MCP SERVER | NO CODE | CLOUD HOSTED

Strapi MCP

Manage content types and assets via conversation.

Strapi MCP connects your AI agent directly to a headless CMS. You can programmatically read content type schemas, fetch existing data entries, create new structured content posts, update records, and upload media assets—all through conversation.

A+ Quality Score 100/100

open-source-cms | api-development | schema-design | media-orchestration | rest-api | content-types

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/strapi

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.

Strapi MCP

9 tools available
Cloud-hosted on Vinkius

This connector lets you treat Strapi like an exposed API endpoint inside your conversational workflow. Instead of navigating admin dashboards or writing boilerplate code, you tell your agent what needs to happen with your site's content. It reads the structure of your entire CMS, letting you see every available field and collection type instantly. You can then ask it to draft a new article by generating all necessary JSON parameters, or update pricing data across multiple product entries without leaving your chat window. If you're building complex systems that rely on accurate content management, connecting via Vinkius makes Strapi's entire ecosystem accessible from one place, letting you manage everything from initial schema discovery to final asset deployment.

Core Capabilities

01 — View Content Schemas

See every top-level content type and its specific fields defined across your CMS.

02 — Browse Existing Data

Retrieve lists of entries for any specified content type, allowing you to audit or analyze current records.

03 — Read Specific Content Details

Get the full details and metadata for a single, specific piece of content using its unique ID.

04 — Build and Save New Entries

Generate structured JSON payloads to create brand new records across various content types.

05 — Modify Existing Content

Update specific fields of a content entry without having to delete and recreate the entire record.

06 — Manage Media Assets

List existing media files in the library or upload new external images for use in your entries.

One Click on Vinkius — From Prompt to Execution

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

01 First, define this MCP as an active integration within your configuration environment.

02 Next, you must bind your deployed Strapi base URL and a verified API key from your admin panel into the parameter matrix.

03 Finally, state your objective in plain language; for example: "List all content types, then fetch the details for 'products' that need price updates."

The bottom line is you give it credentials and a goal, and it executes the necessary API calls to achieve that outcome.

Built For

This is for the backend architect who spends too much time context-switching between development tools and CMS dashboards. It's for the technical marketer who needs to execute content changes programmatically, not manually. If you deal with structured data or managed web content daily, this MCP saves your sanity.

Backend Architect

Validates complex JSON payloads and structural configurations instantly across components without logging into the Strapi UI.

Technical Content Manager

Drives content updates by creating or modifying entries, ensuring consistency and scale for large-scale site publishing.

DevOps Engineer

Automates tedious metadata adjustments and audits structural dependencies rapidly within isolated agent loops to maintain platform integrity.

What Changes When You Connect

01 Schema Discovery: Use `list_content_types` to instantly map out your entire CMS structure. You get a full picture of available data models without opening any developer console.

02 Content Creation at Scale: Instead of manual form filling, you tell the agent to construct content using `create_entry`, generating complex JSON payloads and saving them immediately.

03 Media Management: Seamlessly handle visuals by running `list_assets` or feeding external dependencies into the system via `upload_media_asset`. Your assets are always available for linking.

04 Data Integrity: Protect your live site using `list_cms_users` to audit who has access. This ensures you only work with authorized accounts when making changes.

05 Efficient Updates: Don't rewrite entire posts; use `update_entry` to precisely target and modify specific fields, saving time and minimizing the chance of errors.

Real-World Applications

The Global Content Audit

A technical marketer needs to check if all 'article' types have a proper author assigned. They ask their agent to run `list_entries` for articles, and the agent returns a list of IDs, allowing them to programmatically verify missing metadata fields.

Fixing Broken Product Pages

A developer discovers that several 'product' listings are missing their current inventory count. They use `get_entry_details` on a sample item and then instruct the agent to update all necessary entries via `update_entry`.

Adding New Visual Content

A marketing team gets new banner images from an external source. Instead of manually uploading them, they ask the agent to `upload_media_asset`, which securely pulls the image and makes it available for use in posts.

Schema Validation Check

A backend architect wants to ensure a new feature field is correctly implemented. They first run `list_content_types` to confirm the structure, then request that the agent validate the payload using `create_entry` before committing any changes.

Patterns to Avoid

Trying to bypass data retrieval

✗ AVOID

Asking the AI to 'just look at the database' without specifying which content type or collection. This leads to vague, unformatted results.

✓ INSTEAD

Always start by listing available types using `list_content_types` to define your scope. Then, use `list_entries` and specify the plural ID you want to analyze.

Manually updating every record

✗ AVOID

When 50 products need a price increase, manually finding each entry in the CMS admin dashboard and changing the field one by one.

✓ INSTEAD

Use `list_entries` to gather all product IDs, then issue a single command for the agent to run `update_entry` across the entire batch of records.

Assuming file access

✗ AVOID

Telling the AI to 'use this picture' without providing a source URL. The process stops because the system can’t fetch the external data.

✓ INSTEAD

Always provide a public file URL and ask the agent to run `upload_media_asset`. This fetches the dependency and makes it available in your library.

The Right Fit

Use this MCP if your workflow requires deep, structured interaction with a headless CMS's data layer. You need to read schema definitions (`list_content_types`), batch create records (`create_entry`), or programmatically update specific fields across many entries (`update_entry`). Don't use it if you simply want to view content in a human-readable dashboard format; for that, the native CMS interface is better. Instead, use this MCP when your goal is automation: validating structure, migrating data, or acting as an API wrapper through natural language commands.

The Headache of Context Switching

Today, managing structured content means jumping between tabs. You start in the CMS dashboard to check a schema definition, then switch to your code editor to write JSON payloads, and finally open a spreadsheet just to audit user IDs. It's constant copy-pasting and switching gears.

With this MCP, you keep everything inside your agent chat. You tell it: 'What are my content types?' The response is immediate structure data. Then you say: 'Now create an article with these parameters.' The entire cycle—discovery, drafting, and saving—happens in one continuous conversation.

Strapi MCP Gives You Programmatic Control Over Your Content

You no longer need to manually check if a new article has the correct metadata or if an image asset is linked properly. The agent handles that validation automatically when you ask it to `get_entry_details` or when you run `list_assets`.

Your AI client acts like an API client, giving you precise control over your content lifecycle. You're not just drafting text; you're executing database actions.

Strapi MCP: 9 Tools for CMS Management

Use these tools to read schemas, list content, retrieve specific data, create new records, modify existing ones, and manage all your media assets programmatically.

#	TOOL	DESCRIPTION
01	`create_entry`	Creates a new content entry for a specific type using JSON field parameters.
02	`delete_entry`	Permanently and irreversibly removes an existing content entry from the system.
03	`get_entry_details`	Retrieves all stored field details for a specific, known content entry ID.
04	`list_assets`	Generates a list of every media asset currently stored in the Strapi Media Library.
05	`list_content_types`	Outputs a complete roster of all content types available (collections and single types).
06	`list_entries`	Lists multiple entries belonging to a specified content type.
07	`list_cms_users`	Provides a list of all currently registered users within the CMS system for auditing purposes.
08	`update_entry`	Changes specific fields on an existing content entry using its IDs and new data values.
09	`upload_media_asset`	Takes a public file URL, fetches the image, and uploads it to the Media Library.

See It in Action

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

U Review my Strapi content types and show the schema for 'product'.

Connected to Strapi.
Content Types (`list_content_types` **):**
- `api::user.user`
- `api::product.product`

Schema for `api::product.product`:
- `title` (string), `price_tag` (decimal), `sku_code` (string), `available_inventory` (integer).

U Construct a newly formatted post about system updates in the 'articles' content type.

Building JSON parameters...

Executing Task (`create_entry` **):**
Successfully created entry ID: `402` in the `api::article.article` collection. Title set to 'System Updates V2', formatted correctly.

U Upload a new promotional image dependency securely into the Media Library.

Accessing the Strapi Media Library securely...

Asset Deployed (`upload_media_asset` **):**
The image `promo_banner_spring.jpg` has been successfully uploaded and is assigned the dynamic referential ID: `1005`. It's ready to be bound to your entries.

Frequently Asked Questions

01 **How do I view all the available data models using Strapi MCP?**

You run `list_content_types`. This immediately gives you a comprehensive list of every collection and single type defined in your CMS, helping you map out your entire data architecture.

02 **Can I update multiple entries at once using Strapi MCP?**

Yes. After listing the content types with `list_content_types`, you can use `list_entries` to gather IDs, and then command the agent to run `update_entry` across all necessary records.

03 **What is the difference between list_assets and upload_media_asset in Strapi MCP?**

Use `list_assets` when you need to see what images are already stored on the platform. Use `upload_media_asset` when you have an external image URL that needs to be fetched and added to your media library.

04 **Is deleting content irreversible with Strapi MCP?**

Yes, `delete_entry` is designed as a permanent action. The system explicitly warns that this action cannot be undone, so proceed carefully.

05 **Can I check who has access to the CMS using Strapi MCP?**

You use the `list_cms_users` tool. This function pulls a list of all registered users in your CMS environment for auditing and security checks.

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 `"strapi": { "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

Strapi 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 Strapi. 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	Strapi MCP
Server ID	`019d760d-d7f5-7391-adda-1b7bdc9b9a6e`
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/strapi.