MCP SERVER | NO CODE | CLOUD HOSTED

Microsoft App Store MCP

Manage your app's entire publishing lifecycle.

Microsoft App Store MCP handles your entire app lifecycle management within the Microsoft ecosystem. Use it to monitor submission progress, manage add-ons, and coordinate package flights for beta testing or staged rollouts. It gives your AI client real-time visibility into everything registered under your developer account.

A+ Quality Score 100/100

app-distribution | submission-api | software-lifecycle | devops | app-management

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/microsoft-app-store

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.

Microsoft App Store MCP

8 tools available
Cloud-hosted on Vinkius

Managing an application across multiple releases is a pain. You need constant updates on submissions, in-app products, and rollout stages—stuff that used to require jumping between dozens of dashboards. This MCP connects your development workflow directly to the Microsoft Store Submission API. It lets you check every detail about your apps, from listing all registered applications to viewing metadata for add-ons. Need to know if a beta test package is ready? You can view those flights too. When connected via Vinkius, this tool gives your AI client immediate control over your entire store presence, making the whole publishing process visible and actionable right within your chat or IDE.

Core Capabilities

01 — Check App Inventory

List all applications registered in your developer account.

02 — Track Submission Status

Monitor the current status and details of any pending or completed app submissions.

03 — Manage Add-ons

List and retrieve metadata for in-app products associated with an application.

04 — View Rollout Packages

Get details on package flights used for beta testing or staged rollouts.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/microsoft-app-store** — connect your AI agent in three steps.

01 Subscribe to this MCP and provide your Azure Tenant ID, Client ID, and Client Secret credentials.

02 Your AI client authenticates with the Microsoft Store Submission API using those credentials.

03 You prompt your agent to perform a specific task, like listing applications or checking submission statuses.

The bottom line is you don't have to log into developer portals; your AI client just talks directly to the store data.

Built For

This MCP is for developers and product managers who own apps distributed via the Microsoft Store. If you spend time manually checking submission dashboards, this is for you. It eliminates the context switching between developer portals and your actual work.

DevOps Engineer

Automating release pipelines by ensuring that package flights are correctly configured before a production push.

Product Manager

Keeping track of the submission status and required metadata for new add-ons across multiple apps simultaneously.

Software Architect

Verifying that all dependencies, including listing applications and checking package flights, are correctly registered before a major deployment.

What Changes When You Connect

01 Eliminate dashboard hopping. Instead of logging into the developer portal to check submissions, you simply ask your agent for a list of submissions or use get_submission to confirm status instantly.

02 Keep track of everything in one place. Use list_applications to see all apps registered and then target specific ones using get_application details without leaving your chat window.

03 Coordinate controlled rollouts easily. You can view package flights with list_flights, letting you manage beta testing stages without manual checks for each rollout group.

04 Manage in-app revenue streams. Use list_addons to see all available add-ons and get_addon to check the specific metadata needed for product teams.

05 Get a full picture of your content. Running list_applications followed by list_submissions lets you map out exactly what's pending publication across your entire portfolio.

Real-World Applications

Confirming Beta Availability

A DevOps engineer needs to know if the new build is ready for internal testing. They ask their agent to run list_flights, which returns a list of package flights, allowing them to confirm that 'Internal Testing' is active before notifying QA.

Auditing App Portfolio Status

A Product Manager needs an overview of all apps and their latest submission statuses. They ask the agent to run list_applications first, then iterate through the results using list_submissions to build a comprehensive status report.

Debugging Add-on Metadata

A developer spots an incorrect price on an add-on in production. They use get_addon for the specific product ID, immediately retrieving the metadata needed to file a correction ticket without guessing IDs.

Pre-Release Check

Before launching, an architect needs assurance that all dependencies are accounted for. Running list_applications followed by listing add-ons ensures every in-app product is visible and correctly registered.

Patterns to Avoid

Assuming App Status

✗ AVOID

The developer assumes that because they updated the code, it's submitted. They wait hours and check the portal repeatedly.

✓ INSTEAD

Use list_submissions to proactively check the status of your submission immediately after deployment. This confirms exactly what stage the app is in.

Manual Add-on Tracking

✗ AVOID

The product team has 20 apps and manually checks each one's store page for add-ons, leading to missed updates.

✓ INSTEAD

Use list_applications first, then pair that with list_addons. This systematically gathers all in-app products across your entire catalogue.

The Right Fit

You should use this MCP if your job involves managing the lifecycle of published software within the Microsoft Store. Specifically, you need to check submission statuses (list_submissions), manage beta rollouts (list_flights), or audit your product offerings (list_addons). Don't use it if you only need basic API calls for general data storage; this is specialized publishing logic. If your core pain point is understanding *where* a piece of code is in the store pipeline, this MCP provides the precise tools needed to map out that journey.

The headache of checking app distribution status

Right now, if you want to know what happened with a recent update, you have to log into the developer dashboard. Then you navigate through submission tabs, find the ID, and wait for the page to load just to see if it says 'Pending' or 'Approved.' It’s slow, manual clicking, and you always lose context.

With this MCP, you simply ask your agent about a specific app's status. You get immediate access to list_applications and then use list_submissions to pull the current data without opening a single browser tab. The information comes right back to your workflow.

Get clear visibility with Microsoft App Store MCP

The tedious process of checking add-on details, verifying package flight statuses, or listing all registered apps is gone. You don't have to manually cross-reference IDs across three different tabs.

Now you can get a complete picture of your entire store presence by combining list_applications with list_addons and list_flights in one conversational flow.

Microsoft App Store: 8 Tools for Publishing Management

Use these eight tools to pull granular data on applications, submissions, add-ons, and package flight details from your Microsoft Store account.

#	TOOL	DESCRIPTION
01	`get_addon`	Get details for a specific add-on
02	`get_application`	Get details for a specific application
03	`get_flight`	Get details for a specific package flight
04	`get_submission`	Get details for a specific submission
05	`list_addons`	List add-ons for an application
06	`list_applications`	List all applications in your Microsoft Store account
07	`list_flights`	List package flights for an application
08	`list_submissions`	List submissions for an application

See It in Action

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

U **List all my Microsoft Store apps.**

Fetching applications... I've found 3 apps registered: 'Photo Editor Pro', 'TaskMaster', and 'Game Hub'.

U **What is the status of the latest submission for 'TaskMaster'?**

Checking submissions... The latest submission for 'TaskMaster' (ID #115292) is currently 'PendingPublication'.

Frequently Asked Questions

01 **How do I check the status of all my Microsoft Store apps using the Microsoft App Store MCP?**

You can combine list_applications to get every app name, and then use list_submissions on each one to pull the current submission status for a comprehensive audit.

02 **What is the difference between listing applications and getting an application's details?**

list_applications gives you a simple roster of every app ID. get_application uses that specific ID to fetch deep, rich metadata about one single app.

03 **Can I use the Microsoft App Store MCP to track beta rollouts?**

Yes. You can view package flights using list_flights and then check details for a specific rollout with get_flight, which is critical for controlled deployments.

04 Does this MCP handle add-ons and their metadata?

It does. Use list_addons to see every in-app product available for an app, or use get_addon if you only need details on one specific product ID.

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 `"microsoft-app-store": { "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

Microsoft App Store 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 Microsoft App Store. 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	Microsoft App Store MCP
Server ID	`019d75d4-68ca-7371-9b13-82fb51606fd6`
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/microsoft-app-store.