block.

Measured across the launches off this factory: ETH, 4,300 launches, 0.92% graduated. DJT, 289 at 1.65%. RDDT, 230 at 1.61%. SPY, 103 at 1.20%. NVDA, 211 at 0.29%. TSLA, zero out of 1,288. AAPL, zero out of 1,189.

Read the top of that table honestly. DJT's 1.65% is about five tokens. At ETH's rate you would have expected three, so the gap is noise. SPY's 1.20% is one token. Do not pick a denominator because a one-graduation sample outranked a forty-graduation one.

The bottom of the table is a different kind of fact. At ETH's rate, 1,288 TSLA-paired launches should have produced roughly twelve graduations, and 1,189 AAPL-paired roughly eleven. Both produced none. Runs that long ending at zero are not variance. They are the denominator.

So the burial form orders the field by measured rate, prints the sample size, and greys the options too small to mean anything. Pick TSLA anyway and it shows 0 of 1,288 above the signing prompt. That is a base rate for other people's launches. It is not a forecast for yours, and nothing on this site will pretend otherwise.

Verify it: scan the factory's launch logs for pairToken, batch the curve state through Multicall3, count how many per denominator reached graduation. Three round trips, the same ones the necropolis makes.

> At ETH's rate, 1,288 TSLA-paired launches should have produced about twelve graduations. They produced none.

08

The Tomb — creatorFeeRecipient as a contract, the 70/30 split, and the burn

Pons lets a launcher set creatorFeeRecipient to any address, including a contract. The Tomb is that contract. A launch that names it sends its creator tax there instead of to a wallet, and the Tomb splits every arriving payment: 70% streamed to the address the launcher registered at launch, claimable by them at any time, and 30% held in the Tomb's balance. The split is fixed at deployment. There is no owner function to change it, no pause, and no withdrawal path to a team address. The launcher keeps the majority of their own tax, and nothing is taken from buyers that the curve was not already taking.

The held 30% leaves by one path only. sweep() is public and unpermissioned. Anyone can call it. It market-buys $PYR with the accumulated balance on $PYR's own Pons pool, sends what it bought to 0x000...dEaD, and pays the caller a fixed cut of the swept amount for the gas and the trouble. There is no keeper, no epoch, no multisig, and no schedule anyone has to be trusted to keep. If the Tomb holds a balance and nobody calls sweep(), the balance sits there until someone does, and the caller's cut exists so that a bot will. Every buy and every burn is a transaction anyone can index.

The arithmetic is not favourable, and it is the reason this chapter is short. Almost nothing launched through this factory takes a real bid, and nearly half of what is launched charges no creator tax at all, so the tax on a typical launch rounds to zero. The Tomb's revenue is the thin slice that graduates, times whatever share of the chain's daily flow this site captures. Fees have to be earned before anything is burned. If that share stays small, sweep() reverts on an empty balance indefinitely and nothing is ever burned. That is the base case, not the downside.

> Fees have to be earned before anything is burned.

09

The Thirty-Two — the exemption slots $PYR does not get

Pons grants every launch 32 snipe-tax exemption slots. The tax opens at 9900 bps and decays over roughly three seconds, and any address in the snipeTaxExemptions array pays none of it. The launcher is exempt already. The other 32 slots exist whether they are filled or not, and unused they cost nobody anything.

The obvious move was to hand them to the token. Keep an on-chain list of 32 addresses ranked by staked $PYR, pass it as snipeTaxExemptions on every launch from this site, and holders buy the opening seconds untaxed. It works on day one. It costs the launcher nothing. It would have given $PYR something to point at immediately.

We cut it, and the reason fits in one sentence: the tax those 32 addresses skip is paid by everyone else buying in the same three seconds, so the only live utility of the token would have been the right to be the sniper the rest of this site exists to protect launchers from. A launchpad whose holders get the first block is worse for every buyer of every launch on it. On a site called pyramid.rip, that is not a mechanic. It is a confession.

So the slots go to the launcher. The burial form exposes snipeTaxExemptions as a plain field: paste up to 32 addresses — a second wallet, a market maker, or nothing at all — and the array enters the calldata unchanged. This site inserts no address you did not type, and you can prove it by decoding your own launch transaction.

That leaves $PYR with no utility on day one. Not nearly none. None. The Tomb accrues only when a routed launch books a real creator fee, and almost no launch ever does.

> A launchpad whose token holders get the first block is worse for every buyer of every launch on it.

10

What is not promised — supply, failure modes, verification

$PYR has no presale, no team allocation, and no allocation to this site. It launches on the same Pons curve as everything indexed here, at the same launch fee, with the dev buy atomic and published by transaction hash, and if it graduates it graduates into the same permanently locked position nobody can pull. It is subject to the base rate in chapter 1 exactly like every other row in the graveyard.

The failure modes, stated plainly. If pyramid.rip captures little of the chain's daily launch flow, no meaningful creator tax is ever routed, the Tomb's balance stays empty, and sweep() has nothing to buy or burn — indefinitely. If launchers keep setting zero creator tax, routed launches pay nothing even when they trade. If the base rate holds, the launches that route to the Tomb mostly die before they earn anything, because that is what launches on this factory do. The factory owner can disable launching or move the fee, and this site follows. The RPC that answers the index can lie, which is why the endpoint is yours to change. And nothing here can recover fees whose creatorFeeRecipient key is lost.

The two useful halves of this site are not defensible. The necropolis is public chain data, and any fork can re-derive it in a weekend. The excavation is a one-time extraction: once the stranded escrow is claimed, that reason to return is gone, and what is left is a leaderboard and a launch form on a chain where five addresses already produce a quarter of the flow.

Nothing in this document is a claim that $PYR will be worth more later, that anything will accrue to it on any timeline, or that a launch made here will survive. No yield is offered, no return is paid out of anyone's deposit, and the burn fires only on fees that were actually earned.

Every number printed here is a contract call. eth_getLogs over the factory for the launch set, Multicall3 batches for curve state and description, locker() and the absence of a withdraw for the graduation claim, and your own decoded calldata for the exemption array. Run them. If our numbers and yours disagree, ours are wrong.

> Nothing here is a claim that this token will be worth more later. The burn fires only on fees that were actually earned.