x402 Direct: an agent-initiated commerce layer above x402

x402 settles one signed payment for one resource. Real commerce needs a cart, an identified buyer, and a way to send money back. This paper specifies the three primitives Facet built to close that gap without touching the x402 spec itself, why the gap is real and not already closed by an existing scheme, and how the whole loop was proven end to end on Base mainnet with real USDC.

- PROVEN ON BASE MAINNET
- NON-CUSTODIAL
- RFC 9421 SIGNED
- COMPOSES WITH X402 AND UCP
- SETTLED IN PRODUCTION 2026-07-23

00 ABSTRACT

x402 is an open standard for internet-native payments: a resource server responds `402 Payment Required`, the client resubmits with a signed payment payload, and a facilitator verifies and settles it. Its most widely deployed scheme, `exact`, moves one authorized amount for one resource, once, and by the specification's own account has no built-in mechanism for returning funds. A second scheme, `auth-capture`, does support a refund, but as a scalar amount on one payment, gated by whichever facilitator or contract holds a capture-authorizer role, inside a fixed time window. Neither scheme, nor any other in the spec, defines a multi-item cart, a stateful checkout session, or an identity bound to the payment distinct from the signing wallet. This paper specifies x402 Direct, three primitives layered on top of x402's settlement leg rather than inside it: a signed checkout session that prices a real multi-line cart before any money moves, an agent-facing initiation route that expresses a partial refund, a damaged-goods return, and a full-order cancellation as one primitive with different inputs, and a non-custodial reverse settlement signed by the merchant's own wallet. The full loop, checkout through cancellation, is proven end to end on Base mainnet against a live commerce backend with real USDC, and every claim resolves to a transaction hash or a signature independently verifiable against a published key. The specification anticipated the need for a reverse leg: an `auth-capture` scheme was merged on 2026-05-13, and its reference TypeScript package shipped a client on 2026-05-29. That package states, as of this writing, that it "currently ships the client only" and that "server and facilitator support follow in a later change." The loop documented here settles real value on Base mainnet today, on the `exact` rail, without escrow infrastructure and without any party but the merchant able to sign a refund.

01 WHAT X402 SOLVES, PRECISELY

The x402 flow is short by design. A client requests a resource; the server returns `402` with the terms it accepts; the client signs a payment payload against one of those terms and resubmits; a facilitator verifies the payload and settles it on-chain; the server serves the resource. Four schemes govern how settlement itself behaves: `exact` transfers the stated amount immediately. `upto` transfers up to a client-authorized maximum, settled for actual usage. `batch-settlement` accepts a commitment now and settles it later, in bulk, through a channel, voucher store, or billing cycle, built for volume where per-request on-chain settlement does not make sense. `auth-capture` holds funds in escrow or sends them direct, with a capture-authorizer able to void, capture, or refund inside a deadline.

Every one of the four is a payment scheme: it governs how value moves for a single authorized amount. None defines a cart, an authorization spanning more than one priced line. None defines a checkout session, a stateful object that survives across a quote, a line-item selection, and a settlement. None binds an identity to the request beyond the wallet whose signature the scheme verifies. That is not a gap in the engineering; it is the correct scope for a payment protocol, the same way a card network's authorization message is not supposed to know what is in a shopping cart or who is buying, a merchant's own checkout stack sits on top of the rail for that.

Agent commerce needs that stack too, and it needs the identity to mean something specific: a KYA-equivalent credential the merchant's own server can verify, not just recover a wallet address from a signature.

02 THREE PRIMITIVES, LAYERED ON THE SETTLEMENT LEG

x402 Direct adds three things above x402's settlement leg. None require a change to the x402 spec; each is a layer a resource server and its client agree to speak before the `402` challenge fires.

1. CHECKOUT SESSION

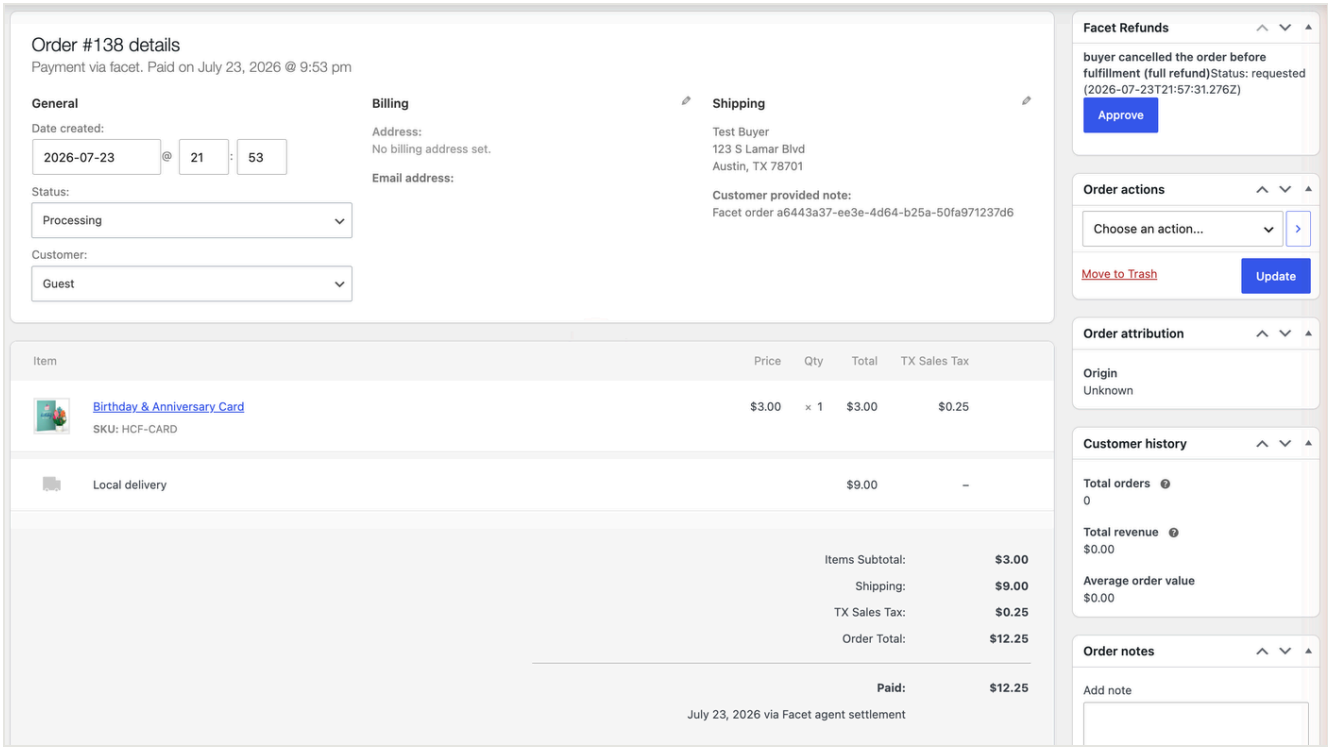
a stateful object, created and completed under RFC 9421 HTTP Message Signatures, that prices an arbitrary number of line items from a live commerce backend (goods, shipping, and jurisdiction-correct tax) before advertising the one `exact`-scheme payment requirement the client ultimately signs. The cart is negotiated above x402; the money still moves through it, once, for the summed total.

2. AGENT-FACING REFUND AND CANCELLATION INITIATION

one route, signed the same way the checkout was, that a buyer's agent calls to open a ticket against a specific, already-settled order. An optional line-item selection makes it a partial refund; omitting the selection makes it a full-order cancellation; the reason is a string the agent supplies and the merchant reviews, so the same primitive covers a pre-shipping change of mind and a post-delivery damage claim without a separate code path for either.

3. NON-CUSTODIAL REVERSE SETTLEMENT

the merchant's own wallet, the same account that received the original payment, signs the ERC-3009 send-back client-side. The protocol layer verifies the signature, checks it against the order it claims to settle, and relays it through a facilitator. It never holds a key that could sign one.



The second primitive, in a real merchant backend. An agent cancelled a settled order over the RFC 9421-signed route, supplying no line selection, so the server derives the entire captured total. It appears in the merchant's own WooCommerce admin as a pending decision with an Approve button. Nothing moves until a human acts, and when they do, the signature comes from the merchant's own wallet, not from Facet. The appended validation report carries the full sequence and the on-chain settlements.

> **Composition, not a fork.** x402 Direct does not modify the `exact` scheme, does not compete with `auth-capture`, and does not reimplement the checkout-session or agent-identity concepts that the Universal Commerce Protocol (UCP) already specifies. It is the specific arrangement of UCP checkout sessions, a KYA-equivalent identity credential, and a merchant-signed reverse leg, composed to sit above the `exact` scheme, the one scheme in the spec that has no reverse leg of its own to conflict with.

03 THE TECHNICAL SPECIFICATION

Every request below is drawn from the real, on-chain-settled run in the companion validation report appended after this paper; the shapes are exactly what shipped, not an illustrative simplification.

> **Create.** `POST /ucp/v1/checkout-sessions`, RFC 9421 signed. Body: a line-item array (`product_id`, `quantity`) and a fulfillment destination. Response: a session id, status `ready_for_complete`, and a `payment_handlers` object naming every rail the site has configured, the x402 handler carrying the server-priced `pay_to`, `amount_atomic`, network, and EIP-712 domain the client must sign against. The session advertises every rail it has configured and privileges none; a Boson-escrow-only site and an x402-only site both create successfully.

> **Complete.** `POST /ucp/v1/checkout-sessions/{id}/complete`, RFC 9421 signed, carrying the buyer's ERC-3009 `TransferWithAuthorization` for the exact amount and domain the create response advertised. Settlement moves the funds and creates the order in the merchant's own commerce backend, buyer wallet stamped on it, in the same call.

> **Refund or cancel, initiate.** `POST /ucp/v1/checkout-sessions/refund`, RFC 9421 signed, body `{order_id, reason, refund_line_items?}`. Authorization is bound to the order's real site id server-side, never to the caller's self-asserted identity, so a forged claim of ownership cannot open a ticket on an order it does not own. Moves no money; opens a ticket the merchant can review.

> **Refund or cancel, approve.** The merchant's own client signs an ERC-3009 send-back from its own `pay_to` to the order's stamped buyer address, for an amount the server derives from the merchant's own commerce order, never from the agent's request. The protocol layer checks signer equals `pay_to`, destination equals the stamped buyer, and amount equals the derived total, before relaying it to a facilitator.

04 WHERE THIS SITS RELATIVE TO WHAT ALREADY EXISTS

X402 AUTH-CAPTURE	X402 DIRECT
Refund is exercised by a capture-authorizer, a facilitator EOA or an escrow contract, inside a fixed capture and refund deadline. The amount is a scalar tied to one payment. There is no line-item selection, because there is no cart; there is no agent identity distinct from the signing wallet, because the scheme does not define one. Spec merged 2026-05-13. Its reference package ships the client only, with server and facilitator support deferred to a later change, so no settlement path exists in the reference implementation as of 2026-07-23.	Refund and cancellation are opened by an identified agent under a signature bound to a specific site and a specific already-settled order. The amount is derived line by line from the merchant's own commerce backend. Approval is a human merchant decision, signed by the merchant's own wallet, not an automated capture-authorizer acting inside a deadline.

The two are not competing implementations of the same idea. `auth-capture` is a payment-processor primitive: escrow, timers, a single authorizer role, useful for buyer protection on a payment that has no commerce system behind it. x402 Direct is a commerce-system primitive: it exists because there is a real order, a real merchant backend, and a real reason a human is reviewing. A site could use both, `auth-capture` for the payment guarantee, x402 Direct for the agent-facing request and the itemized reversal into its own books. The practical difference today is availability: a merchant who wants to give money back on x402 right now cannot wait for escrow settlement to land, and does not have to.

05 SECURITY MODEL

> **Non-custodial by construction, not by policy.** The only two keys that ever sign a money leg are the buyer's, at checkout, and the merchant's own, at refund. The protocol layer verifies signatures; it produces none.

> **Ownership is bound server-side.** A refund ticket is opened against the site the settled order actually belongs to, resolved from the order record, never from a claim the caller's request makes about itself.

> **The amount is never agent-set.** A line-item selection is advisory input, validated against what was actually ordered; the number that gets signed is read from the merchant's own commerce order at approval time.

> **A shared ledger caps every refund path together.** A partial refund, a second partial refund, and a full cancellation on the same order are checked against one running total, so no combination of agent-initiated requests can exceed what was captured.

> **Replay-proof.** Every ERC-3009 authorization carries a single-use nonce consumed on-chain; a facilitator independently re-verifies a signature before broadcasting it.

> **Reviewed adversarially before shipping.** The merchant-signed reverse-settlement path was reviewed across independent attack lenses before its mainnet run, each finding checked by independent review passes; the one real functional gap found (a dead end for a merchant with no managed signer configured) was fixed before merge, not disclosed as a residual risk.

06 WHY THIS BELONGS IN THE OPEN, NOT IN ONE COMPANY'S STACK

x402 Direct is deliberately not a fork, a competing payment scheme, or a reason to leave the spec. It is evidence for a specific, narrow claim: that `exact` plus a checkout-session layer plus an agent-facing initiation route plus a non-custodial reverse leg is enough to run real commerce, cart through cancellation, without a merchant ever handing custody of a refund key to anyone. That claim is falsifiable by exactly the same tools that verify the rest of x402: a transaction hash, a signature, a published key.

The open question worth the Foundation's attention is narrower than a whole new scheme: whether an identity credential distinct from the signing wallet, and a line-item-aware refund initiation bound to a real commerce order, belong as a documented extension pattern above `exact`, the way `auth-capture` documents a different, escrow-based answer to a related but distinct problem. This paper and the validation report it precedes are offered as the worked example.

On precedence, stated carefully because it is checkable. We are not claiming to have invented refunds on x402; the specification's own `auth-capture` scheme describes an escrow-based answer, and it was merged before this work was proven. What we can document is narrower and, we think, more useful: as of 2026-07-23, the reference implementation of that scheme ships a client with server and facilitator support explicitly deferred, and we are not aware of another agent-initiated refund and cancellation loop settling real value on x402 in production. The loop in the appended report did that on Base mainnet, non-custodially, on the `exact` rail. If someone else got there first, the same standard of proof should settle it: a transaction hash, a signature, a published key.

> WHAT THIS PAPER CLAIMS, AND WHAT BACKS IT
>
> Every architectural claim above is proven, not asserted, in the companion validation report: a real 2-item cart settled on Base mainnet through a signed checkout session, a partial refund by line selection, a post-delivery damaged-goods return, and a full-order agent-initiated cancellation, each signed by the merchant's own wallet, each independently verifiable on-chain.

07 REFERENCES

1. x402 Foundation. x402 protocol specification. `github.com/x402-foundation/x402`, `specs/`.
2. x402 Foundation. Scheme: `exact`. `specs/schemes/exact/`.
3. x402 Foundation. Scheme: `auth-capture`. `specs/schemes/auth-capture/scheme_auth_capture.md` and `scheme_auth_capture_evm.md`.
4. x402 Foundation. Scheme: `upto`. `specs/schemes/upto/scheme_upto.md`.
5. x402 Foundation. Scheme: `batch-settlement`. `specs/schemes/batch-settlement/scheme_batch_settlement.md`.
6. Base. commerce-payments contract stack (AuthCaptureEscrow). `github.com/base/commerce-payments`.
7. Universal Commerce Protocol. Specification and documentation. `github.com/Universal-Commerce-Protocol/ucp`.
8. Circle. USDC EIP-3009 `transferWithAuthorization` / `receiveWithAuthorization`.
9. IETF. RFC 9421, HTTP Message Signatures.
10. Facet, LLC. x402 Direct mainnet validation: UCP-signed checkout, merchant-approved refund, and agent-initiated cancellation on Base mainnet. Appended after this paper.