Lull

Litepaper

Lull, in brief

Lull in about ten pages: the problem, how a lull works end to end, pricing and the cross, what is live today, and $LULL in brief.

Version 1 · September 2026 · 3,886 words

What Lull is

Lull is a sealed market-on-open auction for xStocks, the tokenized equities that trade on Solana. A user places an order while the US market is closed. The order stays sealed until shortly before the open, and then it crosses once, together with every other order, at P_{open}: the first fully verified Pyth price of the underlying stock published after the regular session opens.

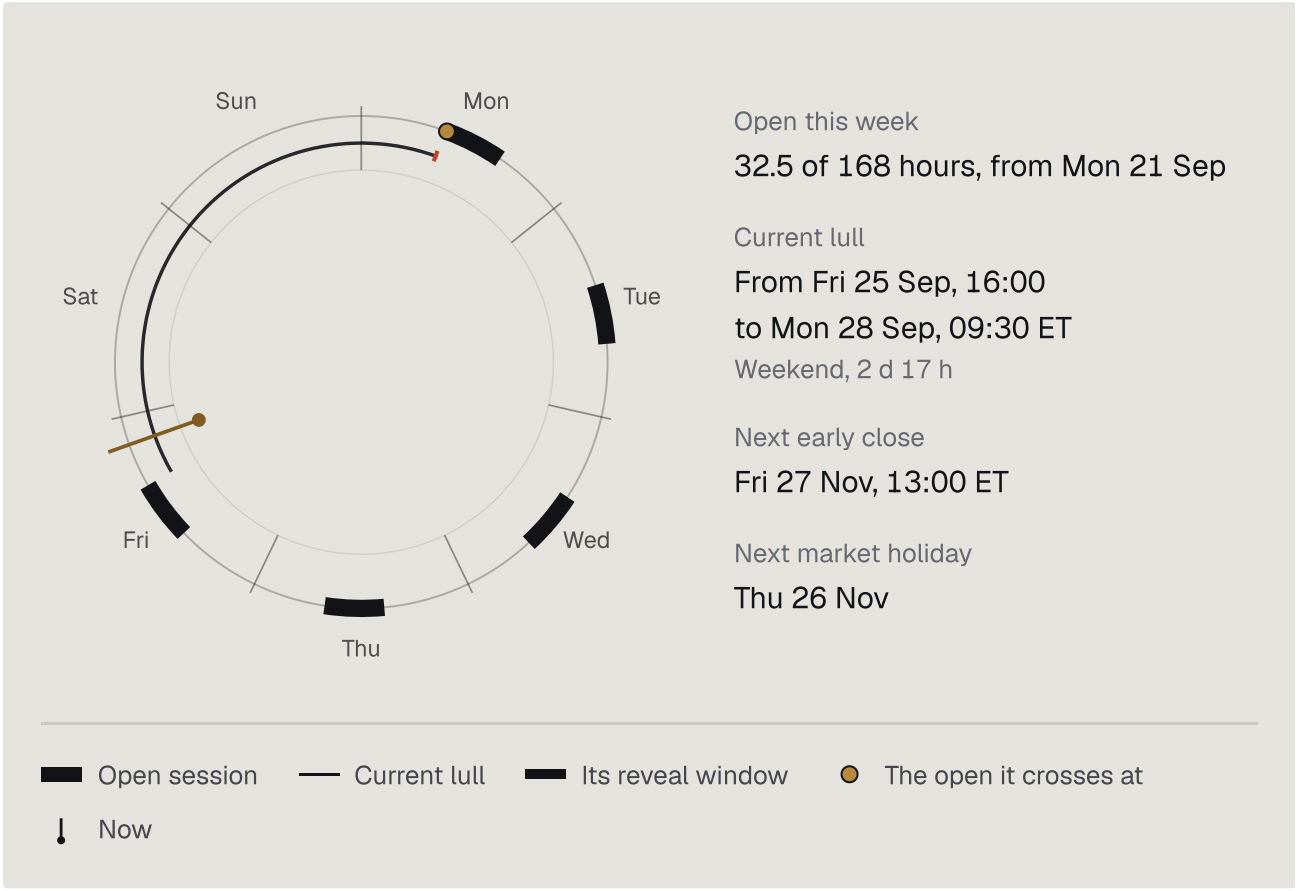
The first market is AAPLx/USDC. AAPLx is the xStock that tracks Apple, and its open price comes from Pyth's `Equity.US.AAPL/USD` feed. Lull runs one market per xStock.

Lull is built for self-custody. The user's wallet signs every transaction, the backend never holds user keys, and the secret that seals an order never leaves the user's browser before the order is revealed on chain.

The problem

The lull

xStocks trade around the clock. Apple trades in the US regular session, 09:30 to 16:00 Eastern Time on weekdays, which is 32.5 of the week's 168 hours. The rest of the week, 135.5 hours or about 80% of it, is a lull. An overnight lull lasts 17.5 hours, and a normal weekend 65.5 hours. A holiday weekend runs to 89.5 hours or more.



The lull week of AAPLx/USDC in New York time, built from the Pyth market-hours schedule of Equity.US.AAPL/USD. Monday is at the top and the week runs clockwise.

During a lull the xStock keeps trading in on-chain pools while its reference market is shut. Three things go wrong:

- **Pools are thin.** An LP in an xStock pool overnight holds a price it cannot hedge until the open, so it quotes wider, provides less depth, or leaves.
- **Prices drift.** The pool price moves with whatever flow arrives. The underlying's official price stays where it closed until the next open.
- **Orders leak.** On a public chain, pending and resting orders are visible. A trader who sees a sell imbalance building on a Saturday can sell ahead of it into thin pools, or position elsewhere, before Monday.

What the data shows

Lull's read-only deployment measures this for every lull. It records the prior close, the first Pyth price after the open, and the xStock's DEX price through the night, sampled once a minute from Jupiter's price API. On 2026-09-26 the drift endpoint reported one complete night with DEX data, 24 to 25 September. The underlying closed at $335.96119 and its first price after the open was $336.02504, a move of 2 bps. Over the same night the xStock's DEX price ranged from 51 bps below the close to 35 bps above it. A holder who sold into the pool at the night's low was about half a percent worse off than at either the close or the open, before any spread or slippage.

One night is an illustration, not a statistic. The DEX series is Jupiter's indicative price, not an executable quote. But it shows the mechanism: through a lull, the pool price wanders against a reference that does not move until the open.

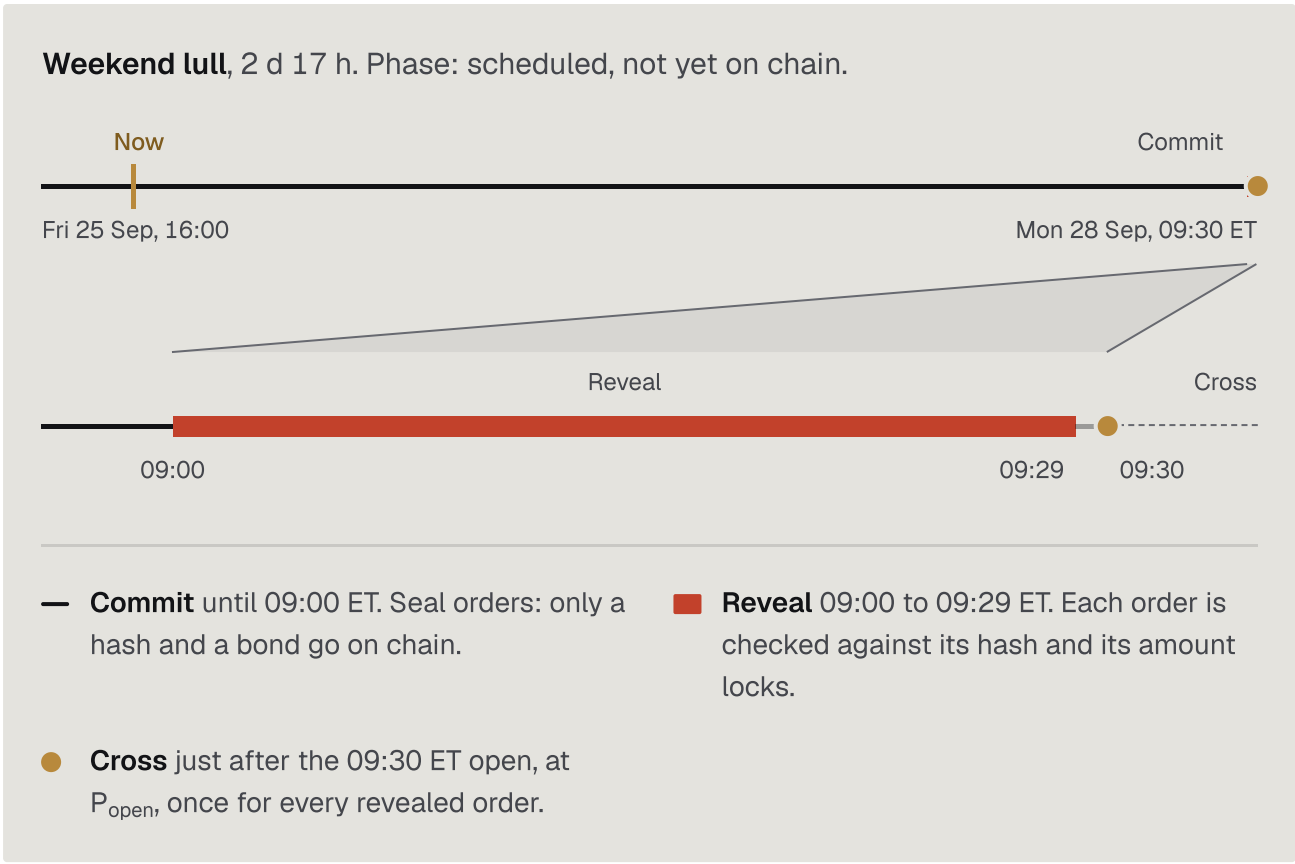
How brokerages solve it

Brokerages handle the same problem with opening auctions. A client sends a market-on-open (MOO) or limit-on-open (LOO) order at any time. The exchange collects every opening order and crosses them at one official price at the open. Nobody outside the broker and the exchange sees the queue. A self-custody wallet has had nothing like this. Lull provides it.

How a lull works

The calendar

Lull's calendar comes from Pyth's own market-hours schedule for the feed that supplies P_{open}. The schedule is a string: a time zone, the weekly hours, and dated overrides for holidays and early closes. Lull's worker parses it with daylight saving time, holidays and early closes, and the parser is tested against the market hours that Pyth's Hermes service computes itself, for every feed it serves. Each lull is published on chain with four times: the close, the reveal window's start and end, and the open. The program cannot read Pyth's schedule, but it enforces the ordering `close < reveal_start < reveal_end ≤ open`.



The current lull of AAPLx/USDC from the Lull API, in New York time. The top bar is the whole lull to scale. The bottom bar enlarges the reveal window and the open.

Step 1: deposit

A user deposits USDC or the xStock into a per-market balance account. Deposits and withdrawals of free balance work at any time. Tokens sit in the market's vault, and the balance is the program's record of the user's share. Only the program's rules move it, and only the owner's signed withdrawal takes tokens out.

Step 2: commit

During the lull the user seals an order. The browser generates 32 random bytes, the salt, and computes a commitment:

```
sha256("lull:v1:order" ‖ side ‖ amount ‖ limit ‖ salt ‖ owner ‖ lull)
```

Example order

Side	Amount to spend	Limit (USD per share)
Buy / Sell	1000 USDC	$ None: market on open

Tag 13 bytes	The text “lull:v1:order”, which separates orders from LP quotes `6c756c6c3a76313a6f72646572`
Side 1 byte	0, a buy `00`
Amount 8 bytes	1,000,000,000 raw quote: 1,000 USDC × 10^6 `00ca9a3b00000000`
Limit 8 bytes	0: market on open `0000000000000000`
Salt 32 bytes	Random, drawn in this browser. Whoever holds it can reveal the order. Show salt · New salt
Owner 32 bytes	Example owner: `Apw4gdEdr17JYaEo8qybGQpLwL2XZsW66jGgHVEdCGMP` `92009c338db30e9e52d6a09c635204c114db3aafaa1b639474227c576377bcd2`
Lull 32 bytes	Example lull: `EVjiyLsgWLGeq7eZBcZsGrXfHrkJgKiBnv1ot1UijcTB` `c884801f38671fc1263b068c461898f0e57f3f99bd183a415c08e120feed5d5a`
	↓ sha256

Commitment
32 bytes
	`fd882ae35cdb1fa83d8d2bc1a6b3e611`
	`f0105b836e8214e40cd9f3346a50fd8a`
	Only this hash, the bond and the owner's address go on chain when the order is sealed.

Example inputs. The hash is computed in your browser with the SDK function the order ticket seals with. The owner and the lull are random example keys, not real accounts, and the salt never leaves this page.

Only the hash and a bond go on chain. The bond is 0.01 SOL by default. No tokens move, and nothing about the order is public: not its side, its size, its limit, or even whether it is an order or an LP quote, since both post the same bond. The browser stores the order and the salt before the transaction is sent, and the user can export them to reveal from another device.

Orders come in two types:

Order	Sized in	Limit
Buy, market or limit on open	USDC to spend	none, or the most it will pay per share
Sell, market or limit on open	the xStock to sell	none, or the least it will accept per share

Buys are sized in USDC because the price does not exist yet. Locking what a buy will spend is the only safe way to size it. The number of shares follows from P_{open}.

Step 3: reveal

The reveal window runs from 30 minutes to 1 minute before the open by default, 09:00 to 09:29 ET on a normal day. The user reveals the order's fields and salt. The program recomputes the hash, checks the market minimum, and locks the amount in the user's balance.

The reveal needs no signature from the owner. The commitment already binds the owner and the lull, so knowing the salt is the authorization, and whoever reveals cannot change the order or redirect it. A user who will be offline can hand the reveal to a relayer. The operator never sees an order early either: the API builds the reveal transaction with every secret field zeroed, and the user's browser fills it in.

A ticket that is not revealed by the end of the window forfeits its bond to the market's treasury. The bond is what makes it costly to commit an order, watch the other reveals, and then decline to reveal.

Step 4: price

Just after the open, P_{open} is captured from Pyth. The rules, with default parameters:

- The price must be published at or after the open plus 30 seconds, and no later than 30 seconds after that.
- It must be fully verified by Pyth's receiver program, and it must belong to the market's feed.
- If its confidence interval is within 25 bps of the price, it is P_{open}.
- If the confidence interval is wider, the lull falls back to the mean of verified samples over 60 seconds, with at least three samples.
- If no valid price arrives within 2 minutes 30 seconds of the open, anyone can cancel the lull, and every revealed order is refunded in full.

Lull's crank goes further than the rules require. It finds the provably first Pyth update after the open plus delay on Hermes, which answers a timestamp with the last update at or before it, so the crank asks once about the end of the capture window and then searches the window (about five requests). It posts that update through Pyth's receiver with every Wormhole guardian signature

verified, and captures it in the same transaction. Each lull records a flag that says whether the captured price is provably the first update after the open plus delay.

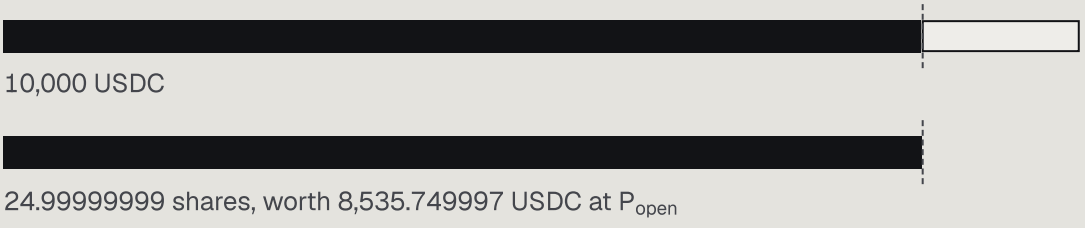
Step 5: cross and claim

With P_{open} known, the revealed book crosses once. Anyone can send the steps, and Lull's crank does. Each ticket is then claimed into its owner's balance. The fill and any unfilled remainder become free balance, and the bond and ticket rent return to the owner's wallet. Once every ticket is claimed or forfeited, anyone can close the lull account, and its rent returns to the market authority. The crank does this too, after saving the LP quotes and fills so the API can keep serving them.

Pricing and the cross in brief

Example inputs

Buys, total	Sells, total	P_{open} (USD per share)
10000 USDC	25 shares	$ 341.43



Heavy side	Buys. The sells fill in full.
Matched at P_{open}	8,535.749997 USDC for 24.99999999 shares Shared pro rata across the heavy side's orders, by amount.
Imbalance	1,464.250003 USDC of buys, about 4.28858038 shares at P_{open} LP quotes absorb what they can, cheapest first. The rest is refunded at claim.
px at P_{open}	3,425,461,389,515 Raw USDC per raw AAPLx × 10^{12}, with the multiplier already applied.

Example inputs. P_{open} starts at the latest Pyth price and can be changed. Every order is market on open and no LP quote is revealed. Shares convert to raw units with the AAPLx mint's live Scaled UI multiplier (1.00326901254), and the matching runs the SDK's mirror of the program's cross.

Classify. Each order is checked against P_{open}. A limit that P_{open} does not meet is refunded. A limit met at P_{open} takes part in matching. If the limit is also met at the worst revealed LP price, the order can take LP liquidity as well. Market-on-open orders always can.

Find the heavy side. Buys are heavy if their eligible USDC is at least the value of the eligible sells at P_{open}. Otherwise sells are heavy.

Match. The light side fills completely at P_{open}. The heavy side shares the matched volume pro rata, by amount. Fills therefore do not depend on when an order was revealed.

Absorb. What is left of the heavy side fills against LP quotes, cheapest spread first, each at its own price: P_{open} plus its spread when buyers are heavy, minus its spread when sellers are heavy. Anything left after that is refunded.

Settle exactly. Every amount is an integer in raw token units. Every payment rounds up and every receipt rounds down, so no token is ever created, and the dust, at most one raw unit per order, stays in the vault. The same math is written in Rust for the program and in TypeScript for the SDK, and the end-to-end test requires every on-chain fill to equal the SDK's prediction exactly.

Units. AAPLx uses Token-2022's Scaled UI Amount: wallets show the raw amount times a multiplier, which is currently about 1.00327 and changes over time. Lull reads the multiplier from the mint once, when it captures the price, and folds it into a single integer conversion rate. No step applies it twice, and nothing assumes it is 1.

AAPLx shares and raw base units, 8 decimals

Shares, as a wallet shows them		Raw base, as the program stores it
1 shares	⇄	99674163

`raw base = ⌊shares × 10^20 / M⌋` with M = `1,003,269,012,540`, a multiplier of 1.00326901254. Back in shares that is 0.99999999, because the raw amount is rounded down.

USDC and raw quote units, 6 decimals

USDC		Raw quote
1000 USDC	⇄	1000000000

`raw quote = USDC × 10^6`. Digits past the sixth decimal are dropped.

Example inputs, converted with the SDK's unit helpers. AAPLx uses the live Scaled UI multiplier of its mint, as the app does. Edit either side of a pair.

The result has a clear shape. The side that relieves the imbalance trades at exactly P_{open}. The side that creates it pays for its absorption, through rationing and LP spreads, within a worst price that

the market's spread cap bounds: at most P_{open} × 1.02 for a market-on-open buy, and at least P_{open} × 0.98 for a sell, with the default 200 bps cap.

Liquidity providers

Buys and sells rarely balance. LPs fill what is left of the heavy side. An LP commits a sealed quote with three fields: a spread, a maximum amount of the xStock to sell if buyers are heavy, and a maximum amount of USDC to spend if sellers are heavy. The quote is sealed like an order, posts the same bond, and is revealed in the same window. Both maximums are locked at reveal, and only the side opposite the heavy one can fill.

Quotes fill cheapest first, and each is paid its own spread. An LP always receives at least its quoted price, and it never fills beyond its maximums. Its reference is the official open price of the underlying, which it can observe and trade around, not a pool price from the middle of the night. A lull holds at most eight quotes, and the spread cap is 200 bps by default.

An LP commits before the book is visible. During the reveal window it can watch the imbalance form, but it can only decline to reveal, which forfeits its bond. Without LP quotes, the heavy side's remainder is refunded. For that reason the launch plan requires at least one market maker per market, with documented quoting obligations.

Why it is built this way

A few design choices do most of the work. Each one trades something away, and the trade is deliberate.

Pro rata, not time priority. Time priority would reward revealing early, and early reveals are exactly the information Lull tries to keep hidden for as long as it can. Under pro rata, an order's fill depends on its size and not on when it was revealed.

Two eligibility classes for limit orders. Letting each limit order take LP liquidity only at prices within its limit sounds natural, but the rule is circular: the LP prices that clear depend on which orders take part, and which orders take part depends on the LP prices. Lull uses the widest revealed spread instead, which is known before the cross begins. Every limit stays safe, and the cross runs in linear time, in batches. The cost is that some limit orders just beyond P_{open} take no LP liquidity, even when a cheap quote would have satisfied them.

One bond for every ticket. The bond is visible when a ticket is committed. A bond that grew with the order's size would reveal the size, and a different bond for LP quotes would reveal the kind of ticket. A uniform bond keeps the commit uninformative. It also means the bond is a floor on the cost of declining to reveal, not a full price for it, especially for large orders. Only encrypted orders remove that option entirely.

Cancellation over a bad price. When no trustworthy price arrives, nothing trades and everyone is refunded. For an order whose purpose is to trade at the open, a missed open is worse than a fill, but better than a fill at a stale price, a pool price or a price from the wrong moment.

Permissionless steps. Anyone can capture the price, tally the orders, finalize the cross, claim tickets into their owners' balances, and cancel a lull after the deadline. Lull's crank does all of this, but nothing depends on it. The worst outcome of an absent crank is a cancelled lull with full refunds.

Who can do what

Party	Can	Cannot
A user	deposit, withdraw free balance, commit, reveal, claim	move another user's balance, or change a sealed order
Anyone	reveal a ticket whose salt it holds; send every step after the reveal window	alter an order, redirect a claim, or use a price from outside the window
The market authority	publish lull times, change parameters within bounds, cancel a lull before it is priced	move user funds, change a sealed order, or set the price
The operator's API and worker	build unsigned transactions and run the crank	sign for users, see salts before a reveal, or move funds
The AAPLx issuer	move, pause or freeze AAPLx in any account, including the vault	affect USDC

The strongest power sits outside this table: whoever holds the program's upgrade authority can replace its code. It is also the only key that can open a market. A market's address depends only on its two mints, so the program checks the upgrade authority to keep anyone else from taking a market first. The plan is to move that authority to a multisig after deployment and later to governance with a timelock, so that users can see an upgrade coming and withdraw before it takes effect.

What is live today

Component	Status
Anchor program	Implemented and tested. **Not deployed on mainnet.** Deploying it needs the project owner's explicit approval.
SDK, worker, API	Implemented. The worker and API run on Railway in read-only mode against mainnet, serving the market, the Pyth-derived calendar, live Pyth prices and drift analytics.
Web app and docs	Implemented. Write flows run against a local stack (`npm run dev:stack`). The drift panel charts each lull's price series, and the cross replay plays the cross recorded by the end-to-end test on a local validator, labelled as such, until a lull crosses on mainnet.
Exact P_{open} capture	Implemented and tested against the mainnet Pyth and Wormhole receiver programs.
Encrypted orders (Arcium)	Designed, not built
$LULL, the protocol fee, staking, governance	Designed, not implemented
Security audit	Not done; a prerequisite for real funds

Five test gates pass. The first checks the calendar parser against real Pyth schedules and Pyth's own market hours. The second runs property tests of the crossing math over thousands of random books, in TypeScript and in Rust. The third runs an end-to-end cross on a local validator holding the real AAPLx and USDC mints and the real Pyth price account. The fourth tests the API against that stack. The fifth tests pricing through the mainnet Pyth and Wormhole receiver programs: the exact first price after the open, and the TWAP fallback, in which three real verified updates price a lull at their mean.

The end-to-end sample cross from 2026-09-24 shows the mechanism with real accounts. P_{open} was a verified Pyth price of $335.75101, and the multiplier read from the AAPLx mint was 1.00326901254. Two buyers spent 2,300 USDC, and a seller delivered 3 AAPLx shares at P_{open}. Two LPs absorbed the remaining 1,293 USDC of demand: the 15 bps quote filled in full before the 30 bps quote. A limit buy set about 5% below the pre-run price did not fill and was refunded. An unrevealed ticket forfeited its bond. The vault kept one raw unit of each token as rounding dust.

The read-only deployment also serves the live calendar. On 2026-09-26 it listed the next ten lulls through the open of 8 October, each with its reveal window, none of them on chain. The market endpoint reported the AAPLx multiplier, the latest Pyth price and Pyth's schedule string, with the market's on-chain parameters empty because the market does not exist on mainnet yet.

The honest limit of Stage 1 is privacy after the reveal. Reveals are public, so the imbalance is readable from the first reveal until the price is captured. With default parameters that is at most about 30.5 minutes, and Lull measures it on every lull. The price itself cannot be moved by anyone who reads it, because it comes from the underlying's regular session. A Stage 2 design encrypts orders with Arcium's multi-party computation, so that nothing is public before the cross. It is not built.

$LULL in brief

$LULL is a design. At the time of writing nothing in the Lull program references it, and none of its uses is implemented.

Launch and supply

Item	Design
Total supply	1,000,000,000, fixed, with no mint authority
Launch	fair launch on a bonding curve (pump.fun style)
Team allocation, presale, private round, vesting	none
Team holdings	only what the team's disclosed launch wallet buys on the curve, like anyone else, plus anything governance later allocates from protocol revenue

The team's launch wallet will be disclosed, and its purchases are visible on chain. This paper does not state how much the team will buy. The curve's parameters belong to the launch platform and will be stated at launch.

Fee and buyback

The design adds a protocol fee of 5 bps on the crossed notional of each filled order. It is a governance-adjustable market parameter under a cap compiled into the program. Orders pay it and LP fills do not, in the token the order receives. Fee revenue buys $LULL on the open market, and the bought tokens are burned by default. Governance may redirect a share to a treasury. Every collection, swap and burn is a public transaction.

The fee needs a program upgrade and an audit before activation. The buyback is not a distribution. Holders receive nothing, and its size depends entirely on volume through Lull, which is zero today. It can be paused or ended, and it may never start. There is no promise of price or return.

Staking for delegated roles

Reveal relayers and crank operators would stake $LULL to be listed as delegates in a registry. The program already allows relayed reveals, because the salt authorizes the reveal. A relayer that accepts a delegated reveal and misses it is slashed, and the slashed stake first compensates the affected user. Delegates may charge users a service fee. Staking earns no protocol yield. The registry is planned, not implemented.

Governance

Token-weighted votes through SPL Governance (Realms), behind a timelock, would control:

- market listings;
- market parameters: the fee, the bond, the open delay, the maximum price lag, the confidence limit, the TWAP window and samples, and the LP spread cap;
- the fee split and the treasury.

Today a market's authority would be a single key that the team controls. It can publish lull times, change parameters within bounds and cancel a lull before pricing. It cannot move user funds. Control moves in phases, from the team's key to a multisig and then to governance. The timelock should be longer than the longest lull, so that no change can land inside a lull users have already committed to.

Phases

1. **Launch** on a bonding curve.
2. **Fee activation**, after a program upgrade and an audit, with buybacks and public reporting.
3. **Staking registry** for relayers and crank operators, after its own audit.
4. **Governance handover** of the market authority, the treasury and the program's upgrade authority.

The phases are ordered by dependency, not by date. Phases 2 to 4 depend on the program being deployed and used, and any of them may change or never happen.

Risks

- **Smart contracts.** The program is unaudited and upgradeable. Until the upgrade authority moves to governance with a timelock, whoever holds it can replace the code.
- **Oracle.** Each cross depends on one Pyth feed. A wrong price that passes every check would be used, and an outage at the open cancels the lull.
- **Issuer.** The AAPLx issuer can move tokens out of any account, including Lull's vault, pause transfers and freeze accounts. xStocks are not offered to US persons.
- **Liquidity.** Without LP quotes, imbalances are refunded rather than filled.
- **Privacy.** Revealed orders and the imbalance are public for up to about 30.5 minutes before the cross. A lost salt or a missed reveal forfeits the bond.
- **Authority.** The market authority can change parameters in ways that make sealed orders unrevealable, and can cancel a lull before pricing.
- **Token.** $LULL gives no claim on anything. The fee, buyback, staking and governance are not implemented and may never be. Bonding-curve launches are volatile and open to automated buyers. Holdings and votes can be concentrated, and governance can be captured.
- **Eligibility.** The app asks each visitor once whether they are a US person and shows US persons a notice instead of the app. That is a self-declaration, not verification, and the program can be used without the app.
- **Regulation.** The treatment of tokenized equities and of tokens is uncertain and can change. A revenue-funded buyback in particular may be viewed by some regulators as creating an expectation of profit, and the design may change or be dropped for that reason.

Notice

This litepaper summarizes the Lull whitepaper, which contains the full specification, the measurements, the risks and the legal notice. It describes software and a token design. It is not an offer to sell, or a solicitation to buy, any security, token or financial instrument, and it is not investment, legal or tax advice. $LULL is designed as a utility and governance token. It carries no right to revenue, assets or profits, and nobody should acquire it expecting a profit. Nothing here promises a price, a return, or that any planned feature will ship. Lull and $LULL are not intended for US persons or for anyone in a jurisdiction where their use is restricted. Figures come from the project's repository, its recorded test runs, or its read-only API as fetched on 2026-09-26. Test figures do not indicate future results.