MCP SERVER | NO CODE | CLOUD HOSTED

Magnolia CMS MCP

Control your content structure and data delivery.

Magnolia (Enterprise Headless CMS) connects your AI agent directly to an enterprise JCR repository. Control complex content structures, audit component schemas, and manage delivery layer endpoints using natural conversation. Provision nodes, execute workspace commands, and perform deep structural queries without manual REST calls.

A+ Quality Score 100/100

jcr-repository | content-orchestration | digital-experience | node-management | enterprise-cms | api-driven-content

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

Magnolia (Enterprise Headless CMS) MCP

10 tools available
Cloud-hosted on Vinkius

Managing a large-scale headless CMS is hard enough without constantly writing boilerplate API requests. This MCP lets you talk to your Magnolia instance like it's a database of pure content structure. You can ask your agent to list active workspaces or find the JSON mapping for a specific delivery endpoint, all through conversation. Need to audit what fields a template expects? Just ask, and the system extracts the component definitions. It’s true structural control over your JCR repository, allowing you to provision new nodes or even trigger publishing workflows just by chatting with it. If managing complex content requires deep visibility into how your site builds its pages, this MCP is essential for any developer working across the Vinkius catalog.

Core Capabilities

01 — Audit and Understand Content Structure

Examine template schemas and discover all active JCR workspaces to map out where content lives.

02 — Manipulate Nodes and Data Payloads

Create, delete, copy, or update content nodes in the repository using simple commands instead of complex payloads.

03 — Query Complex Content Relationships

Execute deep queries across delivery endpoints to retrieve pure JSON mappings and identify specific property values.

04 — Manage Operational Workflows

Trigger platform commands, such as validation checks or content publishing cycles, directly from your agent.

One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to this MCP and provide your Magnolia Host details, port number, and basic authentication credentials.

02 Connect your preferred AI client (Claude, Cursor, etc.) to the Vinkius catalog using the provided connection string.

03 Start by asking your agent a question like, 'List all active JCR workspaces' or 'Show me the schema for this template,' and execute commands naturally.

The bottom line is you get full, conversation-based access to your enterprise CMS content layer without ever touching an API playground.

Built For

This MCP is for the senior technical staff who spend their days debugging complex data flows and wrestling with deep repository structures. If you're tired of manually running REST calls just to figure out what a content node contains, this is for you.

Headless Developer

You use it to test delivery API results and verify JCR properties by talking to your agent instead of writing repetitive manual REST tools.

Content Architect

You rely on this MCP to audit component schemas and manage content trees across multiple, complex site environments efficiently.

Digital Operations Engineer

You monitor workspace health and run automated publishing commands across your entire Magnolia instance in a single conversation flow.

What Changes When You Connect

01 Stop writing boilerplate REST calls. Just ask the agent to run `mg.get_delivery_node` to get the pure JSON mapping for any endpoint path, instantly verifying the data structure you need.

02 Avoid manual schema audits. Use `mg.get_template_schema` to automatically extract all required component and page template definitions, saving hours of console clicking.

03 Manage content lifecycles without risk. You can use `mg.execute_workspace_command` to trigger publishing workflows or validation checks across the entire system in one go.

04 Never lose context visibility again. Use `mg.list_jcr_workspaces` to see every active data domain, from your website content to your DAM assets.

05 Need to move content? You can use `mg.copy_delivery_node` to copy nodes while the system verifies that all structural matching and delivery logic remains intact.

Real-World Applications

Debugging a broken page layout

A developer notices a page is missing data. Instead of manually querying multiple endpoints, they ask their agent to run `mg.get_delivery_node` for the exact path and get the JSON mapping. The response immediately shows that a required property field is null, pinpointing the source issue.

Setting up a new content type

A content architect needs to know what fields are available for a blog post. They ask their agent to use `mg.get_template_schema`. The system returns a detailed list of required fields and types, allowing them to build the node correctly on the first try.

Publishing content across environments

A digital ops team needs to move approved draft content live. They instruct their agent to use `mg.execute_workspace_command` and confirm that the publishing workflow completes successfully, moving content through its lifecycle automatically.

Identifying orphaned assets

A developer suspects some nodes are pointing to outdated data structures. They ask their agent to query using `mg.query_delivery_nodes` to trace explicit payload criteria against cloud logs, finding the exact failing dependency.

Patterns to Avoid

Using generic REST clients

✗ AVOID

Manually hitting endpoints in a browser or an API client to check for a node's properties. This requires guessing paths and manually interpreting JSON responses.

✓ INSTEAD

Use the MCP agent directly. Tell it, 'Get the delivery node for path X.' The `mg.get_delivery_node` tool handles the complex JCR logic so you just get the clean, readable JSON mapping.

Writing custom scripts for schema checks

✗ AVOID

A developer has to write a multi-step script that calls multiple endpoints and then compares schemas manually to see what fields are available on a template.

✓ INSTEAD

The agent handles this with one command: `mg.get_template_schema`. It returns the complete, structured ruleset in text format immediately.

Ignoring workspace separation

✗ AVOID

A developer tries to update content but isn't sure if they are working on the staging site or the live production environment.

✓ INSTEAD

Use `mg.list_jcr_workspaces` first. This identifies all active workspaces (website, dam, etc.), confirming which context domain you need before making any changes.

The Right Fit

Use this MCP if your work requires deep, structural interaction with the JCR repository—meaning you aren't just reading simple content; you are managing how that content is structured, delivered, and published. This tool is mandatory when you need to audit component schemas (`mg.get_template_schema`), verify complex delivery paths (`mg.get_delivery_node`), or manage platform workflows (`mg.execute_workspace_command`). Don't use this if your only task is reading a single, simple article by its public URL; in that case, a standard content API client is enough. You need this MCP when you are fixing the *system* or auditing the *structure*, not just consuming the final output.

The headache of manual CMS data audits

Right now, checking how a piece of content is structured means jumping between developer consoles, writing complex query parameters for every property you want to see, and manually cross-referencing endpoints to ensure the data flow works. You spend half your time just figuring out what fields are available instead of building anything.

With this MCP, you treat your CMS like a conversation partner. Instead of juggling console tabs and boilerplate code, you simply ask your agent about the structure—whether it's listing active workspaces or defining component requirements—and get a clean answer back.

The Magnolia (Enterprise Headless CMS) MCP delivers structural control

You stop spending time on tedious tasks like manually running `mg.get_delivery_children` to check for nested properties or writing repetitive commands just to see which nodes exist in a given scope. The agent handles the deep repository logic.

What's different now is that you get immediate, conversational insight into your entire content graph. You control it at the architectural level, not just the presentation layer.

Magnolia (Enterprise Headless CMS) 10 Tools

Use these ten tools to orchestrate every aspect of your Magnolia content structure, from querying specific node data to executing global workspace commands.

#	TOOL	DESCRIPTION
01	`mg.copy_delivery_node`	Copies content nodes while verifying the structural logic that governs data delivery.
02	`mg.create_cms_node`	Creates new, highly-available content nodes and writes custom JSON payload models natively.
03	`mg.wipe_cms_node`	Irreversibly deletes specified content nodes, permanently dropping all associated database bytes.
04	`mg.execute_workspace_command`	Runs automated validation checks and platform logic across your entire configured workspace.
05	`mg.get_delivery_children`	Extracts properties that drive the nesting structure of content branches.
06	`mg.get_delivery_node`	Retrieves pure JSON mappings for any given endpoint path from the JCR tree.
07	`mg.get_template_schema`	Lists all attached structured rules, showing exactly which fields a content template requires.
08	`mg.list_jcr_workspaces`	Identifies and lists all active JCR workspaces spanning different context domains.
09	`mg.query_delivery_nodes`	Searches through cloud logging to trace specific content payload criteria.
10	`mg.patch_cms_node`	Updates existing nodes, substituting or modifying draft document schemas globally.

See It in Action

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

U Get the node details for path 'tours/bali' from the 'tours_v1' delivery endpoint

Retrieving node '/tours/bali'... I've fetched the JSON mapping. It contains properties like tourName: 'Exotic Bali', price: 1200, and duration: '10 days'. Would you like to see the child nodes or the associated template definition?

U List all active JCR workspaces in our Magnolia instance

I've identified the following workspaces: 'website' (Site content), 'dam' (Assets), 'configuration' (System settings), 'imaging' (Image processing), and 'categorization' (Tags). Which one would you like to explore?

U Show me the schema definition for template 'mgnl-news-article'

Retrieving schema for 'mgnl-news-article'... This template expects 4 main fields: 'title' (Required, Text), 'body' (RichText), 'image' (Link to DAM), and 'author' (Text). I can help you create a new node implementing this structure.

Frequently Asked Questions

01 **How do I check all my Magnolia workspaces using the Magnolia (Enterprise Headless CMS) MCP?**

Use `mg.list_jcr_workspaces`. This tool immediately lists every active JCR workspace, letting you know exactly where your website content, DAM assets, and system configurations are stored.

02 **Can I use the Magnolia (Enterprise Headless CMS) MCP to create new nodes?**

Yes. You can provision new, structured content using `mg.create_cms_node`. This allows you to write complex JSON payloads natively without needing a full API client setup.

03 **What is the best way to find out what fields a template needs?**

Run `mg.get_template_schema`. This tool enumerates all attached structured rules, giving you an explicit list of required fields and properties for any given component or page template.

04 **Does the Magnolia (Enterprise Headless CMS) MCP support content publishing?**

Yes. You can use `mg.execute_workspace_command` to trigger automated validation checks or initiate full publishing workflows, moving content through its lifecycle with a single command.

05 **How do I get the data for a specific page path?**

Use `mg.get_delivery_node`. You provide the endpoint and path, and the tool securely retrieves the pure JSON mapping from the JCR tree for that precise location.

06 **Can I navigate the JCR tree structure through my agent?**

Yes. Use the `mg.get_delivery_children` tool by providing an endpoint and parent path. Your agent will retrieve exclusively the hierarchical descendants, allowing you to understand the branch nesting and property distribution accurately.

07 **How do I audit the required fields for a specific Magnolia component?**

The `mg.get_template_schema` tool parses the YAML definitions of your components. Your agent will list exactly what fields and scalar parameters the template respects, making it easy to verify your content model without opening the code.

08 **Can my agent trigger a content publication command?**

Absolutely. Use the `mg.execute_workspace_command` tool and specify 'activate' or 'publish' as the command name. Your agent will dispatch the JSON payload to Magnolia's command system to transition your content across lifecycle states.

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 `"magnolia-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

Magnolia (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 Magnolia (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	Magnolia (Enterprise Headless CMS) MCP
Server ID	`019d75cb-fb77-705d-9fba-69e5815c4c2d`
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/magnolia-enterprise-headless-cms.