MCP SERVER | NO CODE | CLOUD HOSTED

Syncthing MCP

Manage P2P File Syncs with Conversation.

Syncthing. Take full, conversational control over your private file synchronization networks. This MCP lets you monitor every connected device, check folder completion status across remote nodes, and manage the sync process—all without touching a terminal window. You can verify directory structures remotely or pause specific devices instantly, making it ideal for system admins managing distributed data backups.

A+ Quality Score 98.33/100

file-synchronization | p2p | data-backup | file-management | remote-access | distributed-storage

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

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.

Syncthing MCP

28 tools available
Cloud-hosted on Vinkius

Managing peer-to-peer file synchronization used to mean running multiple command-line tools just to get an overview of what was happening across your network. Now, you connect this MCP and talk to your agent about your infrastructure like natural conversation. You tell it, 'What's the status of my backups?' and it handles the complexity of checking connection health, folder completeness, and device stats. It gives you a single pane of glass view of your entire sync setup.

Whether you need to pause syncing on one machine because of an outage or browse a specific directory path across multiple nodes just to verify file integrity, your agent can execute those commands. This capability moves deep system control out of the terminal and into chat. By connecting this MCP via Vinkius, you get reliable access to manage complex data flows from any compatible AI client.

It's about gaining visibility and granular control over a distributed network—the kind of operational oversight that used to require specialized knowledge and dozens of manual checks.

Core Capabilities

01 — Monitor Device Health

See which devices are connected, their current status, and overall connection metadata.

02 — Control Synchronization Flow

Pause or resume syncing for specific nodes, or restart the entire Syncthing process when needed.

03 — Check Folder Progress

Determine if a folder has completed synchronization and retrieve detailed status reports on database health.

04 — Inspect Data Structure

Remotely list directories matching specific paths across your entire sync network for auditing or verification.

05 — Manage System Settings

Retrieve the full system configuration, check if a restart is required, or even shutdown the service.

One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to this MCP and provide your Syncthing Web UI URL along with the necessary API key.

02 Authorize your AI client to interact with your private P2P sync network data.

03 Ask your agent a question, such as 'Check the status of the marketing folder' or 'Pause syncing on the main backup device', and it executes the required action.

The bottom line is you get deep system control over complex file synchronization operations using plain language prompts.

Built For

This MCP is for infrastructure people who manage critical, multi-node data backups. It targets the sysadmin who dreads manually checking multiple command-line outputs or the power user needing granular control over their home lab setup.

DevOps Engineer

Uses this to monitor distributed sync nodes and validate configurations without ever leaving their primary chat interface.

System Administrator

Checks the health of critical backup services, manages connections, and triggers scans across large-scale multi-device deployments.

Data Manager

Verifies that specific files have synced correctly or checks directory paths on remote machines to confirm data integrity before a major migration.

What Changes When You Connect

01 You instantly gain visibility into the entire sync network. Instead of checking multiple dashboards, you ask for device status and get a consolidated report on connection health via `system_connections`.

02 Avoid manual restarts or complex configuration changes. You can ask your agent to run `get_restart_required` first, ensuring you only restart the service when absolutely necessary, saving time and preventing downtime.

03 Confirm data integrity with minimal effort. By asking your agent to list directories using `system_browse`, you verify file structures across remote nodes without needing SSH into each one.

04 Maintain control over resource usage by pausing or resuming syncs on specific nodes. You can use `system_pause` if a device is experiencing high load, and resume it when the issue clears.

05 Get deep operational stats instantly. Tools like `get_folder_stats` give you comprehensive reports on folder utilization and completion status faster than any manual check.

Real-World Applications

Verifying a Critical Backup Path

The data manager needs to confirm that the 'Q4-financial' directory synced correctly across three different NAS units. They ask their agent to use `system_browse` on specific paths and then run `get_folder_stats`. The agent replies with confirmation of the file count, eliminating manual checks.

Troubleshooting a Frozen Sync

The ops engineer notices one device is stuck. Instead of guessing, they ask their agent to check both `get_device_stats` and `get_events`. The combined data points immediately to a resource bottleneck on the node, allowing them to use `system_pause` until it's fixed.

Pre-Migration Data Audit

A power user needs an inventory of all sync locations before moving data. They instruct their agent to run `get_folders` and then use the resulting list to check the configuration for each one using `get_folder`, building a master checklist in minutes.

Post-Change System Verification

After updating firewall rules, the sysadmin needs to confirm everything is okay. They first use `system_status` and then ask their agent to check device health via `get_health`. If both pass, they know the network is operational.

Patterns to Avoid

Assuming Sync Status

✗ AVOID

Relying on a visual indicator or a basic list to confirm that files have actually reached their target folder.

✓ INSTEAD

Always ask your agent to use `get_db_completion` and `get_folder_stats`. These tools provide concrete percentage metrics, confirming whether the sync process has truly finished.

Over-relying on Single Commands

✗ AVOID

Running only `system_connections` and thinking all systems are green. You miss critical details like current usage or recent errors.

✓ INSTEAD

Combine tool calls. Start with `get_devices`, then follow up by asking the agent to run `get_device_stats` for any device flagged as 'degraded' for a full picture.

Ignoring Configuration Drift

✗ AVOID

Assuming your network configuration is correct after an update and not checking what actually changed.

✓ INSTEAD

Use `get_config` to retrieve the current, authoritative system parameters. If you suspect a change, always run this before applying new settings.

The Right Fit

Use this MCP if your primary pain point is gaining conversational control and visibility over complex, distributed file synchronization networks (P2P sync). You need to know the health status of multiple nodes, manage resource consumption by pausing services, or remotely audit directory structures. Don't use it if you just need simple cloud storage backups; those are handled by dedicated backup tools that don't require deep system control like this. If your goal is only file retrieval without monitoring sync state, a standard

API connector might suffice. But if the *process* of syncing and the *state* of the nodes matter, Syncthing is what you need.

Checking on Your Backup Network Is a Nightmare of Dashboards

Today, checking your multi-node backup network means jumping between multiple web UIs. You check the main dashboard for device status, then open another tab to see folder completion percentages, and finally, maybe you have to run a separate command just to list directories to verify file paths. It's tedious copy-pasting and switching context.

With this MCP, you talk directly to your agent. You ask it, 'What's the status of the Q4 backup?' The agent pulls in device health via `get_devices`, checks folder completion using `get_db_completion`, and verifies file paths—all in one response. You get actionable data without leaving your chat window.

Syncthing MCP Gives You Complete System Control

Manual control used to require knowing which specific CLI command to run, and often required a restart just to make simple changes. If you needed to pause one device while the others kept running, it was complicated.

Now, if you tell your agent to `system_pause` a specific ID, that happens immediately. You manage the network like an operating system—directing power, checking resource usage with `system_status`, and getting immediate confirmation of what changed.

Syncthing: 25 Tools for File Sync Operations

These tools allow you to perform every operational task on your syncthing instance—from checking connection status and listing directories to pausing services or resetting databases.

#	TOOL	DESCRIPTION
01	`system_browse`	Lists directories based on a provided file path.
02	`system_connections`	Retrieves all configured devices and their active connection status.
03	`get_db_completion`	Provides the current synchronization completion percentage for a folder or device.
04	`get_db_file`	Retrieves detailed metadata and status information about one specific file.
05	`scan_db`	Triggers an immediate synchronization scan request for a specified folder.
06	`get_db_status`	Checks the overall database status of a synchronized folder.
07	`get_device_stats`	Gets detailed performance and resource statistics for a connected device.
08	`get_events`	Polls the system to check for recent synchronization or connection events.
09	`get_folder_stats`	Retrieves comprehensive statistics and usage data for a specific synchronized folder.
10	`get_config`	Fetches the complete, current system configuration settings.
11	`get_db_ignores`	Retrieves the list of ignore patterns set for a specific folder's database.
12	`get_device`	Fetches the detailed configuration settings for one specified device ID.
13	`get_devices`	Retrieves a list of all configured devices and their current connection details.
14	`get_folder`	Fetches the detailed configuration settings for one specified folder path.
15	`get_folders`	Retrieves a list of all configured folders and their respective paths.
16	`get_health`	Performs a simple, non-authenticated check to confirm the overall operational status of the instance.
17	`system_pause`	Temporarily halts synchronization activity on specified devices or all devices if no ID is given.
18	`get_random_string`	Generates a strong, random alphanumeric string for use in configuration files.

#	TOOL	DESCRIPTION
19	system_reset	Resets the index database for one or more specified folders.
20	get_restart_required	Checks if a system configuration change mandates a full service restart.
21	system_restart	Initiates a controlled restart of the entire Syncthing service instance.
22	system_resume	Resumes synchronization activity on specified devices or all devices if no ID is given.
23	set_config	Replaces the entire system configuration with a new set of parameters.
24	set_db_ignores	Sets or updates the ignore patterns for a specified folder's database.
25	system_shutdown	Gracefully shuts down the Syncthing service instance.
26	system_status	Gets the current system status and resource usage metrics of the running process.
27	verify_device_id	Validates and formats a raw device ID string into the required operational format.
28	system_version	Retrieves the current software version number of the Syncthing instance.

See It in Action

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

U Show me all connected devices and their current status.

I've retrieved the connection status. You have 3 devices configured: 'Laptop-Main' (Connected), 'Backup-NAS' (Connected), and 'Phone-Sync' (Disconnected).

U Check the system status and resource usage of my Syncthing instance.

The system is healthy. Uptime is 48 hours, memory usage is 156MB, and the current version is v1.27.2. No restart is currently required.

U List the directories inside the path '/data/backups/photos'.

Browsing the path... I found 3 subdirectories: '2023-Vacation', 'Family-Events', and 'Work-Archive'.

Frequently Asked Questions

01 **How do I check if my Syncthing setup needs a restart using the Syncthing MCP?**

You use the `get_restart_required` tool. It checks your current configuration against the running service parameters and tells you definitively if a full system restart is necessary before making changes.

02 **Can I list remote files using the Syncthing MCP?**

Yes, use `system_browse`. You provide the path, and the tool lists matching directories across your entire connected sync network, helping you audit file structures remotely.

03 **What is the difference between `get_folder` and `get_folders` in the Syncthing MCP?**

`get_devices` retrieves a list of all configured devices. In contrast, `get_folders` lists every synchronized folder path, while `get_folder` gets the deep configuration for just one specific path.

04 **How do I check if my backup is finished using Syncthing MCP?**

Use `get_db_completion`. This tool gives you a quantifiable status, reporting the percentage completion and ensuring the folder has fully synced across all connected nodes.

05 **If I change settings, should I use `set_config` or manually restart?**

Before changing anything, always run `get_restart_required`. If it indicates a change is needed, you then use the appropriate setter tool (like `set_config`) followed by `system_restart`.

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 `"syncthing": { "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

Syncthing 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 Syncthing. 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	Syncthing MCP
Server ID	`019e38f6-e0db-73d3-bb98-768e384c5f2f`
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/syncthing.