MCP SERVER | NO CODE | CLOUD HOSTED

Google Play Developer MCP

Manage Reviews, Subscriptions, and Purchases in Chat.

Google Play Developer MCP connects your agent directly to Google's app store backend. You can read user reviews, check subscription statuses, issue refunds, and manage in-app purchases—all without opening the console. It handles everything from acknowledging test purchases to canceling recurring billing.

A+ Quality Score 100/100

app-store-optimization | review-management | subscription-billing | mobile-app-dev | in-app-purchases

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/google-play-developer

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.

Google Play Developer MCP

12 tools available
Cloud-hosted on Vinkius

Need to handle live customer issues for an Android app? This MCP connects your agent straight to Google Play Developer tools. Instead of logging into the massive Play Console dashboard every time a user complains, you just talk to your AI client. You can pull up recent user reviews and draft professional replies instantly. The system also tracks everything related to billing; check if a user's subscription is active, cancel it, or even defer the next charge date. If you need to manage purchases—whether acknowledging a test purchase or confirming an in-app product was consumed—this MCP handles it. By centralizing these actions through Vinkius, your agent gives you direct control over customer accounts and app monetization right from your chat interface.

Core Capabilities

01 — Monitoring user feedback

Automatically pull up recent reviews or check a specific review's details to understand public sentiment.

02 — Managing billing and subscriptions

Check the status of any subscription, cancel it for a user, defer the next billing date, or issue a full refund.

03 — Handling product purchases

Verify if an in-app purchase happened (checking its status) and then confirm that the transaction has been acknowledged or consumed by your backend.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/google-play-developer** — connect your AI agent in three steps.

01 Subscribe to this MCP and provide your Google Play App Package Name.

02 Complete the required OAuth flow using a developer account with appropriate permissions.

03 Start giving commands—your agent uses natural language to call specific tools like listing reviews or refunding subscriptions.

The bottom line is you get programmatic, real-time control over your app's revenue and customer interactions without ever opening the Google Play console web page.

Built For

This MCP is for App Developers who need to test purchases or verify billing states quickly. It's crucial for Customer Support agents who need to look up user purchase status and issue refunds without navigating complex dashboards.

Customer Support Agent

A customer calls about a failed payment. You use the MCP to immediately check the subscription purchase status and, if necessary, initiate a refund.

Community Manager

You spot negative reviews on Google Play. Using your agent, you pull up the review details and draft a professional reply to post directly through the MCP.

App Developer

During testing, you need to confirm that a test in-app purchase was correctly consumed by the backend or acknowledge a specific product purchase immediately.

What Changes When You Connect

01 Instantly manage customer billing. Instead of opening the console to check if a user's subscription is active or if they need a refund, you simply ask your agent, which uses `get_subscription_purchase` and `refund_subscription`.

02 Improve reputation management by automating responses. You can pull up recent feedback using `list_reviews`, identify key issues, and then craft and post replies directly via `reply_to_review`.

03 Streamline product development testing. Developers use tools like `get_product_purchase` and `acknowledge_product_purchase` to verify in-app purchases are handled correctly without manual console interaction.

04 Control monetization flow with precision. If a user needs more time, you can execute the `defer_subscription` tool; if they need immediate access, you can confirm the purchase using `consume_product_purchase`.

05 Get comprehensive product data at a glance. Need to know what items exist? Use `list_inapp_products` to generate an accurate catalog list and get details for any specific SKU with `get_inapp_product`.

Real-World Applications

Handling a high-value refund request

A customer calls, demanding a full credit. Instead of playing phone tag with support tickets, your agent first uses `get_subscription_purchase` to confirm the user's billing cycle status and then executes `refund_subscription` immediately, documenting everything in one chat thread.

Responding to bad press

You see a two-star review mentioning a bug. Your agent uses `list_reviews` to grab the context, drafts a polite and specific reply acknowledging the issue, and then posts it using `reply_to_review`, all without leaving your chat window.

Verifying test purchases

A developer needs to confirm if a payment made during testing was correctly logged. They ask their agent to check the purchase status, running `get_product_purchase` and then confirming it with `acknowledge_product_purchase` for immediate validation.

Stopping unwanted billing

A user calls to cancel their service. Instead of asking them to navigate deep into the Play Console, your agent uses `cancel_subscription`, completing the task in two seconds and confirming it immediately.

Patterns to Avoid

Manual web console navigation

✗ AVOID

Having to log into the Google Play Console, navigate to 'Subscriptions,' find the user's token, click 'Refund,' fill out a form, and wait for confirmation.

✓ INSTEAD

Just tell your agent: 'Refund the subscription linked to this token.' The MCP handles the entire process using `refund_subscription` and confirms success instantly in the chat.

Guessing which tool to use

✗ AVOID

Trying to fix a purchase issue by just listing products (`list_inapp_products`) when you actually need to check if the specific transaction is acknowledged.

✓ INSTEAD

To verify status, always start with `get_product_purchase`. Then, based on whether it needs confirmation or consumption, use either `acknowledge_product_purchase` or `consume_product_purchase`.

Treating reviews as generic text

✗ AVOID

Copy-pasting a complaint into a general support ticket without the context of when it was left or how many stars were given.

✓ INSTEAD

Use `get_review` to fetch the full, specific review data first. Then use your agent's natural language capabilities to draft and post a targeted response via `reply_to_review`.

The Right Fit

Use this MCP if your core pain point revolves around managing user-facing account states—specifically billing, reviews, or purchases. If you spend more time than necessary clicking between the Play Console and your CRM to handle customer inquiries (refunds, status checks, replies), this is for you. However, don't use it if your primary need is simply accessing basic app store metrics like download counts or regional sales data; those require different analytics tools. If you only need to list products without any interaction (like checking the catalog), `list_inapp_products` handles that specific read-only requirement.

The nightmare of manual customer billing support

Right now, when a user calls about a payment issue, you're trapped in the Play Console. You have to manually pull up their account details, check if they were billed correctly, determine if a refund is appropriate, and then initiate that action across multiple tabs and forms. It’s slow, error-prone, and takes five minutes minimum just to verify status.

With this MCP, you keep the customer conversation going while your agent handles the backend work. You simply ask for the user's purchase history or refund status, and the data appears instantly in the chat. The result is that you move from manual investigation to immediate resolution.

The Google Play Developer MCP gives you full control over purchases.

Before this MCP, handling a purchase meant navigating between different transaction logs: one tool for listing all products, another for checking the status of an item like `get_product_purchase`, and then separate clicks to either confirm it with `acknowledge_product_purchase` or mark it as consumed using `consume_product_purchase`. It was always a multi-step, disjointed process.

Now, you speak your intent into the chat. Your agent orchestrates all those steps behind the scenes—checking status, acknowledging receipt, and consuming the item—and gives you one definitive confirmation that it's done.

Google Play Developer: 12 Tools for App Ops

These tools let you manage every part of your app's lifecycle, from reading user reviews to processing payments and issuing refunds.

#	TOOL	DESCRIPTION
01	`acknowledge_product_purchase`	Confirms to Google Play that you have received and processed a user's in-app product purchase.
02	`cancel_subscription`	Cancels an active subscription for a specific user account.
03	`consume_product_purchase`	Marks a purchased digital item as 'used' or consumed by the app, preventing it from being bought again.
04	`defer_subscription`	Postpones the next scheduled billing date for a user's subscription.
05	`get_inapp_product`	Retrieves specific details, like pricing and name, for an in-app product SKU.
06	`get_product_purchase`	Checks the current status of a single in-app product transaction.
07	`get_review`	Fetches all the content and metadata for one specific review using its unique ID.
08	`get_subscription_purchase`	Checks the current status of a recurring subscription purchase, including billing cycle details.
09	`list_inapp_products`	Generates a complete list of all purchasable in-app products available for your app.
10	`list_reviews`	Pulls up the most recent user reviews left on your application.
11	`refund_subscription`	Initiates a formal refund of a user's subscription payment through Google Play services.
12	`reply_to_review`	Posts a drafted response directly to a specific user review left on the app store.

See It in Action

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

U Show me the 5 most recent reviews for our app.

Fetching reviews... You have 5 recent reviews. 3 are 5-star reviews praising the new UI. 1 is a 2-star review mentioning a login bug. Would you like me to draft a reply to the 2-star review?

U Check the status of subscription token 'abc123xyz' for SKU 'pro_monthly'.

Checking subscription... The subscription 'pro_monthly' for this token is currently ACTIVE. The next billing date is scheduled for October 15th.

U Refund the subscription linked to token 'refund789' immediately.

Processing refund dynamically...
Success. The subscription connected to the provided token has been formally refunded via Google Play services directly.

Frequently Asked Questions

01 **How do I use the Google Play Developer MCP to cancel a subscription?**

You tell your agent to cancel the subscription, and it uses the `cancel_subscription` tool. It handles the necessary calls and confirms the cancellation status right away.

02 **Can I check if an in-app product purchase was successful using Google Play Developer MCP?**

Yes, you use the `get_product_purchase` tool to read the current transaction status. This tells you exactly what's happening with that specific item.

03 **What is the difference between acknowledging and consuming a purchase via Google Play Developer MCP?**

Acknowledging confirms receipt (using `acknowledge_product_purchase`), while consuming marks the product as used by your app's logic, preventing re-use. Both are necessary for different types of digital goods.

04 **How do I get a list of all available products in my app?**

You run `list_inapp_products`. This tool pulls up the complete catalog, allowing you to see every purchasable item SKU and its details at once.

05 **Can I draft a response to a review using Google Play Developer MCP?**

Yes. You first use `list_reviews` or `get_review` to read the feedback, then you can ask your agent to write and post a reply directly using `reply_to_review`.

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 `"google-play-developer": { "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

Google Play Developer 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 Google Play Developer. 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	Google Play Developer MCP
Server ID	`019d75a9-0f9f-7202-b4ee-4a6ed85c3f37`
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/google-play-developer.