



HTLC CONTRACTS · EVM, TRON, BITCOIN SCRIPT & SOLANA SPL

Smart Contract Specifications

Hashlock Markets — HTLC contracts across EVM, Tron, Bitcoin Script and Solana SPL.

Hashlock Markets Confidential

1. Contract Surface

Rail	Escrow	Claim	Refund
Ethereum · Base · Robinhood Chain	<code>HTLCFactory</code> — one CREATE2 minimal-proxy clone per swap	<code>claim(secret)</code>	<code>refund()</code> after timelock
Tron	<code>SharedHTLC</code> — one shared pool, one storage slot per swap	<code>claim(secret)</code>	<code>refund()</code> after timelock
Solana	HTLC program — one PDA per escrow, separate classic-SPL and native-SOL paths	<code>claim</code> / <code>claim_sol</code>	<code>refund</code> / <code>refund_sol</code>
Bitcoin	No contract — a P2WSH hashlock + timelock script per leg	Witness spend revealing the preimage (<code>OP_SHA256</code>)	<code>OP_CHECKLOCKTIMEVERIFY</code> refund path

There is no administrative path to escrowed funds on any rail. The protocol fee is routed to a `feeRecipient`. **On a Bitcoin leg the fee is paid on that leg, in satoshi** — the fund builder returns `feePayTo` and `feeAmountSats` alongside the escrow payment, and the leg does not count as funded until both are paid from the same wallet. A fee below the dust limit of 546 satoshi is waived. Verification: 54 / 54 Forge tests on the EVM and Tron rails, and the ETH↔BTC atomic swap is live on mainnet.

Same-chain and cross-chain trades settle under one atomic guarantee — **a swap either completes on both chains or refunds on both, with no bridge or wrapped assets.**

Per-rail notes

EVM — Ethereum, Base, Robinhood Chain. The swap parameters are immutable args in the clone's bytecode, so **there is no `initialize()` to front-run** and no window in which a deployed-but-unconfigured escrow exists. The clone address is deterministic, so a counterparty can compute and check the exact escrow before funding. Native coin and ERC-20 are both supported — `token = address(0)` means native; an ERC-20 leg is an `approve` followed by `createSwap`. Every funding transaction must be signed from the leg's refund address: the factory refuses any other sender, and the escrow refunds to it.

Tron. One pool holds every asset, TRC-20 and native TRX. A TRC-20 leg is `approve(pool, amount + fee)` then `fund(...)`; a native TRX leg is `fund(...)` alone, carrying `amount + fee` as its call value. Claim and refund take the on-chain swap id the pool emitted at funding. **When the pool is rotated, new swaps go to the new pool and the old one keeps honouring claim and refund until its timelocks lapse** — no migration, no stranded escrow.

Solana. Each escrow is a PDA derived from the agreed terms, so an escrow funded on any other terms is a different address. Classic SPL tokens and native SOL are separate instructions; **Token-2022 mints are not supported**. Escrow accounts are never closed, so a settled swap cannot be funded again — and the funder pays the account rent, which is not returned.

Bitcoin. Each side's settlement address on a Bitcoin leg is a compressed public key; the P2WSH is derived once both keys are set. Claim and refund are prepared as a PSBT with one sighash per swept input, and the spend pays the P2WPKH of the key in the branch being spent.

2. The HTLC Primitive

An HTLC locks funds under two conditions: a cryptographic hashlock (SHA-256 of a secret preimage) and a timelock. **The preimage is exactly 32 bytes on every rail** — the EVM and Tron claim functions take `bytes32` and the Solana program enforces the same length, so one secret opens both legs of any pair. SHA-256 rather than keccak256 because Bitcoin Script offers `OP_SHA256` and no keccak. **The EVM factory, the Tron pool and the Solana program each refuse a timelock shorter than 30 minutes**, so no escrow can be created with a window too short for the counterparty to see the funding, read the preimage off the other leg and be confirmed. The recipient claims by revealing the preimage before the timelock expires; otherwise the sender refunds. **This produces a binary, trustless outcome.**

PENDING → ACTIVE → WITHDRAWN | REFUNDED | EXPIRED. The three end states are mutually exclusive: WITHDRAWN means the preimage was revealed and the recipient claimed, REFUNDED means the timelock passed and the funder took the principal back, EXPIRED means the timelock passed with the refund not yet taken. **Exactly one path executes; an escrow cannot be left neither claimable nor refundable.**

3. Fee & Rebate Mechanics

The fee-enabled HTLC applies a volume-tiered fee, routed to a `feeRecipient`, with a rebate parameter that returns a share of the fee to referrers. The taker pays it together with the funding, on top of the leg they fund and in that leg's asset, and it is not refunded if the deal does not complete. **Fees are sub-10 bps (8 bps in the plan; the protocol fee is live and charged)** and scale down with volume; fee logic is enforced in-contract — settlement and fee capture are atomic.

4. Contract Hardening & Invariants

Core invariants hold across every leg: **funds cannot be double-claimed, and funds cannot be lost once locked** — either the preimage is revealed and both legs settle, or the timelocks expire and both legs refund. On EVM and Tron, 54/54 Forge tests (Forge is EVM-only; the Solana SPL program is live on mainnet and not covered by this suite) plus 5-layer hardening (Slither, Halmos, Echidna, Stryker, fast-check) and a runtime invariant monitor cover the contract surface. On Bitcoin, the P2WSH script is minimal by design — the hashlock and timelock are the only spend paths, reducing correctness to standard, auditable script primitives. 2 internal audit rounds are complete, and the QuillAudits external audit is complete (October 2026) — \$10K of external audit spend plus a \$10K bug bounty pool, \$20K in total, already spent.

5. Versioning & Audit Gating

V1 mainnet HTLC contracts are immutable. Improvements — including the Compute Power Exchange module — ship to V2, which is gated on external audit-firm sign-off before mainnet deploy.