MCP SERVER | NO CODE | CLOUD HOSTED

Jira Service Management MCP

Manage support desks, tickets, and knowledge base articles instantly.

Jira Service Management (JSM) MCP connects your AI agent directly to an enterprise-grade IT service desk. Use this to instantly list all active services, retrieve full customer ticket histories, check support queue backlogs, and search knowledge base articles from one prompt. Automate complex IT operations without ever leaving your client.

A+ Quality Score 100/100

itsm | service-desk | incident-management | queues | knowledge-base | automation

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/jira-service-management-jsm

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.

Jira Service Management (JSM) MCP

10 tools available
Cloud-hosted on Vinkius

Need a better way to handle the flood of customer requests? This MCP gives your AI agent deep access to Jira Service Management's core functions. Instead of clicking through multiple pages just to get a full picture, you can ask your agent to pull together all the necessary data points. For instance, you can first use it to list available service desks, then check the current queues for backlogs, and finally retrieve the full details of any specific customer request using its ID or key. This capability is crucial when building complex IT support workflows. When you connect this MCP via Vinkius, your agent gains an entire suite of tools, letting you identify support recipients through `list_customers` or find relevant documentation by listing available knowledge bases, making service delivery automation genuinely practical.

Core Capabilities

01 — Monitor Service Desks and Queues

List all active service desks to understand the project structure, and check specific queues to gauge ticket backlogs.

02 — Investigate Specific Requests

Get a complete breakdown of any single customer inquiry, including participants, custom fields, and full history.

03 — Identify Support Assets

List all organizations or customers associated with the service desk to pinpoint who needs support.

04 — Search Documentation

Find available knowledge base articles relevant to common issues without manual searching.

05 — Track Support Intake

View all current customer requests and list the types of services offered through the portal.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/jira-service-management-jsm** — connect your AI agent in three steps.

01 First, your agent uses `list_service_desks` to find the specific project ID you need to work with.

02 Next, it can run `list_queues` or `list_requests` using that service desk ID to get a summary of open items and backlogs.

03 Finally, you prompt your agent to use `get_request` on a specific ticket key to retrieve the full, detailed conversation history.

The bottom line is that instead of opening Jira yourself, you simply tell your agent what data you need, and it handles the necessary steps within the system.

Built For

The Service Desk Analyst who spends half the day manually navigating ticket IDs. The Operations Manager who needs a real-time view of service backlogs without logging into multiple dashboards. Anyone whose job depends on having perfect context from complex support history.

Service Desk Analyst

Uses this MCP to quickly pull up the full details of an incoming ticket using `get_request`, saving time when following up with a customer.

IT Operations Manager

Runs reports on queue status using `list_queues` and tracks overall service coverage by listing available knowledge bases.

Technical Writer/Content Owner

Uses the MCP to list all possible request types or services offered, ensuring documentation matches current offerings.

What Changes When You Connect

01 Get a full picture of ticket status without clicking through menus. Use `list_requests` to pull a summary list of open items across the board, letting you spot urgent issues fast.

02 Stop guessing who owns the support request. By listing organizations with `list_organizations`, you can group data and understand which business entity needs attention.

03 When investigating a problem, don't waste time reading entire threads. Use `get_request` to pull all necessary details—participants, custom fields, everything—in one go.

04 Stay ahead of backlogs by checking queues with `list_queues`. You see exactly where tickets are piling up and if the triage process is slowing down.

05 Build better self-service flows. Use `list_knowledge_bases` to confirm that documentation exists before sending a customer an article link, ensuring accuracy.

Real-World Applications

Onboarding a New Support Agent

A new agent needs to understand the support landscape. They ask their agent to first run `list_service_desks` and then check `list_request_types`. This gives them an immediate overview of what services are offered and which support portals exist, allowing them to guide users accurately.

Auditing a High-Priority Customer Issue

A manager needs to review the history for a critical client. They start by running `list_service_desks` to confirm the correct project ID, then use that ID with `get_request` to pull every single detail and participant involved in the ticket.

Identifying Documentation Gaps

A team lead realizes customers are asking about a niche topic. They run `list_knowledge_bases` for that service desk, see nothing relevant, and know they need to create new documentation immediately.

Checking Backlog Health Before Shift Start

A shift supervisor needs an immediate status report. They ask their agent to run `list_queues` to see if the 'Unassigned' queue is overflowing, giving them a clear metric of where manpower is needed.

Patterns to Avoid

Searching by vague keywords

✗ AVOID

Asking your agent, 'Find me all tickets about network issues.' This is too broad and might return irrelevant results or fail entirely.

✓ INSTEAD

First, use `list_service_desks` to get the correct project ID. Then, run `list_requests` with that scope, which gives you a list of keys you can then filter.

Assuming IDs are known

✗ AVOID

Telling your agent to check Service Desk 'XYZ' when you only know the name. The request will fail because it needs an ID.

✓ INSTEAD

Always start by calling `list_service_desks` to get a list of valid project keys and IDs before trying to query queues or requests.

Trying to find customer details first

✗ AVOID

Just listing customers without context. You get names, but no idea what service desk they are tied to.

✓ INSTEAD

Use `list_service_desks` first, then use the resulting ID scope when calling `list_customers`. This ensures you're looking at the right set of users.

The Right Fit

Use this MCP if your core job involves managing service tickets, tracking backlogs, or synthesizing complex support histories. If you need to know *who* is asking for help, what services are offered, and what the current status is across multiple internal systems, this is mandatory. Don't use it if you just need a simple list of contacts—you can handle that with basic directory tools. Also, don't rely on this MCP to execute changes; it reads data, but for writing records or changing statuses, you'll need another toolset. If your only goal is to read the system health, use `get_info`; if you want a deep dive into an issue, grab `get_request`.

The Pain of Context Switching in Support

Right now, handling a single complex ticket means bouncing between tabs. You jump from the main Jira dashboard to check the queue status; you click over to the customer record to see their account history; then you open a separate knowledge base portal just to confirm if the solution is documented anywhere. It's a manual nightmare of copy-pasting IDs and juggling screens.

With this MCP, your agent manages that entire process for you. You ask one question—like, 'What's happening with ticket SD-123?'—and the agent handles all the necessary calls: pulling the service desk info, checking the current queue status, and getting the full request history. You get a single answer, not five tabs to read.

Get Full Service History with get_request

Before this, investigating an issue meant opening a ticket, scrolling back through dozens of comments, and manually checking who participated. You often missed key pieces of context or had to ask three different people for the same data points.

Now, using `get_request` pulls everything together. It gives you the full breakdown of participants and all custom fields in one go. What was once a 20-minute forensic investigation is now an instant summary.

Jira Service Management (JSM) MCP with 10 Tools

These ten tools allow your agent to perform everything from listing all active service desks to retrieving the full history of any single customer request.

#	TOOL	DESCRIPTION
01	`get_info`	Retrieves general system information about your JSM instance for health checks.
02	`get_request`	Pulls the complete record, participants list, and custom data fields for a single customer request.
03	`get_service_desk`	Gets project details and branding information for a specific support portal setup.
04	`list_customers`	Provides a list of all customers associated with a service desk, helping identify account holders.
05	`list_knowledge_bases`	Fetches a list of available knowledge articles for the service desk to guide issue resolution.
06	`list_organizations`	Lists all business units within JSM, useful for grouping support data by company.
07	`list_queues`	Retrieves the names and status of defined ticket queues, helping track where tickets are stuck in triage.
08	`list_request_types`	Lists all types of requests (like 'IT Help' or 'Hardware') available through the service portal.
09	`list_requests`	Gives a summary list of current customer requests, including keys and status, for monitoring purposes.
10	`list_service_desks`	Lists all active service desks in the system so you can determine the correct project ID to use.

See It in Action

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

U List all active service desks in JSM.

I'll fetch the list of available service desks from your Jira Service Management instance.

U Show me the latest customer requests.

I'll retrieve the most recent requests across all service desks for you.

U Check the queues for service desk ID '1'.

I'll look up the current queues and their status for that specific service desk.

Frequently Asked Questions

01 **How do I find out which service desks are available using Jira Service Management MCP?**

You use the `list_service_desks` tool. It immediately returns a list of all active project keys and names, so you know exactly what IDs to reference for your next query.

02 **Can I see how many tickets are waiting in the queue with Jira Service Management MCP?**

Yes, run `list_queues`. This shows you all defined queues and helps you gauge the current backlog count for a specific service desk.

03 **I need to check customer details; what tool should I use in Jira Service Management MCP?**

Use `list_customers`. It efficiently lists every customer associated with your designated service desk, helping you identify support recipients and their account data.

04 **How do I get the full history of a ticket using Jira Service Management MCP?**

You must use `get_request`, providing either the request key or ID. This pulls all necessary details, participants, and custom field values for deep investigation.

05 **Does this MCP help me find documentation? (Jira Service Management)**

Yes, call `list_knowledge_bases`. It lists all available articles for the service desk, letting you quickly identify if the answer is already documented.

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 `"jira-service-management-jsm": { "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

Jira Service Management (JSM) 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 Jira Service Management (JSM). 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	Jira Service Management (JSM) MCP
Server ID	`019d75bd-25a7-73f2-8f75-7e93c6c9f481`
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/jira-service-management-jsm.