MCP SERVER | NO CODE | CLOUD HOSTED

Postman MCP for AI Agents

Manage API collections and mock servers through a conversational interface.

Postman connects your API lifecycle to your AI agent. It lets your agent browse collections, explore environments, check mock servers, and monitor health statuses without you having to copy-paste JSON schemas or cURL commands. It's about making your API documentation and testing workflows conversational.

A+ Quality Score 100/100

api-testing | api-documentation | mock-servers | rest-api | collection-management | endpoint-testing

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JULY 2026
vinkius.com/mcp/postman

SCAN TO VIEW

The connectivity layer between AI and the world's software.

Vinkius sits between AI and every application. All communication passes through Vinkius Cloud via the Model Context Protocol (MCP) — with governance, observability, and security at every layer.

Your AI Connections Run Through Vinkius Cloud

The world's largest managed MCP catalog

Vinkius is the connectivity layer where AI connects to the software your business already runs. We handle the hosting, the security, the credentials, the uptime — you get agents that actually do things.

We operate the world's largest managed MCP catalog. Major SaaS platforms, CRMs, databases, and cloud providers — running, monitored, production-ready. This MCP server is hosted and maintained by the Vinkius Cloud for AI Agents.

> *The agent doesn't manage credentials, doesn't manage uptime, doesn't manage security. Vinkius does.*
>
> — Architecture principle

Four Pillars of the Vinkius Runtime

01 — Security by design

Credentials stay encrypted at rest via AES-256. The AI agent never touches raw keys — they're injected into a sandboxed V8 isolate at runtime. Actions are logged, and connections have an emergency kill switch.

02 — Built on MCP Fusion

This MCP server was built with **MCP Fusion**, the open-source framework (Apache 2.0) that powers the entire Vinkius catalog. Schema-as-firewall strips undeclared fields, compiled PII redaction runs at zero overhead, and cryptographic lockfiles produce git-diffable audit trails.

03 — Deterministic observability

Eight immutable metrics per endpoint: request volume, p95 latency, error rate, active connections, cost attribution. A live payload feed logs every tool call with mutation detection.

04 — Autonomous operations

Servers are deployed, monitored, and patched autonomously. New capabilities and security patches ship weekly. Zero-downtime deployments ensure continuous availability across all managed MCP servers.

AES-256	Ed25519	24h TTL	V8 Isolate
Encryption at rest	PKI vault signatures	Ephemeral session keys	Sandboxed execution

One Token. Instant Access.

Every MCP server on Vinkius is accessed through a **Connection Token**. Tokens are generated in the cloud dashboard and produce a unique MCP endpoint URL. Paste this URL into any MCP-compatible client — no SDK required.

A single token can serve **multiple AI clients simultaneously**, or you can issue separate tokens per client for granular access control. Each token tracks its own request count, last activity timestamp, and can be individually enabled or revoked.

MCP ENDPOINT

```
https://edge.vinkius.com/{token}/mcp
```

Claude · Cursor · VS Code · Windsurf · Grok · Gemini

Security Is the Architecture

Security in Vinkius is not a feature — it's the foundation of the runtime. The gateway enforces multiple independent protection layers between AI agents and third-party APIs.

01 — Ed25519 PKI Vault

Every workspace has an Ed25519 Master Key. Session keys are generated ephemerally (24h TTL) and signed by the Master Key. Credentials never leave the vault boundary.

02 — V8 Isolate Sandboxing

Tool code runs inside isolated-vm V8 isolates with 64 MB memory caps and per-request timeouts. No filesystem access, no network access except through the SSRF-guarded fetch bridge.

03 — SSRF Guard

All outbound HTTP requests are DNS-resolved and validated before execution. Private IP ranges (10.x, 172.16-31.x, 192.168.x, AWS metadata 169.254.x) are blocked at the network layer.

04 — DLP & PII Redaction

A ResponseGuard pipeline intercepts every tool response. Configurable redaction patterns strip sensitive fields (emails, SSNs, card numbers) before data reaches the AI agent.

05 — Cryptographic Audit Trail

Every request is signed into a SHA-256 hash chain with Ed25519 signatures. Events form a tamper-proof, SIEM-exportable forensic record.

06 — Honeytoken Trap System

Phantom credentials are injected into isolated environments. If a honeytoken is used outside Vinkius infrastructure, the server is quarantined instantly.

Emergency Kill Switch

EU AI Act Art. 14(1) Compliant

The kill switch is an **emergency halt** mechanism — not a simple toggle. When triggered, it executes three actions atomically:

01 — Server deactivated

The MCP server is immediately taken offline across the entire cluster.

02 — All tokens revoked

Every connection token is invalidated. Total lockout — reconnection blocked until new tokens are issued.

03 — WebSocket connections killed

Active connections terminated via Redis pubsub broadcast. Propagates to every runtime node in the cluster.

Full Visibility. Zero Guesswork.

The Vinkius cloud dashboard includes a full MCP Governance suite — real-time analytics and security controls for production AI operations.

Control Plane

KPI dashboard with request volume, latency, success rate, token consumption, and AI-generated operational briefings.

Agent Activity

Which AI clients are connecting, how often, and what they're doing — real-time session tracking.

FinOps

Cost tracking per tool, payload compression savings, budget optimization signals, and consumption trends.

Tool Health

Slowest and most error-prone tools, with actionable root-cause insights and performance baselines.

Firewall & DLP

PII redaction activity, sensitive data protection counters, and security event timeline.

Incident Log

Error trends, failure rates, status-code breakdowns, and forensic audit trail access.

> Get started at **cloud.vinkius.com** — connect your AI agent in under 60 seconds.

Postman MCP

6 tools available
Cloud-hosted on Vinkius

Postman connects your developer keys to an AI agent so you can bring collaborative API design and monitoring into a pure conversational context. Instead of switching tabs to find a specific endpoint or manually copying headers into a prompt, you can just ask your agent to find the right data. It reads your collections to understand how your internal APIs actually behave, looks at your environments to distinguish between staging and production, and checks your mock servers to see what your front-end team is working against. If you're trying to figure out why a scheduled monitor failed, the agent can pull that history for you instantly. This setup is part of the Vinkius catalog, making it easy to plug into your existing workflow. You stop treating your API docs as a static file and start treating them as a live, queryable database for your AI. It removes the friction of context switching where you spend half your time just trying to get the AI to understand the shape of the data you're sending. You can get exact HTTP payloads and mock server URLs when building your UI bindings, or quickly verify if staging environment URLs differ from production without navigating the app. It turns your API infrastructure into a searchable knowledge base for your agent.

Core Capabilities

01 — Browse API Collections

Your agent can see all available API collections in your account to understand your full API surface.

02 — List Engineering Workspaces

See every workspace your team uses to keep your agent aware of different project scopes.

03 — View Environment Variables

The agent can see your variables for staging and production to avoid using the wrong URLs.

04 — Identify Active Mocks

Quickly find active mock servers when you need to test front-end components against simulated data.

05 — Check Monitor Status

Get immediate updates on your scheduled health checks and see which monitors are failing.

06 — Download Collection Schemas

Your agent can pull complete JSON schemas to learn exactly how to hit your internal endpoints.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/postman** — connect your AI agent in three steps.

01 Subscribe to this MCP and enter your Postman Developer API Key.

02 Connect the MCP to your AI client like Claude, Cursor, or Windsurf.

03 Ask your agent to describe an endpoint, check a mock server, or pull a collection schema.

The bottom line is that your AI finally has the same map of your API infrastructure that your dev team uses every day.

Built For

The backend engineer who's tired of copy-pasting cURL commands, the frontend dev building UI against mocks, and the QA tester who needs to monitor health checks without a dashboard.

Backend Engineer

You use this on a Tuesday afternoon to verify if your staging environment URLs are correct without leaving your IDE.

Frontend Developer

You ask the agent to find the exact HTTP payloads and mock URLs for your new UI components so you don't have to hunt for them.

QA Tester

You use this to quickly check which scheduled monitors failed over the weekend and what the specific error codes were.

What Changes When You Connect

01 Stop copy-pasting JSON: Use get_collection to give your agent the full schema of any API instantly without manual exports.

02 Environment awareness: Use list_environments so your agent knows exactly which variables belong to staging versus production.

03 Faster front-end dev: Use list_mocks to quickly find the correct mock URLs when building out UI components and bindings.

04 Proactive monitoring: Use list_monitors to have your agent alert you to failed cron checks without you having to check a dashboard.

05 Workspace visibility: Use list_workspaces to let your agent see the full scope of your team's engineering projects and collections.

Real-World Applications

Finding mock URLs for UI bindings

A frontend dev is stuck on a login page. They ask the agent to find the mock server for the auth flow. The agent uses list_mocks to find the URL and the JSON schema.

Verifying staging environment URLs

A backend engineer needs to know if the staging URL changed. They ask the agent to check the environments. The agent uses list_environments to provide the latest URLs.

Debugging failed scheduled monitors

A QA tester sees a red light on a dashboard. They ask the agent to check the health monitors. The agent uses list_monitors to report the specific failure history.

Learning internal API schemas

A new hire needs to know how to call the Create User endpoint. They ask the agent to pull the collection. The agent uses get_collection to explain the required headers and body.

Patterns to Avoid

Manually pasting cURL commands

✗ AVOID

Copying a raw cURL from Postman into your AI client and hoping the AI understands the headers.

✓ INSTEAD

Use get_collection to let the agent read the full schema directly. This ensures the AI has the correct context for every request.

Guessing environment variables

✗ AVOID

Asking the AI to write a request and having it guess which URL to use for staging.

✓ INSTEAD

Use list_environments to let the agent see your actual scoped variables. This prevents the agent from hitting production data by mistake.

Checking dashboards for every failure

✗ AVOID

Opening a browser tab every time you want to see if your cron jobs passed.

✓ INSTEAD

Use list_monitors to query your health history directly through your AI client. You get the status without the tab switching.

The Right Fit

Use this if you need your AI to understand the actual structure, variables, and health of your Postman-managed APIs. It's for teams that want the agent to act as a knowledgeable member of the dev team who knows where everything is. It's perfect for backend engineers who need to verify staging URLs or frontend devs who need to grab mock server URLs without leaving their editor. Don't use it if you just want to generate generic code snippets without any context of your actual internal endpoints. If you only need to send a one-off request and don't care about collections or mocks, this might be overkill. This is for living, breathing API lifecycles, not just static documentation. It's not a replacement for actually running tests, but it is the best way to give your AI the context it needs to write correct integration code.

Postman API Lifecycle Management for Teams

Developers usually spend a lot of time context-switching. You have to leave your IDE, open the Postman app, find the right collection, switch to the correct environment, and then copy the JSON body into your prompt. It's a lot of friction that leads to hallucinated API calls because the AI doesn't know the latest schema.

This MCP lets the AI do that work for you. Your agent can look up the collection, see the environment variables, and verify the mock server status in one go. You get a conversational way to interact with your actual API infrastructure, ensuring the code the AI writes actually works the first time.

Postman Mock Server and Monitor Visibility

Tracking down why a scheduled monitor failed or finding the right mock URL for a decoupled front-end test usually requires digging through logs or clicking through multiple workspace folders.

You can just ask the agent. It pulls the monitor history or lists the active mocks immediately. You get instant answers to why is this broken or where is the mock without the manual hunting.

Postman 6 Tools for API Testing and Management

Browse collections, check environment variables, and monitor API health directly through your AI agent.

#	TOOL	DESCRIPTION
01	`list_collections`	Shows all API Collections in your Postman account. This helps the AI understand what APIs are available.
02	`list_workspaces`	Lists every engineering workspace available to your team. You can use this to see the full scope of your team's projects.
03	`list_environments`	Shows your development environments and their specific variables. This lets the AI know the difference between staging and production.
04	`list_mocks`	Lists the mock servers you've set up for front-end testing. It helps your agent find the right URLs for your UI bindings.
05	`list_monitors`	Shows your API health monitors and their recent success or failure history. You can use this to check for errors in your scheduled cron checks.
06	`get_collection`	Downloads a complete schema of a collection so the AI can learn your API structure. This provides the exact headers and request bodies for your agent.

See It in Action

Real prompts you can use once this MCP is connected to your AI agent through Vinkius Cloud.

U Are there any Mock servers currently simulating our Auth API?

I checked your Postman configurations and found 1 active mock server tied to 'User Onboarding Flows'. It exposes exactly 4 static JSON responses for `/login` and `/token` and operates at URL `https://c123.mock.pstmn.io`. Shall I pull its schema details?

U Download our core API Collection. Tell me exactly what parameters I need to submit to the Create User endpoint.

I extracted Collection `col-x71...`, the 'Create User' endpoint (POST `/api/v2/users`) expects a JSON body containing `email` (string, required), `role` (string, defaults to 'viewer'), and `department_id` (number, required). Would you like me to write a Javascript snippet invoking it?

U Did any of our scheduled Postman monitors fail over the weekend?

Looking at the monitor data, your 'Checkout Flow Monitor' failed consecutively on Saturday morning. The internal log indicates an assertion error mapping a `502 Bad Gateway` status on step 3 (`POST /cart`). All other monitors are currently healthy and passing.

Frequently Asked Questions

01 **Can the Postman MCP help me write better API requests?**

Yes. It pulls the full schema of your collections so your agent knows exactly which headers, methods, and body parameters are required for your specific endpoints.

02 **How do I use Postman MCP to check my production health?**

Your agent can check your scheduled monitors to see the success or failure history of your API health checks without you having to open a separate dashboard.

03 **Can my AI agent see my different environments?**

Yes. It can list your environments and their variables, which helps the agent distinguish between your staging, production, and local setups.

04 **How does Postman MCP handle mock servers?**

It lets your agent identify active mock servers so you can quickly get the correct URLs and simulated responses for your front-end testing.

05 **Can I use this to learn my internal API schemas?**

Yes. By pulling your collections, the agent learns the internal structure of your APIs, making it much more accurate when generating code or documentation.

06 **Does Postman MCP work with Cursor?**

Yes, it works with any MCP-compatible client, including Cursor, Claude, and Windsurf, to bring your API data into your workspace.

07 **Will my AI agent know the difference between staging and prod?**

Yes. Because the agent can read your environment variables, it understands the specific configurations for different stages of your deployment.

08 **Can the AI automatically write code using my internal API documentation?**

Absolutely. If you use `get_collection` the AI unpacks the entire Postman hierarchy. Combine this by asking the AI to 'write a Python script to hit my Users Endpoint' and it will natively respect your headers, payload requirements, and auth settings without any context copy-pasting.

09 **How does the agent handle environments like production vs staging variables?**

The agent can call `list_environments` exposing active configurations inside your workspace. If a collection points to `{{base_url}}`, the AI reads your environments array to resolve exactly what URLs or access keys map to staging versus production natively.

10 **Can I query test success rates via AI instead of dashboards?**

Yes. The `list_monitors` connection unrolls the cron checks tied to your Postman collections. The AI inherently sees whether the latest automated integration tests succeeded or failed, making status reports conversational.

Go Live in 60 Seconds

Get your connection token from **cloud.vinkius.com**, then paste the endpoint URL into any MCP-compatible client.

YOUR MCP ENDPOINT

```
https://edge.vinkius.com/[TOKEN]/mcp
```

CLIENT	WHERE TO CONFIGURE
Claude AI	Profile → Customize → Connectors → "+" → Add custom connector → Paste endpoint
Cursor	Settings → Features → MCP Servers → "+ Add New MCP Server" → Type: SSE → Paste endpoint
VS Code	Ctrl/Cmd+Shift+P → "MCP: Add Server" → add `"postman": { "url": "…" }`
Windsurf	MCP Settings → `mcp_settings.json` → Add endpoint URL
ChatGPT	Settings → Tools & plugins → Add MCP server → Paste endpoint
Gemini	Extensions → Add MCP Server → Paste endpoint URL

ASK AN AI ABOUT THIS

Let your preferred AI explain this MCP server

- Ask ChatGPT
- Ask Claude
- Ask Perplexity
- Ask Gemini
- Ask Grok

READY TO CONNECT

Postman is live on Vinkius Cloud.

Get your connection token, paste it into your AI agent, and start building. No SDK. No deployment. Just results.

Start at cloud.vinkius.com →

vinkius.com · support@vinkius.com

INDEPENDENT PLATFORM DISCLAIMER

Vinkius is an independent platform and is not affiliated with, endorsed by, sponsored by, verified by, or otherwise authorized by Postman. All third-party trademarks, logos, and brand names are the property of their respective owners. Their use in this document is strictly for informational purposes to identify service compatibility and interoperability.

DOCUMENT INFORMATION

Generated	July 2026
MCP Server	Postman MCP
Server ID	`019d75f8-83fd-72de-80be-7cbcf6df2fd1`
Platform	Vinkius Cloud for AI Agents
Endpoint	`https://edge.vinkius.com/{token}/mcp`

LICENSE & USAGE

This document is generated automatically by the Vinkius PDF Engine. Content reflects the MCP server configuration at the time of generation and may change as updates are deployed. For the most current information, visit vinkius.com/mcp/postman.