MCP SERVER | NO CODE | CLOUD HOSTED

AI Token Counter MCP for AI Agents

Prevent Context Window Overflows in Large Document Processing

The AI Token Counter gives your AI agents self-awareness about context limits. It accurately counts the number of tokens, whether you're using OpenAI or Claude standards, preventing catastrophic API truncation errors. Use this MCP to safely manage massive datasets and ensure your complex pipelines never crash because a prompt was too big.

D Quality Score 59.84/100

tokenization | context-window | llm-optimization | cost-management | api-limits | encoding

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/ai-token-counter

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.

AI Token Counter MCP

1 tools available
Cloud-hosted on Vinkius

When an AI agent needs to summarize ten documents or process a giant JSON object, it can’t just send the whole thing to the Large Language Model (LLM) API. If that payload exceeds the model's context window—say, hitting the 128k token limit—the entire operation fails and your data pipeline dies. LLMs themselves can’t count tokens accurately before sending a prompt.

This MCP fixes that problem completely. It runs local math using the exact cl100k_base encoding algorithm. This means your agent can measure its own workload *before* it sends anything out. You can check if a massive dataset needs to be chunked, or maybe summarized in stages, all safely within your client workflow. With Vinkius managing this catalog, you connect once and gain the ability to give your agents this crucial self-awareness, turning potential API failures into predictable, manageable steps.

Core Capabilities

01 — Measure Input Data Size

You pass raw text, and the MCP returns a single number: the exact count of LLM tokens that payload contains.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/ai-token-counter** — connect your AI agent in three steps.

01 Feed the AI Counter any raw text—a document chunk, JSON data, or article snippet.

02 The MCP calculates the precise token count offline using standard encoding math.

03 Your agent receives a definitive number. It uses this result to decide if it should chunk the data, summarize it in stages, or send it directly.

The bottom line is: you get absolute certainty about how much text your AI client can safely push into an LLM API without crashing.

Built For

Prompt Engineers and Data Scientists are the primary users. If your workflow involves summarizing, extracting data from, or analyzing large volumes of text—like academic papers, meeting transcripts, or massive JSON logs—you're constantly battling context window limits. This MCP gives you control over that struggle.

Prompt Engineer

They use this to calculate the maximum size for a prompt block, ensuring they never hit an API limit when testing complex instructions.

Data Scientist

They run large datasets through the counter before writing extraction scripts, making sure their RAG pipelines don't fail due to excessive context loading.

Backend Developer (AI Focus)

They integrate this into data ingestion services to safely chunk and process multi-gigabyte document repositories piece by piece.

What Changes When You Connect

01 Stop API crashes dead. By using the `count_tokens` tool, your agent calculates token limits locally before sending data to an LLM, preventing fatal context overflow errors.

02 Manage massive datasets safely. Instead of guessing if a document fits, you get an exact count, allowing your pipeline to chunk text precisely and reliably.

03 Save money on API calls. Knowing the exact payload size means you write more efficient prompts, avoiding unnecessary retries or oversized requests that waste tokens.

04 Improve RAG reliability. For Retrieval-Augmented Generation (RAG) systems, this MCP ensures the gathered context never exceeds the LLM's capacity, keeping your answers grounded and available.

05 Build complex logic. Your agent can now execute decision-making: if token count > X, then chunk; else, summarize directly using `count_tokens`.

Real-World Applications

Summarizing a Stack of Legal Briefs

A paralegal asks their agent to summarize 15 attached legal briefs. Without this MCP, the resulting payload crashes the connection. With it, the agent runs `count_tokens`, sees the total is too high, and automatically chunks the input into five manageable batches for sequential summarization.

Analyzing a Large JSON Data Dump

A data scientist needs to extract key metrics from a 50MB JSON log file. They feed it to their agent; instead of crashing, the agent uses `count_tokens` to measure the size and processes the raw data in smaller, structured blocks.

Building Multi-Document Q&A Systems

The goal is to build an internal knowledge base chatbot. Instead of dumping all source material into one prompt, the agent uses `count_tokens` to measure retrieved documents and intelligently selects only the most relevant 3 sources that fit the context limit.

Handling Long-Form Academic Papers

A researcher wants an AI summary of a PhD dissertation. The full text is too long for one API call. The agent uses `count_tokens` to measure the document size and orchestrates a multi-step process: summarizing by chapter, then summarizing those summaries.

Patterns to Avoid

Sending raw data blindly

✗ AVOID

The agent is told to 'Summarize all 10 documents.' It bundles them into one prompt and sends it. The API returns an error because the total token count exceeds the model's context window.

✓ INSTEAD

Don't rely on blind sending. First, use `count_tokens` on the full payload estimate. If the result is too high, instruct your agent to chunk the documents first; then send each manageable piece individually.

Over-relying on model limits

✗ AVOID

A user assumes their LLM can handle 'a lot of text' and pastes 20,000 words into a single prompt. The API rejects the request due to internal constraints.

✓ INSTEAD

Always use `count_tokens` first. It gives you the hard limit measurement for your specific model type, allowing you to design the prompt structure around reality.

Ignoring encoding differences

✗ AVOID

The agent uses a general text length counter instead of an LLM-specific token counter, leading to inaccurate estimates and failed API calls.

✓ INSTEAD

Use `count_tokens`. This MCP calculates the count using the specific cl100k_base algorithm required by modern large models. It's the correct math for the job.

The Right Fit

You should use this AI Token Counter MCP if your primary pain point is hitting context window limits when processing large, varied inputs like documents, JSON files, or lengthy transcripts. If you need to ensure that every prompt sent to an external LLM API has a measured payload size before execution, this tool is mandatory.

Don't use it if you simply need to count characters (a word processor will do that) or if your input data is always guaranteed to be small and simple. If the task requires more than just counting—for example, generating the summary itself, or retrieving information from a database—you'll need other tools in the Vinkius catalog to complete the workflow. This MCP only provides the measurement; you still need an agent to make the decisions based on that count.

AI Token Counter for Context Window Management in RAG Pipelines

Today, building a Retrieval-Augmented Generation (RAG) system is tedious. You find all the source documents, gather them up, and dump them into one prompt. The agent sends it off hoping it fits. Most of the time, it doesn't. Your pipeline crashes, forcing you to manually trim data or guess at the optimal chunk size.

With this MCP, your agent gets a self-check. It runs `count_tokens` on the full set of retrieved documents. If the count is too high, it can’t crash; instead, it reports back: 'This context is 20% over capacity.' You get controlled data flow, not system failure.

AI Token Counter for Accurate API Cost Management in Data Ingestion

When ingesting large amounts of structured data, like thousands of records from a database dump, you usually copy-paste chunks into your agent's prompt. This is slow and wildly inaccurate because you never know the true token cost of that JSON structure.

This MCP solves the cost guessing game. By running `count_tokens` on the raw dataset before ingestion, you get a precise measure. You can then write code to chunk the data into optimal-size packets, guaranteeing predictable API usage and stable costs.

AI Token Counter: 1 Tool for Context Window Management

Use this tool to count exact LLM tokens in any piece of raw text. The result tells you exactly how much context your AI agent can handle before hitting an API limit.

#	TOOL	DESCRIPTION
01	`count_tokens`	Pass raw text and get the exact token count using cl100k_base, letting you decide if data needs chunking or summarizing before sending it to an LLM.

See It in Action

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

U How many tokens are in this huge block of JSON data?

Token Count Result: There are 8540 tokens. This means you can fit about 12 documents worth of context, but not the whole file.

Action Plan:
- Chunking is necessary.
- Process in batches of ~7000 tokens to leave room for your prompt instructions.

Use this number to guide your agent's next steps.

U Count the tokens in this article so I know if I can fit it into my next prompt.

Article Token Analysis:

METRIC	VALUE
Total Tokens	2,300
Estimated Context Fit (128k)	Safe

Note: This count includes all formatting and headers.

This article is safe for a single prompt; no chunking needed.

U I need the exact token count for this transcript snippet.

✅ **Token Count Result:** 1,520 tokens.

Summary: The snippet is well within most standard model limits. You can proceed directly with summarization or extraction without needing to chunk or pre-process the text further.

Frequently Asked Questions

01 **Why do I need an AI Token Counter MCP for AI Agents?**

You use it because LLMs have strict context limits, and if your input data is too big, the API call fails. This MCP gives your agents the math ability to measure their own workload, preventing crashes.

02 **Does this AI Token Counter help with cost management?**

Yes, it does. By knowing the exact token count of any data chunk before sending it, you can write pipelines that use the minimum necessary tokens, saving money and maximizing your API budget.

03 **What kind of documents can I feed into the AI Token Counter?**

You can feed almost anything: raw text from a document, large JSON logs, academic papers, or meeting transcripts. It counts tokens regardless of the source format.

04 **Is this better than just counting characters?**

Absolutely. Character count is meaningless for LLMs. This MCP uses the specific token encoding math that models like Claude and OpenAI actually use, giving you a precise measure of what the AI will read.

05 **Can I use this with my existing RAG system?**

Yes. Your agent can run this MCP right after retrieval. It measures how many documents were found and tells your agent if it needs to trim or chunk those results before generating an answer.

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 `"ai-token-counter": { "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

AI Token Counter 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 AI Token Counter. 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	AI Token Counter MCP
Server ID	`019eb8a2-294b-7033-9a33-567ffedb4947`
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/ai-token-counter.