StonkBrokers Launchpad Trading Integration

Version: 2026-08-25 (rev 8 -- terminal teams only)

Part 1: Arbitrum One (chainId 42161) - Part 2: Robinhood Chain (chainId 4663)

Audience: GMGN, BasedBot, FOMO, and other trading terminals wiring buy/sell, quotes, charts, and token discovery for tokens launched on StonkBrokers launchpads.

Downloads (from stonkbrokers.io/integration/ or stonkbrokers.wtf/integration/) -- the SAME two ABIs work on BOTH chains:

- StonkSafeLaunchpadV2.abi.json -- curve-phase writes on each pad
- SafeLaunchLensV2.abi.json -- all reads and quotes
- StonkBrokers-Launchpad.pdf -- this guide (section 9 is the full ABI reference: selectors, event topics, revert selectors)

Storage rule (both chains): launch ids are per pad. Always key (chainId, padAddress, launchId). Never key on bare id or token address alone.

PART 1 -- Arbitrum One (pre-bond trading)

1. Arbitrum One deployment

Chain id 42161 - Explorer: https://arbiscan.io

Lens (all reads): 0x7376f9dC6432434D611488CB3E852071ed29E276

Status: the pads are deployed and fully configured with final addresses, and are currently sealed (the launch fee is set prohibitive, so no launches exist yet). Opening is a single owner transaction -- wire your integration against these addresses now and trading starts the moment the first launch arms. Every quote/trade call below behaves identically to the Robinhood pads in Part 2.

1.1 Lane pads (12)

Each pad runs its bonding curve in the lane's quote token. quoteIn and quoteOut amounts are denominated in that quote token's decimals.

Lane	Pad address	Quote token (Arbitrum One)	Dec	Native ETH
WETH	0x9540AC4173E5A8c7970743bd13aEc5E9e97FcD22	0x82aF49447D8a07e3bd95BD0d56f35241523fBab1	18	yes -- buyEth / sell(..., ethOut=true)
USDC	0x42a561Bf35E311A59492ECe678315949a112D9ef	0xaf88d065e77c8cC2239327C5EDb3A432268e5831	6	no
rNVDA	0xea69D73A62Bd25644b2d164BA1A56F521aeA8616	0x0a85b6EBe3C150A1979a65656E304033D62b1E12	18	no
rTSLA	0x58329852104FFeA8103E8377E0DcBF5098667682	0xf912911C9c8D5131929c758e66e6dc54e65cF3ba	18	no
rMSFT	0x1338a39f86824A5a3a6282EEd3Ac399681600D63	0xdA7c9549E87D4F96585701970AcDCc90E8d35024	18	no
rGOOGL	0x4Ad64B9b5990b37616bd297Ab1Cc11C3fbB2FAed	0xcA1942EdEB697A85AF98eF8C74C4910fe9587afB	18	no
rAMZN	0xDBFf2b590B9b04b2A75D6d5249B666C3c373bdAb	0xfba9E358F9AD443236155E5C67f2aADe032Db7A1	18	no
rMETA	0x238d85f92a1B2346C0262254B9E3d233C1ad32BE	0xB65f3005aB2aEede3707F56223C7c45f68cca008	18	no
rCOIN	0x6fce80BCe616aBd45a29fb286E5c86EfE989140D	0x412Ab56d122343dE6ABeaCceC646699BacD85601	18	no
rSPY	0xEECf01bac1BA354451AA2852727ebC304B2e5bA2	0x9a0fda21276F245b89d2B7f1120F42e4466b1cb5	18	no
RAAPL	0x009c43dD3C002B02965e0327ef35da0515c7c746	0xaBa5e0C80e9f58E391214689fB342049C0d31892	18	no
RSPCX	0xbc9Bc56D0018b9c1c71aaB74e917c89CC97A637E	0x5181b7Dd097B42d7787ee78Efab86f43D4E12f44	18	no

The r-prefixed quote tokens are Reality Protocol tokenized stocks (Arbiscan reality label). Our UX applies a US courtesy gate on

stock-quoted lanes; terminals apply their own jurisdiction policy.

1.2 Arbitrum specifics

- Discovery is on-chain only for now. Page the lens per pad:
viewLaunches(pad, startId, count) (pages DOWN ids) and subscribe to the pad events listed in section 5. The HTTP floor API (section 5) adds Arbitrum lanes when the site cutover lands.
- Pre-bond trading is the whole of section 6 -- quoteBuy /
quoteSell on the lens, buy / buyEth / sell on the pad. Function signatures, recipient semantics, setOperator, referral ref, and the boughtOf sell cap are byte-identical to the Robinhood pads.
- Approvals: on quoted lanes approve the PAD for the lane's quote
token, then call buy. The WETH lane takes native ETH via buyEth (msg.value), no approval needed.
- USDC lane is 6 decimals. Never assume 18 when formatting
quoteIn / quoteOut on that pad.
- After bond: locked pools are Uniswap v3 (1%) on Arbitrum One --
there is no up. venue there. Read the pool address from the LaunchBonded / LegBonded events and route post-bond swaps through your standard Uniswap v3 router against that pair.

PART 2 -- Robinhood Chain (the StonkBrokers UX chain)

Chain id 4663 - Explorer: https://robinhoodchain.blockscout.com

2. Launch surfaces on Robinhood Chain

Surface	When users trade	Contract you call
Smart Launch (V2 + V3 pads)	Token is on the bonding curve (pre-bond)	Pad + lens (sections 3-6)
Smart Launch (bonded)	Curve graduated and bonded	Standard DEX swap against the locked pool (section 6)
Stonk Launcher (factory)	Token is on its factory curve (pre-grad)	Per-token StonkCurvePool (section 7)
Stonk Launcher (graduated)	Curve hit graduation threshold	Standard DEX swap against locked LP (section 7)

Pad routing (Smart Launch): minted tokens use V2 pads (section 3). External-token (BYO) launches use V3 pads (section 4). Same ABI on every pad -- and the same ABI as the Arbitrum pads in Part 1.

---

3. Smart Launch V2 pads (mint & standard launches)

Lane	Pad address	Quote asset	Native ETH
WETH	0xFCd61B25BbF3AbD6cf0070D6328E351cc30EEC9f	WETH	yes -- buyEth / sell(..., ethOut=true)
STONK	0x8f6782c5Aa37804d08a9b7bf3984Ff3245Fd6cD4	$STONKBROKER	no
USDG	0xd4F20033586977A2511f4A2DB4aF7C79a340D70a	USDG	no
GME	0x4B9Dcd6CCFAeF0f6D23065Dd78E79d5E20ec8cFD	GME stock token	no
NVDA	0xEe96d955d5634813374ecE4C74F2C0ff71B1F9fB	NVDA stock token	no
AAPL	0xB0453A81Cbf963903409FFF18AD92941e1c7a864	AAPL stock token	no
SPCX	0x0c3b4EDED41696eFF0ed70841f132B519d81c947	SPCX stock token	no
USO	0xDb3C81C841ff88db6cDFbDDB0eE049D162A6053B	USO stock token	no
yBTC	0x472a1AB6aEB77E3479193FBe83b718f9Bf5F8604	TickerYard yBTC (8 decimals)	no

Lens (all reads): 0x25b5Df581f4b2Ed450203f375ad8A28b17F115B3

Scan every Robinhood V2 + V3 pad when indexing -- a token may exist on any lane. The yBTC lane prices via TWAP on the live yBTC/WETH up. pool (0x176A43Aa3B295383b3fd4bc758c6Ca97ED205b1b); quote amounts are 8 decimals.

---

4. Smart Launch V3 pads (external-token launches only)

Same StonkSafeLaunchpadV2 ABI and the same lens as V2.

Lane	Pad address
WETH	0x5BCEefBa6fDf437A7388aDC5c9056c827baca3B3
STONK	0x406fd0B957bb8cF1dd57C78540D009578e971131
USDG	0xF0A06Ac7BBb0cc3049B68c257c3ee27CcEA40eeA
GME	0x5b21F8a5Ef81586627B4725844aD447325d0992B
NVDA	0xDf03953DCA8dB733345278A0c5fd2E81fa2A9B54
AAPL	0xc522DfaE0D1a140257702392B665183a6De7657f
SPCX	0xd82da1D8ef59959b170b59147283Ab1F2F1Ca86A
USO	0x644b19512052A1b6d38d7B16C6c3Fb1d3F7270D2
yBTC	0x2bd7F90cca4660da82aa693cF352dDB6275C76dA

---

5. Discover tokens & live state

HTTP (fastest bootstrap -- Robinhood lanes today, Arbitrum after cutover)

Method	Route	Use
GET	/api/safe-launch/floor	Full launch grid: token CA, pad, id, phase, mcap, logo, quote lane
GET	/api/safe-launch/buys	Live buy tape (SafeBuy logs) for alerts
GET	/api/safe-launch/token-logo?tokens=0x...	Batch logo lookup (<=60 CAs)

Base URL: https://stonkbrokers.io or https://stonkbrokers.wtf

On-chain (canonical, identical on both chains)

```
// SafeLaunchLensV2 -- 0x25b5...115B3 on Robinhood, 0x7376...E276 on Arbitrum
viewLaunch(address pad, uint256 id) -> LaunchView
viewLaunches(address pad, uint256 startId, uint256 count) -> LaunchView[]  // pages DOWN ids
launchIdOfToken(address pad, address token) -> uint256 id   // 0 = none on this pad
```

Indexer events (subscribe on each pad address, both chains):

LaunchCreated, LaunchArmed, SafeBuy, SafeSell, CurveClosed, LaunchBonded, LaunchAborted

Each LaunchView includes the token CA, quote asset, phase flags, tax schedule fields, and core.bonded.

Phase -> is the pad tradeable?

Phase	Trade on pad?
waiting (!armed)	no
buffer / live window / degen trading	yes (curve)
closing (graduated && !bonded)	no -- bond pending
bonded	no -- use DEX (section 6)
aborted	no

Derive phase from LaunchView.core timestamps and flags (same logic as the /api/safe-launch/floor snapshot).

---

6. Smart Launch -- quote & trade (curve phase, both chains)

Always quote through the lens. Tax (snipe shield, buffer, flat post-tax) changes every minute; do not reimplement curve math locally.

```
// Reads -- SafeLaunchLensV2
```

```
quoteBuy(address pad, uint256 id, uint256 quoteIn)
    -> (uint256 tokensOut, uint16 taxBps)
quoteSell(address pad, uint256 id, address seller, uint256 tokensIn)
    -> (uint256 quoteOut, uint16 taxBps)
priceWei(address pad, uint256 id) -> uint256   // spot before tax

// Writes -- StonkSafeLaunchpadV2 @ pad
buy(uint256 id, uint256 quoteIn, uint256 minTokensOut, bytes32 ref, address recipient)
    -> uint256 tokensOut
buyEth(uint256 id, uint256 minTokensOut, bytes32 ref, address recipient) payable
    -> uint256 tokensOut                        // WETH lanes only
sell(uint256 id, uint256 tokensIn, uint256 minQuoteOut, bytes32 ref, address account, bool ethOut)
    -> uint256 quoteOut
setOperator(uint256 id, address operator, bool approved)  // per launch
```

Terminal integration notes

- No global EOA gate unless the launch set eoaOnly (then NotEoa).
Routers, zaps, and smart wallets work by default.
- recipient buys: quote is pulled from msg.sender; tokens and
sell-credit (boughtOf) land on recipient (0 -> msg.sender). Lets your router buy for a user without their quote approval.
- setOperator + sell: your router can sell on a user's behalf if the
user approved your contract as operator for that launch. Proceeds always pay account, never the operator.
- ref: pass bytes32(0) if you do not use referrals. A referral code
only applies when the caller is the recipient or an approved operator.
- Sell cap: boughtOf limits sells to tokens bought on the curve
(SellExceedsCurve). Transferred bags cannot be sold through the pad.
- WETH lanes (both chains): use buyEth with msg.value for native
ETH; set ethOut=true on sell to unwrap proceeds.
- Quoted lanes (STONK, USDG, USDC, stocks): approve the pad for the
quote token, then call buy. User must hold the lane's quote asset -- amounts are in the quote token's decimals (USDC = 6).

After bond (post-curve)

When core.bonded == true, stop calling the pad. Swaps route through the locked token/quote pool minted at bond -- Slipstream CL or Uniswap v3 on Robinhood, Uniswap v3 on Arbitrum One -- read the pool address from LaunchBonded / LegBonded events or the floor API's dexPair field. Use your standard router for that chain against the pool pair.

Live mcap after bond on Robinhood: prefer DexScreener (tokens/v1/robinhood/{CA}) -- trading often migrates to deeper third-party pools than the locked pool.

---

7. Stonk Launcher factory (Robinhood Chain only)

Factory: 0x80a77001456bc986083678F9a112B1EC2Aa07281 - index from block 34876725

Each launch() deploys a token + its own StonkCurvePool.

Discover

Method	Route
GET	/api/launcher/tokens?sort=new
GET	/api/launcher/token/<tokenCA>
GET	/api/launcher/trades?token=0x...

Factory events: Launch (token + pool addresses). Map token -> pool via poolOf(token) on the factory.

Curve-phase trade (pre-graduation)

```
// StonkCurvePool @ pool from factory
buy(uint256 minOut) payable
sell(uint256 amount, uint256 minOut)
```

EOA-only on the factory curve (msg.sender == tx.origin) unless you route through a contract that the pool accepts (most terminals use direct wallet swaps).

After graduation

Call graduate() on the pool when threshold is met (permissionless). Post-grad swaps use the locked LP pair on up. / Uniswap v3 -- same pattern as bonded Smart Launch tokens.

---

8. Common trading reverts (Smart Launch pads, both chains)

Revert	Meaning
NotEoa	Launch has eoaOnly; contract caller blocked
SellExceedsCurve	Sell amount exceeds boughtOf
BuyCapExceeded	Recipient hit per-wallet buy cap (maxBuyPpm)
WindowClosed / NotGraduatable	Curve closed; use DEX if bonded
StalePrice	Stock-lane oracle stale (weekend gaps); pad trade blocked

Use eth_call with the lens quote as minOut with your slippage bps.

---

9. Smart contract ABI reference

Extracted from the deployed ABIs (StonkSafeLaunchpadV2.abi.json, SafeLaunchLensV2.abi.json). Identical bytecode generation on both chains -- every selector and topic below is valid on Robinhood Chain and Arbitrum One.

9.1 SafeLaunchLensV2 -- reads & quotes

Selector	Function
0x36ccaf5d	viewLaunch(address pad, uint256 id) -> LaunchView
0xa2fe271d	viewLaunches(address pad, uint256 startId, uint256 count) -> LaunchView[]
0xe257598e	quoteBuy(address pad, uint256 id, uint256 quoteIn) -> (uint256 tokensOut, uint256 taxBps)
0x23be86e8	quoteSell(address pad, uint256 id, address seller, uint256 tokensIn) -> (uint256 quoteOut, uint256 taxBps)
0xdaaac0fb	priceWei(address pad, uint256 id) -> uint256
0x994aabf0	quoteUsdView(address pad) -> (uint256 usd8, bool fresh)
0x144f2a5d	ethUsdView(address pad) -> (uint256 usd8, bool fresh)

9.2 LaunchView field map

viewLaunch returns (id, core, taxBps, mcapUsd8Now, tokensSold, oracleFresh, legs[], pools[], lockIds[], lpQuote, lpFeeBpsSnap, closedAtTs, bufferSecs, unsoldMode, eoaOnly, openEnded, postTaxBps, bondVenue, maxBuyPpm, icoBoost)

core tuple: (token, creator, startMcapUsd8, gradMcapUsd8, startTaxBps, decayPerMinuteBps, creatorFeeBpsSnap, protocolFeeBpsSnap, windowSecs, startTime, deadline, externalToken, sellsEnabled, armed, graduated, bonded, aborted, loadedSupply, vQuote, vToken, realQuote, buyCount)

Phase derivation: !armed = waiting - armed and inside startTime + bufferSecs + windowSecs (or openEnded) = tradeable - graduated && !bonded = bond pending - bonded = trade the DEX pool - aborted = dead.

9.3 StonkSafeLaunchpadV2 -- trading writes

Selector	Function
0x89bdadac	buy(uint256 id, uint256 quoteIn, uint256 minTokensOut, bytes32 ref, address recipient) -> uint256 tokensOut
0xde159148	buyEth(uint256 id, uint256 minTokensOut, bytes32 ref, address recipient) payable -> uint256 tokensOut (WETH lanes)
0xf99df3d4	sell(uint256 id, uint256 tokensIn, uint256 minQuoteOut, bytes32 ref, address account, bool ethOut) -> uint256 quoteOut
0x16d4b2ed	setOperator(uint256 id, address operator, bool approved)

0xf776449a	graduate(uint256 id) -- permissionless crank
0x9940686e	bond(uint256 id) -- permissionless crank

9.4 StonkSafeLaunchpadV2 -- key reads

Selector	Function
0xa27d865a	launchIdOfToken(address token) -> uint256 (0 = none on this pad)
0x27cca59f	launchCount() -> uint256
0x48e12e45	boughtOf(uint256 id, address wallet) -> uint256 (sell cap)
0x2aa3dade	currentTaxBps(uint256 id) -> uint256
0x05512f86	mcapUsd8(uint256 id) -> uint256
0x5930d3ce	getLaunch(uint256 id) -> core tuple
0x999b93af	quote() -> address (the lane's quote token)
0x3fd1e2bd	quoteDecimals() -> uint8
0x91ec06a9	isWethLane() -> bool
0x8fe6f94e	closedAt(uint256 id) -> uint64
0x231dd7f6	poolsOf(uint256 id) -> (address[] pools, uint256[] lockIds)
0x85c7604e	legsOf(uint256 id) -> address[]
0xbcd577a9	operatorOf(address owner, uint256 id, address operator) -> bool
0x83eeb3bb	launchFeeWei() -> uint256

Owner/admin functions (set*, rescue*, launch creation) are omitted -- terminals never call them.

9.5 Indexer events (topic0)

topic0	Event
0xa4d3b2dc54656491295ce21a0b888cd8df38e69113e44371344da5088e480a43	LaunchCreated(uint256 indexed id, address indexed token, address indexed creator, bool externalToken)
0x617dff9af409b81e92edd8d62fb192f25509657b96d203585e3e6b605d4dea26	LaunchArmed(uint256 indexed id, uint256 supply, uint256 vQuote0, uint64 quoteUsd8, uint64 deadline)
0xba22b06917da96d20a8f4f80d45cbdaaf3294856de78268558edcce22e4298df	SafeBuy(uint256 indexed id, address indexed buyer, uint256 quoteIn, uint256 taxPaid, uint256 taxBps, uint256 tokensOut, uint256 mcapUsd8)
0x2de6d6d1573ee69658d3daae2e752379e6eb0676622a5ade2812088d7cb56581	SafeSell(uint256 indexed id, address indexed seller, uint256 tokensIn, uint256 taxPaid, uint256 taxBps, uint256 quoteOut, uint256 mcapUsd8)
0x35438c0a652764ec4e53a7c6ea24f69d38de411205cb46565a621d1ecec1b4f0	CurveClosed(uint256 indexed id, uint256 raisedQuote, uint256 finalPriceQuoteWei, uint256 mcapUsd8)
0x93a0dc53b22eaff6d32706900ad4faf8945af529c8178e4556a9cba1b688cc87	LaunchBonded(uint256 indexed id, uint256 raisedQuote, uint256 burnedTokens)
0x6106e5df757fa888b432e238e2d1872551de9281e95c84fbcd8a51f512f3a103	LegBonded(uint256 indexed id, address indexed asset, address pool, uint256 lockTokenId, uint256 legQuote)
0x9d64b00dad0b0e3deb20c57ce627e54741af4ed4e4074f04355225f391e1b7a1	UnsoldLpBonded(uint256 indexed id, address pool, uint256 lockTokenId, uint256 tokensPlaced)
0x2dc77d4f2c209c8529f5a2c5230d46dfc0d6b51745614f40599076c187a6fb71	LpFeeBonded(uint256 indexed id, uint256 bondedQuoteWei, uint256 excessQuoteWei)
0x580dfdb1b506406bf92732277b3b526bdf395c999ba181ddd0c576be4b95dc8d	LaunchAborted(uint256 indexed id, uint256 tokensSwept)

LegBonded.pool / UnsoldLpBonded.pool are the locked DEX pools to route post-bond swaps through.

9.6 Revert selectors

Selector	Error	Selector	Error
0x4b28f84a	NotEoa()	0xd8c90bbc	SellExceedsCurve()
0x12341846	BuyCapExceeded()	0xc3d4bf52	SellsDisabled()
0x2f23cfcd	WindowClosed()	0x8199f5f3	SlippageExceeded()
0x9699d9bc	NotGraduatable()	0x4ef36a18	ZeroTrade()

`0x8b4b52de`	`NotClosed()`	`0xf6806bf8`	`UnknownLaunch()`
`0x8db2750a`	`NotArmed()`	`0xf401a420`	`AbortedLaunch()`
`0xe6a0d45f`	`AlreadyGraduated()`	`0xacf04168`	`AlreadyBonded()`
`0x7c214f04`	`NotOperator()`	`0x93687c0b`	`NotCreator()`
`0xbdcdd37d`	`NotWethLane()`	`0x759401d7`	`NothingOwed()`
`0xfaa5e57f`	`FeeOnTransferToken()`	`0x207583fd`	`TokenInUse()`
`0xceaba46f`	`InsufficientLaunchFee()`	`0x3ee5aeb5`	`ReentrancyGuardReentrantCall()`

StalePrice surfaces on stock-quoted lanes through the lens quote reverting -- treat any lens quote revert as "pad not tradeable right now".