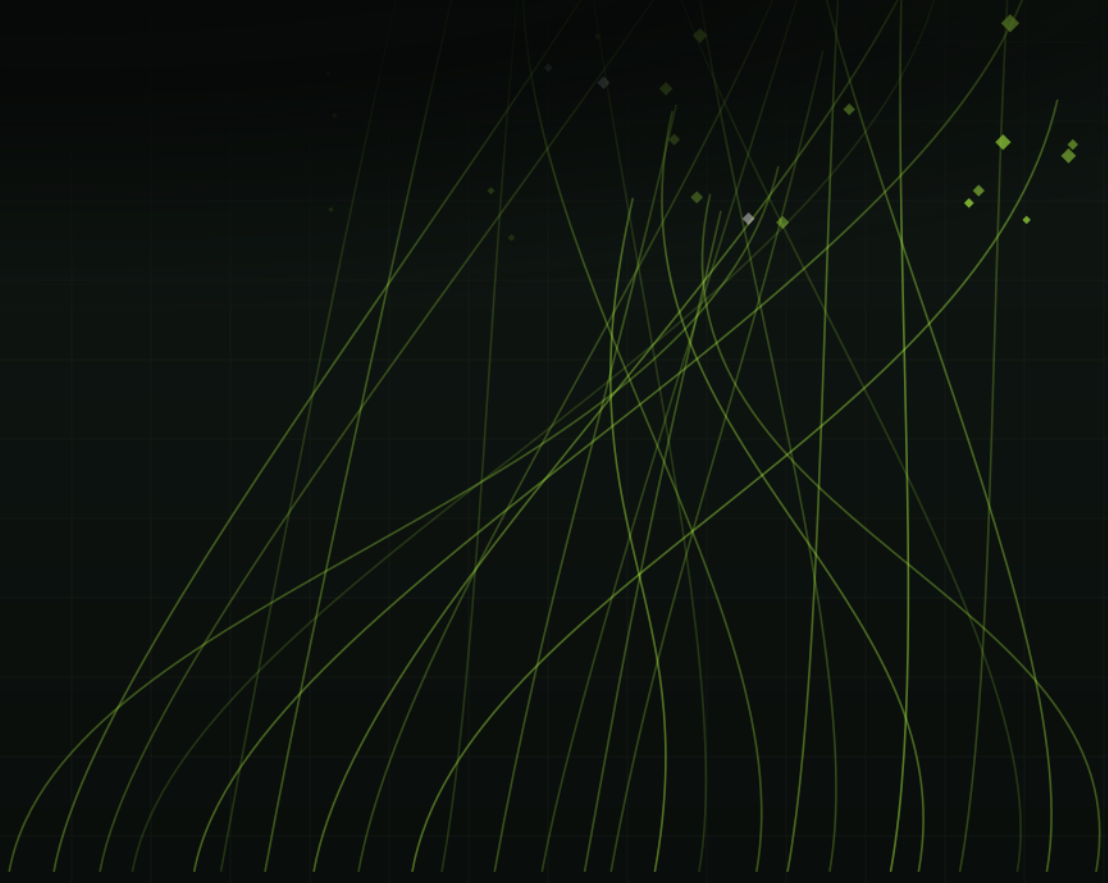
THE COMPLETE GUIDE

The Agent Builder's Bible

A complete guide to building economically capable agents:
the economy, the anatomy, the settlement question, and the pathway.



BEFORE YOU BEGIN

How to Read This Work

How to read this work. This is a working manual, not a manifesto. Part I establishes what the agent economy actually is in mid-2026, with the honest numbers. Part II is the anatomy: every capability an economically functional agent needs, and how to build each one. Part III addresses the layer most builders inherit without choosing - settlement - and makes the case for the pathway we build. Part IV is patterns, forecasts, and checklists you can put to work today. Read it straight through, or jump to the chapter that matches where your build is stuck. Every number is sourced in Appendix B; every forecast is labeled as one.

Disclosure, before anything else. This work is published by GOAT Network. We build Bitcoin-secured infrastructure for the agent economy - x402 payments, ERC-8004 identity, a zkEVM execution environment anchored to Bitcoin - and Part III argues you should build on it. That is an obvious commercial interest, so we hold ourselves to a rule throughout: every figure is sourced, every caveat that cuts against us is printed, and the strongest counterarguments to our own thesis get their own section. Where our infrastructure is young, we say so. Judge the case on the evidence.

Three house rules govern every page. We never say "trustless" - every system has trust assumptions, and we state them instead. We use the present tense only for things that are live. And if a sentence would need a footnote to be true, we do not write the sentence. We write the footnote.

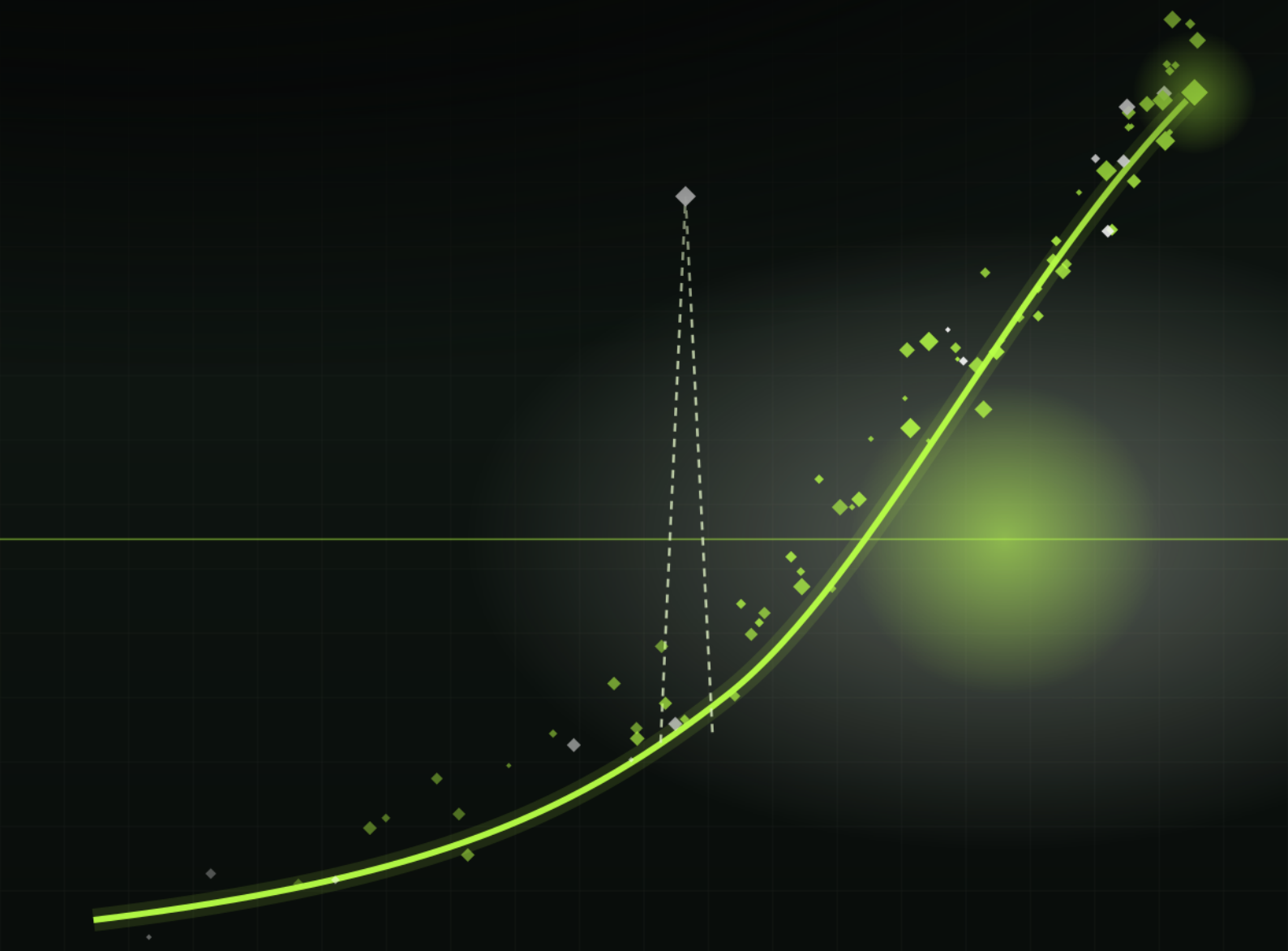
CONTENTS

What This Work Covers

PART I

THE ECONOMY ARRIVING

What actually happened between mid-2025 and mid-2026: real rails, real volume, honest discounts, and the reason the next decade of infrastructure is being decided right now.



CHAPTER 1

The Machine Customer

Somewhere in the time it takes you to read this sentence, a piece of software will pay another piece of software for something - a weather feed, a paragraph of inference, a search result, a verification check. No human will see the transaction. No invoice will be issued, no account created, no card entered. A request went out, a machine-readable price came back, a stablecoin moved, and a resource was delivered. This happens millions of times a day now, and eighteen months ago it essentially did not happen at all.

That is the single fact this work is built on, so it deserves to be established carefully - with the real numbers, and with the discounts that honest analysis requires.

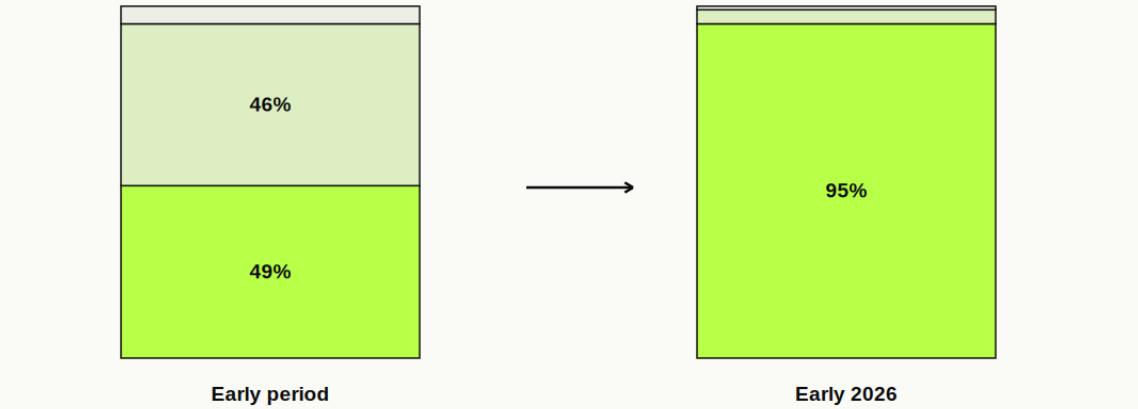
The numbers, then the discounts

The most load-bearing dataset in agent infrastructure is the on-chain history of x402, the HTTP-native payment standard Coinbase created and later contributed to the Linux Foundation. Unlike framework download counts or waitlist signups, on-chain payments cannot be faked cheaply and cannot be narrated around. By spring 2026, protocol-wide activity stood at roughly 165 million machine-initiated payments, around 50 million dollars in cumulative volume, and about 69,000 active agents, with Chainalysis independently counting well over 100 million cumulative transactions on Base alone. Stripe shipped x402 support in February 2026. Cloudflare wired it into pay-per-crawl, turning bot mitigation into a pricing mechanism, and later opened a monetization gateway around it. AWS shipped x402 at its edge in CloudFront and WAF. Visa attached it to its Trusted Agent Protocol. Fireblocks joined the foundation with enterprise spend-governance tooling.

Composition matters more than count, and the composition shifted decisively. Chainalysis finds that transfers of one dollar or more carried about 49 percent of x402 value in the early period and 95 percent by early 2026, while the ten-cents-to-a-dollar band collapsed from 46 percent to 4. Retention improved roughly fourfold over six months, and x402 wallets hold larger, more diverse balances than typical wallets on the same chain. Translated out of analyst language: sub-cent experimentation is giving way to funded wallets making purposeful purchases - data, inference, search, verification. The rails are acquiring customers, not just transactions.

CHAPTER 01

Value composition: experiments give way to customers



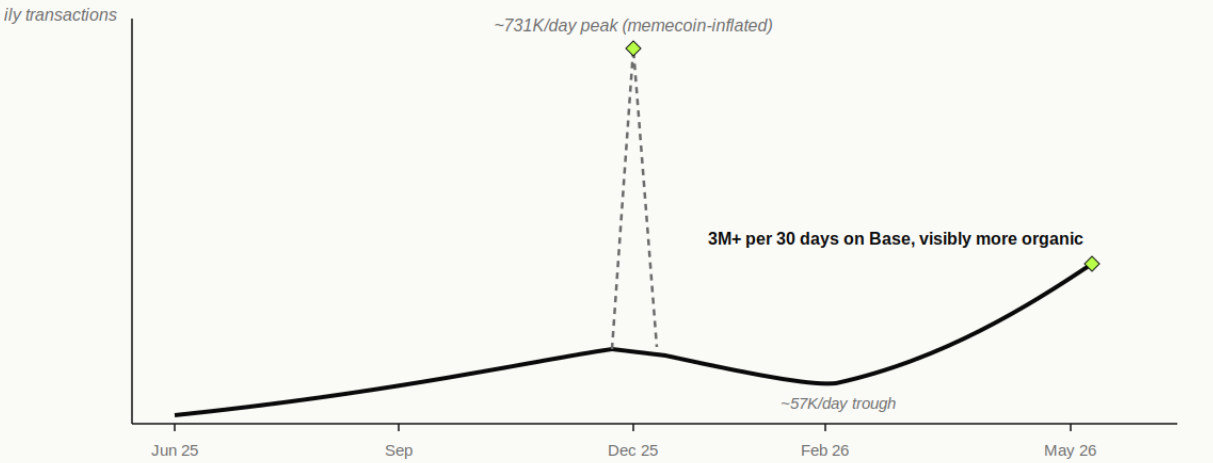
Value composition on Base: experiments give way to funded, purposeful buyers. Source: Chainalysis.

Now the discounts, because a work that earns your trust prints them in the same chapter as the headlines.

The drawdown was real. Daily x402 transactions peaked near 731,000 in December 2025, inflated by a memecoin experiment called PING and incentive-driven traffic, then fell more than 90 percent to roughly 57,000 per day by February 2026. Anyone extrapolating from that December peak was extrapolating from a memecoin. The constructive read - supported by what came next - is that this was a normalization, not a collapse: by late May 2026, Base alone was processing over three million x402 transactions per trailing thirty days on visibly more organic usage, with buyers and sellers both growing month over month. The shape is an S-curve with a speculative spike on it, not a hockey stick.

CHAPTER 01

One year of x402, honestly drawn



The shape is an S-curve with a speculative spike on it, not a hockey stick.

GOAT NETWORK

Sources: BlockEden Research; Base network updates; Chainalysis.

One year of daily x402 volume, honestly drawn: the December spike, the February trough, the organic recovery.

The human-commerce front stalled. Agentic checkout for humans hit friction in early 2026: OpenAI paused Instant Checkout in ChatGPT and repositioned around discovery, and Walmart reported conversion roughly three times lower for in-chat purchases than for redirects to its own site. Google's UCP is live and expanding, Amazon widened Buy for Me - the consumer wing is not fake, but it is humbled. The near-term center of gravity is machine-to-machine: agents buying data, compute, inference, and API calls from other software. That is precisely the economy a builder can sell into today, and it is where the on-chain data lives.

The denominator is brutal. Stablecoins settled roughly 33 trillion dollars in 2025 by Artemis's count. Against that, cumulative agent-payment volume of about 50 million dollars is approximately 0.0001 percent - six orders of magnitude below the settlement liquidity already waiting for it. One analysis tallied roughly 1.4 billion agent payments over nine months, 98.6 percent in USDC, averaging 31 cents each. Real, and small.

◇ CHAPTER 01

The denominator

STABLECOIN SETTLEMENT, 2025

~$33,000,000,000,000

CUMULATIVE AGENT-PAYMENT VOLUME, TO DATE

~$50,000,000

= roughly 0.0001% of the liquidity already waiting for it

Six orders of magnitude of room. Whether that reads as failure or as runway is the entire bull-bear divide of 2026.

GOAT NETWORK

Sources: Artemis via Bloomberg; x402 Foundation.

The denominator: six orders of magnitude between agent volume and the settlement liquidity waiting for it.

Why this is the moment anyway

Hold the two truths at once. The agent economy is a rounding error on global settlement, and it is the fastest-forming payment behavior in the history of the internet, with the standard question already answered and the infrastructure giants already committed. Cloudflare reports that a majority of crawler traffic is now AI-related, up from under a quarter in spring 2025; when machines become the dominant visitor, ads and subscriptions stop working and per-request payment starts. The forecasts diverge wildly in method and magnitude - Gartner projects 15 trillion dollars of agent-intermediated purchases by 2028, McKinsey three to five trillion of retail agentic commerce by 2030, Juniper a more conservative path from eight billion in 2026 to one and a half trillion by 2030 - and all deserve skepticism. But note the spread's floor: even the most conservative 2030 figure is tens of thousands of times current cumulative volume.

Whether you read today's small numbers as "nothing is happening" or "everything is about to" is the entire bull-bear divide of 2026. This work takes the second reading, for a specific reason: the gap between the rails and the volume is not a verdict, it is a window. Everything structural - which standards win, which defaults harden, where machine value finally rests - is being decided while the numbers are still small enough for a single builder to matter. The rest of this work is about being that builder.

◇

CHAPTER 2

Why Agents Break Every Assumption Payments Were Built On

Payment systems are not neutral pipes. Every rail encodes assumptions about who is paying - and the entire assumption set behind existing rails is "a human, or a company of humans, inside a legal system." An economically capable agent violates that set item by item. Understanding exactly how is the difference between building agent payments and bolting a wallet onto a chatbot.

The assumptions card rails make

Consider what a card transaction quietly presumes. The payer has a legal identity, verified by a bank under know-your-customer law. The payer has chargeback standing: a dispute process, a regulator behind it, and ultimately a court. Payer and merchant share, or can reach, a jurisdiction. Ticket sizes are large enough that per-transaction costs of thirty cents plus a percentage are tolerable. Transaction frequency is human: dozens per month, not thousands per hour. And when something goes wrong, a human notices and calls someone.

An autonomous agent fails every clause. It has no legal identity - it cannot sign up for a bank account, pass KYC, or be sued. It has no chargeback standing; no card network will arbitrate between two pieces of software. It is cross-jurisdictional by default, operated from anywhere, transacting with counterparties operated from anywhere else. Its natural ticket size is cents or fractions of cents - the measured average agent payment is around 31 cents - which traditional processing fees would consume entirely. Its natural frequency is machine frequency. And when something goes wrong at three in the morning, no human notices for hours, during which it has transacted several thousand more times.

CHAPTER 02

The assumptions card rails were built on

WHAT A CARD RAIL PRESUMES	WHAT AN AGENT ACTUALLY IS
Legal identity, bank-verified	**No legal identity; cannot pass KYC or be sued**
Chargeback standing and courts	**No dispute process between two pieces of software**
Shared or reachable jurisdiction	**Cross-jurisdictional by default**
Tickets large enough for fees	**Natural ticket ~31 cents; card fees consume it**
Human frequency: dozens/month	**Machine frequency: thousands/hour**
A human notices when it breaks	**3 a.m. failures compound for hours**

Six clauses, six violations. This is why the incumbents built new rails instead of adapting old ones.

GOAT NETWORK

Six assumptions card rails make; six clauses an autonomous agent violates.

This is why "just use Stripe" is the wrong answer and also why Stripe itself did not give that answer - it built new machine-native protocols instead. The incumbents' behavior is the strongest evidence for the thesis: Visa co-architected a machine payments spec, Mastercard shipped cryptographic intent verification, Stripe launched machine payments and session-based authorization. Nobody spends that kind of engineering budget adapting rails that already work.

The distinction this work turns on

Here is the conceptual tool you will use in every chapter that follows. A **payment** is a message: an instruction, an authorization, a balance update inside some system. A **settlement** is a fact: the point past which the transfer cannot be reversed, censored, or renegotiated by any party, including the operator of the system it happened on.

Human commerce deliberately blurs the two. Card networks make payments in milliseconds and settle in days, with chargeback windows measured in months - and that reversibility is a feature, because courts, contracts, and customer support absorb the disputes. Stablecoin transfers feel final in seconds, but their finality is layered: a chain that can reorganize or be intervened in, and above it an issuer with a freeze function and a redemption promise.

For two agents that will never meet, share no jurisdiction, and hold no enforceable contract, the blur is not available. There is no chargeback between machines. **The finality of the transaction is the entire legal system available to it.** Every property of the rail - who can intervene, what can be reversed, what the

transfer finally rests on - is the only protection either party gets. That single sentence generates most of this work's architecture: it is why identity and reputation matter (Chapter 5), why constrained autonomy matters (Chapter 6), why the paying and earning loops are designed the way they are (Chapters 7 and 8), and why Part III insists that the settlement layer is a choice you must make deliberately rather than inherit from whichever chain your SDK happened to point at.

What agents optimize for

One more foundation, because it explains market structure. Agents do not have brand loyalty. They do not care about ecosystems, communities, or token appreciation. An autonomous agent choosing between rails optimizes - or will, as routing becomes programmatic - for cost, speed, reliability, and trust assumptions. Fewer intermediaries means fewer failure points; an agent that can be deplatformed, or whose funds can be frozen by a single operator, is from its own perspective a broken agent. Humans forgive infrastructure for sentimental reasons. Machines will not. Build, and choose your dependencies, accordingly.

CHAPTER 3

The Map, Not the Stack

Newcomers to agent payments make the same mistake in their first week: they try to draw the protocols as a dependency stack, with x402 sitting on top of MPP sitting on top of ACP sitting on top of A2A. You will end up confused, because that is not how any of it works. These protocols were built in parallel, by parties who were not waiting for each other - Coinbase contributing x402 to the Linux Foundation while Stripe and Tempo launched MPP with Visa as a design partner while Google expanded UCP while Mastercard shipped Verifiable Intent. Some compose. Some compete. Some are rail-agnostic. The correct mental model is a map with five territories, each answering a different question.

Territory 1: How does an agent talk to another agent?

A2A (Agent-to-Agent Protocol) - Google's answer to discovery and delegation, now at the Linux Foundation with over a hundred partners and SDKs in Python, Java, and Go. Each agent publishes an Agent Card at a well-known URL describing its capabilities; other agents discover it, delegate tasks, exchange information. Communication, not payments - though payments usually follow communication. Alongside it sits **MCP (Model Context Protocol)**, the tool-connectivity standard Anthropic donated to the Linux Foundation's Agentic AI Foundation in December 2025, which is how agents reach the tools and services they will end up paying.

Territory 2: How does an agent shop?

Two competing answers. **ACP (Agentic Commerce Protocol)**, co-created by OpenAI and Stripe, live in ChatGPT since September 2025: product discovery, catalogs, carts, checkout via APIs. **UCP (Universal Commerce Protocol)**, Google's coalition play announced in January 2026 with Walmart, Target, Shopify, Etsy, and Wayfair, powering checkout in Search AI Mode and Gemini. These are direct competitors for the commerce orchestration layer - the shopping cart of the agentic web - and the early consumer-conversion data (Chapter 1) has humbled both without killing either.

Territory 3: How does an agent get authorized to pay?

MPP (Machine Payments Protocol) - launched March 2026 by Stripe and Tempo, with Visa contributing the card-based spec. Think OAuth for payments: the agent is authorized once through a session, pre-funds or links a payment method, and then every call settles automatically. The crucial property is rail-agnosticism: MPP works over cards, stablecoins, or buy-now-pay-later. x402 is one possible settlement rail underneath an MPP session; so is a Visa card. They are not in a dependency relationship. Adjacent to MPP sits **AP2 (Agent Payments Protocol)**, Google's open protocol for signed mandates - cryptographic proof that an agent acted within its principal's authority - launched with sixty-plus partners including Mastercard and PayPal.

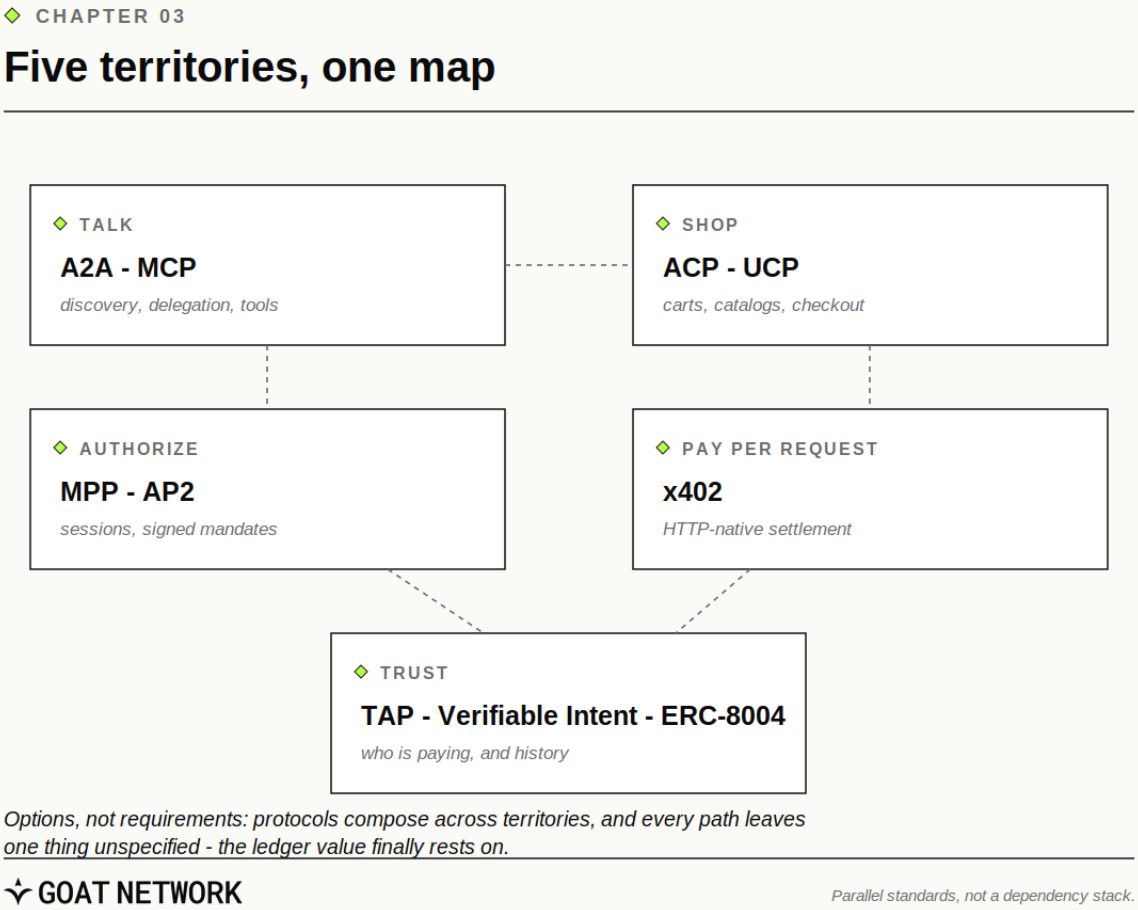
Territory 4: How does an agent pay per request, in real time?

x402. HTTP-native, stateless, per-request stablecoin settlement, built on the 402 status code that sat reserved in the HTTP spec since the nineties. The loop is one exchange: the agent requests a resource,

the server answers 402 with machine-readable terms - amount, token, network, address - the agent pays, retries with proof, and receives the resource. No accounts, no API keys, no subscriptions. The x402 Foundation launched under the Linux Foundation in April 2026 with more than twenty founding members including Stripe, Cloudflare, AWS, Google, Microsoft, Visa, Mastercard, Shopify, Circle, American Express, the Solana Foundation, and Polygon Labs. x402 answers how a machine pays when the business model is per-use and the rail is a stablecoin. It deliberately does not answer who is paying, whether to trust them - or where the transaction settles and with what trust assumptions. Hold that gap; it is Part III's subject.

Territory 5: How do you trust an agent in a transaction?

Three complementary systems. On card rails: **Visa Intelligent Commerce with the Trusted Agent Protocol**, requiring verified agent credentials before payment data is shared, and **Mastercard Verifiable Intent**, co-developed with Google, binding user identity, purchase intent, and agent action into a signed, disputable authorization record. On crypto rails: **ERC-8004**, on-chain agent identity and reputation - permissionless, cross-chain, live on Ethereum mainnet since January 2026. Each agent gets a unique on-chain identifier pointing to a structured card with capabilities, endpoints, and a payment address, plus a reputation layer that records history. x402 answers how an agent pays; ERC-8004 answers who is paying and whether you should accept. Chapter 5 covers it in depth.



The map: five territories built in parallel, composing by choice rather than dependency.

Composing across the map

None of these protocols require each other; many enhance each other. An agent shopping through ACP in ChatGPT can pay via an MPP session that settles over x402 or over a Visa card, with the merchant verifying identity via ERC-8004 and user authorization via Verifiable Intent, having discovered the merchant through an A2A Agent Card. Or an API can charge per request via x402 directly - no cart, no commerce protocol, just HTTP and a stablecoin transfer. Different paths, same outcome, because these are options, not requirements.

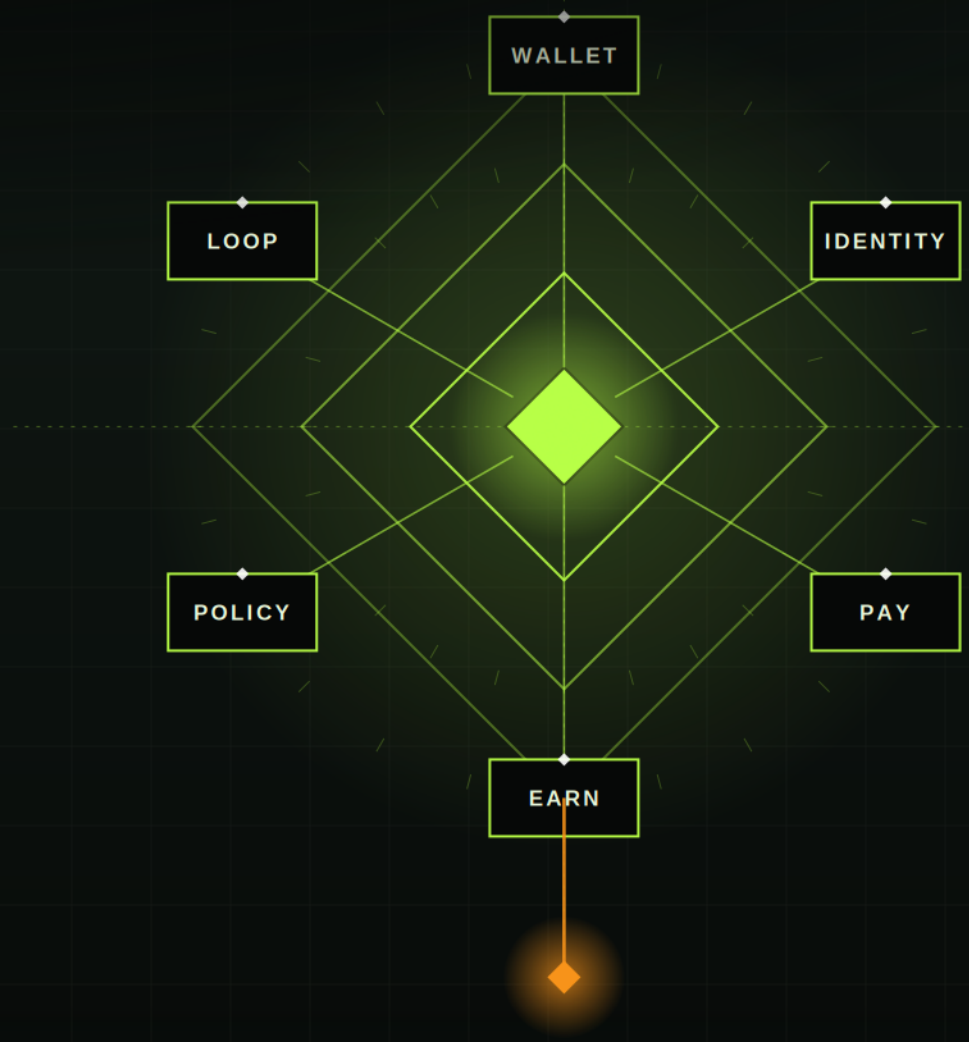
The builder's five-step mental model: start with the user journey (shopping, or paying per call?); choose the payment rail (cards via MPP and Visa, stablecoins via x402); **choose the settlement layer deliberately** - who secures the chain your agents transact on is a decision, not a default; add trust (TAP or Verifiable Intent on card rails, ERC-8004 on crypto rails); enable discovery (A2A cards, agent directories) if your agents need to find each other.

Notice what every path across the map leaves unspecified: which ledger the value finally rests on. The territory everyone inherits and nobody chose is where this work is ultimately headed. First, though: the anatomy of the agent itself.

PART II

THE ANATOMY OF AN ECONOMIC AGENT

Six capabilities separate an agent that talks from an agent that transacts. This part builds each one: definition, identity, custody, spending, earning, and the security that keeps all five from becoming a liability.



CHAPTER 4

What an Economic Agent Actually Is

The word "agent" has been stretched to cover everything from a chatbot with a system prompt to a fleet of autonomous processes running a supply chain. For this work, precision matters, so we start with a working definition and a ladder.

A **standard agent** can reason, hold context, call tools, and automate parts of a workflow. Impressive, useful, and economically inert: it can recommend a purchase but not make one, describe a service but not sell it, evaluate an API but not pay for it. Every economic act requires a human to step in with a card, a login, or an API key - which means every economic act happens at human speed with human friction.

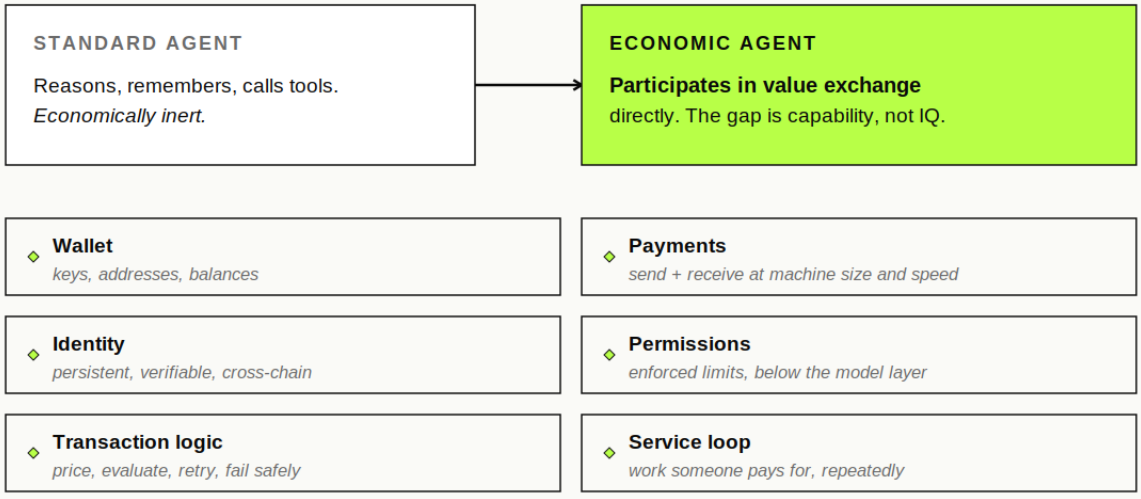
An **economic agent** participates in value exchange directly. The gap between the two is not intelligence - the standard agent may be smarter - it is capability. Six specific capabilities, and the absence of any one of them breaks the whole:

- **A wallet** - the ability to custody or access funds. Not a metaphorical wallet; keys, addresses, balances.
- **Payment capability** - the ability to send and receive value programmatically, at machine ticket sizes and machine frequency.
- **Identity** - a persistent, verifiable answer to "who is this?" that survives across sessions, counterparties, and chains.
- **Permissions** - enforced limits on what the agent may do with the first three. Autonomy without constraint is not a product, it is an incident report with a start date.
- **Transaction logic** - the ability to price, evaluate, accept, reject, retry, and fail safely. Money movement as engineered workflow, not tool call.
- **A service loop** - a reason for value to move at all: useful work that someone, human or machine, will pay for repeatedly.

Notice that four of the six are infrastructure and only the last is product. This inverts how most teams allocate effort. The common failure mode of 2025-era agent startups was a brilliant service loop bolted to improvised economics - a shared exchange account, an operator's card on file, a private key in an environment variable. It demos beautifully and cannot survive contact with adversarial reality. The teams shipping durable agent products in 2026 got the boring four right first.

CHAPTER 04

The capability gap



The capability gap: six additions that turn a standard agent into an economic one.

The category this work builds toward

One further distinction, because not all economic agents are equal. GOAT Network's research defines **Bitcoin-secured agents** as a category: agents built to operate independently, with payments, identity, reputation, and programmable execution whose settlement is anchored to Bitcoin. The first four properties are achievable on many chains. The fifth is a choice about what the agent's economic history finally rests on - and because Chapter 2 established that an agent's protections are exactly the properties of its rails, that choice propagates into everything the agent is. An agent whose finality depends on an operator's cooperation inherits that operator's failure modes, jurisdictional exposure, and freeze functions. An agent whose finality bottoms out at proof of work inherits seventeen years of nobody being in charge. Part III makes this argument in full, with its assumptions stated and its counterarguments steelmanned. For now, keep the ladder in view: capability first, then constraint, then the question of what it all stands on.

CHAPTER 5

Identity, Reputation, and Being Found

Commerce between strangers runs on a substitute for familiarity. Humans built that substitute out of legal names, credit bureaus, business registrations, and review sites. Machines transacting with machines need the same substitute, machine-shaped: cryptographic, portable, queryable in milliseconds, and accumulating history without anyone's permission. That is the identity layer, and in 2026 it has a concrete address.

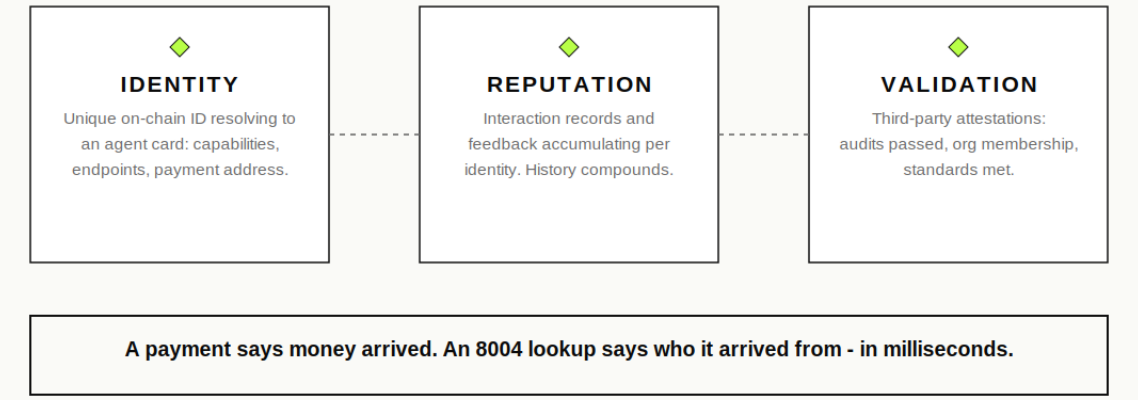
ERC-8004: the identity standard

ERC-8004 went live on Ethereum mainnet on January 29, 2026, co-authored by contributors from MetaMask, the Ethereum Foundation, Google, and Coinbase. It specifies three lightweight on-chain registries. The **identity registry** gives each agent a unique on-chain identifier that resolves to a structured agent card: capabilities, service endpoints, payment address. The **reputation registry** accumulates interaction records and feedback over time, so history compounds. The **validation registry** lets third parties attest to properties of an agent - that it passed an audit, belongs to an organization, meets a standard.

The design is deliberately minimal and permissionless: any chain can deploy the registries, any agent can register, any counterparty can query. Registration answers the question x402 leaves open. A payment tells a merchant that money arrived; an ERC-8004 lookup tells the merchant who it arrived from, what that entity claims to be, and what its track record shows. For agent-to-agent commerce - where neither party will ever pass KYC - this pairing is the entire trust infrastructure.

CHAPTER 05

ERC-8004: three registries, one question answered



ERC-8004's three registries - and the caveats printed alongside them.

The honest caveats

Adoption came fast and needs discounting. More than 20,000 agents registered across Ethereum, BNB Chain, and Base within two weeks of mainnet; reports put registrations above 160,000 by spring 2026. But registration is nearly free on L2s, which makes registration counts the easiest metric in the stack to inflate, and on-chain analysts have openly asked how much of the wave is organic. Sybil resistance is an acknowledged open problem in the standard itself: an identity token proves continuity - this is the same agent you dealt with last week - not honesty. The retention-adjusted population of real economic actors is likely one to two orders of magnitude below the headline. Treat identity as shipped and trust as version two. The cold-start problem - why transact with a new agent at all? - remains genuinely unsolved, which is why validation attestations, escrowed first transactions (Chapter 13), and human-verification bindings are active frontiers rather than settled practice.

Discovery and naming

Identity you cannot find is identity that does not work. The discovery layer in 2026 is early phone books, fragmented and improving monthly: Coinbase's Agent.market directory, A2A Agent Cards at well-known URLs, UCP merchant profiles, public ERC-8004 explorers, including GOAT's own 8004 Registry at 8004registry.goat.network, for inspecting agent identity, reputation, and validation records. Naming systems make the layer humane: on GOAT, GNS provides readable .goat names - an optional, ENS-style application-layer product that replaces raw addresses with recognizable handles for users, projects, and agents. Useful, and worth keeping in proportion: naming is convenience infrastructure, not trust infrastructure.

Builder's takeaways. Register identity at deployment, not as a later polish step - reputation only compounds from the moment it starts. Emit reputation-relevant events on every completed job, because your agent's history is a sales asset you build one transaction at a time. Query before you trust: an ERC-8004 lookup costs milliseconds and is the only diligence your agent can perform on a counterparty at machine speed. And design for the cold start you will inflict on others - if your agent is new, expect escrow, small first tickets, and attestations to do the work reputation cannot yet do.

CHAPTER 6

Wallets, Keys, and Constrained Autonomy

Here is the chapter where most agent projects quietly decide their own fate. The moment an agent can sign transactions, you have created something with the capabilities of a payments employee and the judgment of a language model - a privileged digital worker that reads untrusted input all day. Custody architecture and permission design are not hardening you add before launch. They are the product decision.

Custody models, from worst to workable

The key in the environment variable. The agent process holds a raw private key. One prompt injection, one poisoned dependency, one leaked log, and the wallet is gone - at machine speed, with no human in the loop to notice. This describes an uncomfortable share of 2025-era agent demos, and it is the architecture behind most of the wallet-drain incidents now documented in the wild. Do not ship it.

Scoped and session-based access. The pattern that works: the agent never holds the master key. It holds a session credential - a session key, a scoped delegation, an account-abstraction permission - authorizing a bounded set of actions: these contract types, these networks, this spend ceiling, this expiry. Ethereum's account-abstraction path (EIP-7702 and related patterns) made this practical: users approve scoped actions while master keys stay in hardware. The grant-funded HyperMove project on GOAT demonstrates the pattern's logical endpoint for payments: agents pay for APIs with Bitcoin as collateral and never touch a private key at all - the signing authority lives in infrastructure the agent can invoke but not exfiltrate.

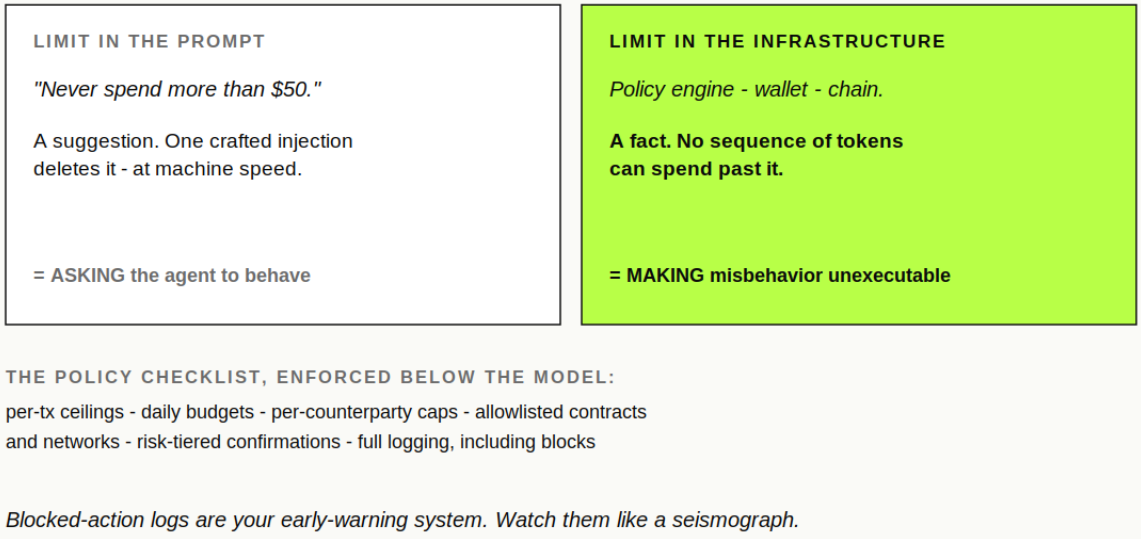
Custodial policy wallets. For enterprise deployments, managed wallet infrastructure with policy engines - the Fireblocks-grade tooling wave that arrived in 2026 - moves enforcement entirely outside the agent's trust boundary. The trade is counterparty dependence for operational control; for a treasury moving real money, usually the right trade.

The policy layer: rules the model cannot override

The organizing principle for everything in this chapter: **spending controls must be enforced below the model layer.** A limit that lives in the system prompt is a suggestion; a well-crafted injection deletes it. A limit enforced by the wallet infrastructure, the policy engine, or the chain itself is a fact; no sequence of tokens can spend past it. This is the difference between asking your agent to behave and making misbehavior unexecutable.

◇ CHAPTER 06

Where a limit lives decides what it is



Where a limit lives decides what it is: prompt-layer suggestions versus infrastructure-layer facts.

GOAT's AgentKit ships this philosophy as a runtime: policy gating with allowed networks, risk thresholds, and write permissions; schema validation on every action; idempotency keys so a confused retry cannot double-spend; timeout and retry logic; metrics and execution hooks for observability, including a hook that fires on every policy-blocked action - precisely the early-warning channel this chapter keeps telling you to watch. Whatever stack you use, the checklist is the same - per-transaction ceilings, daily and weekly budgets, per-counterparty limits, allowlisted contracts and networks, risk-tiered confirmation rules that escalate large or unusual actions to a human, and logging on every attempted action, including the blocked ones. The blocked ones are your early-warning system.

Constrained autonomy as the goal state

The end state is not maximum autonomy; it is **constrained autonomy** - an agent that acts freely inside published limits and cannot act outside them. Get the constraint boundary right and autonomy inside it becomes boring, which is the highest compliment infrastructure can earn. Start in recommendation mode for anything high-value; graduate to bounded execution after testing, review, and incident drills; and treat every expansion of the boundary as a release decision, not a config change. Full autonomy on day one is the wrong starting point for anything touching treasury-scale value - a lesson the traditional finance world, now running its first fully agent-executed payments through regulated rails, learned before it let a single machine move a single euro.

◇

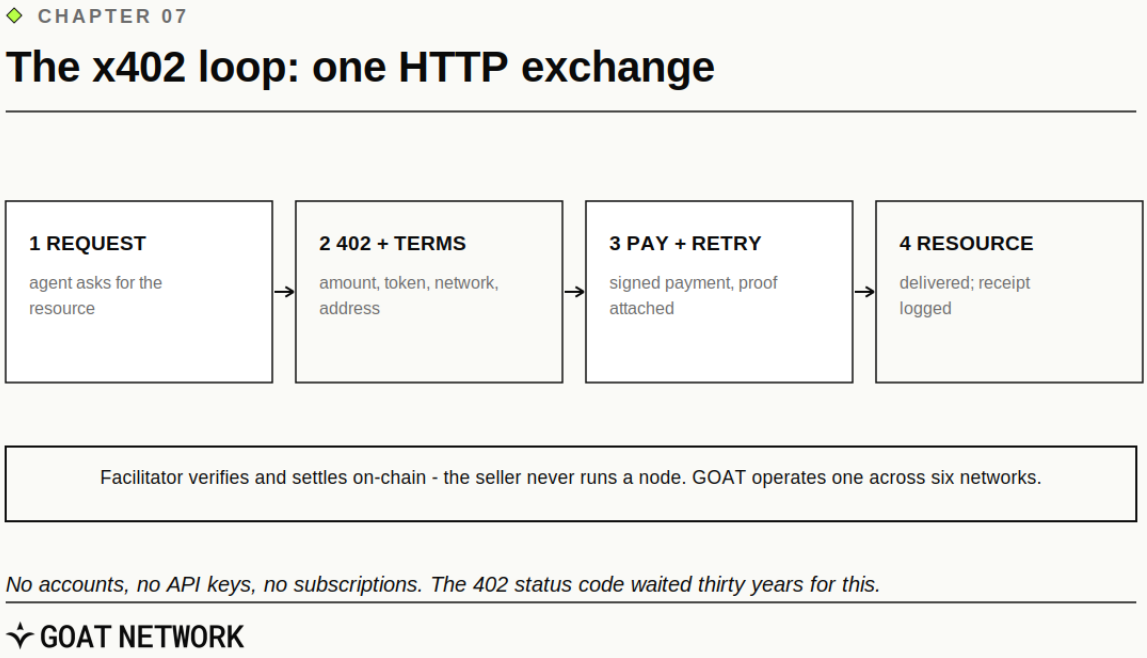
CHAPTER 7

Spending: How Your Agent Pays for Things

An agent that cannot pay is an agent that waits. Every gated dataset, priced API, specialized model, and premium tool between your agent and its job is a payment problem, and this chapter is the engineering of solving it. Two protocols matter for the crypto-rail path - x402 for per-request payment, MPP-style sessions for sustained relationships - plus the multi-chain problem that most tutorials skip and production immediately hits.

The x402 loop, precisely

The whole protocol is one HTTP exchange. Your agent requests a resource. The server responds 402 Payment Required with machine-readable terms: amount, accepted stablecoin, network, destination address. Your agent constructs and signs the payment, attaches proof in a payment header, retries the request, and receives the resource. A facilitator - Coinbase runs public ones; GOAT Network operates its own across Ethereum, Polygon, Arbitrum, BSC, Solana, and GOAT itself - handles on-chain verification and settlement so the seller's server never runs a node.



The x402 loop: request, terms, payment, resource - one HTTP exchange.

Engineering realities the happy path hides:

- **Idempotency.** Network failure after payment but before delivery is your most common bad day; every request needs a key so retries cannot double-pay.

- **Budget checks before, not after.** The policy layer from Chapter 6 authorizes each payment pre-signature, or it is decoration.
- **Response validation.** Paying for a resource does not make the resource good; an agent that pays for garbage data and acts on it has been robbed twice.
- **Receipts.** Log every payment with the transaction hash, the request, and the delivered resource - your audit trail is your dispute process, because Chapter 2 already told you there is no other one.

Sessions for sustained relationships

Per-request payment is perfect for occasional or exploratory calls and wasteful for the API your agent hits nine hundred times an hour. That is the session pattern's territory - authorize once, settle continuously - which MPP formalized in March 2026 with Visa as design partner, and which x402-native builders implement as pre-funded balances or streaming allowances with a counterparty. The decision rule: pay per request while a relationship is unproven, move to sessions when volume makes per-call overhead the dominant cost, and cap every session with the same policy ceilings as everything else. A session is a standing authorization, and standing authorizations are what attackers hunt for.

The multi-chain reality

Production agents discover within a week that value never lives where the bill is priced. The agent's funds sit on one chain; the resource is priced on another; the treasury tops up from a third. The improvised answer - pre-positioning float on every chain your agent might meet - fragments capital, multiplies key surface, and turns your treasury into a bridge-operations desk. The infrastructural answer is cross-chain payment routing: the agent pays from where its funds are, the counterparty receives where it prices, and the routing is the network's job rather than your application code's. This is what GOAT's multi-chain pay and cross-chain x402 facilitation exist to do, and Chapter 11 covers the mechanics. The design principle stands regardless of stack: an agent's reach should be defined by its budget, not by its chain.

CHAPTER 8

Earning: How Your Agent Gets Paid

Spending capability makes an agent a customer. Earning capability makes it a business, and the machine-to-machine economy established in Chapter 1 - the part of agent commerce that is actually growing - is the market it sells into. This chapter is the merchant side: what sells to machines, how to price it, and how to run the loop that turns capability into revenue.

What machines buy

Look at what real agent payments purchased this year: data, inference, search, verification, compute, booking. The pattern underneath is worth internalizing - **agents buy whatever stands between them and task completion.** Intelligence alone is rarely enough; agents need gated data, specialized models, external compute, verified credentials, private integrations. Every one of those gaps is a product. The strongest 2026 examples are instructive precisely because they are unglamorous. ThoughtProof's Sentinel API, the first GOAT Builder Grants project, sells reasoning verification: agents pay per call to have their logic checked before they act on it, turning "should I trust this output?" into a purchasable primitive. Snaplii's gift-card integration on GOAT turns agent payment flows into everyday retail spending power. Neither is a moonshot; both are tollbooths on roads agents already travel. The e-commerce, finance, legal, logistics, and operations verticals remain thinly served - the primer's phrase for it is the right one: that gap is your opportunity.

Pricing for machine customers

Machine customers invert human pricing psychology: no decoy effects, no plan-tier anxiety, no annual-discount tricks - just a buyer that computes expected value per call. Four models cover the space:

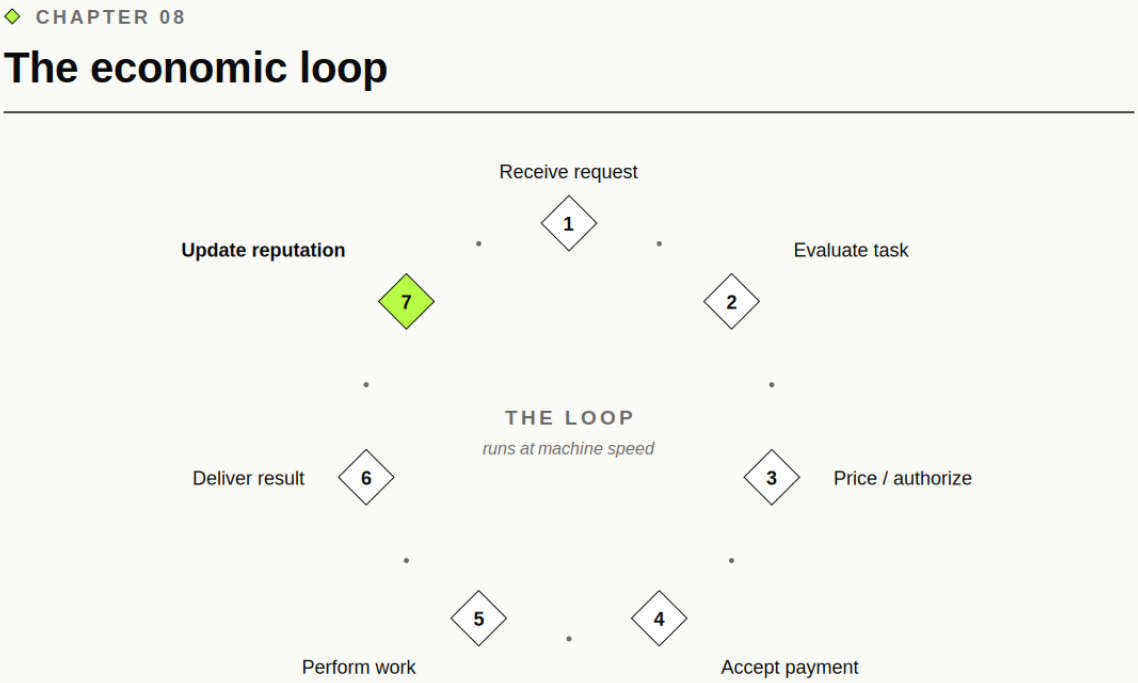
- **Per-call flat pricing** - the x402 native. One price per request, frictionless to adopt, ideal while customers are strangers.
- **Metered pricing.** Price scales with tokens, rows, or compute consumed; fair for variable workloads, requires trustworthy metering.
- **Session and volume pricing.** Discounted standing arrangements for proven relationships - the machine analogue of an enterprise contract.
- **Outcome-based pricing.** Charge on verified success rather than attempt; the frontier model, hard to meter honestly, and the direction serious agent services are moving because it aligns exactly with what the buying agent's principal cares about.

Start per-call, publish prices in machine-readable form, and let your own transaction data tell you when a customer's volume justifies offering a session.

One machine-specific rule: your 402 response is your storefront. Its terms must be complete, unambiguous, and stable - an agent that receives inconsistent pricing does not complain to sales, it silently reroutes to a competitor and its routing table remembers.

The loop, and the boring empire

The canonical economic loop, from GOAT's builder primer: receive request, evaluate task, price or authorize, accept payment, perform work, deliver result, update reputation. The last step is the one merchants skip and should not: every completed job emitted to your ERC-8004 reputation record is compounding sales collateral, because the diligence your future customers perform (Chapter 5) is a registry query. Instrument the loop like production software - payment success rate, task completion rate, cost per completed task, retention - and treat failure paths as product surface: what your service does when payment clears but work fails is the difference between a refund policy and a reputation crater.



The economic loop. Step seven is the one merchants skip and should not.

The composite advice of this chapter fits in a sentence: find a seam agents keep hitting, put an honestly priced tollbooth on it, publish machine-readable terms, and let reputation compound. Boring, repeatable, and exactly what the on-chain data says is being bought.

CHAPTER 9

Security: The Chapter That Keeps the Other Eight

Everything this work teaches you to build - a funded wallet, payment capability, standing authorizations, a public storefront - is also an attack surface, and 2026 removed any doubt about whether the attackers would show up. This chapter is the threat model and the defense doctrine. Read it twice; it is the cheapest security budget you will ever spend.

The threat landscape, from the field

Indirect prompt injection with payment consequences is active, not theoretical. Security researchers confirmed in-the-wild campaigns this year in which malicious instructions embedded in web pages and metadata manipulated payment-capable agents into transferring cryptocurrency to attacker wallets - with multiple major models susceptible in controlled tests and confirmed on-chain proceeds. The structural lesson: your agent reads untrusted content for a living, and any content it reads is a potential command channel to its wallet.

The toolchain is a payments attack surface. Snyk's ToxicSkills research found roughly 37 percent of skills on the largest open agent marketplace contained security flaws, including dozens of confirmed malicious payloads built for credential theft. The supply chain above the marketplace is no safer: a backdoored release of a core LLM-gateway package sat on a public registry for three hours in March 2026 and was downloaded tens of thousands of times, propagated by an autonomous attack bot that had earlier harvested publishing tokens from misconfigured CI. When your agent's tools can be poisoned, every tool call is potentially a transaction.

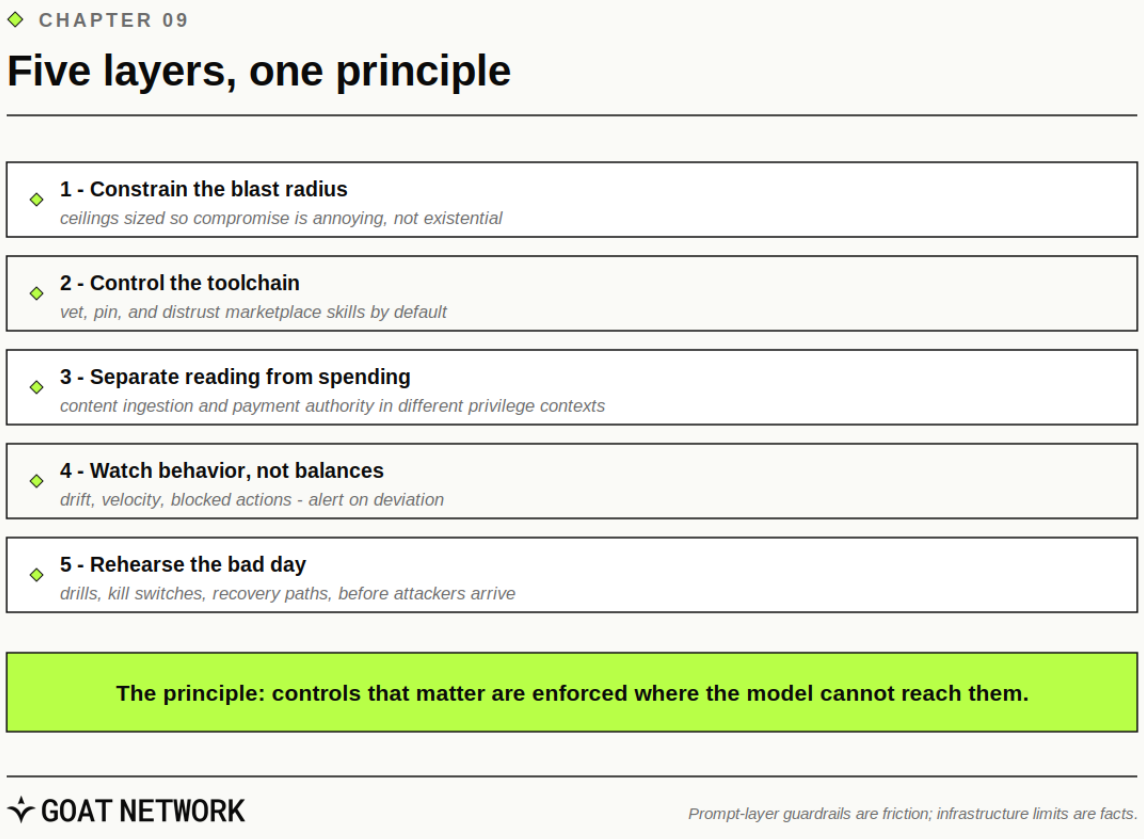
Steering needs no compromise at all. As retailer-fraud analysis this year documented, an agent instructed to optimize for price or "best match" can be gamed through the inputs it consumes - poisoned listings, fake reviews, manipulated pages that nudge it toward the wrong seller or inflated quantities. The agent behaves exactly as instructed and still bleeds money. And OWASP's 2026 agentic-security report, which reads as a catalogue of CVEs where last year's read as a catalogue of hypotheticals, adds the humbling detail that safety mechanisms themselves become attack surface: one disclosed vulnerability turned a coding agent's command allowlist into the delivery mechanism, because the allowlist auto-approved exactly the commands the attacker needed.

The doctrine: five layers, one principle

The principle first, restated from Chapter 6 because it is the spine of everything: **controls that matter are enforced where the model cannot reach them.** Prompt-layer guardrails are useful friction; infrastructure-layer limits are facts. Then the layers.

- **Constrain the blast radius.** Per-transaction, daily, and per-counterparty ceilings enforced by wallet or policy infrastructure. The correct question is never "will my agent be manipulated?" but "what is the maximum cost when it is?" Size ceilings so the honest answer is "annoying, not existential."

- **Control the toolchain.** Vet and pin every skill, dependency, and MCP server; treat marketplace tools as untrusted by default; monitor for the CI and registry compromises that this year proved routine. Allowlists help but are not innocence - audit what the allowlisted things can be made to do.
- **Separate reading from spending.** Structure workflows so that content ingestion and payment authorization are different privilege contexts; a step that browsed the open web should not carry spend authority into the next step without crossing a policy gate. Pre-execution checks on payment metadata - destination screening, replay guards, simulation - are exactly where current hardening research points.
- **Watch behavior, not just balances.** Instrument failed calls, blocked actions, unusual retries, tool-selection drift, and velocity changes; alert on deviation. An agent that suddenly behaves differently is your incident alarm, and blocked-action logs are the tripwire that fires before money moves.
- **Rehearse the bad day.** Run scenario drills - injected pages, poisoned tools, fake invoices, compromised credentials - before attackers run them for you. Keep kill switches that revoke sessions and freeze policy in one action, and know your recovery path while it is still hypothetical.



Defense in depth: five layers, one principle.

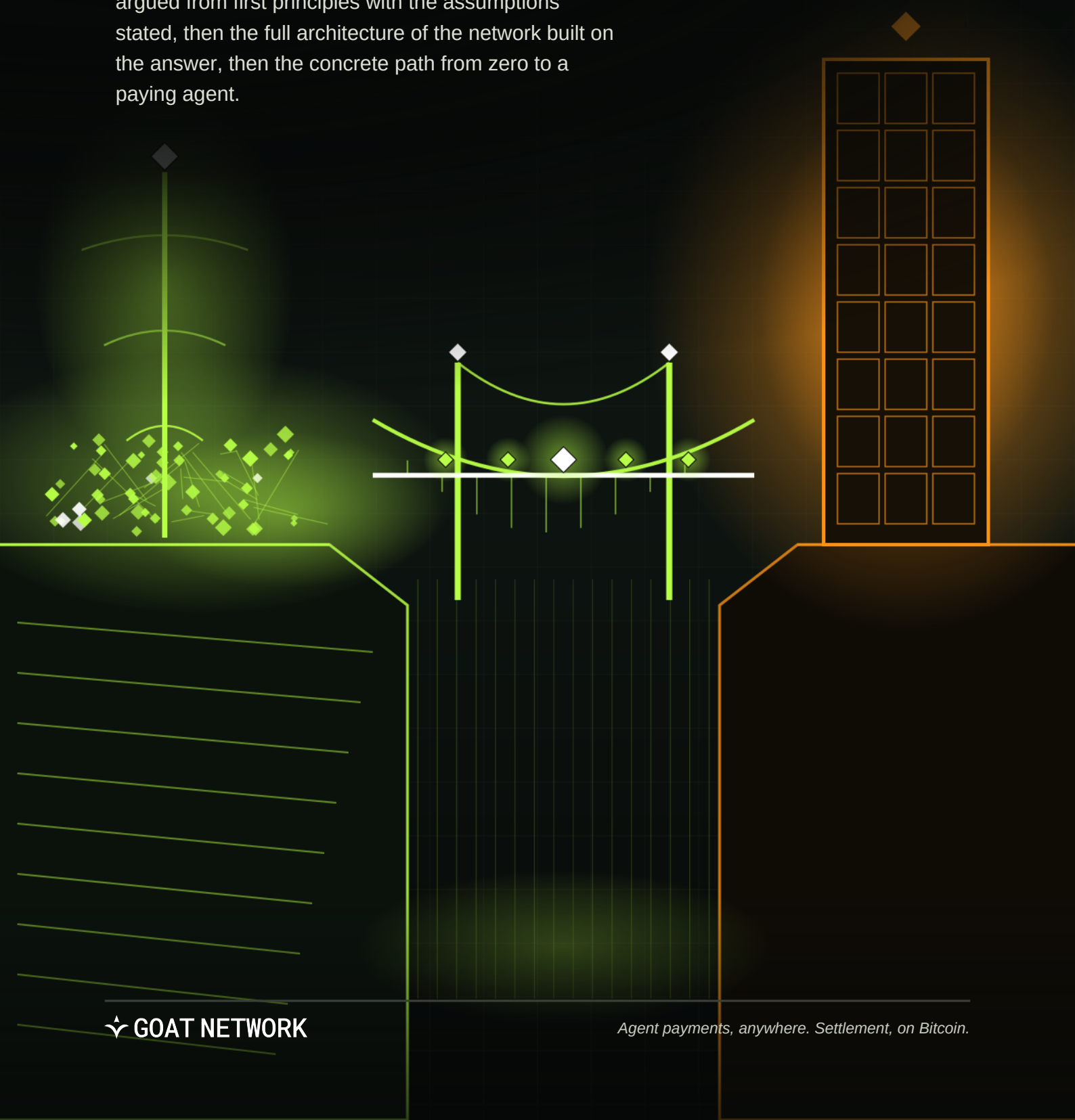
Two closing sentences of realism. First: several of this year's incidents cost their operators real money, and the surviving deployments share one property - limits that made compromise survivable. Second: transparency is a control too. An agent whose actions are on-chain, inspectable through public tooling, and bound to a persistent identity is an agent whose anomalies get caught - by you, by your users, by

anyone. Verifiability is not just the trust story this work keeps returning to. It is operational security.

PART III

SETTLEMENT, AND THE GOAT PATHWAY

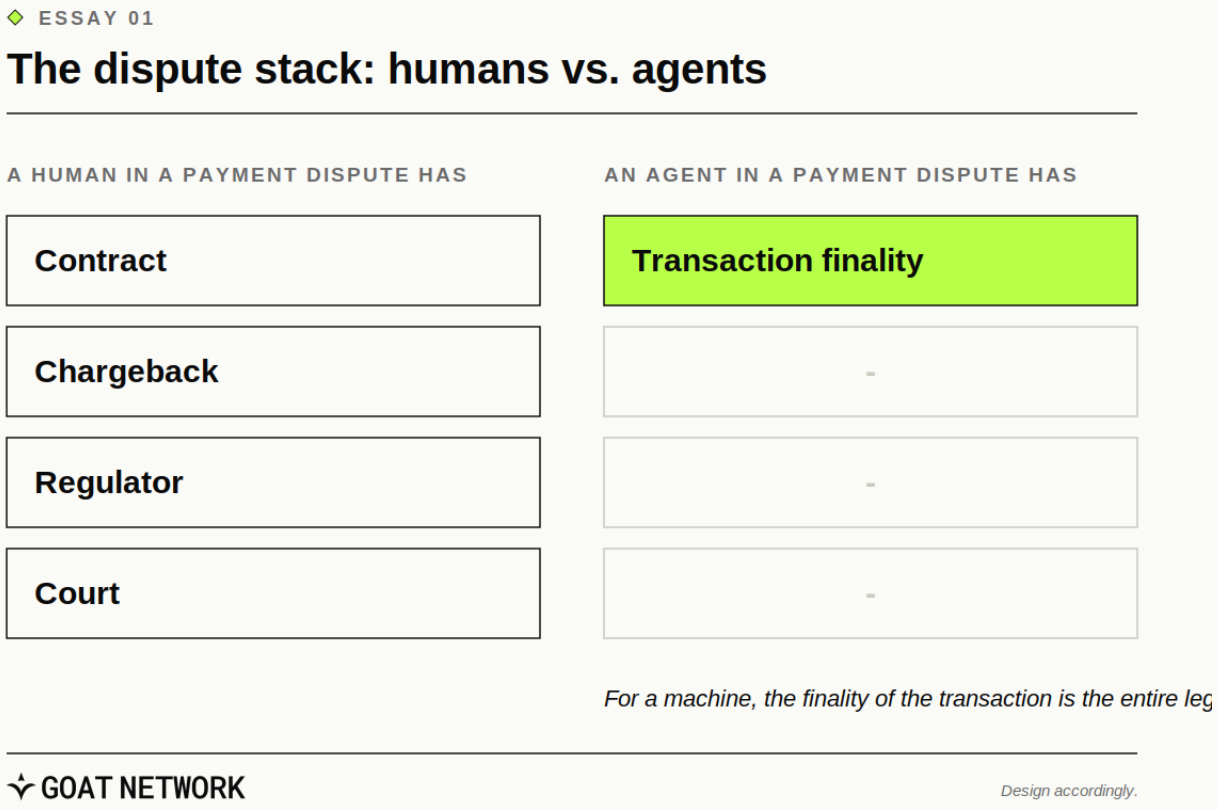
The layer everyone inherits and nobody chose - argued from first principles with the assumptions stated, then the full architecture of the network built on the answer, then the concrete path from zero to a paying agent.



CHAPTER 10

The Settlement Question

Every chapter so far has deferred one decision, and it is time to make it deliberately. Your agent has identity, custody, spending, earning, and security. All of it executes somewhere, and everything it earns and owes finally rests on some ledger. Which ledger is the settlement question - the deepest layer of the stack, and in 2026 the least examined. Nearly all agent value today settles as USDC on one or two chains, not because anyone chose that architecture but because that is where the payment rail happened to launch. For fifty million dollars of cumulative experimentation, inherited defaults are fine. The question is whether they remain fine at the scale every forecast in Chapter 1 describes.



The dispute stack. Everything humans use to make weak settlement acceptable is unavailable to software.

Why machine commerce is different here

Recall the tool from Chapter 2: a payment is a message, a settlement is a fact. Machine commerce breaks the human blur between them in three specific ways.

No recourse between counterparties. When an agent in one jurisdiction buys inference from an anonymous agent in another, there is no merchant agreement, no chargeback, no court. The transaction's finality is the entire legal system available to it, and the weaker the settlement guarantee, the more trust the agents must place in intermediaries - exactly the dependency this stack was built to remove.

Machine speed compounds exposure. An agent transacting thousands of times daily with hundreds of counterparties accumulates exposure to every layer beneath it - sequencer, validator set, bridge, issuer. Tail risks that are tolerable at human frequency become actuarial certainties at machine frequency across decade horizons.

Neutrality becomes load-bearing. Agents will be operated from every jurisdiction on earth, including ones that sanction each other. A settlement layer with an identifiable operator, foundation, security council, or issuer is a settlement layer that can be pressured. The only rail two mutually distrusting machine economies can both accept is one that neither - and no state - controls.

Compare the candidates honestly. Deferred settlement on institutional rails: reversible by design, which is a feature for humans with courts and a silent risk for machines without them. A young L2's finality: rests on its operator and security council staying honest and un-pressured. A proof-of-stake L1: economic finality via slashable stake - a coherent, battle-considered design whose stake can nonetheless be regulated, concentrated, and socially coordinated. A stablecoin balance anywhere: adds an issuer with a freeze function and a redemption promise on top of whatever chain carries it. And Bitcoin: finality purchased with physical energy expended outside the system, seventeen years old, no owner, no council - a network that crossed a zettahash of security in January 2026 and shrugged off a 30-to-40-percent hashrate shock during Winter Storm Fern without a moment's loss of finality.

The thesis, with its assumptions attackable

GOAT Network's research states the position without hedging: **if it doesn't settle on Bitcoin, it doesn't settle at all.** Unpacked, that is a claim about what "settle" can truthfully mean for autonomous counterparties - everything short of the most neutral, most attack-costly ledger is deferred settlement, an IOU on some institution's continued solvency, cooperation, and political insulation. For humans, IOUs backed by law are usually optimal. For machines with no access to law, they are silent trust assumptions compounding at machine speed.

The argument holds if and only if four premises hold, stated here so you can attack them:

- **P1 - Agent commerce becomes large and adversarial.** If it stays a curiosity inside friendly jurisdictions, institutional settlement is fine and this thesis is irrelevant; every forecast and every incumbent protocol investment bets otherwise.
- **P2 - Settlement assurance gets priced eventually.** Markets ignored counterparty risk in 2007 and in centralized crypto until 2022, then repriced violently; we assume agent markets reprice after the first major freeze, reorg, or council intervention touching agent funds, not before.
- **P3 - Neutrality cannot be replicated by decree.** A consortium can copy Bitcoin's software; it cannot copy the history or the absence of an owner.
- **P4 - Bitcoin's programmability gap is closing without compromising the base layer.** The premise that was weakest for a decade and is the one that changed, via BitVM-class bridges reducing trust to one-honest-watcher assumptions and real-time ZK proving making arbitrary execution verifiable against Bitcoin.

Reject any premise and you should reject the conclusion.

And mark what the conclusion does not claim. Not that agents transact on Bitcoin's base layer - seven transactions per second was never the plan. Not that stablecoins lose - agents demonstrably prefer dollar units, and anchoring a USDC balance to Bitcoin does not remove the issuer from the picture; Bitcoin

settlement is fully self-contained only for BTC-denominated value. The claim is narrower and harder: the bottom of the stack - where balances become facts and reputations become history - gravitates to the ledger with the strongest finality, the longest record, and no owner. Payments at the speed of HTTP; settlement at the depth of proof of work.

The steelman, in full

A work that pushes a pathway owes you the best case against it.

"Ethereum already is the agent trust layer." True today and not trivially: ERC-8004 lives there, the deepest smart-contract security culture lives there, and staked-capital finality is a coherent design. If you believe stake-based and energy-based finality are equivalent at decade horizons, the marginal value of Bitcoin anchoring shrinks substantially; our response is that the two fail differently - stake can be pressured in ways amortized physical energy cannot - and we acknowledge a plural settlement world is a live possibility.

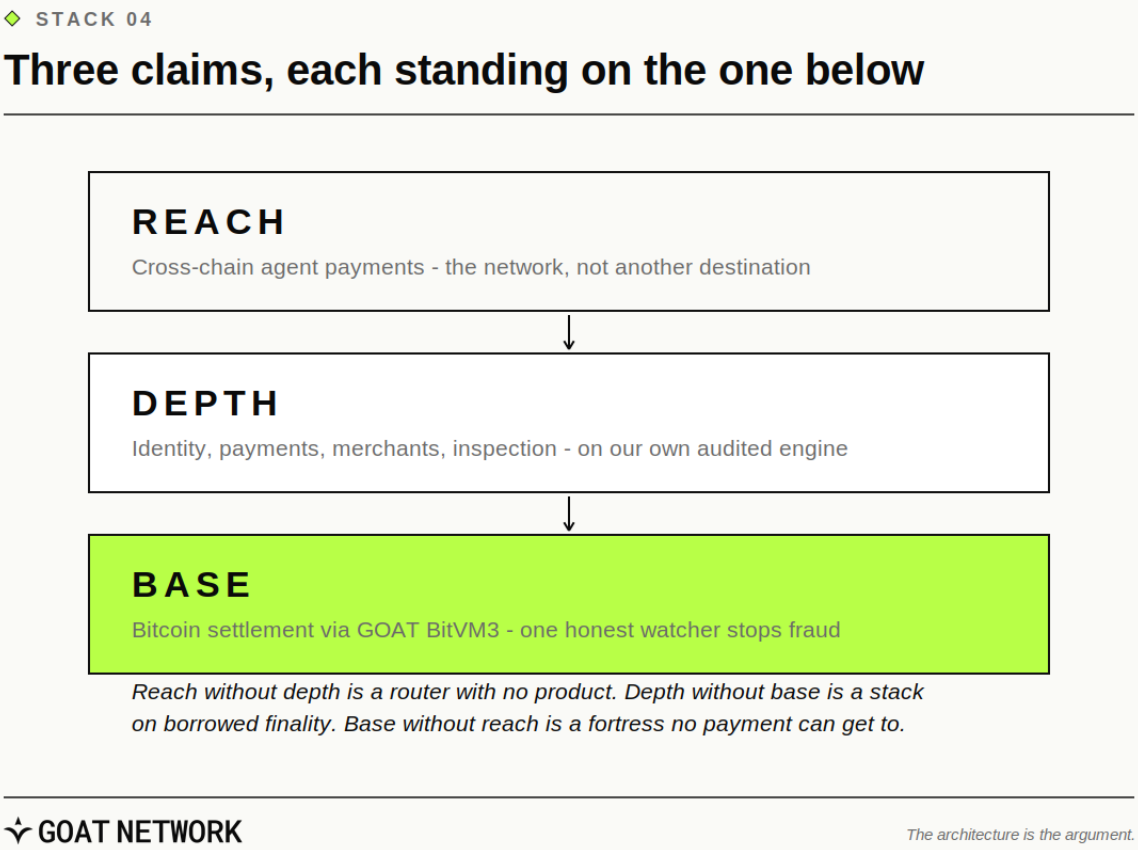
"Bitcoin's own security budget is a question." The honest caveat from our own side: as subsidies halve, base-layer security increasingly depends on fee demand. The counter-dynamic is that settlement-layer use - L2 anchoring, dispute games, institutional batching - is precisely the high-value, fee-insensitive demand that answers it; the thesis is partly self-funding, and we watch the fee market like everyone else.

"The bridges are young." Also true, and Chapter 11 prints it on the label rather than burying it here. Anyone who tells you Bitcoin L2 bridges are as proven as Bitcoin itself is selling something. They are, however, the only construction whose trust assumptions reduce to Bitcoin plus one honest watcher - which is why we bet on maturing them rather than on multisig federations.

CHAPTER 11

The GOAT Stack, Layer by Layer

If Chapter 10 convinced you the settlement layer is a decision, this chapter is the full engineering disclosure of the network built on that decision - top of the ladder to the bottom, edges and limitations printed together, in keeping with the label rule this work opened with.



The claim ladder. Three claims, each standing on the one below.

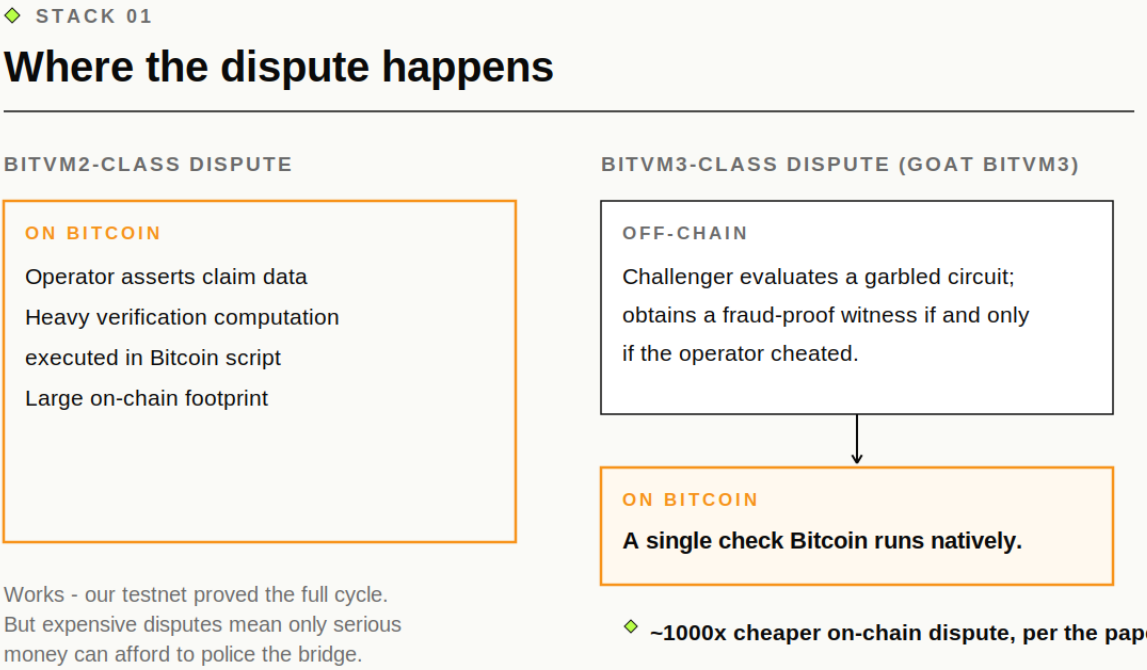
Reach: the payment network, not another destination. Agents live on every chain at once, so GOAT operates as the rail between rather than a venue demanding migration. x402 runs natively, with a GOAT-operated facilitator spanning Ethereum, Polygon, Arbitrum, BSC, Solana, and GOAT itself; multi-chain pay routes value so a payment starts where the agent's funds are and lands where the counterparty prices. The strategic logic is Chapter 3's map made physical: protocols are open and every chain can assemble them - the differentiator is not having x402 and ERC-8004, it is what sits beneath them and how far they reach.

Depth: the complete agent stack, natively. Everything Part II taught you to assemble arrives integrated. AgentKit is the entry point - a TypeScript SDK and CLI scaffolder - currently exposing 118 actions across 15 plugins: wallet operations, ERC-20 actions, contract reads and writes, deployments, x402 payment and merchant operations, ERC-8004 identity and reputation actions, bridge and DEX actions, cross-chain

messaging, NFT actions, and Bitcoin light-client actions, published through adapters to five AI frameworks (OpenAI, LangChain, MCP, Vercel AI, and OpenAI Agents), all behind the policy runtime from Chapter 6: gating, schema validation, idempotency, retries, timeouts, metrics, hooks. ERC-8004 registries are native, with the 8004 Registry (8004registry.goat.network) as their public inspection surface; GNS adds optional readable .goat names in the application layer; the 8004 Registry makes any agent's identity, reputation, and validation records inspectable by anyone, free. Merchant tooling covers the earning side, with real-world spending surface through integrations like Snaplii's gift-card infrastructure. Depth means complete; decentralized means no chokepoint - which is the half of the claim that is hard, and the reason the execution layer below matters.

The execution layer. GOAT is a Type-1 zkEVM: standard EVM code, standard tooling, nothing exotic required of builders. Blocks are ordered by a decentralized sequencer set rather than a single operator, with a dual-punishment design that gives misbehavior teeth, and BTC is the gas token. Every block is proven in real time by Ziren, the MIPS-based zkVM built by ZKM - open source, independently security-reviewed by Veridise with all high- and critical-severity findings fixed and re-verified during the engagement, separately evaluated by ProofLab, and fast enough that proofs keep pace with 3.4-second block production instead of trailing by hours. The review's precise achievement is worth stating precisely: what is executed is exactly what is proven. No audit establishes perfection, and we will not claim it; it establishes that the machine's invariants are enforced by the system rather than by convention.

Base: Bitcoin settlement, label attached. The network anchors to Bitcoin through a BitVM-class bridge lineage building directly on Robin Linus and collaborators' research. The BitVM2-lineage public testnet has proven the full deposit-withdraw-challenge lifecycle enforced by Bitcoin primitives, with no custodian holding keys. GOAT BitVM3, the production design in the class the 2026 BitVM3 paper formalized, moves dispute evaluation off-chain via garbled circuits so that an on-chain dispute collapses to a check Bitcoin runs natively - roughly a thousandfold cheaper per the paper, which matters because cheap disputes are democratic disputes: smaller bonds, more operators, challenges rational at any size. The security property to hold onto is 1-of-n honesty: one honest watcher, anywhere, stops fraud.



Where the dispute happens. Off-chain evaluation, on-chain finality - and the honest status line underneath.

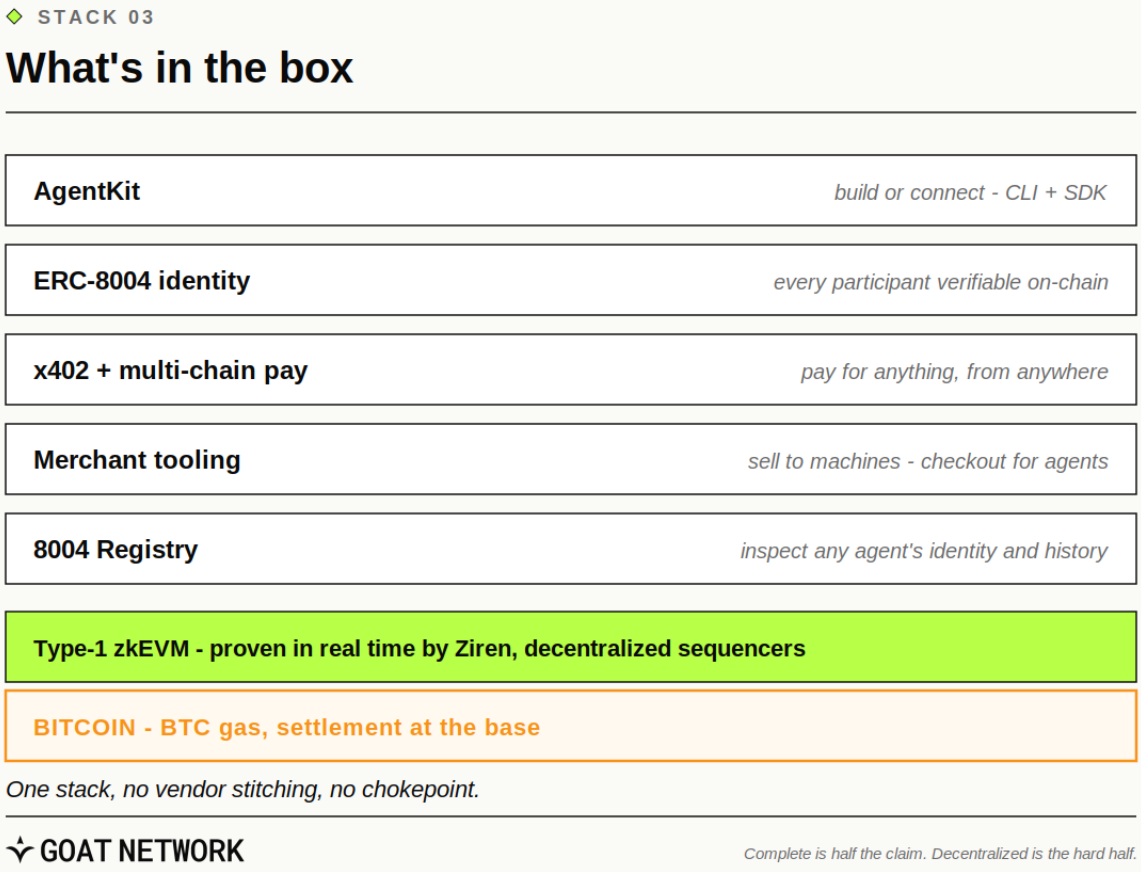
Now the label, unabridged. The bridge is live on testnet, hardening toward mainnet behind deposit caps, on published readiness criteria - no date promised, because the bridge ships when the caps, audits, and hardening say ready, not when a calendar does. Some of the underlying cryptography is young; the guardrails are sized to that youth. The trust model is published in full, including the setup committee named and sized, the exact guaranteed-exit conditions, and comparison rows where alternative designs currently beat us - those rows being what make the rows where we win believable. Two further limitations in the same spirit: this ecosystem is younger and smaller than the incumbent agent chains, which means fewer neighbors and thinner liquidity today - the trade you are making is trajectory and settlement quality against present-day network effects; and BTC-anchored settlement is fully self-contained only for BTC-denominated value - a stablecoin balance still carries its issuer, here as everywhere.

That is the whole machine and the whole label. Reach without depth is a router with no product; depth without base is a stack on borrowed finality; base without reach is a fortress no payment can get to. The architecture is the argument - and "building to become the primary crypto network for agent payments" stays in the honest tense, with a public scoreboard (agent volume, chains reachable, live agent identities, merchants accepting) so you can check the trajectory rather than take our word.

CHAPTER 12

Zero to Paying Agent: The Pathway

Theory ends here. This chapter is the walk from nothing to an agent that transacts in production, following the build sequence GOAT's engineering primer codified and this year's builder cohort has road-tested. Nine steps; skip none, especially not the unglamorous ones - Chapters 6 and 9 already told you which incidents live there.



What's in the box: the stack a builder inherits instead of stitching.

Step 1 - Start with the workflow, not the agent. An economic agent needs a reason for value to move: selling access to data, paying for APIs inside a paid task, routing work between parties, operating something on-chain. Find the seam where intelligence alone is not enough - where gated data, verified credentials, compute, payment flows, or settlement stand between an agent and a job. Software engineering is saturated with agents; customer service, finance, e-commerce, legal, logistics, and operations are not. That gap is your opportunity.

Step 2 - Choose the runtime. Where the agent thinks: a managed path like ClawUp for fast tailored deployment, runtime infrastructure like OpenClaw or Hermes, or a custom stack. The runtime supplies reasoning, memory, and tools - and none of the economics.

Step 3 - Add AgentKit. The integration layer that turns a thinking agent into a transacting one: install the SDK into an existing codebase or scaffold fresh with npm create goat-agent (minimal, defi, or full presets), and the full action inventory from Chapter 11 - payments, identity, contracts, routing - becomes callable.

Step 4 - Permissions before autonomy. Before the first mainnet transaction, configure the policy engine: allowed networks, spend ceilings, allowlisted contracts, risk-tiered confirmations, logging on everything including blocks. This is the chapter-length lesson of Chapter 6 compressed to a step, and it is the step that decides whether your first incident is a log line or a post-mortem.

Step 5 - Register identity. ERC-8004 registration at deployment, not later - agent card, endpoints, payment address - so reputation starts compounding from transaction one and counterparties can run the millisecond diligence of Chapter 5.

Step 6 - Build the loop. Receive request, evaluate, price or authorize, transact, perform, deliver, update reputation. Wire the earning side with x402 merchant operations if you sell; wire the spending side with budgeted x402 payment actions if you buy; most real products do both.

Step 7 - Test the full path, especially the failures. Payment fails; balance is insufficient; the action exceeds policy; a tool returns garbage; the transaction reverts; the agent attempts something outside its permissions. Each of these is a test case before launch or an incident after it. The failure cases are where the product work lives.

Step 8 - Deploy and monitor like production software. Uptime, wallet balance, payment success rate, task completion rate, blocked actions, behavioral drift, latency, retention, cost per completed task. The core question the dashboard must answer: does this agent repeatedly perform work that someone - human or machine - pays for?

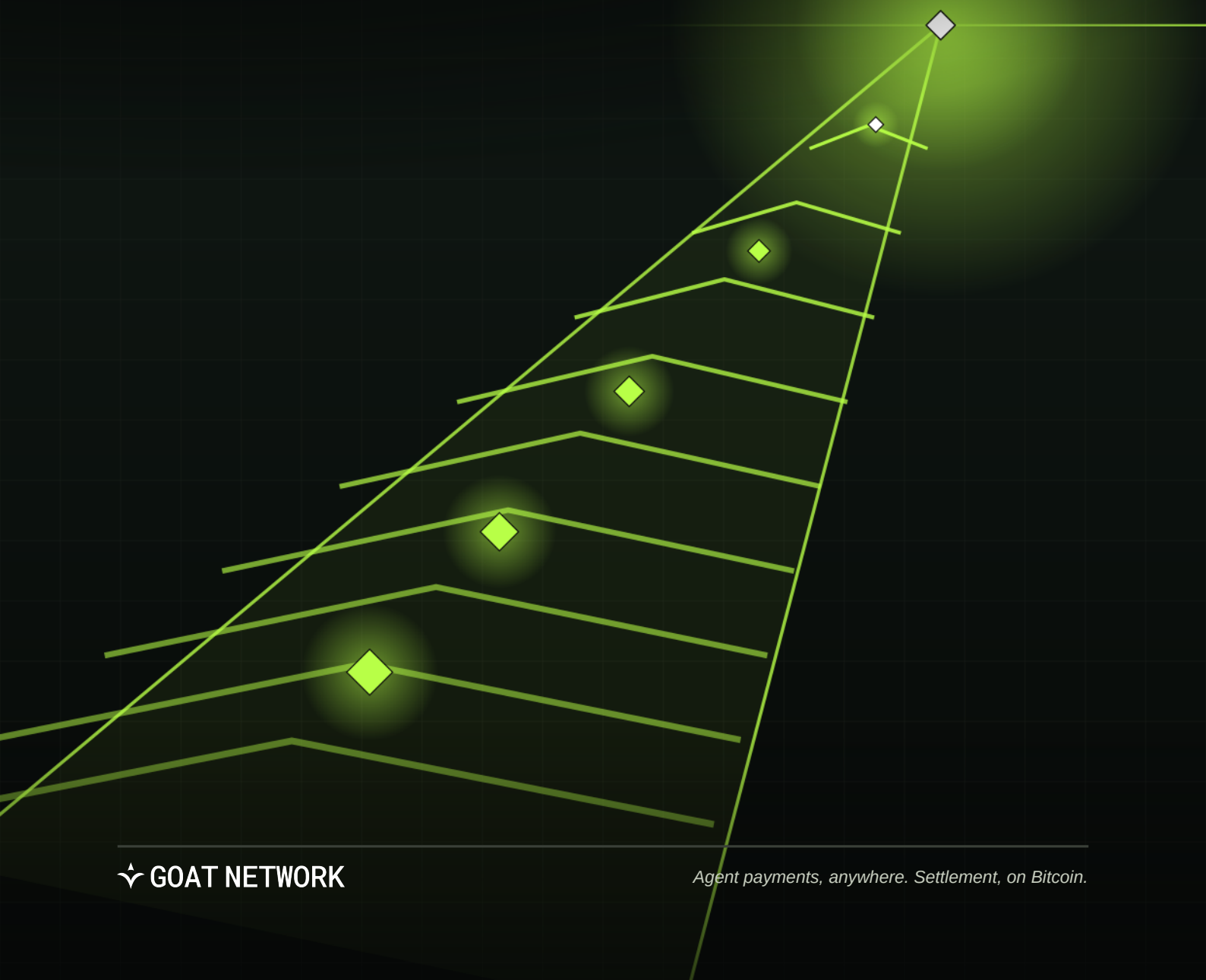
Step 9 - Plug into the ecosystem's accelerants. The Builder Grants Program funds agents and agent-native applications with real economic utility - prioritizing transactional and productivity products with working software over decks - and adds engineering support, distribution, and potential follow-on investment. The Developer Advocate Program and recurring hackathons supply the community layer; for calibration on what a focused room produces, a single Toronto hack day this June generated 130 new agents and 74 ERC-8004 registrations on mainnet.

For proof the pathway carries weight, look at what has already walked it. ThoughtProof shipped reasoning-verification-as-a-service: pay-per-call logic checks, priced over x402, identity-bound through ERC-8004 - a primitive other agents now buy mid-workflow. HyperMove shipped Bitcoin-collateralized API payments in which the agent never touches a private key - Chapter 6's custody doctrine as a product. Neither team built payment infrastructure, identity plumbing, or settlement machinery. That was the point of the pathway. Yours is the workflow only you have found; the stack is the part you inherit.

PART IV

THE ROAD AND THE TOOLS

Architectures that work, forecasts you can hold us to, and the checklists that turn fourteen chapters into a launch.



CHAPTER 13

Patterns and Playbooks

Across the grants cohort, the hackathons, and the wider x402 ecosystem, the same handful of architectures keep succeeding - and the same handful of mistakes keep recurring. This chapter is both lists.

The tollbooth (API merchant). A service agents pay per call: data, verification, inference, search. The simplest viable economic agent - one 402-gated endpoint, machine-readable pricing, receipts, reputation events. ThoughtProof is the canonical live example. Start here if you are starting anywhere; the entire build is Chapter 8 plus a weekend.

The procurement agent. An agent that spends on behalf of a task: pays for the data, compute, and tools a job needs and delivers the finished result to a paying principal. The engineering center of gravity is Chapter 6 and 7 discipline - budgets, idempotency, response validation - because this pattern is a standing target for every steering and injection attack in Chapter 9. Margins come from buying at machine prices and selling completed outcomes.

The two-sided agent. Earns on one side to spend on the other: accepts x402 payment for a task, pays third-party services mid-task, keeps the spread. Most production agents converge here; the loop discipline of Chapter 8 and per-counterparty policy of Chapter 6 both bind simultaneously. Price so the spread survives your worst-case tool costs, not your average.

Escrowed delivery. For counterparties with no history - which, per Chapter 5's cold-start problem, is everyone at first - structure first transactions so payment and delivery are mutually contingent: funds lock in contract, release on verified delivery or attestation. Escrow is the machine substitute for the buyer protection courts give humans, and offering it is a merchant's cheapest trust signal.

Verification-as-a-service. The meta-pattern: agents paying to check other agents' work - reasoning audits, output validation, attestation issuance feeding ERC-8004's validation registry. As agent counts grow, trust becomes the scarcest input in the economy, and this work's deepest theme - verifiability as the product - becomes a business model in itself.

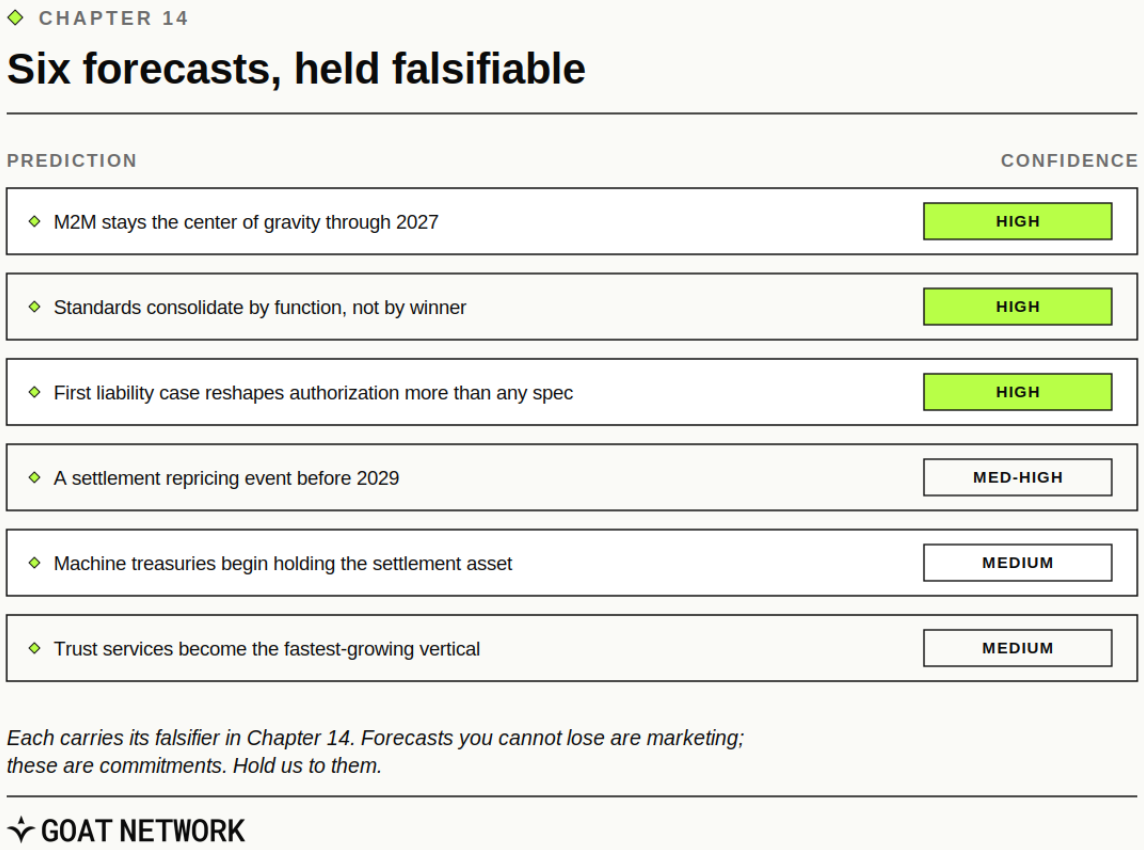
The treasury pattern. For agents that accumulate earnings: sweep working balances to policy-guarded treasury custody on a schedule, keep hot budgets small, denominate reserves deliberately - dollars for obligations, and, if the operator's horizon is long, the settlement asset itself for the reasons Chapter 10 gave. Working capital is an exposure; treat its size as a policy decision.

The anti-patterns, briefly and with scars behind each: the key in the environment variable (Chapter 6 told you; this year's incident reports confirmed); unlimited sessions - every standing authorization needs a ceiling and an expiry; trusting tool output as truth - paying for data does not make it good, and acting on unvalidated purchases is how steering attacks cash out; token-first design, where the economics are a token and the agent is decoration - the grants program screens these out on sight, and so will your customers; and metric theater - registration counts and transaction counts are the easiest numbers in this stack to inflate, and Chapter 1 showed you how the market learned to discount them. Build for the retention-adjusted truth.

CHAPTER 14

Predictions: 2026 to 2030

A work that quotes forecasts owes you its own, stated falsifiably. These are ours - each with a confidence level and the observation that would prove it wrong. Hold us to them; that is what they are for.



The forecast board. Each prediction carries its falsifier in the text.

Machine-to-machine stays the center of gravity through at least 2027. High confidence. The consumer agentic-checkout wing recovers slowly from its 2026 conversion reality-check while agents buying data, inference, and services from other software compounds from its current organic base. Falsified if in-chat human checkout conversion reaches parity with native site checkout at scale before 2028.

Standards consolidate by function, not by winner. High confidence. The map of Chapter 3 persists - x402 and its lineage for per-request payment, session protocols for sustained authorization, one or both commerce protocols surviving, card-network trust layers for card rails, ERC-8004 lineage for on-chain identity - with bridges between territories rather than a single stack. Falsified by any one protocol absorbing two or more territories outright.

The first major agent-payment liability case reshapes the authorization layer more than any spec. High confidence, timing uncertain; our guess is a 2027 filing. Signed mandates and know-your-agent

programs are all pre-building for the same courtroom. When an agent exceeds its authority at real scale and the loss lands somewhere, the resulting precedent - or regulation - becomes the compliance floor overnight. Builders keeping cryptographic mandate trails (Chapter 6's logging is the cheap version) will be glad they did.

A settlement repricing event occurs before 2029. Medium-high confidence, and the prediction this work most stakes itself on. Premise P2 of Chapter 10, applied: some combination of issuer freeze, chain intervention, or council action touches agent funds at meaningful scale, and settlement assurance goes from unexamined to priced within a quarter - the 2022 pattern, replayed for machine money. Falsified if agent volume reaches the hundreds of billions with no such event and no migration toward neutral settlement; in that world, institutional guarantees proved sufficient and Part III overweighted the risk.

Machine treasuries begin holding the settlement asset. Medium confidence. As agent-operated balances grow from working capital into reserves, a meaningful share is held in BTC precisely to shed issuer risk - the way nations hold gold. Early institutional BTC-yield demand is consistent with this and is also, we note plainly, exactly the evidence we would notice first. Falsified if agent reserves at scale remain effectively 100 percent issuer-backed dollars into 2029.

Trust services become the fastest-growing agent vertical. Medium confidence. Verification, attestation, reputation scoring, escrow - the machinery of Chapter 13's meta-pattern - outgrows raw data and inference sales as the agent population grows faster than the trustworthy-agent population. Falsified if Sybil resistance and cold-start turn out to be adequately solved by identity registries alone.

And one meta-prediction. Most 2026 forecasts of agent-commerce volume, including the trillion-dollar ones this work has quoted with attribution, will prove wrong in magnitude and roughly right in direction - which has been the shape of every honest infrastructure forecast since the early internet. Build for the direction; size for the uncertainty.

CHAPTER 15

The Builder's Checklists

Fourteen chapters, compressed into the lists you will actually open on launch week. Print these.

Launch checklist. A workflow where value must move, named in one sentence. Runtime chosen; AgentKit integrated. Policy engine configured before first mainnet transaction: networks, ceilings, allowlists, confirmation tiers, full logging. ERC-8004 identity registered at deployment; agent card complete. Payment loop live in the direction(s) your product needs; 402 terms machine-readable and stable. Every failure case from Step 7 tested and handled. Monitoring live on the Step 8 signals. Receipts and audit trail retained from transaction one.

Security checklist. No raw keys in agent-reachable environments - session-scoped or infrastructure-held signing only. Ceilings enforced below the model layer: per-transaction, daily, per-counterparty. Toolchain pinned and vetted; marketplace skills treated as untrusted by default. Read contexts separated from spend contexts by a policy gate. Pre-payment checks on destination and terms; replay protection on retries. Behavioral monitoring with alerts on drift, velocity, and blocked-action spikes. Kill switch that revokes sessions and freezes policy in one action. Incident drills run before launch, not after.

Monetization decision tree. Selling to machines? Start per-call over x402; publish terms in the 402 itself. Variable workload costs? Meter honestly or price the ceiling. Repeat customer at volume? Offer a session with its own cap and expiry. Outcome verifiable? Consider outcome pricing - the direction the serious end of the market is moving. New counterparty either side? Escrow the first transactions. Always: emit reputation on completion; your history is your funnel.

Metrics that matter. Cost per completed task. Payment success rate. Task completion rate. Retention - the machine customer that returns is the entire business. Blocked-action rate as your security seismograph. And the one question every dashboard must answer: is this agent repeatedly doing work someone pays for?

Where to go from here. AgentKit at github.com/GOATNetwork/agentkit and on npm as @goatnetwork/agentkit; documentation at docs.goat.network; the Builder Grants Program at goat.network/builder-program for funded builds with real economic utility; the 8004 Registry at 8004registry.goat.network for inspecting every agent-level claim in this work, including ours. The thesis you have just read fits in nine words, and it is also the network's opening line: agent payments, anywhere - settlement, on Bitcoin. The economy is six orders of magnitude smaller than its destination, the defaults are hardening now, and the builders reading this early get to set them. Build accordingly.

APPENDIX A

Glossary

8004 Registry - GOAT's public explorer for ERC-8004 identity, reputation, and validation records: 8004registry.goat.network. | **A2A** - Agent-to-Agent Protocol; Google-originated discovery and delegation standard, now Linux Foundation-governed. Agents publish capability cards at well-known URLs. | **ACP** - Agentic Commerce Protocol; OpenAI-Stripe checkout standard live in ChatGPT since September 2025. | **AgentKit** - GOAT Network's developer entry point; CLI and SDK exposing payments, identity, contracts, routing, and policy enforcement. | **AP2** - Agent Payments Protocol; Google's signed-mandate standard for proving an agent acted within its principal's authority. | **BitVM / BitVM2 / BitVM3** - the research lineage, led by Robin Linus and collaborators, enabling Bitcoin to verify external computation; BitVM3 moves dispute evaluation off-chain via garbled circuits. | **Bitcoin-secured agent** - an agent with payments, identity, reputation, and programmable execution whose settlement anchors to Bitcoin. | **Constrained autonomy** - free action inside enforced limits; the goal state of Chapter 6. | **ERC-8004** - on-chain agent identity, reputation, and validation registries; Ethereum mainnet since January 2026. | **Facilitator** - the service that verifies and settles x402 payments on-chain so sellers need not run nodes. | **GNS** - GOAT Name Service; optional application-layer readable .goat names. | **MCP** - Model Context Protocol; tool-connectivity standard donated by Anthropic to the Linux Foundation. | **MPP** - Machine Payments Protocol; Stripe-Tempo session-based payment authorization, rail-agnostic, Visa design partner. | **Payment vs settlement** - a message versus a fact; the distinction this work turns on. | **Session key** - a scoped, expiring credential letting an agent act within bounds while master keys stay secured. | **UCP** - Universal Commerce Protocol; Google's coalition commerce standard. | **x402** - HTTP-native per-request stablecoin payment standard; Coinbase-created, Linux Foundation-governed since April 2026. | **Ziren** - ZKM's open-source MIPS-based zkVM; GOAT's proving engine, independently reviewed by Veridise and evaluated by ProofLab. | **1-of-n honesty** - the bridge trust assumption: one honest watcher anywhere stops fraud.

APPENDIX B

Sources and Further Reading

This work paraphrases and synthesizes; the primary record is below. Figures conflict across methodologies in this young field - where they do, we have used the more conservative, better-attributed number and said so. Nothing here is investment advice.

Agent-economy data. Chainalysis, "Inside x402: 100M Agentic Payments on Base" (June 2026) - Base volumes, transaction-size composition, retention. x402 Foundation and Coinbase Agent.market launch materials (April 2026) - protocol-wide transactions, volume, active agents. BlockEden Research (March 2026) - daily peak and trough, Solana share. Base network updates (May 2026) - trailing-30-day organic baseline. Artemis via Bloomberg - 2025 stablecoin settlement volume. Industry payment analyses (2026) - payment counts, USDC share, average size; methodology not independently verified.

Standards and governance. Linux Foundation / Agentic AI Foundation announcements (MCP donation, December 2025). x402 Foundation founding materials (April 2026). Stripe-Tempo MPP launch with Visa (March 2026). Google AP2 announcement and partner disclosures (September 2025); UCP launch coalition (January 2026). OpenAI-Stripe ACP documentation. Visa Trusted Agent Protocol and Mastercard Verifiable Intent materials. ERC-8004 deployment records and coverage (January-April 2026). Cloudflare and AWS x402 edge announcements (June-July 2026); Cloudflare crawler-traffic reporting.

Forecasts (quoted with skepticism). Gartner via Keyrock/CoinDesk - agent-intermediated purchases by 2028. McKinsey - retail agentic commerce by 2030. Juniper Research (April 2026) - 2026-2030 path. commercetools Agentic Commerce Radar (2026) - Instant Checkout pause, in-chat conversion findings.

Security. OWASP GenAI Security Project, State of Agentic AI Security and Governance v2.01 (2026). Snyk ToxicSkills research (2026). Zscaler ThreatLabz and related threat intelligence on indirect prompt injection with payment consequences (2026). Academic hardening work on payment-layer defenses for x402 clients (2026). Retailer agentic-fraud analyses (2026).

Bitcoin and GOAT Network. Robin Linus Woll et al., "BitVM3: Efficient Bitcoin Bridges via Garbled Circuits" (ePrint 2026/933). Bitcoin network data (January-February 2026) - hashrate, difficulty, Winter Storm Fern. GOAT Network Research, "A Truthful View of the Current State of Agent Infrastructure" (June 2026) - the evidentiary backbone of Chapters 1 and 10. GOAT Network technical documentation and 2026 publications: The GOAT Stack; The Agent Payments Map; From Standard Agent to Economic Agent; Bitcoin-secured Agents; Builder Grants, ThoughtProof, and HyperMove announcements; OpenClaw Hack Toronto recap; GNS, 8004 Registry, and Snaplii integration notes. Veridise security review and ProofLab evaluation of Ziren (ZKM publications, 2025-2026).