MCP SERVER | NO CODE | CLOUD HOSTED

Privy MCP

Manage user and wallet states through conversation.

Privy connects your AI client directly to Web3 identity infrastructure. Manage users, create wallets across Ethereum, Solana, Bitcoin, and Sui, and execute blockchain actions—all through natural conversation. Check user profiles, provision new wallets in batches up to 100, or retrieve specific transaction details without leaving your agent environment.

F Quality Score 3.6/100

web3 | embedded-wallets | user-onboarding | authentication | crypto-infrastructure

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

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.

Privy MCP

12 tools available
Cloud-hosted on Vinkius

This MCP lets you manage the core infrastructure of any Web3 application using simple language commands. Instead of writing complex API calls for every new user or wallet status check, you talk to your AI client, and it handles the heavy lifting. You can instantly create a new user object, look up existing profiles by email address, or provision wallets in large batches. Need to confirm if a transaction succeeded? Just ask. The system retrieves specific wallet data or checks recent transactions using external IDs. This ability to manage complex identity lifecycle—from creating accounts to performing RPC actions on managed wallets—all via chat is what makes this MCP invaluable. When you connect it through Vinkius, you get access to the full spectrum of user and wallet operations in one place.

Core Capabilities

01 — User Profile Lookup

Find users by email address or search for profiles using keywords, retrieving detailed metadata.

02 — Wallet Provisioning

Create new crypto wallets across multiple chains (Ethereum, Solana, Bitcoin, Sui), either one-by-one or in bulk batches of up to 100.

03 — Identity Management

Create new user records and securely delete old ones when they are no longer needed.

04 — Wallet State Modification

Update wallet metadata, change ownership policies, or retrieve specific details for any known wallet ID.

05 — Blockchain Action Execution

Execute direct RPC calls through managed wallets, allowing actions like signing messages or sending transactions.

One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to this MCP and enter your Privy App ID and App Secret credentials.

02 Connect the MCP to any compatible AI client (Claude, Cursor, etc.) within your chosen workspace.

03 Start interacting with it by asking your agent to perform actions like 'search for users' or 'batch create wallets'.

The bottom line is you manage complex Web3 identity and wallet infrastructure using natural language commands inside your existing AI workflow.

Built For

This MCP is essential for any technical team dealing with user onboarding or crypto-native products. It solves the pain point of having to switch between developer tools, documentation sites, and multiple dashboards just to check a user's wallet status or create test accounts.

Web3 Developer

Needs to quickly provision test wallets or inspect existing user account data without leaving the code editor environment.

Product Manager

Must verify linked accounts, check wallet statuses, or confirm profile details when planning new features or handling support requests.

DevOps Engineer

Automates user lifecycle management and updates complex wallet policies using a conversational interface for auditing purposes.

What Changes When You Connect

01 Instead of manually querying APIs to check if a user exists, you simply ask your agent using the `get_user` or `get_user_by_email` tools. It handles the lookup immediately.

02 Need to test out a new feature? Use `batch_create_wallets` to provision up to 100 test wallets at once, eliminating slow manual setup steps.

03 When you need to confirm a complex blockchain action, use `get_transaction_by_external_id`. You get the transaction status without needing raw node access.

04 Managing identity is simpler: Use `create_user` to onboard accounts and `delete_user` when an account needs to be retired, all via conversational commands.

05 The flexibility of `wallet_rpc` lets you perform specialized actions—like signing a message or sending a transaction—that go beyond simple reads.

Real-World Applications

Onboarding a new client

A Product Manager needs to verify if a potential client, Jane Doe, already has an account. They prompt their agent: 'Get the user details for jane@company.com.' The agent uses `get_user_by_email` and immediately returns her profile ID and linked wallet status.

Auditing system accounts

A DevOps Engineer must update the ownership policy on a critical treasury wallet for compliance. They instruct their agent to use `update_wallet`, specifying the new owner details, ensuring the change is logged and executed immediately.

Running large-scale tests

A developer needs 50 fresh test wallets for a network stress test. Instead of running 50 separate scripts, they ask their agent to `batch_create_wallets` with the desired chain type and receive all credentials instantly.

Investigating failed transfers

A support team member receives a ticket about a missing payment. They use `get_transaction_by_external_id`, providing the tracking ID, and the agent confirms if the transaction was successful or stuck.

Patterns to Avoid

Treating it like simple database search

✗ AVOID

Trying to use this MCP just for reading basic user info, when a dedicated CRM API would be better.

✓ INSTEAD

Use `get_user` or `search_users` if you only need profile data. But if the goal is managing wallet state, always use `update_wallet` or `wallet_rpc`.

Writing boilerplate code for every action

✗ AVOID

Manually writing a Python function every time you need to check a user's account status.

✓ INSTEAD

Just talk to your agent. Use the tool calling mechanism with `get_user` or `create_wallet`. The MCP handles the underlying API complexity.

Assuming wallet details are static

✗ AVOID

Calling an outdated function assuming a wallet's ownership hasn't changed since last month.

✓ INSTEAD

Always verify the current state first. Use `get_wallet` before attempting to use `update_wallet` or execute any RPC command.

The Right Fit

Use this MCP if your workflow involves complex, multi-step actions in a Web3 context—specifically creating users, managing wallet ownership/policies, or executing transactions. It's designed for conversational orchestration of state changes. Don't use it if you only need to read simple data that doesn't involve crypto identity (e.g., just listing product names). If your task is purely front-end UI work or reading non-blockchain related records, look into a standard database connector instead. However, if those records *are* tied to an account or wallet ID, this MCP is necessary because it provides the tools to verify and manipulate that core identity state.

Handling user accounts used to be a mess of dashboards and APIs.

Today, if you need to check a user's status or create a test wallet, you usually have to hop through three different systems: the internal database for their profile, the crypto explorer for transaction history, and then write separate scripts just to provision them a clean new account. It’s slow, and every system has its own API keys.

With this MCP, all of that gets pulled into one conversation. You tell your agent what you need—say, 'Find user A's wallet status.' The agent handles the lookup, checks the blockchain transaction history, and tells you the full answer in a single response.

Privy MCP: Manage accounts & wallet states

You no longer need to write boilerplate code for basic user lookups or wallet creation. Whether you use `get_user_by_email` or the bulk action of `batch_create_wallets`, the complexity is abstracted away.

The result is simple: Your agent acts like a specialist who knows every facet of your Web3 identity stack, allowing you to move from asking questions to executing infrastructure changes instantly.

Privy MCP: 12 Tools for Identity Management

These tools allow you to programmatically perform every key action related to user accounts and Web3 wallet infrastructure directly through your AI client.

#	TOOL	DESCRIPTION
01	batch_create_wallets	Creates multiple new crypto wallets in bulk, supporting up to 100 addresses at once.
02	create_user	Registers a brand-new user profile object and links it to one or more accounts.
03	create_wallet	Generates a single new wallet for the system to use with a user.
04	delete_user	Permanently removes a specified user record from the database.
05	get_transaction_by_external_id	Retrieves transaction details by providing an external tracking ID.
06	get_transaction	Fetches full information about a specific blockchain transaction.
07	get_user_by_email	Finds and returns a user's profile data using their email address.
08	get_user	Retrieves all available metadata for a specific user ID.
09	get_wallet	Gets the current details and status of a known wallet address.
10	search_users	Searches the entire user database using keywords or partial profile data.
11	update_wallet	Modifies an existing wallet's metadata, policies, or ownership settings.
12	wallet_rpc	Executes arbitrary blockchain Remote Procedure Calls (RPC) actions on a managed wallet.

See It in Action

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

U Search for users with the term 'beta-tester' in Privy.

I found 3 users matching 'beta-tester'. The most recent is user `did:privy:cl...` linked to `tester1@example.com`. Would you like to see their full profile details?

U Create a new Ethereum wallet with the display name 'Main Treasury'.

Successfully created a new Ethereum wallet. ID: `wa_123...`, Address: `0x71C...`. It is now ready for use in your application.

U Get the user details for email 'alice@company.com'.

Retrieved profile for Alice. She has a linked Google account and one embedded Solana wallet (`6xY...`). Her Privy ID is `did:privy:ck...`.

Frequently Asked Questions

01 **How do I create multiple wallets using the Privy MCP?**

You use the `batch_create_wallets` tool. This lets you provision up to 100 new crypto wallets in a single command, which is much faster than creating them one by one.

02 **Can I check if a user exists with Privy MCP?**

Yes, you can use `get_user` or `search_users`. If the agent returns profile data, the account exists; otherwise, it confirms that no matching records were found.

03 **What if I need to change a wallet's owner?**

You use the `update_wallet` tool. This lets you modify metadata and policies for an existing wallet ID, allowing you to safely transfer ownership or update credentials.

04 **Does Privy MCP handle transaction history lookups?**

Yes. You can fetch specific transactions using `get_transaction` or narrow down the search by providing a known external tracking identifier via `get_transaction_by_external_id`.

05 **Is Privy MCP only for Ethereum wallets?**

No, it supports multiple chains. You can provision and manage assets across several networks including Solana, Bitcoin, Sui, and Ethereum.

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 `"privy": { "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

Privy 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 Privy. 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	Privy MCP
Server ID	`019e38db-4692-73e4-9723-95a742f7d078`
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/privy.