MCP SERVER | NO CODE | CLOUD HOSTED

Advanced Timezone Engine MCP for AI Agents

Handle precise global time conversions and historical DST rules.

Advanced Timezone Engine handles complex time conversions and historical Daylight Saving Time (DST) rules using the IANA database. It lets your AI client calculate precise offsets, detect ambiguous times during transitions, and format ISO 8601 strings accurately for any global location between 1970 and 2030.

A+ Quality Score 100/100

timezone | dst | iana | iso8601 | temporal

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JULY 2026
vinkius.com/mcp/advanced-timezone-engine

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.

Advanced Timezone Engine MCP

5 tools available
Cloud-hosted on Vinkius

The Advanced Timezone Engine handles complex time conversions and historical Daylight Saving Time (DST) rules using the IANA database. When you're building apps that need to schedule meetings across continents or log events in different regions, a simple "plus 5 hours" calculation usually breaks. This MCP gives your AI agent a reliable way to handle the messy reality of global time. It knows exactly when a country switched to Daylight Saving Time in the 90s and which cities are currently sharing the same local time. You can use it to check if a specific time even exists, which is a lifesaver when dealing with those "missing" hours in the spring or doubled hours in the fall. This is a solid piece of infrastructure for anyone who needs their AI to stop hallucinating dates and start providing accurate temporal data. It covers the years 1970 through 2030, making it perfect for both historical audits and future planning. You'll find it's a lot easier to manage these complexities when you have a dedicated tool like this in your Vinkius catalog. Instead of your agent guessing or failing on edge cases, it can now provide precise, localized information for any IANA timezone. This removes the need for manual offset tables or complex logic in your own code.

Core Capabilities

01 — Convert global timestamps

Translate a time from one IANA timezone to another while automatically applying DST rules.

02 — Identify simultaneous timezones

Find every timezone on earth that currently displays a specific local time.

03 — Retrieve historical offsets

Get the exact UTC offset for any specific location at any point in history or the future.

04 — Validate DST transitions

Check if a specific time is valid or ambiguous during spring forward and fall back shifts.

05 — Standardize date formats

Generate precise, timezone-aware ISO 8601 strings for any global location.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/advanced-timezone-engine** — connect your AI agent in three steps.

01 Connect the MCP to your AI client through the Vinkius dashboard.

02 Provide your agent with a timestamp and the IANA timezone name you want to target.

03 Get back a precise, formatted time string that accounts for all local rules.

The bottom line is that it replaces manual offset math with accurate database lookups.

Built For

This is for the backend engineer who's tired of edge cases breaking their scheduling logic or the travel coordinator managing multi-city itineraries. It solves the headache of "what time is it there?" when the answer depends on 50 years of local laws.

Backend Engineer

Handles complex scheduling logic for global users without hardcoding offsets.

Data Analyst

Cleans up historical logs from different regions into a single UTC timeline.

Product Manager

Verifies that a global product launch happens simultaneously across different markets.

What Changes When You Connect

01 No more manual math: Use convert_time to move between timezones without worrying about whether DST is active for a specific region.

02 Prevent scheduling errors: Use check_datetime_validity to catch ghost hours during spring forward transitions that often break calendar invites.

03 Accurate historical data: Pull correct offsets from decades ago using get_historical_offset for better integrity in your data logs.

04 Global synchronization: Find matching locations instantly with find_active_zones to coordinate multi-region events across the globe.

05 Standardized output: Get clean, ready-to-use strings from format_iso8601 that won't break your frontend or database systems.

Real-World Applications

Scheduling a global webinar

A user asks to find a time that works for London and Tokyo. The agent uses find_active_zones and convert_time to find a slot that works for both.

Auditing old logs

A developer needs to know the UTC offset for a server in 1995. The agent uses get_historical_offset to verify the data was recorded correctly.

Booking a flight

A traveler wants to know if a 2 AM flight is valid in a specific city. The agent uses check_datetime_validity to confirm the time actually exists.

Database migration

A team is moving local timestamps to UTC. The agent uses format_iso8601 to standardize the input before running a bulk conversion.

Patterns to Avoid

Hardcoding offsets

✗ AVOID

Just add 5 hours for New York.

✓ INSTEAD

Use convert_time to handle the actual IANA rules, which change frequently and vary by date.

Ignoring DST shifts

✗ AVOID

Assuming a city's offset is constant year-round.

✓ INSTEAD

Use get_historical_offset to check the specific offset for the exact date you are targeting.

Double-booking during fall back

✗ AVOID

Scheduling a meeting during the repeated hour in November.

✓ INSTEAD

Use check_datetime_validity to see if the time is ambiguous or occurs twice during a transition.

The Right Fit

Use this if you need high-fidelity time logic for a global audience. It's the right choice if your AI needs to handle Spring Forward or Fall Back logic without making mistakes. Use it if you're dealing with historical data or need to know which cities share a local time. Don't use it if you only need to convert between two fixed offsets that never change. If you just need a simple current time string for a single location, a basic system clock might suffice, but this MCP is for anyone who needs to be right every single time across different regions.

Advanced Timezone Engine for Accurate Global Scheduling

Most developers try to handle timezones by manually adding hours. It works until a country changes its DST rules or you hit that weird hour in March when the clock jumps forward. You end up with broken calendar invites, missed meetings, and messy database entries that require constant manual fixes.

This MCP takes that weight off your shoulders. It pulls directly from the IANA database to give your AI agent the correct rules for any location. You get perfectly synchronized schedules and accurate timestamps without ever having to write a single line of offset logic.

Advanced Timezone Engine for Historical Data Integrity

Cleaning up old data is a nightmare when different regions have different historical offsets. You have to hunt for rules, check old records, and try to guess which way the clock moved in a specific year.

With this tool, your agent can look up the exact offset for any point in time. You turn a weeks-long data cleanup project into a few simple queries, ensuring your records are actually correct.

Advanced Timezone Engine: 5 Tools for Global Time Sync

Use these tools to convert timestamps, check DST validity, and find global timezones with precision.

#	TOOL	DESCRIPTION
01	`check_datetime_validity`	See if a specific time exists in a timezone or if it's a skipped or repeated hour. This prevents your agent from booking meetings during times that don't actually occur.
02	`find_active_zones`	Get a list of every timezone on earth that matches a specific local time right now. This is useful for finding all locations where a specific event will happen simultaneously.
03	`format_iso8601`	Turn a messy date into a clean, standard ISO 8601 string for any global location. This ensures your data remains consistent across different databases and systems.
04	`get_historical_offset`	Look up the exact UTC offset for a specific place at any point in the past or future. You can use this to verify old records or plan for future legislative changes to time.
05	`convert_time`	Translate a timestamp from one IANA timezone to another while handling all DST rules. This takes the guesswork out of international scheduling and time conversion.

See It in Action

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

U What's the difference between 3 PM in New York and 3 PM in Berlin right now?

Here is the current time comparison for both cities:

CITY	LOCAL TIME	UTC OFFSET
New York	3:00 PM	UTC -4
Berlin	9:00 PM	UTC +2

Berlin is currently 5 hours ahead of New York.

U Is 1:30 AM on March 10, 2024, a real time in New York?

No, that time is **invalid**.

In New York, the clocks jumped forward from 2:00 AM to 3:00 AM on that date. This means the 1:30 AM hour never actually occurred.

U Convert a timestamp of 2020-05-05 12:00:00 from Tokyo to London.

The converted time is **2020-05-05 04:00:00** in London.

This accounts for the Daylight Saving Time rules active in both Japan and the United Kingdom on that date.

Frequently Asked Questions

01 **Can the Advanced Timezone Engine handle past dates?**

Yes, it can retrieve precise UTC offsets for any timezone at any point in history between 1970 and 2030.

02 **How does the Advanced Timezone Engine handle Daylight Saving Time?**

It automatically applies the correct historical and future DST rules for every IANA timezone, so you don't have to calculate them manually.

03 **Can I use the Advanced Timezone Engine to find cities in the same time?**

Yes, you can ask it to find all active timezones globally that share a specific local time.

04 **Will the Advanced Timezone Engine work for every country?**

It supports the IANA Timezone Database, which covers the vast majority of global regions and cities.

05 **How does the Advanced Timezone Engine format dates?**

It can generate standardized ISO 8601 strings, ensuring your dates are formatted correctly for any global location.

06 **Is the Advanced Timezone Engine good for historical data?**

It is ideal for historical data because it looks up the specific offset that was active at any given point in the past.

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 `"advanced-timezone-engine": { "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

Advanced Timezone Engine 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 Advanced Timezone Engine. 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	July 2026
MCP Server	Advanced Timezone Engine MCP
Server ID	`019f266a-a29f-715e-83f9-ae34271afaec`
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/advanced-timezone-engine.