SettleKit

Open-source checkout, settled onchain — now with real $wif support on Robinhood Chain

World is Flat Hackathon — tools and infra

caelum0x/settlekit · worldisflat.fun/build · deadline 2026-11-12 17:30 UTC

The problem

Software sellers — founders, small teams, API/tool builders — want to get paid in the token their buyers already hold, without:

- Standing up a custom payment processor per chain
- Trusting a black-box payment gateway with their funds and their users' funds
- Hand-building delivery logic for every product type (repo access, license keys, API keys, Discord roles, files, webhooks)

And on **Robinhood Chain** specifically: there is no native USDC. Sellers who want to accept the chain's own community token, **$wif**, have no standard way to verify a real onchain $wif transfer and release access automatically.

Who has this problem

- **Solo founders & small API/tool teams** selling a starter kit, a Pro license, or dataset access
- **Small SaaS teams** who want crypto rails without building a payments team
- **$wif holders on Robinhood Chain** who want somewhere real to spend it — not just trade it
- **AI agents** that need to pay for access programmatically (x402), not click a checkout button

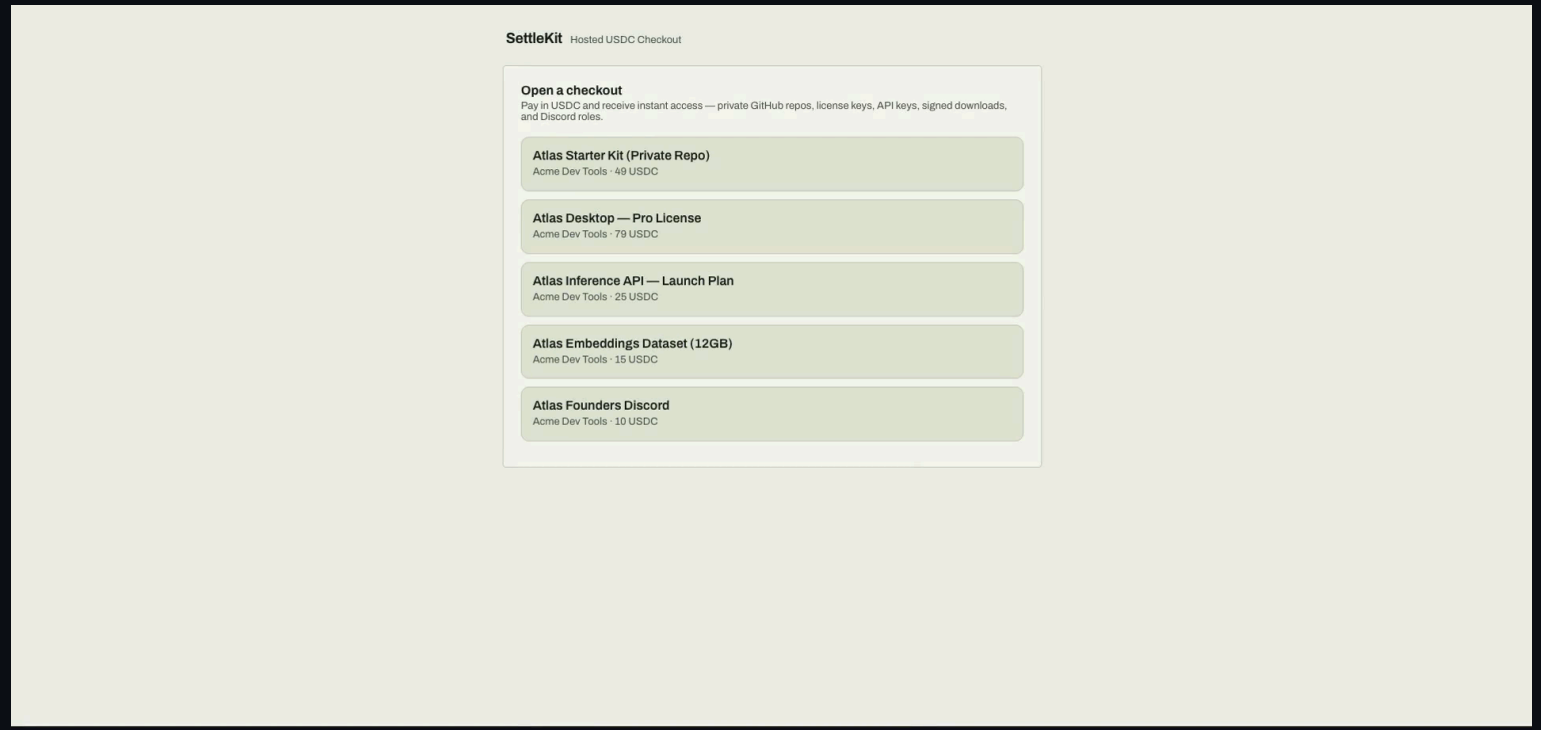
The solution: SettleKit

One link. Any chain. Fail-closed verification. Automatic delivery.

```
Create product -> Share link -> Buyer (human or AI agent) pays on any chain
               -> SettleKit verifies the real onchain transfer
               -> Access delivered: repo invite, license key, API key, file, webhook
```

- Open source, self-hostable, **1% per payment**, no subscription fee
- Already live on Solana, Ethereum, Base, Arbitrum, Hyperliquid, Tempo, Zcash
- **This hackathon**: Robinhood Chain gets a second settlement asset — $wif — alongside its existing USDG support

Live proof — the real hosted checkout

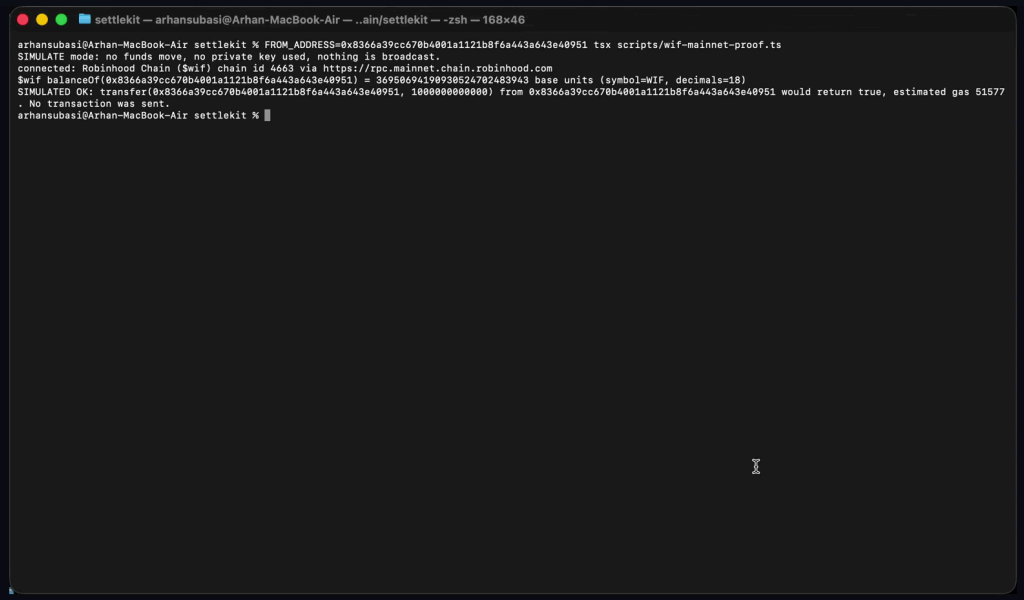


Live checkout: settlekit-checkout.vercel.app

Real product list, HTTP 200

`settlekit-checkout.vercel.app` — **HTTP 200, verified live** (`curl`, 2026-11-10/11)

Live proof — real $wif mainnet read, right now

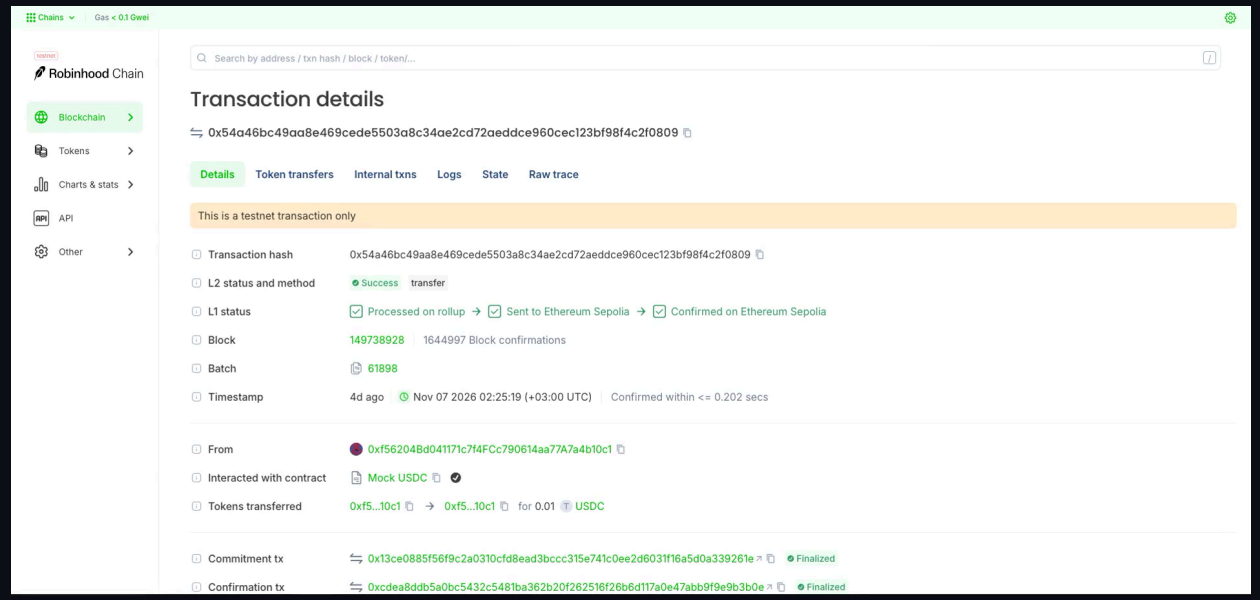


scripts/wif-mainnet-proof.ts — SIMULATE mode

Real run against Robinhood Chain mainnet, just now

Real `eth_call` against **Robinhood Chain mainnet** (`chain id 4663`): contract `0x5173b8e…54773` , symbol `WIF` , 18 decimals, a real balance read, a real simulated `transfer()` — no private key, nothing broadcast.

Live proof — a real, already-settled onchain transaction



Real, verified settlement — Robinhood Chain TESTNET
tx 0x54a46bc4…f0809, status Success (testnet, not $wif)

Robinhood Chain **testnet**, tx `0x54a46bc4…f0809`, status **Success**, block `149738928` — triple-verified (own verifier, direct RPC, public explorer API). Testnet Mock USDC, not $wif — see the honest status slide.

How it works

```
Buyer / agent
    |  pays on Robinhood Chain (USDG or $wif)
    v
Chain registry (packages/chains/src/registry.ts)
    - robinhood (USDG, EIP-3009 direct)
    - robinhoodWif (new, 18 decimals, Permit2-only, experimental)
    v
Fail-closed EVM verifier (packages/chains/src/evm-verifier.ts)
    - reads the real onchain Transfer event
    - checks token, amount, payer, confirmations
    - same code path apps/worker runs in production
    v
Access delivered: repo invite / license key / API key / file / webhook
    v
Public proof at /proof — every confirmed payment, with explorer links
```

No custom receiver contract. $wif settles as a plain ERC-20 transfer, the same pattern as every other EVM chain SettleKit supports.

$wif — what's real today

Shipped this pass (commit `d82a1b4` on `main`):

- `robinhoodWif` added to the chain registry — real mainnet contract, 18 decimals
- x402 facilitator entry: `transferMethod: "permit2"`, honestly marked `experimental`
  (verified: $wif's live bytecode has no EIP-3009/EIP-2612)
- 194/194 `packages/chains` + `packages/x402-facilitator` tests pass with the new invariants
- `scripts/wif-mainnet-proof.ts` — SIMULATE run live against mainnet (shown above);
  LIVE mode ready, gated on a funded owner wallet

Honest gap, found while recording the demo: the checkout UI's network picker isn't extended for $wif yet — a buyer can't select it in that UI.

Why this fits World is Flat

Rule	Status
Rule 3 — use $wif onchain on Robinhood Chain	$wif is live in the registry + verifier; mainnet SIMULATE proof run and verified; **real LIVE tx hash slot below**, pending a funded owner wallet
Rule 4 — stays live, judges will try it	Checkout UI is live (HTTP 200): `settlekit-checkout.vercel.app`. The separate API (`settlekit-api.onrender.com`) is down — disclosed, not hidden, never shown as live
Rule 5 — demo video required	65s narrated demo, real captures only, ships with this deck
Category fit — **tools and infra**	SettleKit *is* payment infra: an SDK + hosted checkout + fail-closed verifier other builders can run their own product on

Real mainnet $wif settlement

```
tx hash:   <TX_HASH>
explorer:  https://robinhoodchain.blockscout.com/tx/<TX_HASH>
verified:  SettleKit's own createEvmVerifier + direct eth_getTransactionReceipt
```

Not yet run as of this deck's build (2026-11-11). Requires the owner's own funded Robinhood Chain mainnet wallet (`LIVE=1 PAYER_PRIVATE_KEY=0x... tsx scripts/wif-mainnet-proof.ts` , real cost: a fraction of a cent of $wif + gas). Until this slide is filled, treat $wif settlement as **proven in SIMULATE only**, not yet executed live — stated here exactly that way, not inflated.

Honest traction

- **Revenue: $0.** This entry has not been monetized — stated plainly, not inflated.
- **194/194** `packages/chains` + `packages/x402-facilitator` tests passing
- **1 real, triple-verified onchain settlement** on Robinhood Chain testnet
- **1 real, live SIMULATE call** against Robinhood Chain mainnet's deployed $wif contract
- SettleKit already settles real payments across 7 chains/assets in production code (Solana, Ethereum, Base, Arbitrum, Hyperliquid, Tempo, Zcash) — this hackathon adds an 8th settlement asset, not a new product

Roadmap

1. Owner runs `LIVE=1` mainnet $wif proof → fills the tx-hash slot above
2. Wire `$wif` into the checkout UI's own `PaymentNetwork` picker (`packages/common`)
   so a buyer can select it end to end, not just through the API
3. Reconnect `settlekit-api.onrender.com` to a live `DATABASE_URL` (owner-only Render dashboard step — code-side timeout/boot-hang fix already shipped)
4. More $wif-priced products listed publicly at `/proof`

Team & links

- **Repo**: https://github.com/caelum0x/settlekit (public, MIT-style open source)
- **Live checkout**: https://settlekit-checkout.vercel.app
- **Demo video**: `docs/worldisflat/media/settlekit-wif-demo.mp4` (this repo)
- **Submission docs**: `docs/worldisflat/SUBMISSION.md`

Built solo, open source, self-hostable — contributions welcome.