

HYPERGROK TRADING DESK

HyperGrok Trading Desk: a policy-gated Hyperliquid perpetuals desk

HyperGrok Trading Desk is an MIT-licensed fork of a Hyperliquid perpetuals desk, and the desk was commissioned by Omega Labs. The fork adds a policy gate, a kill switch and a testnet setting; the shipped strategy is at paper status, and mainnet use is not evidenced in the repository.

Published 21 October 2026 · Updated 21 October 2026 · spectrumweb.co/case-studies/hypergrok-trading-desk

01 / 12

The Problem

Letting an agent place orders on a perpetuals exchange needs written limits, a check that runs before any order is sent, and a way to halt the desk. The open question for an operator is what stops a wrong order from being sent.

02 / 12

What is HyperGrok Trading Desk, and who is it for?

HyperGrok Trading Desk is a Hyperliquid perpetuals desk built from seven role prompts, a deterministic strategy engine and a set of scripts. It is for an operator who wants agents to propose and place orders on a perpetuals exchange within limits the operator writes down in advance. The repository is an MIT-licensed fork maintained by SWC Studio. Omega Labs commissioned the desk. This page reports only what the repository shows.

03 / 12

What problem does it solve?

An agent that holds an exchange key can place orders its operator never intended. The repository answers that risk in code. The operator's limits sit in a file the operator edits. A policy gate checks each order against that file before the sender script runs, and a kill switch lets any role halt new risk. The README states the goal as a check that nothing is sent outside the operator's written limits, on the wrong network, into an unprotected book, past the loss stop, faster than the rate cap, or twice for one ticket.

04 / 12

How is the desk built?

The README describes seven roles: Desk Lead, Market Analyst, Research Analyst, Strategist, Risk Manager, Execution Trader and Trade Reviewer. The Execution Trader is the only role that holds the exchange key. Each trade follows one path: idea, evidence, risk sign-off, policy gate, one send, reconciliation, review.

The strategy engine is standard-library Python. Its rules file says it runs risk checks on a one-minute clock and decisions on a four-hour clock. Every ticket the engine produces goes through the policy gate and the sender, and the engine has no other path to the exchange. For the shipped strategy, the decision score combines momentum, carry and short-horizon reversion factors over liquid perpetual markets.

05 / 12

What controls does the code enforce?

- Written limits. `autonomy.json` holds the operator's limits. The shipped template sets 0.5% risk per trade, 2% total open risk, 3x leverage, three positions, a \$2,000 notional cap per trade, a 2% daily loss stop, four sends per hour and 20 per day, and 10 bps of slippage. It allows BTC, ETH and SOL only, and its mode is testnet.
- Fixed ceilings the policy cannot loosen. `scripts/policy_gate.py` holds ceilings of 1% risk per trade, 4% total open risk, 10x leverage, six positions, a 5% daily loss stop, 10 sends per hour, 60 per day and 50 bps of slippage. A policy that sets any value looser is invalid as a whole, and the gate refuses every ticket until the operator fixes it.
- A check on every ticket. The gate prints each check with its numbers. The checks cover the network, the halt file, ticket age, the market, the risk arithmetic, the account's current equity, risk per trade, protective stops, total open risk, the daily loss stop and the send rate caps. A failed read of the account is a refusal, not a pass.
- Protective stops. Before new risk is added, every open position must have a resting stop. The engine writes a protect ticket for any position that lacks one.
- Time limits on approvals. A passing ticket receives an HMAC-signed approval file that expires after ten minutes. Changing one byte of the ticket invalidates the file, and one send spends it. A ticket is refused once more than 30 minutes have passed since its risk check.
- One sender. `scripts/desk_send.py` refuses to send without a valid approval. It writes its intent to the send ledger before the request leaves the machine, sends once, and does not retry. An unknown result is reconciled by client order id, not resent.

- Halts. Any role can halt the desk, which writes a halt file. While the file is present, only reduce-only tickets pass the gate. The engine halts itself after three consecutive failed exchange reads. `kill_switch.py` resume requires a named human, and it refuses resume values that name an agent.
- Keys. The API wallet key is read from an environment variable, a secrets-manager command, or a file that must be mode 600 (or 400) and owned by the running user. The code reports which source it used and never prints the value.
- A risk ladder. The template halves the per-trade risk budget and the engine's gross exposure at a 3% drawdown from the equity peak, and quarters them at 6%. The ladder only shrinks these numbers.
- Claude Code guard. In Claude Code, `hooks/guard.py` refuses direct exchange requests, writes to `autonomy.json`, and writes that set status: live.

06 / 12

What does the strategy evidence show?

The shipped strategy, `hg-systematic`, is at paper status. Its rules say the engine trades it only while the policy is on testnet. The repository also ships two draft strategies, `hg-carry` and `hg-breakout`, which the engine does not trade.

`BACKTEST.md` is a simulation over 208 days of public hourly market data from Hyperliquid's public endpoint, with fees, slippage and stop fills modelled. The file itself warns against trusting its headline result. The README explains that changing the factor weights changed the simulated outcome sharply, and that a seven-month sample is one stretch of history. This case study quotes no figures from that file. Nothing in it predicts future results.

07 / 12

Current status

The shipped configuration is testnet with a paper strategy. Mainnet use is not evidenced in the repository.

- The policy template sets the mode to testnet, and the shipped strategy has status paper, which the engine trades only while the policy is on testnet.
- The checkout holds no paper record and no sends log, so it contains no recorded testnet run to inspect.
- Moving the desk to mainnet is an operator edit to `autonomy.json`, and the engine refuses paper strategies on mainnet (`engine/strategies.py:133`). Those are design facts, not evidence that mainnet has run.
- The README's worked example is a transcript. This case study does not treat it as a record of a run.

Omega Labs commissioned the desk. The status above describes the repository as checked; it is not a claim about how the desk has run.

08 / 12

Attribution

HyperGrok Autonomous Desk is an SWC Studio fork of HyperGrok Trading Desk by Andrew Wilkinson and Galleon Labs, released under the MIT license. The seven roles, skills and desk vocabulary are the work of the upstream authors. The policy gate, sender, kill switch, Claude Code guard, cycle runner, strategy engine, strategy gate, paper-record gate, re-review and the autonomy skills are the fork's additions, copyright SWC Studio, 2026. SWC Studio maintains the fork. The upstream copyright notice is retained under the same MIT terms. Attribution does not imply endorsement by either party.

09 / 12

Risk and disclaimer

The repository states that it is documentation and scripts, not financial advice. Perpetual futures can liquidate an account, and a desk that runs unattended can lose money up to the loss stop the operator writes. The repository contains no jurisdiction, identity-verification, securities or tax guidance.

10 / 12

Source notes

Paths are relative to the hypergrok repository, branch main at commit 4d915c1.

- README.md
- ATTRIBUTION.md
- template/autonomy.json
- template/strategies/hg-systematic/RULES.md
- template/strategies/hg-systematic/BACKTEST.md
- scripts/policy_gate.py
- scripts/kill_switch.py
- scripts/strategy_engine.py
- engine/secrets.py
- engine/strategies.py
- hooks/guard.py

11 / 12

Frequently Asked Questions

Is HyperGrok Trading Desk running with real money?

The repository does not show it. The policy template is set to testnet, the shipped strategy is at paper status, and the checkout holds no paper record and no sends log. Moving the policy to mainnet is an operator edit to autonomy.json.

Who wrote the upstream desk, and who maintains this fork?

HyperGrok Trading Desk is by Andrew Wilkinson and Galleon Labs, released under the MIT license. SWC Studio maintains the fork, which adds the policy gate, the sender, the kill switch and the strategy engine.

What stops an order that breaks the written limits?

The policy gate in scripts/policy_gate.py checks each ticket against autonomy.json, fixed ceilings, the halt file, the send ledger, the rate caps and the daily loss stop. It reads the account again before it passes a ticket, and a failed read counts as a refusal.

Is this financial advice?

No. The repository describes itself as documentation and scripts, not financial advice. Perpetual futures can liquidate an account.

12 / 12

Sources

HyperGrok Autonomous Desk README
hypergrok, README.md

Attribution notice
hypergrok, ATTRIBUTION.md

Autonomy policy template
hypergrok, template/autonomy.json

Systematic strategy rules
hypergrok, template/strategies/hg-systematic/RULES.md

Policy gate
hypergrok, scripts/policy_gate.py