Fix Your 402

A builder's field guide to x402 discovery, payment challenges, and pre-pay trust

By SmartFlow
smartflowproai.com
Edition 2026.07

Your paid API can be online, settle payments correctly, and still be skipped by a buyer-agent.

The reason is usually not the product behind the paywall. It is the short machine conversation that happens before payment. The agent must discover the resource, understand its terms, receive a valid `402 Payment Required` challenge, and decide that the quote is safe to pay. One missing field or one stale value can stop that conversation before your application code runs.

This guide gives you a repeatable repair loop:

> **See the symptom -> confirm it with one command -> apply a before-and-after fix -> verify with the validator.**

The examples target current x402 v2 payment challenges and the live `/.well-known/x402` discovery pattern described below. Legacy v1 examples are labeled. Do not mix their fields.

This is an implementation guide, not a security audit or a guarantee that a buyer will pay. A clean validation run proves what your endpoint returned during that run. It does not prove future uptime, settlement correctness, or business demand.

Contents

Part I: Start here

1. How to use this guide

Start with one protected resource, not your whole catalog. Choose an endpoint central to a buyer's use case. Run the validator against its exact URL and method. Then open the chapter that matches the first critical finding.

Use the same loop in every chapter:

1. **Symptom:** Read what the buyer-agent or validator sees.
2. **Diagnose:** Replace the example URL and run the single command.
3. **Fix:** Compare the bad and good shapes. Change the source that generates your response, not a copied response after the fact.
4. **Verify:** Run the validator again. Save the report with the commit that produced it.

Fix structural errors before performance errors. A fast response that no buyer can parse is still unusable. A suggested order is:

1. Discovery manifest
2. Live `402` challenge
3. Version and amount coherence
4. Recipient consistency
5. Stability across probes
6. Latency and server errors

Work on a non-paying probe first

Every diagnosis command in this guide is unauthenticated. It should receive a payment challenge, not settle a payment. Do not paste a wallet key, `PAYMENT-SIGNATURE`, authorization token, cookie, or production secret into these commands.

Replace every example value

The domain, resource path, recipient, price, and asset in this guide are examples. Copy the shape, not the business values. In particular:

- `https://api.example.com` is not your service.
- `0x1111111111111111111111111111111111111111` is not your recipient.
- A displayed discovery price is not automatically the same representation as a wire-level amount.
- A Base USDC example does not mean every network uses the same asset address.

Treat the report as evidence, not decoration

Keep the raw report from the validator. If a later run disagrees, compare the probe time, endpoint URL, method, response channel, and raw response before blaming the endpoint or the validator.

2. The two JSON shapes you must keep separate

Many costly x402 mistakes begin here. Discovery and payment challenge data can describe similar terms, but they are not the same document.

Shape A: the discovery manifest

`GET /.well-known/x402` helps a machine find resources and their advertised terms before calling them. This guide uses the per-resource `accepts` shape served by the live ekisar manifest:

```
{
  "version": 1,
  "resources": [
    {
      "url": "https://api.example.com/weather",
      "method": "GET",
      "description": "Current weather for one location",
      "accepts": [
        {
          "scheme": "exact",
          "network": "eip155:8453",
          "price": "$0.01",
          "payTo": "0x1111111111111111111111111111111111111111"
        }
      ]
    }
  ]
}
```

Here, `price` is a human-readable discovery value. The manifest groups terms under each resource.

The validator may label this family `v2-resource-accepts`. That is the validator's schema taxonomy, not a claim that the manifest's own `version` field must be changed from the live value shown above.

Shape B: the v2 payment challenge

An unauthenticated request to the paid resource returns HTTP `402`. In x402 v2, the server puts a Base64-encoded `PaymentRequired` object in the `PAYMENT-REQUIRED` response header. Decoded, it looks like this:

```
{
  "x402Version": 2,
  "error": "Payment required",
  "resource": {
    "url": "https://api.example.com/weather",
    "description": "Current weather for one location",
    "mimeType": "application/json"
  },
  "accepts": [
    {
      "scheme": "exact",
```

```
      "network": "eip155:8453",
      "amount": "10000",
      "asset": "0x833589fCD6eDb6E08f4c7C32D4f71b54bdA02913",
      "payTo": "0x1111111111111111111111111111111111111111",
      "maxTimeoutSeconds": 60,
      "extra": {
        "name": "USD Coin",
        "version": "2"
      }
    }
  ]
}
```

Here, `amount` is a digit string in atomic units for the asset. It is not the display string from discovery.

The rule that prevents copy errors

Do not paste the discovery `accepts` object into `PAYMENT-REQUIRED`. Do not paste the wire-level `PaymentRequirements` object into the manifest and rename one field until it looks right. Generate both views from one internal payment-terms configuration.

Also keep the version labels separate:

- `version` in the discovery document identifies that document's shape.
- `x402Version` in `PaymentRequired` identifies the x402 payment protocol version.

They do not become interchangeable because both contain a number.

v1 and v2 amount keys

The official x402 SDK types make the migration boundary explicit:

Payment challenge	Canonical location	Network form	Amount key
Legacy v1	JSON response body	Legacy names may appear	`maxAmountRequired`
Current v2	Base64 JSON in `PAYMENT-REQUIRED`	CAIP-2	`amount`

The current v2 type and legacy v1 type are useful references when a wrapper library hides the actual response.

---

3. Quick validator setup

The validator is a GitHub Action. There is no `x402-validate` command to install. Add this workflow to your repository as `.github/workflows/x402-validate.yml`:

```
name: Validate x402
```

```yaml
on:
  workflow_dispatch:
  pull_request:

jobs:
  validate:
    runs-on: ubuntu-24.04
    steps:
      - uses: smartflowproai-lang/x402-endpoint-validator@v1
        with:
          endpoints: 'https://api.example.com/weather'
          strict-v2: 'true'
          report-path: 'x402-validator-report.json'

      - name: Save validator report
        if: always()
        uses: actions/upload-artifact@v4
        with:
          name: x402-validator-report
          path: x402-validator-report.json
```

Replace the endpoint with your deployed protected URL. Run the workflow manually or open a pull request.

Use `strict-v2: 'true'` when your intended contract is v2. Compatibility mode can recognize a valid legacy body response, but that does not make the response v2.

The action and report format are documented in the x402 Endpoint Validator repository.

Read the useful part of a downloaded report

```
jq '{
  summary,
  endpoints: [
    .endpoints[] | {
      url,
      passed,
      checks: {
        manifest: .checks.manifest,
        challenge: .checks.body_conformance,
        latency: .checks.response_time_p95,
        payment_required: .checks.payment_required
      }
    }
  ]
}' x402-validator-report.json
```

Do not stop at the top-level pass or fail. The inner fields tell you which layer broke and what the probe actually observed.

4. Reading a scan report

Use this map to go from a report field to the right repair chapter.

Report signal	Meaning	Go to
Manifest is unreachable, not JSON, or unrecognized	Discovery is broken	Chapter 1
`schema_kind: resource-list-bare`	URLs are listed without per-resource terms	Chapter 2
`failure_class: header_only_accepts`	No usable v2 header and no usable body `accepts[]` were found	Chapter 3
`failure_class: v2_header_invalid`	The v2 header exists but cannot be decoded or validated	Chapters 3 and 5
`legacy_placement: true`	Terms were found only in the response body	Chapters 3 and 5
`failure_class: network_format`	A v2 network is not CAIP-2 or is missing	Chapters 3 and 5
`failure_class: amount_key`	The expected amount field is absent or malformed for the detected version	Chapter 4
`channel_mismatch: true`	Header and body disagree on critical terms	Chapters 4, 5, and 6
`channel_mismatch_fields` contains `payTo`	Recipient differs between channels	Chapter 6
Fresh reports disagree on unchanged code	Endpoint, deployment, cache, or observation drift	Chapter 7
Reachability or payment-required behavior fails	The unauthenticated route did not produce a usable challenge	Chapter 8
Response-time check fails	The challenge path exceeded your configured budget	Chapter 8

`caip2_in_header: true` is an observation, not a failure. It means the decoded canonical header contains a CAIP-2 network identifier, which is expected for v2.

5. Evidence and field-note index

This guide was shaped by live endpoints and builder reports. The cases are not interchangeable. Some are public, some have only a live post-fix artifact, and some do not yet have a public permalink. The table says which is which.

Case	Public evidence	Lesson used here
CRYPTYX	x402 issue thread and resolution comment	A request can fail input validation before it reaches payment middleware. Discovery and settlement are separate gates.
RelayShield	public tracking report	A valid settlement does not prove that a discovery index is current. Use a control resource and separate chain evidence from catalog evidence.
NetIntel	live public listing	Probe the actual method and input contract. No public fix-thread claim is made in this edition.
Owervanz	No public permalink is included in this edition.	A pre-payment checklist can be useful before an operator asks for a full audit. No private exchange is quoted here.
ekisar	live post-fix manifest and public validator commit	A per-resource manifest needs terms attached to resource objects, and the validator must recognize the live ecosystem shape.

No private thread, unlinked pass count, or inferred pre-fix payload is presented as public evidence. Use a fresh rerun and a public methodology link before adding any pass count or ecosystem rate to a future edition.

What is in the full guide

This free sample is Part I: the foundations. The full guide (35 pages) adds nine repair chapters, one per failure class found in live scans of production x402 endpoints:

1. Missing or malformed `/.well-known/x402`
2. `resource-list-bare`
3. Header-only accepts and CAIP-2
4. `amount_key` mismatch
5. Mixed v1 and v2 signals
6. `payTo` drift
7. Scan-to-live paywall drift
8. Latency and 5xx under probe
9. What buyer-agents check before paying

Each chapter: the symptom, one diagnostic command, a before-and-after fix, and validator verification. Plus a one-page pre-launch checklist.

Full guide: https://smartflow6.gumroad.com/l/fix-your-402 (launches August 3)

The validator used throughout is free and open source: https://github.com/smartflowproai-lang/x402-endpoint-validator