MCP SERVER | NO CODE | CLOUD HOSTED

Plivo MCP

Manage voice, messages, and billing flows with AI.

Plivo MCP gives your AI client native access to a full global telecom network. It lets you send real SMS messages, manage VoIP calls across E.164 formats, and pull specific billing data directly from Plivo's console. Stop relying on visual dashboards; let your agent handle live communications and deep auditing.

A+ Quality Score 100/100

voice-api | sip-trunking | sms-gateway | telecom-infrastructure | e164-formatting | real-time-communication

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

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.

Plivo MCP

10 tools available
Cloud-hosted on Vinkius

This MCP connects your AI client straight into the global telecommunications infrastructure via Plivo. Instead of logging into a web portal or building complex API calls, you simply tell your agent what needs to happen—whether that's sending an urgent SMS notification or tracing an active call path. Your agent handles the complexity of bridging code into live telecom operations. You can ask it to initiate outbound voice calls, check account billing limits, and audit past communication logs for specific failure codes. If you manage critical communications flows, connecting Plivo through Vinkius's catalog means your AI client has immediate access to advanced voice and messaging capabilities without writing any boilerplate code or binding rules.

Core Capabilities

01 — Dispatch SMS messages

Your agent sends text messages to specific international phone numbers using a designated source number.

02 — Manage active voice calls

You can initiate, check details on, or terminate live outgoing and incoming phone calls.

03 — Audit message history

The system retrieves detailed logs of sent and received messages, including why specific transmissions failed.

04 — Review account capacity

You can check the current billing status or list all available phone numbers attached to your Plivo account.

One Click on Vinkius — From Prompt to Execution

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

01 Enable the Plivo integration module within Vinkius and provide your Auth ID and Auth Token.

02 Your AI client uses these credentials to establish a direct connection to the Plivo API endpoints.

03 You prompt your agent in natural language, directing it to perform specific telecom actions like checking balances or sending messages.

The bottom line is you tell your agent what communication task needs doing, and it executes the required telecom operation instantly through a conversational prompt.

Built For

This MCP targets highly technical teams—the DevOps engineer who needs to script diagnostics on demand, or the backend architect who must verify live SMS integrations. It's for anyone whose daily work depends on reliable, actionable communication data.

DevOps Engineer

Running diagnostic checks on production systems by having their agent ping account balances or sending immediate alerts to admin phones.

Backend Architect

Dynamically testing SMS and VoIP integrations by demanding the AI run test payloads that hit live mobile numbers.

Telecom Operations Specialist

Cleaning up phantom, stuck voice calls or reviewing detailed message logs to find out exactly why a carrier rejected a transmission.

What Changes When You Connect

01 You instantly manage complex communications without writing boilerplate. Need to send an SMS? Just prompt your agent; it handles the `send_sms` payload using E.164 formatting automatically.

02 Tackle live call management from chat. If a connection gets stuck, you can use `list_calls` and then sweep through any phantom connections with `terminate_call` to clean up resources.

03 Deep auditing is now conversational. Instead of digging through dashboards, ask your agent to query the `get_message_details` for specific failure reasons or check `list_messages` for compliance records.

04 Verify network readiness instantly. You can use `list_plivo_numbers` to confirm which phone numbers are active and available before running a new campaign.

05 Get immediate status checks. Use `get_account_info` to pull your current billing balance or check if you have enough funds remaining for the next operation.

Real-World Applications

Incident Response Alerting

A backend architect needs to alert admins about a service outage. Instead of writing code, they ask their agent to send an SMS using `send_sms` from the main number to all key personnel numbers.

Call Quality Assurance

The telecom ops specialist suspects phantom calls are draining resources. They run `list_calls`, identify stuck sessions, and tell their agent to execute `terminate_call` on the specific IDs found.

Pre-Deployment Testing

A developer needs to confirm an integration works before launch. They instruct the agent to make a test call using `make_voice_call` and then check the logs with `get_call_details` for successful connection metrics.

Billing Reconciliation

An accountant needs proof of services rendered last month. They ask their agent to query both `list_messages` and `get_message_details` across a date range to pull verifiable records.

Patterns to Avoid

Over-relying on web dashboards

✗ AVOID

Manually logging into the Plivo portal, navigating through multiple tabs, and copying phone numbers or billing data for a script.

✓ INSTEAD

Ask your agent to run `list_plivo_numbers` and then use the resulting list directly in a scripting command. For balance checks, just prompt for `get_account_info`.

Writing redundant API calls

✗ AVOID

A developer writes separate Python scripts to check call logs and account status, doubling their work and increasing failure points.

✓ INSTEAD

Tell your agent to handle both tasks in one prompt: 'Check the balance using `get_account_info` then list recent calls with `list_calls`.' It runs them sequentially.

Assuming all resources are available

✗ AVOID

Attempting to make a call or send an SMS without first checking if funds are available.

✓ INSTEAD

Always start by running `get_account_info` to confirm your current financial standing before attempting any outbound communications.

The Right Fit

Use this MCP if your core business process involves managing, auditing, or triggering real-time communication—specifically SMS messaging and voice calls. You need deep access to telecom data like call logs (`list_calls`) or granular message failures (`get_message_details`). Don't use it if you only need to read basic contact information; simple directory tools suffice for that. If your primary goal is just sending a single, non-critical alert and you don't care about the source number, a simpler messaging gateway might work. But when reliable E.164 formatting, active call management (`terminate_call`), and billing checks are essential, this MCP is required.

Dealing with Telecom Data Is Always Manual

Today, every time an incident happens or a campaign runs, your team has to log into the portal. You jump between the call history tab and the message logs, copy specific UUIDs, cross-reference them against billing reports, and then manually compile everything into a single ticket update. It's slow, error-prone, and takes half an hour just to gather evidence.

With this MCP, you simply ask your agent for the data. You tell it to audit the logs by running `list_messages` and check the call flow using `get_call_details`. Your agent gathers all the necessary information—the failures, the timings, the involved numbers—and presents it back in a clean, actionable report.

Plivo MCP Delivers Full Communication Control

You no longer have to write separate functions for every type of communication. The agent can initiate calls (`make_voice_call`), manage the session lifecycle, and then record all outcomes using `get_message_details` —all in one conversational workflow.

The biggest difference is that you stop thinking about API endpoints and start talking purely about business outcomes. You get direct control over your entire communication stack.

Plivo with 10 Tools

These tools give your agent specific actions across the telecom lifecycle, allowing you to manage everything from billing checks to live voice calls.

#	TOOL	DESCRIPTION
01	`terminate_call`	Ends an active phone call immediately.
02	`get_account_info`	Fetches the current billing and account details for your Plivo setup.
03	`get_call_details`	Retrieves detailed information about a specific, recorded voice call.
04	`get_message_details`	Pulls granular details regarding any single sent or received message.
05	`list_calls`	Lists a history of recent voice calls made through the service.
06	`list_sip_endpoints`	Retrieves a list of all configured SIP endpoints for routing purposes.
07	`list_messages`	Gets a comprehensive list of both sent and received text messages.
08	`list_plivo_numbers`	Lists all phone numbers registered and available in your Plivo account.
09	`make_voice_call`	Starts a new outbound voice call to a specified number.
10	`send_sms`	Sends an SMS text message from your account's designated source number.

See It in Action

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

U Check Plivo account status and let me know my remaining wallet balance.

Retrieval confirmed via `get_account`. Primary Plivo account is active securely. Current remaining balance mapped to `$412.55`. Would you like me to ping the `list_calls` node to verify active expenses burning constraints right now?

U Send an SMS message to `15551234567` from our main `15559876543` local number saying the servers are online.

Action logged and sent. Executed the mutation `send_sms` seamlessly passing source E.164 and target destinations simultaneously. Plivo acknowledged payload receipt queue mapping it successfully holding object UUID: `a7d-22r9`. I will monitor the background if required.

U Check Plivo network to list all presently active voice phone calls.

Query deployed against the active telecom trunk node (`list_calls`). Result: Empty array bounding currently parsed instances. The network endpoints are clean, reporting identically 0 concurrent routing states or phantom SIP loops tied to your active parameters.

Frequently Asked Questions

01 **How do I check my Plivo account balance using the Plivo MCP?**

Use `get_account_info`. This tool quickly fetches your current billing status and available funds, letting you know if you can afford to run a campaign before you start.

02 **Can I use the Plivo MCP to send SMS messages from multiple numbers?**

Yes. You first run `list_plivo_numbers` to see all registered sources, and then instruct your agent to execute the `send_sms` tool using a specific number as the source.

03 **What is the difference between listing calls and getting call details with Plivo MCP?**

Running `list_calls` gives you a summary of recent activity, showing who called whom. Use `get_call_details` when you need the deep dive—the full transcript or metadata for one specific call.

04 **Does Plivo MCP help me stop stuck calls?**

Absolutely. If a call drops and gets flagged as active, use `list_calls` to find the ID, then tell your agent to run `terminate_call` to clear it from the system.

05 **Can I audit message failures with Plivo MCP?**

Yes. You can list all messages via `list_messages`, and then use `get_message_details` on any specific ID to pull the exact reason a carrier rejected the frame.

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 `"plivo": { "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

Plivo 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 Plivo. 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	Plivo MCP
Server ID	`019d75f7-8e53-709d-b033-6fded2f46e82`
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/plivo.

© 2026 Vinkius · vinkius.com · support@vinkius.com