



AI Agents Will Transact

But Today's Payments Weren't Built for Them

Sivasubramanian Ramanathan

Product Owner | Fintech & Innovation

The Problem: Checkout Was Designed for Humans

Agents don't have fingers to click "Buy Now".

What Developers Face Today

- No standard way for agents to pay
- Multiple competing protocols (UCP, AP2, x402, ACP)
- No testing tools — only real implementations
- Security concerns unclear

What I Built

AgentPayment Sandbox (APS)

- Mock servers for 4 protocols
- Inspector for compliance testing
- Schema validators
- Works offline, no real money

The problem: Developers building AI shopping agents have no way to test payment flows without implementing real protocol servers.

The Emerging Protocols

Google, Coinbase, and OpenAI are defining how agents will pay.

| Protocol | What It Does | Key Innovation |
|----------|--------------------|--------------------------------------|
| UCP | Universal checkout | Shopify, Walmart, Stripe unified API |
| AP2 | Agent Payments | Verifiable Credentials (Mandates) |
| A2A | Agent messaging | Agents talk to agents |
| x402 | Micropayments | HTTP 402 + crypto signatures |
| ACP | E-commerce | OpenAI + Shopify checkout |
| MCP | Tool access | Claude/ChatGPT to APIs |

AP2 is built ON TOP of A2A. x402 extends AP2 for crypto.



Why I'm Exploring This 🔍

My Curiosity

Agentic commerce sits at the intersection of **fintech**, **AI**, and **policy** — areas I find fascinating.

Questions That Drew Me In

1. **Infrastructure Gap**: How will payments evolve for agents?
2. **Trust**: Who's liable when an agent buys wrong?
3. **Security**: Can attackers hijack agent purchases?

My Approach

I build things to understand them. APS is how I learn.

The Trust Crisis

Payments assume a human clicked "Buy". Agents break that.

Old World Problems

- CAPTCHAs block agents
- Card forms need human input
- No proof of agent authority
- Who's liable for wrong purchases?

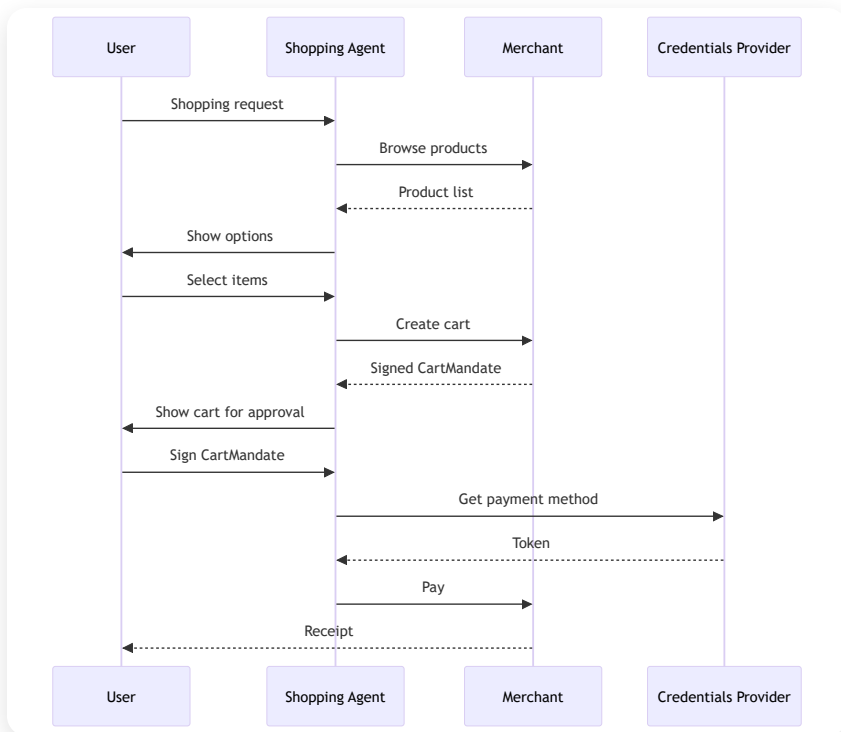
New World Solutions (AP2)

- **Verifiable Credentials** (VCs)
- **Intent Mandates** (what user wants)
- **Cart Mandates** (what merchant agreed)
- **Payment Mandates** (audit trail)

AP2's core innovation: Cryptographic proof of who authorized what.

Life of a Transaction: Human Present

User is available to approve the final purchase.

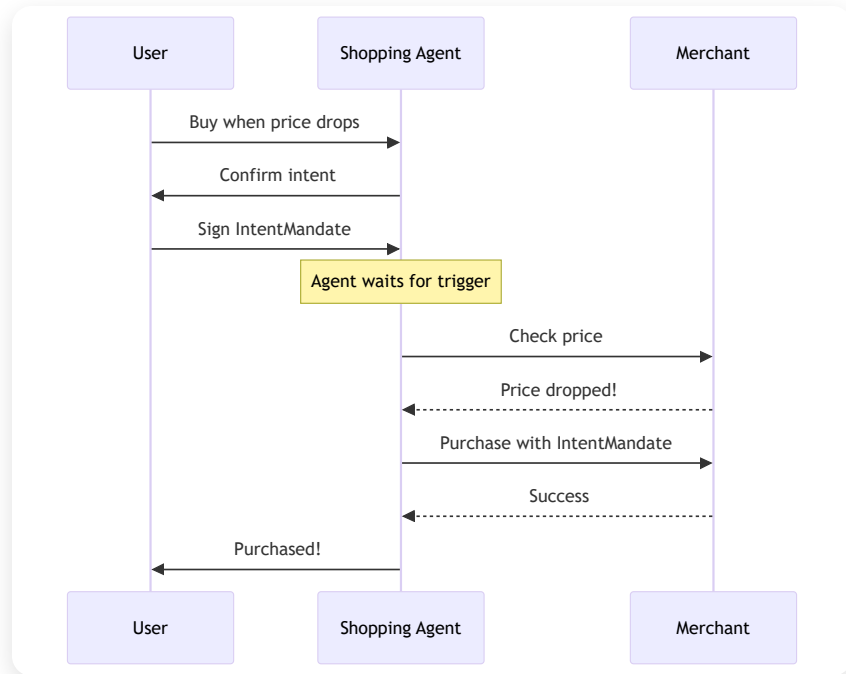


The Flow

1. User → Shopping Agent: "Buy shoes"
2. Agent → Merchant: Browse products
3. Merchant → Agent: Signed CartMandate
4. Agent → User: Show cart for approval
5. User signs CartMandate
6. Agent → Credentials Provider: Get token
7. Pay → Receipt

Life of a Transaction: Human NOT Present

"Buy these shoes when price drops below \$100"



The Flow

1. User → Agent: "Buy when price drops"
2. User signs **Intent Mandate** (budget, constraints)
3. Agent waits for trigger
4. Price drops → Agent purchases
5. Merchant may force user confirmation
6. Receipt sent to user

Key: Intent Mandate limits agent scope.

Security: What Could Go Wrong?

The protocol anticipates adversarial scenarios.

| Threat | Description | AP2 Mitigation |
|---------------------|-----------------------------------|--------------------------------------|
| Prompt Injection | Attacker tricks agent into buying | Intent Mandate limits scope |
| Agent Hallucination | Agent misunderstands request | Cart Mandate requires user sign-off |
| First-Party Misuse | User claims fraud for refund | Signed mandate is evidence |
| Account Takeover | Fraudster uses victim's agent | Device-backed key attestation |
| Man-in-the-Middle | Attacker alters transaction | Cryptographic signature verification |

Dispute Resolution: Mandates provide non-repudiable audit trail.

The Mandate System

Verifiable Credentials are the trust anchors.

1. Intent Mandate

- User's **shopping intent** in natural language
- Budget constraints
- **Signed by user's device key**
- Has expiration time (TTL)

2. Cart Mandate

- Final SKUs, price, shipping
- **Signed by merchant first**
- Then **signed by user**
- Binding contract

3. Payment Mandate

- Shared with network/issuer
- Contains: AI agent presence signals
- Enables: Risk assessment
- Evidence for disputes

Key Property

Non-repudiable: Can't deny you signed it.

How AP2, A2A, MCP Relate

Three layers of agent infrastructure.

| Layer | Protocol | Purpose |
|-------------|----------|-----------------------------|
| Data Access | MCP | Agent ↔ Tools/APIs |
| Agent Comms | A2A | Agent ↔ Agent messaging |
| Payments | AP2 | Agent ↔ Payments (mandates) |

In short:

- MCP: Agents talk to **data**
- A2A: Agents talk to **agents**
- AP2: Agents talk about **payments**

How AP2 and x402 Relate ⚡

AP2 is payment-method agnostic. x402 is crypto payments.

AP2 (Google)

- Supports "pull" payments (cards)
- Roadmap: "push" payments (bank, crypto)
- **Payment agnostic framework**
- Partners: Visa, Mastercard, Adyen

x402 (Coinbase)

- **HTTP 402 "Payment Required"**
- EIP-712 signatures
- Stablecoins (USDC on Base)
- Metered API access

Together: AP2 provides the trust framework, x402 provides the crypto rails.

What APS Tests

I built a sandbox to learn by doing.

Mock Servers (2,700+ lines)

| File | Lines | Tests |
|---------|-------|-----------------------|
| ucp.py | 475 | Discovery, checkout |
| ap2.py | 727 | Mandates, OTP |
| x402.py | 524 | 402, verify, settle |
| acp.py | 340 | Sessions, fulfillment |

Inspector

- Runs test suites against YOUR server
- Checks: endpoints, status codes, fields
- Returns: **Security Score** (0-100)
- Recommendations for compliance

Schema Validators

- Pydantic validators for x402
- CAIP-2 network validation
- EIP-3009 authorization format

Why a PM Built This

"You wrote 2,700+ lines. Aren't you a PM?"

My Philosophy

1. **Build to Understand**
2. Infrastructure needs PMs who get tech
3. De-risk by prototyping

What This Shows

- I can read protocol specs
- I can implement working software
- I can document thoroughly (8 docs, 3 ADRs)





Live Demo

GitHub Pages Challenge

No server = No mock endpoints

Solution

```
const IS_DEMO = hostname
  .includes('github.io');

if (IS_DEMO) {
  return DEMO_DATA[endpoint];
}
```

Live: siva-sub.github.io/AgentPayment-Sandbox

Let's Connect

Sivasubramanian Ramanathan

Product Owner | Fintech & Innovation

 **Open for Roles In**

Product Management • Fintech • Payments
RegTech • Digital Assets

Also open to roles that sit between policy, technology, and stakeholder engagement.

 sivasub.com

 [LinkedIn](#)

 [GitHub](#)



Thank You 🙏

AgentPayment Sandbox

Testing the future of AI agent payments

 [Slides PDF](#)

 [Live Demo](#)

 [Documentation](#)