MCP SERVER | NO CODE | CLOUD HOSTED

Paystack MCP

Manage payments and audit accounts via conversation.

Paystack connects your account directly to any AI client. You stop logging into dashboards and start talking to your payments system. This MCP lets you manage everything from checking transaction statuses and listing customers to tracking subscriptions and running refunds—all through natural conversation.

A+ Quality Score 100/100

online-payments | transaction-verification | customer-management | subscription-billing | fintech-api | africa-payments

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

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.

Paystack MCP

10 tools available
Cloud-hosted on Vinkius

Managing online revenue usually means hopping between Stripe, the bank portal, and the support CRM just to answer one simple question: Did this customer pay? With this Paystack MCP, your agent handles that. You simply ask your AI client a query about payments or accounts, and it pulls the data directly from Paystack. Need to know if a specific transaction went through last week? Ask. Want to see how many active subscriptions you have running right now? Just prompt it. It gives you real-time details on customer profiles, current billing plans, or recent transfers, allowing your agent to act as your financial operations manager without ever leaving the chat window. You get full control over listing transactions, verifying payment status, and reviewing refund records—all guided by the Vinkius catalog.

Core Capabilities

01 — Check transaction history

List all payments that have occurred, or check the detailed status of any single reference ID.

02 — Manage customer records

Retrieve lists of customers and fetch specific profiles to review their payment activity and metadata.

03 — Audit recurring revenue

Monitor all active subscriptions, view billing plans, and track past renewal cycles.

04 — Process financial reconciliation

List recent refunds and outbound transfers to keep your books accurate for accounting purposes.

05 — View payment configurations

See all the configured billing plans and their associated settings within Paystack.

One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to this MCP on Vinkius and provide your specific Paystack Secret Key (sk_...).

02 Connect it to your preferred AI client, like Cursor or Claude.

03 Use natural language prompts to ask questions about payments. Your agent runs the necessary functions and delivers structured data right in the chat.

The bottom line is that you treat your payment gateway like another API endpoint—you just talk to it instead of writing code to call it.

Built For

E-commerce owners who hate paying for expensive dashboard add-ons, and support agents tired of manually checking multiple systems. If your job involves confirming payment status or reconciling financial data daily, this is for you.

Support Specialist

Looks up a customer's complete payment history to resolve billing disputes quickly without escalating the ticket.

E-commerce Owner

Verifies if an order was paid for instantly during a chat with a customer, ensuring sales don't stall waiting on bank confirmation.

Financial Analyst

Extracts large lists of transactions and refund data to prepare reports for monthly accounting reconciliation.

What Changes When You Connect

01 Stop manually checking dashboards. You can use the `verify_transaction` tool to check a payment's real-time status immediately, without logging into any other system.

02 Gain full visibility over customer finances by listing all users using `list_customers` and then getting deep details on one user with `get_customer`.

03 Simplify billing audits. You can use `list_subscriptions` to see everything that's active, and then run `get_subscription` for a detailed breakdown of any single plan.

04 Keep your books clean by running `list_refunds` or `list_transfers`. This lets you quickly gather the necessary records needed for financial reporting.

05 Handle payments like a pro. You can use the MCP to view all available payment plans using `list_plans`, which helps understand what options are available to your customers.

Real-World Applications

Customer support needs immediate proof of payment.

A customer tweets, 'My order hasn't shipped.' The support agent asks the AI client, 'Check transaction status for reference XYZ.' The MCP uses `verify_transaction` and immediately replies: 'Success. Payment of $50 was received on Monday.' The issue is resolved in one chat exchange.

The finance team needs to reconcile last month's payments.

Instead of exporting 12 separate CSV files, the analyst asks the AI client to 'List all transactions and all refunds for June.' The MCP uses `list_transactions` followed by `list_refunds`, compiling a single, unified data set in seconds.

An e-commerce owner needs to check a high-value account.

The agent asks the AI client to 'Show me all details on customer John Doe.' The MCP uses `get_customer` and provides not only contact info but also their payment history, allowing the owner to manage relationships better.

A developer needs to audit subscription billing.

The dev asks the AI client, 'What are our top three active plans?' The MCP uses `list_plans` and then can pull detailed data by running `list_subscriptions`, ensuring there aren't any orphaned or outdated payment configurations.

Patterns to Avoid

Treating the API like a simple database query.

✗ AVOID

The user thinks they just need to run 'SELECT * FROM transactions WHERE customer = John Doe.' This fails because payments are complex and require status checks, not just raw data retrieval.

✓ INSTEAD

You must use specific tools. To get payment details for one person, ask the agent to `get_customer` first, then follow up with `list_transactions` or `verify_transaction` using a reference ID.

Trying to handle multiple account types in one go.

✗ AVOID

The user asks 'Show me all money and customers.' This is vague. The AI will fail because it doesn't know which type of data or time range you care about.

✓ INSTEAD

Be specific with your tools. If you need general records, use `list_customers`. If you need to see the flow of cash leaving the account, use `list_transfers`.

Forgetting that data is paginated.

✗ AVOID

The user asks 'List all transactions.' They only get the first 10 records and assume they have everything. The remaining months of activity are invisible.

✓ INSTEAD

When asking for lists, ask your agent to handle pagination or use a date range (e.g., 'list_transactions from last quarter'). Don't just rely on the default list call.

The Right Fit

Use this MCP if your primary operational bottleneck is checking payment status, retrieving customer financial history, or reconciling transactions across different internal systems. If you need to know *if* a payment happened and *who* paid it, this is essential. Don't use it if you are building the core payment logic (that needs backend code). Also, don't rely on it for tax calculation—it only provides raw financial records. If your goal is purely analytics or reporting that requires custom data modeling outside of simple list retrieval, you might be better off connecting a dedicated BI tool instead of relying solely on transaction logs.

The friction point in e-commerce support today isn't the conversation; it's the manual data gathering.

Today, when a customer calls about a failed payment or asks to see their billing cycle, you open your CRM. Then you copy the user ID and jump over to the payments dashboard. You search by ID, find the transaction record, manually check its status, and then copy that confirmation back into your chat window. This process wastes minutes on every single ticket.

With this MCP, the conversation handles it. Instead of jumping between tabs and copying reference numbers, you simply prompt your agent: 'Check the payment status for customer XYZ.' The tool uses `verify_transaction` under the hood, pulls the definitive answer instantly, and presents a clean, actionable summary in the chat.

Paystack MCP gives you immediate financial clarity.

The manual steps that disappear are logging into separate payment portals just to cross-reference data. You stop having to copy a refund reference from one screen and paste it into another system for tracking.

Now, your agent sees the entire picture: you can list all subscriptions using `list_subscriptions` and immediately confirm if the failure was due to an expired card or a plan limit issue. It's that level of integrated financial awareness.

Paystack MCP: 10 Tools for Financial Operations

These tools give your agent the ability to read payment records, check status codes, list customers, and manage billing details directly through conversation.

#	TOOL	DESCRIPTION
01	`get_customer`	Retrieves the full details for a specific customer profile in Paystack.
02	`get_subscription`	Pulls up all information related to one active or past subscription.
03	`get_transaction`	Fetches the complete record and metadata for a single, specific transaction ID.
04	`list_customers`	Gives you a list of all customers registered in your Paystack account.
05	`list_plans`	Provides a comprehensive view of every payment plan currently configured.
06	`list_refunds`	Lists all recent refunds that have been processed through the system.
07	`list_subscriptions`	Retrieves a list of all active and archived subscriptions you manage.
08	`list_transactions`	Generates a paginated list of recent transactions that occurred on the account.
09	`list_transfers`	Provides an overview and list of all money transfers initiated from Paystack.
10	`verify_transaction`	Checks the current, definitive status (success or failure) of a specific transaction reference.

See It in Action

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

U Check the status of transaction reference 'ref_789456'.

Transaction 'ref_789456' was successful. The customer paid 5,000 NGN on Oct 24, 2026.

U List the last 5 customers registered in our Paystack account.

I've retrieved the latest customers. Here are the 5 most recent signups, including their names, emails, and Paystack customer codes.

U How many active subscriptions do we currently have?

You currently have 124 active subscriptions across various plans. Would you like a breakdown by plan type?

Frequently Asked Questions

01 **How do I check transaction status with Paystack MCP?**

You use the `verify_transaction` tool by giving it the specific reference ID. The agent confirms immediately if the payment succeeded or failed, and why.

02 **Can Paystack MCP list all my customers?**

Yes, you can run `list_customers` to get a full roster of users. From there, you can narrow down details for one person using `get_customer`.

03 **Does Paystack MCP help with recurring billing issues?**

You can track this by listing all active plans via `list_plans`, and then running `list_subscriptions` to see which customers are currently billed.

04 **What is the difference between list_transactions and verify_transaction in Paystack MCP?**

`list_transactions` gives you a paginated history (a list of many). `verify_transaction` checks one single, specific transaction status right now.

05 **Is Paystack MCP useful for accounting reconciliation?**

Absolutely. You can use `list_refunds` and `list_transfers` to pull the exact data needed to reconcile your financial records at month-end.

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 `"paystack": { "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

Paystack 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 Paystack. 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	Paystack MCP
Server ID	`019d75f0-a574-71ed-9b10-3a4379f165ed`
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/paystack.

© 2026 Vinkius · vinkius.com · support@vinkius.com