MCP SERVER | NO CODE | CLOUD HOSTED

Storybook MCP

Inspect component structure and usage instantly.

Storybook MCP connects your AI client directly to your design system's component library. You can programmatically browse every UI element, inspect specific components for their properties and metadata, locate documentation paths, and generate isolated preview URLs—all without leaving your coding environment.

A+ Quality Score 100/100

ui-components | design-system | component-library | frontend-development | ui-testing | documentation

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

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.

Storybook MCP

6 tools available
Cloud-hosted on Vinkius

Your AI agent can now treat your Storybook instance like a functional part of your codebase. Instead of opening a dozen tabs to check component props or find the source code structure, you just ask it. It maps out your entire design system—from top-level categories down to individual atoms. You'll stop wasting time jumping between documentation sites and your IDE. This MCP lets you query component libraries for default arguments and validate complex prop signatures instantly. Whether you're mapping out the overall architecture using Vinkius, or just trying to confirm if a Button component supports an 'isLoading' state, the data is right there in your context. You get immediate feedback on how components are structured, what source paths they rely on, and even isolated sandbox previews for testing changes before committing any code.

Core Capabilities

01 — Map design system structure

List the top-level categories and folder organization of your component library.

02 — Search for components by name

Find specific UI elements within the Storybook based on a keyword or partial name.

03 — Get component metadata

Retrieve detailed information, including default arguments and expected properties, for any given component.

04 — Generate sandbox previews

Create temporary, isolated URLs so you can safely preview how a component will look before implementing it.

05 — Locate source documentation paths

Extract local source code file paths and implementation notes directly from the Storybook index.

One Click on Vinkius — From Prompt to Execution

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

01 Install the Storybook MCP module into your AI client's configuration.

02 Provide your deployed Storybook instance URL in the connection parameters.

03 Ask your agent to perform a task, like 'Find all input components and list their required arguments.' The tool executes the necessary lookups and returns structured data.

The bottom line is you get component-level intelligence fed directly into your chat or IDE prompt.

Built For

Frontend engineers who spend too much time context switching between documentation and code; UI/UX designers who need programmatic validation of design tokens; technical writers needing automated ways to track component usage.

Frontend Engineer

Uses the MCP to confirm props signatures or find existing components instead of writing boilerplate code from scratch.

UI/UX Designer

Validates if a specific design token, like a color variable or spacing unit, is correctly implemented across all rendered component types.

Technical Writer

Automatically gathers the necessary documentation paths and usage examples for newly launched components to keep guides accurate.

What Changes When You Connect

01 Avoids context switching. You don't have to leave your chat or IDE to check if a prop exists or what its type is; the agent pulls that info directly from Storybook.

02 Reduces code duplication. By using tools like `search_components` and `get_story_args`, you ensure every new feature uses an existing, tested component rather than building one from scratch.

03 Speeds up documentation maintenance. Technical writers can use the MCP to automatically pull source paths via `extract_docs_guidance`, keeping guides current with zero manual copy-pasting.

04 Safe testing environment. Generating a preview URL using `get_preview_url` lets you test component changes in an isolated iframe before committing code, catching visual bugs early.

05 Global architectural view. Use `list_categories` to map the entire design system structure at once, giving new team members a bird's-eye view of all available building blocks.

Real-World Applications

A developer needs to confirm prop requirements for an existing widget.

The engineer asks the agent: 'What are the required props for the user avatar component?' The MCP uses `get_story_args` and immediately returns a list of properties, their types (string, boolean), and whether they have default values. No opening documentation pages needed.

A designer wants to see how a component looks with different states.

The designer prompts the agent: 'Show me an isolated preview of the primary button in its disabled state.' The MCP generates a specific, temporary `get_preview_url`, allowing them to validate the visual output without touching the staging environment.

A new hire needs to understand the component hierarchy.

The agent runs `list_categories` and maps out 'Atoms,' 'Molecules,' and 'Layouts.' The new hire can then use `list_components` on a specific category, like 'Atoms/Input' to see every available input variant.

A developer needs to update documentation for a core component.

The engineer asks the agent to find the source code path and usage notes. The MCP uses `extract_docs_guidance` to locate the file (`./src/components/NavBar.tsx`) and pulls implementation details, which are then copied directly into the README.

Patterns to Avoid

Manual Component Lookup

✗ AVOID

A developer has to open Storybook, navigate manually through categories (Atoms > Buttons), and then find the specific component page just to copy a prop name.

✓ INSTEAD

Ask your agent directly using `search_components` for 'Button.' The agent finds it instantly. Then use `get_story_args` on the result to get all props in one clean output.

Blind Code Generation

✗ AVOID

A developer copies prop names from a screenshot or old documentation, but forgets which ones are actually required and what their correct data types are.

✓ INSTEAD

Always use `get_story_args` when generating code. It verifies the component's actual metadata and tells you if the property is mandatory, preventing runtime type errors.

Testing in Staging

✗ AVOID

A designer makes a visual change to a component and has to deploy it to a staging environment just to check how it looks. This takes time and risks breaking other things.

✓ INSTEAD

Use `get_preview_url`. It creates an isolated, temporary sandbox iframe specifically for that component's preview, keeping your main application stable.

The Right Fit

Use this MCP if your workflow involves interacting with a complex, established design system. You need to verify component existence, check prop requirements, or map out the overall architecture. For instance, if you are building a new widget and need to know if 'Card' components exist, use `search_components`. If you are confirming that the 'Avatar' needs a URL string as an input, use `get_story_args`. Don't use this if your problem is simple data retrieval (like fetching user IDs from a database). For raw data lookups, connect to a dedicated Data MCP. Use it if you need component context; don't use it if you just need general knowledge or text generation.

Component documentation and props are always out of sync.

Right now, figuring out a component's required properties is a nightmare. You have to jump between the design spec sheet, the Storybook live view, and finally the actual source code file. Then you copy-paste prop names, hoping they haven't changed since last week.

With this MCP connected through Vinkius, your agent reads the component definition directly. You just ask it what props a component needs—it gives you the name, type, and default value immediately. The friction is gone.

Storybook MCP: Get immediate structural context.

Instead of manually running `ls` commands or traversing folder structures in a file explorer to map out the design system, you can ask your agent to run `list_categories`. It maps the entire hierarchy for you. Then use `list_components` to drill down into every single element.

The process shifts from manual exploration to conversational discovery. You get a complete, structured inventory of every component available in the library, instantly.

Storybook: 6 Tools for Component Inspection

Use these tools to programmatically list components, retrieve metadata, find source paths, and validate your entire design system's component library.

#	TOOL	DESCRIPTION
01	`extract_docs_guidance`	Gets guidance on how to read documentation for a component by extracting its local source code paths.
02	`get_preview_url`	Generates the unique URL needed to run an isolated sandbox preview of a component's visual state.
03	`get_story_args`	Retrieves detailed metadata and default arguments, such as required props, for a specific UI component.
04	`list_categories`	Lists the top-level structural categories (like Atoms or Molecules) that organize your design system.
05	`list_components`	Provides a complete list of all available UI components within the entire Storybook library.
06	`search_components`	Searches for specific components using keywords or partial names across the design system index.

See It in Action

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

U Search for Button components in my Storybook and show their props.

Scanning Storybook index...
Discovered UI Matches (`search_components`):
- `Button/Primary-Action` (ID: comp-btn-01)
- `Button/Ghost` (ID: comp-btn-02)

Inspecting props via `get_story_args` on `Button/Primary-Action`:
- **Props**: `label` (string), `isActive` (boolean), `variant` (primary/secondary), `onClick` (function).
I can generate an isolated preview URL if needed.

U List the categories in the design system and browse the components rendered.

Mapping the interface structure...

Categories (`list_categories`):
- `Atoms`
- `Molecules`
- `Layouts`

Components (`list_components`):
- `Atoms/Input`
- `Atoms/Tooltip`
- `Layouts/Sidebar`

Documentation endpoints have been dynamically assembled.

U Extract the local source code paths from the index for the Navigation Bar component and generate an iframe preview.

Checking local references logically...

Source Paths (`extract_docs_guidance`):
Resolved path: `./src/components/NavBar.tsx`. Documentation notes suggest importing `NavProps` directly.

Preview Generated (`get_preview_url`):
The isolated sandbox iframe is ready at `//storybook-instance/iframe.html?id=layout-navbar--default`.

Frequently Asked Questions

01 **How do I use Storybook MCP to find out what props a component needs?**

You ask your agent to check a specific component. The MCP uses `get_story_args` and returns a clean list of all necessary properties, their data types, and whether they have default values.

02 **Can Storybook MCP help me see how a component looks before coding it?**

Yes. You prompt the agent for a preview URL. It runs `get_preview_url` to generate an isolated sandbox iframe, letting you validate the visual output safely.

03 **Which tool should I use if I want to see all available components?**

Use `list_components`. This function gives you a comprehensive inventory of every single UI element defined in your entire design system, organized by category.

04 **Does Storybook MCP help with documentation linking?**

Yes. You can use `extract_docs_guidance` to pull the local source code paths and relevant implementation notes directly from the index, saving you manual file searching.

05 **What if I only know a keyword for a component?**

No problem. Use `search_components`. It finds components by name or partial keywords across the whole library, narrowing down your focus quickly.

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 `"storybook": { "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

Storybook 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 Storybook. 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	Storybook MCP
Server ID	`019d760d-ba7d-7209-a875-6eb00b2db2be`
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/storybook.