AI Agents Will Transact

But Today's Payments Weren't Built for Them

Sivasubramanian Ramanathan
Product Owner | Fintech & Innovation
Ex BIS Innovation Hub Singapore

🌏 Open for Fintech & Payments roles

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 This Matters to Me 🏛️

From BIS Innovation Hub to Agentic Commerce

My Background

- **Ex BIS Innovation Hub Singapore**
- Cross-border payments, digital innovation
- Financial infrastructure perspective

Why I'm Exploring This

1. **Infrastructure Gap**: Payments weren't built for agents
2. **Trust Crisis**: Who's liable when an agent buys wrong?
3. **Prompt Injection**: Can attackers hijack agent purchases?

> I build to understand emerging infrastructure.

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?

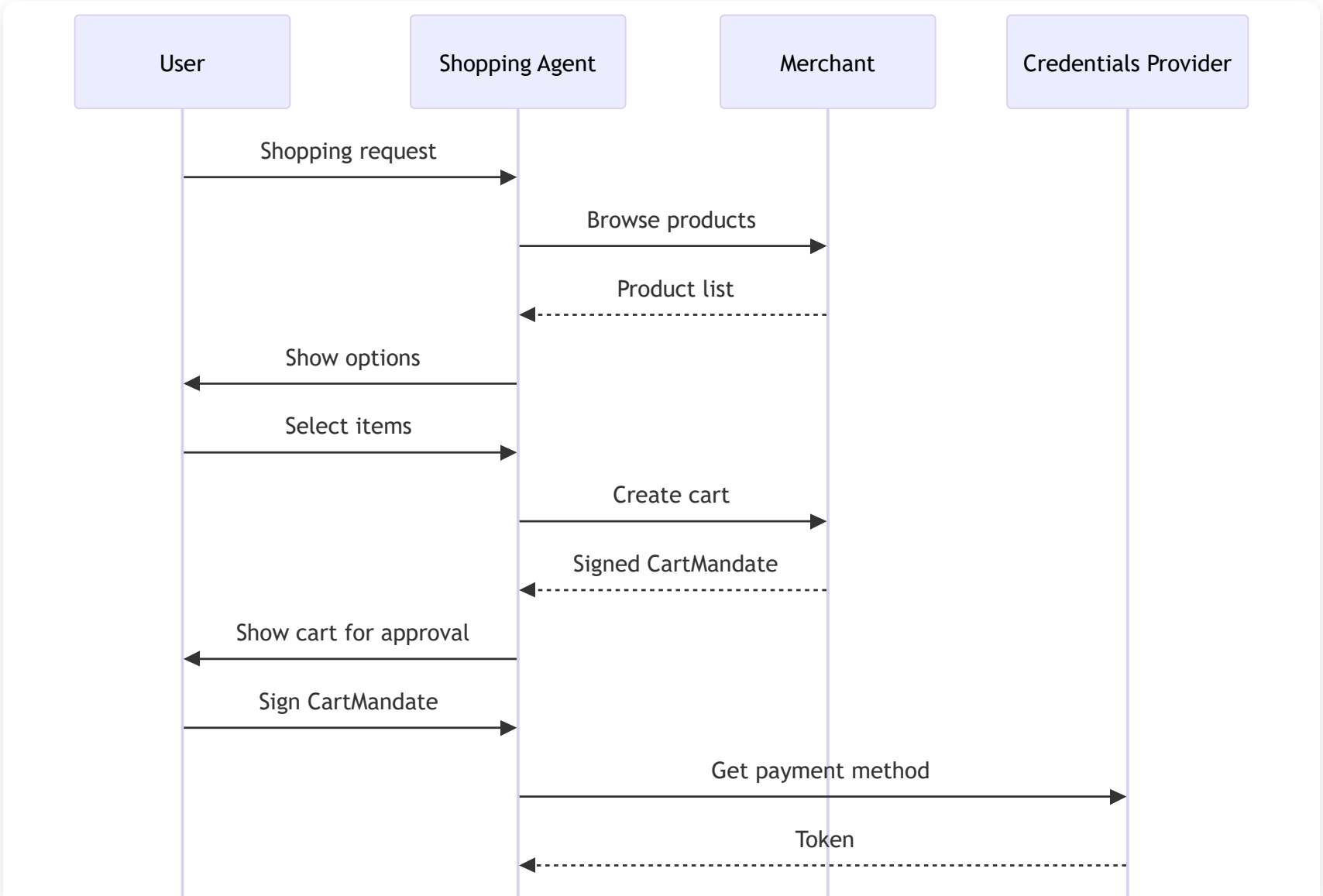
New World Solutions (AP2)

- **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.

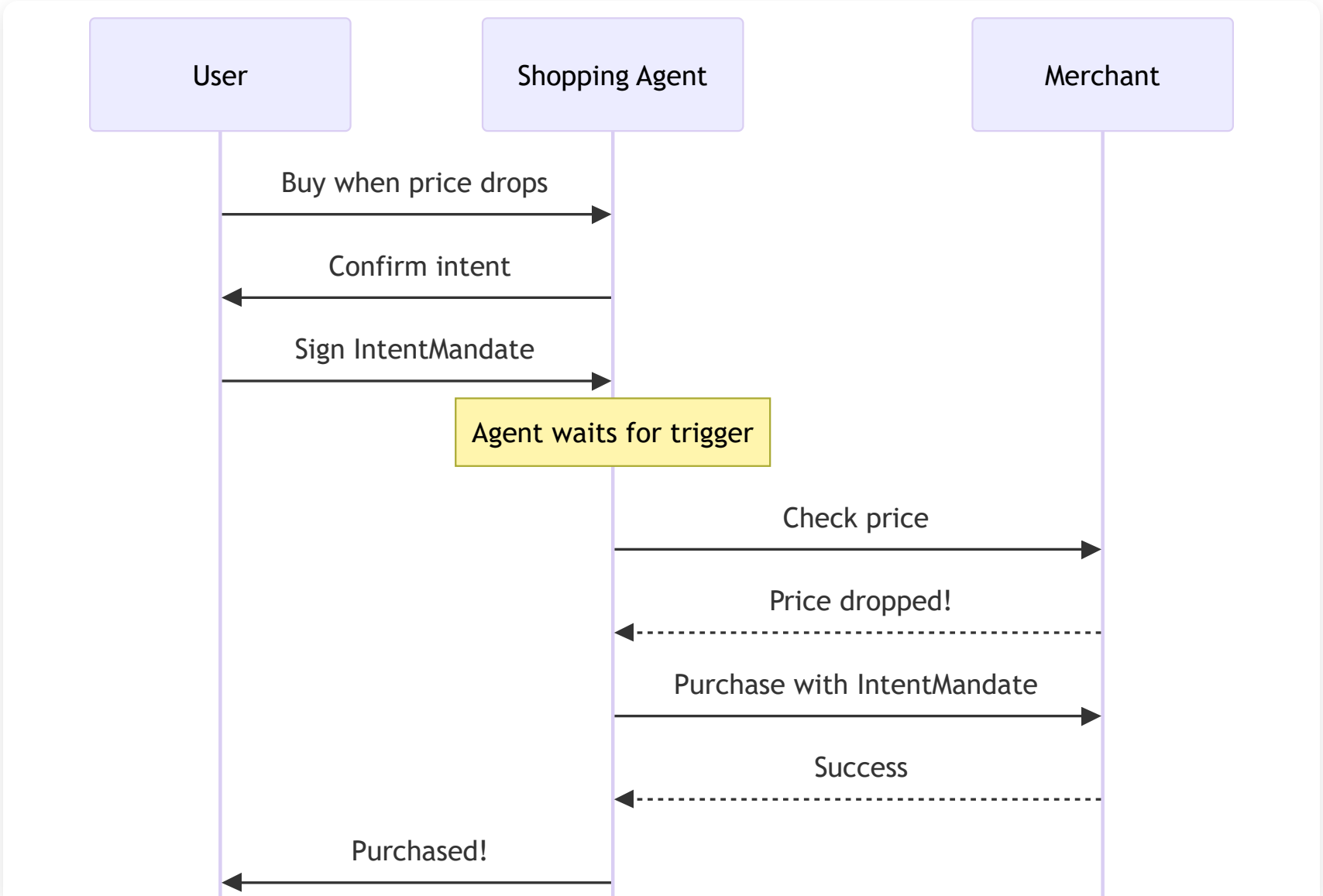
Life of a Transaction: Human Present

User is available to approve the final purchase.



Life of a Transaction: Human NOT Present 🤖

"Buy these shoes when price drops below $100"



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.
> See: google-agentic-commerce/a2a-x402

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

The Role-Based Architecture 🏗️

AP2 defines clear actors with separation of concerns.

Actor	Role
User	Human who delegates task
Shopping Agent (SA)	AI that builds the cart
Credentials Provider (CP)	Digital wallet, holds payment methods
Merchant Endpoint (ME)	Merchant's agent/API
Merchant Payment Processor (MPP)	Sends txn to network
Network/Issuer	Visa, Mastercard, bank

> **Key Insight**: No single entity holds all sensitive data.

Step-Up Challenges 🔒

Any party can require additional verification.

- Issuer can trigger **3D Secure**
- Merchant can require **user confirmation**
- Credentials Provider can request **OTP**

Human Not Present Scenario

If merchant is unsure about Intent Mandate, they can:

1. Force user back to session
2. Present final options
3. Require **Cart Mandate** instead

> This balances **autonomy** with **merchant confidence**.

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 | Ex BIS Innovation Hub Singapore

Open for Roles

Product Management • Fintech • Payments
RegTech • Digital Assets

🌐 sivasub.com
💼 LinkedIn
💻 GitHub

Thank You 🙏

AgentPayment Sandbox

Testing the future of AI agent payments

📄 **Slides PDF**
🔗 **Live Demo**
📚 **Documentation**