

Battle Token Hook

Two tokens enter, one pool bleeds. Epoch based battle vaults on Uniswap v4 for Robinhood Chain. Integration guidelines for the Hookr team.



In partnership with Hookr (hookr.fun) for Uniswap v4 hook launches

Date: 2026-10-13
Repository commit: 810c9aaf668a | source hash: 49cdf5ea538b
Compiler: solc 0.8.26, Cancun, via IR (Foundry)
Tests: 22 unit
Publisher: SB (BVI) Ltd, StonkBrokers
Contact: simple@clutch.market, Telegram simplefarmer69

Status: LIVE on Robinhood Chain 4663 (redeployed 2026-10-13, HK-22 fix)

BattleVault 0x1BaDdAB75dd5223C2A619AA9c88BD211aaC5D304, battle StonkFeeHook 0x6ced3a0048F9f42836a90ee47FFe64645B0bAacc. This generation closes HK-22: the shared StonkFeeHook now reverts PartialFill(filled, expected) on any taxed buy that does not consume its full specified amount, so a price limited partial fill can never pay the fee on, or count Battle inflow for, quote that never reached the pool. The 2026-10-10 generation (0xc849c3683be5006c8E5B8eec4F3f60A9a855D6c8 / 0x9afEd91316E1da0334668Dedd3a1dac1d45c6aCc) is retired with zero launches; integrators must swap both addresses and both ABIs. This generation carries the Hookr partner fee leg: partnerBps 5000 (half of the StonkBrokers protocol leg) pointed at the Hookr Safe 0x8Fdfaa25e2E4dDa297F301ea06458f22186D6bE9 since 2026-10-11 (setPartner), so the Hookr share is paid on every flush through the permissionless flushPartner. Unit suite 22/22 plus 9/9 partner tests, full hooks ledger 162/162 (155 unit + 7 fork), audit findings HK-10, HK-12, HK-13, HK-18, HK-19 and HK-22 closed, Sourcify exact match. Ownership transfer to the company Safe 0x6D6981459be6EB09C778d98527bCCc1a235bbb26 is pending acceptOwnership().

Source and ABIs

Contracts live under hooks/ in the StonkBrokers repository. Byte exported ABIs ship beside this guide under contracts/hooks/integration/ and are served from stonkbroskers.io/integration/hooks/.

Deployed bytecode sizes

Contract	Role	Runtime size
BattleVault	Epoch vault, settle, war chest	20,496 B (83.4% of EIP-170)
BattleToken	Rival ERC-20 per side	1,845 B (7.5% of EIP-170)
StonkFeeHook	Shared fee / lock hook	5,860 B (23.8% of EIP-170)

Contents

- 1. What the hook does
- 2. Why this is not an owned root on Hookr
- 3. Addresses
- 4. Hook identity and hooklist entry
- 5. Pool keys
- 6. Swapping from the Hookr UX
- 7. Contract functions
- 8. Proofs
- 9. Residual risks (accepted)
- 10. Go live status

1. What the hook does

A creator seeds quote (WETH on Robinhood Chain) and names two rival tokens. The vault mints both supplies into itself, opens two full range v4 pools behind StonkFeeHook, and locks all liquidity in the vault. Nobody, including the owner or the creator, can withdraw pooled value.

Every epoch (1h to 7d, default 24h) the token with the larger net quote inflow wins. Settlement is permissionless:

1. A slice of the loser's locked liquidity is pulled (pullBps, capped at 90%).
2. Loser tokens are burned.
3. The quote side plus the round's war chest (buy and sell fees + decaying snipe tax) buys the winner and deepens the winner's locked liquidity.
4. Repeat until knockout (loser liquidity under floorBps of its initial, or maxRounds forces a final pull).

Fee policy: standing buy/sell fees (each at most 10%) and a decaying anti snipe tax feed the war chest. A protocol cut and a creator cut come out of the controller fee; half of the protocol cut (partnerBps 5000) is carved out for Hookr (section 3.1). The 1% pool LP fee accrues to the locked position. Outside LP is refused until the vault ends.

Threat model note: Robinhood Chain has no public mempool (Orbit FCFS). Frame mitigations as anti snipe economics and thin pool guards, never as sandwich protection.

2. Why this is not an owned root on Hookr

Hookr owned roots are unmodified Hookr roots. A custom hook cannot be an owned root, and HookrRouter / HookrQuoter / the Hookr SDK refuse pools whose hook is not a root. The Battle Token Hook is a standalone StonkFeeHook instance pointed at a BattleVault controller, so the integration path is the third party hook path:

- Analyze a Hook: public source at a pinned commit, the 20/20 unit suite (142/142 for the whole hooks/ tree), the exact PoolKey hook address, live readback of vault state.
- Publish a Hook: the hooklist entry (section 4) plus the pool keys (section 5).
- Partner Integration Review: this document and a walkthrough call if useful.

Swaps route through the Uniswap Universal Router with empty hookData. The Hookr UX can quote them with the Uniswap V4 Quoter and surface them in the hook directory while keeping its own router for root pools.

3. Addresses

Item	Chain	Address
BattleVault	Robinhood Chain 4663	0x1BaDdAB75dd5223C2A619AA9c88BD211aaC5D304
StonkFeeHook (battle instance)	Robinhood Chain 4663	0x6ced3a0048F9f42836a90ee47FFe64645B0bAacc
Uniswap v4 PoolManager	Robinhood Chain 4663	0x8366a39CC670B4001A1121B8F6A443A643e40951
Uniswap V4 Quoter	Robinhood Chain 4663	0x8dc178efb8111bb0973dd9d722ebff267c98f94
Universal Router 2.1.1	Robinhood Chain 4663	0x8876789976dEcBfCbBbe364623C63652db8C0904
WETH (quote)	Robinhood Chain 4663	0x0Bd7D308f8E1639FAB988df18A8011f41EAcAD73

```

Deploy parameters (state file hooks/deployments/battle-hook.4663.json): owner
0xD4aF10Bc56b08f8fc2990a3ADeeD4b757ae6A26C (pendingOwner = company Safe
0x6D6981459be6EB09C778d98527bCCc1a235bbb26), protocolTo = StonkBrokers treasury
0xb668382cF44038a3E8140E789060F6A809787CDa, protocolBps 1000, partnerTo = Hookr Safe
0x8Fdfaa25e2E4dDa297F301ea06458f22186D6bE9, partnerBps 5000, minSeedQuote 0.05 WETH, hook flags 10956
(0x2acc) mined through the Arachnid CREATE2 deployer. Deploy txs (2026-10-12): vault
0xa29b2376161434ad9b5a8fa0291e4565b312b52125a9fff20a1d0f766e21aa7a, hook
0x7e2fd1aed2c736926f57cfa4b57e259952127237cec923430f24afbe50322626, setHook
0x0f51928ec187cb7eca3aea533396947773966f5e588b0c743ed97e1bbdd3ac37, setPartner
0x90041d36e05753554aea77565efd8694da3f8bc8eef75bb7ba5ef77b8b6cb7fb, transferOwnership
0x0a0e99039276d708e717ae743de5f9aad405685fa20da0fa214bc612120dfe0. Verification: Sourcify exact match for vault
and hook (https://sourcify.dev/server/v2/contract/4663/0x1BaDdAB75dd5223C2A619AA9c88BD211aaC5D304).
```

Deployed size: BattleVault 20,496 B (83.4% of EIP-170), StonkFeeHook 5,860 B.

3.1 Hookr fee leg

Every taxed swap on a battle pool splits the controller fee into three legs inside StonkFeeHook.afterSwap:

1. $\text{protocolCut} = \text{fee} \times \text{protocolBps} / 10000$ (10% today).
2. $\text{partnerCut} = \text{protocolCut} \times \text{partnerBps} / 10000$ is carved OUT of the protocol cut for Hookr (5000 bps = half of the protocol cut = 5% of the controller fee at today's settings). The StonkBrokers treasury receives the remaining half.
3. The remainder goes to the controller (BattleVault.onSwap), which pays the creator cut and funds the war chest.

Where the partner leg lands depends on partnerTo on the vault (set to the Hookr Safe at deploy): when it is set, poolManager.take(quote, partnerTo, partnerCut) pays Hookr inside the swap; while it is zero the hook takes the carve to itself and books it under partnerAccrued[quote]. Anyone can then call flushPartner(quote) on the hook once the owner has set the recipient, and the whole accrual is paid out in one transfer. Nothing is burned and nothing is skipped: the carve exists from the first swap, only the destination waits on the Hookr CA.

The Hookr CA is wired: partnerTo = 0x8Fdfaa25e2E4dDa297F301ea06458f22186D6bE9 at 5000 bps, so every taxed swap pays Hookr inside the swap and nothing accrues. Repointing it is one owner transaction BattleVault.setPartner(<newCA>, 5000), followed by StonkFeeHook.flushPartner(<quote>) from anyone if anything accrued while the recipient was zero. The partner share can be tuned later with another setPartner, bounded at 10000 (the whole protocol cut). Events to index: PartnerFeeTaken(poolId, quote, partnerCut, to) on every taxed swap (to is the live recipient or zero when accrued), PartnerFlushed(quote, to, amount) on release, PartnerSet(to, bps) on the controller.

4. Hook identity and hooklist entry

StonkFeeHook flags (low 14 bits must match):

Flag	Set
BEFORE_INITIALIZE	yes
BEFORE_ADD_LIQUIDITY	yes
BEFORE_REMOVE_LIQUIDITY	yes
BEFORE_SWAP	yes
AFTER_SWAP	yes
BEFORE_SWAP_RETURNS_DELTA	yes
AFTER_SWAP_RETURNS_DELTA	yes
everything else	no

Hooklist entry:

```
{
  "name": "Battle Token Hook",
  "author": "StonkBrokers",
  "address": "0x6ced3a0048F9f42836a90ee47FFe64645B0bAacc",
  "chainId": 4663,
  "dynamicFee": false,
  "upgradeable": false,
  "requiresCustomSwapData": false,
  "vanillaSwap": true,
  "beforeSwapReturnsDelta": true,
  "afterSwapReturnsDelta": true,
  "source": "https://github.com/Clutch-L4bs/stonkbrosers/tree/main/hooks",
  "docs": "StonkBrokers-Battle-Token-Hook-Hookr-Integration.pdf",
  "tags": ["battle", "epoch", "burn", "robinhood-chain"]
}
```

Notes for the directory card:

- Custom swap data: none. Pass empty hookData.

- Upgradeable: no. Owner levers are parameter bounds only (protocol recipient, partner recipient and share, min seed).
- Liquidity: controller only while the battle runs. Outside add/remove reverts LiquidityLocked.

5. Pool keys

Each battle opens two pools (token A / quote and token B / quote):

Field	Value
currency0 / currency1	sorted by address
fee	10000 (1%)
tickSpacing	200
hooks	StonkFeeHook (battle instance)

Creation: BattleVault.createBattle(BattleParams) seeds quote, deploys both tokens, initializes both pools, and locks liquidity. Pool id is keccak256(abi.encode(key)).

6. Swapping from the Hookr UX

Vanilla from the trader side. Quote with Uniswap V4 Quoter quoteExactInputSingle and empty hookData. Exact output swaps revert while a snipe tax is live (ExactOutputWhileTaxed). Between epoch end and settle, the vault may enforce a trading curfew via the hook.

Partial fills (HK-22): a taxed BUY must fill in full. The hook skims the buy fee from the specified quote input before the pool runs, and v4 only lets afterSwap return a delta in the unspecified currency, so an over skimmed quote fee can never be refunded. afterSwap therefore checks that the pool consumed exactly quoteIn - fee and reverts PartialFill(filled, expected) otherwise. Battle pools are full range and controller owned, so an ordinary buy always fills; only a buy with a sqrtPriceLimitX96 inside the current range, or one that exhausts the pool, trips it. Do not pass a price limit on taxed buys from the Hookr UX (use MIN_SQRT_PRICE + 1 / MAX_SQRT_PRICE - 1 as the limit and bound the trade with the router's minAmountOut instead). Sells are unaffected: their fee and the Battle inflow are counted on the realized quote output.

7. Contract functions

What each public surface does. Arguments match the shipped ABIs (BattleVault.abi.json, BattleToken.abi.json, StonkFeeHook.abi.json). Pool lookups use PoolId = keccak256(abi.encode(key)).

7.1 BattleVault

Launch and settle:

Function	Who	What it does
createBattle(params)	creator	Pulls seedQuote (approve vault first), mints two BattleToken supplies, opens both full range pools behind the battle StonkFeeHook, locks all liquidity in the vault. Returns battleId. Reverts HookMismatch, NativeNotSupported (quote must be ERC-20, not native ETH), SeedTooSmall, BadParams. Emits BattleCreated.
settleable(battleId)	anyone	true when the battle exists, is not ended, and block.timestamp >= epochEnd.
settle(battleId)	anyone	Permissionless epoch settle. Picks the winner by net quote inflow (tie = blockhash coin flip), pulls up to pullBps of loser liquidity (or all at knockout / maxRounds), burns loser tokens, buys the winner with buyShareBps of the budget, locks the rest as winner LP, rolls leftover quote into the next war chest. Emits BattleSettled. Reverts UnknownBattle, BattleEnded, EpochRunning.

Read surface (desk / Hookr analyze):

Function	Who	What it does
battle(id)	anyone	Full Battle struct: creator, quote, tokens[2], pools[2], netIn[2], warChest, epoch clocks, fees, snipe schedule, round, ended, champion, initialLiquidity.
currentSnipeBps(id)	anyone	Live decaying anti snipe tax in bps (linear from snipeStartBps to 0 over snipeDecaySeconds from epochStart). Zero when ended.

Function	Who	What it does
leader(id)	anyone	(side, margin) for the current epoch: which side leads by net quote and by how much.
battleOf(poolId) / sideOf(poolId)	anyone	Reverse lookups from a pool to its battle id and side (0 or 1).
battleCount / minSeedQuote / protocolTo / protocolBps / partnerTo / partnerBps	anyone	Global counters and the two fee destinations (StonkBrokers protocol leg, Hookr partner leg).
feeTerms(poolId)	StonkFeeHook	Controller callback. Returns buy/sell/snipe/protocol terms. Sets paused=true between epochEnd and settle (trading curfew). Outside LP always locked (lpOpen=false) until the battle ends.
onSwap(poolId, feeToController, quoteIn, quoteOut)	StonkFeeHook	Updates net inflow, pays creator cut, funds war chest. Emits WarChestFunded.

Owner (fee destination only; no path to pooled quote, war chest, or locked LP):

Function	Who	What it does
setProtocol(to, bps)	owner	Protocol cut of the controller fee (bps capped on chain). Emits ProtocolSet.
setPartner(to, bps)	owner	Hookr partner leg: bps of the PROTOCOL cut (not of the whole fee) is carved out for to. Default at deploy: to = 0x0, bps = 5000 (half of the protocol cut). While to is zero the hook accrues the carve in partnerAccrued; once Hookr supplies its CA the owner calls setPartner(hookrCA, 5000) and every later swap pays Hookr directly. Reverts PartnerBpsTooHigh above 10000. Emits PartnerSet.
setMinSeedQuote(v)	owner	Floor on createBattle seed. Emits MinSeedSet.
setHook(hook)	owner	One time wire of the battle StonkFeeHook instance (PoolController).
transferOwnership / acceptOwnership	owner / pending	Two step Ownable.

7.2 BattleToken

Plain ERC-20 per side. Full supply mints into the vault at create. Public surface beyond ERC-20:

Function	Who	What it does
burn(amount)	holder	Burns tokens from the caller. Used by the vault at settle to destroy pulled loser inventory.

7.3 StonkFeeHook (battle instance)

Shared fee / lock hook. Immutables: poolManager, controller (the BattleVault). Permissions: beforeInitialize, beforeAdd/RemoveLiquidity, beforeSwap, afterSwap (+ return deltas).

Function	Who	What it does
getHookPermissions()	anyone	Returns the Uniswap Hooks permissions struct. Pure.
beforeInitialize	PoolManager	Only the controller may initialize. Reverts NotController.
beforeAddLiquidity / beforeRemoveLiquidity	PoolManager	Outside LP blocked while feeTerms.lpOpen is false (LiquidityLocked). Controller always passes.
beforeSwap	PoolManager	Reads feeTerms. Controller swaps pass free. Reverts TradingPaused under settle curfew, ExactOutputWhileTaxed when a tax is live and the swap is exact out. Buy path skims fee from specified quote input via BeforeSwapDelta.
afterSwap (buy path)	PoolManager	Reverts PartialFill(filled, expected) when the pool consumed less than quoteIn - fee (price limited or liquidity exhausted buy), so the fee and the Battle inflow are never charged on quote that did not trade.
afterSwap	PoolManager	Sells skim fee from unspecified quote output. Splits the protocol cut to protocolTo, carves partnerBps of it out for partnerTo (or accrues it on the hook while unset), sends the remainder to the controller, then controller.onSwap. Emits FeeTaken, ProtocolFeeTaken (StonkBrokers leg only) and PartnerFeeTaken.

Function	Who	What it does
flushPartner(quote)	anyone	Permissionless. Pays the partner carve accrued while partnerTo was unset to the controller's current partnerTo. Reverts PartnerUnset while the controller still reports a zero partner. Emits PartnerFlushed(quote, to, amount).
partnerAccrued(quote)	anyone	Partner carve parked on the hook for quote (nonzero only while partnerTo was zero at swap time).

Unused after* / donate callbacks revert HookNotUsed.

8. Proofs

Unit suite hooks/test/Battle.t.sol 22/22 on the deployed bytecode: create, buy/sell fees, snipe decay, settle pull + winner buy + lock, knockout, max rounds, tie break, outside LP blocked, owner no value path, curfew, multi battle isolation, plus the audit regressions `test_buy_creatorCutFoldsIntoWarChestWhenBlocked`, `test_exactOutputRefused_zeroFeeBattle`, `test_exactOutputRefusedWhileTaxed`, `test_create_rejectsBadParams`, `test_settle_knockoutLeavesNoldleValue`, `test_hk22_partialFillBuyReverts` and `test_hk22_partialFillSellStillWorks`. Partner leg: hooks/test/Partner.t.sol 9/9 (carve math wei exact, accrual while unset, flushPartner payout, PartnerUnset, PartnerBpsTooHigh, owner only, ProtocolFeeTaken reports the StonkBrokers leg only). Whole hooks/ suite 162/162 (155 unit + 7 mainnet fork).

Audit closure (HK-10 to HK-19 fixed before the 2026-10-10 deploy, HK-22 fixed with the 2026-10-12 redeploy; report docs/Hashlock-Certified-Audit-StonkBrokers-Hooks.pdf):

- HK-10: the ExactOutputWhileTaxed guard in StonkFeeHook.beforeSwap sat below the zero fee early return, so an exact output swap on a zero fee pool slipped past it. The check now runs first.
- HK-12: a knockout settlement could leave idle quote and idle token on the vault. The winner buy now spends the whole leftover quote, then any idle winner tokens and the loser side seed dust are burned, so the vault holds nothing after a knockout.
- HK-13: snipeStartBps + max(buyFeeBps, sellFeeBps) >= 10000 is rejected at createBattle (BadParams), a 100% tax can no longer be configured.
- HK-18 (was H-3): the creator fee leg was a hard `_safeTransfer`, so a reverting creator wallet bricked every taxed swap on that battle. onSwap now pushes through the tolerant `_tryTransfer` in PoolController; a failed push folds the creator cut into the war chest.
- HK-19 (was M-1 / M-2): `_reservedQuoteExcept` walked every battle on each settle and the winner buy accepted a zero fill. The vault tracks warChestByQuote per quote token for an O(1) reserve read and the settle winner buy passes minOut = 1.
- HK-17 (was H-1 / H-2) does not apply to this stack (no holder reward stream).
- HK-22 (High, found 2026-10-12 while double checking the Hookr review items): StonkFeeHook charged the buy fee and BattleVault.onSwap booked netIn on the SPECIFIED quote amount, not on what the pool actually consumed. A price limited buy that specified 50 ETH and filled about 0.0005 ETH paid about 1.5 ETH of fee and still counted 50 ETH of inflow for its side, which would have decided the epoch for dust. afterSwap now requires the pool to consume exactly quoteIn - fee and reverts PartialFill(filled, expected) otherwise (fail closed, because an over skimmed quote fee cannot be refunded through the unspecified delta). Pinned by `test_hk22_partialFillBuyReverts` and `test_hk22_partialFillSellStillWorks`. The vault and hook were redeployed; the 2026-10-10 generation had zero launches.

9. Residual risks (accepted)

- Thin quote pools at settlement size can move the winner mark; settlement is permissionless and retryable.
- Creator chosen names/symbols can collide with other tokens; wallets must key on address.
- Hookr SDK paths that require a root hook will not route these pools; use Universal Router + V4 Quoter.

10. Go live status

1. Redeployed 2026-10-13 via BattleHookDeploy.s.sol on the HK-22 fix bytecode (PHASE=deploy, PHASE=verify readback passed, transferOwnership to the company Safe sent). Three earlier generations with zero launches are superseded and archived: `0x2B90...1db7` / `0x477C...aAcC` (battle-hook.4663.superseded-2026-10-10.json), `0x4Cf7...25C9` / `0xBd78...EAcc` (battle-hook.4663.superseded-2026-10-10b.json) and `0xc849...D6c8` / `0x9afE...6aCc`

(battle-hook.4663.superseded-2026-10-13.json). Never list those addresses. Done.

5. Hookr CA: partnerTo = Hookr Safe 0x8Fdfaa25e2E4dDa297F301ea06458f22186D6bE9 at 5000 bps, set at deploy. Done.
6. Company Safe 0x6D6981459be6EB09C778d98527bCCc1a235bbb26 calls acceptOwnership() on the new vault. Pending on the Safe signers.
2. Addresses filled in above, PDF regenerated. Done.
3. Hooklist entry is final (section 4). Pool keys follow the first live battle, read them off BattleCreated events on the vault.
4. Permissionless: anyone can create a battle. StonkBrokers runs no privileged path to escrowed quote.