pact_

The Accountability and Evidence Layer for Autonomous AI Agents

What is pact_?

Pact is the trust infrastructure for AI agents that negotiate, spend money, and make commitments autonomously. It turns agent interactions into cryptographic evidence that is provable, replayable, and auditable in order for companies to safely deploy agents in real-world commerce. If an agent can act, Pact makes that action accountable.

The Problem

AI agents are starting to buy APIs and data, negotiate prices and SLAs, trigger payments and act across company boundaries, even buying company data proprietarily.

But today, logs are not evidence, payments don't prove intent, failures aren't attributable and legal, compliance, and insurance teams have no visibility into these payments. This makes autonomous agents uninsurable, unauditable, and unsafe at scale.

The hard problem isn't payments. What I'm trying to tackle at pact_ is proving who agreed to what, under which constraints, and who's responsible when something breaks. What's missing is a primitive for negotiation, policy enforcement, deterministic outcomes, and evidence-grade accountability.

My Solution

Pact is a deterministic negotiation and evidence system for agents. When agents interact through Pact, every negotiation round is signed and hash-linked and policies are enforced in-line. We make settlements binary with either commit or abort results and failures classified deterministically. All transactions generate an evidence bundle automatically and responsibility is assigned via Default Blame Logic automatically as well.

Think:
Stripe for payments
Git for code history
pact_ for agent intent and accountability

Current System Core Capabilities

- *Proof-of-Negotiation (PoN)* provide cryptographically signed and replayable transcripts
- *Policy-as-Code (PaC)* enforce price, SLA, reputation, and compliance constraints
- *Atomic Settlement Boundary* makes sure there are no partial or ambiguous outcomes
- *Evidence Bundles* are portable artifacts for audit, disputes, and insurance for every transaction

- *Default Blame Logic* is pact_'s deterministic responsibility assignment
- *Verifier-Only Tooling* allows auditors to not need SDKs or wallets
- *Pact Passport* create a verifiable trust derived from real outcomes of these agents and not promises

How Developers Use It

Buyer agents wrap high-risk actions (spend, negotiate) inside Pact. Then, provider agents add a thin Pact adapter in front of their service. Enterprises and auditors verify outcomes post-hoc from artifacts alone.

Adoption will be incremental because Pact is used only where provability matters, but I see this market being central to transactions in a few years.

Business Model

- Free SDK for adoption
- Microscopic coordination fee when Pact-signed transcripts trigger settlement
- Paid APIs for reputation, creditworthiness, and trust scoring
- Enterprise products for audit, compliance, disputes, and insurance workflows

High-frequency agent interactions × tiny fees = massive scale.

Why I think this Scales

There is no custody of funds, no marketplace and most importantly, no competition with payment rails. Pact sits above all agents, rails, and ecosystems.

Other companies like Radius and Natural focus on agent execution, task orchestration, tool calling, workflow automation, making agents more capable and autonomous, etc. They answer the question of "How does the agent accomplish this task?"

Whereas, Pact answers what exactly was negotiated, who signed which commitments, what constraints were enforced, what failed and whose fault was it, and can a third party verify this without trusting anyone. We produce the necessary cryptographic negotiation transcripts, deterministic replay, evidence bundles, blame judgments, and reputation consequences for that agent's provider.

That's not execution — that's institutional trust. Once one counterparty requires Pact transcripts, everyone else must comply.

Use Case Example: Autonomous Art Authentication Agent

Scenario 1:

This scenario is specifically bizarre for a reason to show that pact_ has broad applications. A museum deploys an AI agent tasked with verifying whether a newly surfaced sculpture is an

authentic, previously undocumented work by Jeff Koons before committing to a $4.2M acquisition. This is not a single API call but it's a multi-party, high-risk, reputation-sensitive task.

What the Agent Must Do Autonomously

1. Purchases high-resolution scans from a private imaging provider
2. Commissions a materials analysis from a lab agent
3. Requests stylistic verification from two independent expert-model providers
4. Negotiates SLAs and pricing for each service
5. Aggregates results and produces a recommendation

Each provider:

1. Charges per analysis
2. Has different confidence levels
3. Has reputational risk if wrong

Why Pact Is Required

Without Pact	***With Pact***
Agreements are informal	Every negotiation is signed and recorded
SLAs are unenforceable	Every SLA is enforced by policy
If analysis is wrong, no one can prove responsibility	Every failure has deterministic blame assignment
The museum bears all risk	Every decision is backed by evidence

Pact Flow

1. Intent Created

   "Verify authenticity of sculpture X.
   Budget ≤ $25,000.
   Confidence threshold ≥ 92%.
   All providers must have reputation ≥ A."

2. Negotiation (PoN)
   a. Imaging agent quotes $4,000
   b. Materials lab quotes $8,500
   c. Style model quotes $6,000
   d. SLAs and confidence guarantees negotiated
   e. All terms signed and hash-linked
3. Policy Enforcement
   a. Pact enforces:
      i. Budget limits

   ii. Reputation thresholds
   iii. Required redundancy (≥2 independent opinions)
b. Execution and Settlement
   i. Providers deliver analyses
   ii. Settlement executes atomically
   iii. Late or failed providers are automatically flagged
c. Evidence Bundle Generated
   i. Pact produces a portable evidence bundle containing:
      1. All signed negotiations
      2. Provider guarantees
      3. Execution results
      4. Integrity proofs
      5. Confidence aggregation logic
d. Decision and Accountability
   i. Agent recommends “Acquire” or “Reject”
   ii. If later challenged, the museum can prove:
      1. What the agent knew
      2. What was promised
      3. Who was responsible for errors

If Something Goes Wrong

If the sculpture is later proven fake:

- Pact deterministically identifies:
  - Which provider failed
  - Which guarantee was violated
    Whether the agent acted within policy
- Evidence can be used for:
  - Refunds
  - Legal claims
  - Insurance recovery
  - Reputation penalties

No ambiguity. No finger-pointing.

Scenario 2:

A core AI agent is tasked with training a small but critical model update overnight.
Instead of trusting a single cloud provider, it shards the job across multiple untrusted compute agents to reduce cost and avoid vendor lock-in. Each compute agent runs arbitrary hardware, charges per millisecond, claims correctness and has asymmetric information.

What the Agent Must Coordinate

The orchestrator agent must:

1. Negotiate price per FLOP
2. Enforce max latency and determinism constraints

3. Require verifiable execution proofs
4. Handle partial completion and retries
5. Avoid paying for incorrect or duplicated work

Why Existing Systems Break

Without Pact:

1. Payments can happen (x402 / crypto)
2. Logs exist
3. Results are returned

But:

1. You cannot prove what was agreed
2. You cannot prove which shard failed
3. You cannot deterministically assign fault
4. You cannot replay the execution path
5. Disputes require humans and guesswork

At scale, this is unworkable.

Pact-Backed Flow (Technical)

Intent Declaration

The agent issues a signed intent:

```
{
  "task": "model_training_shard",
  "hash": "abc123…",
  "budget_per_shard": "≤ $0.002",
  "latency_ms": "≤ 500",
  "determinism": "required",
  "proof": ["execution_trace", "output_hash"],
  "reputation_min": 0.92
}
```

This intent becomes the root hash of the Pact transcript.

Negotiation (Deterministic Rounds)

Each compute agent responds with:

1. price
2. hardware profile
3. expected latency
4. proof type supported

All bids and counters are:

1. signed
2. hash-linked
3. ordered

Policy Enforcement (Pre-Execution)
Before any compute runs, Pact enforces:

- budget caps
- determinism requirements
- provider reputation thresholds
- redundancy constraints (e.g. 2-of-3 verification)

Violations abort before money or compute is wasted.

Settlement Authorization (Not Execution)
Pact authorizes settlement conditionally:

> "Payment is valid if output hash matches intent-derived expectation."
> **This is critical: settlement is gated by evidence, not trust.**

Execution and Evidence Emission
Compute agents execute and return:

- output shard
- execution trace hash
- timing proofs

Each submission is attached to the transcript as evidence.

Deterministic Outcome Resolution
Pact:

1. Verifies hashes
2. Checks SLA timing
3. Matches execution to negotiated terms

Possible outcomes:

- ✅ Valid → settlement commits
- ❌ Invalid output → abort + blame
- ⏱ Timeout → classified failure

Default Blame Logic (DBL)
If a shard is wrong or late:

1. Pact identifies the last valid signed commitment
2. Assigns fault deterministically
3. Emits a signed judgment artifact

This artifact can be:

- Used for refunds
- Fed into reputation
- Consumed by insurers

- Audited later without trusting the system operator

Vision
If your agent can spend money, make commitments, or act autonomously, it must run inside a Pact boundary.

Pact is the accountability layer that makes autonomous commerce possible.