MCP SERVER | NO CODE | CLOUD HOSTED

OpenStreetMap MCP

Query, Edit, and Audit Geospatial Data by Chat

OpenStreetMap MCP lets your AI client interact with the world's largest open geographic database. You can query raw map data, create new points of interest, manage complex boundaries, and track historical changes for any location. It gives you full control over geospatial elements—reading everything from nodes to relations directly through natural conversation.

F Quality Score 3.6/100

geospatial | map-data | cartography | gis | location-services | open-data

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

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.

OpenStreetMap MCP

33 tools available
Cloud-hosted on Vinkius

This MCP connects your AI agent directly to the OpenStreetMap API, letting it handle geographic data like a specialized GIS analyst would. Instead of manually exporting map layers or navigating complex web forms, your agent acts as an intermediary, allowing you to ask questions and make changes using plain language.

For example, you can request all points of interest within a specific bounding box for analysis, or audit the history of a particular road segment over time. Need to contribute to open-source mapping? You can automate creating new map notes, updating existing boundaries, and even managing entire groups of edits by opening and closing changesets—all without leaving your current chat window. Vinkius makes this connection simple; you just connect your preferred AI client once and gain access to the full range of these powerful geospatial tools.

It's about treating map data like a conversation: getting information, making adjustments, or tracking who changed what, immediately.

Core Capabilities

01 — Retrieve raw map data

Fetch all nodes, ways, and relations within specified geographic areas for deep analysis.

02 — Edit map elements

Add or modify points of interest (nodes), roads (ways), or complex boundaries (relations) directly via chat.

03 — Manage edits and changesets

Group related changes, open new work sessions, and officially close completed map updates.

04 — Track data history

Review the full version timeline for any specific element to see who changed it and when.

05 — Communicate about locations

Send private messages or add descriptive notes attached to map features.

One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to this MCP and provide your OpenStreetMap OAuth Token.

02 Connect your AI client (like Claude, Cursor, etc.) through the Vinkius platform.

03 Start by giving a natural language command, such as 'Get all nodes between X and Y coordinates,' or 'Create a new way for this path.'

The bottom line is you use your AI client to talk to the API, which then performs the complex map operations.

Built For

This MCP is built for people who deal with location data daily—GIS analysts needing quick data exports, open-source contributors automating mapping tasks, or developers building location-based apps. If your job involves understanding where things are and how they connect, this is for you.

GIS Analyst

Needs to quickly extract raw map data (nodes, ways, relations) across specific regions without manually downloading files or running complex scripts.

Open Source Contributor

Wants to automate the creation of new map elements or manage changesets via chat, keeping track of edits and providing metadata for the OSM community.

Developer (Geo-focused)

Needs to inspect map element metadata, check version history using get_element_history, and build location services by querying data directly.

What Changes When You Connect

01 Stop manually exporting raw data. The get_map_data tool lets you query nodes, ways, or relations within any bounding box using a simple chat command.

02 Don't just add points; manage the full lifecycle of map edits. Use create_changeset and close_changeset to group your work and submit it cleanly.

03 Need accountability? get_element_history tracks every single change made to an element, showing who did it and exactly when.

04 Go beyond simple querying by sending messages or creating notes. You can use send_message to coordinate with collaborators or create_note to annotate a specific location's context.

05 It handles complex data formats natively. Uploading GPS traces via upload_gpx means you never have to worry about manual file transfer again.

Real-World Applications

Auditing neighborhood infrastructure changes

A city planner needs to verify if all new bike lanes were tagged correctly. They use get_element and then get_element_history for a specific 'way' ID, instantly confirming that the required 'bicycle_lane' tag was added by the correct user at the right time.

Building a custom location dashboard

A developer needs to feed fresh data into an application. They use get_gpx_data and then list_user_gpx to pull coordinates from multiple sources, ensuring their service always runs on the latest map information.

Contributing structured open-source edits

An amateur cartographer finds a missing public fountain. Instead of leaving just a comment, they use create_note to add context, then create_element to define the node, and finally update_changeset to tag it properly.

Coordinating field research teams

A survey team leader needs to check if the ground crew has finished marking a zone. They use list_notes to see all notes within the area, and then can send_message to individual workers for status updates.

Patterns to Avoid

Treating map data like a static database query

✗ AVOID

Trying to get every single detail about an element by repeatedly running 'get_element' calls, resulting in slow, inefficient chats and hitting rate limits.

✓ INSTEAD

Use the batch function. Instead of querying elements one by one, gather all necessary IDs first, then use get_elements to fetch them simultaneously. This is faster and more stable.

Assuming you need a dedicated data export tool

✗ AVOID

Running multiple separate commands just to pull coordinates for a path, leading to messy outputs and forcing manual clean-up.

✓ INSTEAD

Use the specialized upload_gpx or get_gpx_data tools. They handle the entire GPS trace format in one go, giving you structured data ready for consumption.

Writing a narrative instead of asking specific questions

✗ AVOID

Saying 'Tell me about the area around this intersection,' which results in vague lists and doesn't provide actionable data like road names or boundaries.

✓ INSTEAD

Be precise. Use get_map_data and specify your bounding box coordinates, then follow up with an action: 'Now, for that data, please use update_element to add a 'restaurant' tag to all nodes.'

The Right Fit

Use this MCP if your core problem revolves around the *structure* and *provenance* of geographic information. This is not just a simple map viewer; it's an API wrapper that allows you to act as a full GIS specialist via natural language. If you need to read coordinates, modify attributes (update_element), or track who made a change (get_element_history), this MCP works. Don't use it if your only goal is general map visualization; stick to dedicated mapping platforms for that. Also, don't rely on it for data outside the core OSM dataset—it won't help with proprietary corporate records. If you need to interact with completely different types of structured data (like HR databases), look at a specialized database MCP instead.

The Struggle With Manual Map Data Collection

Right now, getting comprehensive map data means jumping between multiple tabs. You might download one file for nodes, another for ways, and then spend time manually cross-referencing them to ensure everything aligns with your project's needs. It’s a process of constant downloading, renaming, and copy-pasting.

With this MCP, that whole workflow collapses into conversation. Your agent fetches exactly what you need—nodes, ways, or relations—within specific coordinates in one go. You get clean, structured data without touching an export button.

Get Map Data and Audit Element History

You don't have to manually check the version history or hunt through old records anymore. Instead of opening a separate web interface just to see who modified a road segment, you ask your agent to run get_element_history.

It pulls up the entire audit trail—who changed it and when—right in your chat window. You don't analyze data; you simply retrieve verified facts.

OpenStreetMap MCP: 27 Map Data Operations Tools

These tools give your AI client granular control over every aspect of OpenStreetMap data, from retrieving raw coordinates to managing complex changesets and map notes.

#	TOOL	DESCRIPTION
01	close_changeset	Marks a collection of edits as complete within the OpenStreetMap system.
02	close_note	Closes a specific map note, removing it from public view.
03	comment_note	Adds an additional comment to an already existing map note.
04	create_changeset	Starts a new, isolated group of edits so you can track your work before submission.
05	create_element	Adds a brand-new geographic element, such as a point (node), road segment (way), or complex boundary (relation).
06	create_note	Writes and publishes a new descriptive map note for a location.
07	delete_element	Removes an existing node, way, or relation from the map data.
08	delete_message	Deletes a specific message record.
09	download_changeset	Downloads the full details of an approved changeset in a specialized format for offline use.
10	get_auth_user_details	Retrieves basic information about the user who connected their account.
11	get_changeset	Reads and displays all details for a specific changeset ID.
12	get_element_history	Retrieves the full audit trail, showing every recorded change to an element by date and user.
13	get_element	Reads all current data for a single geographic element using its unique ID.
14	get_elements	Retrieves multiple nodes, ways, or relations at once based on provided IDs.
15	get_gpx_data	Downloads the raw coordinate data for a specified GPS trace.
16	get_gpx_metadata	Reads descriptive information (like timestamps or device used) attached to a GPS trace.
17	get_inbox	Retrieves all unread and read messages intended for the authenticated user.

#	TOOL	DESCRIPTION
18	get_map_data	Fetches nodes, ways, or relations that fall within a specific set of geographical coordinates (bounding box).
19	get_message	Reads the full content and status of one private message from another user.
20	get_note	Displays the full text and details of a specific map note by its ID.
21	get_outbox	Retrieves all messages that the user has sent but are awaiting delivery confirmation.
22	get_permissions	Checks what specific actions the connected account is authorized to perform.
23	get_user_preferences	Reads settings and personal preferences configured for the authenticated user's account.
24	get_user	Retrieves detailed profile information for a specific, named user on the map platform.
25	list_notes	Lists all existing map notes that are contained within defined geographic coordinates.
26	list_user_gpx	Retrieves a list of GPS traces uploaded or associated with the authenticated user's account.
27	query_changesets	Searches and filters changesets based on criteria like time range, user, or geographic area.
28	search_notes	Finds map notes by searching through their content keywords.
29	send_message	Sends a private message to another specific user within the network.
30	update_changeset	Modifies descriptive tags or metadata for an open, ongoing changeset.
31	update_element	Changes existing data (like names or attributes) on a node, way, or relation that already exists.
32	update_message_status	Marks a message as read or unread to update the sender's status.
33	upload_gpx	Uploads an entirely new GPS trace file (GPX format) for permanent storage and analysis.

See It in Action

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

U Get map data for the bounding box -0.15, 51.50, -0.10, 51.52 (Central London).

I've retrieved the map data for that area. It contains 142 nodes and 24 ways. Would you like me to list the primary road names or points of interest found?

U Show me the history of way ID 4225123.

Inspecting history for Way 4225123... This element has 4 versions. It was originally created in 2015 and last updated by user 'MapMaster' in 2023 to add a 'bicycle_lane' tag.

U Find the 5 most recent closed changesets by user 'JohnDoe'.

Searching for changesets... I found 5 recent closed changesets for 'JohnDoe'. The most recent one (ID: 1452390) involved updates to building footprints in Berlin. Would you like to download the details of any specific changeset?

Frequently Asked Questions

01 **How do I query map data using OpenStreetMap MCP?**

You use get_map_data by specifying a bounding box (the coordinates) and what type of element you want to read. The system returns nodes, ways, or relations found within that exact area.

02 **Can I track changes on OpenStreetMap MCP?**

Yes, use get_element_history with an element ID. This tool retrieves the complete version timeline, showing every user who modified it and when they did it.

03 **What is 'changeset' in OpenStreetMap MCP?**

A changeset groups multiple edits together (like adding five points of interest). You use create_changeset to start the group, make your edits, and then close_changeset when you are done.

04 **Do I need a specific API key for OpenStreetMap MCP?**

Yes. When setting up this MCP on Vinkius, you must enter your unique OpenStreetMap OAuth Token to authorize the agent to write data.

05 **How do I add new points of interest using OpenStreetMap MCP?**

You use create_element and specify that you want to build a 'node' (a point). You provide the coordinates and any necessary tags, and the system handles the creation.

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 `"openstreetmap": { "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

OpenStreetMap 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 OpenStreetMap. 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	OpenStreetMap MCP
Server ID	`019e38cf-3c1b-7080-a16a-39ec0051266b`
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/openstreetmap.