MCP SERVER | NO CODE | CLOUD HOSTED

UUID & ULID Generator MCP

Guarantee unique, cryptographically sound keys.

UUID & ULID Generator provides guaranteed, cryptographically perfect unique identifiers for your applications. Stop relying on LLMs to create IDs; this MCP generates collision-free v4 UUIDs and time-sortable ULIDs with native crypto randomness. It ensures your databases get mathematically sound keys every time.

A+ Quality Score 100/100

cryptography | randomness | data-integrity | id-generation | entropy | database-schema

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

UUID & ULID Generator MCP

2 tools available
Cloud-hosted on Vinkius

When you're building systems that need reliable data, generating unique identifiers is non-negotiable. Standard AI models sometimes fail here; they might hallucinate invalid ID formats or repeat the same key when asked for a batch of values. This MCP solves that fundamental problem by providing true cryptographic randomness directly to your agent. You can generate mathematically perfect v4 UUIDs, which are guaranteed collision-free. Need IDs that keep database indexing fast and records chronological? Use the ULID generator. It creates Universally Unique Lexicographically Sortable Identifiers. Connecting this specialized tool through Vinkius lets you guarantee data integrity regardless of how complex your agent's workflow gets.

Core Capabilities

01 — Create collision-proof UUIDs

Generates standard, 128-bit universally unique identifiers using cryptographic randomness.

02 — Generate time-sortable ULIDs

Produces ULIDs that are guaranteed to sort chronologically, optimizing database primary keys.

03 — Guarantee data uniqueness

Ensures every ID created is mathematically unique and adheres to industry standards (RFC 4122).

One Click on Vinkius — From Prompt to Execution

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

01 Your agent recognizes the need for a guaranteed, structured identifier.

02 The agent invokes this MCP's tools, specifying whether it needs random UUIDs or time-sortable ULIDs.

03 This MCP executes native crypto functions and returns the perfectly formed, unique ID(s) directly to your workflow.

The bottom line is that you get back perfectly structured, guaranteed unique identifiers without any guesswork from the AI layer.

Built For

Data Engineers and Backend Developers need this when their applications depend on reliable data schema. If your work involves creating new records, tracking transactions, or building high-performance databases, you're running into ID generation pain points.

Backend Developer

Needs to implement robust API endpoints that generate unique keys for every resource created in a microservice architecture.

Data Engineer

Requires reliable, chronologically sorted identifiers (ULIDs) to optimize large-scale database indexing and ETL pipelines.

Database Administrator

Uses this to ensure the integrity of primary key generation across multiple tables and systems without manual intervention or potential data collisions.

What Changes When You Connect

01 Avoid data corruption. By using this MCP, you eliminate the risk of your agent fabricating an ID that fails database validation or causes a collision.

02 Optimize indexing with ULIDs. Instead of standard UUIDs, use `generate_ulid` to create identifiers that keep your database records sorted by time, speeding up lookups.

03 Build robust APIs. When you call `generate_uuid`, you get guaranteed v4 IDs using native crypto libraries, which is essential for secure data handling and API key generation.

04 Simplify complex workflows. Your agent doesn't need to remember UUID formatting rules; it just calls the tool and receives a perfectly formatted string every time.

05 Maintain schema integrity. This MCP handles both random keys and time-series keys, giving you two critical tools for maintaining clean database schemas.

Real-World Applications

Tracking high-volume user signups

A startup needs to process 10,000 new user accounts in a batch. Instead of asking the agent to generate IDs (which risks collisions), they use `generate_uuid` repeatedly. The system gets mathematically guaranteed unique keys for every single record.

Building an audit log with time sensitivity

A compliance system needs to track event history where the order matters critically. They use `generate_ulid`. This ensures that even if two events happen milliseconds apart, their resulting IDs will sort correctly in the database, making auditing simple.

Generating unique API keys for clients

A platform needs to issue a set of distinct, non-guessable access tokens. They call `generate_uuid` multiple times within their agent workflow. The resulting list of UUIDs is guaranteed random and compliant with industry standards.

Designing a new relational database table

The schema requires an ID that grows logically over time for efficient indexing. They use `generate_ulid` to populate the primary key field, ensuring that inserting data never degrades query performance due to random keys.

Patterns to Avoid

Asking LLMs for ID batches

✗ AVOID

Prompting an agent: 'Generate 50 unique UUIDs.' The model might return invalid formats, repeat IDs, or fail to adhere to RFC standards.

✓ INSTEAD

Always use the `generate_uuid` tool. It calls native crypto libraries, ensuring every single ID is mathematically perfect and collision-free.

Using UUIDs for time tracking

✗ AVOID

Relying on standard UUIDs to order events in a database. Because they are random, chronological sorting becomes unreliable or requires complex secondary indexing.

✓ INSTEAD

Use the `generate_ulid` tool instead. The ULID is specifically designed to be lexicographically sortable by time, optimizing your indexes.

Manually formatting IDs

✗ AVOID

Writing code that tries to combine timestamps and random strings to create a unique ID. This process is brittle and prone to human error or overlap.

✓ INSTEAD

Let the MCP handle it. Use `generate_ulid` for time-series data, or use `generate_uuid` when pure randomness is the only requirement.

The Right Fit

Use this MCP if your application's core function relies on the absolute integrity and uniqueness of identifiers. If you need to guarantee that IDs won't collide, regardless of how many records are created, this is a must-have. Specifically, use `generate_ulid` whenever time-based sorting (like audit logs or event streams) is critical for performance. Use `generate_uuid` when the ID needs maximum randomness and no temporal relationship matters. Don't use this if you just need placeholder data for local testing; those tools are simpler. But never, ever skip using these specialized generators in production code—trusting the LLM to generate IDs is a single point of failure.

Every developer hates generating unique keys

Think about building a new microservice. You need to create an ID for every record, and that ID has to be globally unique. Today, developers often resort to simple random string generation or let the AI client attempt it. This process is messy; you end up with IDs that either fail database validation, repeat values in high-volume inserts, or are simply impossible to sort chronologically.

With this MCP, generating an ID becomes a single, reliable step. You don't worry about entropy or format compliance. Your agent calls the tool and gets back a perfect key—whether it’s random enough for security, or ordered enough for fast database queries.

Generate Unique IDs with generate_ulid

The biggest time sink is handling data that needs to be sorted by insertion date. If you use standard UUIDs, your index scatters across the database. You spend hours optimizing indexes and writing complex query logic just because the primary key isn't naturally ordered.

Now, simply call `generate_ulid`. The resulting identifier guarantees chronological sorting for your records. It’s a clean fix that makes high-performance indexing routine, not an engineering headache.

UUID & ULID Generator: 2 Tools Available

Access two specialized tools that generate mathematically perfect, industry-standard unique identifiers for your database and application logic.

#	TOOL	DESCRIPTION
01	`generate_ulid`	Creates a time-ordered, sortable ID ideal for database keys that must maintain chronological order.
02	`generate_uuid`	Generates a mathematically perfect v4 UUID using true crypto randomness for maximum collision resistance.

See It in Action

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

U Generate a completely random, collision-proof v4 UUID for a new database record.

✅ **Secure Generation:** The crypto engine returned `f47ac10b-58cc-4372-a567-0e02b2c3d479`.

U I need a time-sortable ULID for high-performance PostgreSQL indexing.

✅ **ULID Created:** `01ARZ3NDEKTSV4RRFFQ69G5FAV` (Lexicographically sortable based on the current millisecond).

U Please generate an array of 5 unique UUIDs to be used as API Keys.

✅ **Keys Generated:** Successfully bridged V8 crypto randomness to create 5 distinct UUID values.

Frequently Asked Questions

01 **Does the UUID & ULID Generator MCP provide standard IDs?**

Yes, it provides two standards: v4 UUIDs using `generate_uuid` for random keys, and time-sortable ULIDs using `generate_ulid` for ordered keys. You pick the right one for your use case.

02 **Why should I use generate_ulid instead of a regular ID?**

You use it when chronological sorting is critical to database performance, especially in audit logs or event tracking systems. The ULID ensures the keys will always sort by time.

03 Is the UUID generated by this MCP truly random?

Yes. It uses native crypto libraries via `generate_uuid`, meaning it's mathematically guaranteed to be collision-free and adhere to RFC 4122 standards.

04 Can I generate a batch of IDs at once?

Yes, your agent can request arrays of identifiers. This is crucial for bulk operations like processing multiple API keys or simulating large datasets.

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 `"uuid-ulid-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

UUID & ULID 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 UUID & ULID 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	UUID & ULID Generator MCP
Server ID	`019e3906-0c08-7226-ab68-ff0233b05741`
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/uuid-ulid-generator.