MCP SERVER | NO CODE | CLOUD HOSTED

Azure Service Bus Topic MCP for AI Agents

Reliably Triggering Cloud Events and Messaging Workflows

The Azure Service Bus Topic MCP lets your AI agent securely publish messages to a single, pre-defined cloud topic. This tool acts as a safe gateway for triggering downstream processes and system alerts without needing global permissions across your entire messaging infrastructure. If you need your LLM to reliably broadcast event data—like user sign-ups or order status changes—this MCP gives it that capability.

A+ Quality Score 100/100

pub-sub | event-publishing | messaging-system | cloud-events | notifications | fan-out

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/azure-service-bus-topic

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.

Azure Service Bus Topic MCP

1 tools available
Cloud-hosted on Vinkius

Your agent can now safely trigger events in your distributed systems using this specialized Azure Service Bus Topic MCP. Instead of giving the AI wide-open access across your entire cloud messaging setup, we strictly scope its power to one single topic. This means your agent gets a surgical ability: it publishes messages and triggers events exactly where you need them.

Think about building microservices or handling complex workflows. When an event happens—say, a user completes registration—your backend needs to tell multiple systems (like billing, notifications, and analytics) that it happened. Before this MCP, connecting AI agents to those critical internal buses was either too risky or far too complicated. Now, your agent simply calls the tool, sends a payload, and triggers a chain reaction of services in your architecture. It's contained, reliable event production, allowing you to connect your LLM logic directly into core business processes using Vinkius as your central catalog point.

Core Capabilities

01 — Publishing Event Messages

The agent sends a new message payload to the configured Azure Service Bus Topic, optionally attaching custom metadata for routing.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/azure-service-bus-topic** — connect your AI agent in three steps.

01 First, you tell your AI client exactly what event occurred and what data it should contain. This includes any necessary custom properties that define the message's type or priority.

02 The MCP validates the request against the Azure Service Bus Topic schema and sends the payload to the cloud topic.

03 Downstream workers, which are subscribed to this specific topic, instantly receive the event and execute their predefined business logic.

The bottom line is that your AI client uses this MCP to reliably broadcast system events into your private cloud bus, triggering automated action across multiple services.

Built For

This is for the DevOps engineer who needs to programmatically test event flows or the Backend architect managing microservices. If you rely on event-driven architecture (EDA), this MCP removes a massive integration hurdle by giving your AI agent controlled access to critical message buses.

Backend Engineer

Using the MCP, they test event producers by having their agent simulate real-world triggers—like a successful payment or profile update—without writing complex integration scripts.

DevOps Architect

They use this to model and validate system resilience. By triggering events through the MCP, they ensure that all downstream workers respond correctly and handle failures gracefully.

What Changes When You Connect

01 Event-Driven Triggers: Your agent can instantly signal that a major event occurred, kicking off complex workflows across services without manual intervention.

02 Security Scoping: Since the MCP is locked to one topic, you eliminate the risk of giving your AI client permissions to sensitive parts of your messaging system.

03 Rich Metadata Passing: You don't just send data; you attach custom properties. This metadata allows subscribed workers to route or process the message based on specific criteria.

04 Decoupled Systems: Your services become truly independent. The publishing agent doesn't need to know *how* the downstream service works, only that it needs to signal an event.

05 Simplified Integration: You get a single, reliable way to model and test event production logic for your entire microservices architecture.

Real-World Applications

Simulating User Sign-Up Events

A developer needs to ensure that when a user signs up, the billing service, notification queue, and analytics platform all get triggered. By using `publish_message`, they can have their agent send one event payload, confirming that the entire system processes the new user registration correctly.

Validating High-Priority Alerts

The operations team needs to confirm that critical alerts (like a server reboot command) are handled immediately. They prompt their agent to publish a message with a 'high' priority custom property, verifying the correct downstream workers pick up and process the urgent command.

Testing Order Processing Workflows

A QA tester wants to simulate an order completion event. They use `publish_message` to send a full order payload, then monitor the system to ensure inventory reduction, payment processing, and shipment notification all fire off sequentially.

Patterns to Avoid

Using direct API calls for every service

✗ AVOID

Trying to write custom code in your agent that directly calls the APIs of three different microservices (e.g., Service A, Service B, and Service C) when one event should suffice.

✓ INSTEAD

Instead, use this MCP's `publish_message` tool. Send a single event payload to the topic. Let the Azure Service Bus handle routing that message out to all three services automatically.

Over-scoping permissions

✗ AVOID

Giving your AI client global messaging permissions, meaning if it's compromised or misused, it could disrupt unrelated topics and systems.

✓ INSTEAD

This MCP solves that problem. Its design limits the agent to one specific topic, keeping its actions contained and safe for critical system triggers.

Relying on synchronous calls

✗ AVOID

Making your agent wait for a response from Service A before sending anything to Service B, which creates brittle dependencies.

✓ INSTEAD

Use event publishing. Send the message via `publish_message` and let all subscribed services react asynchronously, maintaining system stability.

The Right Fit

Use this MCP if your application relies on an event-driven architecture (EDA) or microservices that communicate primarily through a central messaging bus. It's perfect when you need to test the reliable fan-out of notifications—for example, ensuring a single 'OrderShipped' event triggers both inventory updates and customer emails. Don't use it if your goal is simple data storage; that requires a dedicated database MCP. Also, don't use it if you need to read message content or send messages across multiple topics; this tool's power comes from its absolute focus on one specific topic.

Azure Service Bus Topic MCP for AI Agents: Managing Cloud Event Triggers

In traditional systems, triggering a complex workflow means manually coordinating service calls. If an order ships, you might have to write code that first hits the inventory API, waits for confirmation, then posts to the notification queue, and finally updates the analytics database—a messy sequence of clicks and dependencies.

With this MCP, your agent sends one message via `publish_message`. That single action acts as the master switch. The entire downstream process runs automatically because all services are listening for that specific event. You get reliable, decoupled execution.

Azure Service Bus Topic MCP for AI Agents: Defining Event Metadata in Messaging

Manually defining how a message should be handled often means hardcoding logic into the consumer services. You have to update multiple endpoints every time you need to route messages differently (e.g., prioritizing high-value customers).

This MCP lets your agent attach custom properties when publishing. This metadata tells the receiving systems exactly what kind of event it is and how important it is, letting them process it correctly without needing code changes.

Azure Service Bus Topic: 1 Tool for Cloud Messaging Events

Use this tool to send structured messages and event payloads reliably into a single, designated Azure Service Bus Topic.

#	TOOL	DESCRIPTION
01	`publish_message`	Sends a new message to the configured Azure Service Bus Topic, allowing you to define routing metadata via custom properties.

See It in Action

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

U The user profile was updated with a new email address. Trigger the notification flow.

Event Published Successfully

- **Topic:** UserUpdateEvents
- **Payload Status:** Success
- **Properties Attached:** {"user_id": "XYZ123", "action": "email_change", "priority": "medium"}

The system is now processing the email change. You should see logs in the Notification Queue within seconds.

U Simulate a high-priority alert that requires an immediate reboot command for Server Alpha.

ALERT DISPATCHED

The following high-priority event has been published:

- **Event ID:** 987b4f2d
- **Target:** Infrastructure
- **Payload:** {"action": "reboot", "server_id": "Alpha"}
- **Priority:** HIGH

The infrastructure workers are now taking action. Watch the system logs for confirmation.

U Send an event payload indicating a successful product listing creation, including the SKU and category JSON.

Message Sent.
The new product data has been broadcast to the topic.

- **Topic:** ProductLifecycle
- **Payload Data:** {"sku": "P456", "category": "Electronics"}
- **Timestamp:** 2026-06-10T10:30:00Z

The indexing and cataloging workers will pick this up immediately.

Frequently Asked Questions

01 **How do I use the Azure Service Bus Topic MCP to trigger multiple actions?**

You send a single event message through the MCP. The system handles the rest by having various downstream services subscribe to that topic and reacting independently. It's all triggered by one reliable publish action.

02 **Can this MCP handle different types of events, like payments or user sign-ups?**

Yes. You control the event type by including custom properties in the message payload. This allows you to categorize and route the event correctly for whichever service needs it.

03 **Is this MCP secure? Will my AI agents have too much access?**

It is highly contained. The MCP limits your agent's ability strictly to one specific topic, preventing unauthorized publishing or changes across other parts of your infrastructure.

04 **What if I need to send an urgent alert? Can the Azure Service Bus Topic MCP handle priority?**

You can include a priority level in the custom properties when using the MCP. This metadata allows critical downstream workers to recognize and process your message as urgent.

05 **Does this service only work for my own cloud environment, or is it general purpose?**

It connects directly to your specified Azure Service Bus Topic. It's designed specifically to act as a controlled producer gateway into your existing enterprise messaging infrastructure.

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 `"azure-service-bus-topic": { "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

Azure Service Bus Topic 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

DOCUMENT INFORMATION

Generated	June 2026
MCP Server	Azure Service Bus Topic MCP
Server ID	`019e383a-c807-7097-bb21-dc08a4175618`
Platform	Vinkius Cloud for AI Agents
Endpoint	`https://edge.vinkius.com/{token}/mcp`

LICENSE & USAGE