ROBINHOOD CHAIN · UNISWAP V4

# Sable Launchpad

Coins with a character. Anyone launches a coin on a bonding curve; at $6,000 raised it moves into Uniswap v4, from contracts nobody can change — and every coin carries a locked character that answers from real data, never from guesses.

| CHAIN | GRADUATION | FEES | CONTRACTS |
|---|---|---|---|
| Robinhood Chain · 4663<br>Arbitrum Orbit L2 | $6,000 raised<br>into Uniswap v4 | 2% per curve trade<br>1% of the raise at graduation | no owner · no admin key<br>no upgrade path |

VERSION
1.0 — 29 September 2026

COMPANION TO
Sable whitepaper v4.3

STATUS
Live · pre-audit

WEB
launchpad.buildsable.com

ABSTRACT

The **Sable Launchpad** opened to the public on 29 September 2026 at launchpad.buildsable.com, on **Robinhood Chain**. Anyone can launch a coin. It trades on a **bonding curve** — a contract that prices by a fixed formula — until **$6,000 has been raised** into it; then anyone may send the transaction that moves its liquidity into a **Uniswap v4** pool, where it is locked by construction. Three things set it apart. Every coin carries a **character** derived from its creator's own words and sealed into the contract, so it cannot be swapped after people buy; with **Self Aware Mode** on, that character can answer questions about its own curve, holders and trades from figures Sable actually measured. Every number on the site is a **reading** from Sable's own indexer, never a valuation, and a figure that could not be read prints as a dash, never a zero. And the contracts have **no owner, no admin key and no upgrade path**: the fees, the payees and the lock are properties of deployed code that anyone can read. This paper explains how a coin lives and graduates, where every fee goes, and — in its own section — what the launchpad does not claim.

CONTENTS



HOW TO READ THIS PAPER

It is written for three readers at once: **builders** who launch a coin, **traders** who buy and sell one, and the **community** that wants to know how the thing actually works. Each section opens in plain language and gets precise underneath; the contract names in `code` are the functions you can call yourself.

It is a companion to the main Sable whitepaper (v4.3), whose §15 summarises the same launchpad. Where a fact is stated here it is true of the deployed contracts and the production gateway on the date on the cover. **If this paper and the chain ever disagree, the chain is right** — every figure below can be checked by calling a contract's public getters or the public API.

# 01 Why another launchpad

The shape is familiar on purpose. What is different is what a coin is, what the numbers mean, and who can change the rules.

The bonding-curve launchpad is a shape the community already knows: a coin starts cheap, its price rises as people buy, and once enough capital has gone in it moves to an ordinary trading pool. That shape works, so the Sable Launchpad keeps it. Three things about it are changed, and each one is enforced by code rather than by a promise.

### A coin has a character

Every coin carries four traits — tone, stance, domain and verbosity — chosen from its creator's own name and description and sealed into its contract. With **Self Aware Mode** on, the coin can talk about itself: its curve progress, its holders, its trades. It answers only from figures Sable measured, and says so when it cannot see something.

### Readings, not valuations

Every price, chart, holder count and market figure is folded from trades Sable's own indexer read off the chain. Nothing is a quote Sable will honour, nothing is a rating, and a figure that could not be read is printed as a dash with its reason — never as a zero, and never as yesterday's number carried forward.

### Contracts nobody can change

The factory, every curve and the fee splitter have **no owner, no admin key and no upgrade path.** Supply, fee rates, payees and the shape of the curve are fixed when a contract is deployed. Changing any of them means deploying a new contract, in public, at a new address.

### The lock is a property

At graduation the liquidity becomes a Uniswap v4 position owned by the curve contract itself — which has no function that can withdraw it. Not the creator, not Sable. That is checkable by reading the contract's interface, not by trusting this sentence.

One more difference is a matter of voice rather than code. The launchpad says **"raised"** for the graduation threshold and never "market cap", because the two differ by several times and confusing them is the commonest way a launch figure misleads. It lists every coin without curation, and a listing is not an endorsement. And it prints what it cannot read as missing rather than inventing a figure to fill the gap.

# 02 The lifecycle of a coin

Create, trade on the curve, fill, graduate, trade on Uniswap. Every step that moves money is signed by a user's own wallet.

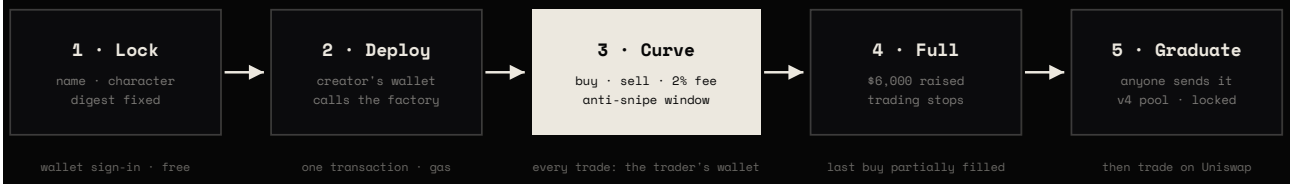


FIG. 1 — A COIN'S LIFE. SABLE RECORDS AND READS; THE CHAIN AND THE USERS' WALLETS DO EVERYTHING THAT MOVES MONEY.

01 **Lock.** On the Create page a creator writes a name, a ticker (2 to 10 letters and numbers), a description of at most two sentences and 160 characters, and optionally adds an image. The page derives the coin's character from those words (§06) and shows it. The creator signs in with their wallet — a signature, not a transaction — and presses **Launch token**. Sable canonicalises everything that defines the coin, hashes it, and records it as a draft. That digest is the coin's fingerprint.

02 **Deploy.** The next screen asks the creator's own wallet to call `create(…)` on the `SableCurveFactory`. The factory deploys two contracts: a plain fixed-supply ERC-20 holding all **1,000,000,000** tokens, and the coin's `SableCurve`, to which the whole supply is handed in the same call. The digest goes on-chain as the curve's `specHash()`. **Sable deploys nothing**: the creator pays the gas and the creator's address becomes the curve's creator payee.

03 **Curve.** Sable reads the chain, confirms the factory really made this curve and that it commits to the locked digest, and lists the coin. From then on anyone buys and sells against the curve from their own wallet. For the first five blocks, one address may spend only a limited amount (§03).

04 **Full.** When the capital raised reaches the coin's target, the last buy is filled only as far as the target, and the curve **stops trading**. Nothing else happens automatically.

05 **Graduate.** Anyone — a holder, a stranger, the creator — sends one transaction, `graduate()`. It takes the 1% graduation fee and moves everything the curve holds into a Uniswap v4 pool, as a position the curve owns and cannot withdraw (§04). The coin trades on Uniswap from then on.

Drafts — coins locked but never deployed — are not coins and are not shown on the board. A draft's page still opens by its link, because the record exists; nothing trades there.

WHAT IS PERMANENT

The name, ticker, description, image digest, the four traits and whether Self Aware Mode is on are all inside the digest the curve commits to. **Nobody can change them after launch** — not the creator, not Sable, not a support request. The token's supply is an `immutable` of its contract; it has no mint function, no owner, no pause and no blacklist.

# 03 The bonding curve

A constant-product formula over virtual reserves: the price rises as the coin is bought and falls as it is sold, by arithmetic anyone can reproduce.

**In plain words.** The curve is a vending machine with a price rule instead of a price tag. Every buy takes tokens out of it and puts money in, which raises the price of the next token; every sell does the opposite. The rule is fixed, public and the same for every coin launched from the factory. Nobody sets a price by hand.

## The formula

The curve keeps two numbers, `quoteReserve` and `baseReserve`, and holds their product constant (the invariant is `invariantK`, fixed at creation). They start from **virtual** values — reserves that exist only in the arithmetic, so the opening price is not zero:

```
virtualQuote  = graduationTarget / 3            // per coin, in the quote asset's units
virtualBase   = 750,000,000 tokens                // the same for every coin
spot price    = quoteReserve / baseReserve
buy:   quoteReserve += net quote in;  baseReserve = ceil(k / quoteReserve)
sell:  baseReserve  += tokens in;     quoteReserve = ceil(k / baseReserve)
raised        = quoteReserve - virtualQuote       // what graduation is measured against
```

Both directions round the new reserve **up**, so the product never falls: rounding lands in the curve's favour by at most one base unit a trade, and no sequence of trades — including adversarial buy-sell alternation — can take out more quote than was put in. The fee is split off the top of a buy and off the proceeds of a sell and forwarded in the same call; it never enters the reserves, which is why `raised` always equals the quote the curve actually holds (§07).

## What the shape means

Because the virtual quote reserve is always one third of the target, every coin has the same shape whatever it is paired with. On the deployed terms, when a coin reaches its target the curve holds four thirds of the target as its quote reserve and 187.5 million tokens as its base reserve. So:

| SOLD BY GRADUATION | STILL IN THE CURVE | MARGINAL PRICE AT TARGET |
|---|---|---|
| 562,500,000 tokens<br>56.25% of the supply | 437,500,000 tokens<br>these go into the pool | 16 x the opening price<br>before fees |

These are statements about the formula, not about what any coin will do. Nothing guarantees that any coin reaches its target.

The implied capitalisation at graduation — last price times the full supply — is several times the $6,000 raised. That multiple depends on the curve's terms, which is exactly why the launchpad never uses "market cap" to name the threshold, and quotes no fixed multiple.

## The opening window

For the first **five blocks** after a curve is created, one address may spend at most **one twentieth of the target** in total (`antiSnipeMaxQuotePerWallet`). It slows down a bot that tries to buy everything in the first moment. It is a speed bump, not a wall: someone willing to fund

many addresses gets past it, and the contract's own documentation says so, because a cap presented as protection is worse than none.

### Guarantees the curve gives a trader

- **Slippage floors.** `buy(quoteIn, minBaseOut)` and `sell(baseIn, minQuoteOut)` revert rather than deliver less than the floor. The site's default is 1%; 0.1%, 0.5% and 5% are offered.
- **Previews.** `previewBuy` and `previewSell` return exactly what a trade would cost and deliver, fee included. The site's "You receive" figure is read from them.
- **Exact graduation.** A buy that would carry the raise past the target is **partially filled**: on an ETH-paired coin the unused part is refunded in the same call, on a token-paired coin it is simply never taken. The raise lands on the target exactly.
- **Refused, not surprised.** An ETH buy must send exactly its amount; a token-paired buy that sends ETH is refused rather than stranding it; a quote token that takes a fee on transfer is caught by a shortfall check at the first buy.
- **One supply.** 1,000,000,000 tokens, 18 decimals, minted once to the curve. A transfer to the zero address is refused, because the supply is immutable and a "burn" that the supply still counts would be misleading.

# 04 Graduation

$6,000 raised, then one permissionless transaction moves the coin into a Uniswap v4 pool whose liquidity nobody can take out.

## The threshold is capital raised

A coin graduates when **$6,000 has been raised** into its curve — quote actually deposited, net of fees. **It is not a market cap**, and the launchpad never calls it one. The threshold is written into each curve as `graduationTarget` in the **quote asset's own units**: when a coin is created, the gateway converts $6,000 into ETH (or whatever the coin is paired with) at that day's rate, and that amount is what the contract measures. Afterwards the target's dollar value moves with the quote's price, so a coin may graduate above or below $6,000 in today's dollars. The site always shows the contract's own measure, and prints the target at today's rate.

## Graduation is not automatic

A buy never graduates a coin. When the raise reaches the target, `isTrading()` becomes false and the curve refuses every buy and sell. It then waits for someone to call `graduate()`, which is **permissionless**: the caller need not hold the coin and pays only gas. On the site, a full curve replaces its Buy button with **Graduate to Uniswap v4**. If two people press it, the second transaction fails with `AlreadyGraduated` and costs only gas. A graduation that reverts for any other reason leaves the curve exactly as it was, so it can simply be sent again.

## What the transaction does

01 Takes the **graduation fee**: 1% of the quote the curve holds, sent to the platform payee (the fee splitter, §07).

02 Builds the coin's pool key: the coin and its quote asset, the **1% fee tier**, tick spacing 200, **no hooks**.

03 **Opens the pool at the curve's own final price.** The curve's last marginal price is its quote reserve divided by its base reserve. The pool receives the raise less the fee, and exactly the number of tokens that makes the pool's opening price equal that final price — about 139 million on the deployed terms.

04 **Burns the rest of the unsold supply** — about 298 million tokens on the deployed terms. They are destroyed, lowering `totalSupply()`; nothing is kept that could later be sold into the pool. Emits `UnsoldBurned`.

05 Deposits the pool's side as a single **full-range position owned by the curve contract**, and emits `Graduated` and `LiquidityPermanentlyLocked`.

THE PRICE DOES NOT JUMP AT GRADUATION

Someone who buys just before a coin graduates pays the curve's last price, and the pool opens at that same price. A test on a fork of Robinhood Chain launches a coin on the deployed factory, fills it, graduates it into the real Uniswap v4 manager and checks the two prices agree to within one part in a million.

**One exception, stated plainly.** Coins created before 29 September 2026 were made by an earlier factory (`0xEE40d7eccC8Eba941195c968158C14224fc667E9`), which deposits every unsold token instead. Their pools open at roughly a third of the curve's final price. Contracts cannot be changed after deployment, so this applies to those coins for good — at the time of writing, one listed coin, $SABRE.

## Locked by construction

In Uniswap v4, adding liquidity, collecting fees and withdrawing are one call, `modifyLiquidity`, distinguished only by the sign of a number. The curve's only route to that call passes the liquidity as an unsigned 128-bit integer, which cannot be negative, and no function takes a delta from a caller. So there is **no argument anyone can supply, from anywhere, that withdraws the position**. The liquidity is not burned; it is owned by a contract that has no way to move it. A test pins the curve's complete external interface so that this cannot quietly change.

The pool's 1% trading fee accrues to that locked position. Anyone may call `collectFees()`, which calls `modifyLiquidity` with a delta of exactly zero — the form that returns earned fees and leaves the principal untouched — and sends them, in both of the pool's currencies, to the platform payee. It is refused before graduation, and a second call in a row simply finds nothing.

## Squatters, and what can still delay a graduation

Uniswap's `initialize` is permissionless, so a stranger who knows a coin's token address can create its pool first at a price of their choosing. Depositing at that price would hand them the difference; refusing alone would strand the coin's liquidity for the cost of their gas. The curve does neither. **The price of an empty v4 pool moves for free** — a swap against it exchanges nothing and walks the price to the swapper's limit — so before depositing, the curve swaps the empty pool back to its own price, under two guards: the swap must exchange nothing at all, and the price must land exactly on target.

What can still delay a graduation is real capital. If someone parks actual liquidity in the pool at the wrong price, realigning would mean trading the coin's money against it, and the curve refuses with `PoolNotEmpty`. That is a **delay, not a loss**: it costs the obstructor capital for as long as they keep it there and earns them nothing; a reverted graduation leaves the curve untouched; and the next call after the position is gone succeeds.

After graduation the coin wears a **GRADUATED** badge for two hours, appears under the Graduated tab for good, and its card's Buy button becomes **Trade on Uniswap**. The curve is closed permanently.

# 05 Pair with any token

One factory, the quote asset chosen per coin, and every figure shown in dollars.

A coin's **quote asset** is what people pay with and what its raise is counted in. The creator picks it at launch, and it is fixed for the life of the coin. One factory serves every pairing:

- **ETH**, the default — the chain's native currency.
- **USDG** — Paxos' Global Dollar on Robinhood Chain, a US-dollar token.
- **Any other ERC-20** — pasted by address. Its symbol and decimals are read off the chain, never typed. $SABLE, Sable's token on Robinhood Chain, is one example.

The factory checks only that a quote is a contract answering `decimals()`, not that it behaves — a rebasing, pausable or blacklisting token could strand a curve. **The site, not the contract, decides which quotes are offered.**

### Dollars, always, through USDG

Every price, raise and capitalisation on the site is shown in dollars, and on Robinhood Chain "dollars" means USDG. The gateway finds a quote asset's dollar rate in a fixed order and records, per trade, the rate it used:

| QUOTE ASSET | WHERE ITS DOLLAR RATE COMES FROM |
|---|---|
| **USDG** | Exactly one dollar. |
| **ETH** | The ETH/USDG Uniswap v4 pool, read on the chain; an operator-set fallback rate only when the pool cannot be read. |
| **$SABLE** | Its own pinned pool against ETH, times the ETH rate. |
| **Any other ERC-20** | The deepest standard Uniswap v4 pool (no hooks) it has against USDG, or else against WETH times the ETH rate. |
| **None of the above** | No rate. The figure is a dash. |

**A token Sable cannot price cannot be launched from the site**: the pairing picker says there is no dollar rate for it and offers no Launch button, because a coin whose figures could never print in dollars would break the rule above. On a buy, amounts are entered in dollars and converted to the quote at the published rate; what goes on the chain is always the quote-unit amount.

Anyone can call the factory directly with any target; Sable lists a coin only if its target is $6,000 at the current rate, within 10%, and its quote can be priced.

SABLE'S OWN TOKENS

**SABLE** is Sable's token on Robinhood Chain (`0xf7894d31D569e6330592D346ecfeFdf4257F3EC1`), and the one the launchpad's Buy $SABLE button trades. **SABL** is the earlier Sable token, on another chain. They are separate tokens with different symbols. **There is no migration, bridge or conversion between them, and none is planned** — anyone who says otherwise is running a scam. Neither token is required to use the launchpad, neither gates access, and neither pays yield or confers any claim.

# 06 Coins with a character

A character chosen from the creator's words, sealed into the contract, and allowed to speak only about what it can measure.

## Four traits, from fixed lists

| TONE | STANCE | DOMAIN | VERBOSITY |
|---|---|---|---|
| cryptic · deadpan · hostile · hype · plain · warm | defiant · earnest · ironic · serene | absurd · art · finance · gaming · music · nature · politics · science | terse · normal · florid |

Traits come from closed vocabularies rather than free text, for two reasons. Free text would be an instruction channel into the character — a creator could write "ignore your rules" and call it a personality. And a fixed vocabulary is something a reader can filter and compare. Each value maps to one fixed sentence in the character's instructions, which the coin's page shows under its Character tab.

## Derived, shown, then locked

The creator does not pick four traits from twenty-one options. The site reads the coin's **name and description** against small, published keyword lists — no model, no network, no randomness — and chooses tone, stance and domain from the words that match; verbosity comes from how much the creator wrote. Where nothing matches, the fallback is seeded from the name, so two unrelated coins do not all come out identical and each is still reproducible from its name alone. The site does not ask for the traits or show them before launch — the creator writes, and Sable chooses. The coin's page shows the committed character afterwards, and a developer launching through the API supplies the traits directly.

At launch the traits, the name, the description, the image digest and the skills are canonicalised and hashed into the launch digest, which the curve commits to on-chain as `specHash()`. **A character that could be edited after launch would be a costume; one inside the digest is a commitment.** A coin's image is committed the same way: its bytes are refused unless they hash to the digest that was locked.

## Self Aware Mode

One switch on the Create page. Off, the coin is mute. On, the coin gets a voice and five **skills** — permission to read specific live facts about itself:

| SKILL | WHAT THE COIN CAN READ |
|---|---|
| **curve.read** | Its curve: raised, progress toward the target, whether it is trading |
| **holders.read** | Its holders, as Sable's indexer counts them |
| **supply.read** | Its supply |
| **trades.read** | Its recent recorded trades |
| **calls.read** | Predictions its creator sealed before launch, and whether they have been revealed |

The coin is handed the figures the gateway actually measured and asked to narrate them. A skill that was not locked is named as unavailable rather than left out, because an omitted fact is exactly what a model fills in. Asked about something it cannot see, the coin says it

cannot see it. A quick launch from the Explore page has Self Aware Mode off; the full Create form is where it is switched on.

### The refusal floor

No character — however hostile, hyped or ironic — can move these rules. They are in the character's fixed instructions, enforced again by checks on every reply, and pinned by tests that attempt to override them:

- It never predicts, forecasts or hints at a future price, valuation or market cap.
- It never advises buying, selling or holding.
- It never claims to be a person, and says it is a character when asked.
- It does not speak for Sable, and its opinions are not Sable's.
- It answers **only in English**, because the checks that enforce the floor can only read English. A reply that is not plausibly English is replaced with a line saying so.

### The public room

Each coin has one public room. People post by signing a message with their wallet (a signature, not a transaction); ticking **Ask** invites the coin to answer in the room. Messages and replies are **stored in the clear, with no expiry, and public by intent**, because they were written to be read by strangers. A coin's reply carries the signed receipt of the inference that produced it; a reply that breaks the refusal floor is not stored. Sable does not moderate for taste — there is a length limit and a rate limit, and that is all. A separate, private way to talk to a coin exists through the API; what is sent there is returned once and stored nowhere.

### Who pays for the voice

A coin's voice is paid from **its own 0.5% trading fee**: the gateway reads the fee off the coin's curve and credits that coin's voice budget with it. A coin that trades can talk; a coin that stops trading runs down its budget and goes quiet, which is honest rather than broken. **Nobody can top a voice up** — not the creator, not a large holder, not Sable — so nobody can buy a louder character. A global daily ceiling, a per-message cap and a per-address rate limit apply on top. A coin can also be quiet because Sable's own inference account cannot pay for a reply at that moment; when a coin cannot answer, the room says which of these is the reason rather than failing silently.

# 07 Fees, and where they go

Three fees, all fixed in deployed contracts, all paid in the coin's quote asset, and split on-chain by a contract with no owner.

| WHEN | FEE | WHERE IT GOES |
|---|---|---|
| **Every buy or sell on a curve** | 2% | 1% to the platform payee · 0.5% to the coin's creator · 0.5% to the voice payee, credited to that coin's own voice budget |
| **At graduation, once** | 1% of the raise | The platform payee |
| **After graduation** | 1% pool tier | Accrues to the locked position; `collectFees()`, callable by anyone, sweeps it to the platform payee. The creator's and voice shares stop at graduation. |

On a buy the fee is taken from what the buyer spends; on a sell, from the proceeds. The three trade shares add up to the fee exactly — the last share takes the rounding remainder, so no dust is left behind. The curve's fee rates are immutables copied from the factory, and the contract refuses any configuration whose trade fees exceed 5% or whose graduation fee exceeds 5%; the deployed rates are the 2% and 1% above.

## The fee splitter

The platform payee is `SableFeeSplitter` (`0x6f830893D5209e32B0277396A1e16B52B159B248`). It divides whatever it receives between three fixed recipients at fixed shares:

| LAUNCHPAD | TREASURY | COMMUNITY BUYBACKS |
|---|---|---|
| 50% | 30% | 20% · buybacks of $SABLE |

Recipients and shares are immutables: there is no owner and no setter. `receive()` only accepts, so a curve's fee transfer can never fail because a recipient was unreachable. Anyone may call `distribute()` (for ETH) or `distributeToken(token)` (for an ERC-20); the split is a pure function of the balance, so who triggers it changes nothing. Shares are floored and the remainder goes to the last recipient, so the three parts always sum to the whole. Because a coin can be paired with any token, the splitter receives whichever assets coins are paired with, including the coin tokens themselves once graduated pools pay fees in both currencies.

## What Sable holds, and does not

Sable never takes custody of a buyer's or a creator's funds. A curve holds its own reserve; every launch, trade and graduation is a transaction the user's own wallet signs; and Sable's income arrives as the fee transfers above, in public, at the addresses above. The launchpad's figures for "fees taken" are computed from recorded trades times each curve's own fee rate as read off the chain — never assumed from a deployment script, and never presented as a treasury balance.

# 08 Readings, not valuations

Sable's own indexer reads the chain; every figure on the site is folded from what it recorded, and a figure it could not read is a dash.

## Sable's own indexer, no third party

- **It reads the chain itself.** The gateway reads every curve's `Bought`, `Sold` and `Graduated` events over JSON-RPC, keyed by transaction and log position so none is recorded twice. No external indexing service stands in between.
- **A twelve-block guard.** Nothing within twelve blocks of the chain head is recorded, so a chain reorganisation can never make a trade appear on the site and then vanish.
- **A two-second head lane.** Every two seconds the indexer checks the chain head and reads every watched curve's new events in one request, so a trade is recorded within seconds of its twelfth confirmation. A slower backfill catches up new coins from their deploy block.
- **Pushed live.** Each recorded trade, launch and graduation is pushed to open pages over server-sent events. When the push is unavailable the page falls back to polling, and the line under the chart says which mode it is in and how far behind the chain head the index is.
- **The curve cache follows the chain.** After any pass that records a trade or a graduation, the coin's curve is re-read, so the raise and the progress bar follow the chain rather than an old snapshot.

## What each figure is

| FIGURE | WHAT IT MEANS |
|---|---|
| **Price** | What the last buyer or seller paid per token, converted to dollars at the rate recorded with that trade. |
| **Mcap** | Last price times `totalSupply()`, both read. Printed only when both inputs were read; never used to name the graduation threshold. |
| **24h change** | Needs a price at both ends of the window. One trade is not a 0% change; it is no change figure at all. |
| **Holders** | Wallets whose recorded buys outweigh their sells on the curve — a **lower bound**, since a plain token transfer never touches the curve. |
| **Charts** | Candles folded from recorded trades at 1m, 5m, 15m, 1h, 4h and 1d. An empty bucket is a gap, not a flat line. |
| **Raised · progress** | The curve's own `raised()` against its own `graduationTarget`, both read off the contract. |
| **Portfolio value** | Balance times the last price someone paid. Not an offer; possibly out of date. |

A DASH, NEVER A ZERO

When a figure cannot be read — the chain did not answer, a rate is missing, nothing has traded yet — the site prints a dash with its reason. It never prints zero in its place, and it never carries a previous reading forward as if it were current. "We could not read it" and "it is zero" are different statements, and a page that merges them misleads in the most comfortable direction.

# 09 Signals

Three badges, three different kinds of claim — a chain fact, a proof of who posted, and a fixed rule. None of them is a rating.

## GRADUATED — a fact the chain states

The curve's `graduated()` is true. The badge is worn for two hours after graduation; the fact stays on the coin's page and under the Graduated tab for good. The Explore page's featured spot shows a coin that graduated in the last hour as **Just graduated** (several share the spot, ten minutes each) and otherwise the live coin closest to its target as **Next to graduate** — never an invented graduate.

## VERIFIED — who posted a code, not who anyone is

A creator asks for a one-time code, posts a sentence containing it from their own X account, and pastes the link. Sable fetches the post from X itself, by its id — never from the pasted address — checks that the author X reports is the account in the link and that the code is in the post, and records the handle. The post's text is read once and not stored.

It proves one thing: **the wallet that created this coin controlled that X account when the code was posted.** It does not prove who the account belongs to, that Sable checked anyone's identity, or anything at all about the coin. A verified creator is still a stranger. If X cannot be reached, no badge is given.

## DANGER — fixed rules, not a rating

Sable does not rate coins. The DANGER badge is raised only by fixed, published rules, each of which names the figure that raised it:

| RULE | RAISED WHEN |
|---|---|
| **Halted** | Sable's record marks the launch as halted. |
| **Creator concentration** | The creator's wallet — the curve's `creatorPayee()`, the address that called the factory, never a declared identity — holds 20% or more of the supply. |
| **Holder concentration** | The three largest wallets together hold 50% or more of the supply. |

Shares are measured against the supply read off the chain. A share the gateway could not read raises **nothing**, because an unread figure must never render as danger. **The absence of the badge is not safety.** It means only that none of these three rules fired on the figures that could be read.

## Favourites

The star on a card is kept in the visitor's own browser. Nothing is sent and nobody is counted, so there is no favourite count anywhere that anyone — Sable included — could inflate.

# 10 For developers

Everything the site does goes through the public API at api.buildsable.com. Reading needs no key; launching needs a wallet session.

A coin belongs to a wallet, so launching uses a **Sign-In With Ethereum** session ( `sess_…` ), not an `sk-sable_` API key. The full guide, with worked requests, is at buildsable.com/docs/launchpad-guide.

## 1 · Sign in

`POST /v1/auth/nonce` returns a nonce valid for five minutes. Sign an EIP-4361 message for an accepted domain (for example `launchpad.buildsable.com` , chain id 4663) with `personal_sign` , and exchange it at `POST /v1/auth/verify` for a bearer token.

## 2 · Lock the attributes

```
POST /v1/community/launches            // Authorization: Bearer sess_…
{ "chain_id": 4663, "slug": "pocket-sun",
  "attributes": {
    "asset":   { "name": "Pocket Sun", "symbol": "SUN", "description": "…" },
    "persona": { "tone": "warm", "stance": "earnest", "domain": "nature", "verbosity":
"normal" },
    "skills":  ["calls.read", "curve.read", "holders.read", "supply.read", "trades.read"] } }
// 201 -> id, slug, status "draft", spec_sha256, attributes_canonical
```

Through the API the creator chooses the persona values directly (the lists are published at `GET /v1/community/config` ). All five skills is Self Aware Mode on; omitting `skills` makes a mute coin. An optional `image_sha256` commits an image whose bytes are uploaded afterwards and refused unless they match. The response returns the exact canonical bytes that were hashed, so the digest can be recomputed independently.

## 3 · Deploy from the creator's wallet

`GET /v1/community/quote?chain=4663&token=native` (or a token address) returns whether the pairing is launchable and the target in the quote's own units, `target_quote_base` . The creator's wallet then calls the factory:

```
SableCurveFactory.create(
  string name, string symbol, bytes32 specHash,     // specHash = spec_sha256
  address quoteToken,                               // 0x0 = ETH
  uint256 graduationTarget)                         // = target_quote_base
// emits LaunchCreated(creator, token, curve, specHash, name, symbol,
//                     totalSupply, quoteToken, quoteDecimals, graduationTarget)
```

## 4 · Bind it

`POST /v1/community/launches/:id/deployed` with `{chain_id, token_address, curve_address, tx_hash}` . Sable reads the chain before listing the coin, and refuses with a fixed code if anything disagrees:

| CODE | MEANING |
|---|---|
| **not_from_factory** | The configured factory did not make this curve ( `curveOf` disagrees). |
| **spec_hash_mismatch** | The curve commits to a different digest than the locked one. |
| **token_mismatch** | The token is not the one this curve sells. |
| **quote_unpriced** | Sable cannot put a dollar rate on the quote asset. |
| **target_off_rule** | The target is not $6,000 at the current rate within 10% — bind promptly after deploying. |

## Reading, with no key

| METHOD | PATH | RETURNS |
|---|---|---|
| **GET** | `/v1/community/launches` | The directory; filter by `status` , `q` , `sort` ( `new` , `raised` , `graduating` , `graduated` ) |
| **GET** | `/v1/community/public/:slug` | One coin: attributes, curve, market readings, badges; `/trades` , `/holders` , `/candles` , `/curve` , `/room` , `/image` beside it |
| **GET** | `/v1/community/stream` | Server-sent events for the whole board: `trade` , `launched` , `graduated` ; one coin at `/public/:slug/stream` |
| **GET** | `/v1/community/config` | Pairings, rates, limits and the trait lists |
| **GET** | `/v1/community/stats` | Launchpad-wide figures |
| **GET** | `/v1/community/portfolio` | Coin balances for `?address=` |
| **POST** | `/v1/community/public/:slug/chat` | A private answer from the coin, returned once and stored nowhere |

Streams are capped per address and in total; over a cap the answer is 429 with a retry hint, and the page falls back to polling. The complete, machine-readable list is `GET /openapi.json` .

MCP

Sable's MCP server has **no launchpad tools today.** It cannot create, buy or sell coins; its `sable_backing_status` tool reads backed tokens only. An agent can still read everything about coins through the public endpoints above.

# 11 Backed tokens

A second kind of launch, built and tested, and not open.

The launchpad also contains a second, very different product: **backed tokens** — tokens for real assets such as tokenised stocks, bullion or credit. There, a licensed issuer locks a specification; a custodian's holdings are checked on a schedule under Sable's Proof of Backing; minting is authorised only against a verified check that covers the new supply; and transfers pause by rule after a discrepancy. There is no bonding curve, no graduation and no trading on Sable.

It is written and tested. **It is not open on production**: its write surface is switched off, and going live is gated on legal counsel. The Create page shows a Backed token option, but launching one is refused. Were it ever opened, Sable would still not issue, custody, sell or endorse any backed token; the issuer deploys under their own key.

# 12 What the launchpad does not claim

The most important section in this paper. Everything above is what the system does; this is where it stops.

- **Nothing here is investment advice.** Launching, buying or holding a coin is a decision each person makes for themselves, with money they can afford to lose entirely. Nothing guarantees that a coin reaches its target.
- **No price predictions, from anyone.** Sable makes none, this paper makes none, and no coin's character is permitted to make one.
- **A listing is not an endorsement.** The directory lists every bound launch with no curation. A halted coin, a coin with no trades and a coin that graduated are listed alike. Being listed is not a rating, a review, a recommendation, or a statement that a coin is lawful where you are.
- **A coin is a character, not a person.** It is not its creator, not a spokesperson and not an adviser, and its opinions are not Sable's.
- **Sable does not verify creators' identities.** VERIFIED proves who posted a code, not who anyone is. A creator is a wallet address.
- **DANGER is not a rating, and its absence is not safety.** It is three fixed rules over readings (§09).
- **"Raised" is never "market cap".** The graduation threshold is capital deposited; a capitalisation, where printed, is labelled as itself.
- **A reading is not a quote.** No figure on the site is a price Sable will honour, and a portfolio value is not an offer.
- **Commitments are not verifications.** `specHash()` fixes what a creator said; whether it is true is outside the contract entirely.
- **The anti-snipe window is a speed bump.** Funding many addresses defeats it.
- **The contracts are not yet independently audited.** Their properties are pinned by tests and by the public interface described above, and every claim can be checked on the chain; that is not the same as an audit.
- **Neither Sable token gates access or pays anything.** SABLE and SABL are not required to use the launchpad and confer no yield, revenue share, governance or claim. There is no migration, bridge or conversion between them.

OPERATING THE VENUE

Sable operates the curves' front end, lists the coins and takes a fee on trades. That is a larger legal exposure than anything else Sable does. **Opening the launchpad was the founder's decision**, and this paper reports no completed legal review of it. Nothing here is an offer to anyone in a place where such an offer would be unlawful.

# A Appendix — contracts, chain facts & glossary

Everything needed to check this paper against the chain.

### CHAIN

| | |
|---|---|
| **Network** | Robinhood Chain (an Arbitrum Orbit L2) |
| **Chain id** | 4663 |
| **Currency** | ETH |
| **RPC** | `https://rpc.mainnet.chain.robinhood.com` |
| **Explorer** | `https://robinhoodchain.blockscout.com` |

### CONTRACTS

| CONTRACT | ADDRESS |
|---|---|
| **SableCurveFactory** | `0x64c03B480b64A872f184c20e25808e1CBF43B211`<br>deployed at block 75,844,764 |
| **SableFeeSplitter** | `0x6f830893D5209e32B0277396A1e16B52B159B248`<br>deployed at block 73,680,675 |
| **Uniswap v4 PoolManager** | `0x8366a39CC670B4001A1121B8F6A443A643e40951` |
| **USDG (Global Dollar)** | `0x5fc5360d0400a0fd4f2af552add042d716f1d168` |
| **WETH** | `0x0bd7d308f8e1639fab988df18a8011f41eacad73` |
| **SABLE** | `0xf7894d31D569e6330592D346ecfeFdf4257F3EC1` |

Each coin's token and curve are deployed by the factory at creation; `curveOf(token)` on the factory names a coin's genuine curve, which is how to tell a launched token from one that merely copied its name.

### THE FACTORY'S TERMS, FIXED AT DEPLOYMENT

| | |
|---|---|
| **Supply per coin** | 1,000,000,000 tokens, 18 decimals |
| **Virtual base reserve** | 750,000,000 tokens |
| **Virtual quote reserve** | graduation target / 3 ( `virtualQuoteDivisor` = 3) |
| **Opening window** | 5 blocks; per-address cap = target / 20 ( `antiSnipeMaxDivisor` = 20) |
| **Trade fee** | 100 + 50 + 50 bps (platform · creator · voice) |
| **Graduation fee** | 100 bps |
| **Graduated pool** | Uniswap v4, 1% fee tier, tick spacing 200, no hooks, full range |
| **Platform payee** | `0x6f830893D5209e32B0277396A1e16B52B159B248` |

THE SPLITTER'S RECIPIENTS

| SHARE | GETTER | ADDRESS |
|---|---|---|
| **Launchpad · 50%** | `sable()` | `0xB0e8D9d81A0CC166Aa3343CDAB530264050A639E` |
| **Treasury · 30%** | `protocol()` | `0x06D47810B2927D7097D9111486e4AC5AbED5B23e` |
| **Community buybacks · 20%** | `buyback()` | `0xeA8FA69eF00f5682E4E93A3401f3Ef41432819af` |

The getters keep the names the contract was written with; the share labels are the ones Sable uses. Anyone can confirm every row above by calling the getters on the explorer or with any JSON-RPC client.

GLOSSARY

| | |
|---|---|
| **Bonding curve** | A contract that sells a token at a price set by a fixed formula, rising as it is bought. |
| **Quote asset** | What a coin is paid for in — ETH, USDG or another ERC-20 — chosen at launch and fixed. |
| **Raised** | Quote deposited into the curve, net of fees. The graduation threshold. Not a market cap. |
| **Graduation** | The permissionless transaction that closes a full curve and moves its holdings into a Uniswap v4 pool. |
| **Full-range position** | Liquidity spread across every price the pool can reach. |
| **Digest · specHash** | The sha256 of a coin's canonical attributes; committed on-chain so they cannot be swapped. |
| **Character** | A coin's four locked traits and, with Self Aware Mode, its skills and voice. |
| **Skill** | Permission for a coin to read one kind of live fact about itself. |
| **Refusal floor** | Rules no character can override: no price predictions, no trading advice, never a person. |
| **Reading** | A figure folded from trades Sable's indexer recorded. Not a valuation and not a quote. |
| **Reorg guard** | The indexer's rule of recording nothing within twelve blocks of the chain head. |
| **Slippage** | How much worse than the quoted amount a trader will accept before the trade reverts. |
| **SIWE** | Sign-In With Ethereum: proving control of a wallet with a signature, not a transaction. |