MCP SERVER | NO CODE | CLOUD HOSTED

# Incident.io MCP

## Map who is on call and what the issue is.

Incident.io connects your AI client directly to incident response data. Use this MCP to automatically list open incidents, check on-call rotations, and map required roles and teams instantly. It lets you manage the entire lifecycle of an outage—from initial alert to resolution report—without leaving your agent's chat window.

A+ Quality Score 100/100

incident-response | sre | on-call-scheduling | workflow-automation | alerting | team-coordination

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

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.

# Incident.io MCP

10 tools available
Cloud-hosted on Vinkius

Managing a major outage means juggling multiple dashboards: one for active alerts, another for team contact info, and yet a third just for who is currently on call. This MCP lets your AI client pull all that information together in a single request. You can ask your agent to find every open incident, check which roles are defined, or verify the current on-call schedule—all through one chat command.

It's built specifically for SRE and IT Ops teams who need fast context during a crisis. If you connect this MCP via Vinkius, your AI client instantly gains access to Incident.io's full API suite. You can get incident details, list all available users and teams, or check the system's predefined incident types. This means automating complex workflows—like triaging an alert and knowing exactly who needs to jump on it—is routine, not complicated.

## Core Capabilities

**01 — List all active incidents**

Your agent can pull a list of every current or historical incident recorded in the system.

**02 — Determine on-call coverage**

It retrieves all defined on-call schedules, telling you exactly who is responsible for what time period.

**03 — Gather team and user context**

The MCP accesses full lists of users and teams so your agent knows who reported the issue or who should be notified.

**04 — Map incident parameters**

Your client can read predefined roles, severity levels, and catalog types to ensure proper classification when creating reports.

**05 — Check specific incident details**

The agent pulls the full history and context for a single incident ID when you ask it to look up that specific case.

## One Click on Vinkius — From Prompt to Execution

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

01 Tell your AI client, 'Check on-call schedules for us.'

02 The MCP executes the `list_schedules` tool and gathers related context like defined incident roles.

03 Your agent compiles this data into a clean, summarized report that you can read or pass to another system.

The bottom line is your AI client gets all the necessary IT Ops data it needs from Incident.io without needing dozens of manual clicks across different dashboards.

## Built For

This MCP is for the Site Reliability Engineer (SRE) who's tired of jumping between ticketing systems, Slack channels, and on-call calendars during a major outage. It's built for IT Operations Analysts and Incident Managers who need instant context to keep things moving fast.

**Site Reliability Engineer (SRE)**

Uses this MCP to automate incident triaging, pulling together the list of open incidents, checking which teams are responsible, and verifying if the correct severity level was assigned.

**Incident Manager**

Relies on it to maintain a real-time overview of all active events. They use it to confirm who is currently on call and what roles need to be engaged for immediate response.

**IT Operations Analyst**

Uses the MCP to audit incident records, checking system metadata like custom fields or predefined catalog types before escalating an issue.

## What Changes When You Connect

01 You don't have to manually cross-reference roles with schedules. By using `list_incident_roles` and `list_schedules`, your agent instantly tells you which team owns an incident based on its type or severity, saving critical minutes during a crisis.

02 Stop switching between user directories and ticketing pages. The MCP's ability to run `list_users` alongside `get_incident` means your AI client can immediately identify the reporter's contact info while simultaneously pulling up the full incident history.

03 Classification is key when time is short. You can check all predefined standards by calling `list_incident_types`, `list_severities`, and `list_catalog_types`. This ensures that every report you file uses consistent, accurate data points.

04 The MCP lets your agent pull a full picture of the environment. If you need to know what information is available for future reports, use `list_custom_fields` and `list_teams` to see exactly what metadata you can capture.

05 You get context on demand. Instead of having to search through endless dashboards, asking your agent to run `list_incidents` or `get_incident` gives you a filtered, summarized view of only the most relevant information right when you need it.

## Real-World Applications

**Triaging an unexpected alert**

An employee gets paged at 3 AM. They ask their agent to 'What do I do about this?' The agent runs `list_incidents`, finds the specific ID, uses `get_incident` for details, then checks `list_schedules` to name the primary on-call engineer immediately.

**Auditing incident data quality**

A compliance team needs to verify that all recent incidents were correctly categorized. They ask their agent to run `list_incident_types`, compare it against the system's defined roles (`list_incident_roles`), and ensure no critical fields were missed using `list_custom_fields`.

**Preparing a post-mortem report**

After an outage, the manager asks their agent to pull all relevant data. The agent runs `list_teams`, compiles a list of involved users (`list_users`), and gathers incident metadata using `get_incident` to build a comprehensive timeline for stakeholders.

**Verifying team access during an outage**

A user can't access the dashboard. They ask their agent to list all active teams (`list_teams`) and cross-reference that with the incident roles (`list_incident_roles`) to figure out which department should handle the permission issue.

## Patterns to Avoid

### Relying on dashboards alone

✗ AVOID

A human spends 15 minutes clicking through Incident.io's web UI, going from the 'Dashboard' to 'Schedules,' then searching for a specific 'User ID.' This is slow and prone to missing context.

✓ INSTEAD

Tell your agent directly: 'Who owns this incident?' The agent calls `get_incident` and automatically cross-references that with `list_schedules` and `list_teams`, giving you the answer in three seconds.

### Manually correlating roles

✗ AVOID

A team member sees an alert for a 'Database Issue' but has no idea which role owns it. They must search through multiple wikis and internal emails to find out.

✓ INSTEAD

Ask your agent, 'What is the primary role for database incidents?' The MCP calls `list_incident_roles` immediately and tells you the correct responsible team.

### Forgetting required context

✗ AVOID

Writing a manual report that misses key details because the person writing it isn't aware of all available metadata fields.

✓ INSTEAD

Ask your agent to check what data is available: 'What custom fields should I be tracking?' The MCP uses `list_custom_fields` and `list_catalog_types` to provide a complete list.

## The Right Fit

Use this MCP if your primary need is real-time, contextual data gathering during an active incident. If you constantly find yourself jumping between different Incident.io views—the user roster, the schedule page, and the incident timeline—this tool solves that coordination problem by giving your AI client a single API gateway to all those details.

Don't use this if you are building a highly complex data transformation pipeline that requires deep write access or conditional logic outside of simple read operations. If your goal is merely sending scheduled emails based on an incident status change, a dedicated messaging service MCP might be better suited. This MCP excels at reading and compiling information; it's not designed to manage the entire workflow state itself.

## Incident responses used to feel like a scavenger hunt.

When something breaks, finding out who owns the problem or who can fix it is hell. You open Incident.io's dashboard, then you have to click over to check the team roster. Then you jump again to see who’s on call this week. You end up bouncing between three different screens and copying names back and forth into your notes.

With this MCP connected through Vinkius, you simply ask your agent: 'Who's responsible for the main database service right now?' The agent runs multiple background checks—checking roles, schedules, and teams—and hands you one clean answer. It turns a 10-minute manual process into an immediate chat response.

## Incident.io MCP: Instant context on every alert.

You no longer need to manually check the incident details, then switch tabs to look up the relevant user ID, and finally open another document to see what severity level that user normally handles. The agent pulls all three pieces of data together in one go.

The result is pure speed. You get a single source of truth for everything happening during an outage. It's not just about getting information; it's about accelerating the moment you need context the most.

# Incident.io: 10 Incident Management Tools

These tools let your agent read all the core data from Incident.io, giving you complete context on incidents, users, teams, and scheduling.

| # | TOOL | DESCRIPTION |
|---|---|---|
| 01 | `get_incident` | Pulls full details and history for a single specified incident. |
| 02 | `list_catalog_types` | Retrieves all defined types of catalog information used by the system. |
| 03 | `list_custom_fields` | Lists every custom field available for incident documentation. |
| 04 | `list_incident_roles` | Shows all predefined roles that can be assigned during an incident response. |
| 05 | `list_incident_types` | Retrieves a list of standard incident classifications used across the platform. |
| 06 | `list_incidents` | Pulls a comprehensive list of all incidents recorded in the system, active or closed. |
| 07 | `list_schedules` | Displays all configured on-call rotations and schedules for teams. |
| 08 | `list_severities` | Lists the defined levels of severity that can be assigned to an incident. |
| 09 | `list_teams` | Retrieves a list of all internal teams within the organization. |
| 10 | `list_users` | Pulls a directory of every user account associated with Incident.io. |

# See It in Action

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

**U** List all ongoing incidents in Incident.io.

I'll fetch the list of current incidents for you.

**U** Show me the on-call schedules for this week.

I'll retrieve the active on-call schedules from Incident.io.

**U** Check the details for incident ID 'abc-123'.

I'll look up the full details and history for that specific incident.

# Frequently Asked Questions

01 **Can Incident.io MCP list all open incidents?**

Yes, the `list_incidents` tool lets your agent pull a comprehensive list of every incident recorded in the system, whether it's active or closed.

02 **How do I find out who is on call using Incident.io MCP?**

You use the `list_schedules` tool. This allows your agent to check all configured rotations and tell you exactly which team members are covering duty right now.

03 **Does the MCP help me find user details for a specific incident?**

Yes, after getting an incident ID via `get_incident`, your agent can cross-reference it with `list_users` to pull up contact or team information about involved parties.

04 **What kind of metadata can I retrieve using Incident.io MCP?**

You can read all the system's standards, including predefined incident types (`list_incident_types`), severity levels (`list_severities`), and custom data fields (`list_custom_fields`).

05 **Can Incident.io MCP list all available teams?**

Absolutely. The `list_teams` tool gives your agent a full roster of every team in the organization, which is useful for reporting and assignment.

# 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 `"incidentio": { "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

# Incident.io 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 →**

## INDEPENDENT PLATFORM DISCLAIMER

Vinkius is an independent platform and is not affiliated with, endorsed by, sponsored by, verified by, or otherwise authorized by Incident.io. 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 | Incident.io MCP |
| Server ID | `019d75b8-fd34-7366-bcf0-c1bc53558e7f` |
| 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/incidentio.