

Paying Without Keys: Machine-Native Micropayments over x402

Gleem Computer LLC
gleem.computer · Robinhood Chain

A gleem research brief · 002

Abstract

An autonomous agent must pay for the data and computation it consumes without a human entering a card or embedding an API key. We describe how gleem uses the x402 protocol over the EIP-3009 “transfer with authorization” primitive to let an agent settle a one-time, capped micropayment with a signature alone — no account, no subscription, no key in a config. We give the protocol as a small state machine, state its safety property (no charge without result), and describe how it is realized on Robinhood Chain, whose settlement stablecoin USDG implements EIP-3009. We are explicit about what is live and what is under construction.

1 The problem: credentials do not scale to agents

Every way a human pays online assumes a human: a card entered by hand, an account created with an email, an API key pasted into a config. An agent that depends on any of these has a human-shaped hole at the center of its autonomy. It cannot open an account. It should not carry the human’s key. And a subscription is the wrong shape entirely: an agent needs to pay for one call, once, not to rent a month of access it may use twice.

The right primitive is a payment an agent can make with a signature and nothing else — scoped to one transfer, for one amount, to one recipient. HTTP status code 402, “Payment Required,” reserved at the birth of the web and long unused, is the natural home for it. The x402 protocol makes it concrete.

2 The exchange

The protocol is small; smallness is the claim. From the agent’s side, paying for a resource is four steps.

1. **Ask.** Request the resource with no payment.

Receive 402 and a challenge: asset, amount, recipient, nonce.

2. **Sign.** Produce an EIP-3009 `transferWithAuthorization` signature — an off-chain authorization for a one-time transfer of the named stablecoin, for the named amount, to the named recipient. No transaction is broadcast; no gas is spent by the agent.
3. **Present.** Send the request again with the signed authorization.
4. **Receive.** The server (or a facilitator) verifies and settles the authorization on-chain, then returns the result.

The agent holds no account and no API key. It holds a stablecoin balance and the ability to sign. Everything else is the protocol.

3 EIP-3009: the primitive

EIP-3009 defines `transferWithAuthorization`: a token holder signs an EIP-712 typed message authorizing a transfer, and *any* party can submit it on-chain and pay the gas. This is exactly the shape a machine payment needs:

- **Off-chain signature, on-chain settlement.** The agent signs; a relay (the facilitator) submits and pays gas, netting it from the amount. The agent never needs the gas token.
- **One-time, scoped.** Each authorization carries a random nonce and validity window, so it is single-use and cannot be replayed.
- **Recipient-bound.** The authorization names the payee, so it cannot be redirected.

On Robinhood Chain, the settlement stablecoin is USDG (Paxos), which implements EIP-3009. An authorization the agent signs is a message the USDG contract will honor.

4 Safety: no charge without result

The protocol’s safety property is structural. Model one request as a small automaton with states `AWAIT`, `CHALLENGED`, `VERIFYING`, `SETTLING`, `SERVED`, `REJECTED`. The transitions are constrained so that no path reaches `SERVED` without passing `SETTLING`, and no path charges a request that ends in `REJECTED`.

A well-behaved server settles only when about to return a result, and returns a paid result only against a settled payment. An agent that receives a 402 and stops has paid nothing.

This is not a property of the implementer’s care; it is a property of the transition graph. A protocol whose safety is structural is one an agent can trust without trusting its counterparty.

5 Validate before settle

A careful facilitator separates two checks. *Validation* is read-only: is the signature well-formed, does it recover to the claimed signer, is there sufficient balance, is the nonce unused? *Settlement* is the on-chain write. Validating before settling means an invalid request is rejected for free, and only a request that will succeed is ever submitted on-chain. It also lets a service expose a cheap “will this pay?” probe distinct from the paid call itself.

6 State on Robinhood Chain

We are explicit. The primitive is live: USDG on Robinhood Chain implements `transferWithAuthorization`, and its authorization signatures verify. The *facilitator* — the service that verifies and settles on a resource server’s behalf — is under construction for chain 4663, because the hosted facilitators serving other chains do not serve Robinhood Chain. gleem therefore operates its own. This brief describes the mechanism; it does not claim the settlement path is in production until that facilitator is. The distinction between a specified mechanism and a live deployment is one we keep everywhere.

7 Conclusion

An agent that pays with a signature alone — no account, no key, no subscription — is an agent

that can fund its own thinking. EIP-3009 supplies the primitive; x402 supplies the choreography; a low-gas EVM chain supplies the floor beneath which micropayments become economical. Assembled, they let an agent buy a cent of information to sharpen a dollar of judgment, without a human in the loop. That is the payment layer an autonomous agent requires, and it is the layer gleem is building on Robinhood Chain.

gleem.computer · see also: the gleem manifesto, and research 001, “Why Agents Need an Operating System.”