

rudof-MCP: A Model Context Protocol Server for Semantic Web Operations

Samuel Bustamante-Larriet^[0009-0005-8631-2682], Diego
Martín-Fernández^[0009-0003-6640-9474], Álvaro
García-Fernández^[0009-0008-9390-6210], Daniel
Fernández-Álvarez^[0000-0002-8666-7660], and Jose-Emilio
Labra-Gayo^[0000-0001-8907-5348]

University of Oviedo, Spain
bustamantesamuel@uniovi.es

Abstract. rudof is a Rust library that offers Semantic Web operations (RDF processing, SPARQL queries, ShEx/SHACL validation, among others). The Model Context Protocol (MCP) provides a standardized interface for AI agents to interact with external systems. This demo presents rudof-MCP, an MCP server exposing rudof’s functionalities to AI agents through natural language. By delegating Semantic Web operations to rudof at inference time, LLMs can reduce dependence on pretraining knowledge and ground their outputs in auditable tool invocations that ensure the accuracy and reproducibility of results. In this demo, we showcase this by comparing Claude with and without access to rudof-MCP on a set of Semantic Web tasks.

Keywords: Model Context Protocol · Semantic Web · Knowledge Graphs · Large Language Models · RDF · ShEx · SHACL · SPARQL

1 Introduction

The convergence of Large Language Models (LLMs) and Knowledge Graphs (KGs) represents a promising frontier in artificial intelligence, where natural language processing capabilities meet the precision of formal structured knowledge [3]. However, this integration faces fundamental obstacles: LLMs operate as black boxes whose reasoning processes remain opaque, and relying on pretraining knowledge for KG-related tasks is risky, as there is no guarantee of accuracy or reproducibility of results. A concrete illustration of this risk is found in evaluations of LLMs on KG-related tasks, which reveal that models exhibit preferences toward specific RDF serialisation formats depending on the task [2].

To address these challenges, we present rudof-MCP, an MCP (Model Context Protocol) server built on top of rudof [1], a Rust library that offers Semantic Web operations (RDF processing, SPARQL queries, ShEx/SHACL validation, among others). MCP is an open protocol that standardises communication between LLMs and external systems, enabling them to invoke auditable tools at

inference time rather than relying on pretraining knowledge, ensuring the accuracy and reproducibility of results. By exposing rudof’s functionalities through this protocol, rudof-MCP allows LLMs to delegate Semantic Web operations to a specialised system.

This work is guided by the following research question: to what extent does MCP-based access to rudof’s capabilities improve the performance of LLMs on knowledge graph engineering tasks, such as exploration, querying, and reasoning over RDF data.

While a full empirical evaluation remains as future work, this demo illustrates the potential of the approach by comparing Claude with and without access to rudof-MCP on a set of Semantic Web tasks, demonstrating how rudof-MCP enables Claude to delegate Semantic Web operations to a specialised system, improving both the accuracy of results and their explainability.

2 MCP Server Capabilities

The rudof-MCP server exposes rudof’s Semantic Web capabilities through MCP. Figure 1 illustrates the main functionalities provided. The server has been implemented using the official Rust MCP SDK version 1.3.0¹. The MCP server implementation is open and available in the rudof GitHub repository².

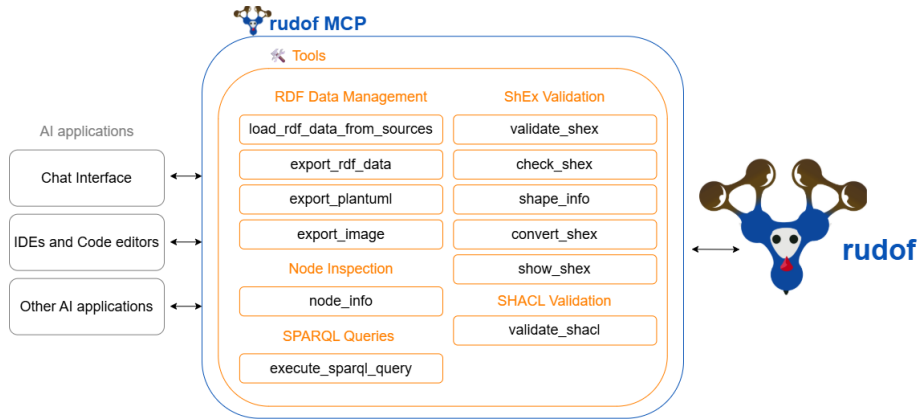


Fig. 1. Main capabilities exposed by rudof-MCP.

3 Demonstration

We demonstrate rudof-MCP using Claude Desktop with a simple RDF dataset describing an organization and its three employees. The demonstration compares

¹ <https://github.com/modelcontextprotocol/rust-sdk>

² <https://github.com/rudof-project/rudof>

two scenarios: without MCP access (Claude relies solely on its language model capabilities) and with MCP access (Claude leverages rudof tools directly). An example of interaction between Claude Desktop and rudof-MCP is shown in Figure 2. The following six tasks were executed across both scenarios:

What are the production employees of Acme, their names and ages?

Thinking about identifying production employees at Acme with personal details ▾

🕒 The user wants to query the RDF data for production employees of Acme, their names and ages. I'll use a SPARQL query via the `execute_sparql_query` tool.

🔍 Loading tools

Result

R Execute SPARQL Query

```

... text
| | ?employee | ?name | ?age |
|-----|-----|-----|
| 1 | ex:Alice | "Alice" | 30 |
|-----|-----|-----|
| 2 | ex:Bob | "Bob" | 27 |
|-----|-----|-----|

```

🟢 Done

`ex:Acme` has 2 production employees, retrieved via `ex:hasProductionEmployees`:

Employee	Name	Age
<code>ex:Alice</code>	Alice	30
<code>ex:Bob</code>	Bob	27

Fig. 2. Claude Desktop interaction with rudof-MCP: a natural language query is translated into SPARQL by Claude and executed via the rudof tool.

- **Inspect specific node:** Retrieve detailed information about `ex:Alice`, including all incoming and outgoing predicates. Without MCP access, Claude reasons over the RDF data provided in the prompt and returns the correct information; with MCP access, Claude invokes the rudof tool and retrieves the same information.
- **Execute natural language query:** Ask “*What are the production employees of Acme, their names and ages?*”. Without MCP access, Claude answers correctly reasoning directly over the RDF data; with MCP access, Claude explicitly translates the question into a SPARQL query and executes it via rudof, returning the correct results.

- **Execute SPARQL query:** Retrieve all persons and their ages in descending order. Without MCP access, Claude generates and executes a Python script using `rdflib` to produce the correct results; with MCP access, Claude delegates execution to `rudof` and obtains identical results.
- **Validate with ShEx:** Validate the dataset against a ShEx schema containing a negative cycle, an invalid construct in ShEx. Without MCP access, Claude fails to detect the error, generating and executing a script via `pyshex`; with MCP access, Claude successfully identifies and reports the invalid construct.
- **Validate with SHACL:** Validate the dataset against a SHACL schema. Without MCP access, Claude generates and executes a script using `pys hac1`, returning the correct result; with MCP access, Claude invokes `rudof`, which applies its SHACL validation algorithm and confirms that all three person nodes satisfy the defined constraints.
- **Visualize RDF graph:** Generate a visual representation of the RDF dataset. Without MCP access, Claude produces an inaccurate rendering, failing to correctly capture the graph structure; with MCP access, Claude invokes `rudof` and returns an accurate rendering.

Reproducibility. Complete instructions for configuring Claude Desktop with `rudof-MCP` are in the `rudof` manual³. Demonstration files and complete Claude Desktop interactions are available in the project’s GitHub repository⁴.

4 Conclusions and Future Work

This work presents `rudof-MCP`, an MCP implementation that exposes `rudof`’s Semantic Web functionalities to AI agents. As demonstrated, agents such as Claude equipped with `rudof-MCP` can perform node inspection, SPARQL querying, ShEx and SHACL validation and RDF visualisation through a specialised system, improving both the accuracy and explainability of results. This suggests that offloading specialised operations to dedicated tools at inference time can enhance LLM capabilities on knowledge graph tasks.

That said, this approach has one primary limitation. The quality of the MCP-enhanced results is fundamentally bounded by the correctness and completeness of the underlying `rudof` library: any bugs or unsupported features will directly affect the agent’s results. This dependency highlights the importance of investing in the robustness and coverage of the tool layer, not only in the LLM integration itself.

Future work will focus on two directions. First, `rudof-MCP` will be evaluated using the LLM-KG-Bench 3.0 benchmark⁵ to provide a quantitative answer to the research question posed in this work: to what extent does MCP-based access

³ https://rudof-project.github.io/rudof/cli_usage/mcp.html

⁴ <https://github.com/rudof-project/eswc2026-rudof-mcp>

⁵ <https://github.com/AKSW/LLM-KG-Bench>

to rudof’s capabilities improve LLM performance on knowledge graph engineering tasks. Second, the rudof library will be improved to provide AI agents with a richer and more reliable toolset for Semantic Web tasks.

4.1 Acknowledgements

This work has been partially funded by the project *SHAKIGRA*: Shaping Knowledge and Interoperable Graphs, code: NAC-ES-PUB-ASV-2025 PID2024-157010OB-I00, from the Spanish Research Agency, and the regional project with code SEK-25-GRU-GIC-24-089.

Disclosure of Interests

The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Labra-Gayo, J.E., Iglesias-Préstamo, A., Martín-Fernández, D., Arnaud, M.A.: Rudof: A Rust Library for handling RDF data models and Shapes. In: ISWC 2024 Posters and Demos. vol. 3828 (2024)
2. Meyer, L.P.: Evaluating Large Language Models for RDF Knowledge Graph Related Tasks - The LLM-KG-Bench-Framework 3 (May 2025). <https://doi.org/10.5281/zenodo.16779482>
3. Pan, S., Luo, L., Wang, Y., Chen, C., Wang, J., Wu, X.: Unifying Large Language Models and Knowledge Graphs: A Roadmap. *IEEE Transactions on Knowledge and Data Engineering* **36**(7), 3580–3599 (Jul 2024). <https://doi.org/10.1109/TKDE.2024.3352100>