COUNTERSIGN · WHITE PAPER · V1.0 · JUNE 2026

The control plane for AI agents that spend money.

A neutral, cross-vendor layer that holds one policy, one freeze, and one tamper-evident ledger across every agent-wallet backend at once — the one thing no single wallet vendor can do.

THE ONE FALSIFIABLE TEST THAT DEFINES V1

Can Countersign **freeze agents across many backends at once, in under a second, with a unified tamper-evident ledger of every attempt?** This paper, and the open repository behind it, answer yes — proven live across four rails in ~432 ms on testnet.

Category · cross-vendor agent-spend governance **Status** · testnet · open core (Apache-2.0)

countersign.network

ABSTRACT

Autonomous software agents are beginning to move money on their own — paying for APIs, data, compute, and one another — across a fast-growing set of mutually incompatible payment rails: MPC wallets, in-enclave signers, on-chain session keys, and issued cards. Each rail ships its own controls, but each can govern only its own ecosystem. No incumbent can sit above all of them, because the incumbents compete. Countersign occupies that empty, neutral seat. It compiles a single declarative spending policy down to each backend's native controls, evaluates every transaction before it executes, and — on one signal — freezes every connected rail concurrently and fail-closed, recording every attempt in an append-only, hash-chained, cryptographically signed ledger. The thesis is falsifiable and we state the test plainly: stop agents across many vendors at once, in under a second, with tamper-evident proof. This paper describes the architecture, reports a four-rail live freeze in ~432 ms, sets out the security model and invariants, and explains why neutrality — not breadth of integrations — is the durable moat.

CONTENTS

01 The agent-spend gap

Until recently, the question "which software is allowed to spend money, and how much?" had a human-shaped answer: a person held the card, approved the invoice, or signed the transaction. Agents break that assumption. An LLM-driven agent can now hold a wallet, call a paid API a thousand times an hour, hire another agent, and settle in stablecoins — at machine speed, with no human in the loop by default.

That unlocks real value and introduces a new failure mode. A misconfigured, jailbroken, or simply over-eager agent can drain a treasury before anyone notices — and the blast radius grows with every wallet and card the operator connects. The controls that exist today are *per-rail*: Coinbase governs Coinbase wallets, Turnkey governs Turnkey signers, an issuer governs its own cards. An operator running agents across several of these has no single place to write a policy, no single button to stop everything, and no single trustworthy record of what every agent tried to do.

Three capabilities are missing at the layer above the rails:

+ **One policy.** The spending rules an operator cares about — per-transaction caps, rolling daily caps, allow/deny lists, human-approval thresholds — should be written once and enforced everywhere, not re-expressed by hand in N vendor consoles.

+ **One freeze.** When something goes wrong, the operator needs a kill switch that stops *every* agent on *every* rail at once, quickly and verifiably — not a frantic tour of vendor dashboards while money leaves.

+ **One ledger.** After the fact, the operator (and their auditor, and their counterparties) needs a single tamper-evident record of every attempt — allowed, denied, or held — across all rails.

Countersign exists to provide exactly these three, and nothing the rails already do well. It is not a wallet and never takes custody of funds. It is the layer that decides, stops, and proves.

+

02 Why neutrality, and why incumbents can't

The structural insight behind Countersign is simple: the cross-rail control layer can only be built by someone who governs none of the rails.

Every wallet vendor and card issuer is, by definition, issuer-side. Their controls are excellent — within their own ecosystem. But they compete with each other, and an operator's whole point in running multiple backends is to avoid depending on any one of them. No issuer can credibly offer to hold the policy and the kill switch for a competitor's rail, and none would trust a rival to hold theirs. The seat above all rails is therefore empty by construction, and it stays empty for any participant who also sells a rail.

This is the same position Plaid took between banks, or a neutral observability vendor takes across clouds: not the deepest in any one silo, but the only party that can see and act across all of them. Neutrality here is not a marketing posture; it is a moat that issuers are *barred* from copying without abandoning their own business.

> The rails are the substrate, not the moat. Breadth of integrations is table stakes — it is what makes the defensible layer *possible*. The moat is the neutral seat, plus the data only that seat can see.

03 Architecture

Countersign is a thin, keyless client talking to a TypeScript Core that holds all cryptographic and vendor-SDK weight. The language boundary between them is the trust boundary: a compromised client still cannot move funds or weaken policy, because it can only call the API.

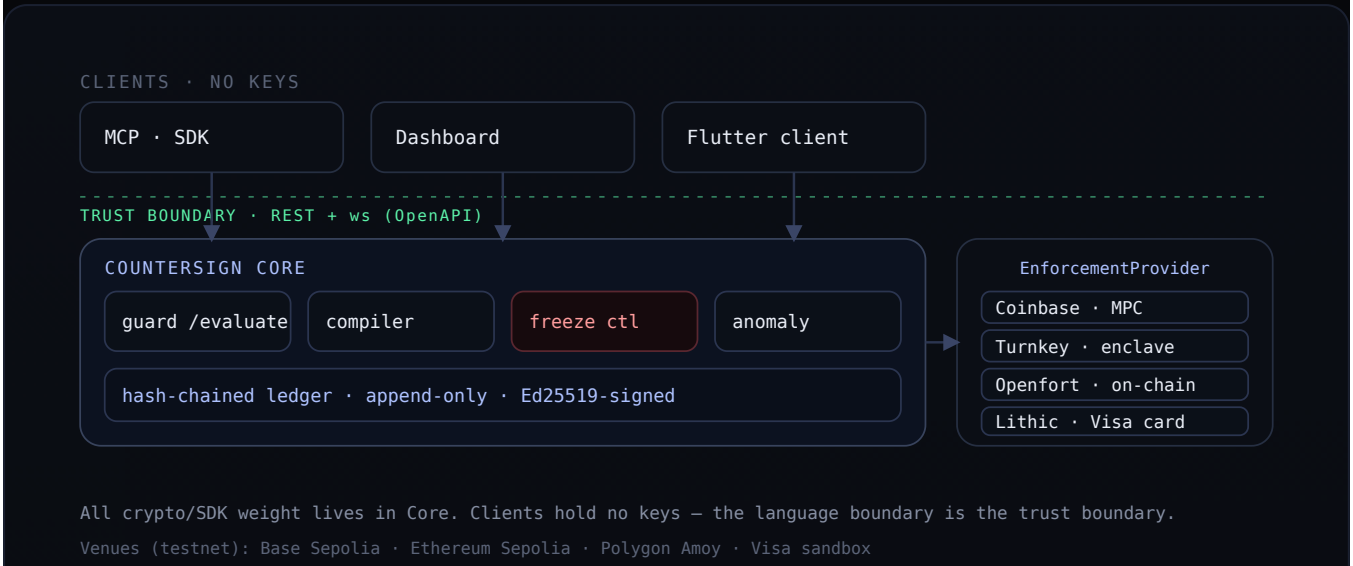

Figure 1. System overview — keyless clients, the Core brain, one interface, four live rails.

The unified policy and the compiler

An operator writes one declarative policy: per-transaction caps, rolling daily caps, allow/deny lists, human-approval thresholds, and the freeze state. The **compiler** — the core intellectual property — lowers that single policy to each backend's *native* controls, so that a compromised agent cannot exceed a cap by going around Countersign; the limit is enforced by the rail itself, not merely requested. Where a backend cannot natively express a field, Countersign enforces it at its own layer and records, honestly and auditably, which guarantee applies. The compiler refuses malformed input (for example, it rejects non-hex addresses) so policy injection cannot smuggle rules into a backend.

The pre-flight spend guard

Before an agent moves money it calls `/evaluate`. The Core answers **allow**, **deny**, or **needs-approval**, fail-closed. This single call is the product's most important surface: it runs on every transaction, which means Countersign observes an agent's entire cross-rail spend — the raw material for anomaly detection and reputation that no single-rail vendor can assemble.

The freeze controller

The kill switch fans out concurrently to every connected backend, bounded by a timeout. A confirmed freeze resolves in well under a second. A backend whose freeze will not confirm is **escalated** — Countersign revokes the session rather than reporting a dangerous wallet as safe. Default-deny is absolute: no decision and no backend response means the transaction does not execute.

The ledger

Every request, decision, and freeze is appended to a hash-chained, Ed25519-signed, append-only ledger. Tampering breaks the chain; the public key is exposed so anyone can verify it independently. The ledger is simultaneously the audit trail, the compliance artifact, and — because it records exactly what was evaluated — the usage meter.

04 The falsifiable test & results

A security claim is only worth as much as the test that could disprove it. Countersign's claim is deliberately runnable: freeze agents across many backends at once, in under a second, with a tamper-evident ledger of every attempt.

~432ms	4	<1s	115+
FOUR-RAIL FREEZE	RAILS, TWO RAIL TYPES	CROSS-VENDOR SLO	AUTOMATED TESTS

The headline run — `packages/agent-harness/live-freeze.ts` — drives three reference agents across three crypto backends and a virtual Visa card, then issues a single freeze. All four confirm in roughly 432 ms on testnet, and the signed hash chain re-verifies intact afterward. Because

Lithic is a card rather than a crypto wallet, the run proves the thesis is rail-*type*-agnostic: the same policy, freeze, and ledger govern cards and wallets under one action.

RAIL	ENFORCEMENT MODEL	FREEZE PRIMITIVE	STATE
Coinbase	native MPC session caps (Base Sepolia)	revoke session	`live`
Turnkey	in-enclave CEL pre-sign policy	gate signature	`live`
Openfort	on-chain session-key policy	flip on-chain guard	`live`
Lithic	card session caps (spend_limit)	PAUSE / CLOSE	`live`
Mock	faithful simulation, all 3 modes	all of the above	`tests + demo`

A fully runnable, credential-free mock suite reproduces all three enforcement modes and the fail-closed scenarios, so the claim can be exercised by anyone with the repository — no vendor accounts required — before they ever connect a real backend.

+

05 Security model & invariants

Countersign is a security product, so its design is expressed as invariants that must never be violated. They are the spine of the codebase and the contract with operators.

+ **Don't build cryptography.** Integrate vendor MPC/TEE. Agents use session keys, never master keys. Countersign adds the layer above, not another key-management implementation.
+ **Build above the wallets.** The moat is cross-vendor aggregation, not the wallet. No vendor logic leaks past the `EnforcementProvider` interface.
+ **Fail-closed.** No decision or no backend response means the transaction does not execute. Default deny, always.
+ **Backend-agnostic core.** Policy, ledger, and API never learn vendor specifics; only adapters do.
+ **The ledger is the source of truth.** Append-only and hash-chained, enforced at the database level as well as in application code.
+ **Testnet only.** No mainnet, no real custody, no PII/KYC until after a third-party audit.

The trust boundary is the language boundary. Clients are written in a different language from the Core and hold no keys; they can only call the API, so client compromise cannot move funds or weaken policy. Hardening already in place includes a database-level append-only trigger, an on-chain external anchor (every freeze countersigns the ledger), input validation with hex-address policy-injection defense, single-use websocket tickets so keys never sit in a URL, rate limiting, reserve-before-send accounting to close a daily-cap race, and a fail-closed boot guard. Secrets are scanned in CI. The remaining work before any real-value operation is native-enforcement parity on every rail and an independent security audit — and the paper says so rather than implying guarantees it has not yet earned.

06 The moat

Five moats, ranked by when each becomes real. The first is available on day one; the rest compound with usage and with the path toward custody.

#	MOAT	TYPE	WHEN IT'S REAL
1	**Neutrality**	positional	day one — issuers are structurally barred from copying it
2	**Cross-rail anomaly brain**	data network effect	compounds with usage — only the cross-rail seat sees the whole pattern
3	**Switching cost**	lock-in	as Countersign becomes the system-of-record for a fleet
4	**Regulatory barrier**	structural	earned with custody — the wall that slows you keeps rivals out
5	**Agent identity & reputation**	registry effect	the long game — the cross-rail "agent passport"

Moat 2 deserves emphasis because it is the one that grows quietly. Since `/evaluate` is called on every transaction, Countersign accumulates a view of each agent's behavior *across all of its rails* — velocity, counterparties, blocked-attempt bursts, cumulative drift. A rail-specific player physically cannot see this pattern, so only Countersign can build genuine agent-spend anomaly detection and, in time, a shared reputation signal that makes the whole network safer the more of the agent economy joins it.

07 Adoption, network effects & economics

Countersign is shaped for a self-serve, meeting-free motion: open-core, distributed over MCP, installable in one command, with a public falsifiable demo and a ledger that is itself shareable proof.

Three loops

- **Distribution.** The MCP server drops Countersign inside Claude, Cursor, and similar clients where agent builders already work — the kill switch and spend guard, one line, embedded mode, no credentials. The open-source front door is discoverable and forkable.
- **Data flywheel.** The spend guard runs on every transaction, so usage directly improves the anomaly brain and reputation signals — which makes the product safer, which drives more adoption, which produces more data.
- **Trust & agent-to-agent.** The hash-chained ledger is shareable compliance proof that pulls in the wider organization; and when agents hire agents, the payer wants the payee governed, so governance propagates along the spend graph.

Economics

Price against avoided loss, not cost of goods: a rogue agent can drain a treasury, and Countersign is cheap insurance against that. Adoption is free and the front door is never gated — a generous testnet tier fuels the flywheel. Revenue turns on with outcomes, not adoption: mainnet operation, the anomaly brain, the sub-second cross-vendor freeze SLA, signed compliance exports with retention, SSO/RBAC, and a self-host license for enterprise. The ledger doubles as an exact, tamper-evident usage meter — the meter is the audit log. The eventual endgame, once Countersign owns an MPC share, is an insured-freeze / system-of-record offering priced on governed mainnet volume.

08 Ecosystem roadmap

Integrations are sequenced crypto-first — permissionless, with agents spending today and zero regulatory drag — then settlement, then the identity and mandate layer that is the moat, then fiat and cards for the mass market, and finally custody.

TIER	THEME	REPRESENTATIVE INTEGRATIONS	STATE
0	Crypto enforcement + front door	Coinbase, Turnkey, Openfort, Lithic card, SDK + MCP	`live · testnet`
1	Settlement & protocol	x402, Circle / USDC, Privy	next
2	Identity & mandate (the moat)	Google AP2, Skyfire (KYA), Nevermined	next
3	Fiat & cards (mass market)	Stripe, Visa, Mastercard, Crossmint, Ramp / Brex, AWS Bedrock	later
4	Enterprise custody (endgame)	Fireblocks / Dfns / Sodot	endgame

The build today is Tier 0, proven live. In one breath: prove the cross-rail freeze on the crypto backends, make x402 and USDC first-class, become where mandates and agent identity live, extend onto cards for the mass market, and finally own the custody layer. For every integration the rule is the same — do not rebuild the vendor's cryptography; wrap it behind the interface and extract only the capability that the neutral layer needs.

09 Status & limitations

Countersign is testnet-only today. The core, freeze controller, policy compiler, signed and append-only ledger, mock provider, REST/ws API and dashboard, the spend guard with human-in-the-loop approval, the SDK and MCP server, x402 governance, and an early anomaly-freeze are built and tested. All four rails are live on testnet, each with scripts proving real enforcement, and the four-rail freeze is proven live. The honest open items are native-enforcement parity on every rail (so each guarantee is backend-enforced end-to-end), webhook event streams, and — non-negotiably — an independent third-party security audit before any mainnet or real-custody use. The Flutter client beyond a scaffold and push notifications are deferred. Nothing in this paper should be read as a claim of mainnet readiness or custody; Countersign holds policy, freeze, and proof, and never takes custody of funds.

10 Conclusion

As agents start to spend, someone has to be able to decide, stop, and prove — across every rail at once. That layer cannot be one of the rails, because the rails compete. Countersign takes the empty, neutral seat above them.

The contribution is not new cryptography; it is the aggregation. One policy compiled to each backend's native controls; one freeze that stops everything fail-closed in under a second; one tamper-evident ledger of every attempt. The claim is falsifiable and already met on four live rails in ~432 ms. The moat is neutrality and the cross-rail data only the neutral seat can see. The invitation is the same one that makes the claim credible: don't take our word for it — run it, freeze four vendors yourself, and verify the ledger.

References & further reading

[1] Countersign — repository, README and architecture (docs/architecture.md), `countersign.network`.

[2] Countersign threat model — docs/THREAT-MODEL.md and SECURITY.md (assets, trust boundaries, invariants).

[3] Headline live freeze — `packages/agent-harness/live-freeze.ts`; per-rail `smoke.ts` / `spike.ts` enforcement proofs.

[4] Policy compiler & unified policy schema — `packages/policy` (the cross-vendor compiler, core IP).

[5] Hash-chained ledger — `packages/ledger` (append-only, Ed25519-signed, on-chain anchored).

[6] x402 — open standard for internet-native (HTTP-402) machine payments, `x402.org`.

[7] Enforcement backends — Coinbase CDP / Agentic Wallets, Turnkey, Openfort, Lithic (per docs/sdk-research).

[8] Moat & integration roadmap; pricing & go-to-market — Countersign strategy notes.

Countersign holds policy, freeze, and a tamper-evident ledger — it never takes custody of funds. Testnet only; mainnet follows a third-party security audit.