MCP SERVER | NO CODE | CLOUD HOSTED

DBpedia MCP for AI Agents

Query Structured Knowledge Graphs and Wikipedia Entities

DBpedia connects your AI agents to the world's largest open knowledge graph, structured around Wikipedia data. It lets you execute complex semantic queries using SPARQL, find specific entities with keyword searches, and pull real-time updates on global topics like people, cities, or scientific concepts.

A+ Quality Score 100/100

sparql | wikipedia | linked-data | semantic-web | knowledge-graph

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

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.

DBpedia MCP

8 tools available
Cloud-hosted on Vinkius

This MCP gives your agent a direct pipeline to DBpedia, the organized backbone of Wikipedia's knowledge. Instead of wading through unstructured articles, you can ask for precise data points: "What are the major population centers in Japan?" or "List all people related to quantum physics."

It handles everything from running complex SPARQL queries against public endpoints to fetching linked data (RDF/JSON-LD) for any resource. Need to know what's changed on a Wikipedia page since yesterday? You can monitor real-time updates, too. If you need a robust way to ground your AI client in factual, global knowledge, connecting via Vinkius and using this MCP is the fastest path. Your agent gains instant access to structured relationships between entities—a massive upgrade over simple web scraping.

Core Capabilities

01 — Execute complex graph queries

Run powerful SPARQL queries against public endpoints to extract highly structured data about global resources.

02 — Search and identify entities

Find specific Wikipedia resources using keywords or by completing prefixes, guiding your agent directly to the right topic.

03 — Retrieve linked resource details

Fetch all related data (RDF/JSON-LD) for a single entity, giving you a complete picture of its connections.

04 — Monitor real-time content changes

Track recent edits and updates across the global knowledge graph to ensure your information is current.

One Click on Vinkius — From Prompt to Execution

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

01 Confirm connection: Subscribe to this MCP and verify access to the public DBpedia endpoints via Vinkius.

02 Formulate the query: Tell your AI client exactly what you need—a search term, a graph pattern for SPARQL, or a resource name.

03 Get results: The agent executes the request using the appropriate tool, and you receive structured data (JSON-LD) detailing relationships and facts.

The bottom line is that you don't have to write complex scraping code; your agent just asks for knowledge, and this MCP returns it in a clean, usable format.

Built For

Data scientists and researchers who constantly deal with interconnected global data need this. If your workflow involves extracting facts or building semantic models from public sources, you're wasting time manually scraping pages when you should be querying a graph.

Data Scientist

Runs SPARQL queries to extract structured datasets about global phenomena (like population trends or scientific relationships) that simple API calls can't handle.

AI Engineer

Gives their agents a factual, reliable source of general knowledge and real-time event data for grounding complex reasoning tasks.

Academic Researcher

Searches for highly specific entities or monitors how Wikipedia articles on niche topics are being updated in real time.

What Changes When You Connect

01 Stop manual scraping. Instead of copy-pasting data from web pages, you run a SPARQL query to get the exact structured dataset you need immediately.

02 Stay current on facts. Tools like `get_live_changes` let your agent monitor real-time updates across Wikipedia, ensuring the knowledge it uses is fresh.

03 Pinpoint resources fast. Use `lookup_search` or `lookup_prefix` to quickly find specific entities (like people or scientific works) before running a query.

04 Get full context. Once you identify an entity using `get_resource`, you pull all its linked data, giving your agent a complete relationship map.

05 Handle bulk requests easily. Tools like `retrieve_live_articles` let you gather data for multiple resources at once without writing repetitive code.

Real-World Applications

Building an academic research tool

A researcher needs to track all documented connections between 'Quantum Computing' and 'Cryptography'. They use `query_sparql` to pull a structured dataset of relationships, which is far faster than manually cross-referencing dozens of Wikipedia pages.

Monitoring breaking news coverage

A journalist needs to know if major city profiles have been updated recently. They use `get_live_changes` and then `get_live_resource` on specific city pages to confirm the latest edits before writing a report.

Developing an internal knowledge base

A developer needs to enrich their application with reliable global entity data. They use `lookup_search` to identify key industries and then `get_resource` to pull the full linked properties for those entities.

Comparing historical vs current facts

An AI agent needs to compare old information with new developments on a topic like 'Artificial Intelligence'. It uses `query_sparql` to get baseline data and then `get_live_resource` to see the most up-to-date context.

Patterns to Avoid

Trying to scrape entire websites

✗ AVOID

Asking the agent to 'just read the whole Wikipedia page on space travel' and hoping it extracts all structured data points. The result is often messy, unstructured text that requires manual cleanup.

✓ INSTEAD

Instead, use `get_resource` or a targeted `query_sparql`. Identify the specific entity (e.g., 'Space Travel') first using `lookup_search`, then ask for the linked data to get clean facts only.

Assuming all links are current

✗ AVOID

Using old, static API calls that pull cached versions of information, leading to reports based on outdated population counts or scientific claims.

✓ INSTEAD

Always use the live tools. Use `get_live_resource` or `query_live_sparql` whenever timeliness is important to guarantee you're working with recent edits.

Querying without knowing the entity type

✗ AVOID

Running a vague query that returns thousands of irrelevant results, making it impossible for the agent to determine which data points are relevant.

✓ INSTEAD

Start by scoping your search. Use `lookup_prefix` or `lookup_search` first to narrow down the target Wikipedia entity, then use `get_resource` on that specific ID.

The Right Fit

Use this MCP if your core problem is extracting structured facts or relationships from publicly available knowledge graphs. If you need to ask 'What are the connections between X and Y?' or 'Give me a list of all resources about Z in JSON format,' this is the right tool. Don't use it if you need to process proprietary, private business data; DBpedia is public domain. Also, don't rely solely on its basic `query_sparql` without confirming freshness; always check if `get_live_changes` or `query_live_sparql` is necessary for your specific topic.

DBpedia MCP: Accessing Structured Knowledge Graph Data

Today, getting structured data from Wikipedia feels like a scavenger hunt. You click through tabs, copy numbers into spreadsheets, and spend hours manually piecing together relationships between people, concepts, or locations just to build one basic dataset.

With this MCP, you simply ask your agent for the knowledge graph connections you need. Instead of tedious clicking, you get clean, structured output immediately, giving you a ready-to-use JSON object containing all the facts and relationships you requested.

DBpedia MCP: Tracking Real-Time Wikipedia Updates

The biggest pain point is knowing if the data you pulled yesterday is still accurate today. You're forced to bookmark pages and check them manually, wasting time validating facts that might have changed overnight.

Now, your agent uses `get_live_changes` and `get_live_resource`. This lets it monitor for updates automatically, ensuring every fact—from population counts to scientific theories—is synchronized with the very latest Wikipedia edits.

DBpedia: 8 Tools for Semantic Queries and Knowledge Graph Analysis

These tools let your agent execute everything from basic keyword searches to complex, real-time SPARQL graph queries against the DBpedia knowledge base.

#	TOOL	DESCRIPTION
01	`retrieve_live_articles`	Gets the latest data and details for a list of specified resource names from Wikipedia.
02	`get_live_changes`	Lists all recent change events that have occurred on the DBpedia Live Sync API.
03	`get_live_resource`	Fetches the most current data and metadata for a specific Wikipedia page or resource.
04	`query_live_sparql`	Executes a SPARQL query specifically against the DBpedia Live endpoint for real-time updates.
05	`lookup_prefix`	Performs an autocomplete search, suggesting potential Wikipedia resource names as you type keywords.
06	`lookup_search`	Searches for DBpedia resources using general keywords to pinpoint relevant entities quickly.
07	`get_resource`	Retrieves the full linked data (RDF, JSON-LD) structure for a specific, identified DBpedia resource.
08	`query_sparql`	Runs a standard SPARQL query against the main public DBpedia endpoint to gather structured information.

See It in Action

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

U What are some key facts about Tokyo's history and demographics?

Tokyo: Key Data Points

- **Population:** Approximately 13.9 million (Metro Area)
- **Founding/Key Event:** Established as a major center during the Edo Period.
- **Related Entities:** Yokohama, Edo, Japanese history.

I pulled this data using structured lookups and linked properties to give you clean facts.

U Find all resources related to 'Quantum Computing' that mention specific algorithms.

Quantum Computing Related Algorithms Found:

- **Algorithm Name:** Shor's Algorithm (Context: Cryptography)
- **Algorithm Name:** Grover's Algorithm (Context: Search/Optimization)
- **Core Resource URI:** dbpedia://quantum_computing

These are the primary, structured links connecting algorithms to the main field.

U List recent changes on Wikipedia regarding deep learning architecture.

Latest Updates for Deep Learning Architecture:

- **Event 1 (Time: Today, 10:30 AM):** Article updated with new links to Transformer models.
- **Event 2 (Time: Yesterday, 4:15 PM):** Added population data update for key research hubs.
- **Summary:** The knowledge graph is highly active in this area; I recommend running a detailed SPARQL query next.

Frequently Asked Questions

01 **How do I use the DBpedia MCP to get structured data from Wikipedia?**

You start by telling your agent what kind of information you need, like 'all people who worked on quantum computing.' The MCP then uses SPARQL queries and entity lookups to pull back a clean JSON structure with facts, bypassing messy text.

02 **Can the DBpedia MCP track changes on Wikipedia articles?**

Yes. You can use the live tools within this MCP to monitor recent edits or find out exactly when specific resources were last updated, which is critical for research integrity.

03 **Is the data from DBpedia reliable enough for academic work?**

Since it's sourced directly from Wikipedia and structured by the global knowledge graph, it provides a highly detailed and interconnected view of facts. Always cross-reference with primary sources, but the structure is excellent.

04 **What if I don’t know the exact name of the entity?**

No problem. Use the `lookup_search` tool within the MCP first. You just need to provide a few keywords, and it will suggest potential Wikipedia entities for you to select.

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 `"dbpedia": { "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

DBpedia 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	DBpedia MCP
Server ID	`019e3887-ed80-739e-93e8-c221a417d9fb`
Platform	Vinkius Cloud for AI Agents
Endpoint	`https://edge.vinkius.com/{token}/mcp`

LICENSE & USAGE