WHITEPAPER

1. Abstract

2. The problem

3. How LOWTIDE works

4. Architecture

5. Security and privacy model

6. Token mechanics ($TIDE)

7. What's next

8. References

Download PDF ↓

Roadmap →

Holder one-pager →

Docs →

Product information

LOWTIDE whitepaper

This paper describes the LOWTIDE design as set out in the product overview, backend design and build documents. It is a design paper: the service is in development and launches in phases (see the roadmap for what is live). Details the documents do not yet fix are marked *to be announced*.

Version: November 2026 draft. Chain: Robinhood Chain (chain id 4663). Settlement asset: USDG. Token: $TIDE, launching soon (contract `TBA at launch`).

Download PDF ↓

1. Abstract

LOWTIDE is an inference market for open-weight models served from bonded GPU hosts. Each host posts a price for every hour of the week. The router measures each host's load, so a quiet host's live price can drop to 0.4× its posted price, and a busy host's can rise to 1.2×. Buyers prepay USDG into a vault they can withdraw from. Agents can pay per call over x402, with payment channels first and `upto` as the fallback. Every response carries a usage record signed by the router and the host. Hourly Merkle roots of those records settle on chain. Hosts post

a bond and are audited for model swaps. Later phases add *Windows*, which are tradable reservations of future capacity, and a sealed tier on attested confidential hardware.

---

2. The problem

Paying for GPU time and turning it into useful output are different things. Published research shows that headroom exists, and that using it is hard:

- A two-month Microsoft training-cluster study measured 52.32% mean utilization of *allocated* GPUs. That is one system's efficiency gap. It does not measure capacity available for resale. [R1]
- SpotServe served language models on interruptible cloud machines. It reports up to 54% lower cost per token. Average latency rose by less than 18%, and 99th-percentile latency rose by 90%. [R2]
- ServerlessLLM reports 3.6–8.2× faster checkpoint loading than the PyTorch and Safetensors baselines it tested. That covers loading weights only, not token generation. [R3]

These are results from the cited studies, not LOWTIDE measurements. LOWTIDE has not shown how much idle supply it can reach, or what discount it can sustain.

The buyer side has its own problem. Cost-sensitive text work, like extraction, classification and summarization, can often wait for capacity. Today that work is bought at list prices with little visibility into what a request will cost, who served it, or whether the model was the one claimed.

LOWTIDE's test is whether spare capacity can produce useful outputs cheaply enough once serving overhead, payments, failed jobs and host

pay are counted.

---

3. How LOWTIDE works

1. **Hosts post prices.** A bonded host signs a price sheet with a base input and output price per million tokens for each of the 168 hours of the week, plus a floor. The pricebook rejects any sheet whose floor is above 0.6× its lowest hourly base price.
2. **The router measures load.** Live price = base price × load factor. The load factor is `clamp(0.4 + utilization, 0.4, 1.2)`, measured by the router, not the host. The price at the start of a request is the price billed, and a request is never re-priced mid-stream.
3. **Buyers set caps.** `X-Lowtide-Max-Price` caps what a request may cost. Interactive requests above the cap fail with `429 price_above_cap`. With `X-Lowtide-Flex: <deadline>`, the request becomes a night job that runs in a cheaper hour before the deadline at a price locked up front.
4. **The request runs on a host.** The router picks a healthy host and calls it through the host agent, which runs beside vLLM, checks model files against pinned digests and meters usage. If the first host fails before any token reaches the client, the router tries one more.
5. **Both sides sign the usage.** Each response ends with a usage envelope signed by the router and the host: host, model digest, token counts, price, and prompt and output hashes. Buyers can recount tokens with the model's public tokenizer.
6. **Hours close and settle.** After each hour the router builds a Merkle tree of that hour's records and signs the hour close. Three settlement signers check it, and two of the three must sign before the vault debits buyers and credits hosts.

Ways to pay

MODE	WHO IT'S FOR	HOW IT WORKS	PHASE
Prepaid vault	Apps and teams	Deposit USDG, use an API key, withdraw the unreserved balance at any time	P1
Payment channel	Agents (primary)	Open a channel, send signed vouchers after each response, 2-hour challenge period on a one-sided close	P2
x402 `upto`	Agents (fallback)	Offered when a request's maximum cost is at least 0.05 USDG. The router can stream up to 256 output tokens or 0.02 USDG before the on-chain authorization lands	P2
x402 `exact`	Known-cost calls (embeddings)	Pay the exact cost up front	P2
Window draw-down	Window holders	Requests draw down a reserved Window	P3

Fees

The buyer price is the host's live price plus an LOWTIDE fee of 4%. Buyers who opt in to audit sampling pay 1% less in total. Windows: the host's Window price plus 4%. Cancelling a Window before it starts

keeps 5% for the host, and resale costs 1% of the sale price plus 10% of the markup.

---

4. Architecture

LOWTIDE has two halves that meet at one frozen interface. That interface is the shared package of EIP-712 types, money math, Merkle rules and golden test vectors.

Serving and product

- **Router:** OpenAI-compatible API (`/v1/chat/completions` with streaming, tools and JSON mode, `/v1/embeddings` , `/v1/batches` , `/v1/models`), plus `/v1/prices` , `/v1/usage/:id` , `/v1/status` , channels, Windows and disputes.
- **Pricebook and load:** signed sheets, router-measured load, live prices and forecasts.
- **Scheduler:** night jobs at a locked price, spread across cheap hours.
- **Host agent:** runs beside vLLM, verifies model digests and GPU class, meters and signs usage.
- **Auditor:** likelihood audits, disguised canary requests, sequential tests and evidence bundles.
- **SDK and MCP tools:** TypeScript SDK with usage verification and payment helpers. MCP tools: `lowtide_complete` , `lowtide_prices` , `lowtide_schedule_job` , `lowtide_usage` .
- **Web app:** price page and wallet dashboard.

Chain and settlement (Solidity on Robinhood Chain)

- `LowtideVault` : buyer deposits, reserves, caps, withdrawals and withdraw-all.

- `HostRegistry`: host bonds, probation, unbonding and credits.
- `UsageLedger`: hourly usage roots.
- `ChannelManager` and `X402Escrow`: agent payments.
- `WindowMarket`: ERC-1155 capacity reservations, where 1 unit is 1,000,000 tokens.
- `AuditCourt`: challenges and rulings.
- `FeeSplitter` and `OfpkHooks`: fee distribution and the switched-off token hooks.
- Off-chain services: chain indexer, x402 facilitator, three settlement signers, keepers and a ruling service.

Stack: Bun, TypeScript, Hono, tRPC, Drizzle, Postgres with Timescale, Redis with BullMQ, viem, Foundry. Inference runs on vLLM's OpenAI-compatible server in a container pinned by digest. [R4]

---

5. Security and privacy model

Money safety

- **Withdrawals:** a buyer can always withdraw the balance minus the reserve. Withdraw-all completes within 2 hours. The router stops serving that buyer within 1 second of the request, and settlement can't debit any usage that started after it.
- **Settlement:** a 2-of-3 EIP-712 settlement with one settle per hour. Debits must equal the hour's root totals, and per-buyer hourly and daily caps apply. Host credits can be clawed back for 24 hours.
- **Channels:** every accepted voucher is stored immediately, so the latest one can be claimed during the 2-hour challenge period.
- **Beta cap:** deposits are capped at 500 USDG per buyer until an external contract audit is complete.

Honest hosts

- Hosts bond at least 2,000 USDG, or 20% of their trailing 30-day credits if that is higher, and serve through a 72-hour probation.
- The auditor samples served answers and scores them under the claimed model. It also runs hidden test requests that hosts can't tell apart from real traffic.
- A swap challenge needs two independent signals. A 100% slash for a proven model swap needs confirmation by a human review panel. Other breaches (throughput, metering, canary) are slashed at 20%, and a second strike delists the host.
- Audits are statistical. They catch sustained model swaps, not every single response.

Privacy

- Prompts and outputs are not stored, and request bodies are not logged. The exception is opt-in audit samples, which are encrypted and kept for up to 72 hours.
- Open hosts can read prompts, and so can LOWTIDE's router, unless you use the sealed tier.
- **Sealed tier:** hosts on Intel TDX with GPU attestation publish an HPKE key [R7]. The SDK checks the attestation and encrypts on the client, so neither the host operator nor LOWTIDE can read the prompt.
- On-chain roots contain hashes and totals, never prompts or outputs. Deposits and withdrawals are public token transfers that reveal wallet addresses, amounts and timing. This is pseudonymous, not anonymous.
- USDG can be frozen by its issuer. Permit2 is a shared dependency. [R6]

What a usage record proves: that the router and host signed those counts and prices, and, once settled, that the record is in a published hourly root. It does not prove that the answer is correct.

---

6. Token mechanics ($TIDE)

$TIDE is launching soon. It is a utility token. It is never a share of fees or a credit stream, and holding $TIDE alone earns nothing.

ITEM	DESIGN
Contract address	`TBA at launch`, launching soon, announced through the official channels only. Copy it from For holders.
Fee discount tiers	Buyers lock $TIDE to reach a tier that lowers the 4% fee. Default discounts are 0 / 0.25 / 0.50 / 1.00 percentage points across four tiers. Thresholds are timelocked settings, to be announced. Unlocking has a 7-day cooldown, and tiers count locked balance, not wallet balance.
Early Window access	Tier 1 and above get early access to Windows.
Host bond weight	Hosts can add $TIDE to their bond to rank higher in routing (bonus capped at 25%). Bonded $TIDE is slashable.
Fee split	80% to the work (audits, routing and settlement operations, launch hosting), 10% to the treasury, 5% to bonded hosts by bond and audited uptime, 5% burned.

ITEM	DESIGN
Burn	The 5% burn share buys $TIDE and burns it.
Legal gate	Token hooks and the burn switch on only after written legal review. Until then they stay off.

7. What's next

See the roadmap. In order, with no dates set:

1. **P1, endpoint and pricing:** the OpenAI-compatible endpoint, off-peak pricing, withdrawable balances, usage roots and host bonds, with 3 launch models on studio-run hosts, the first outside hosts, and mainnet protocol contracts under the 500 USDG beta cap after an external audit.
2. **P2, agents and night jobs:** payment channels, x402 `upto`, the scheduler and the MCP tools.
3. **P3, Windows and the sealed tier:** the Windows board and resale, sealed hosts, and audits at scale.
4. **Token hooks:** only after written legal review.

Each phase launches only when its gates pass.

8. References

- [R1] Jeon et al., *Analysis of Large-Scale Multi-Tenant GPU Clusters for DNN Training Workloads*, USENIX ATC 2019: https://www.usenix.org/system/files/atc19-jeon.pdf
- [R2] Miao et al., *SpotServe*, ASPLOS 2024: https://arxiv.org/pdf/2311.15566
- [R3] Fu et al., *ServerlessLLM*, OSDI 2024: https://www.usenix.org/system/files/osdi24-fu.pdf
- [R4] vLLM OpenAI-compatible server: https://docs.vllm.ai/en/stable/serving/online_serving/openai_compatible_server/
- [R5] Robinhood Chain docs (connecting, contracts, transaction finality): https://docs.robinhood.com/chain/connecting/ · https://docs.robinhood.com/chain/contracts/ · https://docs.robinhood.com/chain/transaction-finality/
- [R6] Permit2: https://github.com/Uniswap/permit2
- [R7] HPKE, RFC 9180: https://www.rfc-editor.org/rfc/rfc9180
- [R8] x402: https://github.com/x402-foundation/x402
- [R9] EIP-712: https://eips.ethereum.org/EIPS/eip-712 · ERC-1155: https://eips.ethereum.org/EIPS/eip-1155
- LOWTIDE product overview PDF (v1.0, October 12, 2026): /documents/LOWTIDE-Overview.pdf · Build PDF: /documents/LOWTIDE-Build.pdf

Third-party names describe compatibility only; no affiliation.

Workflows

Productivity

FAQ

Get started

Developers

Teams

Resources

Why spare GPUs?

Developer flow

Articles

Console features

Information

Docs

Whitepaper

Roadmap

For holders

Terms status

Privacy

Pricing

Explore

Getting started

Workflows

Overview PDF

Build PDF

Billing records

LOWTIDE
Product information

©2026

LOWTIDE