MCP SERVER | NO CODE | CLOUD HOSTED

Klarna MCP

Manage payments and orders from chat.

Klarna MCP manages your entire e-commerce payment and order lifecycle. It lets your AI agent handle everything from creating checkout sessions to capturing payments, issuing refunds, or updating shipping details using simple natural language commands.

A+ Quality Score 100/100

checkout | buy-now-pay-later | order-management | payment-gateway | fulfillment | fintech

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

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.

Klarna MCP

8 tools available
Cloud-hosted on Vinkius

Your AI client can now manage complex payment flows without needing to log into a separate portal. This MCP connects directly to Klarna's APIs, letting you automate both the initial checkout phase and everything that happens after the purchase is made. You simply tell your agent what needs doing—like 'refund order 98765-XYZ' or 'update shipping for order ABC'—and it executes the steps through natural conversation. This capability makes monitoring payment status, processing captures, and handling refunds significantly faster than traditional methods. By connecting this MCP via Vinkius, you give your agent access to a complete suite of financial tools, treating the entire merchant account as just another set of functions available to your AI client.

Core Capabilities

01 — Initiating Checkout Sessions

The system creates payment sessions by requesting specific order amounts and item details.

02 — Placing Confirmed Orders

It formally places an order using a secured authorization token from your checkout widget.

03 — Completing Payments

You capture authorized orders to trigger the actual payment from the customer's account.

04 — Financial Adjustments

The system processes full or partial refunds, and it cancels any authorized but uncaptured orders.

05 — Fulfillment Updates

It updates shipping information and tracking numbers for existing orders directly in the record.

06 — Data Lookup

You retrieve detailed status, financial totals, and metadata for any given Klarna order ID.

One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to this MCP and provide your required credentials: the Klarna API Key (Username), Shared Secret, and Region Domain.

02 Ask your AI client to perform an action, such as 'Get details for order 98765-XYZ' or 'Create a payment session for $150.00'.

03 The MCP executes the necessary API calls and returns the specific status, tokens, or data you requested in plain text.

The bottom line is that your agent handles all the complex authentication and API interactions so you don't have to touch a dashboard.

Built For

This MCP is for e-commerce operations staff, financial controllers, and developers. If you spend time manually checking order statuses or running refunds in a web portal, this tool saves your day.

E-commerce Operations Manager

Handling post-sale issues like tracking updates or processing partial refunds by simply talking to the agent.

Financial Controller

Auditing payment session totals and checking order status across multiple transactions without logging into the dedicated financial portal.

Software Developer

Integrating Klarna's specific two-phase payment logic (authorization then capture) directly into custom application workflows using natural language triggers.

What Changes When You Connect

01 Process complex financial adjustments instantly. You can issue a partial refund using the `refund_klarna_order` tool, eliminating manual portal clicks for billing issues.

02 Automate your checkout flow entirely. Start by calling `create_payment_session`, which generates the client token needed to move forward in the purchase process.

03 Maintain full visibility into transactions with one command. Use `get_order_details` to pull status, totals, and metadata for any order ID you provide.

04 Handle post-purchase logistics effortlessly. If a customer changes their address, use `update_order_shipping` instead of emailing support or logging into the backend system.

05 Securely finalize sales cycles. The process involves using `create_klarna_order` after receiving an authorization token, followed by calling `capture_klarna_order` to collect funds.

06 Prevent financial loss with control functions. If a sale is authorized but never completed, you can use `cancel_authorized_order` to prevent accidental charges.

Real-World Applications

The Customer Changes Their Mind

A user needs to cancel an order that was just placed. They ask their agent to 'Cancel the authorized order for ID 123'. The agent uses `cancel_authorized_order` and confirms the cancellation, saving the operations team from manually navigating the refund flow.

Handling a Shipping Address Change

The customer calls support because they moved. Instead of emailing an address change request to fulfillment, the agent uses `update_order_shipping` with the new details, immediately updating the order record and letting shipping proceed.

Processing a Mid-Month Refund

A product return needs a partial refund. The controller asks their agent to 'Refund $50 on order 98765-XYZ'. The agent uses `refund_klarna_order` and confirms the exact amount and status change.

Confirming Payment Readiness

The developer needs to check if a payment session is ready for checkout. They ask the agent to 'Check details for payment session XYZ-ABC'. The agent uses `get_payment_session` and returns the live status data.

Patterns to Avoid

Treating it like a simple database lookup

✗ AVOID

Asking 'What is the order status?' and getting only text back. This misses the financial context, like whether funds have been captured or if there was an associated refund.

✓ INSTEAD

To get full visibility, use `get_order_details`. This tool pulls all necessary metadata, including transaction totals and current statuses, giving you a complete picture instead of just a single line item.

Bypassing the two-step process

✗ AVOID

Attempting to refund an order without first confirming it was captured. This leads to API errors because Klarna needs proof the funds were processed.

✓ INSTEAD

Always verify the status first using `get_order_details`. Once you confirm the payment is finalized and ready for adjustment, then use `refund_klarna_order`.

Forgetting to create the session

✗ AVOID

Trying to place an order without initiating a payment session first. The system fails because it doesn't know what amount or item set to associate with the purchase.

✓ INSTEAD

You must start by using `create_payment_session`, supplying the required order amounts and items. This initializes the necessary checkout flow before you can proceed to place the formal order.

The Right Fit

Use this MCP if your core pain point is managing the transactional complexity of e-commerce payments—specifically, moving from an authorized state to a captured state, or handling post-sale adjustments like refunds and shipping updates. You need a tool that understands the difference between initiating a payment session and actually capturing funds. Don't use it if you just need to read basic customer contact info; for simple reads, dedicated CRM

connectors are better. Also, don't rely on it solely for accounting ledger entry; this is a payments gateway, not your general ledger. This MCP excels at the 'action-taking' phase of e-commerce—the moments where status must change (like running `capture_klarna_order`).

The payment portal bottleneck.

Today, managing a single order requires logging into the Klarna merchant dashboard. You navigate to an order ID, check its status, find the action button for a refund or capture, input the correct amount, and then hit submit—all while dealing with dozens of tabs and copy-pasting IDs.

With this MCP, you simply tell your agent what needs doing. Instead of clicking through five screens to process an adjustment, you ask it to 'Refund $100 on order 98765-XYZ'. The payment happens instantly via the API call, and the status updates in natural conversation.

Klarna MCP: Controlling your payments.

The manual steps that vanish include checking order totals across different APIs, manually initiating captures after checkout, and cross-referencing shipping addresses between systems. These actions used to require a sequence of clicks and human verification at every step.

Now, your agent handles the entire transaction lifecycle—from `create_payment_session` through to `update_order_shipping`. It's not just an API wrapper; it's an active participant in your sales process.

Klarna MCP: 8 Tools for Payment Ops

These eight tools let you manage everything related to your e-commerce payments, from initial sessions and order placement to refunds and shipping updates.

#	TOOL	DESCRIPTION
01	`cancel_authorized_order`	Cancels an order that was authorized for payment but hasn't been captured yet.
02	`capture_klarna_order`	Triggers the final payment processing and captures the funds for a confirmed order.
03	`create_payment_session`	Initializes a new Klarna payment session, requiring details like the amount, currency, and specific items purchased.
04	`get_order_details`	Retrieves comprehensive status and financial data for any specified Klarna order ID.
05	`get_payment_session`	Gets the current details of an existing payment session using its unique identifier.
06	`create_klarna_order`	Places a formal order, requiring a valid authorization token obtained earlier in the checkout process.
07	`refund_klarna_order`	Processes a refund against an already captured and finalized order payment.
08	`update_order_shipping`	Updates the shipping address or tracking number associated with an existing customer order.

See It in Action

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

U Create a Klarna payment session for $150.00 in USD.

I've initialized the session. The order amount is 15000 (cents). Here is your session ID: 'abc-123' and the client token for your frontend widget.

U Show details for Klarna order ID '98765-XYZ'.

I've retrieved order 98765-XYZ. The current status is 'AUTHORIZED' and it has not been captured yet. The total amount is $150.00 USD.

U Capture order '98765-XYZ' for the full amount of $150.00.

Successfully triggered capture for order 98765-XYZ. Klarna has processed the payment, and the status has been updated to 'CAPTURED'.

Frequently Asked Questions

01 **How do I refund a Klarna order using the Klarna MCP?**

Use the `refund_klarna_order` tool. You just need to provide the specific order ID and the amount you want refunded, and the agent handles processing the full or partial refund.

02 **Can I check if a payment session is active with Klarna MCP?**

Yes, use `get_payment_session` to retrieve all current details for an existing payment session ID. This confirms its status and available data points.

03 **What’s the difference between placing an order and capturing it with Klarna MCP?**

Placing a formal order uses `create_klarna_order` (which needs a token), but that doesn't take money. You must then use `capture_klarna_order` to actually trigger the payment from the customer.

04 **Can I update shipping information using Klarna MCP?**

Yes, run the `update_order_shipping` tool. Just give it the order ID and the new tracking or address details, and the record gets updated instantly.

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 `"klarna": { "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

Klarna 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 Klarna. 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	Klarna MCP
Server ID	`019d75c1-5a4d-70cc-8f86-c3b4bc7c0262`
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/klarna.