MCP SERVER | NO CODE | CLOUD HOSTED

Tesla Fleet API MCP

Control hardware and track telemetry across your fleet.

Tesla Fleet API allows your agent to remotely control physical hardware on active Tesla vehicles. Check GPS locations, monitor battery health, manage door locks, or trigger lights and horns across an entire fleet—all through a secure, programmatic interface.

A+ Quality Score 100/100

telemetry | hardware-actuation | fleet-operations | gps-tracking | battery-monitoring | remote-control

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

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.

Tesla Fleet API MCP

8 tools available
Cloud-hosted on Vinkius

Managing a large vehicle fleet means more than just tracking data; it requires physical interaction with the cars themselves. This MCP lets you run commands that actuate real-world hardware on Tesla vehicles. You can pull live telemetry like GPS coordinates and battery state of charge (SoC), or you might need to manually trigger a relay, such as opening the charging port or locking all doors remotely. The system handles complex requirements, including knowing when a vehicle is asleep and needing to wake it up first before any command will work. By connecting this MCP via Vinkius, your AI client gains direct, programmatic control over core operational functions, making physical fleet management automated.

Core Capabilities

01 — Monitor Vehicle Status

Get live data on battery charge (SoC), mileage, GPS location, and internal temperatures for specific vehicles.

02 — Actuate Physical Locks

Send commands to remotely lock or unlock the doors of a vehicle.

03 — Wake Vehicles from Sleep

Safely wake up vehicles that are in an idle, sleeping state so subsequent commands will execute successfully.

04 — Control External Systems

Remotely trigger physical elements like the horn, headlights, or charging port relays.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/tesla-fleet-api** — connect your AI agent in three steps.

01 First, your agent must call `tesla_wake_up_vehicle` to ensure the vehicle is online and ready to accept commands.

02 Next, after waiting 10-15 seconds for the system to stabilize, you can execute specific actions like fetching data using `tesla_get_vehicle_data` or triggering a physical relay.

03 The agent receives real-time status updates or confirmation that the hardware command (like locking doors) was successfully sent.

The bottom line is: it provides reliable, multi-step control over vehicle electronics and relays, respecting the car's sleep cycle to prevent errors.

Built For

This MCP is built for Enterprise Operators and Logistics Engineers who deal with large fleets. If you spend your day coordinating physical movements—like remotely checking if a parked vehicle can be accessed, or needing to know the exact charge level before a worker arrives—you need this.

Fleet Manager

Coordinates remote access and monitoring for multiple vehicles across different sites, ensuring all assets are accounted for.

Automotive Logistics Engineer

Designs automated workflows that require physical interaction with vehicle components, such as sequencing charging cycles or confirming lock status before a handover.

What Changes When You Connect

01 The `tesla_get_vehicle_data` tool lets you get instant access to critical metrics like SoC, odometer reading, and exact GPS coordinates without needing physical proximity.

02 Never guess if a car is locked up. Use `tesla_control_doors` to remotely verify the status of vehicle locks or change them instantly from your agent.

03 `tesla_wake_up_vehicle` handles the critical first step: it forces the vehicle out of sleep mode, which allows all subsequent commands—like those for lights or climate control—to work reliably.

04 You can use `tesla_trigger_climate` to adjust internal temperatures before personnel arrive, making the car comfortable immediately upon arrival. This saves time and effort.

05 For simple alerts or locating a vehicle, running `tesla_honk_horn` or `tesla_flash_lights` acts as a reliable, remote attention-grabbing mechanism for the entire fleet.

Real-World Applications

Prepping an arrival location

A logistics worker is due in 30 minutes. Instead of driving up and having to manually open doors or adjust the air conditioning, your agent first calls `tesla_wake_up_vehicle`. Then, it uses `tesla_trigger_climate` and `tesla_control_doors` to ensure the car is warm and accessible when they arrive.

Emergency location tracking

A fleet manager needs to know if a vehicle left its designated parking zone. The agent first runs `tesla_wake_up_vehicle`, then calls `tesla_get_vehicle_data` repeatedly until the GPS coordinates confirm it's back within range.

Confirming security status

Before leaving a site, an operator needs to ensure all cars are locked down. The agent first runs `tesla_list_vehicles` to see what's available, then loops through and calls `tesla_control_doors` on each one to confirm they are secured.

Remote vehicle signaling

A parked car needs attention. The agent first runs `tesla_wake_up_vehicle`, then triggers a sequence of physical warnings by calling both `tesla_honk_horn` and `tesla_flash_lights` to draw immediate attention.

Patterns to Avoid

Trying to read data immediately

✗ AVOID

Calling `tesla_get_vehicle_data` or checking the SoC without first waking the car. The API will fail instantly, returning an HTTP 408 Timeout error because the vehicle is sleeping.

✓ INSTEAD

Always start with `tesla_wake_up_vehicle`. Wait at least 10 seconds after that call before attempting any read operation like fetching data with `tesla_get_vehicle_data`.

Assuming network connection

✗ AVOID

Attempting to use a physical relay command, like `tesla_control_doors`, when the local gateway is temporarily unstable. The command fails silently or partially.

✓ INSTEAD

Ensure your agent confirms successful token engagement and uses the designated wake-up sequence first. Treat every hardware action as critical.

Ignoring the fleet list

✗ AVOID

Running a single command for one specific car, but forgetting to check if other cars need attention or tracking.

✓ INSTEAD

Start by using `tesla_list_vehicles` to enumerate all assets in your fleet. Then, loop through that list and run necessary checks on each ID.

The Right Fit

Use this MCP if your operational requirement involves physically interacting with the car's hardware: locking doors, flashing lights, adjusting climate, or reading live telemetry (GPS/SoC). You must use it when a manual physical check would normally be required. Don't use this if you just need to read historical logs or process raw spreadsheet data; those tasks are better handled by time-series databases or document processing tools. If your goal is pure data

transformation without hardware action, skip this MCP. However, if the workflow hinges on knowing whether a car is powered up and ready to actuate relays, this is your primary control plane.

Tracking vehicle status used to be an impossible mess of manual checks.

Today, checking a fleet's status means logging into multiple vendor dashboards. You might check the GPS location in one tab, then switch to another portal just to see the battery SoC. If you need to know if doors are locked, that requires a third click, and every time you do this, you risk reading stale data or hitting an outright timeout.

With this MCP, your agent handles it all. You tell your client what status you need—say, 'Give me the location and battery of car XYZ'—and the system executes the complex multi-step sequence, including waking up the vehicle if necessary. You get a single, clean output that tells you exactly where the car is and how charged it is.

The tesla_wake_up_vehicle tool brings automated control to physical actions.

Previously, if your workflow required any action—like running `tesla_get_vehicle_data` or even just flashing the lights—and the car was parked and 'asleep,' the entire process failed with a cryptic error. You'd have to manually intervene and wait for the system to come back online.

Now, your agent handles that critical pre-step automatically. It executes `tesla_wake_up_vehicle`, waits the required time, and only then proceeds with the rest of your logic. That prevents timeouts and makes complex remote operations reliable.

Tesla Fleet API: 8 Tools Available

These tools give your agent direct access to physically interact with the vehicle's hardware and read its live performance metrics.

#	TOOL	DESCRIPTION
01	`tesla_control_charge_port`	Remotely activates the charging port relay after safely waking up the vehicle.
02	`tesla_control_doors`	Controls the physical locks, allowing you to secure or unlock the doors of a vehicle.
03	`tesla_flash_lights`	Triggers external headlights by flashing them after waking up the car.
04	`tesla_get_vehicle_data`	Pulls key telemetry data, including battery SoC, GPS coordinates, and internal temperatures, only after waking the vehicle first.
05	`tesla_honk_horn`	Remotely activates the physical horn mechanism to sound a loud alert after waking up the car.
06	`tesla_list_vehicles`	Retrieves a list of all vehicles in the fleet that are currently tracked by the system.
07	`tesla_trigger_climate`	Engages the internal climate control system to set temperature states before arrival, after triggering wake-up.
08	`tesla_wake_up_vehicle`	The critical first step: triggers the vehicle's ignition sequence to pull it out of an idle sleep state.

See It in Action

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

U Check active fleet execution tracking natively extracting explicitly the battery SoC of vehicle XYZ safely resolving sleep delays initially.

Parsed logically evaluating native wake sequences (`wake_up_vehicle`) avoiding timeouts implicitly. After latency bounds resolved cleanly, (`get_vehicle_data`) extracted telemetry cleanly outputting 65% battery level appropriately safely routing inherently.

U Actuate physical lock boundaries explicitly mapping the endpoints locking the doors inherently securely natively targeting 'car-aabbcc' dynamically.

Logical validation executed cleanly bounding constraints passing seamlessly checking actively wake states safely smoothly bounding locking explicit native parameters. (`control_doors`). Commands received explicitly locking.

U Sound the explicit vehicle horn targeting proxy array bounds locating physical target effectively resolving native bounds gracefully mapping targets.

Validating native API timeouts smoothly tracking cleanly isolating `honk_horn` correctly asserting constraints parsing actively safely formatting physical responses perfectly cleanly.

Frequently Asked Questions

01 **How do I use tesla_get_vehicle_data reliably?**

You must always run `tesla_wake_up_vehicle` first, then wait 10-15 seconds. Only after that sequence completes should you call `tesla_get_vehicle_data` to pull the actual telemetry data.

02 **Can I use tesla_control_doors before waking up the vehicle?**

No. The API will fail if the car is sleeping. You must always start with `tesla_wake_up_vehicle` to ensure the physical relays can be accessed and manipulated.

03 **Which tool shows me all cars in my fleet?**

`tesla_list_vehicles` provides a comprehensive list of every vehicle ID currently tracked by the system. You use this first if you need to run an action on multiple units.

04 **What happens if I forget to wait after waking up the car?**

If you don't wait, your agent will likely encounter a timeout error (HTTP 408) because the system needs time to transition from sleep mode back into full operational status.

05 **Can I check battery SoC and GPS at the same time with tesla_get_vehicle_data?**

Yes, `tesla_get_vehicle_data` pulls a master telemetry package that includes both the current State of Charge (SoC) percentage and the precise GPS coordinates in one call.

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 `"tesla-fleet-api": { "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

Tesla Fleet API 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 Tesla Fleet API. 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	Tesla Fleet API MCP
Server ID	`019d7611-a674-70f9-8ef5-27847c5953b0`
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/tesla-fleet-api.