Orbit · Whitepaper v1 · September 2026

Yield-bearing tokenized SpaceX, accessed through Mooncoin burn

ORBIT is a redeemable vault share backed by SPCX, the tokenized SpaceX asset used by ORBIT, and funded by external fee income. Participants mint it by destroying Mooncoin in a 24-hour descending-price auction.

mooncoin.it · A technical specification of a live mechanism. Figures are illustrative; yield tracks trading activity.

Abstract

Mooncoin trades generate fees denominated in SPCX, the tokenized SpaceX asset used by ORBIT. Rather than letting that revenue dissipate, Orbit routes it into a vault and sells access to it through a daily Dutch auction whose only accepted payment is irreversibly burning Mooncoin. Fee income supplies the SPCX; the burn decides who receives newly issued shares. The output, ORBIT, is an ERC-4626 share redeemable through the controller for its pro-rata portion of the vault. 69% of each recognized inflow becomes auction inventory and 31% is added to the backing of shares already outstanding, so SPCX per ORBIT grows as fee income arrives. This document specifies the mechanism: the price curve and its derivation, the fixed-point arithmetic and its rounding directions, the exact sequence at epoch open, the state transitions for burn-and-mint and for redemption, the sources of yield, and the conservation invariants the implementation maintains.

`MOONCOIN → ORBIT → SPCX burn, hold, redeem`

Contents

1 The yield gap

Onchain asset classes found native yield. Tokenized stocks are still catching up.

Onchain finance solved yield for almost everything. Stablecoins earn through lending markets. ETH earns through staking. Liquidity positions earn trading fees. Each found a native mechanism that pays holders for holding.

Tokenized equity is still catching up. A tokenized stock position mostly tracks a price. There is no consensus role to stake into and no protocol emissions, and its unit count changes only if the issuer's terms provide for it.

ASSET	NATIVE YIELD MECHANISM
Stablecoins	lending markets
ETH	staking
Liquidity positions	trading fees
Tokenized equity	*price exposure*
ORBIT	Mooncoin trading fees, settled in SPCX

Figure 1. The gap Orbit closes. Yield does not have to originate inside the asset. It can be routed from an adjacent economy, which is exactly what Orbit does.

Orbit's insight is that **yield does not have to come from the asset itself.** It can come from an adjacent economy entirely. Mooncoin trades. Those trades pay fees. Those fees settle in SPCX. Route them to SPCX holders, and a tokenized stock position starts growing in SPCX terms, funded by a revenue stream with nothing whatsoever to do with SpaceX's business.

That is the whole idea: **fee income accrues to holders through increasing SPCX per ORBIT.** The rest of this paper is the machinery that makes it work, specified precisely enough to verify.

2 What Orbit does

Trading volume is a revenue stream. Orbit converts it into a balance sheet.

A memecoin with real trading activity produces real revenue: every swap pays a fee. That revenue is normally captured by liquidity providers, by the launch platform, or by a treasury that holds it indefinitely. The token itself stays what it was on day one: a claim on nothing.

The usual remedy is buy-and-burn, which spends the revenue in the token's own market. Orbit keeps the revenue instead, as an asset that stays on the balance sheet.

Fees are *retained* as an asset, specifically **SPCX, the tokenized SpaceX asset used by ORBIT**, inside a vault, and a separate token, ORBIT, is issued as a redeemable claim on that vault. The claim is an ERC-4626 share, redeemed through the controller at the vault's pro-rata price.

The design question is then narrow: *at what rate should the protocol issue claims on its own balance sheet, and what should it charge?* Orbit's answer is that it charges Mooncoin, and burns it. Every participant mint is paid for by a verified reduction in Mooncoin total supply, and its price is discovered daily by auction against a fixed opening reference rather than assumed.

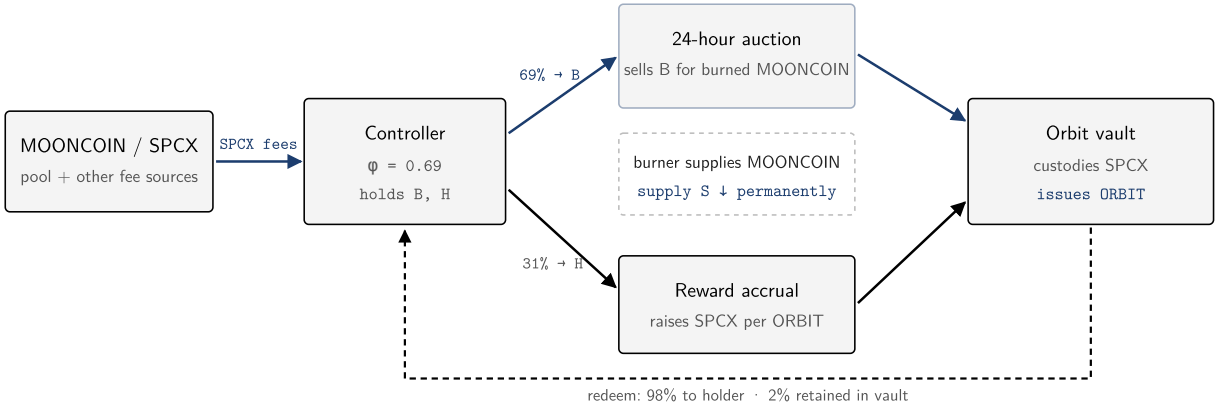

Figure 2. System flow. SPCX sits in controller inventory (B, H) until it is auctioned or reported, then in the vault behind ORBIT. Holders exit by redeeming through the controller or by transferring ORBIT. The 2% exit fee returns to the vault rather than being taken as protocol revenue, so it accrues to holders who remain.

3 Notation and system state

All quantities are integers in the token's smallest unit. Both MOONCOIN and SPCX have 18 decimals, enforced at construction.

Symbol	Domain	Meaning
`B`	$\mathbb{N}$	Issuance budget: recognized SPCX held by the controller, allocatable to auctions
`H`	$\mathbb{N}$	Reward budget: recognized SPCX awaiting the next report
`E`	$\mathbb{N}$	Current epoch's unsold allocation, always `E ≤ B`
`R`	$\mathbb{N}$	Remaining issuance allowance for the controller's mint path
`F`	$\mathbb{N}$	Exit fees already returned to the vault but not yet recognized
`A`	$\mathbb{N}$	Vault assets recognized as backing
`N`	$\mathbb{N}$	Effective ORBIT supply used for conversion
`S`	$\mathbb{N}$	MOONCOIN total supply
`σ`	$\mathbb{N}$	Seed shares: ORBIT held by the controller from genesis, not redeemable through it
`ρ`	$\mathbb{Q}^+$	Opening reference: SPCX per MOONCOIN, supplied once at epoch open
`T`	`86,400 s`	Epoch duration, 24 hours
`U`	`86,400 s`	Reward unlock window and minimum report interval, 24 hours
`t`	`[0, T)`	Elapsed time within the current epoch
`φ`	`0.69`	Issuance share of incoming fees
`θ`	`0.02`	Exit fee rate

Internally the contract represents prices in fixed point scaled by $\Sigma = 10^{36}$, denominated in *raw MOONCOIN per raw SPCX*. We write p for that internal price and $r = \rho \cdot \Sigma$ for the scaled reference. The opening reference is expressed in SPCX per MOONCOIN. The auction price uses the opposite denomination and includes the time-dependent multiplier: $p(t) \approx m(t) \cdot \Sigma^2 / r$, so at opening their scaled product corresponds to $4\times$. A higher p means *less* SPCX per MOONCOIN, a worse deal for the burner.

> **Why T = U.** The epoch length and the reward unlock window are both 24 hours. What keeps unlocks from overlapping is the report interval: after each report the controller sets `nextReportAt = reportTime + 86,400`, and it requires the vault's unlock window to equal 86,400 seconds. One day's gain has fully unlocked before the next report can land. Reports happen at most once per interval: the first is eligible one day after genesis, each later one a day after the previous actual report.

4 Fee dynamics and the 69/31 partition

One inflow, two destinations, three recognition routes, no rounding leak.

4.1 Where the fees come from

Every swap in the MOONCOIN/SPCX pool pays a trading fee. Those fees settle in **SPCX**, the numeraire side of the pair, and accrue to Orbit as the pool's designated fee beneficiary. Fee income is therefore a direct function of trading volume: more Mooncoin activity means more SPCX flowing into the vault, which is the sense in which *every trade fuels the vault.* Pool fees are one external source; any SPCX sent to the controller is recognized by the same rules.

Critically, the protocol never estimates this. It measures. Recognition is always based on an observed change in the controller's own SPCX balance, never on a reported volume figure or an expected payment.

4.2 Three routes in, one partition

SPCX reaches the controller three ways, which differ in *when* the split is recognized:

1. **Pull collection**: the operator claims accrued fees from the pool's fee manager. The amount credited is the controller's balance delta across that single call, not the value the fee manager reports. B and H update immediately.
2. **Authenticated push**: an atomic transfer from the designated fee payer, accompanied by a unique receipt identifier. Identifiers are recorded, so the same receipt can never be credited twice. B and H update immediately.
3. **Balance recognition**: SPCX sent by plain transfer changes the controller's balance only. It is measured and split at genesis or at the next successful epoch opening.

All three converge on the same partition. For an inflow of x:

$$\begin{aligned} \Delta B &= \lfloor\, 0.69 \;\cdot\; x \,\rfloor \\ \Delta H &= x \;-\; \Delta B \end{aligned} \tag{1}$$

Defining ΔH as a *remainder* rather than as $\lfloor 0.31x \rfloor$ is deliberate: it makes the partition exactly conservative, $\Delta B + \Delta H = x$, for every x, with no dust stranded on either side. The floor on ΔB also means issuance never rounds up at the reward budget's expense.

Unattributed balance is measured as

$$\text{unrecognized} \;=\; \text{balance}_{\text{SPCX}}(\text{controller}) \;-\; (B \;+\; H) \qquad \textit{// reverts if negative} \tag{2}$$

and partitioned by (1). The live allocation E is already inside B, so it is not subtracted again. If (2) would be negative, meaning the controller holds less SPCX than it claims to, the transaction reverts rather than proceeding on inconsistent state.

4.3 Timing: fees arriving mid-epoch wait

This is the nuance most easily missed. The epoch allocation E is fixed at open. SPCX that arrives *during* a live epoch does **not** increase E, whichever route it takes: collected fees and authenticated receipts raise B and H immediately, while plain transfers wait in the balance until the next opening recognizes them.

Either way, new SPCX becomes auctionable only at a later successful opening, normally the next day's epoch; an opening is a keeper transaction, not an automatic event. Meanwhile B and E diverge: fills decrement both equally, while recognized inflow increments B alone. This is also why the invariant $E \le B$ is maintained automatically rather than enforced by a check.

5 The 24-hour epoch

What happens, exactly, when an epoch opens and closes.

5.1 Opening

Opening an epoch is a single atomic operation. Either every step below succeeds, or none of them is recorded, including the reference price itself, which is not published unless the epoch that uses it also opens.

Algorithm: openEpoch(ρ, observedAt): atomic

1 guard	`not paused ∧ genesis complete ∧ previous epoch has expired`

2 reference	`require ρ within configured bounds`
	`require observedAt ≠ 0 ∧ observedAt ≤ now ∧ now − observedAt ≤ 120 s`
	`store ρ as the opening reference`

3 config	`assert vault hooks, no strategy debt, no accountant, unlock window = 86,400 s`
4 close	`if previous epoch not yet closed:` **`B ← B − E, H ← H + E, E ← 0`**
5 recognize	`x ← unrecognized balance (2)` **`B ← B + ⌊0.69x⌋, H ← H + x − ⌊0.69x⌋`** *// the 69/31 split*
6 band	`p_floor ← ⌈Σ² / 2r⌉ ; p_start ← 8 · p_floor`
7 allocate	**`E ← B`** `; require E > 0`
8 commit	`id ← id + 1 ; start ← now ; end ← now +` **`86,400 s`**

Two properties follow from this ordering. First, the 69/31 split of any pending balance happens at step 5, *before* the allocation at step 7, so SPCX that arrived by plain transfer since the last open is partitioned first and only its 69% share becomes auctionable. Second, the band computed at step 6 is fixed for the next 24 hours: the reference cannot be replaced while an auction is live, so the price schedule cannot change mid-fill.

The reference comes from the keeper's configured 30-minute pool average. The contract checks its bounds and the observation timestamp but does not recompute the average, so the auction runs against a **fixed opening reference** that the live market may drift from during the day. If B is zero or issuance is paused, no epoch opens.

5.2 Closing

An epoch stops accepting fills at 24 hours. Its inventory moves when the keeper closes it, directly or inside the next successful opening (step 4). Whatever remains unsold does **not** roll into the next auction. It is reclassified from issuance to rewards:

$$B \leftarrow B - E\ , \qquad H \leftarrow H + E\ , \qquad E \leftarrow 0 \tag{3}$$

Closing runs once per epoch: a flag records it, and a second close reverts. This is the mechanism behind *only sold inventory becomes new ORBIT*: a quiet epoch pays existing holders instead of accumulating an overhang for the next auction to liquidate.

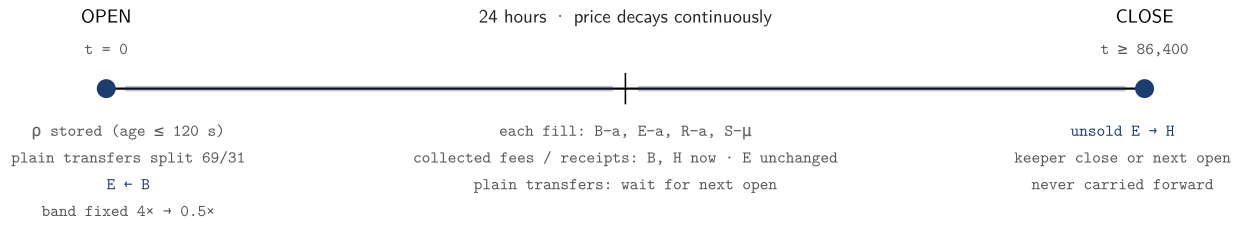

Figure 3. The epoch, end to end. The band is fixed for the full 24 hours. Mid-epoch inflow never enlarges the live allocation; it becomes auctionable at a later successful opening.

6 The price curve

A geometric descending-price auction, anchored to the published band.

6.1 Why descending-price

A descending-price auction finds where inventory sells without a commitment phase: one transaction burns MOONCOIN and delivers ORBIT. The auction opens expensive and gets cheaper, and each burner chooses when to fill. Different burners fill at different times and so at different prices, and transactions in the same block share one timestamp, so ordering within a block decides who reaches scarce inventory first.

The reference sets the *band*; burners decide *where in the band* the inventory sells.

6.2 Deriving the curve

Let $m(t)$ be the **price multiple**: how many times the opening-reference value of MOONCOIN a burner pays per unit of SPCX received. We require a constant *relative* rate of decline, the natural choice when the priced quantity is itself a ratio, since it makes the auction scale-invariant in ρ:

$$\frac{\mathrm{d}m}{\mathrm{d}t} = -\lambda\, m \implies m(t) = m_0\, e^{-\lambda t} \tag{4}$$

With band endpoints $m_0 = 4$ and $m(T) = ½$, the decay constant is fixed:

$$\lambda \;=\; \frac{\ln\left(\, m_0 \;/\; m(T)\,\right)}{T} \;=\; \frac{\ln 8}{T} \;=\; \frac{3\,\ln 2}{T} \tag{5}$$

Substituting (5) into (4) and rewriting in base 2 gives the curve the contract implements:

$$m(t) \;=\; 4 \;\cdot\; 2^{-3t/T} \;=\; 2^{\,2\,-\,3t/T} \tag{6}$$

Equivalently $\log_2 m(t) = 2 - 3t/T$, which is linear: the curve is a straight line on a logarithmic price axis, descending by exactly one factor of two every 8 hours. Over a 24-hour epoch the band passes through four clean anchors:

$$\begin{aligned} m(0\text{ h}) &= 4 && \textit{// open: highest price for the burner} \\ m(8\text{ h}) &= 2 \\ m(16\text{ h}) &= 1 && \textit{// parity with the opening reference} \\ m(24\text{ h}) &= 0.5 && \textit{// limit; not executable at exactly } t = T \end{aligned} \tag{7}$$

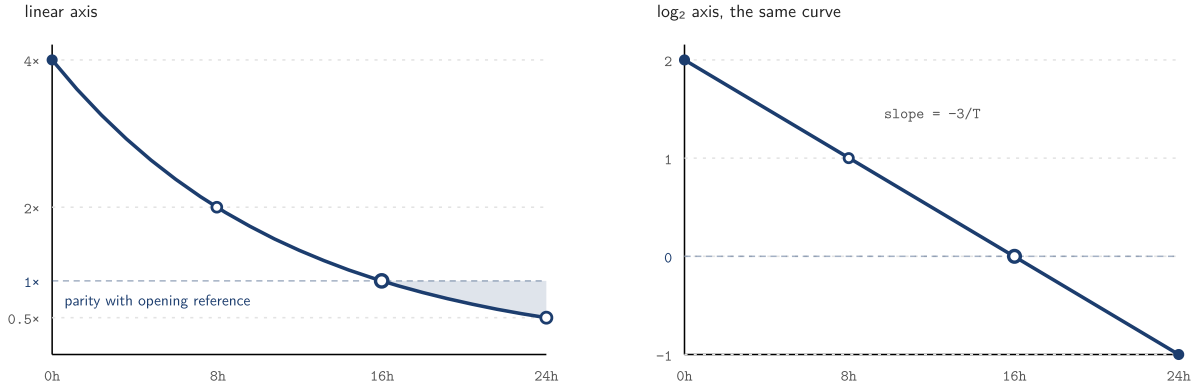

Figure 4. The daily band, two views. Left: $m(t)$ on a linear axis; past 16 hours (shaded) a burner receives more SPCX than the opening-reference value of the MOONCOIN burned. Right: $\log_2 m$ is exactly linear with slope $-3/T$, which is what makes the decay scale-invariant in ρ.

6.3 Fixed-point implementation

The contract does not store m. It derives the live price from the stored *floor* of the internal price, so the conservative endpoint is the one held exactly:

$$\begin{aligned} p_{\text{floor}} &= \lceil\, \Sigma^2 \;/\; (2r)\,\rceil && \textit{// } \Sigma = 10^{36} \\ p(t) &= \lceil\, p_{\text{floor}} \;\cdot\; \exp2(e(t)) \;/\; 10^{18}\,\rceil \\ e(t) &= \lfloor\, 3(T - t) \;\cdot\; 2^{64} \;/\; T\,\rfloor && \textit{// 192.64 fixed point} \end{aligned} \tag{8}$$

At $t = 0$ the exponent is exactly $3 \cdot 2^{64}$, for which `exp2` returns exactly $8 \cdot 10^{18}$, so $p(0) = 8 \cdot p_{\text{floor}}$ exactly. The stored floor itself already carries the ceiling from its own division. The maximum exponent reached is 3 in 192.64 form, far inside the function's domain.

The intermediate product $p_{\text{floor}} \cdot \exp2(e)$ can exceed 2^{256} while the quotient does not, so (8) is evaluated with a 512-bit multiply-divide rather than a naive product. This is load-bearing, not defensive: at the smallest admissible reference the naive form would overflow.

6.4 Rounding directions

The final rounding step of each calculation is oriented the same way: against the burner, in favour of backing. Three properties follow directly from the floors and ceilings:

$$\begin{aligned} p_{\text{floor}} \cdot 2r &\geq \Sigma^2 && \textit{// the floor never sits below the advertised 0.5× endpoint} \\ a \cdot p &\leq \mu \cdot \Sigma && \textit{// backing issued never exceeds what was paid for} \\ \text{fee} \cdot 10^4 &\geq 200 \cdot g && \textit{// the exit fee is never below 2\%} \end{aligned} \tag{9}$$

These describe the last rounding step of each calculation. The exponent is floored and `exp2` has finite precision, so they are not a claim that every intermediate step favours the vault relative to exact real arithmetic.

7 Burn and mint

The participant issuance path. Permissionless, priced by the band, paid for in destroyed supply.

> **The core transaction**
>
> A burner supplies MOONCOIN. The contract verifies that total supply actually falls, draws the corresponding SPCX from the issuance budget, deposits it into the vault, and delivers freshly minted ORBIT to the receiver. **Every participant mint requires a verified burn, and so did genesis.**

Burning μ units of MOONCOIN at time t entitles the burner to

$$a(t) = \left\lfloor \frac{\mu \cdot \Sigma}{p(t)} \right\rfloor \approx \mu\,\rho \cdot 2^{3t/T - 2} \tag{10}$$

units of SPCX backing. The left-hand expression, using the implemented price, is authoritative; the exponential is the idealized curve, and the two differ by rounding. At opening with $r = 3 \cdot 10^{35}$ and $\mu = 10^{18}$, the ideal gives 75,000,000,000,000,000 raw SPCX and the contract gives 74,999,999,999,999,999. Each fill is subject to three caps:

$$a \leq B \quad \textit{// issuance budget} \qquad a \leq E \quad \textit{// this epoch} \qquad a \leq R \quad \textit{// issuance allowance} \tag{11}$$

The shares delivered are whatever the vault issues for a at the prevailing share price, which the burner bounds with a minimum-output parameter. Fills are all-or-nothing: if the minimum is not met, the entire transaction reverts and **no MOONCOIN is destroyed**.

Algorithm: burnForOrbit(μ, s_{min}, w, k, d)

require	`k = epoch.id // stale-epoch fills are rejected`
	`now ≤ d // caller-supplied deadline`
	s_{min} `> 0, w ∉ { 0, self, vault }`
	`not paused ∧ vault configuration unchanged`
price	`p ← p(t) // reverts if the epoch is closed or expired`
compute	`a ← ⌊ μ·Σ / p ⌋, require a > 0 ∧ a ≤ min(B, E, R)`
effects	**`B ← B − a ; E ← E − a ; R ← R − a`**
burn	`pull μ MOONCOIN from caller, then burn`
	`assert ΔS = −μ ∧ Δbalance(self) = 0 // supply really fell`
deposit	`s ← vault.deposit(a, w)`
	$\Delta balance_{SPCX}$`(vault) = +a ∧ Δshares(w) = +s`
require	$s \geq s_{min}$ `// else revert the whole transaction`

Three details are worth drawing out. First, **state effects precede external calls**: the budgets are decremented before any token is touched, and the entire function is non-reentrant. Second, the burn is verified rather than assumed: the contract checks that MOONCOIN's total supply decreased by exactly μ and that its own balance returned to where it started, so a token that merely forwards to a dead address rather than reducing supply would fail closed. Third, every balance change is asserted with an exact before-and-after delta, so a fee-on-transfer or rebasing token cannot silently short the vault.

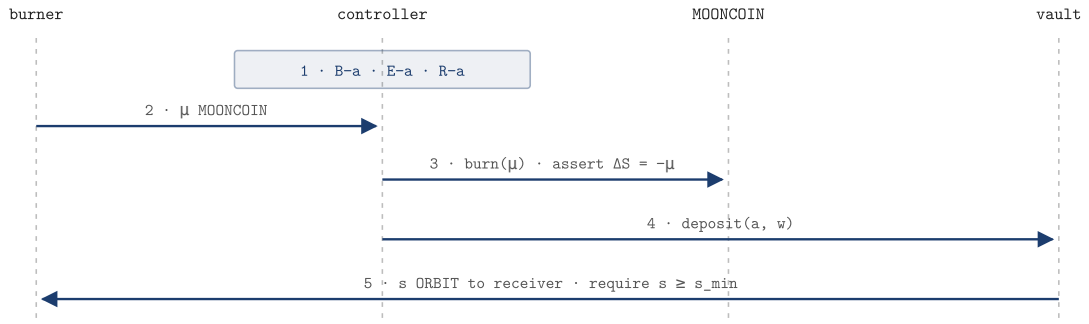

Figure 5. Burn-and-mint sequence. Budgets are decremented before any external call, and supply reduction is verified on-chain before any backing is deposited. The receiver may differ from the burner. The whole sequence is atomic: any failed assertion reverts the burn.

7.1 Supply dynamics

Each fill moves the system along two axes at once: MOONCOIN supply falls by μ, and vault assets rise by a. Across an epoch the total is a sum over fills at their own times: $a_{\text{total}} = \Sigma_j \lfloor \mu_j \Sigma / p(t_j) \rfloor \approx \rho \, \Sigma_j \, \mu_j / m(t_j)$. New shares are issued at the vault's prevailing share price, so an ordinary mint adds SPCX and shares in proportion and does not dilute SPCX per share. Separately, R caps the cumulative SPCX deposited through the controller's issuance path, genesis included, at 69,420 SPCX; rewards are not counted against it.

8 Redemption

The property the rest of the design exists to support. Pro-rata, priced by the vault, independent of the auction.

> **Independent of the auction**
>
> Redemption does not depend on the auction or on the operator. If issuance is paused, if no epoch opens, if the operator stops acting, **redemption still works** under the current vault configuration. Changes to vault configuration go through the 48-hour timelock (§12).

A holder redeeming s shares receives their pro-rata portion of vault assets, less the exit fee:

$$\begin{aligned} g &= \text{previewRedeem}(s) \approx \lfloor\, s \cdot A / N \,\rfloor \quad \textit{// } N \textit{ = effective supply, before redemption} \\ \text{net} &= \lfloor\, 0.98 \cdot g \,\rfloor \\ \text{fee} &= g - \text{net} = \lceil\, 0.02 \cdot g \,\rceil \end{aligned} \tag{12}$$

The fee is not revenue. It is transferred back into the vault, where, once reported and unlocked, it raises assets per share for every holder who did not redeem. Mechanically it is a Pigouvian transfer: the redeemer compensates remaining holders for exiting. Net + fee = g exactly, and for integer g the fee equals $\lceil 0.02g \rceil$, so it is never below 2%.

Redeemed ORBIT shares are burned. Their corresponding SPCX leaves the vault, with the retained fee returned for remaining holders, and the remaining shares split future rewards among fewer holders.

Algorithm: redeemWithFee(s, n_{min}, w, d)

require	`σ > 0, s > 0, n_min > 0, now ≤ d, w ∉ { 0, self, vault }`
pull	`transfer s ORBIT from caller to self, assert Δ = +s`
preview	`g* ← previewRedeem(s) // taken before redeeming`
redeem	`g ← vault.redeem(s, self, self, maxLoss = 0)`
	`assert g = g* // no slippage tolerated`
	`assert Δshares(self) = 0 // exactly s burned, seed untouched`
	`assert priorShares ≥ σ // the seed is never consumed`
split	`net ← ⌊0.98·g⌋ ; require net ≥ n_min`
	`fee ← g − net`
pay	**`fee → vault ; net → w`**
	`assert Δbalance_SPCX(self) = 0 // controller keeps nothing`
effects	`F ← F + fee // recognized at the next report`

The final assertion is the structural one: across a complete redemption the controller's own SPCX balance is unchanged. Everything that came out of the vault went either to the redeemer or straight back into the vault. The controller is a conduit, never a counterparty.

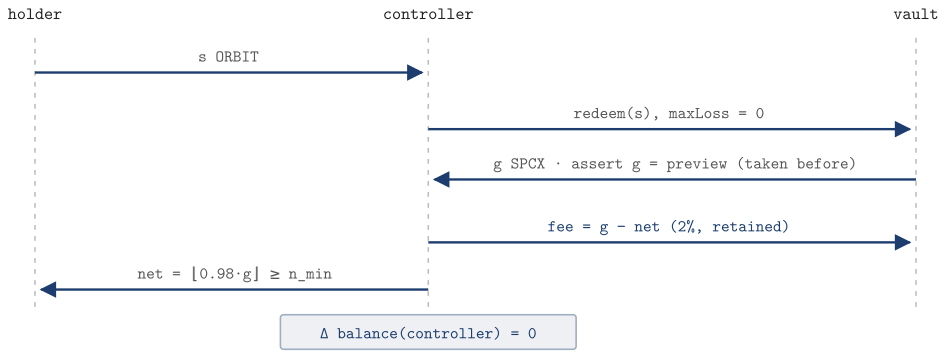

Figure 6. Redemption sequence. The controller never retains value from a redemption. The 2% goes back into the vault, where it accrues to holders who remain.

8.1 Why the fee cannot be bypassed

ORBIT is a standard ERC-4626 share. A plain vault would let any holder call `redeem` directly and avoid the fee entirely. Orbit closes that surface by installing the controller as the vault's deposit and withdraw hook. The hooks return a non-zero limit only for the controller's own synchronous, in-flight operation, identified by a transient receiver and amount set immediately before the call and cleared immediately after.

Every other caller sees a limit of zero. That covers all four ERC-4626 entry points and each of their overloads, and delegated calls made through an ERC-20 share allowance. The gate is keyed on the *operation*, not on the caller's identity, so under the current hook no address is exempt, including the team's.

9 Yield dynamics

Three sources, one distribution channel, a 24-hour unlock.

"Yield-bearing" is a specific claim, so it is worth stating exactly where the yield comes from. SPCX per ORBIT rises from three sources. The first two are destinations for the same external funding; the third is a transfer from exiting holders to those who stay. SPCX donated directly to the vault is also reported as gain.

Source	Mechanism	When
1. Reward share	31% of every recognized SPCX inflow is routed to H rather than to issuance	Credited on recognition; distributed at the next report
2. Unsold allocation	Whatever the auction does not sell is reclassified from B to H at close, by (3)	At epoch close; distributed at the next report
3. Exit fees	The 2% from every redemption is returned to the vault and accumulated in F	Immediately on redemption; recognized at the next report

Note what this implies. Auction demand decides how funded inventory is shared out: on a *quiet* day more of it flows to existing holders as rewards, on a *busy* day more becomes new ORBIT. Every unit of external funding ends up behind ORBIT either way.

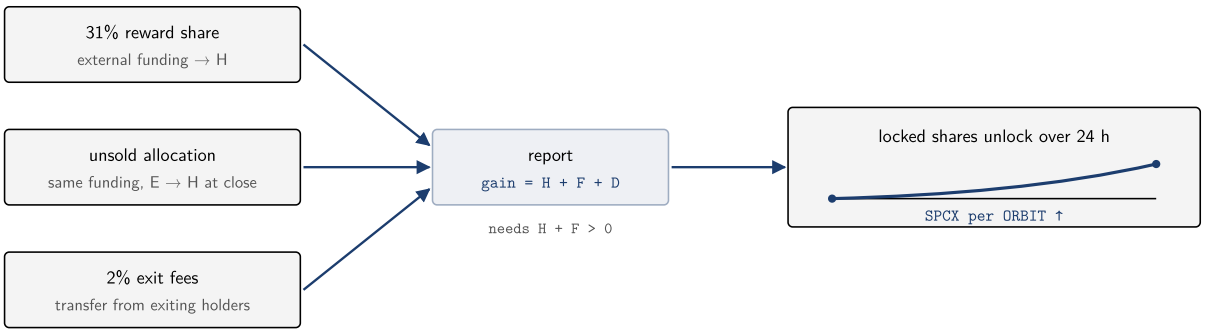

Figure 7. Three sources, one channel. The first two are the same external funding taking different routes; redemption fees move value between holders. All of it reaches holders through the same report-and-unlock path.

9.1 Recognition and the 24-hour ramp

The reward budget H, the accumulated exit fees F, and any unattributed donations D inside the vault are recognized together in a periodic report, eligible at most once per 86,400 seconds:

$$\text{gain} \;=\; H \;+\; F \;+\; D \;, \qquad H \;\leftarrow\; 0 \;, \qquad F \;\leftarrow\; 0 \qquad \textit{// requires } H + F > 0 \tag{13}$$

A donation alone cannot trigger a report. A gain does **not** raise the share price instantaneously. The vault adds the gain to its accounted assets and mints itself locked shares worth exactly that gain, so the price is unchanged at the report. The locked shares then leave the effective supply linearly over 24 hours. For one report with no prior locked gain and no other flows, SPCX per ORBIT follows

$$\begin{aligned} L_0 \;&=\; G \;\cdot\; N_0 \;/\; A_0 \qquad \textit{// locked shares minted at the report} \\ u \;&=\; \min(\,\text{elapsed} \;/\; 86{,}400\;,\; 1\,) \\ P(u) \;&=\; \frac{A_0 \;+\; G}{N_0 \;+\; L_0\,(1 \;-\; u)} \qquad \textit{// SPCX per ORBIT} \end{aligned} \tag{14}$$

Here A_0 and N_0 are assets and effective shares before the report and G is the gain. The locked shares unlock linearly; the share price does not. It rises along a convex curve: with $A_0 = N_0 = G = 100$, it is 1 SPCX per ORBIT at the report, 1.333 halfway through and 2 at completion. Exact figures are integer-rounded, and the vault's own getters are authoritative.

The unlock removes the instant jump a report would otherwise create, so buying ORBIT the block before a report captures nothing immediately. A holder who enters before a report does share pro-rata in the unlock that follows, and can leave at any point: by redeeming (2% fee) or by selling ORBIT on the market (no redemption fee, though pool fees and slippage apply). Because the report interval equals the unlock window, a new unlock begins only after the previous one completes.

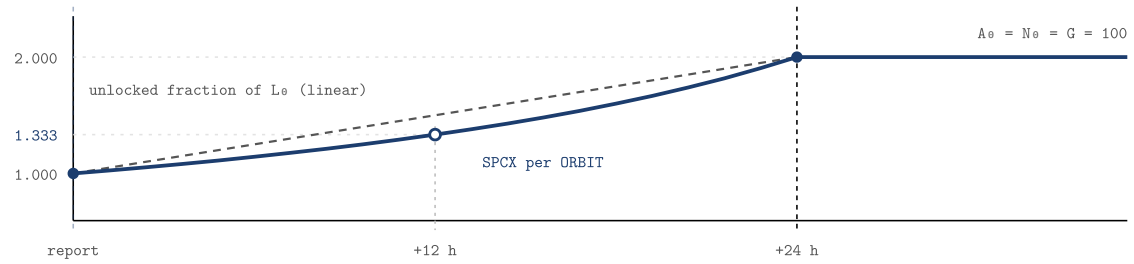

Figure 8. Why the unlock is a ramp. Locked shares unlock linearly (dashed, 0 to 100% over the same vertical span); SPCX per ORBIT follows equation (14) (solid). A step would make the report block the single most valuable block in which to hold ORBIT; the unlock spreads that across the day.

10 A worked example

One epoch, two outcomes. Round numbers, illustrative only.

Suppose a day's trading produces **1,000 SPCX** in fees, the reference is $\rho = 0.0001$ SPCX per MOONCOIN, and the budget was empty before. At open, (1) partitions the inflow:

$$B = 690 \quad \textit{// auctioned today} \qquad H = 310 \quad \textit{// to holders} \qquad E \leftarrow B = 690 \tag{15}$$

Outcome	Fill	MOONCOIN burned	Backing issued	Rewards from inflow*
A. Full fill at parity someone takes the lot at hour 16, $m = 1$	690 / 690	6,900,000	690 SPCX	310
B. Half fill at parity only 345 clears; the rest expires	345 / 690	3,450,000	345 SPCX	**655**

In outcome A, 6.9 million MOONCOIN leave circulation permanently and 690 SPCX of new backing enters the vault against newly minted ORBIT. Holders are allocated the 310 reward share, which unlocks over 24 hours once reported.

In outcome B, only half the inventory sells. The unsold 345 SPCX is *not* carried into tomorrow's auction; by (3) it moves to H, so the reward allocation from this inflow is **655** instead of 310. Half as much MOONCOIN is burned and, at a constant share price, half as many new ORBIT are issued.

∗ Cumulative reward allocation attributable to this inflow, whenever it is reported. The keeper may report the 310 before the epoch closes, so H at the moment of close can be lower; the cumulative allocation is the same.

> **The structural point.** Weak auction demand does not idle the protocol: it routes more of the funded inventory to existing holders and issues less new supply. Outcomes A and B put the same 1,000 SPCX behind ORBIT; they distribute it differently between new entrants and incumbents. What an individual holder realizes also depends on supply, entry timing and unlock progress.

Note also what does *not* happen in either case: SPCX arriving during the day never enlarges this epoch's auction; it becomes auctionable at a later successful opening (§4.3).

11 Conservation invariants

Relations the implementation maintains after every successful transaction in solvent states, under the current configuration.

$$\begin{aligned}
&\textbf{I1} && \text{balance}_{\text{SPCX}}(\text{controller}) \;\geq\; B \;+\; H \\
&\textbf{I2} && E \;\leq\; B \\
&\textbf{I3} && \text{balance}_{\text{SPCX}}(\text{vault}) \;\geq\; A \;+\; F \\
&\textbf{I4} && \Sigma_i \,\text{previewRedeem}(s_i) \;\leq\; A \\
&\textbf{I5} && R \text{ non-increasing, } R \;\leq\; R_0 \\
&\textbf{I6} && \text{shares}(\text{controller}) \;\geq\; \sigma
\end{aligned} \tag{16}$$

Scope. I4 counts each disjoint share position once, at a single state, converting against the effective supply N. I6 is a property of the controller's own paths. Reporting also checks for any shortfall and reverts if it finds one.

In words: every unit of budget the controller claims, it actually holds (I1); the auction can never allocate more than the budget backs (I2); the vault always covers what it owes plus fees not yet recognized (I3); the sum of all claims never exceeds the assets behind them (I4); the controller's issuance allowance only counts down (I5); and the seed position is never redeemed through the controller (I6).

I2 deserves a note because it is the one that is not obvious. At open, E is set equal to B. Each fill decrements both by the same a. Mid-epoch inflow increases B alone, per §4.3. The gap therefore only ever widens, and the close in (3) subtracts a quantity bounded above by B, so it cannot underflow.

Two further properties are structural rather than numerical. The controller has **no rescue function, no arbitrary-call facility and no upgrade path**: no role can use it to withdraw funds arbitrarily. The two holder-facing entry points are burn-and-mint and redemption, and both are priced.

12 Custody and trust model

Three roles, an enforced delay between intent and effect.

SPCX is custodied in the Orbit vault, a standard ERC-4626 vault deployed from a pinned, byte-verified implementation. The controller is installed as its deposit and withdraw hook and holds exactly one vault role, reporting manager (role bitmap 32). It cannot add strategies, change limits, or alter configuration.

Role	Can	Cannot
Operator keeper · `0x1FeE…f058`	Supply the opening reference (a trusted input); open and close epochs; collect fees and deliver authenticated fee receipts; trigger reward reports. May mint for itself only under the same public burn rules as anyone.	Withdraw funds arbitrarily. Award itself shares. Change any parameter.
Admin 2-of-3 Safe `0x5693…c180` → 48 h timelock `0xB4EE…C496`	Pause and resume *issuance* immediately. Through the timelock, after 48 hours: reconfigure vault roles, hooks and settings, and the timelock delay.	Execute a timelocked change before the delay, or after it has been cancelled.
Canceller held by the keeper address	Cancel any pending timelock operation before it executes	Propose or execute anything

The delay is the substance of the design. Any change touching the vault's configuration is visible on-chain for 48 hours before it can execute, and the canceller, a key separate from the admin Safe, can cancel it within that window. The controller owner and the vault's role manager are both the timelock. Pausing *issuance* does not pause *redemption*.

The current governance configuration requires a 48-hour delay. Timelocked changes can alter permissions, withdrawal hooks and the delay itself, including changes that restrict redemption.

> **What the delay does not do.** A timelock is a warning system, not a prohibition. It gives 48 hours of notice and a cancellation window; it does not remove the admin's powers. Holders can monitor it and react.

13 Parameters

Production configuration, as read on-chain. Controller values are constants; the unlock window and the timelock delay are governance-mutable.

Symbol	Value	Meaning
φ	0.69	Share of inflow routed to the issuance budget B
1 − φ	0.31	Share routed to the reward budget H
θ	0.02	Exit fee, retained in the vault
T	86,400 s	Epoch duration, 24 hours
U	86,400 s	Reward unlock window, 24 hours (vault setting, governance-mutable)
$m_0 \to m(T)$	4 → 0.5	Auction price multiple band, halving every 8 hours
freshness	120 s	Maximum age of the reference at epoch open
Σ	10^{36}	Internal fixed-point price scale
R_0	69,420 SPCX	Issuance allowance for the controller mint path, genesis included
genesis	69,420 MOONCOIN	Burned at the 4× opening price; resulting shares held by the controller as σ
delay	172,800 s	Governance timelock, 48 hours (governance-mutable)
decimals	18	Both MOONCOIN and SPCX, enforced at construction

> **Status.** Orbit is live in production. Controller `0x93AE26cbc660aCf55B22892eEeECB80fc0579d86`, vault `0x104D8f490916146F9654D161892eDcC718410000`, timelock `0xB4EEf8B8d5015b019034cDC5A127726e3b00C496`. The current configuration is published at mooncoin.it/api/orbit/v1/config.
> Verified on-chain at block 69,828,455 (22 September 2026): the deployed controller's runtime bytecode is byte-identical to a build of `contracts/orbit-pilot/src/OrbitController.sol` at commit `969f653` (main). The vault is a minimal proxy to implementation `0x8baea8ac4c7b8f893b0273873c62c89a5c190b0c`. All values in this section were read from the contracts at that block.

14 What this unlocks

A template, not a one-off.

Strip Orbit down and the pattern is general: **take a high-velocity speculative asset, capture the external fee income it already produces in SPCX, and allocate it between new shares and existing holders, with new shares open to anyone willing to retire supply.** Nothing in the mechanism is specific to SpaceX. The numeraire could be any tokenized asset that settles onchain; the burn token could be any asset with genuine trading volume.

What makes the SpaceX case compelling is the pairing. Memecoin volume is abundant, volatile and short-horizon. Tokenized SpaceX exposure is scarce and long-horizon. Orbit routes the fee income of one to holders of the other, in 24-hour increments, with every participant mint permanently reducing the supply of the thing being spent.

Three properties fall out of the construction that are worth stating plainly:

- **Fee income accrues to holders through increasing SPCX per ORBIT.** Rewards and unsold inventory are reported to the vault, exit fees stay in it, and mints enter at the prevailing share price. Under normal flows each is designed to preserve or raise SPCX per share.
- **Controller issuance is capped.** The mint path can deposit at most $R_0 = 69{,}420$ SPCX over its lifetime, a constant written into the controller.
- **The exit does not depend on the auction.** Redemption needs no live epoch, no keeper action and no reference price, and is priced pro-rata by the vault.

A memecoin whose fee stream builds an SPCX position for ORBIT holders, and burns itself down doing it. That is the trade Orbit makes available, once every 24 hours.

Figures in this document are illustrative. Yield depends on Mooncoin trading activity, auction fill behaviour and redemption volume, and is not fixed or guaranteed. ORBIT is denominated in SPCX, a token issued by a third party whose rights are defined by that issuer's terms. Nothing here is investment advice.

`mooncoin.it`