MCP SERVER | NO CODE | CLOUD HOSTED

Kontent.ai MCP

Control content publishing from your chat client.

Kontent.ai (Enterprise Headless CMS) MCP lets you manage complex enterprise content workflows directly through your AI client. You can audit schemas, create new content containers, update specific language texts for items, and push variants live—all without logging into the Kontent UI.

A+ Quality Score 100/100

content-orchestration | enterprise-cms | content-lifecycle | language-variants | content-delivery

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/kontentai-enterprise-headless-cms

SCAN TO VIEW

The infrastructure that powers AI agents in the real world.

Vinkius connects AI to the world's software through secure, enterprise-grade infrastructure — enabling real-world execution at scale, built on the Model Context Protocol (MCP).

Your AI Connections Run Through Vinkius Cloud

The world's largest managed MCP catalog

Vinkius is the cloud infrastructure where AI agents connect 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.

Kontent.ai (Enterprise Headless CMS) MCP

10 tools available
Cloud-hosted on Vinkius

This MCP gives your agent full control over your Kontent.ai CMS. Instead of clicking through menus to manage content, you talk to it. You can discover exactly what fields a piece of content needs by inspecting content type definitions. Need to update text for one language? Your agent handles the `upsert_language_variant` call and places that version into Draft status immediately. When everything is ready, running the publish workflow transitions those specific variants to Published, making them live via your delivery APIs—all in a conversation. Connecting this MCP through Vinkius means you connect once from Claude, Cursor, or any compatible client and get access to this entire content lifecycle management system.

Core Capabilities

01 — Audit Content Structure

Inspect available fields and structural requirements for any registered content type.

02 — Manage Item Containers

List existing content items or create/update the top-level containers that hold all your content.

03 — Update Multilingual Variants

Change specific text fields for a single language (like Portuguese) within an item, moving it to Draft status automatically.

04 — Publish Content Live

Transition specific content variants from Draft into Published status, making them instantly available on the front end.

05 — Govern Taxonomy and Assets

List all organizational tags or audit uploaded media files to get their precise identifiers and cloud URLs.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/kontentai-enterprise-headless-cms** — connect your AI agent in three steps.

01 Subscribe to this MCP, providing your Kontent.ai Environment ID and Management API Key.

02 Your agent connects using these credentials, granting it full programmatic access to the CMS backend.

03 You issue a command—for example, 'Update the article body for English'—and the system executes the necessary content writes or publishes.

The bottom line is that you manage your entire content lifecycle through natural conversation, never needing to touch the Kontent UI again.

Built For

Content Architects and Digital Editors need this. They're tired of juggling multiple browser tabs, manually copying asset IDs, and waiting for a developer to make small content updates. This MCP lets them manage the entire publishing workflow right where they work.

Digital Content Editor

Needs to update localized body copy or change metadata across dozens of articles quickly, ensuring each language variant is correctly drafted before review.

Content Architect

Must audit the entire content type schema and taxonomy tree across multiple projects to ensure consistency before a new feature launches.

Frontend Developer

Requires listing uploaded media assets or inspecting content types to verify field names and asset identifiers for accurate mapping in their downstream applications.

What Changes When You Connect

01 Go beyond simple content updates. You can audit the full schema using `get_content_type` and verify exactly what fields are available before writing any code.

02 Never worry about stale links again. Use `list_assets` to get precise identifiers and cloud URLs for all media, ensuring your frontend always points to the right source.

03 Manage content status changes instantly. Instead of manually updating a variant in the UI, calling `publish_variant` moves the item live with one command.

04 Maintain multilingual integrity easily. You can run `upsert_language_variant` to update text for Portuguese or French without affecting the English copy or leaving the record un-drafted.

05 Audit your entire site structure by using `list_taxonomies` and `list_content_types`. This helps content architects verify that all necessary organizational tags exist across environments.

Real-World Applications

Preparing a Global Launch Campaign

The marketing team needs to launch an article in five languages. They first use `list_items` to find the main container, then call `upsert_language_variant` for each language's copy, and finally run `publish_variant` sequentially for all versions to ensure a perfect, simultaneous global rollout.

Debugging a Broken Homepage Feature

The developer realizes the homepage hero image is using an outdated asset. They use `list_assets` to get the current identifier and then pass that ID into the content type schema check using `get_content_type` before updating the item container via `upsert_item`.

Auditing Content Schema Drift

The content architect needs to verify if a new department has added mandatory fields to the 'Article' type. They call `get_content_type` and compare the returned structure against known schemas, ensuring no required element is missing.

Mass Content Migration

The team needs to update five different content items (using `get_item` first) and change their associated category tags. They use `upsert_language_variant` for the text changes, followed by calls to ensure the correct taxonomy groups are assigned.

Patterns to Avoid

Assuming content is live

✗ AVOID

A developer updates a language variant using `upsert_language_variant` and assumes it's ready for production, but forgets to run the publishing step.

✓ INSTEAD

Always follow up any text update with a call to `publish_variant`. The `upsert_language_variant` function only places content into Draft status; you must explicitly transition it using `publish_variant`.

Updating without knowing the structure

✗ AVOID

A user tries to add a custom field (e.g., 'Product SKU') to an article because they think it should be there, but doesn't know if the content type supports it.

✓ INSTEAD

First, run `get_content_type` on the specific schema. This call provides the definitive list of available fields and parameters, preventing data errors.

Updating everything at once

✗ AVOID

Trying to update an item container (`upsert_item`) but also changing its text content in the same command, which can lead to overwrites or failure.

✓ INSTEAD

Keep your actions separate. Use `upsert_item` only when modifying the shell/metadata, and use `upsert_language_variant` only for touching the actual body copy.

The Right Fit

Use this MCP if your core problem involves managing complex content lifecycles across multiple languages or environments. Specifically, you need to audit structured data (like field schemas via `get_content_type`), manage versioning, and programmatically publish content status (Draft to Published). Don't use this if all you need is a simple key-value store connection; for that, an object

database connector would be better. Also, don't use it if your only goal is retrieving a list of items—while `list_items` works, remember that the real power comes from the subsequent actions like updating variants or publishing.

Managing content across different languages and versions used to be an administrative nightmare.

Today's process involves logging into a CMS, navigating deep menu structures, manually switching between tabs for English, German, Spanish, and Japanese. You copy the item codename from one screen, paste it into another form, update the text block, then repeat the whole tedious cycle for every single language variant.

With this MCP, your agent handles the complexity. You simply tell your client to 'Update the body copy for German' on a specific article. The system finds the right item, targets the correct language variant, updates the content, and leaves it in Draft status—all without you clicking anything.

Kontent.ai MCP gives your agent granular control over every piece of metadata.

You no longer have to guess what data fields are available or where an asset lives. You can ask the system to run `get_content_type` to map out every required field, or use `list_assets` to get a definitive list of all media IDs needed for your site build.

This shift means you move from being limited by the UI's flow to having total conversational control. You define the process—from schema discovery to final publication—and it executes.

Kontent.ai (Enterprise Headless CMS) 10 Tools

Use these tools to read the structure of your content types, list assets, manage taxonomies, or perform full content writes and publications via your AI agent.

#	TOOL	DESCRIPTION
01	get_item	Retrieves the metadata for a single piece of content using its unique codename.
02	get_taxonomy	Fetches detailed information, including all nested terms, for a specific organizational tag group.
03	get_content_type	Retrieves the precise structural definition and required fields for any content type schema.
04	list_assets	Lists all uploaded media files and documents, providing their identifiers and cloud URLs.
05	list_items	Retrieves a list of every top-level content container present in the Kontent.ai project space.
06	list_taxonomies	Lists all available hierarchical taxonomy groups, similar to site tags or categories.
07	list_content_types	Retrieves a list of every content type schema defined and registered in the environment.
08	publish_variant	Changes a specific language variant's status from Draft to Published, making the content immediately live via APIs.
09	upsert_item	Creates or updates the top-level container for a piece of content without changing its actual text fields.
10	upsert_language_variant	Updates the actual body copy and structured data elements for an item, placing the new version into Draft status.

See It in Action

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

U List all content items in our project

I've retrieved your content items. Highlights include 'Homepage' (codename: homepage), 'About Us' (about_us), and 'Q4 Roadmap' (q4_roadmap). Would you like to check the language variants for any of these?

U Publish the 'default' variant for item 'q4_roadmap'

Publishing variant... Done. The 'default' language version of 'q4_roadmap' has been transitioned to the Published stage. It is now live and available via your delivery endpoints.

U What are the structural fields for the 'Article' content type?

The 'Article' content type (codename: article) contains 5 fields: 'title' (Text), 'slug' (URL slug), 'body' (Rich text), 'author' (Content relation), and 'hero_image' (Asset). I can help you create a new item implementing this schema.

Frequently Asked Questions

01 **How do I publish an item using Kontent.ai MCP?**

You use the `publish_variant` tool. This function takes a specific content container and language variant, changing its status from Draft to Published so it's live on your delivery APIs.

02 **What is the difference between list_items and list_content_types in Kontent.ai MCP?**

`list_items` gives you a roster of actual content pieces (e.g., 'About Us'). `list_content_types` tells you about the *structure* of content, like defining that every article must have a 'title' and 'body'.

03 **Can I update text for only one language using Kontent.ai MCP?**

Yes, absolutely. You use `upsert_language_variant`. This tool lets you target the actual content fields of an item while specifying exactly which language (e.g., Portuguese) needs updating.

04 **Does Kontent.ai MCP help me find asset IDs?**

Yes, running `list_assets` will retrieve a list of all uploaded media and document files, providing you with the necessary cloud URLs and identifiers needed for your frontend applications.

05 **What should I do if I need to create a new piece of content?**

First, run `upsert_item` to create the top-level container. Then, use `upsert_language_variant` for each language's text, and finally publish it.

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 `"kontentai-enterprise-headless-cms": { "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

Kontent.ai (Enterprise Headless CMS) 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 Kontent.ai (Enterprise Headless CMS). 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	Kontent.ai (Enterprise Headless CMS) MCP
Server ID	`019d75c3-18b2-70e9-9c9d-02c1ab53a724`
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/kontentai-enterprise-headless-cms.