x402Interop: A Cross-Chain Interoperability Layer for the x402 Payment Economy

THE3RDWEBLABS
the3rdweblabs@gmail.com
github.com/the3rdweblabs

Concept Note, Version 1.0, November 2026

Abstract. x402 gives any web resource a native way to ask for payment and any client, human or autonomous, a native way to answer. Today that exchange only works when payer and seller share a chain and a token. A Base-native agent cannot pay a Sui-native API without first acquiring Sui assets, and a seller must operate on every chain it wishes to be paid on. We propose x402Interop, an extension layer that sits on top of x402 version 2 rather than replacing it. A single facilitator uses NEAR Intents to turn a payment made on the payer's chain, in the payer's token, into a payment received on the seller's chain, in the seller's token. The payer signs one transfer to a quote-bound deposit address; the swap and delivery are performed by the intents network; the facilitator never fronts liquidity and therefore holds no float risk. The design uses only mechanisms the x402 specification already defines: a new scheme, a protocol extension, the reserved payment-flow field and the non-terminal pending-settlement code. We describe the design, its failure modes, a Lightning path for Bitcoin, the current state of x402 scheme coverage on our target chains, and the open problems that remain.

1. Introduction

HTTP has carried a reserved but unused status code, 402 Payment Required, since its earliest revisions, and it remains reserved in the current specification [3]. The x402 protocol gives that code a working meaning: a server answers with 402 and a machine-readable list of acceptable payments, the client signs one of them and retries with the payment attached, and a facilitator verifies and settles it [2][4]. An API, a dataset or an inference endpoint can then be priced per request, with no accounts, keys or invoices.

This works well inside one ecosystem and poorly across ecosystems. Every x402 offer names a single network and a single asset. A buyer holding USDC on Base cannot answer an offer priced in USDC on Sui. Payers must hold, bridge or pre-position balances on every chain where the services they want live; sellers must pick one chain, or run infrastructure on several and fragment their revenue and audience. As agents begin to buy from one another at machine speed, this fragmentation becomes the main brake on an open x402 economy.

x402Interop removes that constraint with one rule: *neither the payer nor the seller should ever need a wallet on a chain they do not already use*. The payer pays from where their funds are; the seller is paid where they want to be paid; everything between is infrastructure. We build that infrastructure on NEAR Intents [5], an intent-based cross-chain swap network, and package it as an x402 extension so that existing sellers and clients can adopt it with minimal change. In the spirit of the original peer-to-peer cash proposal [1], the aim is to remove a trusted counterparty from the payment path rather than add one.

2. Background

2.1 x402 version 2

The reference repository, now maintained under the x402 Foundation, defines x402 as three components: types, scheme logic and transport representation [4]. A *resource server* answers an unpaid request with a `PaymentRequired` object, carried over HTTP in a base64-encoded `PAYMENT-REQUIRED` header. It holds a resource description, an `accepts` array and an optional `extensions` object. Each accepts entry names a scheme, a network in CAIP-2 format [11], an amount in atomic units, an asset, a recipient (`payTo`), a timeout and scheme-specific `extra` data. The client picks an entry and retries with a `PAYMENT-SIGNATURE` header holding a `PaymentPayload`: the entry it accepted, plus scheme-specific payload data. A *facilitator* exposes `/verify`, `/settle` and `/supported`, so sellers need not operate chain infrastructure.

Three properties of version 2 matter here. First, **extensions** are a keyed map; each value carries server-provided `info` and a JSON `schema` for it, clients echo them in the payload, and a client may append information but not delete or overwrite what the server supplied [4]. Second, a scheme can declare a **payment flow**. The default `authorization` flow verifies, runs the resource, then settles. The `upfront` flow settles before the resource runs, giving the server finality first. A non-default flow must be declared in `extra.paymentFlow`, and clients must not construct payments for flows they do not recognize [4]. Third, settlement has a **non-terminal pending state**: a facilitator may return `settlement_pending` with the broadcast transaction hash when confirmation cannot yet be established, so the caller can reconcile before retrying [4]. The EVM `exact` scheme (built on EIP-3009 [10]) also states a safety invariant we preserve: the facilitator cannot modify the amount or destination of a signed transfer [6].

2.2 NEAR Intents

NEAR Intents lets a user express a desired outcome, such as “deliver this token on that chain to this address”, and lets competing solvers fulfil it. Its 1Click API wraps the flow in four calls: request a swap quote, submit the deposit transaction, list supported tokens, and read swap execution status [5]. A quote returns a deposit address bound to that quote, a deadline, and refund instructions. When funds arrive, the swap executes and the destination asset is delivered to the recipient named in the quote; if execution fails, funds return to the stated refund address. Each property maps onto a need x402Interop has.

3. Design Goals

One wallet. A payer pays from a single native balance and a seller receives into a single native balance. Pre-positioned liquidity on the payer’s side was considered and rejected because it only moves the multi-wallet burden onto the payer.

No float. The facilitator must not pay the seller from its own funds and recover later from the payer. That creates inventory, pricing and insolvency risk and makes the facilitator a bank. In x402Interop the payer’s own funds are the only funds in motion.

Extension, not fork. x402Interop reuses x402 core types and packages, and layers on top through a scheme, an extension and the existing flow fields. A client or seller that ignores Interop continues to work exactly as before.

Minimal seller adoption. The target is that an existing seller changes its facilitator URL and, if the resource-server SDK requires it, registers one scheme and one extension. The facilitator continues to settle same-chain payments normally, so one endpoint presents native and Interop choices together.

Open facilitation. We operate the first facilitator, but the protocol must let others run compatible ones that can be listed, discovered and verified.

4. Protocol

4.1 Overview

A seller prices a resource in its native asset, for example USDC on Sui, and lists that as an ordinary `exact` entry. Through the Interop scheme and extension it also advertises that it can be paid from other chains. A buyer whose balance is elsewhere picks the Interop entry. The buyer plugin asks the facilitator for a quote converting the buyer's origin asset into the seller's destination asset, with the seller's address as recipient and the buyer's address as refund target. The facilitator obtains the quote from NEAR Intents and returns the origin amount and a deposit address. The buyer signs an origin-chain payment to that address using the origin chain's ordinary x402 mechanism. The seller forwards the payload to the facilitator, which broadcasts the origin transfer, reports it to NEAR Intents and tracks the swap to a terminal state. The seller then serves the resource. Figure 1 shows the sequence.

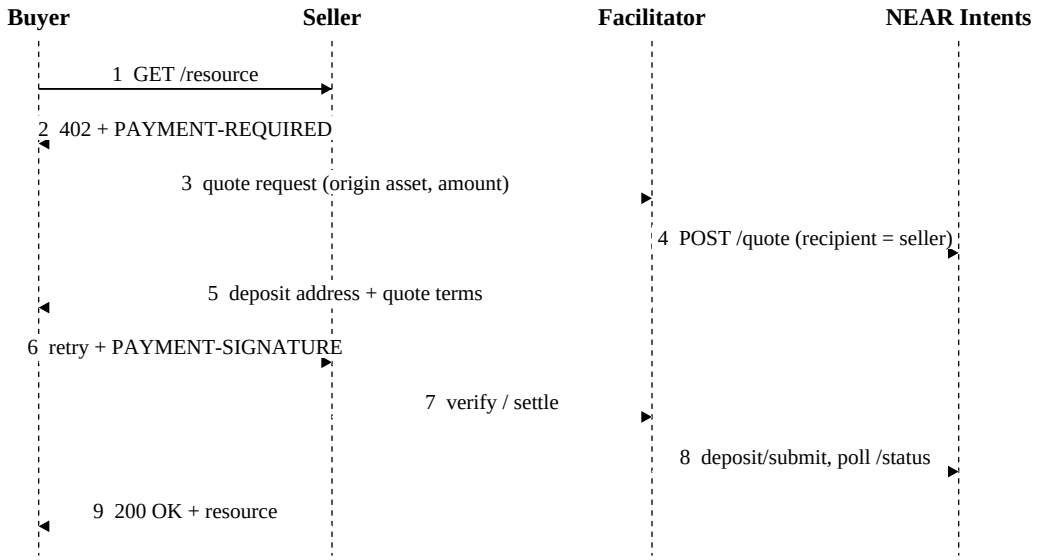


Figure 1. Message flow for a cross-chain x402Interop payment.

4.2 Scheme, extension and payload

The seller's Interop entry uses a new scheme identifier. Its `network`, `asset`, `amount` and `payTo` describe what the seller receives, on the seller's chain. The accepted origin networks are advertised under an `interop` key in `extensions`. Because settlement is slow relative to a same-chain transfer, the entry declares `extra.paymentFlow` explicitly. An illustrative `PaymentRequired` fragment follows; identifiers are placeholders and will be aligned with the specification as the scheme is formalized.

```
{
  "x402Version": 2,
  "accepts": [{
    "scheme": "interop",
    "network": "<destination, CAIP-2>",
    "amount": "250000",
    "asset": "<destination token>",
    "payTo": "<seller address>",
    "maxTimeoutSeconds": 300,
    "extra": { "paymentFlow": "upfront" }
  }],
  "extensions": {
    "interop": {
      "info": {
        "facilitator": "https://<interop-facilitator>",
        "quote": "/interop/quote",
        "origins": ["<CAIP-2>", "<CAIP-2>"]
      },
      "schema": { ... }
    }
  }
}
```

The buyer's payload echoes the unchanged entry in `accepted`, so core matching rules still hold. Inside the scheme-specific `payload` it carries the quote identifier, the origin network and asset, the deposit address, and an inner payload that is an ordinary origin-chain x402 payment whose recipient is that deposit address. The buyer plugin appends the quote data to the echoed extension info, which the version 2 extension rules permit provided the server's original info is left intact.

The longer-term aim is a standard Interop scheme per chain family, accepted upstream, describing how to pay or receive across chains not native to the service. Until then the facilitator ships its own scheme implementations and a buyer plugin that registers them into an existing client.

4.3 The Interop facilitator

The facilitator exposes the standard `/verify`, `/settle` and `/supported` endpoints, listing the Interop scheme and extension in `/supported`, plus one non-standard quote endpoint. It is a thin coordinator over the 1Click API: it turns a buyer's request into a quote, validates payments against that quote, submits the deposit hash, and polls swap status until a terminal state. Same-chain requests bypass the swap and settle as in plain x402. The implementation is TypeScript throughout, built on the 1Click TypeScript SDK and the x402 core packages.

Verification binds four things: the quote identifier, the seller's resource and recipient, the buyer's origin address, and a validity window that must not outlive the quote deadline. Because the origin transfer is signed to the deposit address, the facilitator can broadcast it but cannot redirect it, preserving the invariant stated for the EVM scheme [6]. A payment that fails any binding is rejected before funds move. For slow swaps the facilitator returns the non-terminal pending code with the origin transaction hash, never an empty hash, so the seller can reconcile rather than guess.

4.4 The buyer plugin

Existing x402 clients should not have to be rewritten. In version 2 a client registers schemes explicitly, so the plugin is a single registration call that adds the Interop scheme to an existing client. When the seller offers no option on a chain where the buyer holds funds but does offer an Interop entry, the plugin requests a quote, constructs the origin payment to the returned deposit address, and hands it back as a normal payload. The plugin must refuse an entry whose payment flow it does not recognize, as the specification requires [4].

4.5 Discovery

Discovery already has a home in x402: the Bazaar extension and the facilitator discovery endpoints list resources and can filter by network, recipient, scheme and extension key [4]. x402Interop does not build a rival catalogue. It publishes Interop-capable resources through Bazaar and adds a cross-chain view, so agents anchored on NEAR can find services anchored elsewhere and the reverse. The index is advisory; a seller's own 402 response remains the source of truth for price and recipient.

5. Settlement and Fund Safety

The central safety property is that *funds only ever travel along a path that ends either in the seller's wallet or back in the buyer's wallet*. The buyer's transfer goes directly to a deposit address that NEAR Intents has bound to a specific recipient and refund address. The facilitator never custodies the principal and never advances funds to the seller. If the swap cannot complete, the quote's refund path returns the origin asset to the buyer; the seller, having not seen a completed settlement, does not release the resource. The facilitator bears only the gas cost of broadcasting the origin transfer, as in ordinary x402.

Sellers choose when to release a resource. The x402 flow model gives two natural settings, and a third is a scheme-level refinement:

Policy	Resource released when	Seller exposure
Strict (upfront)	Settlement completes: swap reports success and delivery on the destination chain	None; highest latency
Deposit-final	Origin transfer is final and the swap is accepted for execution	Swap failure after release; buyer is refunded, seller is not paid
Tiered	Strict above a value threshold, Deposit-final below it	Bounded by threshold

A tiered policy suits metered, low-value calls where waiting for full cross-chain delivery would cost more in latency than the amount at risk. These policies move risk to the seller explicitly and visibly, rather than hiding it inside the facilitator's balance sheet. Since clients are told to prefer the default authorization flow when it is offered [4], a seller that lists both native and Interop options naturally steers same-chain payers to the faster path.

6. Bitcoin and the Lightning Network

Bitcoin is a first-class day-one asset but poorly suited to HTTP-speed settlement on its base layer. For BTC payers we plan a Lightning leg [7]: the payer pays in BTC over Lightning and the seller receives a supported token such as USDC. We intend to carry the request-and-pay exchange with L402 [8], which already pairs an HTTP 402 challenge with a Lightning invoice, provided it interoperates cleanly with the NEAR Intents flow.

This is the least settled part of the design. A Lightning payment completes in seconds, while an on-chain deposit into an intent does not, and bridging the two without the facilitator fronting funds is the open problem. Hold invoices, whose preimage is released only after delivery is confirmed, are the most promising direction, since the payer's funds stay locked rather than lent. We treat Lightning as an experimental track, not a launch dependency.

7. Network Coverage and the Scheme Gap

The first release targets the chains NEAR Intents supports that matter most for payments. The exact token matrix is read at runtime from the 1Click token listing rather than hardcoded, since listings change.

Family	Chains

EVM	Arbitrum, Base, Berachain, BNB Chain, Ethereum, Optimism, Plasma, Polygon, Avalanche, Monad, X Layer, Scroll
Bitcoin and forks	Bitcoin, Bitcoin Cash, Litecoin, Dash, Zcash, Dogecoin
Other layer-1s	Aptos, NEAR, Solana, Stellar, Sui, TON, Tron, XRP, Starknet

Table 1. Day-one chains.

Because each origin payment reuses a chain's native x402 mechanism, coverage depends on which chains already have an `exact` scheme. Our review of the specification, undertaken for this version, found the picture in Table 2. It will change quickly and must be re-checked at each release.

Status of exact scheme	**Chains**
Specified in the reference repository	EVM chains (EIP-3009 and Permit2 transfer methods), Solana, Sui, Stellar, TON, Starknet; XRP Ledger bindings are referenced in the core specification
Proposed upstream or third-party only	Tron (specification proposed); NEAR and Aptos (third-party implementations found; official status to confirm)
No x402 scheme	Bitcoin, Bitcoin Cash, Litecoin, Dash, Zcash, Dogecoin

Table 2. x402 scheme coverage of day-one chains, as reviewed in November 2026.

For chains in the second group, the facilitator can ship the missing mechanism itself; for the UTXO chains in the third, the origin leg is a signed transaction paying the deposit address, which the facilitator broadcasts. Stablecoins (USDC, USDT, USDT0) are the primary settlement assets for sellers; native and wrapped assets are accepted from payers where the intents network lists them.

8. Latency

An HTTP client will not wait indefinitely, and a cross-chain swap, however fast, takes longer than a same-chain transfer. We address this in three ways. Quotes are requested before payment, in parallel with other client work, so the signed payment is the only step on the critical path. Sellers select a release policy from Section 5 to match the value and urgency of each resource. And for slower paths the seller treats the facilitator's pending-settlement response as a receipt to reconcile against, rather than holding a connection open. Measured latency distributions are among the first deliverables of the reference implementation.

9. Security Considerations

Facilitator trust. A malicious facilitator could return a deposit address it controls. The buyer plugin therefore checks that the quote names the seller's `payTo` as recipient and the buyer as refund target, and sellers may independently confirm receipt on the destination chain. As an option, the plugin may request the quote from NEAR Intents directly, leaving the facilitator to validate only, at the cost of handling that service's access requirements on the client side. Sellers choose which facilitators they accept, and a public registry of third-party facilitators will carry verification instructions.

Quote manipulation and replay. Quotes are single-use, bound to a resource and payer, and expire. Signed payments that outlive their quote are invalid, and the origin chain's own nonce rules prevent replay of the transfer.

Slippage and price. Sellers price in their native asset and request exact-output quotes where the network supports them, so the seller is paid the listed amount and slippage falls on the payer, who sees the origin amount before signing.

Dependency risk. The design depends on NEAR Intents liveness and solver competition. If the network is unavailable, same-chain x402 continues to work and Interop entries are simply withdrawn from the 402 response.

10. Implementation and Roadmap

The reference implementation is TypeScript throughout, on the x402 core packages and the 1Click SDK. Work proceeds in order: (1) confirm in the resource-server SDKs whether a facilitator can contribute scheme entries and extensions on a seller's behalf, which decides whether adoption is a pure URL change or a one-line registration; (2) implement the facilitator with the EVM and Sui schemes and a two-chain demonstration, an EVM buyer paying a Sui seller; (3) release the buyer plugin and a drop-in seller guide; (4) add the remaining chain families, supplying schemes where none exist upstream; (5) prototype the Lightning leg; and (6) propose the Interop scheme upstream and publish the facilitator registry specification. The project builds on our earlier payment tooling for Sui [9], whose checkout flow informed the seller and buyer ergonomics described above.

11. Conclusion

x402 gives the web a native way to charge per request, but only within one chain at a time. x402Interop removes that boundary without asking sellers to change their business or buyers to change their wallets. By delegating cross-chain execution to an intents network, binding every payment to a quote, building only from mechanisms x402 version 2 already defines, and refusing to hold float, it keeps the facilitator a coordinator rather than a counterparty. If the open problems in Sections 5, 6 and 8 resolve as we expect, any x402 service will be payable from any supported chain, in any supported token, with one signature.

References

[1] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008. https://bitcoin.org/bitcoin.pdf

[2] x402 Foundation, "x402: A payments protocol for the internet. Built on HTTP." https://github.com/x402-foundation/x402 and https://www.x402.org

[3] R. Fielding, M. Nottingham, J. Reschke (eds.), "HTTP Semantics," RFC 9110, IETF, June 2022.

[4] x402 Foundation, "x402 Protocol Specification, version 2." https://github.com/x402-foundation/x402/blob/main/specs/x402-specification-v2.md

[5] NEAR Intents documentation, including the 1Click API. https://near-intents.org

[6] x402 Foundation, "Scheme: exact on EVM." https://github.com/x402-foundation/x402/blob/main/specs/schemes/exact/scheme_exact_evm.md; and the per-network documents indexed in scheme_exact.md in the same directory.

[7] J. Poon and T. Dryja, "The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments," 2016. https://lightning.network/lightning-network-paper.pdf

[8] Lightning Labs, "L402: Lightning HTTP 402 protocol specification." https://github.com/lightninglabs/L402

[9] THE3RDWEBLABS, "SuiOutKit: drop-in checkout SDK for Sui." https://suioutkit.xyz

[10] P. J. Kim, K. Britz, D. Knott, "EIP-3009: Transfer With Authorization," Ethereum Improvement Proposals, 2020.

[11] Chain Agnostic Standards Alliance, "CAIP-2: Blockchain ID Specification." https://chainagnostic.org/CAIPs/caip-2