

CRYPTO OTC ROOM • PRODUCT & ARCHITECTURE

How a trade actually settles, and what sits underneath it.

Hashlock Markets • October 2026. The trade lifecycle, the interfaces a desk touches, the services behind them and the security model. Contract-level detail is in Smart Contract Specification.

01 • LIFECYCLE

Four steps, and only two ways it can end

Step	What happens
Negotiate	A sealed RFQ or a direct quote, over encrypted messaging inside the desk's existing Telegram flow
Lock	The initiator locks funds in an HTLC; the counterparty locks the other leg against the same hash
Settle	The preimage is revealed and funds release atomically on both chains
Or refund	If a leg is never completed, the timelock returns funds to both sides. There is no half-completed state

Each HTLC follows a deterministic five-state machine — PENDING → ACTIVE → WITHDRAWN | REFUNDED | EXPIRED — and the dual-HTLC trade coordinates two instances, one per chain leg, through the swap statuses `agreed` → `initiator_funded` → `counterparty_funded` → `initiator_claimed` → `counterparty_claimed`, with `refunding`, `refunded`, `expired` and `failed` as the other outcomes.

Finality-aware timelocks. The counterparty leg is always shorter than the initiator leg by at least a 2-hour cross-chain finality buffer, **which guarantees the initiator can always claim before their own contract expires**. The 30-minute figure that also appears in the contracts is the per-contract minimum timelock each programmable rail enforces, not the gap between the legs. The asymmetry is what makes the guarantee hold in practice, not just on paper.

02 • PRODUCTS

What a desk can settle

OTC settlement: a sealed RFQ plus HTLC atomic settlement, with zero counterparty risk on the leg. A swap has exactly two legs.

03 • INTERFACES

Where the desk actually works

- **Web app** — the desk-facing interface: RFQs, negotiation threads and settlement.
- **REST API** — the `/v1` API, with webhooks and a maker WebSocket feed at `wss://<host>/v1/ws`, for desks that want to drive settlement programmatically.
- **MCP server** — hosted at `https://hashlock.markets/mcp`, or run locally over stdio, for agent-driven access.
- **Telegram bot** — a distribution channel, and where negotiation already happens, [which is the reason adoption costs a desk no workflow change](#).

Screenshots are in Product Screens. Live endpoints are internal and shared under NDA.

04 • SERVICES

The architecture behind it

Hashlock runs as a modular microservice architecture. Settlement happens on-chain in the HTLC contracts; the off-chain services orchestrate the trade lifecycle and watch the chains. Funds are never held by Hashlock — they are locked in on-chain HTLCs at all times.

Service	Responsibility
API gateway	JWT auth, rate limiting, CORS
Trade service	RFQ lifecycle, direct trades; the trade state machine
Keeper	Submits claims and refunds where they are permissionless — EVM, Tron and Solana. It pays gas and never receives principal: a claim it submits pays the leg's fixed recipient, a refund pays the funder. On Bitcoin it cannot act — a spend needs the owner's signature, so the receiver claims that leg.
Chain watcher	Monitors Bitcoin, Ethereum, Base, Robinhood Chain, Tron and Solana for HTLC lock, claim and refund events; detects preimage reveals; reorg detection
Messaging service	Negotiation threads — quotes, counter-prices and the terms each side accepted
Solver service	Planned. Intent aggregation and batch broadcasting, for the solver network — see Roadmap & Milestones for the launch order. Activates after the next contract generation and its audit sign-off

05 • SECURITY

The security model

- **Customer fund safety** — funds locked in on-chain HTLCs, never custodied. [The dual-HTLC flow guarantees both legs complete or both refund](#). The initiator generates the secret on its own machine and only `sha256(secret)` ever reaches Hashlock, until the initiator reveals it on chain by claiming. The fund, claim and refund endpoints return unsigned transactions; the user signs them with their own key or HSM.
- **Financial precision** — Decimal.js in the application and BigInt on-chain; NUMERIC database columns, never FLOAT; per-token decimal enforcement (ETH 18, USDC 6, BTC 8).
- **Three-tier authorization for AI-assisted development** — a control on how Hashlock's own code is changed, not on customer funds. Tier 3, covering settlement and custody-adjacent code, is human-only. Tier 2, features and schema, requires human approval. Tier 1, tests and monitoring, is autonomous.
- **Audit hardening** — 2 internal audit rounds complete, 5-layer hardening (Slither, Halmos, Echidna, Stryker, fast-check) and a runtime invariant monitor. The external audit with QuillAudits is complete (October 2026).

06 • COVERAGE

Chains and onboarding

HTLCs are implemented across Bitcoin, Ethereum, Base, Robinhood Chain, Tron and Solana. Same-chain and cross-chain trades settle under one atomic guarantee, with no wrapping and no bridge. The one pair not offered is TRON↔TRON: one side of a Tron trade must sit on another chain. [Settlement is live on mainnet across all six chains](#), with ETH↔ETH and ETH↔BTC verified end to end; the Bitcoin leg runs as a P2WSH script, live on mainnet, requiring no contract deployment.

Institutional onboarding follows a fixed path: Demo → Sandbox Pilot → Limited Production → Full Onboard.