MCP SERVER | NO CODE | CLOUD HOSTED

Stigg MCP

Manage billing, usage, and subscriptions via conversation.

Stigg brings billing and subscription management directly into your AI workflow. Use this MCP to provision new customers, manage active plans, or report usage metrics without leaving your client. You can handle the entire customer lifecycle—from initial account creation using rest_create_customer to tracking metered feature usage via gql_report_usage—all through natural conversation.

A+ Quality Score 100/100

billing | subscriptions | saas-pricing | usage-based-billing | customer-management

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

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.

Stigg MCP

12 tools available
Cloud-hosted on Vinkius

Stop switching between billing dashboards just to check a customer's status or adjust their plan. This connector lets you take full control of your entire pricing and packaging workflow, right from any MCP-compatible client. You can handle the whole customer journey, whether it's creating a new account, updating details, provisioning a subscription, or canceling service when needed. Need to know how much usage a user has consumed? Just ask your agent to report the metered feature usage in real time for accurate billing. Because this MCP is hosted on Vinkius, you get access to these core operations—like customer profile retrieval and plan management—all from one place. Your AI acts like an instant, hands-on billing ops assistant that never gets tired of API calls.

Core Capabilities

01 — Manage Customer Profiles

Create new accounts or fetch existing customer details using both REST and GraphQL endpoints.

02 — Provision Subscriptions

Set up a new subscription for a customer or retrieve the current state of their active plans.

03 — Control Subscriptions

Cancel an existing subscription or update basic customer information when necessary.

04 — Track Usage Metrics

Report metered feature usage to ensure billing accuracy and enforce entitlements in real time.

One Click on Vinkius — From Prompt to Execution

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

01 First, subscribe to this MCP on Vinkius and input your Stigg API Key.

02 Next, tell your AI client exactly what you need—for example, 'Get the details for customer X' or 'Report usage of feature Y'.

03 The agent uses the appropriate tool (like gql_get_customer or rest_report_usage) to perform the action and gives you a direct answer.

The bottom line is that your AI client turns complex billing API calls into simple, conversational instructions.

Built For

Product Managers who need instant status checks; Customer Success reps needing to provision trials mid-call; and Developers who want to test complex billing flows right from their IDE without leaving the terminal.

Customer Success Specialist

Uses this MCP to quickly provision trial subscriptions or update customer details during a support call, eliminating manual dashboard navigation.

Product Manager

Checks current feature entitlements and plan statuses for various users instantly, verifying if a feature is accessible before writing product documentation.

Software Developer

Tests billing logic by reporting usage metrics or creating test customers directly from the terminal while building new features.

What Changes When You Connect

01 Avoid dashboard hopping: Instead of jumping between the Stigg console to check a customer's plan status or feature entitlements, you just ask your agent. It gives you the data instantly.

02 Process complex lifecycle changes quickly: Need to create an account and immediately set up their first paid subscription? Use gql_provision_customer or rest_create_subscription in one go with simple instructions.

03 Maintain billing accuracy: Don't rely on manual reporting for usage. Tell your agent to report the metered feature usage (like using gql_report_usage) so billing is always accurate and up-to-date.

04 Flexible integration options: Whether you prefer calling out updates via rest_update_customer or fetching data through GraphQL, this MCP handles both REST and GraphQL actions for maximum flexibility.

05 Centralized control: You can manage the entire customer journey—from initial provisioning to final cancellation using rest_cancel_subscription—without ever leaving your primary development environment or chat tool.

Real-World Applications

Onboarding a new enterprise client

A Customer Success rep asks their agent: 'Create customer ID cust-456 and provision them the Enterprise plan.' The agent executes gql_provision_customer, ensuring both the account and the subscription are live immediately.

Verifying usage limits during a sprint

A Developer runs into an apparent billing issue. They prompt their agent to 'Report 150 units of usage for feature X.' The agent uses rest_report_usage, providing immediate confirmation and allowing the dev to confirm if the limit was hit.

Auditing a customer's plan status

A Product Manager needs to know what features 'cust-123' has access to. They ask their agent to use gql_get_entitlements_state, getting instant confirmation without opening the billing console.

Handling service termination

Support receives a cancellation request. The rep directs the agent to 'Cancel subscription for cust-123.' The agent uses rest_cancel_subscription and confirms the status change, logging the action immediately.

Patterns to Avoid

Using multiple tools sequentially

✗ AVOID

The user manually runs 'rest_get_customer' to get an ID, then copies that ID and pastes it into a separate prompt for 'gql_get_entitlements_state', wasting time.

✓ INSTEAD

Ask your agent in one go: 'Get the customer details AND their entitlements state for cust-123.' The agent handles both calls automatically.

Forgetting to specify the API style

✗ AVOID

The user is unsure if they should use REST or GraphQL, leading to multiple failed attempts and confusion about which data source is correct.

✓ INSTEAD

Just tell your agent: 'I need the customer details.' The agent will know how to select the right function (like gql_get_customer or rest_get_customer) based on context.

Attempting provisioning without an ID

✗ AVOID

The user tries to run 'gql_provision_subscription' without first giving the agent a customer ID, causing the process to fail immediately.

✓ INSTEAD

Always start by ensuring the account exists. First, use rest_create_customer or gql_get_customer. Then follow up with 'Now provision a subscription for them.' This ensures data integrity.

The Right Fit

Use this MCP if your primary pain point involves coordinating multiple billing actions: creating accounts, changing plans, tracking usage, and retrieving customer status. It's the central hub for SaaS finance operations.

Don't use it if you only need to read simple data that doesn't relate to billing (e.g., reading a user's name from an internal CRM). For those cases, look for a dedicated integration tool. Also, don't use this MCP just because you have trouble with one API type; since it supports both REST and GraphQL actions, it handles the technical choice for you.

The Pain of Switching Contexts to Check Billing Status

Right now, checking a customer's status means opening your billing dashboard. You pull up their profile, click into 'Subscriptions' to see the plan, and then maybe jump over to a usage tab just to check if they hit their limit. It takes five clicks, three tabs, and you risk losing track of which data point came from where.

With this MCP, those steps vanish. You simply tell your agent what you need—'What is the current plan and usage for cust-123?' The entire operation runs in the background, giving you a clean, direct answer without ever leaving your chat window.

Stigg: Instant Billing Control

Gone are the days of copying and pasting customer IDs between different systems. Gone is the manual process of confirming if a new subscription was created correctly or if usage reports were properly sent.

You manage everything—from rest_create_customer to gql_report_usage—through natural conversation. It makes billing operations feel like talking to an expert coworker, not wrestling with a dozen dashboards.

Stigg: 12 Tools for Billing Operations

These tools allow you to execute every core function of customer management, subscription control, and usage reporting via the Stigg API.

#	TOOL	DESCRIPTION
01	`rest_cancel_subscription`	Cancels a customer's subscription account using the REST API.
02	`rest_create_customer`	Creates an entirely new customer profile using the REST API.
03	`rest_create_subscription`	Sets up a brand new subscription for a customer via the REST API.
04	`rest_get_customer`	Retrieves all available details about an existing customer profile using the REST API.
05	`rest_get_subscription`	Fetches all specific information regarding a given subscription plan via the REST API.
06	`gql_get_customer`	Retrieves customer details using the flexible GraphQL query language.
07	`gql_get_entitlements_state`	Checks the current state of a user's feature entitlements via GraphQL.
08	`gql_provision_customer`	Creates and optionally provisions a subscription for a customer using GraphQL.
09	`gql_provision_subscription`	Sets up a new subscription plan directly via the GraphQL endpoint.
10	`gql_report_usage`	Reports metered feature usage data using the structured GraphQL query language.
11	`rest_report_usage`	Records metered usage for billing purposes through a dedicated REST API call.
12	`rest_update_customer`	Modifies existing details for a customer using the REST API.

See It in Action

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

U Create a customer with ID 'cust_123', name 'Alice', and email 'alice@example.com' using REST.

I've successfully created the customer 'Alice' (ID: cust_123) in Stigg. Would you like to provision a subscription for them now?

U Report 50 units of usage for feature 'api-calls' for customer 'cust_123'.

Usage reported. I've added 50 units to the 'api-calls' feature for customer 'cust_123' via the REST API.

U Get the details for customer 'cust_123' using GraphQL.

Retrieving data... Customer 'Alice' (cust_123) is currently active with email 'alice@example.com'. I can also list their active subscriptions if you'd like.

Frequently Asked Questions

01 **How do I create a customer using the Stigg MCP?**

You use rest_create_customer or gql_provision_customer. You just need to tell your agent the necessary details, like name and email, and specify which endpoint you want to use.

02 **Is reporting usage difficult with Stigg?**

No. You simply ask your agent to report the metered feature usage by using gql_report_usage or rest_report_usage, providing only the units and the feature name.

03 **Can I retrieve customer details in multiple ways with Stigg?**

Yes. You can use both GraphQL (gql_get_customer) for flexible querying, or REST (rest_get_customer) if you prefer direct resource retrieval via API calls.

04 **What happens when I try to cancel a subscription?**

The agent executes rest_cancel_subscription. It confirms the cancellation and updates the customer's status immediately, making it visible in your chat history.

05 **Does Stigg handle complex provisioning?**

It does. You can use gql_provision_customer to provision both a new customer account and an associated subscription plan simultaneously.

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 `"stigg": { "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

Stigg 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 Stigg. 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	Stigg MCP
Server ID	`019e38f4-4e9c-7083-9e6e-c575c09eba5d`
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/stigg.