ARCHITECTURE · IDENTITY · CANTON

An agent that can trade is a *liability*

Why we anchored identity on the Canton Network — and what we actually shipped.

Soon, the biggest traders on-chain won't be human. They'll be agents.

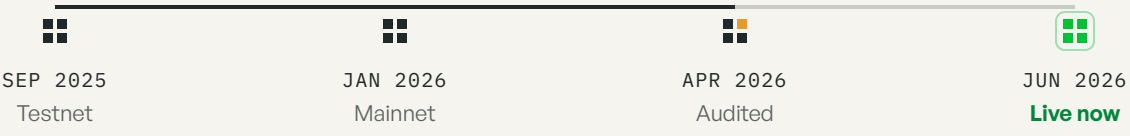
We think this is the least controversial prediction in crypto. Agents already read markets faster than people, execute without sleep, and scale across venues without hiring anyone. The volume is coming. What's missing isn't capability — it's accountability.

An agent that can trade is a liability until it can do three things:

01	Read	understand the market before it trades.
02	Act	operate inside hard limits, with an audit trail.
03	Prove	hold one verifiable identity across every wallet it controls.

TRACK RECORD

● 9 MONTHS LIVE ON CANTON



We don't have to predict the agent era — we can measure it. In the last 30 days, 57% of calls to our own intelligence API came from machines: MCP sessions and programmatic agents, not humans in a browser. The buyers of accountability infrastructure are already on the wire.

57% machine traffic · last 30 days · Coinversa API

● MCP server live — they're already calling it

The first two are engineering problems, and the industry is solving them at speed — including us: our intelligence API indexes the entire Hyperliquid clearinghouse in real time (3B+ trades, 2M+ wallets reconciled to the cent) and is live and self-serve for anyone today. Our execution layer wraps every order in guardrails — daily caps, leverage limits, allowed markets, scoped keys — and runs in our app today, with the agent-callable API shipping next.

The third is different. **Prove isn't an engineering problem. It's an architecture decision.** And it's the one we want to explain, because it's the reason Coinversa runs an identity layer on the Canton Network — and because "we put it on a blockchain" deserves more scrutiny than it usually gets.

The delegation problem

A trading agent isn't a wallet. It's a spray of wallets — one per strategy, per venue, per risk bucket. That's good practice: isolation limits blast radius. But it destroys the answer to the only question that matters when something goes wrong:

Who is this agent, what does it control, and what has it done?

Today, on every venue we know of, the honest answer is "nobody can say." Wallets are pseudonymous by design. An agent that blows up, manipulates a market, or runs away with delegated funds is just... addresses. The operator can deny everything. The victim can prove nothing. And any institution that wants to delegate capital to an agent has no artifact to point at when compliance asks what they're actually exposed to.

If agents are going to run serious capital, somebody has to be able to answer that question — cryptographically, not reputationally. We call that requirement KYA — Know Your Agent

Why not just attest on a public chain?

The obvious move is to write attestations to the chain the trades happen on. We rejected it for one reason that outranks every other consideration:

A trader's history is *alpha*.

A profitable agent's record — entries, exits, sizing, timing — is the strategy. Publish it, even as "just attestations," and you've open-sourced the edge. Accountability that requires broadcasting your trade history isn't accountability; it's a tax that only honest traders pay. Any identity system serious traders would actually adopt has to deliver **verifiability to the parties that need it and invisibility to everyone else**.

That requirement — selective disclosure as a protocol property, not an API promise — is what the Canton Network does that transparent chains structurally cannot. Canton's privacy model means contract data exists only for the parties to the contract. Not encrypted-but-visible. Not "private until someone indexes it." Structurally unavailable to non-parties.

Canton

We didn't pick Canton for the logo. We picked it because it's the only production network we found where *"your audit trail is private"* is enforced by the ledger rather than by our goodwill.

Privacy at the protocol · Selective disclosure · Canton mainnet

A sentence on mechanism, because "private blockchain" is usually hand-waving. In DAML, every contract declares its signatory and observer parties — and visibility ends there. There is no global mempool, no public state to index, no "everyone validates everything." Canton's synchronizers route encrypted messages between only the nodes hosting the parties to each transaction; a participant that isn't party to a contract doesn't receive it, can't index it, and can't see that it exists. When we say a trader's history is structurally private, we mean visibility is defined per-contract in the type system above — not by an access-control list we promise to maintain.

The landscape, in numbers

Two facts frame the bet: the flow is already automated, and the private ledger we anchored on is already carrying institutional volume.

AUTOMATED SHARE OF TRADING FLOW

bot / algorithmic share of volume · industry estimates, 2026 · methodologies vary

CANTON, BY THE NUMBERS

Canton

$6T+
tokenized real-world assets on network

600+
institutions, incl. JPMorgan, Goldman, DTCC

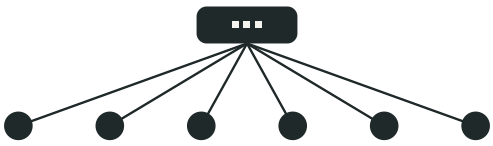
$355M
a16z-led raise, Jun 2026 · ~$2B valuation

$4T+
monthly processed volume

JPM Coin native · Jan 2026 | DTCC tokenized Treasuries · 2026 | Broadridge repo · $1T+/mo

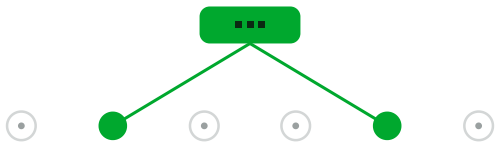
WHO CAN SEE YOUR TRADE?

TRANSPARENT CHAIN — 6 / 6 SEE IT



Every node receives, validates, and indexes every record — including your attestation patterns.

CANTON — 2 SEE IT · 4 BLIND



Only the parties to the contract hold the record. Non-parties can't see that it exists.

Canton figures: Digital Asset / Canton Network, Dec 2025–Jun 2026 (company press releases & CoinDesk). $6T assets on-network and 600+ institutions per Dec 2025; $4T+ monthly volume and the $355M / ~$2B-valuation raise per Jun 2026. Bot-share figures are industry estimates and vary by methodology.

Coinversa, in numbers

Canton's scale is the tailwind. This is what we've already shipped on top of it — indexed, in production, and independently audited. The "Read" and "Act" layers aren't a roadmap; they're load-bearing today.

COINVERSA, BY THE NUMBERS

Coinversa.

3B+
trades indexed across the Hyperliquid clearinghouse, in real time

2M+
wallets reconciled to the cent

57%
of our API traffic is machines, not humans

33/33
softstack audit findings resolved

- Canton validator live · Sep 2025
- Identity layer on mainnet · Jan 2026
- ISO 27001 whitebox + contract audit

LIVE NOW

Agents are already the buyers — and they're already calling our **live MCP server** today. `developers.coinversa.ai`

Coinversa figures: internal platform metrics, Jun 2026. Audit: softstack (ISO 27001-certified), Apr 2026 — independent smart-contract audit + whitebox API/frontend review; all findings resolved or acknowledged.

What we considered and rejected

Three architectures got serious evaluation before Canton. Each fails the same test in a different way.

01 Encrypted blobs on an EVM chain

Write commitments to Ethereum or an L2, encrypt the payload. Integrity: yes. But selective disclosure doesn't exist — anyone can see that you transacted, when, and how often, and traffic analysis on attestation patterns leaks strategy on its own. And the moment a third party needs partial access, you're building a key-management product, which is the actual hard problem you were trying to avoid.

FAILS — ACTIVITY IS VISIBLE TO EVERYONE

02 Zero-knowledge attestations

Elegant on paper: prove properties of your history without revealing it. In practice, proving costs and circuit maintenance for a fast-moving trade schema are heavy, the tooling is young, and anchoring proofs on a public chain still leaks timing and frequency. We expect ZK to complement this architecture later — proofs over the encrypted history — not to replace the private ledger underneath it.

DEFERRED — A COMPLEMENT, NOT A FOUNDATION

03 A trusted database with signed receipts

The pragmatic option: we keep records, we sign them, you trust us. But the entire point of KYA is that the answer to "who verifies the verifier?" shouldn't be "the company that profits from the answer." If accountability for agents reduces to "trust the platform's database," nothing was built.

FAILS — WE BECOME THE TRUST ROOT

Canton was the remaining option where per-party visibility is a ledger property, integrity is a contract invariant, and we are a party to the system rather than its trust root.

What we shipped

Our identity layer is live on Canton mainnet. We've run a Canton validator since September 2025 — on testnet first, mainnet since January 2026 — so this isn't a recent bolt-on. It's small, deliberately: four DAML templates, audited, doing three jobs.

01 One identity, many wallets

A `UserIdentity` contract anchors each user (and, next, each agent) as a Canton party — created from a passkey, no seed phrase. Wallets attach through a `LinkWallet` choice — you sign a challenge with the wallet being linked, our service verifies it, and only then is the choice exercised. The ledger enforces the rest: address validity, no duplicates, the 50-wallet cap. The contract exposes `IsWalletLinked` as a read-only check, which means anyone we authorize can verify "this address belongs to this identity" without learning anything else about it.

USERIDENTITY — SIGNATORIES AND INTEGRITY CONSTRAINTS (EXCERPT, AUDITED BY SOFTSTACK)

```
template UserIdentity
  with
    operator : Party
    userParty : Party
    primaryAuthKey : Text
    authKeys : [(Text, Text)]      -- (authIdentifier, encryptedSharedKey)
    encryptionKeyHash : Text       -- SHA256 of the shared encryption key
    linkedWallets : [WalletInfo]
  where
    signatory userParty
    observer operator
    ensure
      primaryAuthKey `elem` map fst authKeys &&
      length linkedWallets == length (dedup (map (.address) linkedWallets)) &&
      length linkedWallets <= maxLinkedWallets &&
      all (\w -> validateWalletAddress w.walletType w.address) linkedWallets
```

Note what the ledger itself enforces: the identity is signed by the user's party, not ours; wallet lists can't contain duplicates or malformed addresses; and the 50-wallet cap is a contract invariant, not an API rule.

02 Private, tamper-evident history

A trade, swap, or bridge becomes a `TransactionCommitment` : minimal public metadata (venue, type, chain), plus the full details encrypted client-side — we store ciphertext and a SHA-256 `dataHash` , and the contract's `VerifyIntegrity` choice lets anyone with standing confirm the record hasn't been altered. Only the user holds the decryption key. **We cannot read our own users' trading history.** That's not a policy; it's the design.

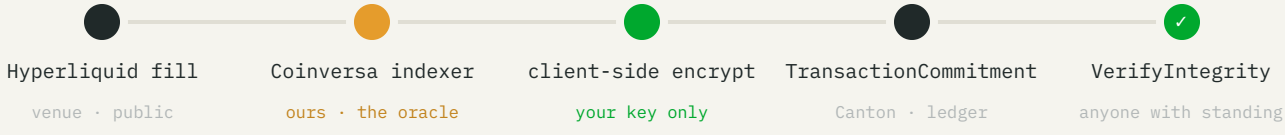
TRANSACTIONCOMMITMENT — THE ENCRYPTED RECORD AND ITS INTEGRITY CHECK (EXCERPT)

```
template TransactionCommitment
  with
    operator : Party
    txHash : Text
    sourceWallet : Text
    platform : Platform
    encryptedData : Text          -- ciphertext only; encrypted client-side
    dataHash : Text               -- SHA256 of encryptedData
  where
    signatory operator
    ensure
      encryptedData /= "" &&
      Text.length dataHash == 64
    nonconsuming choice VerifyIntegrity : Bool
      with providedHash : Text
      controller operator
      do return $ providedHash == dataHash
```

03 The KYA gate

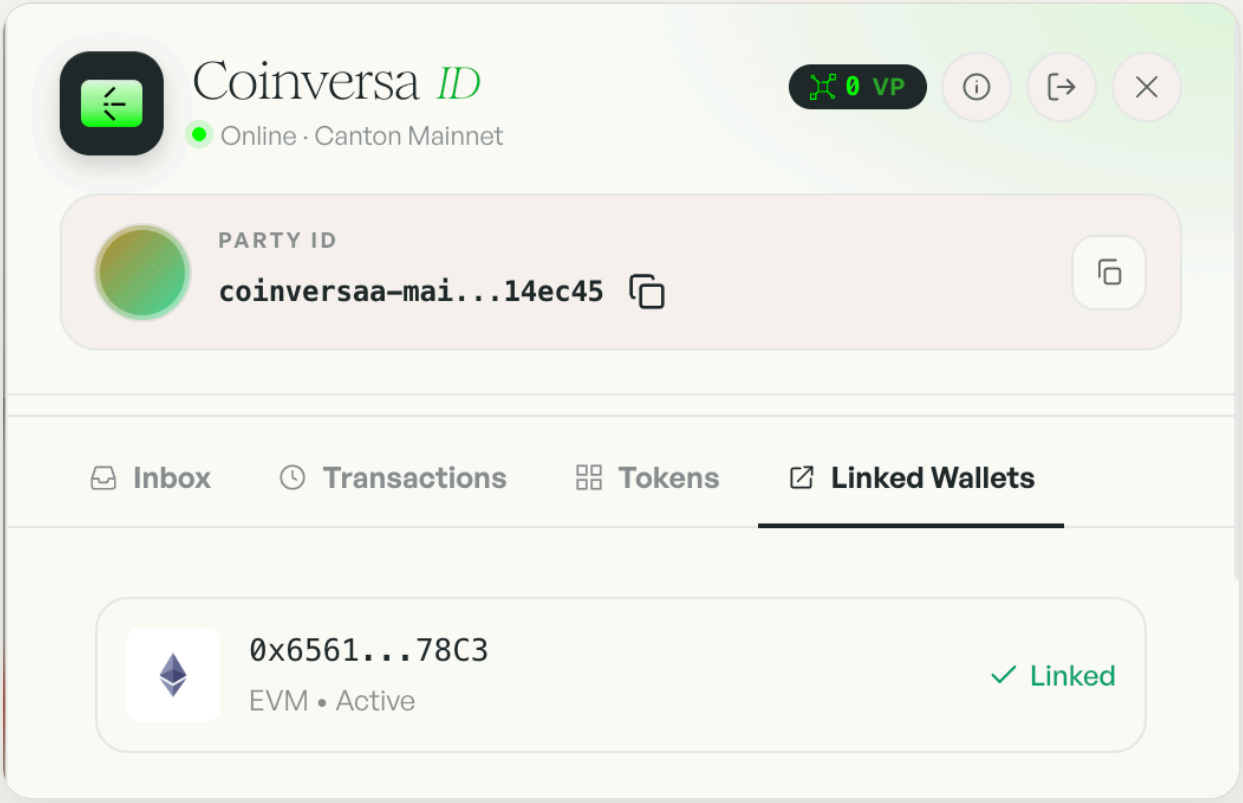
Our attestation endpoint refuses to record a trade for any wallet that isn't linked to an identity. No identity, no attestation — which means an attested history is only buildable by an accountable party. This is the primitive everything else grows from.

THE ATTESTATION PATH — TRUST, NAMED AT EACH HOP



The architecture, running

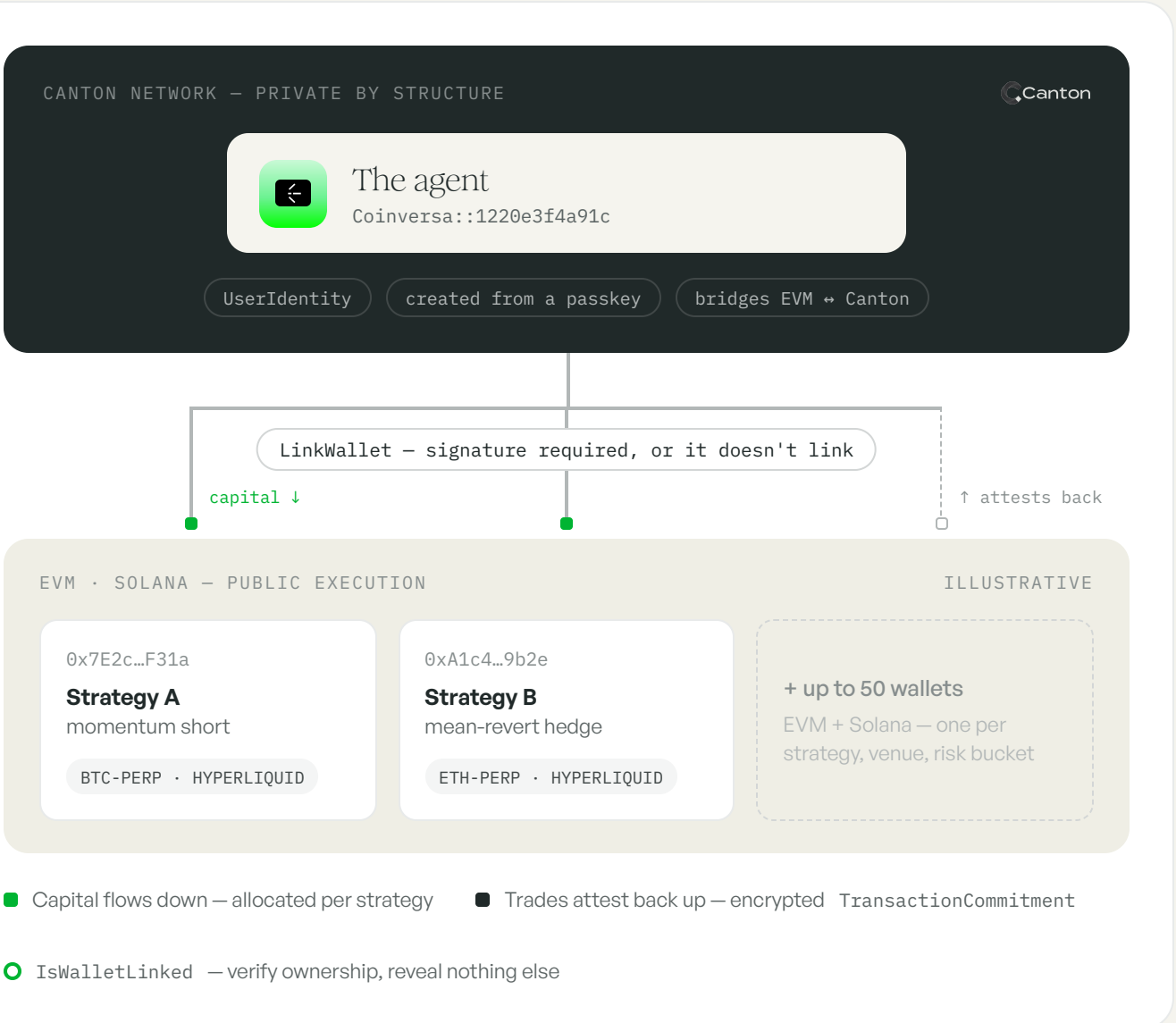
The two templates above aren't a whitepaper diagram. Here is `UserIdentity` resolving in the live app — a single Canton party on mainnet, with real wallets attached through the `LinkWallet` choice.



The `UserIdentity` contract, in production — one Canton party, real linked wallets, on mainnet today.

One agent. Many wallets. *One identity.*

This is how the architecture works for users today — and how it extends to agents next: the agent becomes a Canton party with its own wallet, allocates capital down to strategy wallets on EVM, and every trade attests back up into one private, tamper-evident history.



What "audited" means here

In April 2026, softstack — an ISO 27001-certified German security firm — completed two independent engagements: a smart contract audit of the Canton/DAML identity layer, and a whitebox review of our backend API and frontend.

AUDITED BY softstack ISO 27001 · April 2026 · reports public

SMART CONTRACT AUDIT — CANTON / DAML

20 findings

5 medium 15 low

16 resolved · 4 acknowledged

WHITEBOX REVIEW — BACKEND API + FRONTEND

13 findings

4 high 8 medium 1 low

All 13 resolved · fixes confirmed

We're sharing the numbers, not just the checkmark, because "audited" with no findings attached is marketing. **Thirty-three findings, named severities, and a resolution trail is what an audit actually looks like.** Full reports are public.

RESOLUTION TRAIL — ALL 33 FINDINGS 29 resolved · 4 acknowledged · 0 open

What this doesn't solve yet

An honest architecture post owes you the trust model, not just the feature list. Four things we want on the record:

01 **Wallet-link verification runs through our service today.**

When you link a wallet, you sign a challenge with that wallet and our backend verifies the signature before exercising the on-ledger `LinkWallet` choice. The ledger enforces format, dedup, and the wallet cap — but the signature check itself is off-ledger, operator-side. We chose this for UX speed; moving verification on-ledger (or to user-controlled choices) is on the roadmap, and until then, "wallet X belongs to identity Y" is backed by our service's verification plus the audit trail, not by ledger consensus alone.

02 **Commitments are operator-signed.**

Today, `TransactionCommitment` contracts carry our signature, not the user's. The user's protections are real but specific: the payload is encrypted client-side before it reaches us (we physically cannot read it), the SHA-256 hash makes tampering evident, and the user-signed `UserIdentity` controls what wallets the history can attach to. What the user does not yet have is co-signatory power over each record. The agent-facing release changes this: an agent party co-signs its own commitments.

03 **The venue binding is an oracle problem, and we are the oracle.**

A Hyperliquid fill happens off-Canton. The binding between that fill and its commitment runs through our indexer — the same engine that reconciles 3B+ trades to the cent, but ours. `VerifyIntegrity` proves a record hasn't changed since commitment; it cannot prove the record was true at commitment. Reducing that trust — signed venue data where available, independent attestors, cross-checks against public clearinghouse state — is the long-term work, and we'd rather name it than have it discovered.

04 **KYA is not KYC, and keys can be lost.**

A Canton party is a cryptographic identity, not a legal one — KYA answers "which agent, controlling which wallets, did what," not "which human owns it." The two compose (a legal entity can hold the party), but don't confuse them. And encrypted history has the property you'd expect: each passkey holds an encrypted copy of the shared key, so losing one passkey is recoverable — losing all of them means a history nobody, including us, can ever decrypt. That's the price of "we can't read it." We think it's the right price.

Where this is, honestly

Open and self-serve today	The intelligence API and MCP server. `developers.coinversa.ai`
Live on Canton mainnet, powering our app	Identity creation, passkey auth, wallet linking and verification, encrypted transaction commitments, the KYA gate on attestation. Not yet exposed as a developer API — that comes with the agent-facing release.
Validating	The full attestation pipeline for live trading flow — running, being verified against production volume before we call it done.
Early access	Identity, passkey auth, and wallet linking are live in the Coinversa app today — invite-only while we scale onboarding. **Request access →**

Where it goes

Today every user gets a Canton identity. The roadmap is symmetrical and we don't think it's subtle: **every agent gets one too.** Same templates, same linking, same encrypted history — an agent with a party ID, a provable wallet set, and a tamper-evident record of everything it has done, disclosable to exactly the parties with the right to see it.

When the first regulator, allocator, or insurer asks "can your agent prove what it did?" — the answer shouldn't have to be invented that quarter. We're building it now, while the question is still optional.

ROADMAP — SAME PRIMITIVES, EXTENDED TO AGENTS

NOW
Intelligence API self-serve · identity live inside the app

NEXT
Agent-callable execution · identity exposed as a developer API

THEN
Agent parties co-sign their own records · independent attestors

Read. Act. *Prove.*

Read is live for everyone today. Act and Prove are live inside our app and coming to agents next. The third is why we're on Canton.

Coinversa builds infrastructure for on-chain trading agents — market intelligence, guardrailed execution, and identity anchored on the Canton Network. The API is live and self-serve at `developers.coinversa.ai`. The app is in invite-only early access — join the waitlist.

Coinversa. Canton softstack