>_ AgentRails

x402 Protocol Whitepaper

HTTP-Native Payments for the Agent Economy

February 2026

www.agentrails.io

Abstract

The x402 protocol enables HTTP-native payments for AI agents and automated systems. By leveraging HTTP status code 402 (Payment Required), the protocol allows any API to request payment inline with the HTTP request/response cycle. Agents pay per-request in USDC stablecoin using EIP-3009 (transferWithAuthorization), eliminating the need for API keys, subscription tiers, and manual credential management.

This whitepaper describes the x402 V2 protocol specification as implemented by AgentRails, including the payment flow, security model, network support, and enterprise governance extensions.

The Problem

Today's API economy relies on a stack of legacy abstractions: developer portals, API keys, OAuth tokens, subscription tiers, and monthly invoices. This model was designed for human developers, not autonomous agents.

For AI agents, this model breaks down:

- Agents cannot sign up for accounts or complete email verification
- API keys are static secrets that must be provisioned, stored, and rotated
- Subscription tiers force pre-commitment to usage levels
- Each new vendor requires human approval and procurement
- Invoice reconciliation across dozens of services is operationally expensive

The result: agents are bottlenecked by human gatekeeping at every API boundary. The x402 protocol removes this bottleneck by making payment the authentication mechanism.

Protocol Design

Core Principle

x402 uses HTTP status code 402 (Payment Required) as defined in RFC 7231. The protocol is stateless, requires no pre-registration, and works with any HTTP client that can read response headers and retry requests.

Payment Flow

1. CLIENT sends a standard HTTP request to a protected endpoint.

2. SERVER returns HTTP 402 with a PAYMENT-REQUIRED header containing a base64-encoded JSON payload specifying: protocol version, price (in smallest units), currency, network (CAIP-2 format), receiver address, and payment description.

3. CLIENT inspects the payment requirements, checks budget limits, and signs an EIP-3009 transferWithAuthorization message using the agent's wallet private key.

4. CLIENT retries the original request with a PAYMENT-SIGNATURE header containing the signed payment payload (from, to, value, validAfter, validBefore, nonce, signature).

5. SERVER (or facilitator) verifies the signature off-chain using EIP-712 typed data hashing and ECDSA signature

recovery. If valid, the facilitator submits the transferWithAuthorization transaction on-chain.

6. SERVER returns the requested data along with a PAYMENT-RESPONSE header containing the transaction hash and settlement status.

Security Model

EIP-3009: transferWithAuthorization

x402 uses EIP-3009, a USDC-native standard that allows gasless, authorized transfers. The payer signs a typed data message (EIP-712) authorizing a specific transfer. The facilitator submits the transaction, paying gas fees on behalf of the agent. This means:

- Agents never need native gas tokens (ETH, etc.)
- Each authorization is single-use (unique nonce)
- Authorizations have time bounds (validAfter, validBefore)
- The payer's private key never leaves the agent's environment

Off-Chain Verification

Before submitting on-chain, the facilitator verifies the signature off-chain using EIP-712 typed data hashing and ecrecover. This prevents invalid transactions from consuming gas and enables sub-second verification.

Facilitator Role

The facilitator is a trusted intermediary that verifies signatures and submits on-chain transactions. In hosted mode, AgentRails operates the facilitator. In enterprise mode, organizations run their own facilitator with full control over settlement timing, batching, and network selection.

Network Support

x402 V2 supports multiple EVM-compatible networks using CAIP-2 identifiers:

- **Base (eip155:8453)** -- Coinbase L2, low gas fees, fast finality
- **Base Sepolia (eip155:84532)** -- Base testnet for development
- **Ethereum (eip155:1)** -- Ethereum mainnet for high-value transactions
- **Ethereum Sepolia (eip155:11155111)** -- Ethereum testnet
- **Arc (eip155:5042002)** -- Circle's L2 with native USDC (testnet)

Multi-network support allows servers to accept payment on any supported chain, and agents can pay on the network with the lowest fees or fastest settlement.

Enterprise Extensions

AgentRails extends the base x402 protocol with enterprise governance features:

- **Policy engine** -- Server-side enforcement of spending limits, rate controls, and destination rules

- **Audit logging** -- Every payment decision logged with timestamps, agent identity, and on-chain hashes
- **Kill switches** -- Instantly revoke an agent's payment capability
- **Approval workflows** -- Route high-value transactions through human approval via Power Automate
- **Trust scoring** -- Verify service reputation before authorizing payment

SDK Integrations

AgentRails provides SDK packages that handle the full x402 flow automatically. When an agent's HTTP request receives a 402 response, the SDK:

- Parses the PAYMENT-REQUIRED header
- Checks the agent's budget and policy limits
- Signs the EIP-3009 authorization
- Retries the request with the PAYMENT-SIGNATURE header
- Returns the data to the agent transparently

Available SDKs:

- langchain-x402 (PyPI) -- LangChain toolkit with x402 payment tools
- crewai-x402 (PyPI) -- CrewAI toolkit for multi-agent payment crews
- AgentRails.SemanticKernel.X402 (NuGet) -- Semantic Kernel plugin
- AgentRails.AgentFramework.X402 (NuGet) -- Microsoft Agent Framework tools
- Copilot Studio connector -- OpenAPI spec import for Power Platform

Comparison: x402 vs Traditional API Access

Traditional: Sign up, verify email, add credit card, generate API key, store securely, rotate periodically, pay monthly subscription regardless of usage, reconcile invoices.

x402: Agent calls API, gets 402, pays exact amount per-request, gets instant access. No signup, no credentials, no subscriptions. Every payment is cryptographically verifiable.

Learn More

Protocol spec: github.com/kmatthewsio/AgenticCommerce | Docs: www.agentrails.io/docs | Sandbox: sandbox.agentrails.io/swagger