MCP SERVER | NO CODE | CLOUD HOSTED

Matrix/Element MCP

Manage secure chats and account state.

Matrix/Element MCP connects your AI agent directly to decentralized Matrix communications. You manage rooms, send secure messages, synchronize account status, and handle user discovery—all through natural conversation. It gives you full control over complex chat networks without needing a dedicated client.

F Quality Score 3.6/100

matrix | element | chat | decentralized | e2ee

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

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.

Matrix/Element MCP

19 tools available
Cloud-hosted on Vinkius

This connector lets you treat your private Matrix chats like any other data source available to your AI agent. Instead of logging into an app just to check on group discussions or send a quick update, you tell your agent what you need done. It handles everything from joining new chat rooms and sending encrypted messages to checking the latest conversation state. You can use it to search for specific users across large networks, manage your profile details, or even upload keys for end-to-end security. Integrating this MCP via Vinkius lets your AI client handle complex communication tasks—like setting up an alert room or coordinating incident response—right within your existing workflow.

Core Capabilities

01 — Manage Chat Rooms

Create, join, knock on, and leave chat rooms using simple commands.

02 — Send Messages and Events

Dispatch messages or custom events to any room with transaction tracking.

03 — Synchronize Account State

Fetch the latest status updates from your homeserver, keeping all conversations current.

04 — Discover Users

Search the global Matrix directory to find and connect with other users by name or handle.

05 — Handle Security Credentials

Query, upload, and manage end-to-end encryption keys for secure communication.

One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to the Matrix/Element MCP and provide your specific Matrix Homeserver URL and Access Token.

02 Your AI client reads the available tools, allowing you to define a conversational task (e.g., 'Send an alert message to the ops room').

03 The agent executes the necessary tool calls—like sending messages or syncing state—and returns the results directly to your conversation.

The bottom line is that your AI client acts as a direct interface, running complex Matrix actions without you having to switch applications.

Built For

Anyone who relies on secure or decentralized group communication needs this. Think DevOps engineers managing incident rooms 24/7, community managers coordinating across large networks, and privacy-focused developers needing chat data integrated into code.

DevOps Engineer

Automatically manage alert channels or response rooms, ensuring immediate message delivery and state updates during an incident.

Community Manager

Coordinate user interactions across different private chat networks by creating rooms, joining groups, and managing member states.

Software Developer

Build applications that require reading conversation history or sending messages based on real-time user actions within a secure communication channel.

What Changes When You Connect

01 Automate communication: Instead of manually opening your chat app to send an update, you simply ask your agent to use the `send_message` tool. It handles routing and transaction tracking automatically.

02 Maintain security context: Use tools like `query_keys` or `upload_keys` to manage encryption keys directly through your workflow, ensuring communications stay end-to-end encrypted without manual steps.

03 Stay current on activity: Running the `sync_client` tool pulls all new messages and state changes from the homeserver. You always have the latest version of a room's history available to your agent.

04 Build networks: Need to connect with someone? Use `search_user_directory` to find user handles, or use `create_room` and `join_room` to build specific working groups on demand.

05 Handle account lifecycle: You can manage the full account life cycle—from calling `register_account` to using `deactivate_account`—all through natural conversation with your agent.

Real-World Applications

Incident Response Coordination

The ops lead needs to get an immediate status update on a server issue. They ask their agent to run `get_room_state` on the #alerts room, and the agent pulls the latest 10 messages, identifying which engineer last responded using that data.

Onboarding New Team Members

A community manager needs to bring a new user into a private project group. They instruct their agent to first `search_user_directory` for the user's handle, then use `create_room`, and finally run `join_room` so everyone is connected.

Archiving Project Discussions

A developer needs all the finalized conversation details from a finished project. They ask their agent to retrieve the room state using `get_room_state`, which aggregates messages, allowing them to archive the full context.

Cross-Platform Messaging

You need to notify multiple stakeholders that a deployment is finished. You tell your agent to use `send_message` and specify the target rooms (e.g., #devops, #product), ensuring all channels receive the alert.

Patterns to Avoid

Treating it like a chat app

✗ AVOID

Just dumping messages into the agent without context or specifying actions. The agent can't figure out if you want to read history, join, or message.

✓ INSTEAD

Always start by defining the action. For example: 'First, use `search_user_directory` for John Smith. Then, use `join_room` to get into his project chat.' This gives the agent a clear path.

Ignoring permissions

✗ AVOID

Attempting to create or join rooms in areas where your account hasn't been properly logged in or authorized.

✓ INSTEAD

Before acting, run `login_account` if needed. If you are setting up a new workspace, use `register_account` first.

Assuming real-time data

✗ AVOID

Asking for the latest messages without telling the agent to refresh its view of the room's state.

✓ INSTEAD

Always run `sync_client` at the start of a session to ensure your agent is working with the absolute current conversation status.

The Right Fit

Use this MCP if your core workflow relies on highly secure, decentralized communication and you need to automate actions across multiple rooms or users. Specifically, if your process involves creating new group contexts (`create_room`), managing user identity (`search_user_directory` , `change_password`), or ensuring data integrity through encryption key handling (`upload_keys`). Don't use this if all you need is a simple one-off message send; the basic messaging capabilities might suffice. However, if your task requires

knowing the history of conversations, managing room membership (using `knock_room` and `leave_room`), or coordinating actions across multiple chat states simultaneously, this MCP is essential because it gives your agent the full spectrum of tools required for true communication automation.

The problem isn't just checking messages; it's managing context.

Right now, if you want to coordinate an incident response or manage a team project, you have to jump between the chat application, check who is available in the directory, and then manually create new rooms. You spend time copying user IDs into different platforms just to ensure everyone gets the alert.

With this MCP, your agent handles that entire sequence conversationally. Instead of multiple clicks and context switches, you simply tell your AI client: 'Set up a private room for Project Alpha with Bob and Sarah.' The agent manages the `search_user_directory`, calls `create_room`, and invites everyone—all in one go.

Matrix/Element MCP lets you manage secure chat state.

You don't have to remember which rooms need updating or who the current participants are. The agent can use `get_room_state` to pull history and `sync_client` to guarantee it has the most recent messages, giving you a complete picture of what happened.

The difference is that communication moves from being an app-based chore into a core data operation within your workflow. You don't just read chats; you automate actions *on* them.

Matrix/Element: 19 Tools for Communication Ops

These tools let you control nearly every aspect of your Matrix communication workflow, from sending a simple message to managing complex encryption keys.

#	TOOL	DESCRIPTION
01	`change_password`	Updates the account password for Matrix login.
02	`claim_keys`	Retrieves and claims end-to-end encrypted keys from the homeserver.
03	`create_room`	Initiates a brand new chat room within Matrix.
04	`deactivate_account`	Permanently deactivates the current user's Matrix account.
05	`download_media`	Downloads media files shared within a room from the homeserver.
06	`get_room_state`	Fetches specific state events and history for an existing chat room.
07	`join_room`	Adds the user to a specified Matrix chat room using its ID or alias.
08	`knock_room`	Requests temporary access to a private Matrix room.
09	`leave_room`	Removes the user from a specific chat room.
10	`login_account`	Authenticates and logs into the Matrix account.
11	`logout_account`	Signs out of the current Matrix session.
12	`query_keys`	Checks for existing end-to-end encryption keys on the homeserver.
13	`register_account`	Sets up a brand new Matrix user account.
14	`search_user_directory`	Searches the global directory for specific users or handles.
15	`send_message`	Transmits a message or custom event into a specified Matrix room.
16	`set_room_state`	Manually updates the state events for an existing chat room.
17	`sync_client`	Synchronizes the client's local data with the homeserver to get the latest status.
18	`upload_keys`	Uploads necessary end-to-end encryption keys to the homeserver for security.

#	TOOL	DESCRIPTION
19	upload_media	Sends media files (images, videos) to the homeserver.

See It in Action

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

U Sync my Matrix client to see if I have any new notifications.

I've triggered a sync. You have 3 new messages in '!engineering:matrix.org' and a new invitation to join the 'Project X' room.

U Send a message to room !abc:matrix.org saying 'The deployment is complete'.

Message sent successfully to room !abc:matrix.org with transaction ID 'txn_12345'.

U Search the user directory for 'bob'.

I found 2 users matching 'bob': Bob Smith (@bob_s:matrix.org) and Bob Builder (@builder:matrix.org).

Frequently Asked Questions

01 **How do I use the Matrix/Element MCP to send messages?**

Use `send_message` and specify the target room ID or alias. You just tell your agent who needs the message, and it handles dispatching the content.

02 **Can I find users in a large network using Matrix/Element MCP?**

Yes, use `search_user_directory` to look up user handles. This tool queries the global directory so you can get started on communicating with new contacts.

03 **Is the communication secure when I use Matrix/Element MCP?**

Absolutely. The MCP includes tools like `query_keys` and `upload_keys` to manage end-to-end encryption, ensuring your agent handles communications securely.

04 **What if I need to start a new chat room with the MCP?**

You'll use the `create_room` tool. This initializes a brand new matrix space that you can then populate and manage using other tools like `join_room`.

05 **Does Matrix/Element MCP only work for existing rooms?**

No, it handles the entire lifecycle. You can use `create_room` to start fresh or `get_room_state` on an existing room to pull its history.

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 `"matrixelement": { "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

Matrix/Element 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 Matrix/Element. 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	Matrix/Element MCP
Server ID	`019e38bf-17df-7302-bd76-d79df87edddd`
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/matrixelement.