MCP SERVER | NO CODE | CLOUD HOSTED

Polyrhythm Calculator MCP

Pinpoint perfect timing across complex beats.

Polyrhythm Calculator instantly analyzes complex rhythmic patterns, determining precise alignment points and attack timestamps for multi-layered music composition. Use this MCP to calculate exactly when different rhythms coincide, finding the perfect sync point or generating millisecond timelines for every note in a sequence.

A+ Quality Score 100/100

rhythm | polyrhythm | music-theory | tempo | bpm | timing

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

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.

Polyrhythm Calculator MCP

4 tools available
Cloud-hosted on Vinkius

Designing music with multiple independent rhythms is tough. You need mathematical precision that goes way beyond simply tapping out a beat. This MCP lets you treat rhythm like a measurable data set. Instead of guessing where two different patterns will fall into sync, it calculates the exact moment they align. You can analyze multi-layered beats—whether simple duple rhythms or highly complex polyphonic structures—to get the detailed temporal information needed for professional composition and sound design. When your agent runs this MCP through Vinkius, you get all that timing data pulled together, letting you focus solely on the art instead of the math.

Core Capabilities

01 — Find rhythmic synchronization points

It identifies the exact pulse where multiple independent rhythms or layers hit their alignment point.

02 — Calculate layer attack timestamps

The MCP generates precise millisecond timelines for every single note across all defined rhythm layers.

03 — Determine grid subdivision resolution

It calculates the minimum required time unit (grid resolution) necessary to accurately map a complex rhythm pattern.

04 — Assess overall rhythm complexity

The MCP gives you a score or measure of how intricate and varied a given rhythmic pattern is, helping guide composition choices.

One Click on Vinkius — From Prompt to Execution

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

01 You provide the system with the core parameters: the rhythm layers (e.g., 3 beats against 4 beats), the tempo, and the duration.

02 The MCP processes this data through its internal algorithms to calculate all necessary temporal metrics, like sync points or attack lists.

03 Your agent returns a clean, structured output detailing the precise timing—down to the millisecond—for every element in the polyrhythm.

The bottom line is that it takes abstract musical concepts and turns them into concrete, measurable data points you can use immediately.

Built For

This MCP is for composers, music theorists, and sound designers who need perfect timing. If your current workflow involves manually adjusting BPM or cross-referencing spreadsheets to find rhythmic conflicts, this tool saves you hours of frustration.

Audio Engineer / Sound Designer

They use it when designing complex soundscapes that require multiple elements (like machinery noises and vocal tracks) to hit perfect, calculated sync points.

Composer / Music Producer

They rely on it to map out polyrhythmic structures before writing a single note, ensuring every layer interacts rhythmically without conflict.

Music Theorist / Academia

They use it to model and analyze complex rhythmic relationships for research or educational material, needing precise timing data over general concepts.

What Changes When You Connect

01 Eliminate manual timing errors. Instead of guessing where rhythms clash, use `find_sync_point` to get the exact pulse where every layer aligns.

02 Map out complete rhythmic sequences instantly. Use `get_layer_attacks` to generate millisecond timestamps for dozens of notes across multiple tracks in one go.

03 Understand your composition's density. Run `get_rhythm_complexity` to objectively measure how intricate a pattern is, helping you balance tension and release.

04 Set the perfect foundational timing unit. Use `get_grid_resolution` to find the absolute minimum time step needed for accurate digital representation of any rhythm.

05 Work with confidence knowing your data is mathematically sound. This MCP removes the guesswork from high-level musical arrangement.

Real-World Applications

Syncing three conflicting instrument loops

A sound designer needs to make a kick drum loop, a hi-hat pattern, and an arpeggiator hit at the same point repeatedly. Instead of trying to manually adjust BPM until they line up, they use `find_sync_point` to get the exact pulse where all three layers reset.

Modeling complex ethnic drumming patterns

A composer is researching West African polyrhythms involving five different repeating beats. They run `get_layer_attacks` on each of the five tracks to get a comprehensive, ordered list of attack timestamps for accurate sequencing.

Analyzing compositional difficulty

A music student is learning about polyrhythms and needs to grade their own composition's mathematical challenge. They use `get_rhythm_complexity` to get an objective metric of the pattern's density, which they can then cross-reference with the grid resolution.

Creating a rhythm from scratch at high tempo

A producer needs a repeatable sequence for 180 BPM. They use `get_grid_resolution` first to ensure their timing units are small enough, then they feed that resolution into other tools to build out the full attack list.

Patterns to Avoid

Treating rhythm like a simple timeline

✗ AVOID

Trying to manually calculate overlap by just dividing total beats by number of layers, which ignores tempo and subdivision errors.

✓ INSTEAD

Don't estimate. Use `find_sync_point` to let the MCP handle the complex mathematics for you. It finds the true alignment point regardless of how many conflicting rhythms are involved.

Using generic timing tools

✗ AVOID

Relying on basic DAW functions that only calculate beats per minute without accounting for layer-specific attack timings.

✓ INSTEAD

Use `get_layer_attacks`. This tool specifically calculates the millisecond timestamp for *each* note in a sequence, giving you granular control over timing.

Ignoring necessary precision

✗ AVOID

Assuming that 1/16th notes are always accurate enough, even when dealing with rapid tempo changes or asymmetrical meters.

✓ INSTEAD

Check `get_grid_resolution` first. This tool tells you the smallest subdivision unit required to keep your polyrhythm mathematically pure and precise.

The Right Fit

Use this MCP if your core problem is temporal alignment: determining when multiple, independent rhythmic patterns will coincide or what their precise timing should be. You need a mathematical answer, not an artistic suggestion. For example, if you have two repeating loops—one at 3 beats and one at 4 beats—you use `find_sync_point` to know exactly where they meet. Don't use this MCP if your goal is simply to generate random beats or create simple, single-layer grooves; for that, a basic sequencer tool will work fine. Only turn to the Polyrhythm Calculator when you need forensic timing data: knowing the exact millisecond attack timestamps (`get_layer_attacks`) or establishing the absolute minimum grid size (`get_grid_resolution`).

Timing multiple rhythms usually means spreadsheets and headache.

When you're working on a complex track, you often have to manually cross-reference different loops—say, a drum machine pattern against an arpeggiator—to ensure they hit the same point every time. This ends up in a messy spreadsheet where you're tracking beats and calculating overlaps by hand, spending hours just confirming that your timing is mathematically clean.

With this MCP, you drop the parameters into your agent. It immediately handles all the math. Instead of weeks spent fixing tiny synchronization errors, you get an instant, precise answer detailing exactly when every layer aligns.

Polyrhythm Calculator: Perfect timing data in one place.

The biggest time sink disappears: the manual verification of sync points and attacks. You no longer need to calculate or cross-check complex

Now, when you're building a piece, your focus shifts entirely to musicality. The MCP handles the math so you can do the art.

overlaps using multiple methods; you just ask for the alignment point.

Polyrhythm Calculator with 4 Tools

Use these specialized tools to analyze, calculate, and generate precise temporal data for complex rhythmic structures.

#	TOOL	DESCRIPTION
01	`get_grid_resolution`	Calculates the smallest possible time unit needed to accurately represent a given rhythm pattern.
02	`get_layer_attacks`	Generates detailed attack timestamps for every individual layer in a polyrhythm.
03	`find_sync_point`	Determines the exact pulse or point in time where all defined rhythmic layers align completely.
04	`get_rhythm_complexity`	Provides a metric that assesses how intricate and varied the overall rhythm pattern is.

See It in Action

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

U At what pulse does a 3:4 polyrhythm reset?

A 3:4 polyrhythm reaches its alignment point at pulse 12, which marks the completion of one full rhythmic cycle.

U Give me the attack timestamps for a 2:3 rhythm at 120 BPM for 1 measure.

At 120 BPM, the attacks for layer 0 occur at 0.0ms and 500.0ms, while layer 1 attacks occur at 0.0ms, 400.0ms, and 800.0ms.

U What is the grid resolution for a 5:4 rhythm at 100 BPM?

The minimum subdivision unit is 20.0ms, with 3 ticks per beat.

Frequently Asked Questions

01 **How does Polyrhythm Calculator find the sync point?**

It calculates the least common multiple (LCM) of your given rhythms to determine the precise pulse where all layers will align. It outputs this alignment point so you can build on it.

02 **Do I need to specify BPM when using get_layer_attacks?**

Yes, providing the tempo is critical because attack timestamps are measured in milliseconds relative to a defined beat rate. The MCP requires this for accurate calculation.

03 **What if my rhythms aren't simple integers (e.g., 5:3)?**

The tool handles complex ratios. You input the ratio, and it calculates the necessary temporal data based on that fractional relationship, giving you a clean sync point.

04 **Can get_grid_resolution help with tempo changes?**

Yes. By determining the minimum subdivision unit, it ensures your timing grid is fine enough to maintain mathematical integrity even when changing tempos or meters drastically.

05 **Is Polyrhythm Calculator just for drums?**

No, it's designed for any time-based sequence. You can use it to model anything from machine cycles to vocal phrase timing, provided the input is rhythmic.

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 `"polyrhythm-calculator": { "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

Polyrhythm Calculator 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 Polyrhythm Calculator. 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	Polyrhythm Calculator MCP
Server ID	`019eff93-02a3-7050-8fdc-e754e8a6b8d1`
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/polyrhythm-calculator.