MCP SERVER | NO CODE | CLOUD HOSTED

Nanoid Generator MCP

Generate Short, Collision-Proof IDs for Any System

Nanoid Generator creates unique IDs that are URL-safe, cryptographically random, and short enough for any modern database or web link. Forget long 36-character UUID strings; this MCP generates collision-resistant identifiers in just 21 characters, making your data cleaner and faster to use across every system.

A+ Quality Score 100/100

unique-id | cryptography | url-safe | randomness | data-integrity | uuid-alternative

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/nanoid-generator

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.

Nanoid Generator MCP

1 tools available
Cloud-hosted on Vinkius

When you need a unique identifier, length shouldn't be a problem. This MCP lets any agent generate production-grade IDs that are inherently URL-safe. It uses cryptographic randomness to guarantee security while keeping the output much shorter than traditional UUIDs. You can tell your AI client exactly what format you need—whether it’s purely numeric for an order reference or custom characters for internal tokens. The resulting ID is optimized for data integrity, meaning you don't have to worry about encoding headaches when generating millions of records. It makes reliable, short-form identifiers available directly within the Vinkius catalog.

Core Capabilities

01 — Generate standard unique IDs

The tool creates collision-resistant, URL-safe strings perfect for general database keys and API resources.

02 — Create numeric order references

You can force the output to use a specific alphabet, letting you generate IDs composed only of numbers for human readability.

03 — Produce secure, custom-alphabet keys

It allows defining any character set, ensuring the generated ID fits your exact business format while maintaining high entropy and security.

One Click on Vinkius — From Prompt to Execution

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

01 Specify the requirements for the unique identifier—for instance, if you need a standard 21-character key or a numeric code.

02 Send the request to your AI client, which executes the `generate_nanoid` tool using the defined constraints (size, alphabet).

03 Receive the final ID: a short, cryptographically secure string ready for immediate use in URLs, database entries, or API payloads.

The bottom line is that you get clean, compact IDs instantly, without having to manage complex encoding rules or worry about generating strings that are too long for your infrastructure.

Built For

This MCP is critical for backend developers and data architects who hate dealing with database bloat caused by unnecessarily long identifiers. It's for the engineer tired of UUIDs breaking up URLs or requiring clumsy encoding steps just to store a simple record ID.

Backend Engineer

Uses it when designing API endpoints and database schemas, ensuring every resource key is short, fast, and universally safe.

DevOps Specialist

Needs unique IDs for tracking deployment artifacts or log lines where space efficiency matters across massive data streams.

Product Manager (Technical)

Defines the required ID formats, making sure that user-facing elements like short links or order numbers are both readable and globally unique.

What Changes When You Connect

01 Saves space and bandwidth. Instead of 36-character UUIDs cluttering every URL or database row, this MCP produces highly compact identifiers in just 21 characters.

02 Guaranteed security. It uses crypto.getRandomValues() to generate IDs, ensuring the randomness is cryptographically strong—it's not just a random string generator.

03 Total flexibility with custom alphabets. Need an ID that only contains numbers for human readability? Or maybe letters and hyphens for a specific client format? You define it.

Real-World Applications

Building a global URL shortener

The marketing team needs unique links for thousands of campaigns. Instead of using a standard UUID, the agent uses nanoid to generate compact, collision-safe IDs that fit perfectly into `bit.ly/XYZ` format without requiring any complex encoding.

Creating e-commerce order references

The operations team needs human-readable identifiers for customer support tickets and invoices. They prompt the agent to generate_nanoid with a numeric alphabet, resulting in an ID like '47829163' that is easy for staff to read and communicate.

Securing temporary API keys

A partner integration needs high-entropy access tokens. The agent uses the tool, specifying a large character set and ensuring the resulting ID has enough bits of entropy for maximum security, avoiding weak or predictable key generation.

Indexing internal document metadata

The data science team is building a knowledge graph. They use nanoid to generate unique keys for each indexed document chunk, ensuring that the primary index ID doesn't conflict with other system IDs and remains clean.

Patterns to Avoid

Relying on standard UUIDs

✗ AVOID

Using a full 36-character UUID in every single URL path segment, which looks messy and wastes characters.

✓ INSTEAD

Instead, use the `generate_nanoid` tool. It gives you an ID that's shorter, cleaner, and just as secure for web paths or database lookups.

Using generic random string functions

✗ AVOID

Asking your agent to 'make a random 10-character key,' which often results in low entropy or predictable patterns.

✓ INSTEAD

Always use `generate_nanoid`. It's cryptographically secured and uses proper methods, so you know the ID is genuinely unique.

Ignoring format requirements

✗ AVOID

Accepting a default nanoid output when your system requires IDs that are strictly numeric (e.g., for accounting systems).

✓ INSTEAD

When calling `generate_nanoid`, pass the appropriate alphabet string, like '0123456789', to ensure the resulting ID meets strict business rules.

The Right Fit

Use this MCP if your primary pain point is identifier length or encoding complexity. If you need a unique key for a URL, database index, or API resource, and you want it shorter than 36 characters, nanoid is the answer. The tool's ability to customize the alphabet makes it ideal for systems with strict format requirements (e.g., only digits). Don't use this if you need complex relational data structures; that requires a different kind of tooling. Also, don't assume 'random' means safe—always verify the output is generated using `generate_nanoid` to guarantee cryptographic strength.

Managing Unique IDs Used To Be a Nuisance

Think about it: every time you build something that needs an ID—a user account, a document record, a short link—you get bogged down by UUIDs. They're 36 characters long, making URLs look awful and forcing developers to write extra code just to handle percent-encoding in the database layer.

With this MCP, you pass your requirement (e.g., 'I need an ID for a document index') to your agent. It generates_nanoid handles the heavy lifting, giving you a clean, compact, and cryptographically sound string that just works.

Generate Unique IDs with Nanoid Generator MCP

You no longer have to copy-paste long UUIDs into documentation or worry about them breaking the aesthetic of a public-facing URL. You can also specify that your ID must be purely numeric, which was previously a manual filtering headache.

The result is immediate: reliable, short identifiers for any context—from internal tokens to customer-facing links. It's clean data input for your agent.

Nanoid Generator: 1 Tool Available

This MCP allows you to generate unique identifiers with specific controls over length, alphabet, and cryptographic security.

#	TOOL	DESCRIPTION
01	`generate_nanoid`	Generates a unique URL-safe ID using cryptographic randomness, supporting custom alphabets for specific formats like numeric order codes or tokens.

See It in Action

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

U Generate a short, unique ID for a URL shortener like bit.ly.

ID: V1StGXR8_Z5j | 12 chars, URL-safe, no encoding needed. Collision-safe for billions of links.

U I need an 8-digit numeric order ID for our e-commerce checkout.

ID: 47829163 | Numeric alphabet, 8 digits. Perfect for human-readable order references.

U Create a secure 32-character API key for our partner integration.

ID: kB7xZ9pQ2mN5wR8jF3vH6yT1sL4cD0aX | 32 chars, crypto-random. Entropy: 190+ bits.

Frequently Asked Questions

01 **Is the nanoid generator secure enough for API keys?**

Yes. The tool uses cryptographic randomness, ensuring high entropy required for production-grade security. You can generate strong keys that are also compact and URL-safe.

02 **Can I make an ID that is only numbers using nanoid generator?**

Absolutely. When generating_nanoid, you simply pass the numeric alphabet string ('0123456789') to constrain the output characters.

03 **What's the main difference between this MCP and a standard UUID library?**

The key difference is length. This MCP produces IDs that are significantly shorter than traditional UUIDs, while maintaining the same high level of collision resistance.

04 **How do I make sure my generated ID doesn't conflict with existing ones?**

Because it uses cryptographic randomness and is highly efficient, the chance of a collision is astronomically low. You don't need to worry about that manually.

05 **Does nanoid generator support different character sets for IDs?**

Yes, you have full control over the alphabet used by passing a specific string during the `generate_nanoid` call. This is great for internal system codes.

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 `"nanoid-generator": { "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

Nanoid Generator 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 Nanoid Generator. 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	Nanoid Generator MCP
Server ID	`019e38c5-1e02-719b-9316-45d08a70ebeb`
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/nanoid-generator.