The agent proposes. Ava decides, signs and proves.

Ava is the execution layer that lets an AI agent move capital onchain inside limits a human set in advance, and leaves a receipt anyone can re-read from the chain.

01 Agents can plan. They cannot hold keys.

A language model asked to supply 300 USDC to a lending market can describe the transaction, encode the calldata and explain the risk. It cannot send it. Sending needs a signing key, and a key held by a model is a key held by whatever text the model last read. A prompt injection, a hallucinated address, a misread amount: for that holder these are the normal operating environment.

Sign every transaction by hand	The agent stops being useful for anything that recurs.
A hot wallet with a cap	The cap bounds how much moves. It says nothing about what the money moves into.
A raw key in a .env file	Free and instant. Readable by every process on the machine, from the first injected tool call on.

What is missing is not a better wallet. It is an authority layer: a place where the question *who said this agent could, and can you prove it afterwards* has a machine-checkable answer.

02 What Ava is

The agent never holds a key and never decides. It submits a request to a deterministic service. The service evaluates the request against authority a human recorded earlier, then either refuses with a typed reason or signs exactly what was authorised. No model sits anywhere in the security path.

Decides	Seven ordered gates over state Ava reads itself. A false gate ends the request with a typed refusal.
Signs	Keys live in a Turnkey enclave, one wallet per agent session. A prompt injection in one session cannot reach another session's balance.
Settles	The transaction is broadcast, then re-read from the chain and compared with what was authorised.

Records	Every execution that reaches a chain becomes a hash-linked receipt. Mandate refusals land on the agent's track record.

Agents connect from Claude Code, Cursor, Codex or anything that speaks MCP. The human grants a mandate once. Everything after that is checked by code, and the part a human must see, the exact preview of what will be signed, is bound to the signature by digest rather than by a flag.

03 How it works

The path of one request

A request arrives over MCP or REST and walks one path. The custodian is asked for a signature at exactly one point, after every gate has passed, and every execution that reaches a chain leaves a receipt.

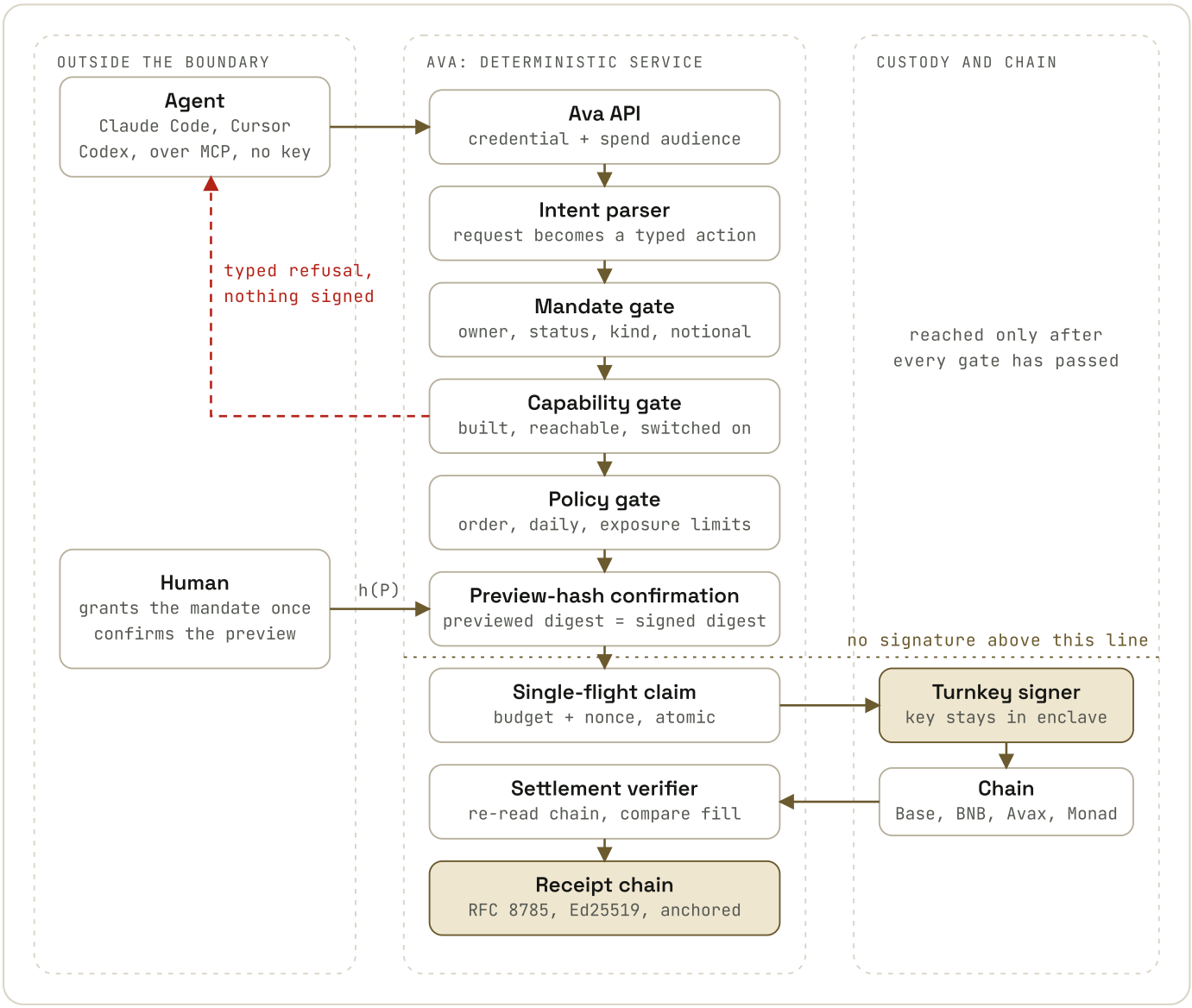


FIG 1 The key is requested once, after the gates. Everything above the dotted line can only refuse.

Seven gates, one conjunction

The route to a signature is an ordered conjunction. Each gate is a predicate over state Ava holds, not over the agent's say-so. The capability gate requires a registry row to be built, reachable on the calling surface and switched on by the operator; a route that settled once and was then switched off does not execute, because history is evidence, not permission. On mainnet the preview gate binds approval to the digest of the exact action: venue, chain, market, wallet, amount and mandate terms. Change one field between preview and signing and the request refuses. The claim binds the caller's idempotency key to the canonical digest of the economic action and reserves cumulative budget and the wallet nonce in one atomic step, so a retry under the same key cannot sign twice.

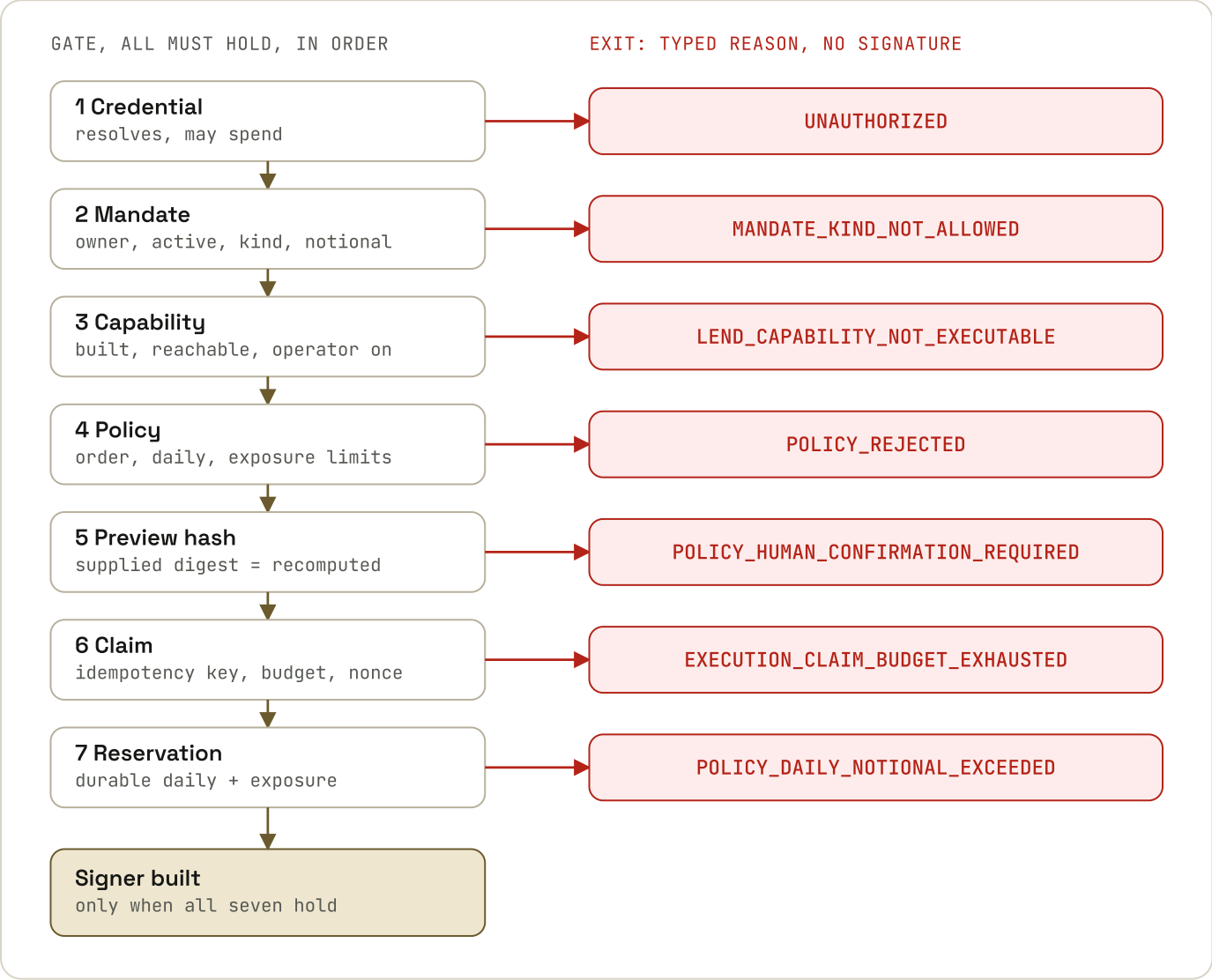


FIG 2 A refusal is an answer, not an error: each exit names its gate, so an agent learns what to change, and mandate refusals land on its track record.

A mandate is a bound that exists before the action

A mandate records capital, constraints (allowed action kinds, maximum notional, drawdown, slippage) and exit conditions, hashed and optionally EIP-712 signed by the owner. The owner can pause, resume or cancel it at any moment, and each non-active state refuses with its own code. The terms hash covers the terms, not the status, so a receipt issued under a mandate stays checkable after the mandate is paused or closed. Amending a limit mints a new hash.

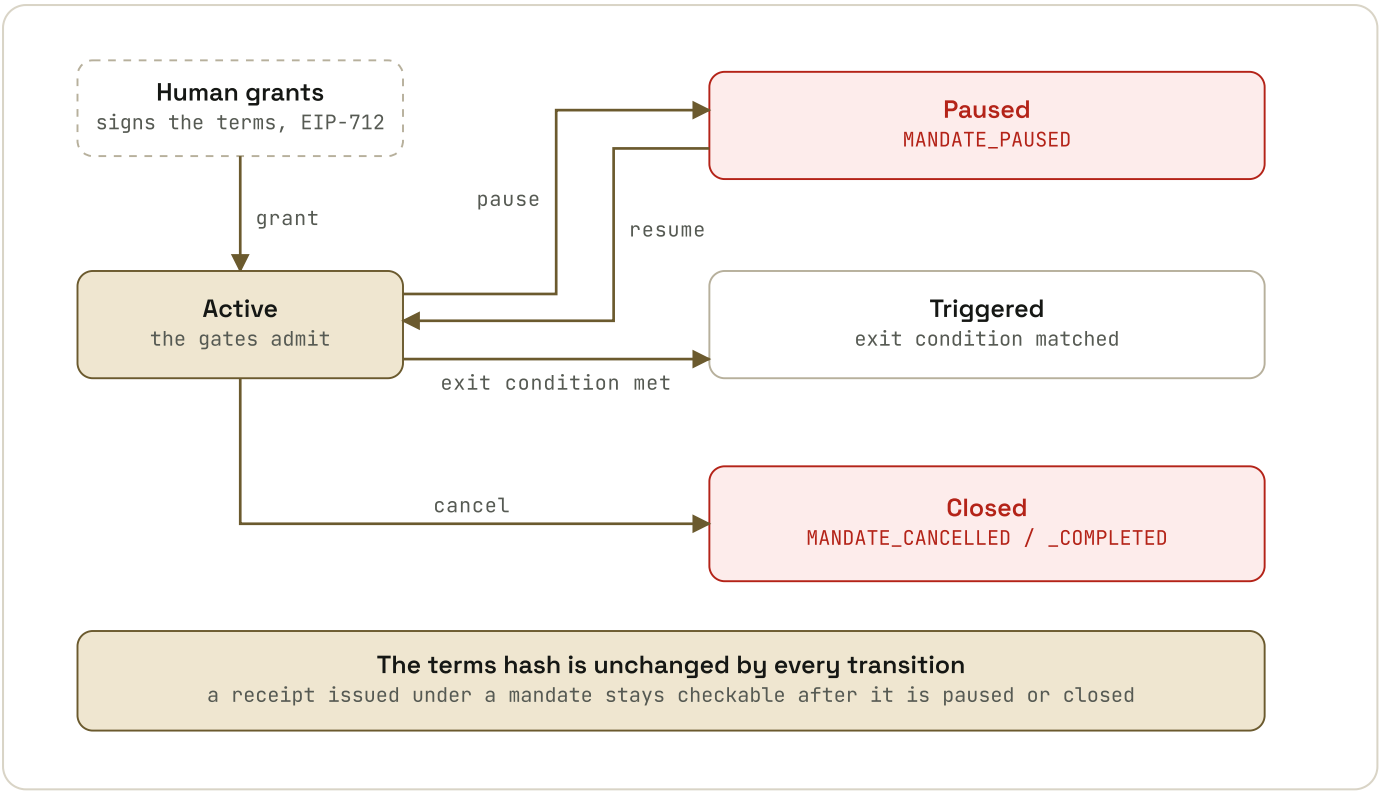


FIG 3 Status moves. The terms hash does not, which is what lets a receipt outlive the mandate it was issued under.

A receipt a stranger can check

Each receipt is hashed over RFC 8785 canonical JSON and linked to its predecessor by hash. The format carries an Ed25519 signature over that hash, and a scheduler anchors chain heads in a transaction on Base. Editing any receipt breaks every link after it, up to the anchored head, which an edit could only match with a SHA-256 collision. The verification endpoint is public. Integrity, on-chain proof and settlement agreement are separate verdicts: a receipt the chain contradicts is reported as contradicted, never folded into confirmed.

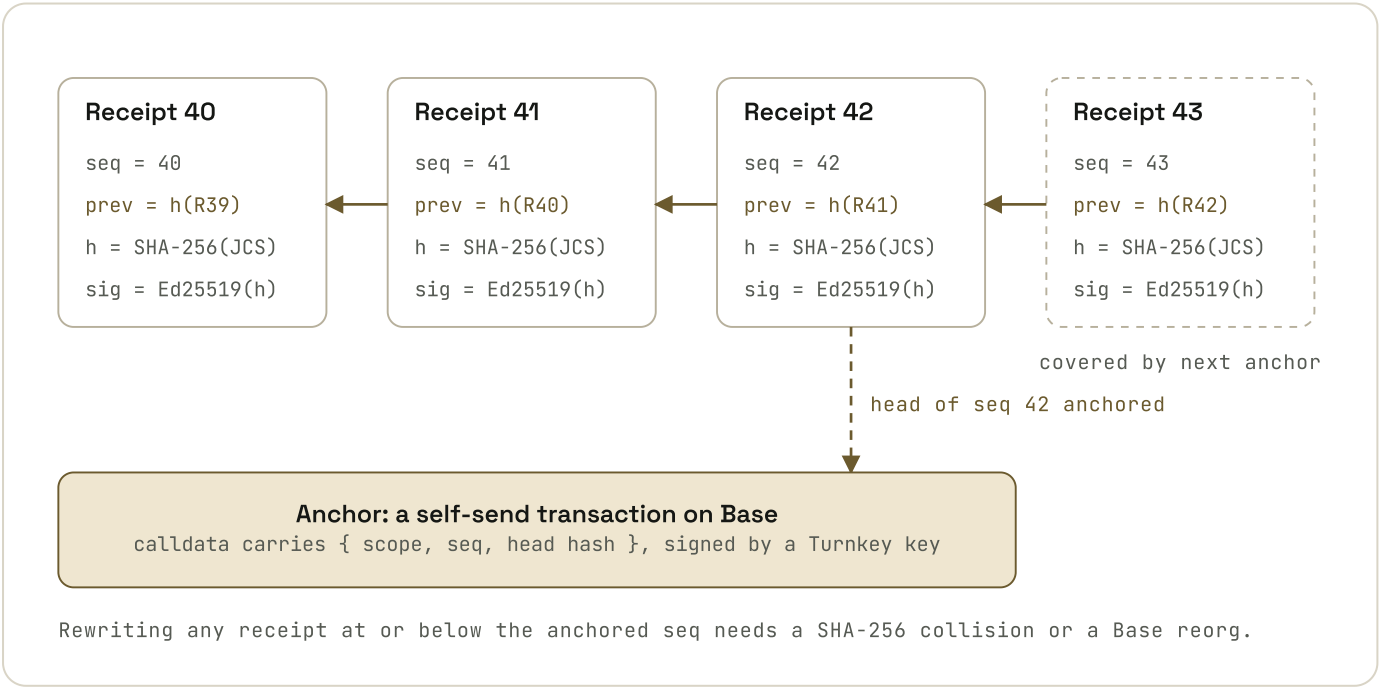


FIG 4 The chain is checkable without trusting Ava, and the anchor pins its head on a ledger Ava does not control.

04 Mainnet proof

7 settlements across 4 chains, each produced by Ava's own signer. The list below is the canonical one: before the site is allowed to build, every row is re-read from that chain's public node and the contract each transaction called is compared with the venue named beside it.

CHAIN	VENUE	ACTION	TRANSACTION	BLOCK
Base	Morpho gtUSDCp	supply USDC	`0x97b78540…36bbba`	49,315,911
Base	Morpho gtUSDCp	withdraw USDC	`0xce4109e4…6b0a7f`	49,327,027
Avalanche	Aave v3	supply USDC	`0x12990c11…f4209f`	92,231,544
BNB Chain	Aave v3	supply USDC	`0x7737cf5c…08b9de`	114,508,635
Monad	Aave v3	supply USDC	`0x50940c95…8715cf`	92,954,146
Monad	Aave v3	supply USDC	`0x1b23c35d…628af7`	93,291,520
BNB Chain	SushiSwap via LI.FI	swap, no venue adapter	`0x14850931…de0542`	114,518,492

Lending routes execute through the API under a signed mandate and a human-approved preview. The LI.FI-routed swap on BNB Chain shows the same signer carrying a different kind of action to a confirmed settlement. One of them resolves through Ava's own public re-read: `/v1/receipts/exec_lend_aave_62d2a6763c984af8be707a58/verify`.

05 Business model

Ava is paid where value is delivered, discloses the price where the human decides, and stays out of the funds path. Four lines, in the order they open.

1. A disclosed execution fee

Ava's execution fee on swaps and bridges is a visible fee on every trade, 0.25% by default, stated on the ticket and enforced server-side: the caller cannot pick a cheaper rate, and a quote whose fee differs from the configured rate is refused. It is collected through the venue's native integrator-fee rails, the LI.FI integrator fee and the Jupiter platform fee, so Ava never routes user funds through a contract of its own and adds no new contract to audit. The fee is shown in the preview the human confirms and recorded in the receipt.

2. Agent wallet plans

The unit of value is an agent that can be trusted with money, and that is the unit that gets provisioned. The first wallet is free; more wallets are a paid plan, so an owner running six agents pays for six bounded blast radii.

3. Verification and audit export

Whoever answers for the capital, an auditor, an investor or a compliance function, needs the record retained, queryable and exportable. The receipt chain and the public verification method are the base; retained history and export are the product, priced by seat or by retained volume.

4. Ecosystem partnerships

Chains and protocols that want agent-driven volume fund the work of making their chain live inside the same gates. Ava does not sell agent activity data.

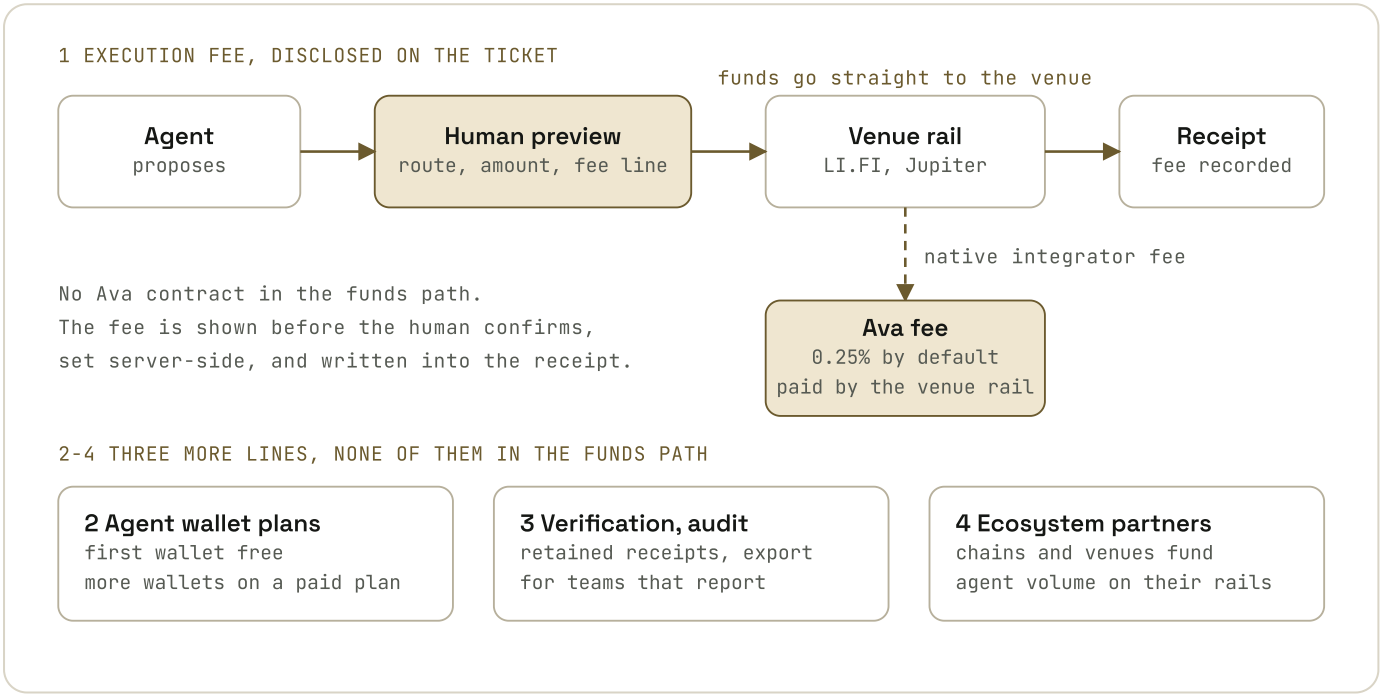


FIG 5 The fee rides the venue's own rail, so earning never requires touching the money.

06 The Ava flywheel

More agents mean more mandated executions. Each execution leaves a receipt, and receipts accumulate into a public, re-checkable track record. A longer record is more trust, and trust is what brings more capital and more agents. Fees from the executions fund more venues and more chains, which widens what an agent can be trusted to do. A token for fee discounts and staking will be detailed at launch.

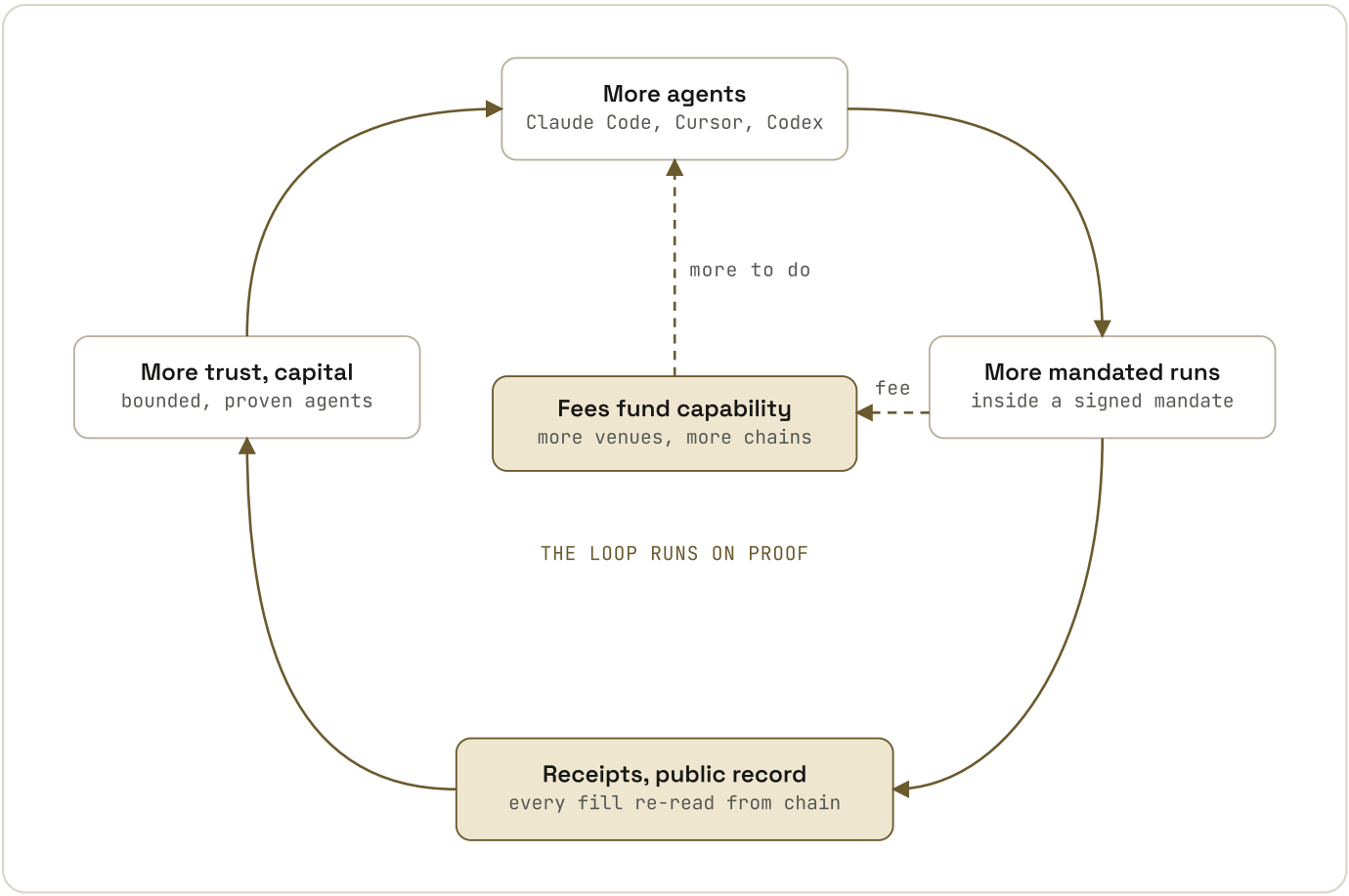


FIG 6 Every turn of the loop deposits proof, and proof is the one input a competitor cannot copy.

07 Who it is for

Developers running coding agents that touch money	Two or three terminals, Claude Code or Codex, projects that trade, rebalance or pay. They already trust an agent to edit code. Ava gives each session a wallet of its own with a limit set before it starts, and an answer to what the agents did last night.
Crypto-native users running standing strategies	They already think in mandates. Ava puts the bound between a script holding a key and a custodial product that takes discretion away: delegation they can audit.
Teams and funds that must show their work	A transaction hash proves something happened. A receipt chain under a signed mandate proves it was allowed, which is the question an investor or auditor actually asks.
Chains and protocols	Agent-driven volume with a safety story attached: every execution on their chain passes the same gates and ends in a receipt anyone can re-read from their public node.

08 Links

Whitepaper, PDF	`getava.xyz/ava-whitepaper.pdf, 48 pages`
GitHub, public repo	`github.com/getava-xyz/ava-the-ai-agent`
Connect an agent	`getava.xyz/install`

Ava. Agents move the capital. The record stays yours. getava.xyz