shopware

The Future of AI-Powered Commerce

AGENTIC COMMERCE

Protocols, Standards, and Reference Architectures for Agent-Based Commerce

Abstract

This comprehensive whitepaper provides developers with an in-depth guide to the Agentic Commerce ecosystem. Discover how AI agents are revolutionizing digital commerce through standardized protocols (UCP, AXP, A2A, ACP, AP2). Learn implementation strategies, explore technical specifications, study real-world API examples, and understand how to build the future of autonomous commerce.

Author

Stefan Hamann
Co-CEO, Shopware
Founding Member, Agentic Commerce Alliance

Version & Date

v1.0
2026-01-14

Contents

1 INTRODUCTION: THE DAWN OF AGENTIC COMMERCE

In the rapidly evolving world of digital commerce as of January 2026, AI agents are revolutionizing how transactions occur. Moving beyond human-centric interfaces, agentic commerce empowers autonomous agents to handle discovery, evaluation, negotiation, and purchases. This whitepaper, authored by Stefan Hamann, expands on the protocol ecosystem, providing comprehensive coverage of each protocol with detailed technical specifications, implementation guides, and practical examples.

1.1 The Strategic Imperative for Developers

Agentic commerce is projected to drive $190–$385 billion in U.S. online spending by 2030 (Morgan Stanley). Developers must focus on agent interfaces, leveraging the Agentic Commerce Alliance's open protocols for interoperability.

Challenges include LLM hallucinations, state management, and trust—solved by structured protocols.

1.2 Overview of the Protocol Ecosystem

The agentic commerce ecosystem comprises multiple interoperating protocols, each addressing specific aspects of autonomous commerce. The major protocols and their maintainers are:

Protocol	Core Responsibility
UCP	Canonical transaction and discovery envelope. Defines products, carts, checkout sessions, identity linking, and post-purchase lifecycle across merchants and agents.
AXP	Product semantics and experience layer. Adds structured product models, quality signals, and sandboxed experience embeddings (3D, AR, configurators) on top of UCP.
ACP	Checkout and transaction orchestration. Coordinates buyer intent, agent execution, merchant checkout APIs, and payment delegation in AI surfaces.
AP2	Payment authorization and settlement proof. Defines cryptographically verifiable mandates (Intent, Cart, Payment) for agent-initiated transactions.
A2A	Inter-agent communication and coordination. Enables negotiation, task delegation, and multi-agent workflows without sharing internal state.
MCP	Model context transport. Provides structured context exchange between models and external systems used by agents.

Table 1: Agentic Commerce Protocol Overview

The stack includes six major protocols, layered for comprehensive coverage. The foundation is built on communication standards (MCP, A2A), upon which commerce-specific protocols (UCP, AP2, ACP) operate, with AXP providing the richest experience layer.

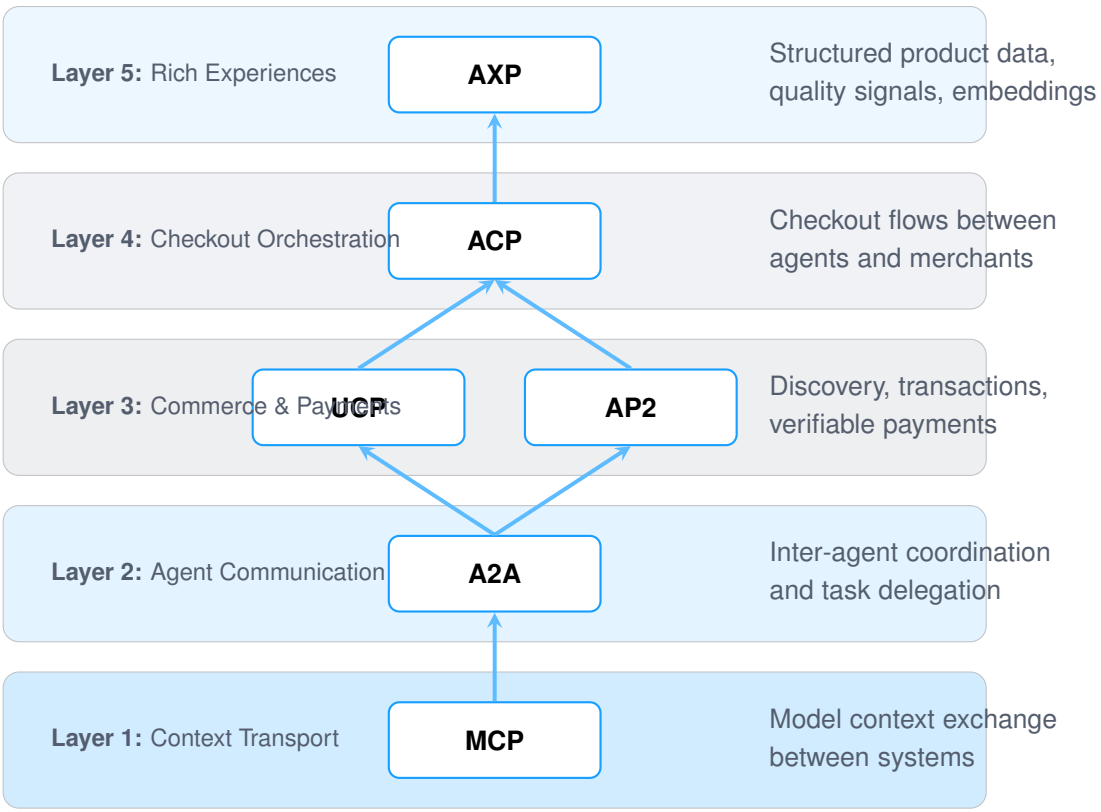

Figure 1: Agentic Commerce Protocol Stack – MCP and A2A form the communication foundation upon which commerce protocols (UCP, AP2, ACP) and experience layers (AXP) are built.

2 EXPANDED PROTOCOL BREAKDOWN

The agentic commerce ecosystem consists of multiple protocols, each designed for specific use cases and integration scenarios. The following table provides an overview of each protocol's primary purpose and application:

Protocol	Primary Use Case & Application
UCP	**Universal Commerce Protocol** — The foundational protocol for agentic commerce. Enables product discovery, checkout sessions, identity linking, and order management. Designed for interoperability across platforms, agents, and businesses. Supports both native and embedded checkout flows.
AXP	**Agentic Experience Protocol** — Extends UCP with rich product experiences. Provides structured product data (variants, configurators, subscriptions), quality signals (reviews, returns, trust scores), and sandboxed experience embeddings (3D viewers, AR, configurators).
A2A	**Agent-to-Agent Protocol** — Enables communication and collaboration between AI agents. Supports multi-agent coordination, task delegation, and inter-agent negotiation. Essential for complex purchasing decisions requiring multiple specialized agents.
ACP	**Agentic Commerce Protocol** — Interaction model for connecting buyers, AI agents, and businesses. Enables seamless checkout flows in AI assistants (e.g., ChatGPT) with secure payment token delegation. Focuses on instant checkout experiences.
AP2	**Agent Payments Protocol** — Provides verifiable and auditable payment mandates. Uses cryptographic proofs (ES256) for payment authorization. Supports Intent/Cart/Payment verifiable credentials (VDCs) for secure agent-driven transactions.

Table 2: Protocol Use Cases and Applications

2.1 UCP: Universal Commerce Protocol

Status: Industry Standard (Apache 2.0)

Website: https://ucp.dev

GitHub: https://github.com/google/ucp

UCP is the common language for platforms, agents, and businesses. It defines building blocks for agentic commerce—from discovering and buying to post-purchase experiences—allowing the ecosystem to interoperate through one standard without custom builds. UCP was built by the industry, for the industry to solve fragmented commerce journeys that lead to abandoned carts and frustrated shoppers, and to enable agentic commerce.

2.1.1 Co-Developers and Industry Adoption

UCP was co-developed by industry leaders including:

- **Co-Developers:** Google, Shopify, Etsy, Wayfair, Target, Walmart
- **PayPal Involvement:** PayPal collaborates on UCP and AP2, providing payment infrastructure and standards contributions
- **Endorsed by:** Adyen, American Express, Ant International, Best Buy, Carrefour, Chewy, Commerce, Flipkart, Gap, Kroger, Lowe's, Macy's, Mastercard, PayPal, Salesforce, Sephora, Shopee, Stripe, The Home Depot, Ulta, Visa, Worldpay, Zalando

2.1.2 Core Capabilities

UCP focuses on three core capabilities in its initial launch:

- **Checkout:** Support complex cart logic, dynamic pricing, tax calculations, and more across millions of businesses through unified checkout sessions.
- **Identity Linking:** OAuth 2.0 standard enables agents to maintain secure, authorized relationships without sharing credentials.
- **Order Management:** From purchase confirmation to delivery. Real-time webhooks power status updates, shipment tracking, and return processing.

2.1.3 Key Features and Capabilities

UCP is built for flexibility, security, and scale, designed to facilitate the entire commerce lifecycle:

- **Built on Industry Standards:** REST and JSON-RPC transports with built-in support for Agent Payments Protocol (AP2), Agent2Agent (A2A), and Model Context Protocol (MCP), enabling different systems to work together without custom integration.
- **Scalable and Universal:** Surface-agnostic design that can scale to support any commerce entity (from small businesses to enterprise scale) and all modalities (chat, visual commerce, voice, etc.).
- **Businesses at the Center:** Built to facilitate commerce, ensuring retailers retain control of their business rules and remain the Merchant of Record with full ownership of the customer relationship.
- **Open and Extensible:** Open and extensible by design, enabling development of community-driven capabilities and extensions across verticals.
- **Secure and Private:** Built on proven security standards for account linking (OAuth 2.0) and secure payment (AP2) via payment mandates and verifiable credentials.
- **Frictionless Payments:** Open wallet ecosystem with interoperability between providers to ensure buyers can pay with their preferred payment methods.

2.1.4 Deployment Options

UCP supports two primary integration patterns:

- **Native Checkout:** Integrate and negotiate directly with a seller's checkout API to power native UI and workflows for your platform. This approach gives you full control over the checkout experience while leveraging UCP's standardized APIs.
- **Embedded Checkout:** Embed and render business checkout UI to support complex checkout flows, with advanced capabilities like bidirectional communication, and payment and shipping address delegation. This approach allows businesses to maintain their branded checkout experience within your platform.

2.1.5 Use Cases

- Agent discovering products via semantic search across multiple merchants.
- Handling subscriptions with recurring mandates.
- Multi-agent coordination for complex purchasing decisions.
- Voice-commerce through conversational interfaces.

2.1.6 Security Considerations

- HTTPS required for all communications.
- OAuth 2.0 with short-lived tokens to mitigate session hijacking.
- AP2 integration for cryptographic payment proofs.
- Verifiable credentials backed by cryptographic proof of user consent.

2.1.7 Performance Optimizations

- Cache catalog data with appropriate TTL.
- Use GraphQL for efficient queries with precise data fetching.
- Implement webhook-based updates instead of polling.

2.1.8 Architecture Overview

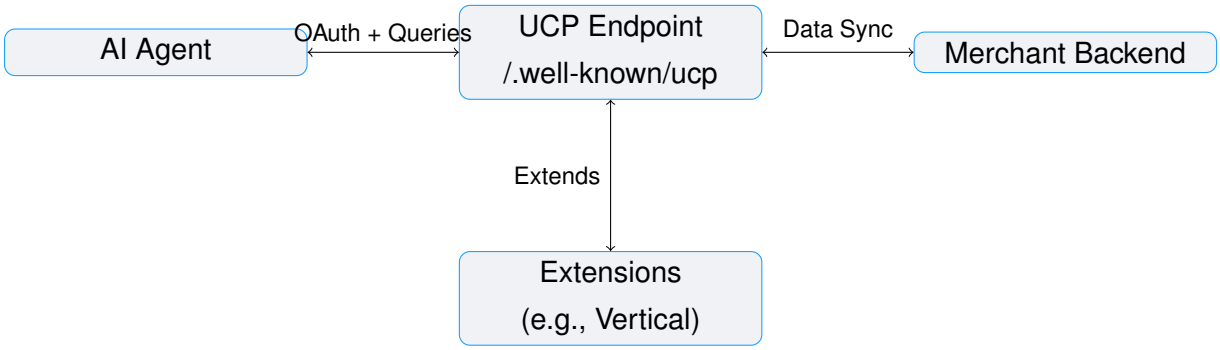

Figure 2: Expanded UCP Architecture.

2.1.9 API Examples (Expanded)

Checkout session:

Listing 1: UCP Checkout with Subscriptions

```
{
  "line_items": [
    {
      "price_id": "price_1ABC",
      "quantity": 1,
      "adjustable_quantity": true
```

```
    }
  ],
  "mode": "subscription",
  "billing_address_collection": "required",
  "success_url": "https://agent.example/success"
}
```

Order update webhook payload:

Listing 2: UCP Webhook Example

```
{
  "event": "order.updated",
  "data": {
    "order_id": "ord_123",
    "status": "shipped",
    "tracking_url": "https://shipper.example/track/xyz"
  }
}
```

2.1.10 Implementation in Shopware

Shopware provides a reference implementation of UCP through the SwagUcp plugin, available as open source on GitHub.

Installation:

1. Clone the repository: `git clone https://github.com/agentic-commerce-lab/SwagUcp.git`
2. Copy to plugins directory: `cp -r SwagUcp custom/plugins/SwagUcp/`
3. Install dependencies: `composer install`
4. Install and activate: `bin/console plugin:refresh && bin/console plugin:install -activate SwagUcp`
5. Run migrations: `bin/console database:migrate -all SwagUcp`
6. Clear cache: `bin/console cache:clear`

Features:

- UCP Discovery Profile at `/.well-known/ucp`
- Checkout Session API (Create, Get, Update, Complete, Cancel)

- JWK Signing Keys for Webhook Verification
- Agent Domain Whitelist Security
- Request Signature Verification (ES256)
- Automatic Key Generation

Documentation: https://github.com/agentic-commerce-lab/SwagUcp

Developer Tip

For local testing, use Docker with Shopware 6.7 and ngrok. Monitor logs for OAuth errors. The plugin automatically generates cryptographic keys on first access to the discovery endpoint.

2.2 AXP: Agentic Experience Protocol

Status: Prototype (Protocol Version: 2026-01-13)

GitHub: https://github.com/agentic-commerce-lab/AXP-protocol

Maintainers: Shopware, Agentic Commerce Alliance

The Agentic Experience Protocol (AXP) extends UCP to enable rich product experiences in agentic commerce interfaces. AXP is based on A2A, ACP, and UCP standards, and uses the same security protocols (ES256 signatures) as UCP. AXP is maintained by Shopware and the Agentic Commerce Alliance, providing structured product data, quality signals, and sandboxed experience embeddings for AI agents.

2.2.1 Core Requirements

AXP addresses three core requirements for modern agentic commerce:

1. **Product Data:** Structured product information supporting complex types (variants, configurators, events, subscriptions).
2. **Quality Data:** Trust signals including reviews, returns, intent information, and merchant reputation.
3. **Experience Embedding:** Sandboxed live experiences from merchants (3D viewers, configurators, AR).

2.2.2 UCP Compatibility

AXP is designed to work as an addon within UCP or as a standalone protocol:

Mode	Discovery Endpoint	Description
UCP Addon	/.well-known/ucp	AXP capabilities in UCP profile
Standalone	/.well-known/axp	Independent AXP discovery
Both	Both endpoints	Maximum compatibility

Table 3: AXP Deployment Options

2.2.3 Product Data Capabilities

- `dev.axp.product_data` — Base product data access
- `dev.axp.product_data.variants` — Variant product support
- `dev.axp.product_data.configurator` — Complex product configurators
- `dev.axp.product_data.events` — Event/ticket products
- `dev.axp.product_data.subscriptions` — Subscription products
- `dev.axp.product_data.bundles` — Product bundles

2.2.4 Quality Data Capabilities

- `dev.axp.quality_data.reviews` — Product reviews and ratings
- `dev.axp.quality_data.returns` — Return statistics and reasons
- `dev.axp.quality_data.intent` — Purchase intent signals
- `dev.axp.quality_data.trust` — Merchant and product trust

2.2.5 Experience Embedding Capabilities

- `dev.axp.experience_embedding.viewer_3d` — 3D product viewers
- `dev.axp.experience_embedding.configurator` — Visual configurators
- `dev.axp.experience_embedding.ar` — Augmented reality experiences
- `dev.axp.experience_embedding.video` — Interactive video
- `dev.axp.experience_embedding.custom` — Custom merchant experiences

2.2.6 Supported Transports

- **REST API:** OpenAPI 3.1 specification
- **MCP:** Model Context Protocol for AI agents
- **A2A:** Agent-to-Agent protocol

- **GraphQL:** Query language support
- **WebSocket:** Real-time updates

2.2.7 Use Cases

- Agent evaluating product fit via AR embedding before purchase.
- Analyzing aggregated reviews for sentiment-based recommendations.
- Configuring complex products (e.g., custom laptops) with real-time pricing.
- Assessing merchant trust scores for purchase decisions.

2.2.8 Security Considerations

- Uses same ES256 signing algorithm as UCP.
- Sandbox iframes for embedded experiences to prevent XSS.
- Content Security Policy (CSP) strictly applied.
- Resource limits enforced (memory, CPU, network).
- Experience content signing for integrity verification.

2.2.9 Performance Optimizations

- Compress JSON responses with gzip/brotli.
- Use CDNs for media and experience assets.
- Lazy-load 3D/AR content.
- Cache quality data aggregates.

2.2.10 Architecture Overview

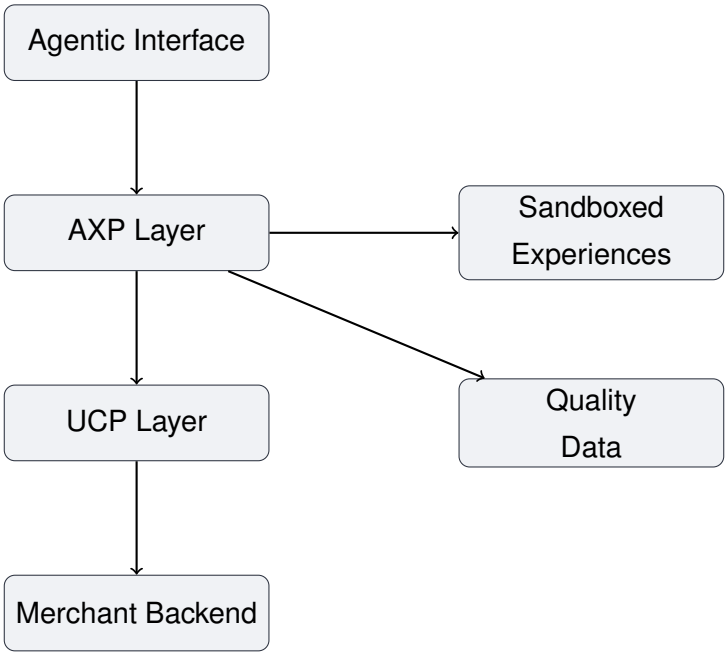

Figure 3: AXP Architecture with Experience Embeddings and Quality Data.

2.2.11 API Examples (Expanded)

Review aggregate response:

Listing 3: AXP Review Aggregate

```
{
  "reviews": {
    "aggregate": {
      "rating": 4.3,
      "count": 1247,
      "recommendation_percentage": 94,
      "verified_purchase_count": 1089
    },
    "aspect_ratings": [
      {"aspect": "quality", "rating": 4.5},
      {"aspect": "value", "rating": 4.1}
    ]
  }
}
```

Merchant trust data:

Listing 4: AXP Merchant Trust

```
{
  "merchant_trust": {
    "trust_score": {"score": 94, "tier": "gold"},
    "verification": {
      "business_verified": true,
      "address_verified": true
    },
    "performance_metrics": {
      "on_time_delivery_rate": 97.3,
      "resolution_rate": 99.1,
      "response_time_hours": 2.4
    }
  }
}
```

2.2.12 Implementation Notes

- Deploy standalone or as UCP addon (shared signing keys).
- Use `/.well-known/axp` for standalone discovery.
- AXP products can be directly used in UCP checkout sessions via `/api/axp/products/{id}/ucp-item`.
- In Shopware, use custom fields for trust signals and product quality data.

2.3 A2A: Agent2Agent Protocol

Status: v0.3.0 (Apache 2.0, Linux Foundation, 21.4k+ GitHub stars)

Website: https://a2a-protocol.org

GitHub: https://github.com/a2aproject/A2A

The Agent2Agent (A2A) Protocol addresses a critical challenge in the AI landscape: enabling gen AI agents, built on diverse frameworks by different companies running on separate servers, to communicate and collaborate effectively—as agents, not just as tools. A2A provides a common language for agents, fostering a more interconnected, powerful, and innovative AI ecosystem.

2.3.1 Why A2A?

As AI agents become more prevalent, their ability to interoperate is crucial for building complex, multi-functional applications:

- **Break Down Silos:** Connect agents across different ecosystems.
- **Enable Complex Collaboration:** Allow specialized agents to work together on tasks that a single agent cannot handle alone.
- **Promote Open Standards:** Foster community-driven approach to agent communication.
- **Preserve Opacity:** Allow agents to collaborate without sharing internal memory, proprietary logic, or specific tool implementations.

2.3.2 Key Features and Capabilities (Expanded)

- **Standardized Communication:** JSON-RPC 2.0 over HTTP(S).
- **Agent Discovery:** Via “Agent Cards” detailing capabilities and connection info.
- **Flexible Interaction:** Supports synchronous request/response, streaming (SSE), and asynchronous push notifications.
- **Rich Data Exchange:** Handles text, files, and structured JSON data.
- **Enterprise-Ready:** Designed with security, authentication, and observability in mind.

2.3.3 Available SDKs

A2A provides official SDKs for multiple languages:

- **Python:** `pip install a2a-sdk`
- **Go:** `go get github.com/a2aproject/a2a-go`
- **JavaScript:** `npm install @a2a-js/sdk`
- **Java:** Available via Maven
- **.NET:** `dotnet add package A2A`

2.3.4 Agent Capabilities

With A2A, agents can:

- Discover each other's capabilities.
- Negotiate interaction modalities (text, forms, media).
- Securely collaborate on long-running tasks.
- Operate without exposing their internal state, memory, or tools.

2.3.5 Use Cases

- Real-time price negotiations between buyer and seller agents.
- Multi-agent coordination for supply chain optimization.
- Collaborative research across specialized knowledge agents.
- Cross-platform task delegation and completion.

2.3.6 Security Considerations

- OIDC-based authentication for agent identity.
- Encrypted message transport for sensitive data.
- Authorization schemes defined in Agent Cards.
- Audit logging for compliance and debugging.

2.3.7 Performance Optimizations

- WebSockets for low-latency bidirectional communication.
- SSE for efficient streaming of updates.
- Long-Running Operations (LRO) support for complex tasks.

2.3.8 Architecture Overview

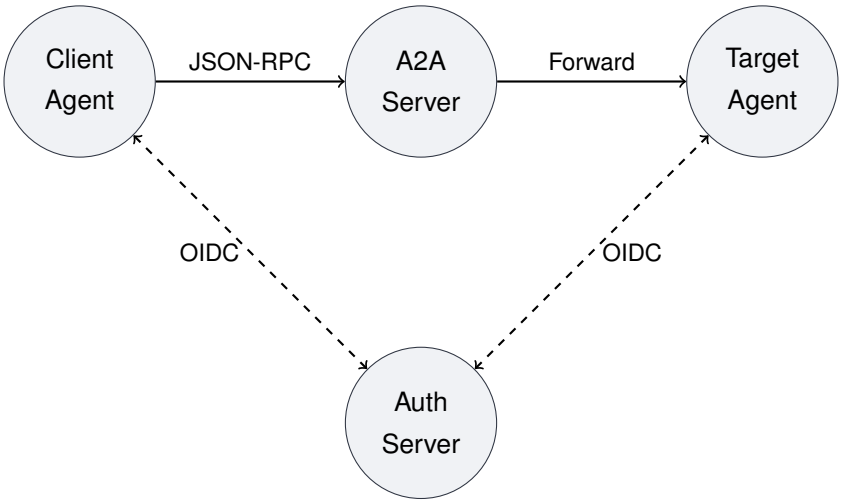

Figure 4: A2A Protocol: Agent Communication with Authentication.

2.3.9 API Examples (Expanded)

Task creation example:

Listing 5: A2A Task Creation

```
{
  "jsonrpc": "2.0",
  "method": "tasks/send",
  "params": {
    "message": {
      "role": "user",
      "parts": [{"text": "Find best price for laptop"}]
    }
  },
  "id": 1
}
```

Agent Card discovery: Query `/.well-known/agent.json`

2.3.10 Roadmap

- **Agent Discovery:** Formalize authorization schemes in AgentCard.
- **Agent Collaboration:** Investigate QuerySkill() for dynamic capability checking.
- **Task Lifecycle:** Support dynamic UX negotiation within tasks.
- **Transport:** Improvements to streaming reliability and push notifications.

2.3.11 Implementation Notes

- Publish Agent Cards at well-known endpoints.
- Handle async operations with appropriate timeouts.
- The protocol is an open source project under the Linux Foundation.

2.4 ACP: Agentic Commerce Protocol

Status: Draft (Apache 2.0, Maintained by OpenAI and Stripe)

Website: https://agenticcommerce.dev

GitHub: https://github.com/agentic-commerce-protocol/agentic-commerce-protocol

The Agentic Commerce Protocol (ACP) is an interaction model and open standard for connecting buyers, their AI agents, and businesses to complete purchases seamlessly. It enables commerce transactions through AI agents while maintaining security and trust.

2.4.1 Value Proposition

- **For Businesses:** Reach more customers. Sell to high-intent buyers by making products and services available for purchase through AI agents—all while using existing commerce infrastructure.
- **For AI Agents:** Embed commerce into applications. Let users discover and transact directly with businesses without being the merchant of record.
- **For Payment Providers:** Grow volume. Process agentic transactions by passing secure payment tokens between buyers and businesses through AI agents.

2.4.2 Key Features and Capabilities (Expanded)

- **Specifications:** OpenAPI (YAML) for HTTP API spec, JSON Schema for data models.
- **Agentic Checkout API:** Standardized checkout endpoints for agent integration.
- **Delegate Payment:** Secure payment delegation between agents and payment providers.
- **RFCs:** Human-readable design documents with rationale, flows, and rollout plans.
- **Examples:** Sample requests and responses for quick implementation.

2.4.3 Reference Implementations

ACP has been first implemented by both OpenAI and Stripe:

- **OpenAI:** Integration with ChatGPT and other AI agent surfaces.

- **Stripe:** Leverage existing payment and merchant tooling.

2.4.4 Use Cases

- Instant checkouts in ChatGPT and other AI assistants.
- Agent-managed subscriptions and recurring payments.
- Cross-platform commerce through multiple AI agent surfaces.
- Seamless payment token handling without exposing card details.

2.4.5 Security Considerations

- Tokenize all payment credentials (SharedPaymentToken).
- Never store raw card data in agent systems.
- PCI DSS compliance through payment provider integration.
- Cryptographic verification of transaction authorization.

2.4.6 Performance Optimizations

- Batch requests for high-volume scenarios.
- Async processing for long-running transactions.
- Caching of merchant capability profiles.

2.4.7 Architecture Overview

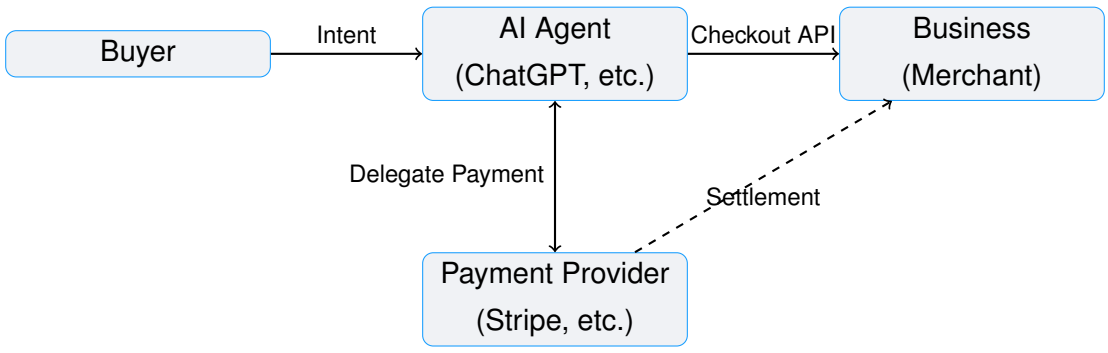

Figure 5: ACP Architecture: Buyer, Agent, Business, and Payment Provider Interactions.

2.4.8 Governance

ACP is jointly governed by OpenAI and Stripe as Founding Maintainers, with a clear path toward broader community governance and neutral foundation stewardship as the ecosystem matures.

2.4.9 Implementation Notes

- Review OpenAPI specs and JSON Schemas from the repository.
- Choose reference implementation (OpenAI or Stripe).
- Follow documentation guides for integration.
- Test using provided examples.
- All changes must include updated OpenAPI/JSON Schemas, examples, and changelog entries.

2.5 AP2: Agent Payments Protocol

Status: v0.1.0 (Google, 2.7k stars)

AP2 for verifiable payments.

2.5.1 Key Features and Capabilities (Expanded)

- **Mandates:** Intent/Cart/Payment VDCs.
- **Types:** Cards, crypto, transfers.
- **Flows:** Human-present/not-present.

2.5.2 Use Cases

- Authorizing agent purchases under limits. - Auditing transactions.

2.5.3 Security Considerations

Cryptographic signing (ES256); store mandates securely.

2.5.4 Performance Optimizations

Pre-validate mandates.

2.5.5 Architecture Overview

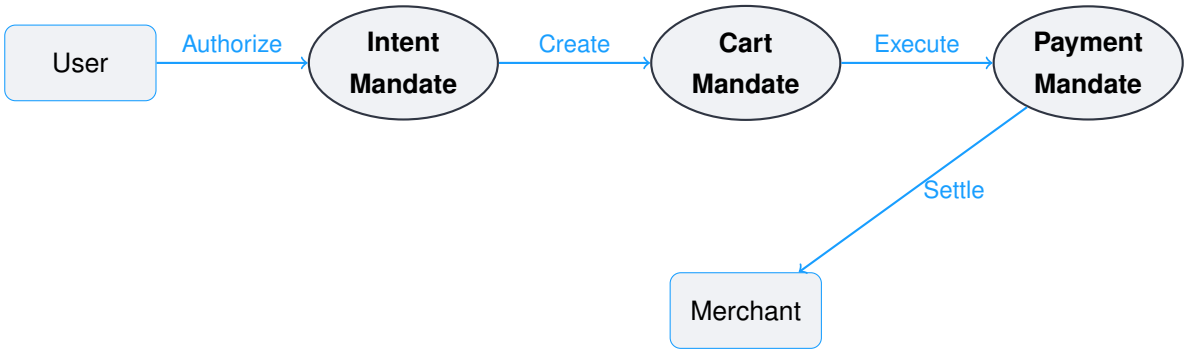

Figure 6: AP2 Mandate Flow: Verifiable Credentials for Agent Payments.

2.5.6 API Examples

Intent Mandate JWT (truncated): `eyJhbGciOiJFUzI1NiJ9...`

2.5.7 Implementation Notes

Use JWK for keys.

2.6 PayPal Agentic Commerce & Store Sync

Status: Production

Website: https://www.paypal.com/us/business/ai

Documentation: https://developer.paypal.com/docs/agentic-commerce/

Built on PayPal's trusted commerce infrastructure, PayPal's agentic commerce services are a suite of solutions that unlock agentic shopping experiences for buyers, sellers, and AI agents through natural-language interactions. These services power customers' ability to discover products, manage shopping carts, process payments, and complete transactions seamlessly across multiple merchant partners and commerce platforms.

2.6.1 Store Sync: The Core of PayPal's Agentic Commerce

Store Sync is PayPal's flagship agentic commerce service that makes any merchant's product data discoverable within leading AI channels while seamlessly dropping orders to existing fulfillment and management systems. Store Sync consists of two core components:

1. Product Feed Integration

- Automatically synchronizes product catalogs from multiple merchant partners and e-commerce platforms
- Maintains near-real-time inventory data, pricing information, and product attributes
- Enables AI agents to access comprehensive product information for intelligent shopping recommendations
- Supports Google Shopping CSV format with required fields: `id`, `item_group_id`, `title`, `description`, `link`, `image_link`, `price`, `availability`, `brand`

2. Cart Orchestration

- Streamlines the entire shopping journey from product discovery to checkout completion
- AI agents can create and manage shopping carts, apply discounts and coupons, handle shipping options
- Coordinates payment flows across different merchant systems
- Headless checkout architecture enables seamless integration with any AI surface

2.6.2 Benefits for Merchants

- **Fast Integration:** Product catalogs become discoverable through strategic partners including Wix, Cymbio, Commerce (BigCommerce & Feedonomics), and Shopware
- **Increased Discovery & Conversion:** Products are discoverable within AI shopping surfaces, driving conversions with intent-driven shopping
- **Preserve Customer Relationships:** Merchants remain the merchant of record and retain control of brand visibility and customer communications
- **One-to-Many Compatibility:** A single PayPal integration enables merchants to be seen across multiple AI shopping surfaces including Perplexity and the PayPal app shopping agent

2.6.3 Strategic Partnerships

PayPal has established major partnerships for agentic commerce:

- **OpenAI:** Powers instant checkout and agentic commerce in ChatGPT
- **Perplexity:** Launched Instant Buy ahead of Black Friday
- **Microsoft:** Powers Microsoft's launch of Copilot Checkout
- **Google Cloud:** Agentic commerce solution for merchants
- **Mastercard:** Partnership for agentic commerce standards

Platform Integration Partners:

- **Shopware:** Deep integration for Store Sync and Cart Orchestration
- **Wix:** E-commerce platform integration
- **Cymbio:** Multi-channel commerce enablement
- **Commerce (BigCommerce & Feedonomics):** Enterprise commerce integration

2.6.4 Key Differentiators

- **Global Scale:** Leveraging 400M+ active accounts, providing reach to 200+ markets

- **Foundational Trust:** 25+ years of PayPal innovation in fraud prevention, identity verification, and buyer-seller protection
- **Smart Wallet:** Vaulted payment experience enabling seamless checkout without redirects

2.6.5 Cart Orchestration API

The Cart API provides three core endpoints for the complete cart lifecycle:

Endpoint	Description
`POST /merchant-cart`	Create a new shopping cart
`PUT /merchant-cart/{id}`	Update cart (complete replacement)
`POST /merchant-cart/{id}/checkout`	Complete checkout and fulfill order

Table 4: PayPal Cart Orchestration API Endpoints

Cart Status Values:

- `CREATED` — Cart successfully created with payment token, ready for payment
- `INCOMPLETE` — Cart has validation issues (check `validation_issues`)
- `READY` — Previously incomplete cart is now ready for payment
- `COMPLETED` — Order finished, payment captured

2.6.6 Architecture Overview

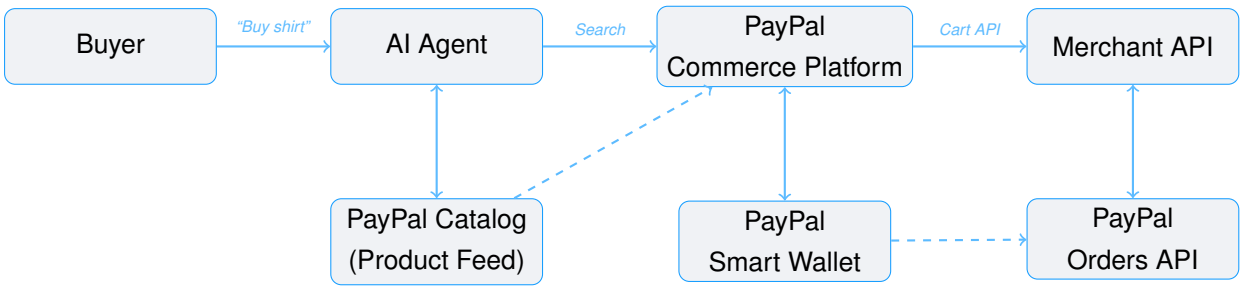

Figure 7: PayPal Store Sync Architecture: From buyer intent through AI agent to merchant fulfillment.

2.6.7 System Flow

The complete agentic commerce flow with Store Sync:

1. **Product Discovery:** Buyer asks AI agent for a product → Agent searches PayPal Catalog
2. **Cart Creation:** Agent sends `POST /merchant-cart` with items and shipping address
3. **Cart Validation:** Merchant API validates inventory, calculates taxes, returns cart with totals
4. **Payment Approval:** PayPal Smart Wallet handles payment (no redirect needed)
5. **Checkout Completion:** Agent sends `POST /merchant-cart/{id}/checkout` with payment token
6. **Order Fulfillment:** Merchant processes order, PayPal captures payment

2.6.8 Product Feed Integration Options

PayPal supports multiple ingestion methods for merchant product feeds:

- **SFTP/FTP Server:** Host files on your server, share credentials with PayPal
- **Cloud Storage (Your Bucket):** Google Cloud Storage or Amazon S3 with read access
- **Cloud Storage (PayPal Bucket):** Upload to PayPal-provisioned GCS path
- **API Access:** Expose paginated catalog API for daily synchronization
- **Direct URL:** Publicly accessible HTTPS URLs with consistent naming

2.6.9 API Example: Cart Creation

Listing 6: PayPal Store Sync Cart Creation Request

```
{
  "items": [
    {
      "variant_id": "SHIRT-BLUE-M",
      "quantity": 1,
      "name": "Blue Cotton T-Shirt (Medium)",
      "price": {"currency_code": "USD", "value": "25.00"}
    }
```

```
  ],
  "shipping_address": {
    "address_line_1": "123 Main Street",
    "admin_area_2": "San Jose",
    "admin_area_1": "CA",
    "postal_code": "95131",
    "country_code": "US"
  },
  "payment_method": {"type": "paypal"}
}
```

Listing 7: PayPal Store Sync Cart Response

```
{
  "id": "CART-ABC123",
  "status": "CREATED",
  "payment_method": {
    "type": "PAYPAL",
    "token": "EC-7U8939823K567"
  },
  "totals": {
    "subtotal": {"currency_code": "USD", "value": "25.00"},
    "shipping": {"currency_code": "USD", "value": "5.99"},
    "tax": {"currency_code": "USD", "value": "2.70"},
    "total": {"currency_code": "USD", "value": "33.69"}
  },
  "validation_issues": []
}
```

2.6.10 Protocol Collaborations

PayPal is actively involved in multiple agentic commerce protocols:

- **UCP:** Endorsed and contributing to Universal Commerce Protocol development
- **AP2:** Collaborating with Google on the Agent Payments Protocol for verifiable, auditable transactions
- **Standards:** Working with Mastercard and other partners on agentic commerce standards

2.6.11 Security & Trust

- **JWT Authentication:** PayPal-issued tokens for all API calls

- **Tokenized Payments:** Never expose raw card data
- **Purchase Protection:** Available on eligible transactions
- **Seller Protection:** Available on eligible purchases
- **PCI DSS Compliance:** Enterprise-grade security infrastructure

2.6.12 Implementation Steps

1. **Request Access:** Contact PayPal AI team to enable Store Sync
2. **Product Feed Setup:** Configure product feed ingestion (SFTP, Cloud, API, or URL)
3. **Merchant Manifest:** Provide merchant metadata (storeName, storeUrl, currency, country)
4. **Cart API Integration:** Implement `/merchant-cart` endpoints for cart lifecycle
5. **PayPal Orders Integration:** Connect Cart API with PayPal Orders API v2 for payment processing
6. **Testing:** Validate integration with test scenarios before go-live

Developer Tip

Store Sync requires completing the product feed integration before setting up cart orchestration. Use PayPal's JWT tokens for authentication—tokens are generated and managed by PayPal, not merchants. The Cart API uses PUT for complete replacement of cart data, not incremental updates.

3 INDUSTRY ENDORSEMENTS AND ECOSYSTEM

The agentic commerce ecosystem has gained significant industry support, with major players across retail, payments, and technology endorsing and implementing these standards.

3.1 UCP Endorsers

The Universal Commerce Protocol is endorsed across the ecosystem:

Retailers & Marketplaces:

- Best Buy
- Carrefour
- Chewy
- Etsy
- Flipkart
- Gap
- Kroger
- Lowe's
- Macy's
- Sephora
- Shopee
- Target
- The Home Depot
- Ulta
- Walmart
- Wayfair
- Zalando

Payment Providers & Networks:

- Adyen
- American Express
- Ant International
- Mastercard
- PayPal
- Stripe
- Visa
- Worldpay

Technology & Platforms:

- Google
- Salesforce
- Shopify
- Commerce

3.2 ACP Partnerships

The Agentic Commerce Protocol has production implementations from:

- **OpenAI:** ChatGPT commerce integration
- **Stripe:** Payment infrastructure for agentic transactions

3.3 A2A Community

The Agent2Agent Protocol is an open source project under the Linux Foundation with:

- 21,400+ GitHub stars
- 2,200+ forks
- 134+ contributors
- Active community discussions

3.4 PayPal Partnerships

PayPal's agentic commerce initiative includes partnerships with:

- **OpenAI:** Instant checkout in ChatGPT
- **Microsoft:** Copilot Checkout integration
- **Google Cloud:** Merchant solutions
- **Mastercard:** Agentic commerce standards
- **Perplexity:** Instant Buy features

4 THE INTEGRATED WORKFLOW: SYMBIOTIC PROTOCOL INTERACTIONS

The agentic commerce protocols work together to enable seamless end-to-end transactions. Each protocol handles a specific aspect of the commerce journey.

1. **Discovery (UCP/PayPal):** Agent discovers products via UCP endpoints or PayPal's commerce APIs.
2. **Evaluation (AXP):** Agent retrieves quality data, reviews, and trust signals.
3. **Negotiation (A2A):** Multi-agent coordination for complex decisions.
4. **Checkout Orchestration (ACP):** Checkout flow management between agent and merchant.
5. **Settlement (AP2):** Cryptographically verifiable payment mandates.
6. **Post-Purchase (UCP webhooks):** Order tracking and fulfillment updates.

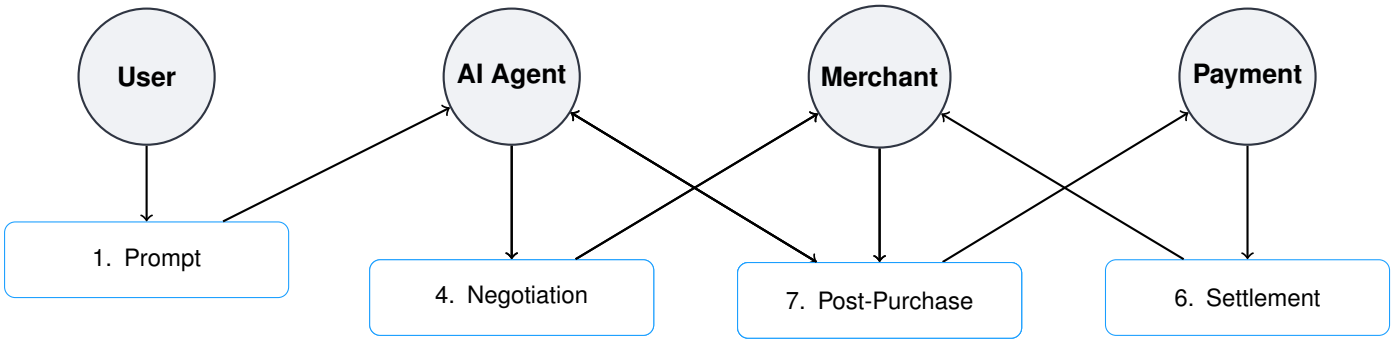

Figure 8: Complete Agentic Commerce Flow: From User Intent to Post-Purchase.

5 ROADMAP, CHALLENGES, AND CONCLUSION

5.1 Future Developments

- **Q1 2026:** AXP moves to stable release; broader UCP adoption.
- **Q2 2026:** B2B extensions in A2A; enhanced agent discovery.
- **Q3 2026:** Cross-protocol interoperability improvements.
- **2027:** Path toward W3C standards; neutral foundation governance.

5.2 Protocol Evolution

- **UCP:** Continued expansion of capabilities and vertical extensions.
- **ACP:** Moving from draft to stable with broader merchant adoption.
- **A2A:** Enhanced agent discovery and dynamic skill querying.
- **AXP:** Production-ready experience embedding and quality signals.
- **AP2:** Expanded payment method support and mandate types.

5.3 Resources

- **UCP:** https://ucp.dev | https://github.com/google/ucp
- **ACP:** https://agenticcommerce.dev | https://github.com/agentic-commerce-protocol/agentic-commerce-protocol
- **A2A:** https://a2a-protocol.org | https://github.com/a2aproject/A2A
- **AXP:** https://github.com/agentic-commerce-lab/AXP-protocol
- **PayPal:** https://www.paypal.com/us/business/ai
- **Agentic Commerce Alliance:** https://agentic-commerce.dev
- **Shopware UCP Plugin:** https://github.com/agentic-commerce-lab/SwagUcp

5.4 Conclusion

Agentic commerce represents a fundamental shift in how transactions occur in the digital economy. The protocols described in this whitepaper—UCP, ACP, A2A, AXP, AP2, and PayPal's agentic commerce services—form a comprehensive ecosystem that enables AI agents to discover, evaluate, negotiate, and complete purchases autonomously.

5.4.1 Actionable Steps for eCommerce Developers

As an eCommerce developer, here's how you can get started with agentic commerce today:

1. Implement UCP Foundation

- Start by implementing UCP discovery endpoints (`/.well-known/ucp`) to make your store discoverable by AI agents.
- Set up checkout session APIs to enable agent-driven purchases.
- For Shopware users, leverage the open-source SwagUcp plugin as a reference implementation.
- Configure OAuth 2.0 for secure agent authentication and identity linking.

2. Enhance with AXP for Rich Experiences

- Implement AXP product data capabilities to provide structured product information with variants, configurators, and subscriptions.
- Expose quality data endpoints (reviews, returns, trust signals) to help agents make informed purchase decisions.
- Enable experience embedding for 3D viewers, AR experiences, and interactive configurators.
- Use AXP as a UCP addon or standalone protocol, depending on your architecture.

3. Integrate Payment Protocols

- Implement AP2 mandates for verifiable, auditable agent payments.
- Support PayPal's agentic commerce APIs for instant checkout experiences.

- Ensure cryptographic signing (ES256) for all payment-related operations.
- Store mandates securely and implement proper validation workflows.

4. Enable Multi-Agent Collaboration

- Publish Agent Cards at `/.well-known/agent.json` endpoints.
- Implement A2A protocol support for agent-to-agent communication.
- Enable ACP checkout flows for seamless integration with AI assistants like ChatGPT.

5. Security and Compliance

- Implement domain whitelisting for authorized AI agents.
- Use request signature verification (ES256) for high-security scenarios.
- Ensure all communications use HTTPS.
- Follow GDPR and privacy regulations for agent-driven transactions.
- Implement proper audit logging for compliance and debugging.

6. Testing and Validation

- Test with major AI platforms (OpenAI, Google Gemini, Anthropic).
- Use the Agentic Commerce Playground at `https://agentic-commerce.dev` for testing.
- Monitor API calls and implement proper error handling.
- Set up webhook endpoints for real-time order updates.

5.4.2 Join the Agentic Commerce Community

The agentic commerce ecosystem is built on open standards and thrives through community contribution. Here's how you can get involved:

Contribute to Protocol Development

- **AXP Protocol:** Join the development of the Agentic Experience Protocol at https://github.com/agentic-commerce-lab/AXP-protocol. AXP is actively maintained by Shopware and the Agentic Commerce Alliance, and we welcome contributions from the community.
- **UCP:** Contribute to the Universal Commerce Protocol at https://github.com/google/ucp.
- **A2A:** Participate in the Agent2Agent Protocol development under the Linux Foundation.

Join the Agentic Commerce Alliance

- Visit https://agentic-commerce.dev to learn more about the Alliance and its mission.
- Participate in working groups and contribute to protocol specifications.
- Share your implementation experiences and help improve documentation.
- Connect with other developers building agentic commerce solutions.

Share Your Implementation

- Open source your UCP/AXP implementations to help others learn.
- Contribute reference implementations and code examples.
- Write blog posts and tutorials about your agentic commerce journey.
- Present at conferences and meetups to spread awareness.

5.4.3 The Path Forward

The protocols described in this whitepaper are not just theoretical concepts—they are production-ready standards being implemented by major retailers, payment providers, and technology platforms today. As an eCommerce developer, you have the opportunity to:

- **Be an Early Adopter:** Position your business at the forefront of the agentic commerce revolution.
- **Shape the Standards:** Contribute to protocol development and help define the future of commerce.

- **Build Innovative Solutions:** Leverage these protocols to create unique, AI-powered shopping experiences.
- **Join a Growing Ecosystem:** Connect with developers, merchants, and platforms building the future of commerce together.

The future of commerce is agentic. The protocols are ready, the tools are available, and the community is welcoming. Start implementing today, contribute to the standards, and help shape the future of how AI agents and humans will shop together.

Get Started: Visit https://agentic-commerce.dev to explore the protocols, access the developer playground, and join the Alliance. Contribute to AXP at https://github.com/agentic-commerce-lab/AXP-protocol and help build the future of agentic commerce.