Nien: The Enterprise Trust Layer for Agentic Finance

Zero-trust environment with identity, policy, and custody for agents that move money

Nien Labs

June, 2026

Abstract

Enterprises are about to hand their day-to-day finance function to software: agents can already reason about invoices, payouts, reconciliation, and idle-cash placement, and the binding constraint is no longer capability but trust — who is accountable, what is actually running, and whether a given movement of funds should be allowed at all. The shift is enterprise-shaped: by 2030, Gartner expects a fifth of all monetary transactions to be programmable for agents, inside a B2B payment flow forecast to reach $224 trillion (Juniper), so the layer that governs the agents becomes foundational. We propose Nien as that layer, carried by two crypto-native mechanisms: ANID, an ERC-8004-based identity and reputation layer whose Reputation Registry Nien forks so feedback is written only by authorized Nomos contracts, making a trust score the public, append-only record of how real policy engines have judged an agent rather than a custodian's opinion; and Nomos, a deterministic, client-owned policy contract on BNB Chain that sits in front of every money-moving action, with a guarantee even Nien cannot circumvent — the MPC custody core co-signs a settlement only once the client's Nomos contract has approved it. Critically, Nomos is confidential by construction: rather than publish its rules, it commits to its policy and evolving state on-chain and proves every verdict in zero-knowledge, so any party can verify that a decision was reached correctly against the committed policy without the policy, the limits, or the underlying financial state ever being revealed. The result is one layer on which an enterprise can orchestrate any number of financial agents, its own or third-party, without handing any of them the keys to the treasury, without taking Nien's word that the rules are enforced, and without exposing those rules to anyone.

1. Introduction to Nien

Nien is not an agent. It is the layer that sits **beneath** agents — the on-chain foundation that governs enterprise finances, enables financial-agent orchestration, and makes agentic trust transparent rather than custodial.

Money is becoming programmable, and the entities moving it are increasingly autonomous. The missing primitive in this shift is not a faster rail or a smarter model. It is a **trusted place** that lets software touch enterprise capital safely, and lets the enterprise verify on its own, not on a vendor's word, that the trust is real. Nien is that place. It sits beneath any agent and any rail, and it does three things: it establishes **who and what** an agent is, it decides **whether an action is allowed**, and it **holds and moves the money** under that decision. Each of the three is anchored on BNB Chain so that the enterprise can audit every one of them without Nien's participation.

Nien is crypto-native at its core. Stablecoins are the natural money of agents: programmable, instant, global, and final. The layer's custody, settlement, and identity primitives are designed for that world first. But the world does not yet run entirely on stablecoins, so Nien integrates deeply with regional traditional-finance players, meeting enterprises where their money already is. This is an on-ramp, not a fallback. Every enterprise onboarded through conventional rails becomes a client already operating inside a crypto-native governance and custody fabric, and as stablecoin settlement matures, that client crosses over with no change to how its agents are governed.

This paper describes a single layer on which an enterprise can orchestrate any number of financial agents, its own or third-party, without handing any of them the keys to the treasury, and on which it can prove to itself, on-chain, that its rules are being enforced.

2. The Enterprise Agentic Future

The direction of travel is no longer in question. Agents will handle almost all of finance, for enterprises in particular. The repetitive, rule-bound, judgment-light work that consumes finance teams today is precisely the work

autonomous software does best, and the gap between agents that can draft the work and agents that do it end-to-end is closing fast.

What this future unlocks for a large enterprise is concrete. Working capital is continuously deployed, instead of dragged across week-long quarter-end exercises. Liquidity is positioned against real obligations, not a static buffer. Compliance is maintained continuously, rather than reconstructed after the fact. Vendor payments leave on the day they fall due, not in a Friday batch. Reconciliation is a side-effect of execution rather than a downstream chore. Mid-market companies see the same gains compressed: the finance function stops being a department and becomes a layer.

The open question is not **whether** enterprises run their finances on agents, but under **what constraints**. The binding constraint is trust and control, not capability, and crucially, an enterprise CFO cannot accept that constraint being satisfied by a vendor's promise. The trust must be auditable in a place the enterprise does not control. That place, today, is a public chain. Whoever supplies a credible enterprise-grade trust layer on that chain supplies the foundation the agentic enterprise settles on.

3. Where Enterprise Finance Meets Agents

Look closely at an enterprise's finance function and it decomposes into three broad buckets. All three sit squarely in the agentic cross-hairs:

- **Accounts Payable.** The **procure-to-pay** cycle: capturing invoices, routing approvals, paying vendors, and disbursing funds on time and in policy.
- **Bookkeeping & Invoicing.** The **order-to-cash** and **record-to-report** cycles: issuing invoices, collecting receivables, reconciling accounts, maintaining the ledger, and closing the books.
- **Money Management.** Treasury and liquidity management: optimizing working capital, deploying idle cash, positioning liquidity against upcoming obligations, and keeping the whole within a defined risk budget.

Each of these is repetitive, rule-bound, and data-rich, the exact profile of work where autonomous agents win decisively (Figure 1).

There is one point worth stating plainly: **Nien does not build these agents.** That is not our product, and we do not believe it should be any single vendor's product. We believe in a world of **competitive** agents, where enterprises bring their own edge to build accounts-payable, bookkeeping, and treasury agents, and where the best agent for a task is chosen on its merits rather than bundled by a custodian.

In that world, with many capable agents built by many parties all wanting to act on an enterprise's money, the scarce resource is no longer the agent. It is a **trusted, verifiable layer** on which any of them can operate safely. That is where Nien becomes essential.

4. What Nien Does

Nien exposes three capabilities over a custody core. Together they let an enterprise turn loose a fleet of agents on its finances without ever losing control of its funds, and without taking Nien's word for it that control is being preserved.

- **Agent identity and reputation (ANID).** Nien issues every agent a verifiable cryptographic identity through its **ANID** system, anchored in **ERC-8004**′s on-chain **Identity Registry**. ANID binds the agent to an accountable owner and to **what is actually running**, so a compromised or hijacked agent cannot quietly impersonate a trusted one and misappropriate funds. On top of identity, ANID accrues a portable, cross-company reputation that travels

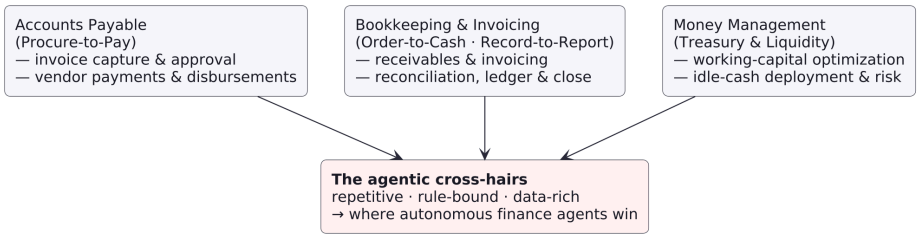

Figure 1: The three buckets of enterprise finance: Accounts Payable, Bookkeeping & Invoicing, and Money Management, all converging into the agentic cross-hairs.

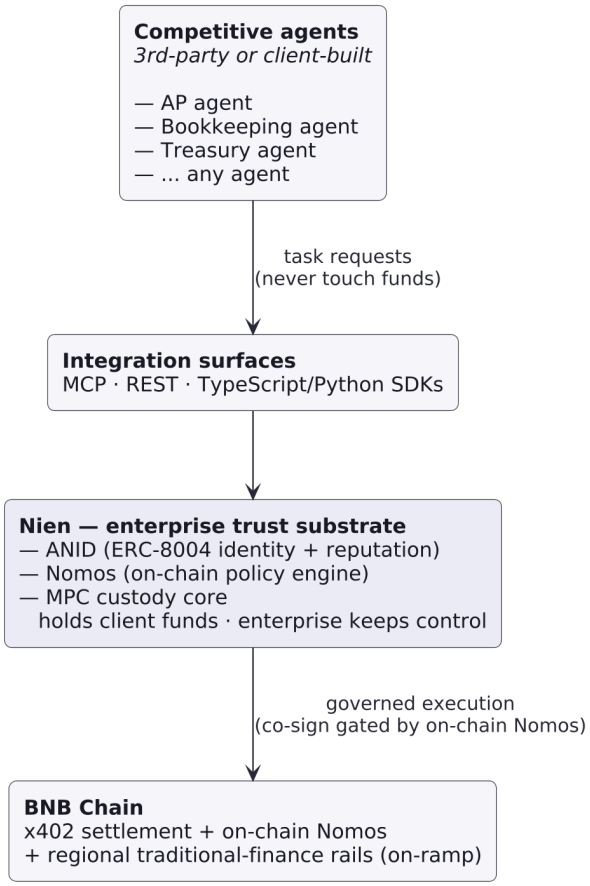

Figure 2: Competitive agents, third-party or client-built, connect through open surfaces and submit task requests. They never touch funds. ANID, Nomos, and an MPC custody core sit between every agent and every rail, with the governing contract on BNB Chain.

with the agent, and is **only writable by authorized Nomos policy engines**, never by Nien. ANID is the subject of §5.

- **Effective money management.** Capital is managed across **whitelisted, risk-defined** venues and market definitions, so idle funds can be put to work, but only within boundaries the enterprise has pre-approved on-chain.
- **Governance via Nomos.** Most importantly, Nien enforces governance on every action an agent attempts via **Nomos**, an on-chain deterministic policy engine the enterprise owns. No movement of money happens except through this gate, and the gate is a public smart contract, not a private service.

4.1. Agents plug in; they never touch the funds

Nien exposes open integration surfaces: an **MCP** server, a **REST API**, and **SDKs** in TypeScript and Python. Through these, any third-party or client-built agent connects and **indirectly** requests access to a client's funds in order to perform one of the three corporate-finance tasks above. The agent never directly touches the client's money. It submits a **task**, a request to act, into the layer (Figure 2).

4.2. The on-chain governance engine

Once a task arrives, the off-chain Nien orchestrator verifies the requesting agent's ANID and prepares the task for **Nomos** evaluation on BNB Chain. The Nomos contract, owned by the enterprise, evaluates the request against the policy state it holds and emits one of four outcomes (Figure 3):

- **Execute.** The action is valid and in policy; Nomos emits `Approved`, the MPC custody core observes the on-chain approval, co-signs, and the payment settles on the appropriate rail.
- **Block.** The action is out of policy, or the request is inconsistent with what the agent is authorized to do. Nothing moves.
- **Escalate.** The action needs a superior agent's or a human's permission before it can proceed.
- **Audit log.** Every decision, on every path, is recorded on-chain in tamper-evident form.

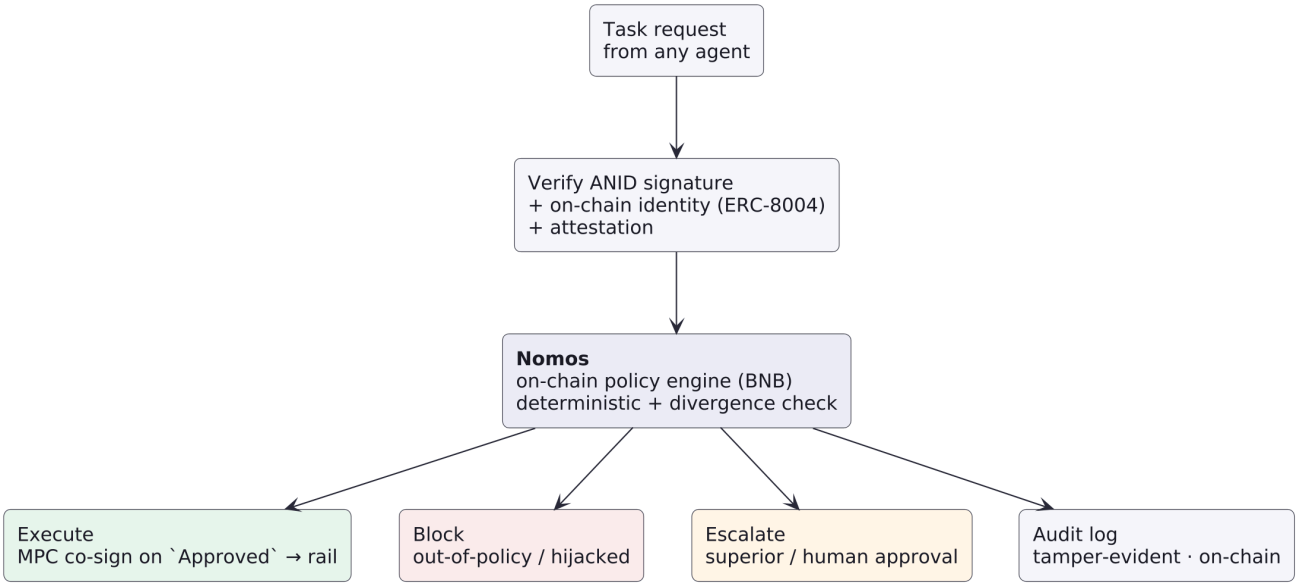

Figure 3: A task request from any agent is identity-verified against ERC-8004, then routed by the on-chain Nomos contract to one of four outcomes: execute, block, escalate, or audit-log.

The result for the enterprise is leverage. It can integrate any number of agents, or bring its own, and use Nien as the layer to **orchestrate** them, without granting granular access to each agent and without worrying about where company funds might leak. Policy is set once, in a contract the enterprise controls, and every agent inherits it.

4.3. Custody and control: MPC underneath, anchored on-chain

Underneath the governance engine, funds are held and moved through an internal **MPC-based threshold signing system**. Money moves only on a threshold of signatures, and the operating enterprise always holds a share of control, so no single party, including a key holder, can move funds alone. What is new in this design is that the MPC service is **bound to the on-chain Nomos verdict**: a co-sign physically cannot be produced unless the enterprise's Nomos contract on BNB Chain has emitted an `Approved` event for the exact task being signed. This binds **"policy was satisfied"** to **"a payment exists"** in code the enterprise can audit. A valid request that fails Nomos never reaches a signature, and Nien has no path to fabricate one.

5. ANID — the identity and reputation layer for agents

5.1. The problem: agents cannot prove who they are

Agents increasingly talk to other agents, and to shared tooling, through protocols such as MCP. Those protocols were never built to answer the most basic question in a financial setting: **who is actually on the other end?** When an agent sends a request over MCP, nothing about the channel proves the sender is the agent it claims to be. Anyone, or any compromised process, can present itself as a trusted agent. Impersonation is effectively free, and in a world where agents move money, free impersonation is catastrophic.

5.2. Built on ERC-8004 — open identity, public reputation

The Ethereum community is already standardizing the answer. **ERC-8004** ("Trustless Agents") defines three on-chain singleton registries, **Identity**, **Reputation**, and **Validation**, that let agents be discovered, identified, and assessed across organizational boundaries without pre-existing trust. The **Identity Registry** mints an ERC-721 token per agent (the `agentId`) and points at an off-chain JSON file declaring the agent's services, payment address, and supported trust mechanisms. The **Reputation Registry** records structured feedback against an `agentId`. The **Validation Registry** accepts external validator responses for re-execution, TEE, or zkML claims.

Nien adopts ERC-8004's Identity and Validation registries **as-is** on BNB Chain. Identity is plumbing: a keypair, a public-key registry, and a standard for binding them to a registered entity. Holding the registry is not a business, and we claim no ownership of the primitive itself. Every Nien-onboarded agent is registered on the open ERC-8004 Identity Registry, and every task an agent submits is signed with its registered key. An unsigned task, or one signed by the wrong key, is rejected before evaluation.

5.3. The Nien fork: reputation only written by Nomos

The piece of ERC-8004 we deliberately diverge from is the **Reputation Registry**. As specified, anyone can call `giveFeedback(agentId, …)` against any agent — Sybil resistance is left to off-chain aggregators. For a financial layer where reputation drives operating credit, that is the wrong default. A reputation score that anyone can write is a reputation score nobody can underwrite against.

Nien forks the Reputation Registry with a single, surface-level change: feedback writes are gated to a curated set of **authorized Nomos contracts.**

$$\texttt{giveFeedback(agentId, …)} \text{ succeeds iff } \texttt{msg.sender} \in \mathcal{N},$$

where $\mathcal{N}$ is the on-chain set of authorized Nomos contract addresses, maintained by the Nien Identity Council and freely auditable. A Nomos contract is the **only** thing on the network that can leave a review on an agent, and a Nomos contract leaves a review **only** as the natural by-product of evaluating that agent's task against a real enterprise policy and observing the outcome on-chain. Every reputation event is therefore traceable to a real enterprise, a real policy, and a real settled (or rejected) action, not to an opinion. Event signatures, storage layout, read functions, and indexers are unchanged, so any tool that already consumes ERC-8004 feedback consumes Nien's. They simply see a smaller, curated set of `clientAddress` values, every one of them an authorized Nomos.

Because Nomos contracts are themselves owned by enterprises, the reputation a Nien-registered agent earns is the **aggregated public record of how the network's real-money policy engines have judged it**. Reputation becomes a credit-bureau-style signal, public, append-only, and observed across companies, but with provenance no permissionless feedback system can offer.

5.4. The ANID stack

A bare ERC-8004 identity answers only one question: **who** holds the key. Built for finance, ANID adds three layers of identity and authority on top of the open standard (the lower zone of Figure 4):

1. **Who is accountable?** (L0): the cryptographic identity above, a keypair registered to an ERC-8004 `agentId`, bound through a signed ownership chain to an accountable legal entity.
2. **What is actually running?** (L1): **execution attestation**. An agent can commit to a hash of its prompt, tools, and code (Nien verifies the hash, never the contents). When the agent runs in a TEE, the hardware proves that commitment and the agent earns a higher operating tier. This is opt-in; most remote third-party agents run unattested, which is fine, as below.
3. **What is it allowed to want?** (L2): **intent binding**. Each agent carries a signed **intent manifest** declaring its purpose, permitted recipients, and amount/time envelope. Authority is scoped to that manifest, never granted in blanket.

The rest of the stack acts on these: at **L3**, Nomos (§6) scores every action for divergence from declared intent, catching a hijacked agent the moment it drifts; **L4** issues short-lived, attenuating capability tokens; **L5**, reputation and lineage, is the public ERC-8004 layer written exclusively by Nomos.

This is why a third-party agent need not run in a TEE for funds to stay safe. It is untrusted by design: it only requests, and Nomos decides. Attested or not, it can do nothing outside the policy it was granted. Attestation does not make an agent safe to use; it lets a trusted one earn more latitude.

Scope and limits. ANID itself proves two things and no more: **which** model, prompt, tools, and code ran (provenance, not a guarantee that the reasoning was "correct"), and the **bounds** on a compromised agent's blast radius through intent and scope. Acting on that identity, Nomos (§6) detects and blocks divergence from declared intent. None of this verifies that an agent's reasoning is correct. That is unprovable, and Nien does not assert otherwise.

5.5. Divergence, formally

Let an agent declare an intent manifest $\boldsymbol{i}$ for a task, an embedding over (purpose, recipient class, amount envelope, temporal window). Let $\boldsymbol{x}$ be the observed task payload at submission, embedded into the same space. Define the **divergence**

$$D(\boldsymbol{x}, \boldsymbol{i}) = 1 - \frac{\boldsymbol{x} \cdot \boldsymbol{i}}{\| \boldsymbol{x} \| \, \| \boldsymbol{i} \|} \in [0, 2].$$

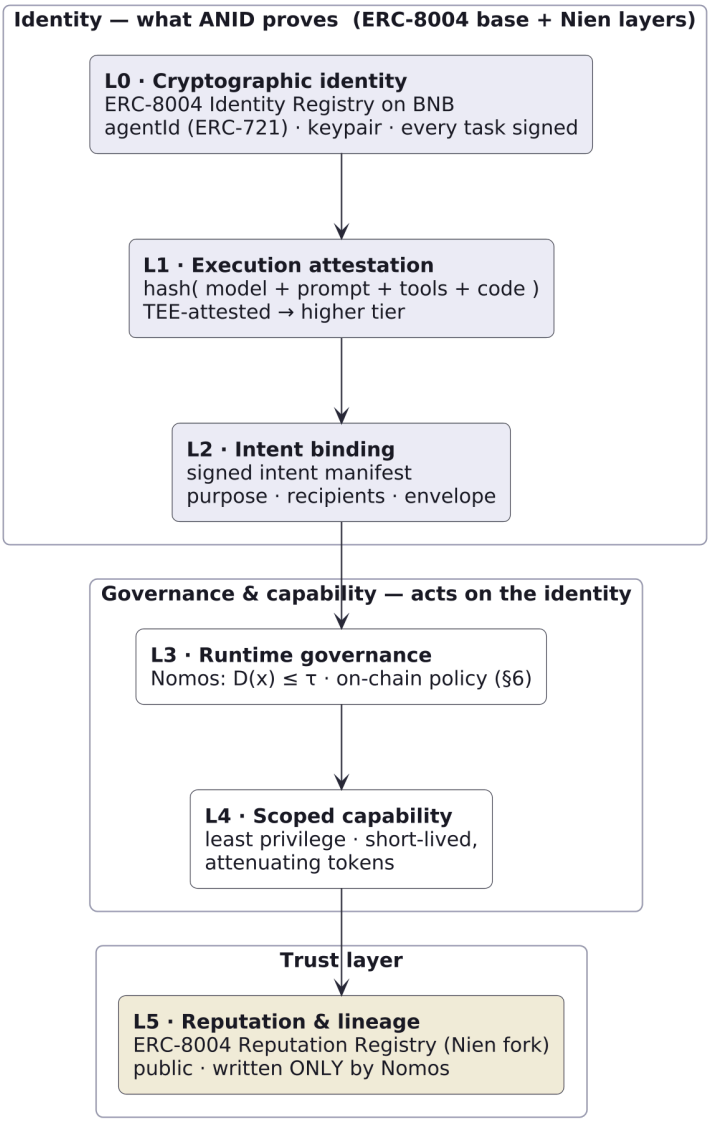

Figure 4: The ANID stack, in three zones. Identity (L0–L2) is built on ERC-8004 as adopted by Nien. The on-chain Nomos engine of §6 acts on that identity (L3), and capability is scoped to it (L4). The trust layer (L5) is the public, Nomos-only reputation surface.

Nomos admits the task only when $D(\boldsymbol{x}, \boldsymbol{i}) \leq \tau$, with τ a per-enterprise, per-tier policy parameter. A hijacked agent whose declared intent was "pay approved vendors" but whose observed task drifts toward "transfer to an unknown EOA" produces a divergence above τ before any value is risked.

5.6. The trust layer: reputation Nien alone can see, but anyone can read

This is where the network, and the durable advantage, live. A fair objection comes first: agent vendors will never hand over their data, because the model, prompts, and business logic behind an agent are their own edge. They are right, and ANID does not ask them to. The trust layer does not run on what an agent guards. It runs on what is observable at the transaction boundary, the behaviour every Nomos contract on the network already sees as the governance and settlement gate every task passes through: which agents pay reliably, which counterparties honour terms, what diverges, what gets blocked, what is disputed. None of that asks an agent vendor to surrender anything proprietary. It is a by-product of throughput, the same way a credit bureau learns whether you repay without ever seeing how you run your business.

From observed behaviour, written exclusively by Nomos contracts into the forked ERC-8004 Reputation Registry, Nien maintains two scores for every agent:

- a **trust score**, derived from settlement history and counterparty outcomes, and
- a **performance score**: the share of an agent's submitted tasks actually executed by the Nomos contracts it faces, aggregated across all of its deployed instances.

Both scores are public, anyone can read them straight from the chain. Both are also non-forgeable, because only a Nomos contract that has actually evaluated and settled a task is allowed to write to them.

Let $s_a(t)$ denote agent a's trust score at time t, and $g(o_t) \in [-1, 1]$ a structured map from outcome o_t to a signed reward (a settled, undisputed payment scores positive; a blocked or reverted task scores negative). Then

$$s_a(t+1) = \lambda s_a(t) + (1-\lambda) g(o_t), \quad \lambda \in (0, 1).$$

The half-life parameter λ controls how heavily history weighs against recent behaviour. Because the update is the on-chain Nomos contract emitting a feedback event, an agent's record is **portable across every enterprise it touches, and no single enterprise can reproduce it.**

5.7. Why ANID compounds

This is what turns ANID from a feature into a durable advantage. Four effects strengthen as the network grows:

1. **Portable reputation.** An agent's standing is built across every enterprise it touches, written by authorized Nomos contracts, and verifiable by anyone with an RPC endpoint.
2. **Two-sided acceptance.** Every counterparty that checks an ANID before transacting makes holding one more useful, and every ANID holder makes verifying more useful. Each x402 resource server is already a candidate verifier, so acceptance rides x402's own adoption.
3. **Shared fraud signal.** A cross-company record of bad actors grows stronger with every member, contributed without exposing anything proprietary. The forged invoice in §11 was caught precisely because it carried no valid ANID signature.
4. **Underwriting on top.** Once cross-company history exists, Nien can do what no participant can alone: extend credit to agents, guarantee settlement, price counterparty risk, and arbitrate disputes.

For any of this to hold, an identity must be costly to discard. An agent cannot be allowed to shed a bad record by minting a fresh key. Every ANID is issued under an **accountable issuer**, bound through a signed ownership chain to a real legal entity (L0) rather than self-minted anonymously, so a misbehaving agent's history stays attached to the entity behind it. The more agents and companies on the network, the deeper that history runs and the harder the layer is to replicate.

5.8. Reputation as operating credit

A credit score does more than describe a borrower. It decides how much they can borrow, on what terms, and how closely the lender watches. ANID scores play the same role for agents. The score is not a passive label; it is the input an enterprise's Nomos contract reads to decide how much latitude an agent earns.

An enterprise can bind its **operating tiers** to an agent's ANID score directly in policy. A new or low-scoring agent runs in a tight tier: small spend caps, a narrow recipient and venue whitelist, human approval on most actions, frequent escalation. An agent with a long, clean record earns a higher tier: larger limits, a broader whitelist, fewer mandatory approvals, more room to act between checkpoints. Trust is graduated. An agent earns latitude by behaving well over time, rather than being handed it on day one or denied it forever.

Because the score is portable across the network and lives on-chain, this deepens the credit-bureau effect. An agent that built a strong record serving one enterprise can start at a higher tier with the next, instead of beginning from zero each time it is deployed somewhere new. Reputation becomes working capital the agent carries with it.

Nien does not set anyone's risk appetite. It supplies the score; each enterprise decides what each tier is allowed to do, the same way a lender sets its own thresholds on top of a bureau's number (§6). The score informs policy. It never replaces it.

5.9. Integration: one-time and autonomous

For an enterprise that already runs agents, adopting ANID is a **one-time, low-friction** step. An identity is issued and attached at agent registration (a handful of fields, a key, and an on-chain ERC-8004 mint), after which signing, verification, and reputation accrual happen automatically on every task. There is nothing to maintain by hand. ANID is designed to be integrated and managed **autonomously by the agents themselves**, purpose-built for the financial world rather than retrofitted from a general-purpose identity scheme. An agent that exists today can be ANID-native within a day.

6. Nomos — the on-chain policy engine for enterprise funds

ANID establishes **who** and **what**; Nomos decides **whether**. Nomos is Nien's on-chain policy engine: a smart contract, deployed by each enterprise to its own address on BNB Chain, that lets it express, as policy, exactly how its money may move, and then enforces that policy deterministically on every task, from every agent, on every rail.

The crucial property of Nomos is not that it is a policy engine. Many products are. The crucial property is that **it lives on a chain the enterprise does not depend on Nien to read.** Its authority over the MPC custody core is not a vendor commitment; it is a code-level binding the enterprise can verify against the deployed contract bytecode. Crucially, this verifiability does not require the policy to be public: Nomos commits to its policy and state on-chain and proves every decision in zero-knowledge (detailed below), so the enterprise — and anyone else — can confirm that policy was applied correctly without the policy itself, or the financial state it reads, being exposed to competitors. The enterprise is not trusting Nien to apply policy. It is verifying that Nomos applied policy, on-chain, and watching the MPC custody core react.

6.1. Policy as composition

The power is in composition. Rather than a fixed set of toggles, Nomos exposes policy **building blocks** that an enterprise combines into rules as sophisticated as its finances demand (Figure 5):

- **Limits & caps**: per-transaction, daily, and velocity limits over rolling windows.
- **Allowlists & categories**: permitted recipients, spend categories, and counterparty risk tiers.
- **Compliance & sanctions**: recipients screened against sanctions and watchlists (OFAC and regional equivalents) through an on-chain sanctions oracle, with transaction-risk screening as a signed input; restricted counterparties are blocked or escalated.
- **Balances & buffers**: minimum balance floors and liquidity buffers that must be preserved.
- **Approvals & escalation**: thresholds above which a human, or a designated superior agent, must approve.
- **Adaptive controls**: an intent-divergence threshold τ (§5.5), and limits that scale with an agent's ANID reputation score, tightening automatically when behaviour drifts.

These compose. A single policy can say, in effect: **this agent may pay allowlisted suppliers up to a per-line-item cap, never drawing the treasury below its liquidity floor, with anything above a threshold escalated to a human, and its limits shrink automatically if its reputation falls.** Every money-moving task is checked against the full composition on-chain before anything is signed.

Formally, let $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$ be the policy primitives an enterprise has composed. A task t from agent a in on-chain policy state Σ is admitted iff

$$\bigwedge_{i=1}^{n} p_i(t, a, \Sigma) = \text{true}.$$

The verdict function

$$V : \mathcal{T} \times \mathcal{A} \times \Sigma \longrightarrow \{\text{execute}, \text{ block}, \text{ escalate}, \text{ audit}\}$$

is a pure function of its inputs. Determinism is by construction: the same task, agent, and policy state always produce the same verdict. Because that policy state is held confidentially (detailed below), Nomos does not ask third parties to recompute V from raw inputs; instead, every verdict carries a zero-knowledge proof that V was evaluated correctly against the policy and state the contract has committed to. Anyone can verify that proof against the on-chain commitments, so correctness is publicly checkable even when the inputs are not.

6.2. The deterministic rule, on-chain

One design rule is non-negotiable: **the decision is deterministic, with no LLM in the decision loop.** If the decider were itself a language model, a hijacked or prompt-injected requester could simply argue it into approving, relocating the very vulnerability the system exists to close. A model may **inform** a signal, for instance an anomaly score that feeds the divergence calculation, but it is never the authority. Only when the full policy passes on-chain does Nomos emit `Approved(taskHash, agentId, …)`, and only on observing that event does the MPC custody core co-sign. This binds **"policy was satisfied"** to **"a payment exists"**: a request that fails policy never reaches a signature, and the binding is enforced by code anyone can inspect.

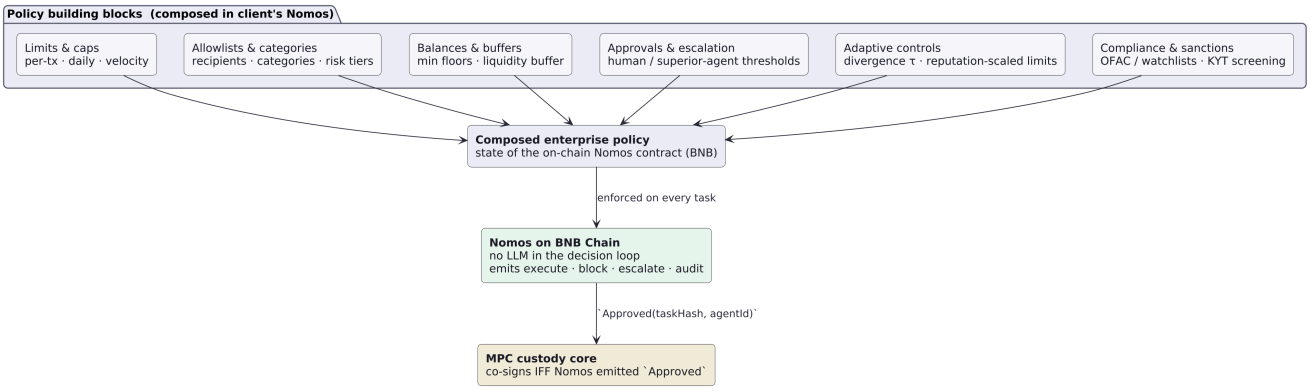

Figure 5: Composable policy. An enterprise assembles building blocks into a policy stored as state in its own Nomos contract on BNB Chain. The on-chain engine enforces it on every task and emits the verdict; the MPC custody core co-signs only on `Approved`.

A verdict is never a bare yes-or-no. Every evaluation produces a structured **decision trace**: which policy primitives ran, the inputs each saw, and, on a block, the specific primitive that failed. Because that trace would otherwise reveal the very policy and financial detail Nomos keeps confidential, it is not written to a public log; it is disclosed selectively. The calling agent receives the remediation signal it needs, so a blocked task comes back naming the exact rule it violated rather than an opaque rejection, and the enterprise can grant scoped disclosure — to its own finance team, an auditor, an insurer, or a regulator — that proves which policy was enforced without publishing it to the world. For a finance team, that is the difference between a gate they must trust and one they can interrogate.

6.3. Authored once, inherited everywhere

Every task resolves to one of four outcomes: **execute, block, escalate, or audit-log** (§4.2). Every decision, on every path, is recorded on-chain in tamper-evident form. The result for a finance team is leverage. Policy is authored once, centrally, in a contract the enterprise owns, and inherited automatically by every agent the enterprise runs, today and tomorrow. The same Nomos instance governs the AP agent, the bookkeeping agent, the treasury agent, and any agent that exists six months from now.

6.4. Confidential by default: zero-knowledge verification

On-chain usually means public, yet an enterprise's spending limits, vendor allowlists, treasury buffers, and counterparty relationships are precisely the detail it cannot post in the clear for competitors to read. Nien resolves this with zero-knowledge proofs. Nomos does not hold its policy or its evolving state as public plaintext; it holds a cryptographic **commitment** to them. Each task is evaluated against the committed policy, and Nomos verifies a zero-knowledge proof that the evaluation was performed correctly — that the verdict is exactly what the committed policy yields for that task — before it emits `Approved`. The proof reveals nothing about the rules, the limits, or the financial state they read; it reveals only that the rules were followed.

This is what lets Nien be confidential and trustless at once. A public chain on its own forces a choice between the two: publish the policy and lose confidentiality, or hide it and ask everyone to trust a private service. The proof dissolves the choice. What stays public and verifiable is the part that should be — the verdict, the `Approved` binding to custody, and the agent's reputation update; what stays private is the part that should be — the policy itself and the state it evaluates. The verifier on BNB Chain checks each proof against the on-chain commitments, so any party can confirm correctness without Nien's participation and without seeing a single confidential value.

Producing these proofs is computational work, and by default Nien runs the proving service on the enterprise's behalf, so adopting Nien requires no zero-knowledge infrastructure of its own. Because the **proof**, not the prover, is the source of truth, this delegation costs the enterprise nothing in trust: a faulty or dishonest prover cannot construct a proof that verifies against the committed policy, so it can never move an out-of-policy payment. For enterprises that prefer to keep even the plaintext policy entirely in-house, Nien offers self-proving as an opt-in — the enterprise runs the prover itself, and Nien sees only commitments and proofs.

6.5. Why on-chain, and what it costs

On-chain Nomos is the difference between **trust me** and **check the chain**. The cost is real, every policy evaluation that touches contract storage is a BNB transaction with finite gas and finite latency, and the design accommodates it. High-frequency primitives (cap checks, allowlist membership, velocity counters) live entirely in Nomos state and

evaluate in a single call. Heavier signals (divergence scoring, anomaly detection) compute off-chain and post their result as a signed input that Nomos verifies against a pre-registered signer, keeping computation off-chain while keeping the **authority** on-chain. The choice of BNB Chain is load-bearing here: low fees and sub-second block times make per-task contract evaluation practical at enterprise volume, in a way an L1 with mainnet-Ethereum gas economics could not be.

7. Active Treasury — capital that works within bounds

Idle operating capital is dead weight. Active treasury puts it to work, but only inside boundaries the enterprise sets in advance, so safety is never traded for yield.

7.1. Whitelisted, risk-defined venues

An enterprise, through its Nomos policy, pre-approves a set of **venues** and positions where capital may be placed, each with a risk tier and an exposure limit. Capital only ever moves between the operating balance and this whitelist; it can never reach an address or instrument outside it. On crypto rails today, the whitelist spans on-chain lending markets, money-market-style positions, and comparable risk-defined venues, all settled on BNB Chain. The same structure carries to traditional rails, where the venues are regulated instruments rather than on-chain ones.

7.2. The liquidity buffer

Yield is worthless if it locks up money the enterprise needs. Nien keeps a **liquidity buffer** of instantly spendable funds, sweeps the surplus into yield venues, and pulls capital back ahead of known obligations. Because the agent already sees the payment schedule, the buffer is **dynamic** rather than a static threshold, sized to the obligations actually coming due. Formally, the buffer L_t is held at or above the payouts projected over a horizon h,

$$L_t \geq \sum_{k:\ t \leq t_k \leq t+h} \text{outflow}_k,$$

with the surplus deployed to yield and recalled before each obligation falls due. A schedule-aware buffer is the difference between active treasury and a blunt bank sweep.

7.3. Who runs it, and why it stays safe

Any treasury agent can operate this framework, whether a client brings its own or uses one of Nien's first-party agents. Identity and on-chain governance bound it the same way they bound every other agent: the treasury agent can only move funds between the operating balance and the whitelisted venues, never to an arbitrary recipient, and every allocation passes the same Nomos verdict on-chain. The yield is real upside; the risk stays inside a box the enterprise drew in code it controls.

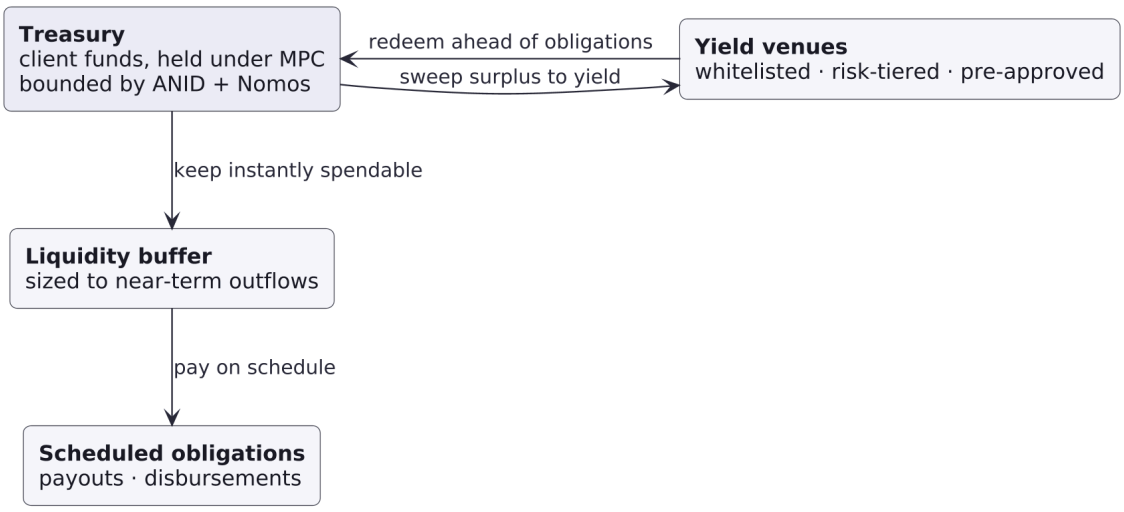

Figure 6: Active treasury. Surplus is swept to whitelisted, risk-tiered venues and recalled ahead of obligations; a dynamic buffer keeps near-term outflows instantly spendable. Funds never leave the whitelist.

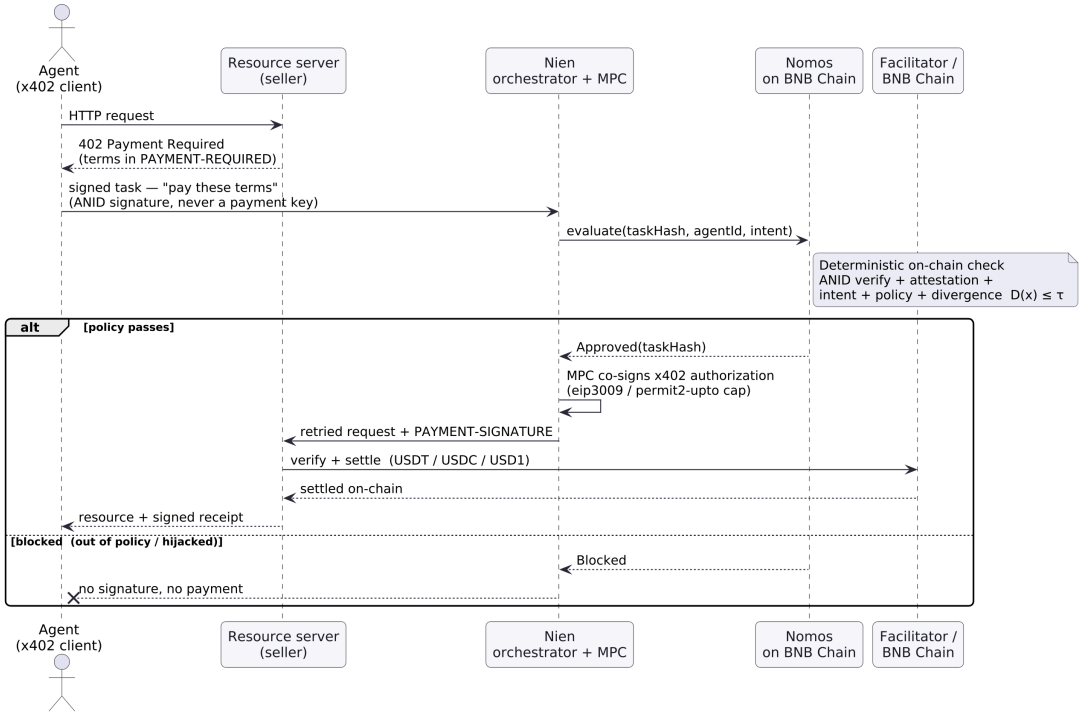

Figure 7: Nien in an x402 payment on BNB Chain. The agent submits a task rather than signing; Nomos runs on-chain, and MPC custody co-signs the x402 authorization only on an `Approved` verdict. A blocked request produces no signature and no payment.

8. Crypto-native settlement: Binance x402 on BNB Chain

Nien's crypto-native settlement default is **x402** on **BNB Chain**, an open standard stewarded by the x402 Foundation at the Linux Foundation that lets a payment ride along a single HTTP request. The agent never holds a payment key; it submits a task. Nien runs the on-chain Nomos check and, only on a pass, the **MPC custody core co-signs the x402 authorization** with stablecoin support for USDT, USDC, and USD1. A `permit2-upto` ceiling becomes a Nomos spend cap. This binds **"policy was satisfied"** to **"an x402 authorization exists"** (Figure 7): x402 secures the settlement; Nien secures the decision to settle; Nomos makes that decision auditable.

9. Meeting enterprises where they are: web2 today, stablecoin tomorrow

A trust layer for agentic enterprise finance that only serves stablecoin-native customers has cut off most of its market. The overwhelming majority of enterprise treasuries do not yet live on stablecoins. They run on bank rails, card networks, regional clearing systems, and a constellation of bank APIs and ERPs. Telling a CFO to rip those out and migrate her treasury to BNB Chain is the wrong pitch, nobody onboards that way. Nien is built to start from the other end.

9.1. Web2 first, on rails the enterprise already trusts

A web2 enterprise can onboard to Nien with its existing rails untouched. The same agents, the same Nomos, and the same ANID identity govern whatever rail the enterprise's money already moves on. Day one looks identical to day zero from the bank's perspective: the same payouts leave the same accounts on the same rails. What is new is what sits above them, an on-chain policy contract the enterprise owns, an identity layer for every agent, and an audit trail the CFO and compliance team can read directly from BNB Chain. A new client inherits the trust layer before moving a single dollar onto a stablecoin.

This is not a fallback path. It is the path. Every enterprise onboarded through conventional rails becomes a client already operating inside a crypto-native governance and custody fabric. As the world stablecoin-natives, counterparty by counterparty, market by market, that client crosses over with no change to how its agents are governed.

9.2. The silent migration

The layer is built so that the migration is silent. The same enterprise can deploy a Nomos policy that governs both rails: payouts on a domestic bank rail today, x402 settlements on BNB tomorrow, under one composed policy and one custody core. The ANID identity assigned to its AP agent at onboarding works in both worlds. The audit trail

is unified. Stablecoin-native and not-yet-stablecoin-native enterprises operate on the same governance fabric, only their settlement rails differ, and as a treasury moves stablecoin-native, the agents already trained against its Nomos continue to operate unchanged. The agent does not know which rail it is paying on; the policy decides.

9.3. The India Stack: a web2 on-ramp already in motion

The clearest proof that the on-ramp is real, not rhetorical, is that Nien is already building on one. India runs a national set of open financial rails, the **India Stack**, that does in regulated web2 much of what Nien otherwise does on-chain: UPI for instant settlement, the Account Aggregator framework for consented financial-data sharing, Aadhaar and DigiLocker for identity and KYC, and GSTN for invoicing and tax. An Indian enterprise operating on these rails already has programmable settlement, verifiable identity, and a digital audit trail, just not on a blockchain. We have started here, in parallel with stablecoin-native clients, because it is the largest live instance of the web2-first pattern this section describes.

Nien meets that enterprise exactly where it is, and the governance layer is unchanged. The same ANID identity binds each agent, and the same Nomos policy decides whether an action is allowed. What differs is only the leg beneath the verdict, where custody and settlement happen. On the crypto-native path, both resolve on-chain: a **Safe** smart-contract wallet, guarded by the Nomos policy, settling over x402. On the India Stack path, custody and settlement resolve through our **regulated integration partners** on web2 rails. A Nomos approval releases a payment instruction to a licensed partner that holds the funds and moves them over UPI, rather than to an MPC co-signer moving them over BNB.

The difference is a deliberate trade, not a compromise. A Safe contract earns trust by being public, verifiable code; a regulated partner earns it through licensing, audit, capital adequacy, and legal recourse, the assurances an Indian enterprise and its regulator already accept. For a treasury that has never touched a stablecoin, compliance-backed custody is not a weaker guarantee; it is the **recognizable** one. Nien supplies the same enforced policy and the same on-chain audit trail on top of it, so the enterprise still verifies its own rules rather than trusting Nien's word.

And because only the settlement leg differs, the silent migration of §9.2 applies unchanged. An enterprise onboarded through the India Stack is already operating inside Nien's governance and identity fabric from day one. As its counterparties begin to accept stablecoins, individual flows cross from the UPI leg to the x402 leg under the same Nomos policy, with no agent rewired and no reputation re-earned. The India Stack is how a crypto-native trust layer reaches a market that does not yet run on crypto, and brings it onto the chain one settled payment at a time.

9.4. Stablecoin rails for cross-border enterprise settlement

The fastest path from a web2 enterprise onto a stablecoin rail is the one place where the gap to legacy is most visible: **cross-border settlement.** A SWIFT-routed B2B payment settles in two-to-five days, charges correspondent-bank and FX fees that compound across hops, goes dark over weekends and bank holidays, and offers no real-time status to either side of the transaction. The same dollar moved on USDT, USDC, or USD1 over BNB Chain settles in seconds, for cents, around the clock, with on-chain finality both sides can verify.

The reason enterprises have not already made this switch is not technical. It is that no layer has given them the controls they would lose on the way: policy enforcement on every counterparty equivalent to what their treasury has today, an audit trail their compliance team accepts, and the ability to keep their existing bank rail running for the counterparties that are not yet stablecoin-ready.

Nien is built for exactly this case. Cross-border payouts are where the value of stablecoin rails is most concrete, and the enterprise's vendor base is rarely homogeneous in its readiness. Nien's layer lets the enterprise treat that heterogeneity as a policy parameter: stablecoin-payable counterparties settle on x402 over BNB; everyone else continues on the existing bank rail. The same AP agent runs across both. The same Nomos policy bounds both. Vendor classification is encoded in the allowlist primitive and updated like any other policy parameter, a vendor flips from "fiat-only" to "stablecoin-payable" with a single signed update to the Nomos contract, no agent redeployment, no rewiring.

As more counterparties accept stablecoins, the share moving on BNB grows. Cross-border float disappears for those flows. Cost-per-payment falls measurably for the line items already migrated, while the residual bank rail keeps the rest of the treasury operating exactly as it did before. **Cross-border becomes the wedge that brings a web2 treasury onto stablecoin rails one vendor at a time, without ever switching off the rails the enterprise already trusts.**

The same pattern generalizes: cross-border vendor payouts, marketplace seller settlements, contractor payroll across jurisdictions, intercompany transfers between subsidiaries in different regulatory regimes. Each is a use case where the latency and cost of legacy rails are most painful, and each can adopt stablecoin settlement independently under the same Nomos governance.

9.5. Why BNB Chain

The choice of BNB Chain is what makes the path above credible, on three axes.

- **Throughput and cost.** On-chain Nomos is only practical if per-task contract evaluation is cheap and fast. BNB Chain's fee economics make a few-cents-per-task policy evaluation realistic; an L1 with mainnet-Ethereum gas costs would force most of Nomos off-chain and destroy the on-chain verifiability the design exists to deliver.
- **Stablecoin liquidity.** USDT, USDC, and USD1 are deeply liquid on BNB Chain, and Binance x402 settles natively on top. The cross-border story above is rail-true today, not a roadmap.
- **Adjacency to the enterprise on-ramp.** The Binance ecosystem is the most direct path from a non-crypto-native enterprise's fiat treasury onto a stablecoin one. Building on BNB Chain places Nien on the same rail an enterprise will traverse anyway as it stablecoin-natives its operations.

We do not run our own chain. The enterprise audience does not benefit from one. They benefit from the chain their counterparties, their stablecoins, and their facilitators already use.

9.6. One chain for identity and policy

BNB Chain is not one option among several; it is the canonical chain. Every ANID identity and every Nomos policy lives on BNB and nowhere else. Identity and the authority over an enterprise's funds are single-homed there by design, so there is one place to read an agent's reputation, one place to verify and audit a policy, and one audit trail that never fragments across networks.

Other-chain support exists, but strictly as a **settlement proxy**. Where a client's operations require moving funds on another chain, the task is still identified, evaluated, and approved by that client's Nomos contract on BNB; only once the on-chain `Approved` event exists is the other chain used to settle. Reputation, policy, and the audit trail never leave BNB, and the design is built to consolidate those proxy flows back onto BNB as the ecosystem matures. Other chains are a spoke; BNB is the hub.

10. Security and threat model

Nien exists to make one guarantee precise: **a valid key alone can never move money out of bounds, and the proof that it cannot is on-chain.** This section states what that buys, what it assumes, and what it does not claim.

10.1. What Nien defends against

Threat	Nien's defense
Spoofed or impersonated agent	ANID signature checked against the open ERC-8004 Identity Registry on BNB (§5).
Prompt injection / hijacked agent	Intent binding plus divergence scoring; the Nomos contract blocks an out-of-intent action even when the key is valid (§6).
Stolen or leaked agent key	MPC threshold signing means one key cannot move funds, and out-of-policy actions are blocked at the on-chain Nomos gate regardless.
Leaked or unauthenticated endpoint	Unsigned or wrongly-signed tasks are rejected before they are ever evaluated.
Malfunctioning or rogue agent	Per-transaction caps, line-item budgets, and balance floors, composed in Nomos, bound the blast radius.
Confused deputy in multi-agent flows	Scoped, attenuating capability; sub-agents inherit strictly narrower scope (L4).

Threat	Nien's defense
Compromised perimeter or insider	Forged actions lack a valid ANID signature and are flagged on that basis.
Payment to a sanctioned or restricted counterparty	Recipients are screened against an on-chain sanctions oracle (OFAC and regional watchlists), with transaction-risk screening as a signed input; restricted counterparties are blocked or escalated (§6).
Vendor (Nien) overreach	The Nomos contract is owned by the enterprise; the MPC core co-signs only on its `Approved`. Nien has no path to fabricate an approval.

10.2. Trust assumptions

- **Custody is threshold-honest.** Funds move only on an MPC threshold of signatures, and the operating enterprise holds a share, so no single party, including a compromised component or a stolen agent key, can move funds alone.
- **The Nomos contract is the authority, not the agent.** The deterministic gate runs on-chain and is never an LLM, so a hijacked requester cannot argue its way past it (§6).
- **The identity root is sound.** ANID's value rests on the integrity of the on-chain ERC-8004 Identity Registry and the Nien fork of the Reputation Registry, both deployed on BNB Chain.
- **Attestation roots hold** wherever hardware attestation (a TEE) is used to establish what is running.
- **BNB Chain liveness.** Nomos relies on BNB Chain for finality and event delivery. We assume the standard liveness guarantees the chain itself provides; sustained chain-level outages degrade the layer to "no movement," which is the safe failure mode.

10.3. Coverage for residual loss

Bounding the blast radius is not the same as eliminating it. A policy gap, a mispriced venue, or a faulty external signal can still produce a loss inside the bounds the enterprise set. For that residual, Nien offers a **coverage layer**: a pool that backstops losses on governed actions, so an enterprise is not left carrying tail risk alone. Coverage is what turns **bounded** into **institution-ready**.

What makes Nien's coverage different is the data beneath it. Because every governed action, settlement, and dispute is recorded on-chain and attributed through ANID, coverage can be underwritten on an agent's real track record rather than a flat actuarial guess: a high-reputation agent operating inside a tight policy is cheaper to cover than a new one running loose. Reputation prices the premium, policy bounds the exposure, and the pool absorbs what is left. The same on-chain history that powers operating credit (§5.8) powers coverage.

11. Nien in practice

The layer is already at work with early enterprise clients. Four episodes, drawn from very different sectors, show the same on-chain governance core doing its job: catching what a valid key alone would have let through, and turning capital management into measurable upside.

> **FWC Inc. — when one agent's signal guards another agent's spend.** *cross-agent governance · prevented loss*
>
> FWC Inc. runs a staffing business through an internal agent, **Crew Source**, which reads filled timesheets to compute the invoice values billed to its end-clients. Two separate events, weeks apart, only mattered because Nien connected them.
>
> **First**, Crew Source, which also monitors platform behaviour, flagged one worker as a flight risk after a stretch of inactivity and notified the accounts manager. Acting on that signal, the manager wrote a single new rule into FWC's Nomos contract on BNB: **when the behaviour-monitoring agent raises this flag, withhold the worker's salary payout until a human clears it.**
>
> **Second**, on the next payroll run, FWC's accounts-payable agent did exactly what it was built to do and moved to disburse that worker's salary. Nomos matched the standing rule, blocked the payment on-chain, and the

withheld salary became leverage. The worker returned company assets in his possession, equipment that, without the hold, would have walked out the door unnoticed.

The two agents never spoke to each other, and neither was redesigned. A monitoring agent's observation became a payments agent's hard constraint **only because both operated on the same on-chain layer**, with a human-authored rule bridging them.
Takeaway. Governance lives below the agents, in a contract the enterprise owns, so a signal raised anywhere in the fleet can bound spend everywhere in it, without rewiring a single agent.

Hardware manufacturer (undisclosed) — catching a malfunctioning agent before it overspends. *malfunction containment · prevented loss*

A procurement agent responsible for sourcing chemicals for a hardware-manufacturing client malfunctioned and attempted to buy far more than the budget allotted for that line item. The agent held valid credentials; nothing about the request was unauthorized in the conventional sense. The client's Nomos contract caught the breach of the line-item budget on-chain and stopped the spend before any funds moved.

As entirely new sectors hand operational authority to agents, rogue and malfunctioning agents are not an edge case. They are a baseline risk of the technology. A budget overrun here is mundane; the same failure mode at scale is not.
Takeaway. Trust in a young ecosystem comes from a layer that bounds the blast radius of an agent's mistakes, not from hoping every agent is flawless, and from doing so in a contract the buyer can audit.

Supaboard — identity as the last line of defense. *injected request blocked · breach contained*

Supaboard is an agentic-AI company whose product answers natural-language queries over a customer's own data. Internally, it had deployed a full suite of agents to run its finances, most importantly invoicing. A security flaw elsewhere in its stack exposed the endpoint that calls the invoicing agent **without secure authentication**, and attackers used it to inject prompts instructing the agent to submit fraudulent invoices.

Those invoices never carried a signature from the Nien identity layer, because they did not originate from a properly attested, authorized path. Nien flagged them on exactly that basis, blocked the agent, and raised the alarm, which led Supaboard to find and close the leaking endpoint. The compromise reached Supaboard's perimeter; it did not reach Supaboard's money.
Takeaway. When an attacker gets **inside** and speaks with a hijacked agent's voice, cryptographic identity anchored on a chain the enterprise can read is what still tells a real action from a forged one.

Stablecoin-native fintech — active treasury that beats the on-chain benchmark. *enterprise treasury · yield upside*

A stablecoin-native fintech client used a third-party treasury agent operating against the framework in §7. Strictly inside the boundaries the client encoded in its Nomos contract, whitelisted venues, risk-tiered exposure, and a schedule-aware liquidity buffer, the agent generated roughly **80% more yield** than the fintech would have earned holding its USDC in standard on-chain money-market funds. No policy was changed, no custody constraint was loosened, and every allocation passed the same on-chain verdict before settlement.

The result compounded into conviction: the fintech moved more of its treasury onto Nien, and the layer now governs the majority of its idle balance.
Takeaway. Inside a risk framework the enterprise controls on-chain, active treasury is not a custody trade-off. It is a measurable return on capital, with the constraints visible to the CFO in code rather than in a vendor's terms-of-service.