MCP SERVER | NO CODE | CLOUD HOSTED

# Metronome MCP

## Automate your usage-based billing lifecycle.

Metronome lets you automate usage-based billing, contract management, and revenue operations through your AI agent. Send usage events, query complex data across customers, create contracts, and process invoices—all from natural conversation.

A+ Quality Score 98.33/100

usage-based-billing | revenue-operations | metering | contract-management | event-ingestion | saas-metrics

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/metronome

SCAN TO VIEW

# The infrastructure that powers AI agents in the real world.

Vinkius connects AI to the world's software through secure, enterprise-grade infrastructure — enabling real-world execution at scale, built on the Model Context Protocol (MCP).

# Your AI Connections Run Through Vinkius Cloud

The world's largest managed MCP catalog

Vinkius is the cloud infrastructure where AI agents connect 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.

# Metronome MCP

31 tools available
Cloud-hosted on Vinkius

You can manage entire revenue cycles without leaving your chat window. This MCP connects your billing backend to your workflow; instead of manually exporting sheets or navigating multiple financial dashboards, you simply talk to your agent. You tell it which customer needs a new rate card defined, or that usage events need recording. It handles the ingestion and data storage using tools like `ingest_events`. Need to know what's owed? Your agent fetches current balances or lists specific invoices. If an invoice is wrong, you don't call accounting; your AI client simply runs `void_invoice` and gets a record of the change. You can also manage the underlying structure—creating products, defining billable metrics (`create_billable_metric`), and setting up customer records. This functionality fits right into the Vinkius catalog, giving you one place to handle all your usage-based billing needs.

## Core Capabilities

**01 — Record Usage Events**

Send high volumes of usage data events into Metronome for tracking.

**02 — Manage Customer Accounts**

Create, retrieve details on, or archive entire customer records and their associated contracts.

**03 — Process Invoices and Balances**

Fetch current invoices, check net balances, or regenerate documentation when needed.

**04 — Define Billing Structures**

Set up product definitions, create rate cards, and define billable metrics used for charging customers.

**05 — Audit Activity Logs**

Pull detailed audit logs to review changes made across the billing system.

## One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to this MCP and enter your Metronome API Token.

02 Your AI client authenticates the connection, giving it direct access to all billing tools.

03 You prompt your agent with a request, like 'List all customers who haven't paid in 60 days,' and get actionable data back.

The bottom line is that you use natural language conversation to execute complex financial operations directly through your AI client.

## Built For

This MCP serves the Product Manager who needs real-time usage data for billing reports, or the Finance Analyst tired of manually reconciling invoices across different platforms. If your job involves tracking revenue based on consumption, this is built for you.

**Billing Operations Specialist**

They use the MCP to manage the full lifecycle of customer accounts, setting up contracts and voiding old invoices instantly.

**Product Manager**

They analyze feature usage by ingesting events via `ingest_events` to determine which products are most valuable for billing metrics.

**Financial Analyst**

They use the MCP to fetch detailed net balances and run reports on customer groups without exporting data into spreadsheets first.

## What Changes When You Connect

01 You instantly track consumption by sending events using `ingest_events`, ensuring all usage data is captured for accurate billing records.

02 Need to fix an invoice? Run `void_invoice` or use `regenerate_invoice` to manage payment status and correct documentation without manual intervention.

03 Define your pricing model entirely within the agent. Use tools like `create_product`, `create_rate_card`, and `add_rate` to build complex billing logic on demand.

04 Never manually reconcile data again. Your AI client can fetch current account statuses using `get_net_balance` or run detailed reports with `list_invoices` in seconds.

05 Manage the customer lifecycle from start to finish: you can use `create_customer` to onboard a new client and later run `archive_customer` when they leave.

06 Audit compliance is simple. Pull full records of all system changes by calling `get_audit_logs`, giving you an immediate trail for every billing action.

## Real-World Applications

#### Quarterly Billing Reconciliation

The finance analyst needs to confirm that the usage data matches the signed contract. They prompt their agent: 'Check the net balance for GlobalTech and list all contracts.' The MCP uses `get_net_balance` and `list_contracts`, providing a single, verifiable answer instead of requiring manual report pulls from multiple departments.

#### Correcting Overdue Invoices

A customer claims an invoice is wrong. Instead of emailing support, the ops specialist asks their agent to 'Check and fix invoice inv_556677.' The MCP uses `get_invoice` first to review the details, then runs `void_invoice` if necessary before generating a corrected version.

#### Onboarding a New Service Line

A product manager adds a new feature that needs billing. They ask their agent to 'Define this as a usage-based product and set the base price.' The MCP uses `create_product` and `create_rate_card`, instantly updating the system's financial capacity for the new service.

#### End of Client Relationship

When a client leaves, the billing specialist must clean up records. They tell their agent to 'Archive Acme Corp.' The MCP executes `archive_customer`, which automatically voids any pending invoices and archives all associated contracts.

## Patterns to Avoid

### Mixing Billing with General Accounting

✗ AVOID

Trying to use Metronome to reconcile bank statements or manage payroll records. This MCP focuses only on usage-based billing data.

✓ INSTEAD

If you need general ledger reconciliation, look for an accounting platform connector instead. Use this MCP strictly for managing revenue and consumption metrics.

### Listing Tools Manually

✗ AVOID

Writing out every single function name in a long list of code snippets without context.

✓ INSTEAD

Instead, describe the goal: 'I need to know if I can change pricing for an existing customer.' The MCP handles this via high-level actions like `edit_contract` or setting up new rates using `add_rate`.

### Ignoring Usage Limits

✗ AVOID

Running a large data query without first checking the available commits, risking rate limiting or incorrect billing calculations.

✓ INSTEAD

Always start by reviewing usage limits. Use `get_usage_groups` to see consumption patterns before attempting massive data ingestion via `ingest_events`.

## The Right Fit

Use this MCP if your primary pain point is managing the lifecycle of usage-based revenue, from event capture to final invoice generation. You need a system that tracks 'how much' and 'who owes what.' Don't use it if you just need simple CRM tasks like updating phone numbers; while `set_custom_field_values` handles data points, this MCP is deep in the financial ledger. If your goal is to manage general assets or payroll, look for a dedicated accounting connector instead of relying on billing tools.

## The daily struggle with usage reports and invoices

Right now, when you need to know what's owed, the process is painful. You open the main dashboard, find the customer list, then click over to the invoicing tab. If a rate changed last month, you have to manually cross-reference old contracts against current product definitions, all while copying data into an Excel sheet just for reporting.

With this MCP, that entire sequence collapses. Your agent knows where everything lives—the usage logs, the contract details, and the rate cards. You simply ask: 'What is GlobalTech's total outstanding balance?' The answer comes back instantly, backed by all the necessary data points.

## Metronome MCP delivers full billing control

You eliminate manual cross-referencing between usage logs and contract terms. You don't have to manually void invoices or regenerate

The difference is control. You move from being a data copier who spends hours piecing together

documentation because the system tracks every change, giving you a clean audit trail via `get_audit_logs`.

reports into an operator who asks a question and gets a definitive, actionable answer.

# Metronome MCP - 30 Tools for Billing Operations

Use these tools to manage everything from basic customer creation and complex event ingestion to voiding invoices and setting up new rate cards.

| # | TOOL | DESCRIPTION |
|---|---|---|
| 01 | add_custom_field_key | Creates a new field key for custom data tracking. |
| 02 | add_rate | Adds a specific pricing rate to an existing product's rate card. |
| 03 | archive_customer | Voids pending invoices and archives all contracts associated with a customer account. |
| 04 | create_alert | Sets up automated notifications when specific usage thresholds are crossed. |
| 05 | create_billable_metric | Defines and activates a new unit of measure that can be charged to customers. |
| 06 | create_commit | Establishes a commitment amount or usage limit for a customer's billing cycle. |
| 07 | create_contract | Generates and formalizes a new service agreement between the company and a client. |
| 08 | create_customer | Registers a brand-new customer into the billing system. |
| 09 | create_notification | Sets up an automated lifecycle event notification for specific triggers. |
| 10 | create_product | Defines a new product, specifying if it's usage-based, fixed fee, or subscription. |
| 11 | create_rate_card | Establishes the base pricing structure for multiple products. |
| 12 | edit_contract | Makes changes to an existing customer service contract, like adjusting terms or dates. |
| 13 | get_audit_logs | Retrieves a comprehensive list of actions taken within the billing system for review. |
| 14 | get_customer | Fetches all current details about a specific customer by their ID. |
| 15 | get_invoice | Retrieves the full record and status for one specific invoice number. |
| 16 | get_net_balance | Checks the current total amount due or credited against a customer's account. |

| # | TOOL | DESCRIPTION |
|---|---|---|
| 17 | `get_services` | Retrieves a list of all services offered by the platform. |
| 18 | `get_usage_groups` | Gathers paginated data summaries for usage metrics across groups. |
| 19 | `get_usage` | Retrieves batched, detailed records of consumed usage units. |
| 20 | `ingest_events` | Accepts and processes large batches of raw usage events into the billing system. |
| 21 | `list_balances` | Provides a summary list of current account balances across multiple customers or groups. |
| 22 | `list_billable_metrics` | Shows all existing units of measure that the company charges for (e.g., per API call). |
| 23 | `list_contracts` | Retrieves a list and status summary of all active customer contracts. |
| 24 | `list_credit_types` | Shows the different types of pricing units available for billing. |
| 25 | `list_customers` | Retrieves a searchable list of all registered customer accounts, supporting filters. |
| 26 | `list_invoices` | Generates a summary report listing multiple invoices and their payment status. |
| 27 | `list_products` | Lists all defined products, helping you see what services can be billed. |
| 28 | `regenerate_invoice` | Re-generates a specific invoice to correct minor data errors or updates. |
| 29 | `search_events` | Searches for usage events using a specific transaction ID, useful for testing. |
| 30 | `set_custom_field_values` | Adds or updates custom data values to an existing customer record. |
| 31 | `void_invoice` | Marks a specific invoice as void, preventing it from being paid and removing it from active billing. |

# See It in Action

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

**U** List all customers currently registered in Metronome.

I've retrieved the customer list. You have 15 active customers, including 'Acme Corp' (ID: cust_821) and 'GlobalTech' (ID: cust_442). Would you like more details on a specific one?

**U** Show me the usage data for customer cust_821 for the last 30 days.

Fetching usage data... For 'Acme Corp', I see 1.2M events ingested across 3 billable metrics. The primary driver is 'API Requests' with 950k units. Would you like to see the breakdown by day?

**U** Retrieve the details and status for invoice inv_556677.

Invoice inv_556677 for 'GlobalTech' is currently 'Finalized' with a total amount of $4,250.00. It covers the period from Oct 1 to Oct 31. Would you like me to list the line items?

# Frequently Asked Questions

01 **Can Metronome MCP help me track usage events?**

Yes. You can use `ingest_events` to send high volumes of raw usage data into the billing system for immediate tracking and calculation.

02 **How do I check if an invoice is correct using Metronome MCP?**

You first run `get_invoice` to pull all details. If it's wrong, you can use `void_invoice` to cancel it before running `regenerate_invoice` for the corrected version.

03 **What if I need to change a customer's pricing?**

You first define or update the product using `create_product`, then use `add_rate` to set new prices, and finally apply those terms by running `edit_contract`.

04 **Does Metronome MCP let me list all customers?**

Yes. You can run `list_customers` to get a filtered or comprehensive view of every customer account in the system, helping you manage them all at once.

# 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 `"metronome": { "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

# Metronome 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 Metronome. 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 | Metronome MCP |
| Server ID | `019e38c1-180e-7026-9fff-7b58aef64b1d` |
| 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/metronome.