

# Sardis

## Payment infrastructure for AI agents

Infrastructure that lets AI agents make real payments without anyone trusting them with money.

# AI agents can reason. They cannot be trusted with money.

## What enterprises want

- Agents that buy API credits autonomously
- Agents that pay vendors without human approval for every \$10
- Agents that manage cloud infrastructure budgets
- Full audit trail for compliance

## What stops them

- Give agent a credit card = no spending limits, no merchant control
- Give agent a wallet = can drain entire balance
- Prompt injection can override "soft" guardrails
- No audit trail that satisfies compliance teams

The fear: an AI agent hallucinates a \$50,000 cloud bill. No reversal. No audit trail. No one to call.

**This is already happening.** Without a policy layer, a single prompt injection triggers unlimited spend. No audit trail. No reversal. No compliance defense.

### TODAY

#### Agent uses API key

Prompt injection

Pays \$5,000

No limits. No reversal. No audit.



### WITH SARDIS

#### Agent requests payment object

Policy: max \$50, openai.com only

Blocked. Logged. Safe.

\$0 exposure.

Replace reusable credentials with one-time payment objects.

# The future of payments is not a new rail. It is a new authorization primitive.

## Old world: Reusable credentials

- Credit cards: reusable PAN, unlimited scope
- API keys: no per-call bounds, no merchant control
- Wallets: full balance access, drain risk



## New world: One-Time Payment Objects

- Single-use, merchant-bound, signed tokens
- Bounded by amount, merchant, time, and policy
- Settle over any rail: stablecoin, card, ACH, Tempo

No credential is ever reused. Each payment mints a fresh, cryptographically signed object. Processors and wallets move the money. Sardis decides whether a payment is allowed before it reaches a rail. **We fit above processors, wallets, and agent-payment protocols rather than replacing them.**

Every new protocol that launches — x402, MPP, ACP, TAP — increases the need for one consistent policy layer. More rails means more complexity for the customer, which is exactly the problem we solve.

Protocol depth: UTXO cells, 22-state lifecycle, ZKP privacy, mandate trees (see appendix)

# AI agents will become a meaningful class of economic actors.

Payment rails were built for human-initiated transactions, not autonomous software with bounded permissions. The missing product in that stack is the control layer that decides which machine payments are allowed.

Agent-mediated B2B spend (2028)

**\$15T+**

Gartner, Nov 2025

AI agent infra (2030)

**\$50B+**

46% CAGR

## THE INCUMBENTS SEE IT TOO

Coinbase launched **x402** in May 2025

Mastercard launched **Agent Pay** in April 2025

Google launched **AP2** in September 2025

Visa launched **TAP** in October 2025

Stripe machine payments remain in **private preview**

**The timing is real, not hypothetical.** Large payment and platform companies all moved into agent payments during 2025. They are building intent, identity, and settlement surfaces. Sardis sits above those surfaces as the control layer. **Fragmentation helps us:** the more protocols that exist, the more enterprises need one agnostic policy layer.

**Enterprises will still need deterministic controls.** Approvals, audit logs, and compliance checks do not disappear just because the rails get better.

# The missing layer between intent, identity, and settlement.

THREE LAYERS, THREE OWNERS

## Intent

AP2-style flows describe what the agent is trying to buy

Google AP2 consortium standard

## Identity

Identity layers verify which agent is acting

TAP attestation protocol

Gap: Authorization, escrow, lifecycle

## Settlement

Processors and crypto rails actually move the money

Stripe MPP, Coinbase x402, MC Agent Pay



## Sardis

The operating layer that connects intent, identity, and settlement

Policy engine

Approvals + orchestration

Lifecycle management

Multi-rail settlement

Privacy + compliance

AP2-like flows and identity layers handle intent and verification. Processors and crypto rails handle settlement. **Sardis is the authorization and lifecycle layer that connects them** — the missing piece that decides whether a payment is allowed to happen before a single cent moves.

# Open protocol + production platform. Deterministic control over agent money.

Developers integrate in 2 API calls. Fast execution. No custody. Deterministic enforcement.

## Policy Layer

Spending Mandates

Policy Engine

Rate Limits

Merchant Allowlists

Fail-Closed

## Settlement

MPC Wallets

4 Chains (Base + Tempo)

Virtual Cards

MPP (Tempo)

Fiat Rails

## Audit Trail

Merkle Audit Trail

KYC / AML

Sanctions Screening

Evidence Export

### INTEGRATION

```
from sardis import SardisClient
client = SardisClient(api_key="sk_...")
wallet = client.wallets.create(policy="Max $100/day")
tx = wallet.pay(to="openai.com", amount="25.00")
```

4

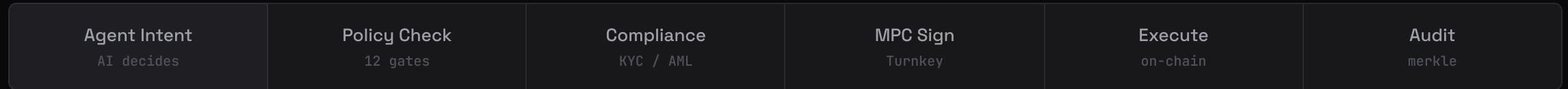
lines to safe payment

### WHAT SHIPS UNDERNEATH

Spending mandates are the entry point.  
Underneath: merchant checkout, virtual cards,  
cross-chain settlement, escrow, compliance  
automation, and 15 framework integrations.

**Every capability ships with the same  
cryptographic enforcement.**

# Every payment passes the mandate. No exceptions.



## ALLOWED

Agent: "Pay \$10 to openai.com for API credits"  
Mandate: merchant=openai.com, max=\$50/tx  
✓ Policy passed → signed → executed

## BLOCKED

Agent: "Pay \$200 to gambling.com"  
Mandate: merchant not in allowlist, amount > \$50  
✗ Policy violation → blocked → logged

```
● ● ● sardis policy-engine
> agent requests: $50,000 to unknown-vendor.xyz
  └─ Gate 3: FAIL - merchant not in allowlist
  └─ Gate 5: FAIL - exceeds daily limit ($500)
  └─ DENIED in 12ms. Zero funds moved. Audit hash: 0x7a3f...c891
```

We don't hold money. We enforce rules. If an agent violates a mandate, the transaction fails cryptographically before it executes. Nothing to reverse because nothing moved.



# I built the core platform solo while the category was still forming.

Built the API, mandates engine, wallets, smart contracts, SDKs, and integrations alone. The moat is not one feature; it is integrated execution speed across the entire stack.

**47+**

API endpoints

**50+**

Database tables

**34**

Published packages

**12+**

Smart contracts

**4**

chains today

## Protocol Depth

- AP2 — Google / PayPal / Visa / Mastercard consortium standard
- TAP — Ed25519 + ECDSA agent identity attestation
- ERC-8183 — on-chain jobs and reputation
- MPP — Stripe machine payments surface

## Capabilities Already Built

- **Policy from plain English** — human rules become deterministic machine enforcement.
- **One control plane across rails** — the same policy model governs stablecoins, cards, and agent-payment protocols.
- **Merkle audit proofs** — tamper-evident history without trusting Sardis.

**Security audits are budgeted.** We are the policy engine, not the custodian. Settlement providers handle money movement; Sardis handles deterministic enforcement.

# Developers are adopting. Distribution is ahead of revenue.

15K+

organic installs

15

integrations shipped

2

live distribution surfaces

LIVE

Activepieces

MARKETPLACE

MCP Server

PUBLISHED

INTEGRATING

AutoGPT

IN TALKS

CrewAI

PR SUBMITTED

Composio

TOOL SUBMITTED

OpenClaw

Browser Use

Vercel AI SDK

OpenAI Agents

Stagehand

ECOSYSTEM ACCESS

Stripe (MPP early access)

Coinbase (x402 support)

Base (primary chain)

DESIGN PARTNERS

AutoGPT — integration package shipped and active discussion on distribution

DEVELOPER PLATFORM REACH

15 integrations put Sardis in front of a large developer surface area. Activepieces is already live; the rest create top-of-funnel distribution before monetization.

**No paying production customers yet.** What we do have is strong developer pull: 15K+ organic installs, zero paid acquisition, and early evidence that integrations can become a pipeline for design partners.

# Usage-based pricing. Grows with customer volume.

TIER	TARGET	PRICE	INCLUDES
Free	Sandbox exploration	\$0	10 tx/day, testnet only
Developer	Builders shipping to mainnet	\$29/mo + 0.1%	Unlimited agents, mainnet, audit logs
Growth	Startups scaling	\$199/mo + 0.5%	SSO, compliance exports, priority support
Enterprise	Large orgs	\$2K-\$10K/mo + vol. disc.	99.99% SLA, dedicated CSM, custom policies

**Pricing reality:** Sardis monetizes on top of settlement rail costs. Customers are paying for control, approvals, and auditability, not just raw payment throughput.

MULTIPLE REVENUE SURFACES

Policy engine subscription	\$29-\$499/mo
Transaction fees	0.1-0.5% per payment
Compliance-as-a-service	\$1K-\$10K/mo enterprise
Framework integrations	free → paid conversion

Each customer expands from mandates into checkout, cards, escrow, compliance. LTV grows with every surface adopted.

YEAR 1 SUCCESS CRITERIA

**DEVELOPER WEDGE**  
Hundreds of developers using Sardis in sandbox or staging through integrations and the free tier.

**PAID STARTUPS**  
The first 10-15 AI-native startups paying for governed production spend and approvals.

**ENTERPRISE MOTION**  
2-3 pilots underway where approvals, audit exports, and finance controls matter enough to justify a broader rollout.

# Developer wedge first. Enterprise expansion second.

## PHASE 1 · MONTHS 1–3

### Developer Adoption

- Framework integrations across core agent surfaces
- MCP server for Claude / Cursor
- Framework-native integrations

## PHASE 2 · MONTHS 4–6

### Design Partners & First Paid Conversions

- High-intent design partner motion
- Selected ecosystem co-marketing
- First paid AI-native startups onboarded

## PHASE 3 · MONTHS 7–12

### The Prerequisite for Enterprise AI Deployment

Enterprises need spend controls before they let agents touch meaningful budgets.

**We are not selling a generic payment tool; we are giving AI teams a path from sandbox experimentation to governed production spend.**

Enterprise pilots follow once agent teams need governed spend in real workflows, not just sandbox demos.

## LAND AND EXPAND

Developers bring Sardis in to solve agent spend safely. Production deployment then pulls in finance, security, and audit stakeholders who need approvals and evidence before real money moves.

# The default alternative isn't a better startup. It's internal glue code.

## THE STATUS QUO

### DIY Soft Guardrails

Startups bound API keys with prompt instructions. A single prompt injection bypasses the wrapper. Rate limits are advisory, not enforced. In-house wrappers are prompt injection bait.

## SARDIS APPROACH

### Deterministic Enforcement

Payment rails move the money. Sardis sits in the policy path. Every transaction hits deterministic policy gates before a cent moves. Fail-closed. No prompt can override a cryptographic gate.

## THE ALTERNATIVE

### Blind Trust

Hand a \$100K wallet to a reasoning engine and hope it follows instructions. One hallucination and the entire balance is gone. No audit trail. No reversal.

## SARDIS APPROACH

### Non-Custodial Enforcement

We do not hold money. We enforce rules. If an agent violates a mandate, the transaction fails cryptographically before it executes. Nothing to reverse because nothing moved.

### The Corporate Card Illusion

Startups give agents \$500-limit virtual cards. But cards settle post-transaction. They cannot read prompt context, enforce allow-listed endpoints, or fail cryptographically at the intent layer. **Corporate cards manage employees. Sardis manages code.**

# Solo founder with technical depth. Hiring the commercial layer around it.

## Efe Baran Durmaz

20. Built the initial platform solo: 47+ endpoints, 34 published packages, 12+ contracts, and 15 framework integrations. Background includes Nokia AI automation, a Bilkent full merit scholarship, and a national exam rank of 1,405 out of 3.5M. I spent the last year learning the agent-payment stack by building inside it.

**Top 0.04%** National exam, 1,405 / 3,527,443

**Nokia** AI Automation Engineer

**Bilkent** Full merit scholarship, 3.53 GPA

**Built solo** API, policy engine, contracts, SDKs, and integrations

Prior builds across Rust, Python, TypeScript, Solidity, and C++, including shipped developer tools and a Steam game.

## Hiring: Founding Core

The next constraint is not product depth. It is distribution, design-partner conversion, and enterprise trust. That is why the first senior hires are a forward deployed engineer and a sales engineer who can turn developer adoption into paid deployments.

### Forward Deployed Engineer HIRE #1

Technical evangelist who converts SDK installs into design partners and paid deployments.

### Sales Engineer

Enterprise sales motion. Converts design partners into paid contracts. Works the pipeline while I ship the protocol.

### Core Protocol Engineers (2-3)

Cryptography, distributed systems, chain execution. Build the moat deeper.

SARDIS LABS, INC., DELAWARE C-CORP (STRIPE ATLAS) · 10% EQUITY POOL FOR FIRST 10 HIRES

# Key risks, and how we plan to de-risk them.

- 1 Category timing risk**  
If agent-mediated spend remains experimental, adoption slows. Our hedge is a developer wedge that works even before large enterprise budgets move.
- 2 Governance ownership risk**  
Processors or model companies may try to absorb this layer. Our view is that buyers will want one independent control plane across multiple rails and protocols.
- 3 Founder and GTM risk**  
I built the stack first; now the bottleneck is conversion and trust. The first hires are a forward deployed engineer and sales engineer to close that gap.
- 4 Security and compliance risk**  
This market demands audits, approvals, and disciplined launch sequencing. We treat those as core product work, not cleanup after launch.
- 5 Deployment sequencing risk**  
The first phase is earning the right to sit in production workflows. Revenue should follow that trust, not precede it.

# Raising a \$2M seed to convert developer pull into paid deployments and enterprise readiness.

\$2M provides 24 months of runway. Disciplined burn: remote-first team, founder in low-cost geography, capital focused on trust and distribution.

## Use of Funds

Team (eng + GTM, remote-first)	40%
Compliance + Security Audits	10%
Infra + GTM (events, tools)	10%
Runway (24-month buffer)	40%

## What Success Looks Like

The first hires are GTM-heavy because product depth is ahead of commercial depth. The capital is for trust, distribution, and disciplined expansion.

MONTH 1  
First paid customer live. Mainnet transactions flowing.

MONTH 3  
5+ paying customers. GTM hire in seat. SOC2 started.

MONTH 6  
\$10K MRR. Design partner pilots converting. Card issuing program active.

MONTH 12  
\$50-75K MRR. Enterprise pilots. TPV \$5M+/month. Series A ready.

This is a control-layer bet: win the right to govern production agent spend, then monetize the volume that follows.



# Sardis

Every AI agent becomes an economic actor.

Sardis is the control layer that makes that safe.

*The thesis is simple: agents will need bounded payment permissions before enterprises trust them with money. Sardis is building that control layer.*

WEBSITE

[sardis.sh](https://sardis.sh)

EMAIL

[efe@sardis.sh](mailto:efe@sardis.sh)

GITHUB

[EfeDurmaz16](https://github.com/EfeDurmaz16)