derien × x402 — Resolution Scheme v0.1

Status: Draft design proposal — September 2026
Audience: x402 implementers, agent/wallet developers, facilitators, programmable-money issuers, settlement-network operators
Protocol relationship: Proposed x402 V2 payment scheme. This document is **not** an adopted x402 Foundation specification.

---

1. Abstract

x402 standardizes how a resource server states a payment requirement and how a client provides a signed payment authorization. derien adds a population-resolution step between authorization and settlement.

Under the proposed `derien` scheme, a seller still specifies the exact asset, amount and destination it must receive. A buyer may satisfy that obligation using a different approved machine-native monetary asset. The buyer signs a bounded authorization over that funding asset. Compatible obligations are admitted into a live resolution population. derien resolves them continuously and deterministically across multiple participants and assets. A programmable settlement contract then executes the resulting transfers atomically.

The scheme does **not** require derien to custody assets, take a principal position, quote a bid/offer spread, bridge between chains, or replace x402. x402 remains the request, authorization and response envelope. The underlying blockchain or programmable ledger remains the settlement system. derien is the resolution layer between them.

The intended stack is:

```
Agent / application
        ↓
       x402
payment requirement + signed authorization
        ↓
      derien
continuous population resolution
        ↓
programmable settlement adapter / contract
        ↓
 x402 network (Base, other EVM, etc.)
```

---

2. Design objective

The objective is to let an x402 payment requirement become an **authorized monetary intention** before it becomes an individual transfer.

Normal x402 `exact` semantics are approximately:

```
Seller requires 91 EURx
        ↓
Buyer signs transfer of 91 EURx to seller
        ↓
Facilitator settles transfer
```

The proposed derien semantics are:

```
Seller requires 91 EURx
        ↓
Buyer authorizes up to 100 USDx to satisfy that requirement
        ↓
Compatible requirements enter derien live state
        ↓
A multi-party, multi-asset closure is found
        ↓
Seller receives exactly 91 EURx
Buyer delivers the resolved amount of USDx
        ↓
All legs settle atomically
```

No currency is literally transformed. Ownership of existing machine-native assets changes around a valid multilateral closure.

---

3. x402 compatibility basis

x402 V2 defines three core wire objects for HTTP:

- `PaymentRequired`, carried in `PAYMENT-REQUIRED`
- `PaymentPayload`, carried in `PAYMENT-SIGNATURE`
- `SettlementResponse`, carried in `PAYMENT-RESPONSE`

Each payment option in `PaymentRequired.accepts` identifies a `scheme`, `network`, `amount`, `asset`, `payTo`, timeout and optional scheme-specific `extra` data. x402 payment schemes define how the `payload` is constructed, verified and settled. x402 V2 uses CAIP-2 network identifiers and explicitly supports modular payment schemes.

The proposed derien integration therefore uses x402 as designed: `derien` is a **payment scheme**, not a competing HTTP protocol.

3.1 Proposed scheme identifier

```
scheme = "derien"
```

This identifier is provisional until reviewed through the x402 Foundation process.

---

4. v0.1 scope

v0.1 deliberately constrains the problem.

In scope

- x402 V2
- one settlement network per resolution domain
- EVM mechanism binding first
- Base Sepolia as the reference test deployment
- multiple fungible machine-native monetary assets on the same network
- exact seller-side obligations
- buyer-selected funding asset from an approved domain set
- short-lived, signed funding authorizations
- continuous deterministic derien resolution
- atomic multi-asset settlement in one network transaction
- `exact` x402 payment as an optional fallback advertised by the seller

Out of scope

- cross-chain atomic settlement
- bridges
- unsecured credit
- derien-issued money
- derien custody
- principal inventory
- market making
- discretionary FX pricing
- partial satisfaction of a single x402 resource payment
- long-dated payment promises

A future version may support additional x402 networks and settlement-domain types. Cross-chain resolution must not be added until atomicity and failure semantics are defined independently of bridges or principal credit.

---

5. Resolution domain

A **derien resolution domain** is the population of payment intentions that can be settled atomically on one programmable settlement network.

For v0.1:

```
one derien domain = one x402 CAIP-2 network + one settlement contract + one rate policy
```

Example:

```
domain: derien:eip155:84532:v1
network: eip155:84532          # Base Sepolia
settlementContract: 0x...
```

All assets used in one resolution MUST be transferable by the domain settlement mechanism on that network.

The x402 `network` is therefore the shared settlement ledger for v0.1. derien is not itself the ledger.

---

6. Payment requirement mapping

The seller expresses its economic requirement using the standard x402 `PaymentRequirements` fields.

Example:

```
        {
          "scheme": "derien",
          "network": "eip155:84532",
          "amount": "91000000",
          "asset": "0xEURxToken",
          "payTo": "0xSeller",
          "maxTimeoutSeconds": 2,
          "extra": {
            "resolutionDomain": "derien:eip155:84532:v1",
            "settlementContract": "0xDerienSettlement",
            "resolverUrl": "https://resolver.example/v1",
            "ratePolicy": "mid-v1",
            "intentTtlMs": 1000,
            "fundingAssets": [
              "0xUSDxToken",
              "0xEURxToken",
              "0xGBPxToken",
              "0xJPYxToken"
            ],
            "partialSettlement": false
          }
        }
```

6.1 Meaning of the standard x402 fields

For the `derien` scheme:

- `amount` — the exact quantity the seller must receive, in atomic units
- `asset` — the exact machine-native monetary asset the seller must receive
- `payTo` — the seller/beneficiary address
- `network` — the common settlement network for this resolution domain
- `maxTimeoutSeconds` — the maximum time the x402 request may remain unresolved

The derien scheme MUST NOT reinterpret the seller's `amount`, `asset`, or `payTo` as suggestions. They are settlement invariants.

6.2 Scheme-specific `extra`

`resolutionDomain`
Unique identifier for the derien population into which the authorization may be admitted.

`settlementContract`
Contract or settlement adapter authorized to execute resolved transfers.

`resolverUrl`
Reference endpoint for submitting and querying intents in the reference deployment. This is transport/discovery metadata, not a requirement that all derien implementations use HTTP.

`ratePolicy`
Identifier for the domain's common reference-rate policy.

`intentTtlMs`

Maximum lifetime of an unresolved intent within the derien population. It MUST NOT exceed the x402 payment timeout.

`fundingAssets`
Assets the seller/domain permits buyers to use as funding assets for this requirement.

`partialSettlement`
MUST be `false` in v0.1. An x402 resource payment is either satisfied in full or not satisfied.

---

7. Buyer authorization payload

The buyer selects the `derien` payment option and chooses one permitted funding asset available in its wallet.

The buyer does **not** sign a transfer to the seller. It signs a bounded authorization to the derien settlement mechanism, cryptographically bound to the seller's original x402 requirement.

Proposed `PaymentPayload.payload`:

```
        {
          "intentId": "0x...",
          "payer": "0xBuyer",
          "fundingAsset": "0xUSDxToken",
          "maxDebit": "100000000",
          "validAfter": "1788350400",
          "validBefore": "1788350402",
          "nonce": "0x...",
          "requirementHash": "0x...",
          "resolutionDomain": "derien:eip155:84532:v1",
          "settlementContract": "0xDerienSettlement",
          "signature": "0x..."
        }
```

7.1 `requirementHash`

`requirementHash` MUST bind at least:

```
x402Version
resource.url
scheme
network
seller amount
seller asset
payTo
resolutionDomain
settlementContract
expiry / timeout
```

This prevents a resolver or facilitator from redirecting the authorization to a different seller, asset, amount or resource.

7.2 `maxDebit`

`maxDebit` is the maximum amount of the buyer's selected funding asset that may be taken to satisfy the bound payment requirement.

It provides the buyer's economic constraint. The settlement mechanism MUST reject any resolution that debits more than `maxDebit`.

7.3 Signature

The reference EVM mechanism SHOULD use EIP-712 typed data. The exact typed structure will be defined in the EVM binding document.

Conceptually:

```
DerienAuthorization(
  intentId,
  payer,
  fundingAsset,
  maxDebit,
  requirementHash,
  resolutionDomain,
  settlementContract,
  validAfter,
  validBefore,
  nonce
)
```

The authorization MUST be replay-protected and one-time consumable.

---

8. Verification

An x402 facilitator or resource server supporting the `derien` scheme verifies the payment authorization before admitting it to the resolver.

Verification MUST confirm:

1. x402 version, scheme and network match the selected `PaymentRequirements`.
2. `requirementHash` correctly binds the seller's requirement.
3. Buyer signature is valid.
4. Authorization is within its validity window.
5. Nonce has not already been consumed.
6. `fundingAsset` is permitted by the requirement/domain.
7. `maxDebit` is positive and within configured limits.
8. Buyer has sufficient spendable balance/allowance for the settlement mechanism, or another mechanism-specific guarantee of debit capacity.
9. `settlementContract` and `resolutionDomain` match trusted domain configuration.

A valid authorization is **not yet a settled payment**. It is an executable monetary intention eligible for derien resolution.

---

9. Canonical derien intention

The x402 adapter converts the verified payment into the internal canonical object:

```
        {
          "intentId": "0x...",
          "payer": "0xBuyer",
          "beneficiary": "0xSeller",
          "deliver": {
            "asset": "0xUSDxToken",
            "maxAmount": "100000000"
          },
          "satisfy": {
            "asset": "0xEURxToken",
            "exactAmount": "91000000"
          },
          "requirementHash": "0x...",
          "domain": "derien:eip155:84532:v1",
          "validBefore": "1788350402",
          "sequence": "..."
        }
```

This is the boundary between x402 and the derien core.

The derien core does not need to understand HTTP resources. It needs only an authenticated obligation, funding capacity, constraints, asset identifiers, expiry and deterministic ordering information.

---

10. Rate policy

The seller's required asset amount is fixed by x402. The buyer's debit amount must be calculated under the domain's common reference-rate policy and remain at or below `maxDebit`.

A resolution therefore references a signed or otherwise verifiable **Rate Snapshot**:

```
{
  "snapshotId": "0x...",
  "policy": "mid-v1",
  "timestamp": 1788350400123,
  "rates": {
    "USDx/EURx": "...",
    "EURx/GBPx": "...",
    "GBPx/JPYx": "..."
  },
  "signature": "0x..."
}
```

The protocol does not prescribe a commercial rate source in v0.1. A domain MUST declare its rate policy and verification method.

A compliant derien resolver MUST NOT add a bid/offer spread. Any permitted rounding rule MUST be deterministic and documented by the domain.

---

11. Resolution semantics

The resolver maintains a live ordered population of valid intents.

When a compatible set can satisfy all participating seller-side obligations under the rate snapshot and buyer-side debit constraints, the resolver emits one `Resolution`.

Example population:

```
A must satisfy 91 EURx, funds USDx
B must satisfy 78 GBPx, funds EURx
C must satisfy 15,000 JPYx, funds GBPx
D must satisfy 100 USDx, funds JPYx
```

A valid closure may produce:

```
A's USDx  → beneficiary of D's USDx requirement
D's JPYx  → beneficiary of C's JPYx requirement
C's GBPx  → beneficiary of B's GBPx requirement
B's EURx  → beneficiary of A's EURx requirement
```

No participant needs to hold the currency required by its own seller.

11.1 Determinism

A domain MUST define sufficient ordering and rate-snapshot rules such that the same admitted state produces the same resolution.

The reference derien implementation uses the existing deterministic continuous resolver. The open protocol

specifies observable behavior and conformance requirements; it does not require other compliant implementations to use the same internal algorithm.

11.2 Full satisfaction

For v0.1, every x402 payment included in a resolution MUST be satisfied in full.

If an intent cannot be fully satisfied before expiry, it expires unresolved.

11.3 No principal position

The resolver and settlement contract MUST NOT supply inventory to complete a closure. Every credited asset unit must be sourced from a valid participating debit of the same asset within the settlement transaction.

---

12. Resolution object

Proposed canonical form:

```
{
  "resolutionId": "0x...",
  "domain": "derien:eip155:84532:v1",
  "inputStateHash": "0x...",
  "rateSnapshotId": "0x...",
  "intents": ["0xA", "0xB", "0xC", "0xD"],
  "transfers": [
    {
      "asset": "0xUSDxToken",
      "from": "0xA",
      "to": "0xSellerD",
      "amount": "100000000"
    },
    {
      "asset": "0xEURxToken",
      "from": "0xB",
      "to": "0xSellerA",
      "amount": "91000000"
    },
    {
      "asset": "0xGBPxToken",
      "from": "0xC",
      "to": "0xSellerB",
      "amount": "78000000"
    },
    {
      "asset": "0xJPYxToken",
      "from": "0xD",
      "to": "0xSellerC",
      "amount": "15000000000"
    }
  ],
  "resolverSignature": "0x..."
}
```

A resolution is indivisible. Its transfers are not independent payments.

---

13. Settlement

13.1 v0.1 atomicity rule

All transfers in a resolution MUST settle atomically on the domain network:

```
all transfers commit
       OR
no transfers commit
```

The reference EVM mechanism uses a `DerienSettlement` contract that validates the resolution and executes all token debits/credits in a single transaction.

13.2 Conservation

For every asset in a resolution:

```
sum(debits(asset)) = sum(credits(asset))
```

The settlement contract MUST reject any resolution that violates asset conservation.

13.3 Authorization safety

For every participating intent, settlement MUST verify:

- authorization signature
- unconsumed nonce
- validity window
- funding asset
- actual debit ≤ `maxDebit`
- bound seller requirement
- seller receives exactly the required asset and amount
- rate snapshot/policy validity where required

13.4 No custody requirement

The preferred mechanism is atomic pull settlement using token allowance / permit-style authorization so participant assets remain in their wallets until the resolution transaction executes.

A production EVM binding must specify the safest practical authorization mechanism, potentially using Permit2 or token-native authorization methods. The core scheme does not require derien to take custody.

---

14. x402 request lifecycle

The v0.1 synchronous flow is:

```
1. Client → Server
   GET /resource

2. Server → Client
   402 Payment Required
   accepts: [derien, exact]

3. Client
   selects derien
   selects funding asset
   signs bounded derien authorization

4. Client → Server
   retries request with PAYMENT-SIGNATURE

5. Server / Facilitator
   verifies authorization
   submits canonical intent to derien

6. derien
   holds intent in live population for ≤ intentTtlMs
   resolves compatible population continuously

7a. If resolution succeeds
    settlement contract executes atomic multi-asset resolution
    facilitator returns successful SettlementResponse
    server returns resource

7b. If no resolution before expiry
    derien returns unresolved/expired
    server returns payment failure
    client MAY retry using another advertised x402 scheme, e.g. exact
```

14.1 Why synchronous first

Serving the resource before resolution would create seller settlement risk unless a separate guarantee/escrow architecture exists. v0.1 avoids that problem: the resource is served only after atomic settlement succeeds.

A later version may define a deferred-resolution model analogous to x402 batch settlement, but only if seller claims are fully collateralized or otherwise guaranteed without introducing principal FX risk.

---

15. Settlement response mapping

On successful resolution, the server returns the standard x402 `SettlementResponse` / `PAYMENT-RESPONSE`.

Example:

```
        {
          "success": true,
          "payer": "0xBuyer",
          "transaction": "0xMultilateralResolutionTxHash",
          "network": "eip155:84532"
        }
```

The transaction hash points to the atomic multilateral settlement transaction, not a dedicated bilateral payment transaction.

Optional scheme-specific response metadata may include:

```
        {
          "resolutionId": "0x...",
          "requirementHash": "0x...",
          "settledAsset": "0xEURxToken",
          "settledAmount": "91000000",
          "fundingAsset": "0xUSDxToken",
          "fundingDebit": "100000000"
        }
```

Any such metadata SHOULD be added in an x402-compatible scheme field or extension without altering the core `SettlementResponse` semantics.

---

16. Failure modes

A derien payment may fail for:

- invalid signature
- expired authorization
- unsupported funding asset
- insufficient balance or allowance
- `maxDebit` too low under the current rate snapshot
- no compatible closure before expiry
- settlement transaction revert
- stale/invalid rate snapshot
- replayed nonce
- domain mismatch
- settlement-contract mismatch

Recommended scheme-specific reason codes:

```
derien_invalid_authorization
derien_unsupported_funding_asset
derien_insufficient_capacity
derien_no_resolution
derien_intent_expired
derien_rate_constraint
derien_settlement_reverted
derien_domain_mismatch
```

No-resolution is a normal outcome, not a protocol fault.

---

17. Fallback behavior

A seller SHOULD initially advertise a conventional x402 payment option alongside derien:

```
"accepts": [
  {
    "scheme": "derien",
    "network": "eip155:84532",
    "amount": "91000000",
    "asset": "0xEURxToken",
    "payTo": "0xSeller",
    "maxTimeoutSeconds": 2,
    "extra": { "...": "..." }
  },
  {
    "scheme": "exact",
    "network": "eip155:84532",
    "amount": "91000000",
    "asset": "0xEURxToken",
    "payTo": "0xSeller",
    "maxTimeoutSeconds": 60
  }
]
```

This makes adoption non-disruptive:

- agents with the required asset can pay normally
- agents that prefer population resolution can choose derien
- if no closure is available quickly enough, the client can fall back to `exact`

The seller does not need to migrate its application economics to use derien.

---

18. Security model

18.1 Resolver authority

The resolver MAY select counterparties and closure topology but MUST NOT be able to:

- change the seller's required asset
- change the seller's required amount
- change `payTo`
- debit an unapproved funding asset
- debit more than `maxDebit`
- settle after authorization expiry
- reuse an authorization
- create asset units not supplied by participating debits

These invariants SHOULD be enforced by the settlement mechanism, not merely by resolver policy.

18.2 Facilitator trust

The facilitator is responsible for verification and settlement submission in the x402 model. A derien-capable facilitator additionally interfaces with the resolver.

The design SHOULD preserve x402's principle that a facilitator cannot alter a buyer's signed economic authorization without invalidating it.

18.3 Denial-of-service controls

`Open to build on` does not mean unmetered anonymous state mutation. A resolution-domain operator MAY require authentication, stake, technical quotas or rate limits to protect shared state, provided such controls are separate from the protocol's no-spread/no-resolution-fee economic model.

---

19. Privacy

v0.1 does not define a privacy protocol.

A resolver minimally needs enough information to validate:

- funding asset and capacity
- required settlement asset and amount
- authorization constraints
- expiry and ordering

Future deployments may use pseudonymous wallet identifiers, encrypted intent submission, secure enclaves, zero-knowledge proofs or privacy-preserving shared-ledger techniques. None are required for x402 compatibility and none should be embedded into the core scheme prematurely.

---

20. Economic properties

The `derien` scheme is designed so that derien:

- takes no principal position
- adds no bid/offer spread
- does not require inventory
- does not charge for protocol access or resolution under the proposed derien public model

This does not imply that the underlying network, token issuer, wallet, facilitator, RPC provider or other infrastructure is fee-free.

The rate policy determines the reference valuation used by the resolver. The settlement contract enforces participant bounds and seller satisfaction.

---

21. Reference implementation split

The project should publish four separate components.

21.1 `derien-spec`

Open specification containing:

- canonical intention schema
- x402 mapping
- resolution object
- rate snapshot object
- deterministic ordering requirements
- settlement invariants
- conformance vectors

21.2 `derien-x402`

Open SDK adapters for:

- x402 client scheme registration
- x402 resource-server scheme registration
- facilitator verification/settlement integration
- translation between x402 objects and canonical derien intents

21.3 `derien-settlement-evm`

Open reference settlement contract for a single EVM resolution domain, plus tests.

21.4 `derien-engine`

The production high-performance resolver implementation.

The protocol must be implementable without access to proprietary engine internals. Conformance is defined by observable deterministic results and settlement invariants, not by mandating implied-N as an algorithm.

---

22. Reference deployment

The first public demonstration SHOULD use:

```
Network: Base Sepolia (eip155:84532)
Assets: four test ERC-20 monetary tokens
        USDx / EURx / GBPx / JPYx
x402:   V2 seller + client
Scheme: derien
Resolver: derien reference/production engine adapter
Settlement: DerienSettlement EVM contract
Fallback: x402 exact
```

Demo sequence:

```
Agent A requests paid resource priced in EURx; A holds USDx
Agent B requests paid resource priced in GBPx; B holds EURx
Agent C requests paid resource priced in JPYx; C holds GBPx
Agent D requests paid resource priced in USDx; D holds JPYx

All four sign derien authorizations.

The resolver closes the population.

One atomic transaction moves:
USDx, EURx, GBPx and JPYx to the four respective sellers.

Each x402 request receives a successful PAYMENT-RESPONSE and its resource.
```

This demonstrates the entire architecture without requiring a new blockchain, bridge, custodian or principal market maker.

---

23. Conformance requirements

A v0.1 implementation is derien-compatible only if it satisfies all of the following:

1. It accepts standard x402 V2 payment envelopes.
2. It preserves seller `amount`, `asset` and `payTo` exactly.
3. It cryptographically binds buyer authorization to the seller requirement.
4. It enforces buyer funding-asset and maximum-debit constraints.
5. It uses a declared common rate policy without adding a derien spread.
6. It resolves only from admitted authorized intents.
7. It does not introduce principal inventory.
8. It preserves asset conservation per resolution.
9. It settles all legs atomically within one v0.1 resolution domain.
10. It provides replay protection.
11. It produces deterministic results under the domain's declared state-ordering rules.
12. It maps successful settlement back to a valid x402 `SettlementResponse`.

---

24. Open questions for v0.2

The following should remain explicitly open until the reference flow works:

- exact EIP-712 authorization structure
- Permit2 vs token-native authorization for the EVM binding
- canonical asset identifier beyond the x402 network-local `asset` field
- rate oracle / signed snapshot mechanism
- deterministic timestamp and sequencing model across multiple facilitators
- resolver discovery and failover
- multi-resolver consensus, if ever required
- privacy model
- deferred settlement / batch-compatible derien mode
- settlement-domain governance
- non-EVM bindings
- cross-chain atomicity

None of these is required to prove the v0.1 proposition.

---

25. The proposition in one sentence

> ***x402 tells an agent what must be paid and carries its authorization; derien lets compatible authorized payments resolve across the population before the settlement network moves the money.***

---

26. Normative references

1. x402 V2 specification — `PaymentRequired`, `PaymentPayload`, `SettlementResponse`, scheme logic and scheme-specific `extra`:
   https://github.com/x402-foundation/x402/blob/main/specs/x402-specification-v2.md
2. x402 HTTP 402 core concepts — V2 `PAYMENT-REQUIRED`, `PAYMENT-SIGNATURE`, `PAYMENT-RESPONSE`:
   https://docs.x402.org/core-concepts/http-402
3. x402 facilitator flow — scheme/network-specific verification and settlement:
   https://docs.x402.org/core-concepts/facilitator
4. x402 V2 launch — modular networks and schemes, CAIP-based identifiers:
   https://x402.org/x402-v2-launch/
5. x402 batch settlement — separation of payment authorization from eventual onchain settlement for high-frequency agentic commerce:
   https://x402.org/x402-batch-settlement/
6. x402 network and token support — CAIP-2 network identifiers and network-local asset identifiers:
   https://docs.x402.org/core-concepts/network-and-token-support

---

27. Status note

This draft intentionally maps derien onto x402 rather than creating a separate agent-facing payment protocol. The `derien` scheme described here is a proposal, not an existing x402 Foundation standard. The next implementation step is a Base Sepolia proof of concept that registers a custom x402 scheme, accepts buyer funding authorization, feeds valid intents into the derien resolver and atomically settles a four-asset closure through a reference EVM contract.