MCP SERVER | NO CODE | CLOUD HOSTED

Unleash MCP

Audit feature flags, environments, and user contexts instantly.

Unleash connects your AI agent directly to your feature flag system. It lets you manage complex product rollouts—from auditing environments to checking if a specific user sees a new beta feature—all through natural conversation. You can evaluate flags, list projects, and monitor usage metrics without ever opening the Unleash UI.

A+ Quality Score 100/100

feature-flags | feature-management | deployment-strategies | rollout-control | environment-management | software-development

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/unleash-feature-toggles

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.

Unleash (Feature Toggles) MCP

11 tools available
Cloud-hosted on Vinkius

Managing features across different parts of your application is complicated. You need to know if 'new-dashboard-v2' should only show up for internal users or if it’s rolled out globally, depending on where the request comes from. This MCP gives your agent full control over that process.

It lets you audit your entire feature setup—listing all projects and environments configured across your infrastructure. You can verify targeting rules by listing users and segments, checking who is supposed to see what. Need to know if a flag works for the backend or only on the client side? The agent handles it. Plus, you don't have to manually report usage metrics; you can send that data straight back through your AI client, making sure everything stays synced. When you connect this MCP via Vinkius, your entire feature lifecycle becomes conversational.

Core Capabilities

01 — Audit Infrastructure Layout

List every project, environment, and segment defined in your Unleash system to understand the full scope of your feature management.

02 — Determine Feature Visibility by Context

Evaluate which features are enabled for a specific user, or based on client properties, simulating how your application will render flags in real time.

03 — Manage and Inspect Flag Statuses

Get comprehensive details on all feature flags within a project, verifying their current rollout status and strategy configurations.

04 — Track Usage Metrics

Report flag usage data from both backend SDKs and frontend clients directly through the agent's tools.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/unleash-feature-toggles** — connect your AI agent in three steps.

01 Subscribe to this MCP, then enter your Unleash API URL and the required API Token (whether it's for Admin, Client, or Frontend access).

02 Your AI client authenticates with the platform, giving your agent full read/write control over feature flag data.

03 You simply ask your agent questions—like 'What flags are active for user 123?'—and get immediate answers backed by your Unleash source of truth.

The bottom line is you gain an API-driven conversational interface to manage complex, mission-critical feature rollouts without writing code or navigating web UIs.

Built For

DevOps engineers who get frustrated having to click through multiple dashboards just to check a flag's status. Product Managers who need instant confirmation of rollout scope before announcing a feature, and Software Engineers who want to verify flag evaluation context right from their editor.

DevOps Engineer

Quickly auditing environments and segments across multiple projects without leaving the command line or chat window.

Product Manager

Checking the status of a feature toggle's rollout strategy across different user groups or markets for a planned launch.

Software Engineer

Verifying flag evaluations and context properties directly while coding, ensuring features behave correctly before testing even begins.

What Changes When You Connect

01 Stop relying on guesswork. You can check the exact status of a feature flag for a specific user context using the `get_frontend_features` tool, verifying visibility before code deployment.

02 Gain total infrastructure oversight by running `list_projects`, `list_environments`, and `list_segments` directly through your agent—all without leaving your chat interface.

03 Automate metric reporting. Instead of manually compiling usage data, you can use `report_client_metrics` or `report_frontend_metrics` to feed live analytics back into the system.

04 Simplify user validation. The agent lets you call `list_users` and `list_segments`, allowing you to quickly verify targeting rules for complex rollouts.

05 Speed up development cycles. You can use `get_client_features` to audit the entire feature set on the server side, making sure all flags are accounted for before a major release.

Real-World Applications

Checking Rollout Scope for a Beta Feature

A PM needs to confirm if the 'new-dashboard-v2' feature is visible only to internal users. They ask their agent, and it uses `get_frontend_features` to check the flag status using specific user properties, providing an instant yes/no answer without needing a sandbox environment.

Auditing Multi-Environment Configuration

An SRE suspects a staging environment is configured incorrectly. They use `list_environments` and then `list_projects` to quickly map out the entire infrastructure layout, identifying which project belongs to which deployment stage.

Verifying Backend Feature Availability

A developer needs to know if a certain feature is enabled for the backend API. They ask their agent, and it invokes `get_client_features`, returning the full server-side flag status list immediately.

Tracking Live User Behavior

The team needs to track how often users interact with a new checkout flow. An engineer uses `register_frontend` first, then calls `report_frontend_metrics`, ensuring the usage data flows back into their analytics dashboard.

Patterns to Avoid

Assuming Flag Status

✗ AVOID

A developer assumes a feature is enabled because they saw it in documentation, but fail to check its current status against active user contexts.

✓ INSTEAD

Always verify the flag state using `get_frontend_features` by providing the target User ID and relevant context properties. This guarantees you're working with the live system truth.

Manually Listing Everything

✗ AVOID

A PM has to jump between the Environments tab, Projects list, and Users section of the Unleash UI just to get an overview.

✓ INSTEAD

Use `list_environments` combined with `list_projects` through your agent. You get a single, comprehensive audit report in one conversation.

Missing Context on Reporting

✗ AVOID

The team reports usage metrics without specifying if the data came from the client or the backend SDK.

✓ INSTEAD

Be precise: use `report_client_metrics` for backend data, and `register_frontend` followed by `report_frontend_metrics` for web-based user activity.

The Right Fit

Use this MCP if your primary pain point is the complexity of feature governance. You need to programmatically audit, evaluate, or report on feature flag states across multiple contexts (client vs. server). This tool shines when you need to answer questions like, 'For user X in environment Y, should feature Z be visible?'

Don't use this if your goal is simple CRUD operations outside of flags—like just managing API keys or sending emails; those require a

messaging MCP. Also, don't use it if you only need to list projects once and never check status again; simply reading the `list_projects` tool might suffice.

However, you must use this if you need to combine multiple actions: checking project details (`list_projects`), verifying segments (`list_segments`), AND evaluating flags for a user context (`get_frontend_features`). The combination of tools is what makes it powerful.

Auditing your feature rollout status used to mean jumping between five different tabs.

Today, checking the full scope of a new release is tedious. You have to navigate to the Project list, then check the environments, and for every single flag, you might need to cross-reference segments or users just to see who gets access. It's clicking through dashboards until your finger hurts.

With this MCP, that entire audit becomes a conversation. Ask your agent to 'List all projects and their associated environments.' You get the full infrastructure map instantly. What changes is that you stop navigating; you start asking.

Unleash (Feature Toggles) gives you real-time feature context.

Previously, if a developer wanted to know how the code would behave for a specific user in a test environment, they had to manually simulate that user's properties and run checks. This was slow and error-prone.

Now, you can pass context variables directly to your agent. By using `get_frontend_features` , you tell your AI client exactly who the user is and what their profile looks like, so it evaluates the flags correctly, every single time.

Unleash (Feature Toggles): 11 Tools Available

These tools give your AI agent programmatic access to every core function of your feature flag system, allowing you to audit configurations and track usage data directly.

#	TOOL	DESCRIPTION
01	`list_environments`	Retrieves a comprehensive list of all operational environments configured in Unleash.
02	`list_project_features`	Lists every single feature flag defined within a specific project ID.
03	`list_projects`	Fetches an exhaustive list of all projects managed by Unleash.
04	`list_segments`	Retrieves details on every user segment configured in your feature management system.
05	`list_users`	Provides a list of all individual users and their associated IDs within Unleash.
06	`get_client_features`	Fetches the complete set of feature flags and strategies for server-side evaluation.
07	`report_client_metrics`	Sends flag usage metrics data reported from a backend SDK instance.
08	`register_client`	Registers and authenticates a new backend SDK instance for metric reporting.
09	`get_frontend_features`	Determines which feature flags are enabled by optionally providing context like user IDs or properties.
10	`report_frontend_metrics`	Sends flag usage metrics data reported from a frontend SDK instance.
11	`register_frontend`	Registers and authenticates a new frontend SDK instance for metric reporting.

See It in Action

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

U List all Unleash projects and their descriptions.

I've retrieved the projects. You have 3 projects: 'Default' (Main project), 'Mobile-App' (Flags for iOS/Android), and 'Beta-Testing' (Experimental features).

U What feature flags are enabled for user 'user_88' in the frontend?

Evaluating flags for 'user_88'... The following flags are enabled: 'new-dashboard-v2', 'dark-mode-beta', and 'premium-checkout'. Flags like 'experimental-chat' remain disabled for this user context.

U Show me all feature flags in the 'Mobile-App' project.

In the 'Mobile-App' project, I found 4 flags: 'biometric-login' (Enabled), 'offline-mode' (Disabled), 'push-notifications-v3' (Enabled), and 'legacy-sync' (Disabled).

Frequently Asked Questions

01 **How do I check if a feature flag works on the backend with Unleash (Feature Toggles) MCP?**

You use the `get_client_features` tool. This fetches all feature flags and strategies specifically for server-side evaluation, giving you the definitive backend status.

02 **Can I list user data using Unleash (Feature Toggles) MCP?**

Yes, use the `list_users` tool. It fetches a comprehensive list of all users configured in your feature management system so you can verify targeting rules.

03 **What if I need to track usage data from my mobile app? Should I use Unleash (Feature Toggles) MCP?**

Yes. You should register the client using `register_client` and then send usage metrics with `report_client_metrics` to ensure your backend SDK is reporting accurately.

04 **Is Unleash (Feature Toggles) MCP better than just checking flags in the UI?**

Absolutely. The agent allows you to chain together multiple checks—like listing projects, then checking flag status for a segment—in one go, which is impossible manually.

05 **How do I ensure my frontend metrics are tracked correctly with Unleash (Feature Toggles) MCP?**

First, use `register_frontend` to authenticate the web instance. Then, pass usage data via `report_frontend_metrics` for accurate front-end tracking.

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 `"unleash-feature-toggles": { "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

Unleash (Feature Toggles) 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 Unleash (Feature Toggles). 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	June 2026
MCP Server	Unleash (Feature Toggles) MCP
Server ID	`019e3902-039a-71b2-97c4-7a329a71a79f`
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/unleash-feature-toggles.