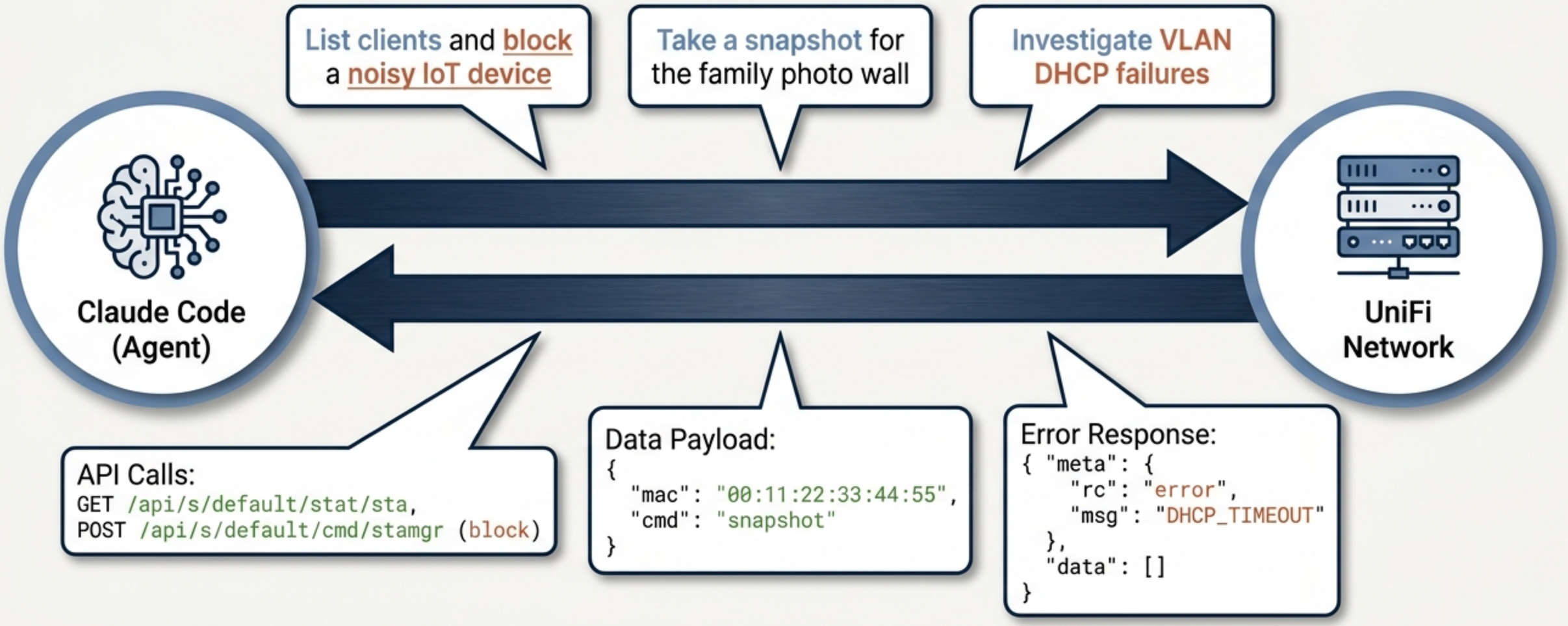
Building a Custom UniFi MCP in Three Days

A technical post-mortem on composing a fresh Model Context Protocol (MCP) server, navigating undocumented APIs via status codes, and packaging 103 tools into a frictionless, two-command plugin.

```
> TOOLS_SHIPPED: 103
> UNIT_TESTS_PASSED: 208
> DEVELOPMENT_TIME: 12 Hours
> STATUS: Ready for Claude Code
```

Elevating Claude Code to Layer 2 Network Management

The goal was an agentic integration—not a chat assistant that knows networking, but an autonomous system that can proactively list, mutate, and troubleshoot home network state via the UniFi API.

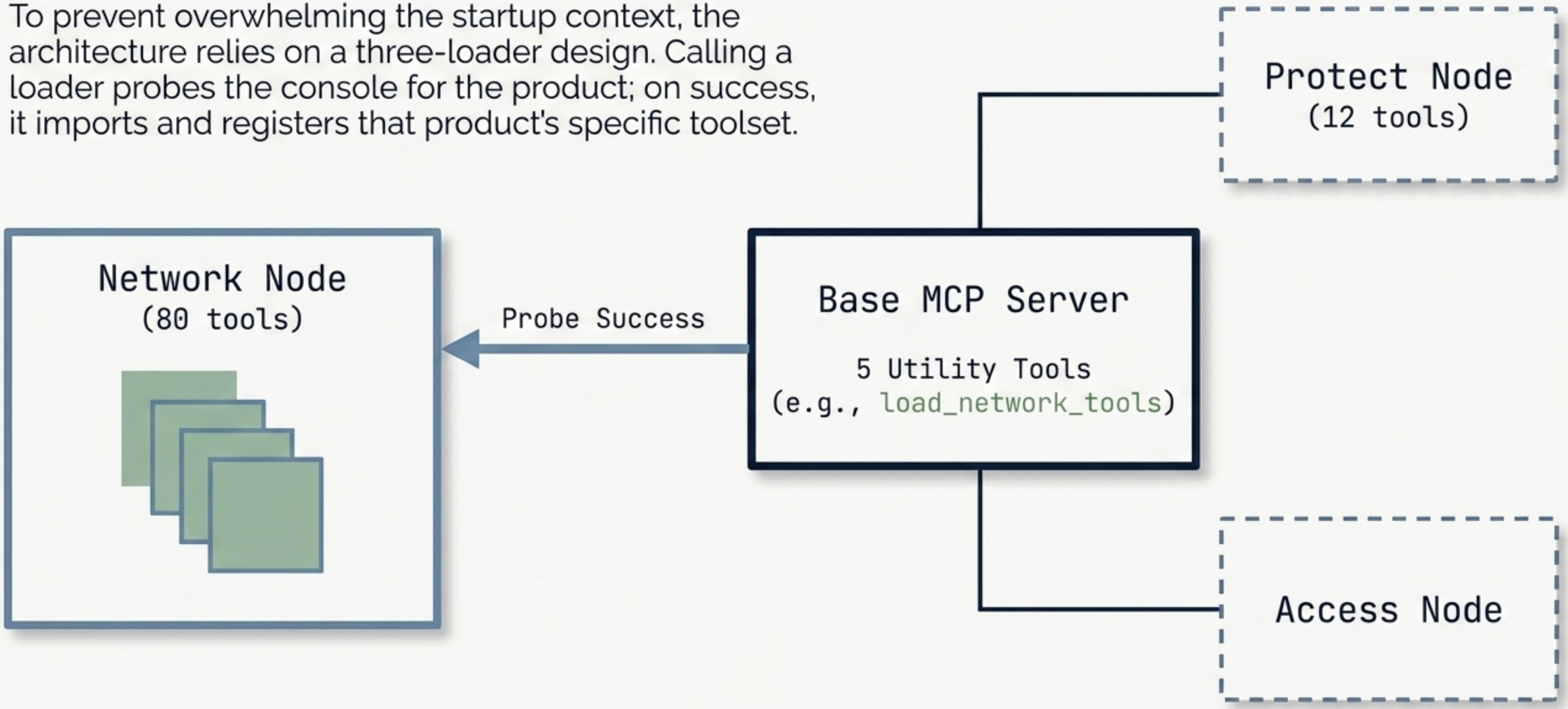


Evaluating the Existing Open-Source Landscape

	`sirkirby/unifi-mcp`	`enuno/unifi-mcp-server`
License	MIT	Apache 2.0
Coverage	Broad (Network, Protect, Access)	Deep Network (Zone Firewalls, Topologies)
Auth Pattern	Cookie	API-Key (Modern OS 4.0+)
Strengths	Preview/Confirm Destructive Ops	Advanced Routing Features
Verdict	Both excellent. Neither complete alone. A fork inherits technical debt; a fresh architecture composes their best patterns.	

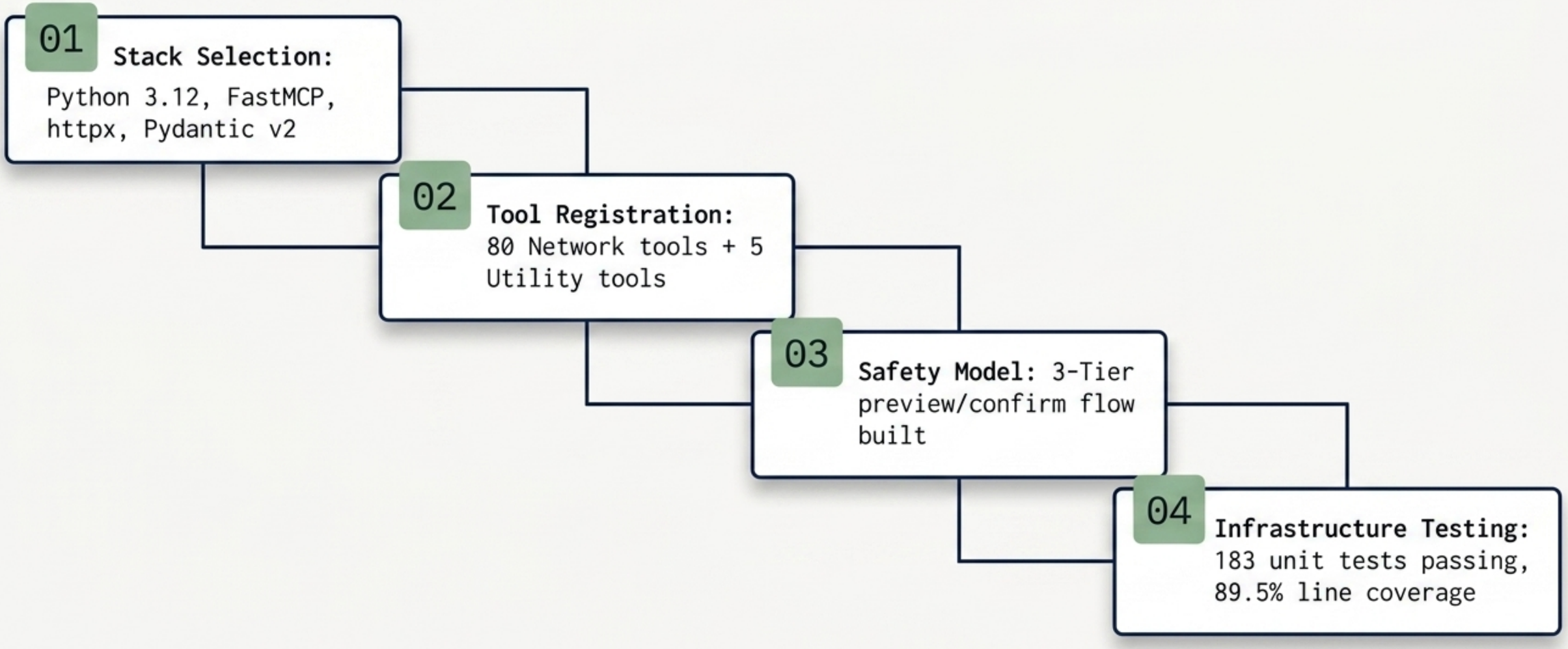
Designing a Lazy-Loaded Topology

To prevent overwhelming the startup context, the architecture relies on a three-loader design. Calling a loader probes the console for the product; on success, it imports and registers that product's specific toolset.



Phase 1 Execution Yields 85 Tools in One Day

The initial MVP focused entirely on throughput and infrastructure design. By the end of day one, the server was functional in local tests via a wrapper script, but untested against live firmware state.



The Live Hardware Audit Exposes Three Critical Blockers

Pydantic Serialization Crash

FastMCP blew up trying to serialize UnifiClient into a JSON schema.

Multi-Tenant Proxy Trap

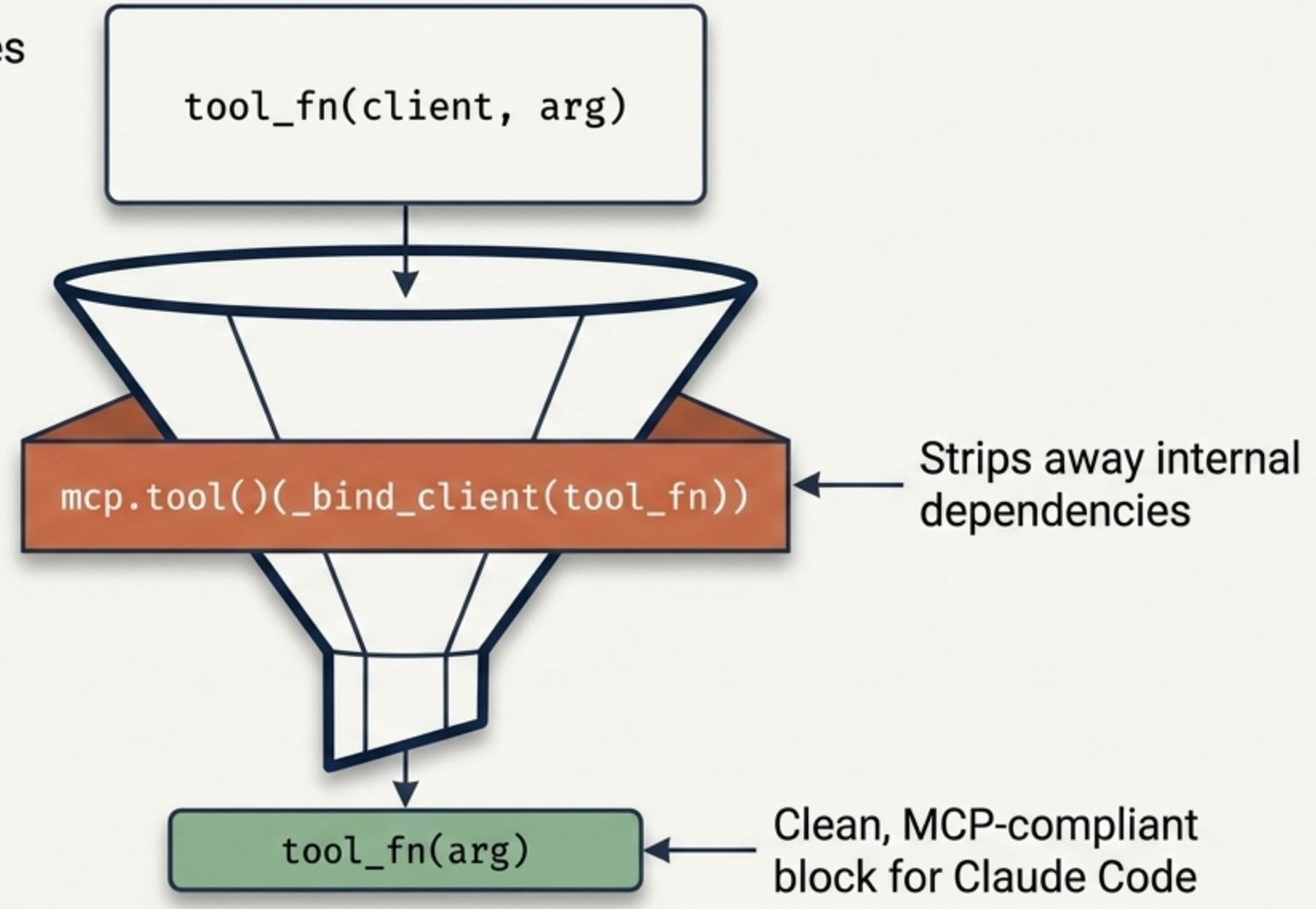
Expecting value: line 1. Hitting an uninstalled product endpoint returned a a 200 HTML landing page, crashing the JSON parser.

Legacy Endpoint Drift

Protect probe returned 500 because the legacy /proxy/protect/api/bo- otstrap required cookie auth, not modern API-keys.

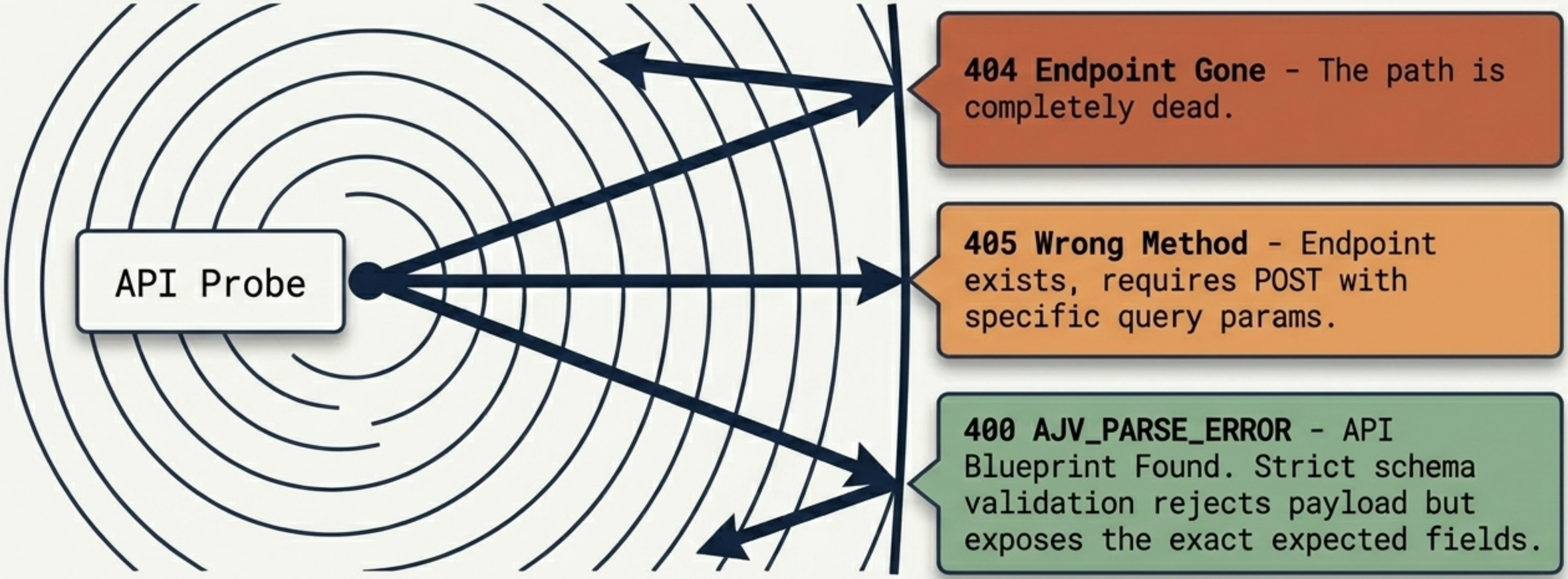
The Closure Trick for FastMCP Dependency Injection

FastMCP introspects signatures to build protocol schemas. Passing shared infrastructure crashes the serializer. Pre-binding the client via a closure wrapper hides the dependency from the public signature while keeping it accessible internally.



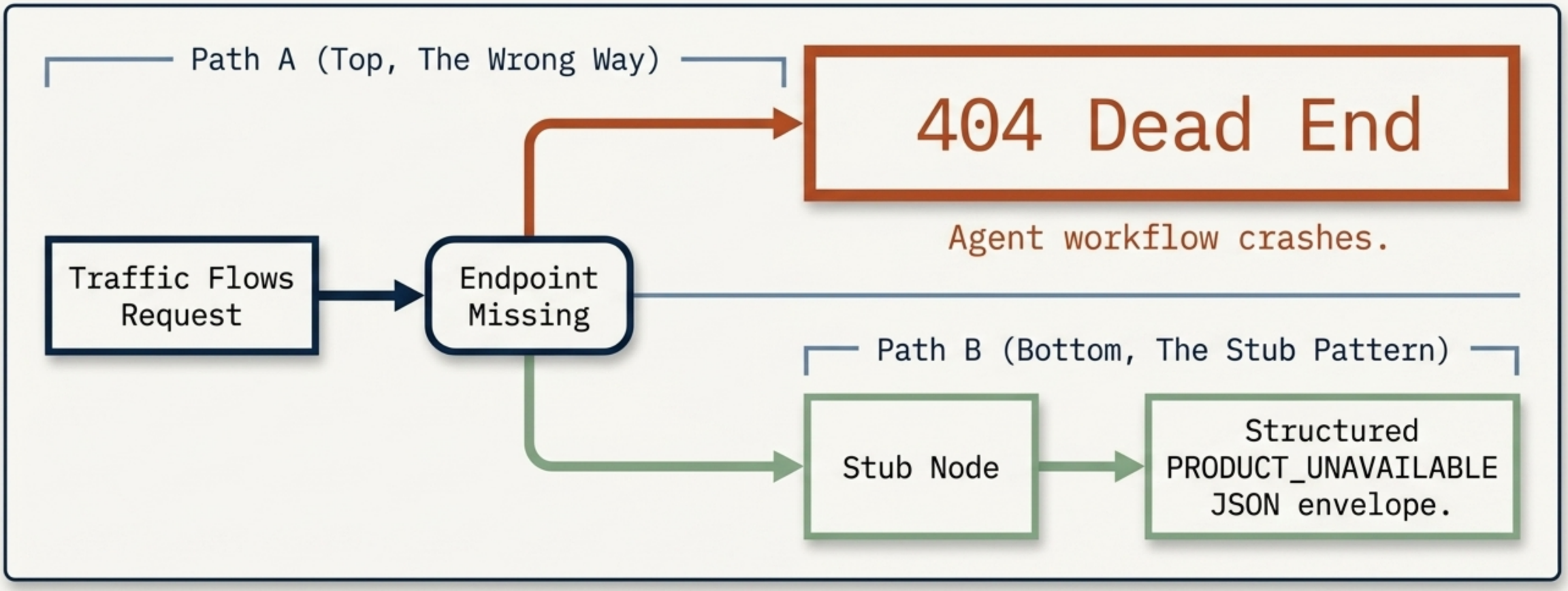
Navigating Undocumented APIs via Endpoint Archaeology

Without documentation, HTTP status codes become a navigation system. Sending plausible payloads and reading the AJV_PARSE_ERR0R rejections mapped the exact boundaries of the firmware.



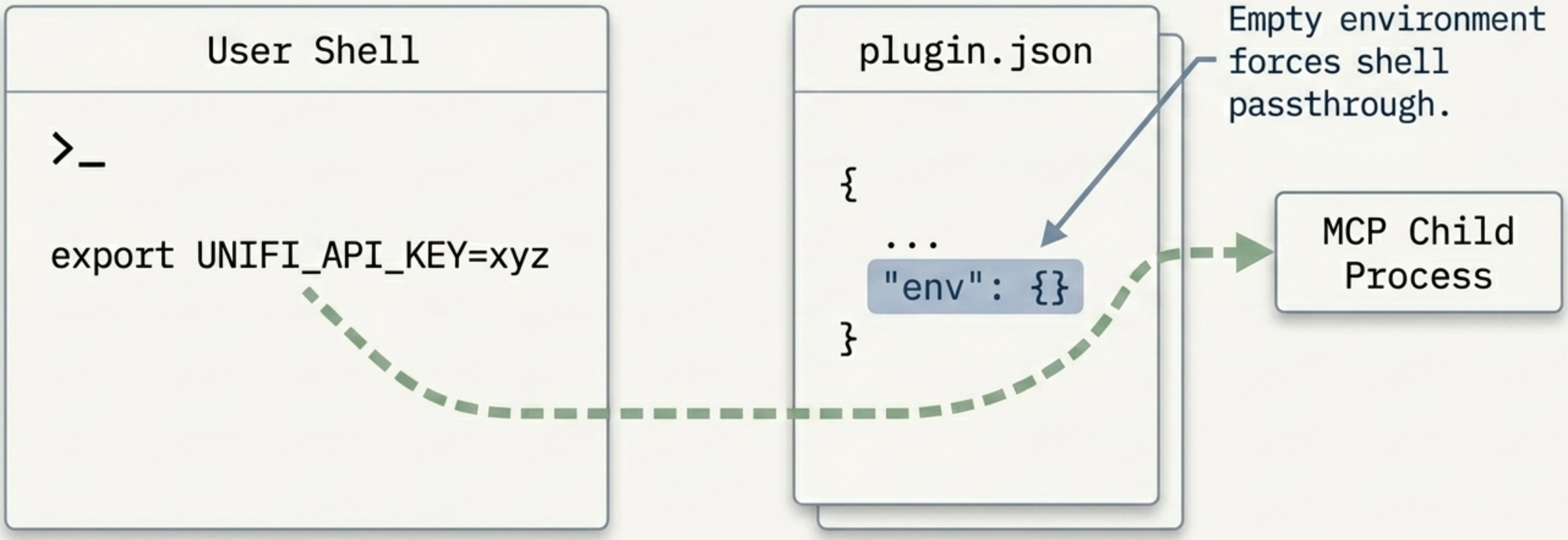
Graceful Failures Preserve the Agent Workflow

When firmware lacks support for specific endpoints, synthesizing fake data is a lie. Dropping tools causes abrupt agent failures. Registering explicit stubs documents missing paths and keeps the workflow alive for future updates.



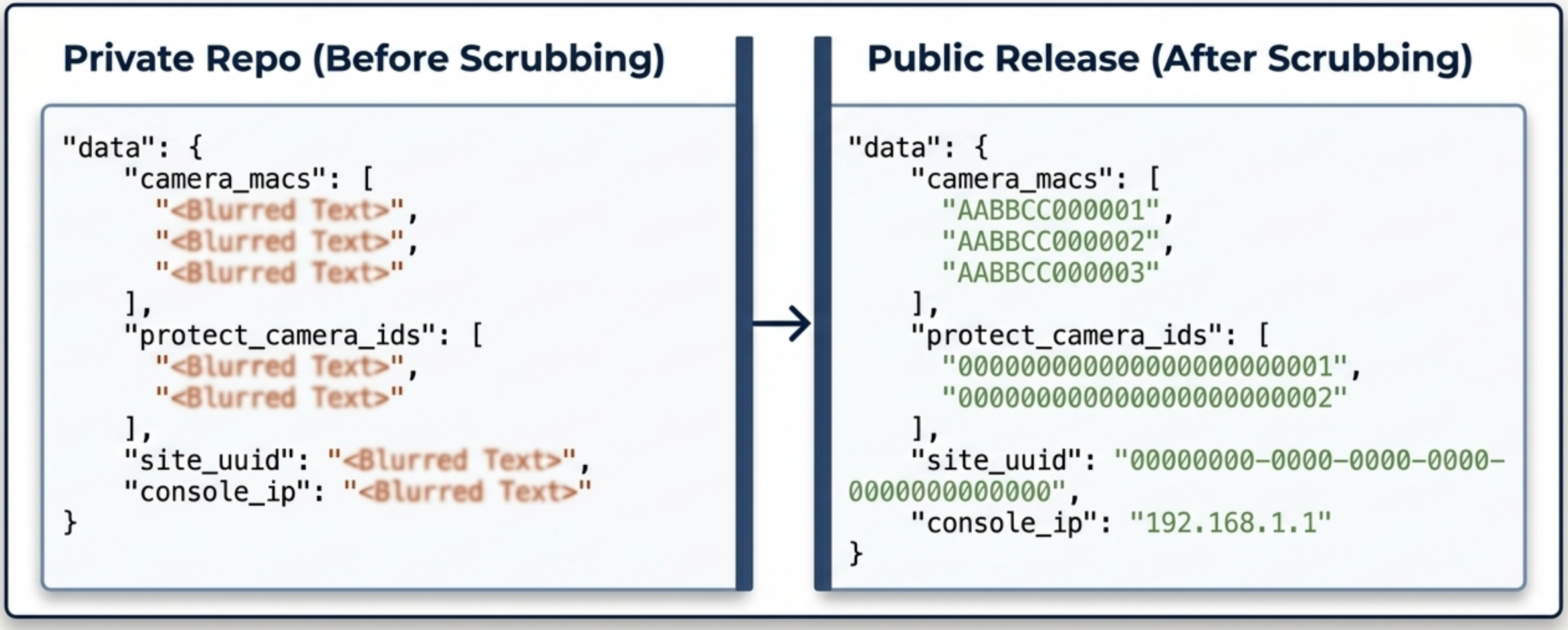
Secret Hygiene via the Empty Environment Block

Setting env: {} in the configuration instructs Claude Code to pass the launching shell's environment straight through to the child process. The plugin itself never stores, sees,or asks for the API key.



Identifier Scrubbing for Public Release

Publishing real network topography in a public MIT repository is a privacy failure. Every live response fixture and documentation snippet underwent a systematic sanitization pass.



A 12-Pattern Framework for Agentic Integrations

1. Architecture & Design

- Compose over clone.
- Lazy async resolvers.

2. API Integration

- Probe by status code.
- AJV negative-space mapping.
- Fail at the boundary (non-JSON).

3. Security & Privacy

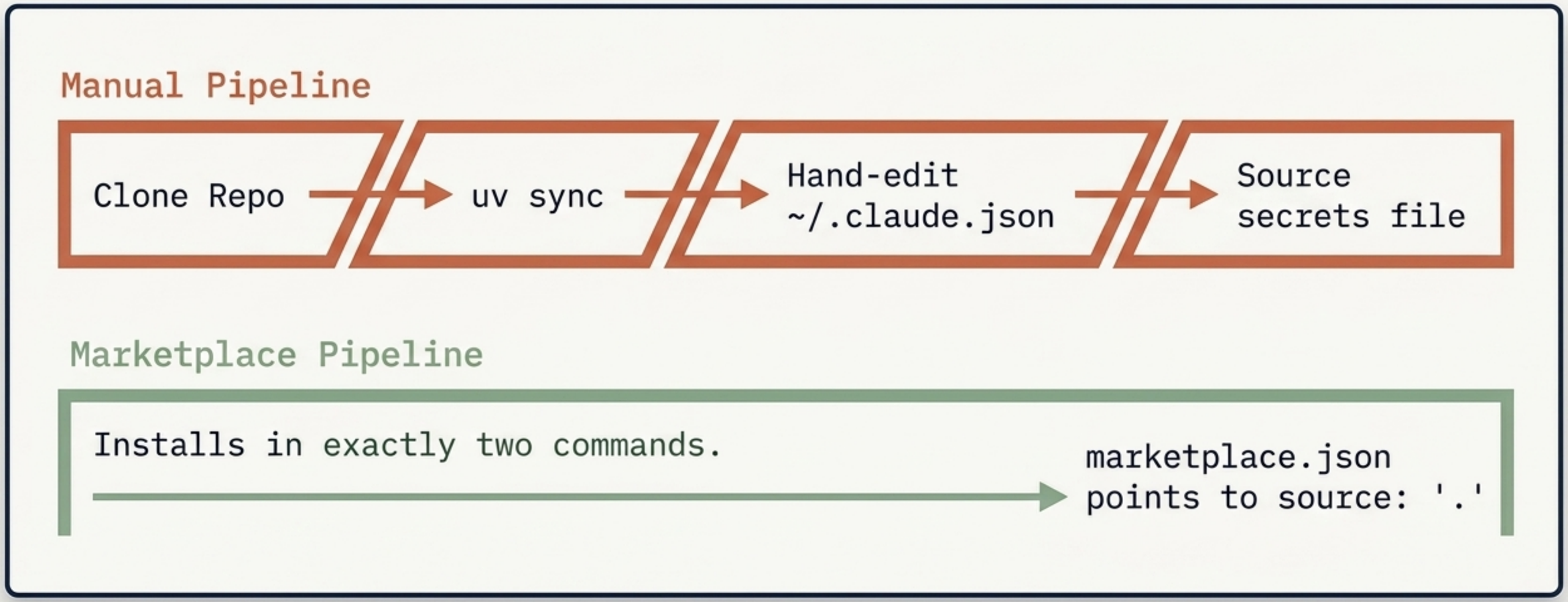
- Empty env: {} for secret passthrough.
- Systematic identifier scrubbing.

4. UX & Tooling

- FastMCP closure dependency injection.
- Register stubs over fake data.
- Marketplace of one.
- Live captures over invented fixtures.
- Parallelize docs via subagents.

Engineering a Zero-Friction Install Experience

A powerful tool surface is useless if it requires a 10-minute integration manual. By shipping the server as a native Claude Code plugin, the friction drops to near zero.



The 30-Second Launch Sequence

The architecture is proven. The endpoints are mapped. The stubs are waiting to automatically promote themselves on the next firmware update. Go break something on your network.

```
> claude mcp add-marketplace unifi
https://raw.githubusercontent.com/chris2ao/unifi-mcp/main/marketplace.json

> claude mcp install unifi-mcp
```

103 Tools Live

Unifi OS 4.0+ Ready

MIT Licensed