Rules Are Not Suggestions

A History of MCP Non-Compliance

Sterling Dreyer • MCP Dev Summit • April 2, 2026

About Me & Arcade

Sterling Dreyer

Founding Engineer

- Manage Infrastructure at Arcade
- Work on the core product
- Worked on building out the MCP integration

Arcade.dev

The runtime for MCP

- Multi-user scoped Auth for agentic tools
- Server evaluation suites
- MCP Server hosting

An MCP server isn't **compliant** just because it connects.

Integrating with MCP Was… Hard

It was a gamble if the client would connect

Non-authed servers typically worked fine. Servers using OAuth were hit or miss

Is it the client or the server?

Tried a bunch of different servers. Couldn’t figure out why it was so inconsistent

Felt like a widespread issue

If I’m struggling, what's everyone else experiencing?

So We Analyzed the Entire Ecosystem

41,924

MCP Servers

analyzed

218,410

Tools

inspected

We built two tools to understand the problem:

ToolBench

Scans source code for spec compliance and tool definition quality

MCP Debugger

Live-tests running servers against every requirement in the spec

94%

of servers failed compliance

6% Fully complied with requirements

9% Failed one check

85% Failed two or more checks

Let's dig in.

What's going wrong, and how do we fix it?

4 Spec Revisions in 12 Months

v1
Nov 2024

- JSON-RPC 2.0 transport
- Tools, Prompts, Resources
- Sampling
- Roots
- Logging
- No auth requirements

v2
Mar 2025

- OAuth 2.1 + DCR
- PKCE required
- Streamable HTTP (replaces SSE)
- Tool annotations

v3
Jun 2025

- Elicitation
- PRM (SHOULD)
- Origin header
- Structured output
- SEP-1613 inputSchema

v4
Nov 2025

- PRM → MUST
- Origin → MUST
- CIMD (replaces DCR)
- Resource indicators
- URL elicitation

Conformance Testing: Under-Invested

A conformance suite exists — but nobody knows

50 GitHub stars vs. thousands on the spec and SDKs. Most developers have never heard of it.

Auth tests are client-only

Server scenarios cover initialize and tools, but DCR and OAuth auth tests only validate client behavior — not server compliance.

Narrow coverage

No tests for PRM discovery, Origin headers, or inputSchema — the top failure modes in our data.

Solution: Invest in Conformance Testing

The foundation exists. It needs investment, visibility, and more coverage.

Promote the existing suite

github.com/modelcontextprotocol/conformance exists today. Make it a first-class part of the ecosystem.

Add server-side auth scenarios

Auth scenarios only test clients today. Add server-side tests for PRM, Origin headers, and OAuth endpoints.

Make it the standard CI gate

It already has a GitHub Action. Push it into every SDK's getting-started guide and CI templates.

Map every requirement to a test

Today it covers ~4 scenarios. The spec has dozens of MUSTs. Close the gap and ship tests with each revision.

inputSchema: 19% of All Tools Are Broken

43,251

tools affected
of 218,410 total

Null or empty inputSchema	19,535 (8.9%)
Empty {} — missing type: "object"	23,716 (10.9%)

Root Cause: Frameworks Fail Silently

Python SDK	Untyped params silently default to {"type": "string"}
TypeScript SDK	z.discriminatedUnion() silently dropped → empty {}
Go SDKs	Uninitialized struct → invalid schema, zero validation

Solution: Framework-Level Enforcement

73% of servers are built on 3 frameworks. Fix those, fix the ecosystem.

Reject invalid inputSchema at registration

Throw instead of silently falling back to {}

Require type: “object” in all tool schemas

Frameworks should validate that every inputSchema includes type: “object” before registering the tool.

Warn on untyped or ambiguous parameters

Flag when params silently default to {“type”: “string”} or when discriminatedUnion() drops fields.

API Keys: The Path of Least Resistance

The first MCP spec (2024-11-05) had zero auth requirements.

How It Works

- Build a server around an API
- Pass API key as env var or header
- **Simple. Works. Ships fast.**

The Problems

- User has to go mint a key in some UI
- One key = one user (or all users)
- No scopes, no granular permissions
- **Less Secure**
- Not in the spec — it's a workaround

v2 Introduced OAuth with DCR

2025-03-26 — The spec said: use OAuth 2.1 with Dynamic Client Registration

The Promise

- User just signs in with existing account
- Supports scopes and granular permissions
- Standard, interoperable protocol

The Reality

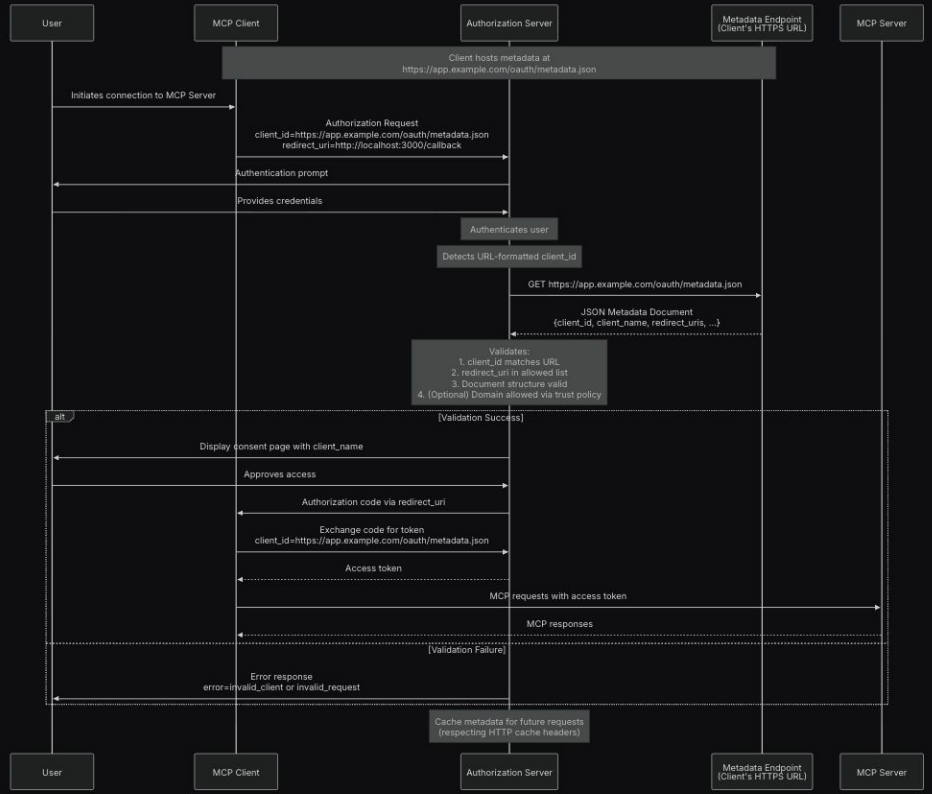
- People still used API keys
- OAuth adoption: still low
- ~22% among v2+ servers — still a minority
- And those who tried it... struggled

The Scale Problem

A GitHub MCP server needs a full OAuth front door just to call the GitHub API. Want Linear too? Another full setup. GitHub, Box, and Slack don’t even support DCR.

The MCP OAuth Flow

What the spec actually requires for a single authenticated connection:



The Complexity Gap

API Key Flow

1. Get API key from dashboard
2. Set as environment variable
3. Pass it in a header

Done. Anyone can build this.

OAuth 2.1 + DCR Flow

1. PRM discovery (RFC 9728)
2. AS metadata lookup (RFC 8414)
3. Dynamic client registration
4. Authorization code + PKCE
5. Token exchange + refresh
6. Scope validation + resource binding

What Fails in Practice

67% fail	22% fail	24% fail	80+% fail
PRM Endpoint	401 Status	WWW-Auth Header	Origin Header validation
SHOULD*	MUST	MUST	MUST

Solution: Framework OAuth Support

Frameworks should make full OAuth compliance the default path, not a maze.

Built-in PRM endpoint generation

If a framework's OAuth helper is enabled, automatically serve /.well-known/oauth-protected-resource. No extra code.

Guided OAuth setup wizards

Step-by-step scaffolding that walks developers through DCR, PKCE, and scope configuration. Make the 9-step flow a 3-step CLI command.

Compliance-by-default templates

Starter templates that ship with 401 responses, WWW-Authenticate headers, and Origin validation already wired up.

MCP should endorse compliant frameworks

Official badge program for frameworks that produce spec-compliant servers out of the box. Market incentive to get it right.

Other Solutions

Beyond the frameworks — what else can the ecosystem do?

Solution: Registry Gatekeeping

Stricter requirements for publishing to official registries.

- **Run conformance suite before listing a server**
- Reject servers that fail MUST requirements
- Show compliance badges — A/B/C/D/F grades

Solution: Client-Side Rejection

This is how HTTPS adoption actually happened.

How HTTPS Won

- Browsers showed warnings
- Then started blocking
- Suddenly everyone had certificates
- **Non-compliance had consequences**

MCP Equivalent

- Clients refuse non-compliant servers
- Warn about missing PRM
- Reject broken inputSchema
- **Block arbitrary header injection**

Right now, non-compliance has zero consequences. That's why 94% fail.

Rules are not **suggestions.**

The spec says MUST.
The ecosystem says "meh."

The violations are measurable.
The fixes are known.
The gap is entirely about enforcement.

Thank You

Sterling Dreyer • sterling@arcade.dev

arcade.dev • mcpdebugger.dev • toolbench.arcade.dev