MCP SERVER | NO CODE | CLOUD HOSTED

Key Finder MCP

Instantly map every harmonic relationship between keys.

The Relative & Parallel Key Finder instantly maps out every structural relationship between musical keys. Give it any key, and it returns all its related partners—relative, parallel, homonymous, and those separated by a third-distance interval. It's an essential reference tool for composers needing instant analysis of Western music theory.

A+ Quality Score 100/100

music-theory | composition | key-signatures | modulation | musicians

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/relative-parallel-key-finder

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.

Relative & Parallel Key Finder MCP

3 tools available
Cloud-hosted on Vinkius

Writing music often means knowing the rules first. This MCP connects your AI agent to complex Western music theory logic. Need to know every possible key related to C Major, or figure out exactly how many flats a specific key signature needs? This utility handles all that structural mapping for you. It's perfect for composers and students who need fast, reliable answers about harmonic relationships. Instead of flipping through thick textbooks or cross-referencing multiple charts, your AI client pulls the data directly. Vinkius hosts this MCP as part of a massive catalog, so once you connect your preferred agent, you gain instant access to all these musical tools. You just ask for the relationship between keys, and it provides a complete map.

Core Capabilities

01 — Map key relationships

It determines relative, parallel, antiparallel, homonymous, and third-distance partners for any given key.

02 — Detail key signatures

It provides a technical breakdown of any key's signature, detailing accidental count, type (sharps/flats), and mode.

03 — Reference modulation guides

It pulls up a complete reference table listing all known key signatures for planning modulations.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/relative-parallel-key-finder** — connect your AI agent in three steps.

01 You ask your AI client to find the structural relationships or details of a specific musical key.

02 The MCP uses its internal logic to calculate and retrieve all necessary harmonic data based on standard music theory rules.

03 Your agent receives a clean, structured list containing relative keys, signature counts, or full modulation reference tables.

The bottom line is you get an immediate, expert-level analysis of complex key relationships without doing the math yourself.

Built For

This MCP is for anyone who works with musical structure. Composers need it when they're stuck on a bridge or verse transition. Students use it to pass theory exams. Music educators rely on it to give accurate, instant feedback on harmony.

Composer

They check the relationships between two sections of music to ensure their modulation point is harmonically sound.

Music Theory Student

They quickly verify complex key signature rules or identify homonymous keys for an assignment without manual reference checks.

Audio Engineer / Sound Designer

They need to confirm the tonal center and related keys when designing soundscapes that mimic musical structure.

What Changes When You Connect

01 Instantaneous Key Mapping: Using `find_key_relationships`, you stop guessing. You immediately see all the relative, parallel, and homonymous partners for any key, keeping your compositions harmonically tight.

02 Technical Clarity on Signatures: Never doubt a key's composition again. `get_signature_details` instantly tells you if a key has sharps or flats, how many, and what mode it uses.

03 Modulation Planning Support: Need to move from one section of music to another? The `lookup_modulation_guide` gives you a complete reference chart for reliable key changes.

04 Saves Research Time: Instead of cross-referencing multiple charts or flipping through textbooks, your agent pulls this complex theory data in seconds.

05 Structured Output: All results are delivered cleanly. You don't get raw data dumps; you get actionable lists and detailed reports for immediate use.

Real-World Applications

The Bridge Problem

A composer needs a key change for the bridge section but isn't sure which keys are harmonically related to the main verse. They ask their agent, and it uses `find_key_relationships` to provide several viable options (relative, parallel) that fit the existing harmonic palette.

The Theory Exam Prep

A music student is studying key signatures for an exam. Instead of memorizing charts, they use `get_signature_details` to check how many accidentals a specific key has and whether it's in major or minor mode.

The Arrangement Update

An arranger needs to map out the entire song structure for potential key changes. They use `lookup_modulation_guide` first to get an overview, then confirm specific transitions using the full scope of keys found by `find_key_relationships`.

Identifying Tonal Centers

A sound designer wants to know all the tonal centers that share the same underlying accidentals as a given key. They use `get_signature_details` to verify the exact accidental count, helping them build cohesive soundscapes.

Patterns to Avoid

Assuming Key Relationships

✗ AVOID

Manually guessing that a key is related just because it sounds similar. This leads to awkward modulations and harmonic inconsistency in the final piece.

✓ INSTEAD

Always let your agent run `find_key_relationships` first. It provides a complete, mathematically proven list of all possible structural partners, ensuring you don't miss any.

Ignoring Signature Differences

✗ AVOID

Writing music assuming two keys share the same number of sharps or flats without checking for mode (major vs. minor). This results in ambiguous harmonic instructions.

✓ INSTEAD

Use `get_signature_details` to confirm both the accidental count AND the specific mode (sharps/flats/none) before finalizing any key choice.

Stuck on Modulation

✗ AVOID

Spending hours cross-referencing old theory books just to find a reliable modulation path between two distant keys.

✓ INSTEAD

Start with `lookup_modulation_guide` to map out the entire territory. Then, use that reference table in conjunction with `find_key_relationships` for precise transitions.

The Right Fit

Use this MCP if your problem is purely structural: you need a complete list of related keys, or you need to verify the technical composition (accidental count/mode) of a key. If you are designing an entire song structure and need to map every possible transition point, start with `lookup_modulation_guide`. However, don't use this if your goal is chord voicings for a specific measure; that requires a different type of tool. Also, if you just want to know the difference between C Major and G Major, simple dictionary lookups are fine. But if you need all *structural* keys related to them—including antiparallels and third-distance partners—you must use `find_key_relationships`.

The Challenge of Key Mapping

Writing music requires more than just knowing which chords sound good together. You have to manage the theory behind it. Today, figuring out what keys are structurally related means flipping through dense textbooks or constantly cross-referencing complex harmonic charts. This process is slow and highly prone to human error; you can easily miss a relative key or misread the signature count for a difficult modulation.

With this MCP, that manual work disappears. Your agent performs the entire structural analysis instantly. You input one key, and it spits out an exhaustive list of every related partner—relative, parallel, homonymous, and more. It’s not just fast; it's comprehensive.

Mastering Key Relationships with find_key_relationships

Before, determining all the possible keys related to your main key required multiple steps: first checking for parallels, then relative minors, and finally referencing specific interval rules. This meant juggling three or four different data sources just to get a full picture of available tones.

Now, one command does it all. The `find_key_relationships` tool gives you the whole structural map in a single output. You see every possible key option immediately, allowing you to compose with complete confidence.

Relative & Parallel Key Finder (3 Tools)

Use these specific tools to analyze the complex structural rules of Western music, finding every relationship between keys with simple commands.

#	TOOL	DESCRIPTION
01	`find_key_relationships`	Finds all structural partners for a key, including relative, parallel, antiparallel, homonymous, and third-distance keys.
02	`get_signature_details`	Describes the technical composition of a given key by reporting its accidental count, type (sharps or flats), and mode.
03	`lookup_modulation_guide`	Accesses a complete reference table detailing all known musical key signatures for modulation planning.

See It in Action

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

U What is the relative and parallel key of G Major?

The relative key of G Major is E Minor, and its parallel key is G Minor.

U How many sharps are in the key of D Major?

D Major has 2 sharps (F# and C#).

U Show me all keys that share the same tonic as Eb Major.

The homonymous keys for Eb Major are Eb Major and Eb Minor.

Frequently Asked Questions

01 **How do I use find_key_relationships to find all related keys?**

You simply ask your agent for the relationships of a specific key. For example: 'What are the find_key_relationships for C Major?'. The tool returns every structurally linked partner in one list.

02 **Does get_signature_details only count sharps?**

No, it reports on both sharps and flats. You can ask it to check the signature details of any key, and it will give you the total accidental count, type (sharps/flats), and mode.

03 **What is lookup_modulation_guide used for?**

It provides a full reference chart showing all known keys. This guide is crucial when planning large-scale modulations across an entire piece of music.

04 **Can I use find_key_relationships to compare multiple keys?**

Yes, you can ask the agent to compare two or more keys using this tool. It will then list all the structural relationships that exist between them.

05 **Is this useful for simple chord theory?**

This MCP focuses on deep harmonic structure and key relationships, not simple chords. Use it when you need to know *why* certain keys are related, not just what chords they contain.

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 `"relative-parallel-key-finder": { "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

Relative & Parallel Key Finder 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 Relative & Parallel Key Finder. 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	Relative & Parallel Key Finder MCP
Server ID	`019f010f-ac86-736b-a943-cfe5bb0f5edd`
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/relative-parallel-key-finder.