MCP SERVER | NO CODE | CLOUD HOSTED

Wikidata MCP

Query global knowledge graphs from your AI agent.

Wikidata provides direct, structured access to the world's largest open knowledge graph. Run complex SPARQL queries, perform semantic vector searches for entities and properties, or fetch detailed facts about any item. Manage verifiable data relationships right from your agent.

A+ Quality Score 98.33/100

knowledge-graph | sparql | structured-data | semantic-search | data-retrieval | open-data

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

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.

Wikidata MCP

8 tools available
Cloud-hosted on Vinkius

You can treat Wikidata like a massive, interconnected database that speaks naturally with your AI client. Instead of asking your agent to guess at a fact based on general knowledge, you tell it to look up the source directly. You'll find everything from historical dates and scientific concepts to people's documented achievements. The tool lets you run deep SPARQL queries across millions of linked items—the kind of complex relationship mapping that generic AI models just can't do reliably. It also supports modern vector search, so if you know what you mean but not the exact keywords, your agent finds it anyway. When you connect this MCP through Vinkius, you get to tap into all these capabilities from one place, making structured data retrieval a natural part of any workflow.

Core Capabilities

01 — Querying relationships with SPARQL

Execute complex queries to map patterns and find specific links between entities across the entire knowledge graph.

02 — Retrieving verified facts by Item ID

Fetch all known statements, properties, and data points associated with a specific entity on Wikidata.

03 — Searching by meaning (Semantic Search)

Find relevant entities or properties based on the underlying concept of your query, not just keyword matches.

04 — Updating knowledge records

Add new statements, descriptions, or relationships to Wikidata items (requires OAuth access).

One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to the MCP and provide your User Agent identifier as required by Wikimedia policy.

02 If you need to write data back to Wikidata, connect an OAuth 2.0 Access Token.

03 Your agent can then execute complex queries or retrieve specific facts directly from this connection.

The bottom line is that your AI client gains a direct pipeline into highly structured, globally verifiable knowledge data.

Built For

This MCP serves researchers who need to verify facts, developers building systems on top of public data, and data scientists requiring structured datasets. If your job involves connecting disparate pieces of known information, you need this.

Academic Researcher

Verifies historical dates or scientific relationships by executing complex SPARQL queries against established records.

Data Scientist

Extracts structured datasets about entities for analysis or model training without needing to download raw CSV files.

Backend Developer

Finds entity IDs and property schemas, then uses the API to automatically enrich application data with external knowledge.

What Changes When You Connect

01 Go beyond simple keyword matching. Use `search_items_vector` to find entities related by meaning, which is crucial when you're brainstorming or defining abstract concepts.

02 Map complex relationships instantly. With `execute_sparql`, you can ask the system to find patterns—like 'all famous writers who lived in London and wrote about space travel.'

03 Get all the verifiable facts on a subject. Running `get_item_statements` provides a complete breakdown of an item's known properties, letting you build comprehensive profiles.

04 Automate data enrichment by finding structured knowledge. Use `get_item` to pull core details for any entity, feeding clean data directly into your application logic.

05 Contribute verified information. If you have new facts about an item, use `create_statement` or `set_item_description` (with OAuth) to update the graph.

Real-World Applications

Fact-Checking Academic Research

A student needs to verify a claim about a historical figure's primary occupation. Instead of relying on generalized LLM knowledge, they ask their agent to run `get_item` and then use `get_item_statements` against the specific person's ID to pull only verifiable records.

Building Product Knowledge Bases

A developer wants to build a tool that connects related software concepts. They run multiple `search_properties_vector` queries to map out all associated technical terms and relationships, ensuring their internal data structure matches the global standard.

Complex Data Discovery

A journalist is researching economic trends across several continents. They use `execute_sparql` to write a query that finds patterns linking 'population growth' with 'infrastructure spending' in multiple regions simultaneously, something impossible with simple chat prompts.

Semantic Data Lookup

A marketing analyst needs data on 'early 20th-century photography techniques.' They use `search_items_vector` because the exact term might not be in the graph, but the semantic similarity engine guides them to the correct related items.

Patterns to Avoid

Assuming general knowledge is sufficient

✗ AVOID

Asking your agent: 'What are the main properties of Q42?' The LLM will provide a summary, but it won't give you the underlying IDs or structured statements needed for coding.

✓ INSTEAD

Use `get_item_statements` on the item ID. This forces the system to pull all documented facts and their specific property links, giving you machine-readable data.

Handling complex cross-domain queries

✗ AVOID

Asking for 'the relationship between Roman coinage and modern industrial farming.' The LLM will hallucinate plausible connections because the query is too abstract.

✓ INSTEAD

Break it down. Use `execute_sparql` to first find all properties related to 'Roman coinage,' then use a second query on those results to see their documented links to other domains.

Trying to update data without permissions

✗ AVOID

Telling the agent: 'Set the description for Q192713.' The request fails silently or returns an error because no token was provided.

✓ INSTEAD

You must first connect your OAuth 2.0 Access Token, then run `set_item_description` to successfully modify the record.

The Right Fit

Use this MCP if your task is inherently about verifiable facts, structured relationships, or deep data querying. If you need to build a system that requires knowing 'who is related to whom' in a documented way—whether it's science, history, or technology—this is the tool. You should use `execute_sparql` when you know the exact pattern of data you want; use `search_items_vector` when you are browsing for concepts and need semantic help. Don't use this if your goal is creative text generation or summarization from unstructured documents, because while it can provide source material, its primary job is structured retrieval. If all you need is a general conversation about the topic, stick to your agent's native chat capabilities; only bring in Wikidata when the answer *must* come from a highly controlled, global knowledge graph.

Pulling facts from scattered sources feels like detective work.

Today, if you need to verify multiple facts about an entity—say, a person's life or a scientific concept—you end up clicking through Wikipedia, academic databases, and specialized wikis. You copy the date from one site, paste the occupation from another, and manually map out how they relate.

With this MCP, your agent handles the cross-referencing automatically. Instead of spending time on tabs and copy-pasting, you ask a single question, and the system executes complex queries to pull all verified statements in one go. You get structured data immediately.

Accessing Structured Knowledge with Wikidata

The tedious manual steps—copying an ID from a source, then running another search to find all its linked properties, and finally cross-referencing dates—all disappear. You simply ask your agent to retrieve the statements for the item.

What's different now is that you aren't just getting text; you're getting clean, structured data ready to be used in code or a database.

Wikidata with 8 Tools

These tools allow you to query, search, and even update data within the massive Wikidata knowledge graph directly from your AI client.

#	TOOL	DESCRIPTION
01	`create_statement`	Adds a new statement (fact) to an existing Wikidata item when provided with write permissions.
02	`execute_sparql`	Runs sophisticated SPARQL queries to find relationships and patterns across the entire dataset.
03	`get_item_statements`	Retrieves every known property and statement linked to a specific Wikidata item ID.
04	`get_item`	Pulls the core details and metadata for any specified Wikidata Item using its unique identifier.
05	`get_similarity_score`	Compares a piece of text against an entity to calculate how semantically similar they are.
06	`search_items_vector`	Performs hybrid searches across items, finding matches based on both keywords and overall meaning.
07	`search_properties_vector`	Searches for properties (types of facts) using a mix of keywords and semantic understanding.
08	`set_item_description`	Updates the main textual description of an item on Wikidata when provided with write permissions.

See It in Action

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

U Search for Wikidata items related to 'artificial neural networks' using vector search.

I've performed a vector search for 'artificial neural networks'. The most relevant item is 'artificial neural network' (Q192713), described as a computational model inspired by biological neural networks. Other related items include 'deep learning' (Q197536) and 'machine learning' (Q11660).

U Run a SPARQL query to find the 5 most populated cities in Brazil.

Executing the SPARQL query... The 5 most populated cities in Brazil are: 1. São Paulo (Q174), 2. Rio de Janeiro (Q8678), 3. Brasília (Q2844), 4. Salvador (Q36851), and 5. Fortaleza (Q43463).

U Get all statements for the Wikidata item Q42.

I've retrieved the statements for Douglas Adams (Q42). Key statements include: instance of 'human' (P31), occupation 'writer' (P106), and notable work 'The Hitchhiker's Guide to the Galaxy' (P800).

Frequently Asked Questions

01 **How do I use Wikidata with my agent for general fact retrieval?**

Use the `get_item` tool first. This fetches all core metadata and statements for any item ID, giving you a comprehensive overview of its known properties.

02 **Can I run complex queries using execute_sparql in my chat client?**

Yes, `execute_sparql` allows your agent to run advanced queries against the entire knowledge graph. This is how you find patterns that simple searches miss.

03 **What’s the difference between searching items and searching properties with Wikidata?**

Use `search_items_vector` when you want to find an entity (a person, place, or thing). Use `search_properties_vector` when you are looking for a specific type of fact or relationship.

04 **Is Wikidata MCP useful for updating data?**

Yes, if you have write access via OAuth, you can use tools like `create_statement` or `set_item_description` to add new facts or update existing descriptions.

05 **Do I need a specific item ID to get all statements?**

You must provide the unique Wikidata Item ID (e.g., Q42) to use `get_item_statements`. This tells the system exactly which entity's data you want.

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 `"wikidata": { "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

Wikidata 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 Wikidata. 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	Wikidata MCP
Server ID	`019e390b-29cf-732c-93ee-ba73ef2b84f5`
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/wikidata.