English

1. Tools

LoKI supports MCP (Model Context Protocol) tools. These tools can integrate local programs, web APIs, or automations into the context of the LLM.

Here are the tools for each MCP mode. These are examples and are not fully developed in some cases. This is intentional, so that you can at least see how the tools work and can tackle the expansion or development of new tools yourself. More on this in the Editor Doc.

All tools use the JSON-RPC 2.0 protocol.

In the Plugins folder, you will find the following folders:

local: This folder contains the tools for the local MCP mode. These are examples and are written in Python. The Python tools also use the Python312 environment from the AppImage so that there are no system dependencies exist. This folder also contains tools that access the ModelUseFiles folder and can be used by the LLM.

stdio: This folder contains tools that allow more system access. These include tools that use the browser (ms-playwright) integrated into the AppImage. The integrated Node modules are also used here. Note that all tools in this folder use an mcp-tool-loader that handles scanning the tools. When creating your own tools, please model them after the existing tools to ensure functionality.

http: This contains an example of a working server. This server is automatically started by LoKI in this mode.

The respective mode is set via the Settings page. The preset must be selected before starting the LLM. While the LLM is active, no change to the MCP mode is possible because the plugins/tools are scanned when the LLM starts, depending on the mode, and the LLM is then announced in the context. The plugins/tools found are displayed on the main page on the right-hand side in the “Loaded Plugins” window. The description of each tool is also shown there.

Please note that if you wish to create your own tool, you should model it after the existing tools. For the tools in the stdio folder, a custom mcp-tool-loader.ts file is used, which must be included in any case so that the tools are recognized when the LLM starts. Furthermore, a unique description must be provided, as this is injected into the LLM’s prompt so that the LLM knows how to use it.

The browser tools in the stdio folder support the browser integrated into LoKI (Playwright).

2. Editor

The editor is an integrated code and text editor. It allows you to edit plugin scripts, configuration files, and documentation. Syntax highlighting makes it easier to work with JSON, Python, or JavaScript.

The editor can transfer documents directly to the currently running model using the three buttons at the bottom, for example, to explain code, check it, or automatically generate documentation.

Tip: If you set the ModelUseFiles path in the settings to the Plugins/local, Plugins/stdio, or Plugins/http folder and then select “local” in MCP mode, the LLM can access the tools directly via the file tools, read them, extend them with the correct prompt, and so on.

Note: The plugins/tools in the folder are kept very simple and are only intended to demonstrate how LoKI works. For example, the Calc tool in the Local folder is very simple. However, it is ideal for extension by a useful LLM if you follow the tip above.

3. Prompt Example

-- Create Test.c with a “Hello World” program

-- Read test.c

in Image to Text mode:

-- Describe the image.

-- Open Wikipedia for this topic.

Agent Mode

Create an HTML file showing a rocket launch. Save it as Rocket.html and open it in a browser.