MCP SERVER | NO CODE | CLOUD HOSTED

Home Assistant MCP

Control your whole smart home with conversation.

Home Assistant MCP connects your smart home ecosystem—lights, thermostats, media players, and sensors—to any AI agent. Use natural language prompts to monitor device status or run complex automations across local or cloud-based Home Assistant instances.

A+ Quality Score 100/100

smart-home | home-automation | device-control | local-control | automation-triggers

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

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.

Home Assistant MCP

15 tools available
Cloud-hosted on Vinkius

Connecting your entire smart house to an AI client is simple. This MCP lets you use conversation to manage everything from turning lights on and off to adjusting the climate temperature.

Your agent talks directly to your home's central hub via the REST API. You can ask it what the current humidity level is, check if a specific door sensor was tripped, or even tell it to run an entire automation sequence—all without opening a single app dashboard. It's like having a dedicated smart home professional that lives inside your AI workflow.

If you're building complex systems and need reliable connections across multiple platforms, Vinkius provides the central point where your agent can talk to thousands of services, making this connection effortless. You get real-time data on every entity, history logs for troubleshooting, and control over specific devices—all from one prompt.

Core Capabilities

01 — Check device status

The agent lists all connected smart home components and tells you their current operating state.

02 — Adjust physical devices

You can tell the system to change settings, like setting a thermostat temperature or dimming lights.

03 — Trigger complex actions

The agent runs predefined automations and sends custom events through your home network.

04 — Manage calendar data

You can list or query scheduled events stored in Home Assistant calendars.

One Click on Vinkius — From Prompt to Execution

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

01 Connect your Home Assistant instance URL and a long-lived access token to this MCP.

02 Tell your AI agent what you want done, for example: 'Set the living room lights to 50% brightness.'

03 Your agent sends the specific service command through the API, updating your physical devices in real time.

The bottom line is that it translates natural language commands into precise smart home actions.

Built For

This MCP is for anyone whose job involves managing a physically automated environment. Property managers who oversee multiple buildings, or automation engineers building custom scripts will find this invaluable.

Property Manager

Monitoring sensor readings across different units to quickly identify maintenance issues or verify security states.

Automation Engineer

Testing and calling specific Home Assistant services (like `script.turn_on`) in a development workflow without needing the UI.

Smart Home Enthusiast

Building complex, multi-step routines that involve checking sensor data before activating lights or media players.

What Changes When You Connect

01 Instant device status checks. Instead of navigating through multiple apps, you simply ask the agent to check a specific sensor's state using `get_entity_state`, and it tells you the number right away.

02 Comprehensive control over all devices. You can adjust lights, set temperatures via `call_ha_service`, or even manage media playback—all through natural conversation without writing code.

03 Deep operational visibility. Use `list_entity_states` to see every single device connected to your network at a glance, making it easy to find the exact ID you need for advanced scripting.

04 Troubleshooting and auditing. When things go wrong, check historical data using `get_entity_history` or review system activity with `get_logbook_entries` to pinpoint exactly when an issue started.

05 Advanced automation building. You can simulate triggers by running a custom event via `fire_ha_event`, allowing you to test your entire home's logic flow before deploying changes.

Real-World Applications

Checking security status after a power outage

A property manager asks their agent, 'What is the current state of all door and window sensors?' The agent uses `list_entity_states` to report back whether every access point was properly secured or if any were left open.

Setting up a movie night routine

A user prompts, 'Start Movie Night.' The agent executes multiple commands using `call_ha_service`: it dims the lights, lowers the motorized blinds (cover), and turns on the media player.

Investigating a temperature fluctuation

An automation engineer asks the agent to check climate data. The agent uses `get_entity_history` on the thermostat entity to show a graph of how the temperature drifted over the last six hours, identifying when the HVAC system failed.

Running complex scripts for testing

A developer wants to test a new automation sequence. Instead of triggering it manually, they use `fire_ha_event` with a custom event type, ensuring their workflow logic works correctly without physical intervention.

Patterns to Avoid

Copying API calls directly

✗ AVOID

The user tries to manually construct the JSON payload for every single service call (e.g., writing out `domain: light, service: turn_on...`) and pastes it into their agent prompt.

✓ INSTEAD

Just tell your agent what you want in plain English: 'Turn on the kitchen lights.' The agent handles generating the correct parameters and calling the appropriate function (`call_ha_service`).

Focusing only on current state

✗ AVOID

The user asks, 'Is the living room light on?' but doesn't know if it was ever off before or what its history is.

✓ INSTEAD

To get a complete picture, first use `get_entity_state` for the immediate status. Then, run `get_entity_history` to analyze state changes over time.

Overlooking available actions

✗ AVOID

The user only knows how to toggle a switch and doesn't realize they can set specific brightness or color temperatures.

✓ INSTEAD

Always run `list_available_services` first. This shows you every possible action, like setting color temperature for lights or changing HVAC modes.

The Right Fit

Use this MCP if your primary need is translating natural language into physical actions within a controlled environment, such as a smart home or industrial building. If your task involves reading data from diverse sources—like combining sensor readings with calendar events—this MCP provides the necessary state management tools (`get_entity_state`, `list_available_services`). Don't use this if you just need to manage cloud-based SaaS records (use a database connector) or if your devices communicate via protocols other than Home Assistant. If you are only building simple, standalone scripts that don't require state monitoring, writing direct API calls might be faster; however, for robust agentic workflows, the conversational approach here is unmatched.

The pain of manually checking every smart device dashboard.

Right now, if you need to know why the bedroom light is dim and if the thermostat was working correctly yesterday, you have to open three different apps. You check the lights app for brightness, then switch over to the climate control page to see mode details, and finally jump into a logbook just to verify when the last change happened. It's constant context switching and clicking.

With this MCP, your agent handles it all in one go. You ask, 'What was the activity status of the bedroom zone?' The agent uses its tools—like `get_entity_state` and `get_logbook_entries` —to gather the data points and present you with a single, cohesive answer.

The Home Assistant MCP gives you control over your entire physical environment.

You never have to manually write out specific service calls or worry about the correct JSON structure for every domain. You don't need to remember if a light uses `light.turn_on` or another service call; you just ask your agent, and it figures out the proper parameters using tools like `list_available_services`.

It changes everything. Your AI client moves from being a helpful chatbot into an actual operational controller for your physical world.

Home Assistant MCP: 15 Tools for Automation

These tools let you list device states, adjust temperatures, run automations, and track system logs across your connected smart home.

#	TOOL	DESCRIPTION
01	`get_api_status`	Run a quick test to confirm the Home Assistant API is online and ready for commands.
02	`get_calendar_events`	Retrieves scheduled events from any calendar linked in your system.
03	`list_ha_calendars`	Shows all the individual calendars configured within Home Assistant.
04	`check_ha_configuration`	Validates the overall configuration of your Home Assistant system for errors.
05	`list_ha_components`	Lists all major integrations and components that are currently running in Home Assistant.
06	`get_ha_config`	Retrieves detailed information about the overall system configuration settings.
07	`list_ha_events`	Shows every type of custom event that is actively tracked by your Home Assistant setup.
08	`get_entity_state`	Checks the current status of one specific device or sensor using its unique ID.
09	`get_entity_history`	Gathers historical data points for a single entity, helping you analyze usage patterns over time.
10	`get_logbook_entries`	Retrieves a chronological list of significant activity entries and system logs.
11	`render_ha_template`	Executes complex template code using Jinja2 to calculate or format advanced state data.
12	`list_available_services`	Displays a comprehensive list of all actions (services) that can be performed across every device domain.
13	`list_entity_states`	Lists the current state, last change time, and attributes for every connected entity.
14	`call_ha_service`	The main function: it executes any specific service call, like turning on a light or setting a temperature.

#	TOOL	DESCRIPTION
15	`fire_ha_event`	Sends a custom, triggered event to initiate automations elsewhere in the system.

See It in Action

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

U List all my smart home entities and show me the lights.

I found 47 entities in your Home Assistant instance. Here are the lights: light.living_room (on, brightness: 128), light.kitchen (off), light.bedroom (on, brightness: 200, color_temp: 370), light.office (off). Would you like to control any of these?

U Turn on the living room light and set it to 50% brightness.

Calling light.turn_on on light.living_room with brightness 128 (50%)... Done! The living room light is now on at 50% brightness.

U What is the current temperature in the bedroom and what mode is the thermostat?

Checking climate.bedroom... Current temperature: 21.5°C. The thermostat is in 'heat' mode with a target temperature of 22°C. Humidity is 45%.

Frequently Asked Questions

01 **How do I check if this Home Assistant MCP works with my local network?**

Yes, it supports both Nabu Casa cloud and local instances. You simply provide the correct URL format when configuring your agent's connection details.

02 **Can I use the get_entity_state tool to see all my devices?**

You can list everything using `list_entity_states`. This function returns a comprehensive list of every entity ID, its current status, and key attributes.

03 **What is the best way to control lights with Home Assistant MCP?**

Use the `call_ha_service` tool. You specify the domain (like 'light') and the service name (like 'turn_on'), along with any necessary parameters like brightness or color.

04 **Does this MCP help me troubleshoot my automations?**

Absolutely. By using `get_logbook_entries` and checking entity history via `get_entity_history`, you can trace back exactly when a state changed and why, solving tough automation puzzles.

05 **Do I need to know coding if I use the Home Assistant MCP?**

No. You don't write code; your agent does. You just talk naturally to it, letting it handle the API calls using tools like `call_ha_service` in the background.

06 **What Home Assistant URL should I use?**

For local instances, use your Home Assistant's local network URL: `http://YOUR_IP:8123` (or `http://homeassistant.local:8123`). For cloud access via Nabu Casa, use your remote URL: `https://YOUR_INSTANCE.ui.nabu.casa`. The API is accessible at the same base URL as your Home Assistant frontend.

07 **How do I get a Long-Lived Access Token?**

Log in to your Home Assistant web interface, go to your user profile (click your name in the sidebar, then **Profile**), scroll down to **Long-Lived Access Tokens**, click **Create Token**, give it a name (e.g., 'MCP Server'), and copy the generated token. This token never expires unless manually revoked.

08 **What devices and integrations are supported?**

All Home Assistant integrations are supported since the API works at the service/state level. This includes lights (Philips Hue, LIFX, etc.), thermostats (Nest, Ecobee, etc.), covers/blinds, switches, media players (Sonos, Chromecast, etc.), sensors, cameras, locks, vacuums, and 1000+ other integrations. Use `list_entity_states` to discover all available entities.

09 **Can I trigger Home Assistant automations from the API?**

Yes! You can trigger automations in multiple ways: 1) Call the `automation.trigger` service directly, 2) Fire a custom event using `fire_ha_event` that matches an event trigger in your automation, 3) Call the `script.turn_on` service to run scripts. You can also control devices directly which will trigger related automations.

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 `"home-assistant": { "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

Home Assistant 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 Home Assistant. 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	Home Assistant MCP
Server ID	`019d75b2-2025-7317-b1a6-1921f44380ac`
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/home-assistant.