MCP SERVER | NO CODE | CLOUD HOSTED

Telegram Bot API MCP

Automate messaging, manage chats, and delete content.

Telegram Bot API MCP connects your AI agent directly into Telegram, letting you treat it like a certified bot application. This lets you read incoming user messages, manage group members, delete content, and send rich media alerts—all without needing to open the native app or worry about complex APIs.

A+ Quality Score 98.33/100

instant-messaging | bot-integration | push-notifications | automated-replies | group-management | media-distribution

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/telegram-bot-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.

Telegram Bot API MCP

14 tools available
Cloud-hosted on Vinkius

This MCP turns any standard conversational window into a powerful messaging director. By authorizing your agent as a Telegram Bot, you bypass general client restrictions entirely. You gain operational control over chats: read incoming messages, check group member lists, and send structured alerts instantly. Need to update an old message or clean up spam? Your agent can edit existing text or delete unwanted posts outright. For media distribution, simply give it a URL, and your AI client handles sending photos and documents across the chat. This level of control over real-time messaging is exactly what Vinkius provides, letting you manage complex workflows that span outside traditional databases.

Core Capabilities

01 — Read incoming messages and updates

The agent pulls recent communications from a chat, allowing it to analyze user queries or system alerts.

02 — Send formatted text and media alerts

You can send plain text messages, photos via URL, and documents using public links directly into any specified chat.

03 — Moderate group content

The agent can delete inappropriate user posts or edit the text of messages that were previously sent by the bot itself.

04 — Audit and manage groups

You can list who has administrative rights in a channel, check the total member count, or get details on any chat type.

One Click on Vinkius — From Prompt to Execution

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

01 First, provision the generic Telegram connectivity structure within your Vinkius limits module.

02 Next, go to @BotFather in Telegram to request a new bot token and assign that unique key as your MCP authorization credential.

03 Finally, instruct your AI client: 'Check for new messages mentioning X, then reply to their chat ID with Y.' Your agent handles the rest.

The bottom line is you get full operational control over Telegram messaging, letting your AI agent act like a built-in bot.

Built For

This MCP targets Ops Engineers and Community Managers who are sick of manually checking chat logs or waiting for developers to build custom integrations. If you spend time copy/pasting information from Telegram into Jira tickets, this is for you.

Community Manager

Checks group member lists to see who has admin rights and uses the agent to send bulk announcements or delete spam.

Support Engineer

Retrieves message updates when a user reports an issue, allowing the AI to immediately diagnose the conversation context without manual logging.

DevOps Engineer

Sets up automated push notifications that send build failure photos or deployment alerts straight into dedicated on-call Telegram groups.

What Changes When You Connect

01 You can stop manually checking chat logs. Using `list_bot_updates` lets your agent pull fresh messages instantly, so you never miss a key user query or status update.

02 Media distribution is simple: instead of uploading files, just provide the URL and use tools like `send_photo_by_url` or `send_document_by_url`. The AI handles delivering it flawlessly.

03 Need to correct an alert? Use `edit_message_text` to rewrite a previously sent message with updated data. It's better than deleting and resending, because the history stays clean.

04 Group management becomes auditable. You can use `list_chat_administrators` to instantly verify who has control over sensitive channels, or `get_chat_member_count` for quick audience sizing.

05 The system is resilient enough that you can handle complex interactions by using the agent to read messages (`list_bot_updates`) and then reply with a structured confirmation via `send_text_message`.

Real-World Applications

Automating support triage after hours

A Tier 1 Support Engineer wants to know if the API is down. Instead of manually checking logs, they ask their agent to run `list_bot_updates`. The agent finds the recent messages and sends a structured reply via `send_text_message` confirming the status.

Cleaning up group spam

A Community Manager spots inappropriate replies in a large public channel. They ask their AI client to delete the offending posts using `delete_chat_message`, keeping the main conversation flow clean and professional.

Sending urgent deployment notices

A DevOps team needs to notify an on-call group of a build failure. They provide the link, and the agent executes `send_photo_by_url` immediately, pushing the critical image straight to Telegram.

Auditing channel access

A Product Manager needs assurance that only certain leads can administer a private group. They use the tool to call `list_chat_administrators`, getting an immediate list of authorized users for review.

Patterns to Avoid

Trying to read data without context

✗ AVOID

The user tries to only look at the last five messages using a basic API call, missing critical metadata like who sent it or if the group is private.

✓ INSTEAD

To get full conversational history and ensure you capture all necessary details, use `list_bot_updates`. This tool pulls comprehensive updates from users, providing structured context for analysis.

Over-relying on simple text posts

✗ AVOID

Sending a critical alert about a system failure using only plain text; the necessary image or document link is lost.

✓ INSTEAD

For visual alerts, always use `send_photo_by_url` with an absolute HTTPS URL. This guarantees that the high-fidelity media arrives directly in the chat alongside your message.

Ignoring existing content

✗ AVOID

Needing to correct a mistake on a public announcement, but only having the option to delete it and re-send the whole thing.

✓ INSTEAD

Don't delete and resend. Use `edit_message_text` instead. This allows you to update the content of a specific message while maintaining its original place in the chat history.

The Right Fit

Use this MCP if your workflow requires constant, bidirectional communication with Telegram—specifically if you need your agent to read incoming messages (`list_bot_updates`), moderate group chats (using `delete_chat_message` or `edit_message_text`), or send alerts/documents based on external URLs. Don't use it if your primary need is simply reading raw message history from a non-bot source, as you'll want a different read-only connector type. If you only ever plan to write simple text and never manage group roles (like checking permissions via `list_chat_administrators`), this MCP gives you much more control than you'll need, so stick with it. But if your entire process is contained within a single database or internal app, don't use this; use an internal CRM connector instead.

Dealing with Telegram messages the manual way sucks.

Today, when you need to run an alert, you open the chat, scroll up manually to find the relevant thread, copy the ID, switch tabs, paste it into your automation tool, and then type out the response. If the message is long or critical, you spend five minutes just trying to organize what was said.

With this MCP, that whole process vanishes. You simply tell your agent: 'Check for new messages mentioning X.' It pulls all the relevant data—the context, the user ID, and the specific query—and presents it cleanly, ready for action.

Manage Chat Media & Messages with Telegram Bot API MCP

Manually managing content means constantly jumping between reading messages, then having to open a separate interface just to delete spam or pin an important announcement. You're always fighting the UI overhead.

Now, you tell your agent exactly what to do: 'Delete that message,' or 'Send this photo.' The bot executes the action directly within the chat environment. It’s operational control in one prompt.

Telegram Bot API: 14 Tools for Message Control

These tools give your agent granular control over every aspect of a Telegram chat, allowing it to read updates, manage content, and distribute media from within the platform.

#	TOOL	DESCRIPTION
01	`delete_chat_message`	Permanently removes a message from a chat, whether it was sent by the bot or another user.
02	`forward_message`	Sends an existing message from one Telegram chat to a completely different chat while keeping the original sender's information intact.
03	`get_chat_member`	Pulls specific profile details about a single member within a chat.
04	`pin_chat_message`	Marks an important message so it stays visible at the top of the conversation.
05	`unpin_chat_message`	Removes a pin marker from a chat message, returning it to normal scrolling visibility.
06	`get_chat_details`	Retrieves detailed information about any given chat, whether it's a user, group, or channel.
07	`get_bot_info`	Gets basic operational details about the bot itself, like its unique ID and username.
08	`edit_message_text`	Rewrites the text of a message that was already sent by the bot previously.
09	`list_chat_administrators`	Generates a list of all users who currently hold administrator permissions in a specific group or channel.
10	`get_chat_member_count`	Reports the precise total number of people belonging to a given chat.
11	`list_bot_updates`	Pulls new messages and activity updates from users in real-time, acting like continuous long polling for fresh data.
12	`send_document_by_url`	Transmits a file or document to the chat using only a public download link.
13	`send_text_message`	Sends basic text content directly to any specified user or group.
14	`send_photo_by_url`	Delivers an image file to the chat using a public HTTPS link.

See It in Action

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

U Scan the bot updates, check for new interactions, and send a structured text message back to the latest user chat ID resolving their query.

Interrogating 'list_bot_updates' securely natively...
Found 2 recent updates. Extracting newest payload context. Chat ID: 8472910.
User asks: 'Server Status'.
Dispatching resolution dynamically utilizing 'send_text_message'. Payload correctly executed and transmitted to the Telegram Chat.

U Send a markdown formatted message to the #engineering channel.

Formatting payload with `send_message` ...
Message "Deployment to production has started. **Status: Green**" sent successfully to `@engineering_alerts` (ID: -100123456).

U Fetch the last 5 messages from our customer support group.

Calling Telegram API (`get_updates`)...
Retrieved 5 recent messages from Chat ID -100987654:
1. User44: 'Is the API down?'
2. AgentSmith: 'No, looking into it...'
3. User44: 'My token is failing.'
4. AgentSmith: 'Can you DM me your client ID?'
5. User12: 'I have the same issue.'

Frequently Asked Questions

01 **How does list_bot_updates work with Telegram Bot API MCP?**

list_bot_updates pulls all new messages and activity updates from users, acting like continuous polling. Your agent receives a structured payload of the latest interactions instantly.

02 **Can I send files using the Telegram Bot API MCP?**

Yes, you can use `send_document_by_url` or `send_photo_by_url`. These tools require a public URL and handle streaming the media directly into any chat.

03 **What if I need to change an old message?**

You use the `edit_message_text` tool. This allows your agent to rewrite the text of a specific message previously sent by the bot, keeping it visible in the chat history.

04 **How do I check group permissions using Telegram Bot API MCP?**

You run `list_chat_administrators` to generate an immediate list of everyone who has admin rights. This is useful for compliance checks and governance.

05 **Is the Telegram Bot API MCP better than just posting a web hook?**

Yes, because this MCP gives your agent operational control over the messages—it can read updates (`list_bot_updates`) and *reply* or *moderate* content using tools like `send_text_message`.

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 `"telegram-bot-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

Telegram Bot 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 Telegram Bot 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	Telegram Bot API MCP
Server ID	`019d7611-1d71-7191-a58e-097db5ad2582`
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/telegram-bot-api.