Contents

[MODEL CONTEXT PROTOCOL (MCP) 2](#_Toc208496632)

[What is MCP? 2](#_Toc208496633)

[Why MCP Was Needed? 2](#_Toc208496634)

[How MCP Solves These Problems? 2](#_Toc208496635)

# MODEL CONTEXT PROTOCOL (MCP)

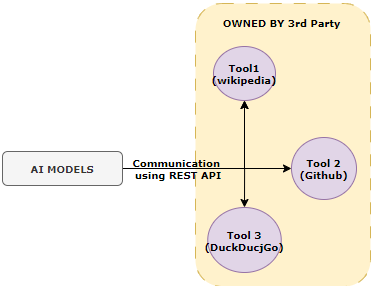
## WHAT IS MCP?

* **Model Context Protocol (MCP)** is an **open standard** introduced by **Anthropic** to simplify and standardize how **AI models (LLMs)** interact with external tools and data sources.

|  |
| --- |
| USB Analogy   * Think of MCP like a USB port:   + Your laptop (AI model) connects to devices (tools) using USB.   + Devices (keyboard, mouse, etc.) must follow USB standards.   + Even if the device changes, laptop still works with it — no need to change your laptop. * Similarly, with MCP:   + AI models connect to tools using MCP.   + Tools must follow MCP standards.   + Even if tools change, your AI model still works — no need to update your code |

## WHY MCP WAS NEEDED?

Before MCP, AI applications used **REST APIs** to connect to tools/external services(provided by 3rd Party service providers). This approach has several challenges:



Problems with REST APIs Implementation:

1. **Tight coupling**: Any change in the tool’s API required changes in the client code.
2. **Maintenance overhead**: Frequent updates from third-party tools meant constant refactoring.
3. **Scalability issues**: Integrating multiple tools require custom logic for each.

## HOW MCP SOLVES THESE PROBLEMS?

Anthropic introduced a **standardized protocol layer** between the AI model and external tools called MCP

**Instead of:** LLM → REST API → Tool

**Now:** LLM → MCP Protocol → Tool

A diagram of a server

AI-generated content may be incorrect.

Instead of accessing the tool via rest API, LLMs will be accessing the tools using MCP

Benefits:

* **Loose coupling**: Tools can evolve independently.
* **Plug-and-play**: New tools can be added without changing client logic.
* **Standardization**: All tools follow the same protocol format.

## MCP IN GENERATIVE & AGENTIC AI APPLICATIONS

Example

* Let’s say we are building an **AI assistant** that answers user queries. For that it might need access to various 3rd party services like Wikipedia, A vector database or a search engine
* **With REST APIs**
  + We need to write custom code for each tool.
  + If the tools (Wikipedia) change their API, we must update our code.
  + Every tool had a different format, making integration complicated.
* **With MCP:**
  + We can connect all tools via MCP.
  + **No need to change our code when tools update — because those service providers will be adhered to MCP standards.**

### WORKFLOW

* **REST API Workflow:** User → AI Assistant → Wikipedia API (REST) → JSON Response → Display
* **MCP Workflow:** User → AI Assistant → MCP Protocol → Wikipedia Tool → Standardized Response → Display

# COMPONENTS OF MCP PROTOCOL

1. MCP Client
2. MCP HOST
3. MCP Server

A diagram of a computer server

AI-generated content may be incorrect.

## MCP CLIENT

* The **Client** is the **interface** where users interact with the system.
* It captures user inputs (like commands, queries, or requests) and sends them to the Host.
* It displays the responses received from the system.
* Think of it as the **front-end** or **user-facing component**

## MCP HOST

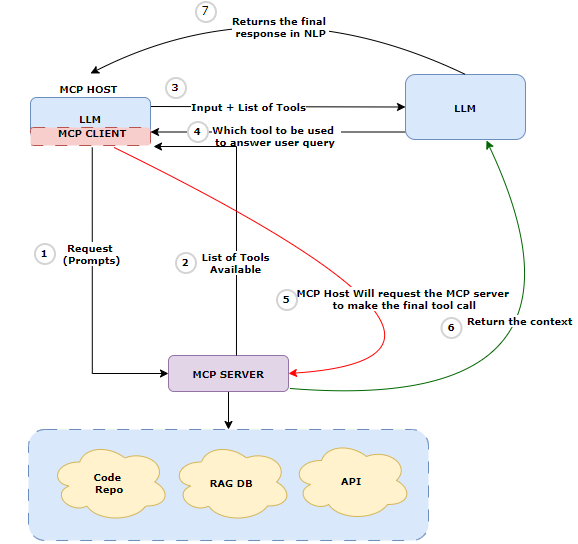
* The **Host** acts as a **communication bridge** between the Client and the Server.
* It receives requests from the Client, forwards them to the Server, and returns the Server’s responses back to the Client.
* It may also handle tasks like formatting, validation, or logging.
* Think of it as the **middleware** that manages data flow.

## MCP SERVER

* The Server is the core processing unit.
* It performs actual computation, logic execution, or integration with external systems.
* It receives structured requests from the Host, processes them, and sends back results.
* Think of it as the back-end engine that powers the protocol.
* The MCP servers are implemented by 3rd parties (external system ) for seamless integration with their system.
* Those service provider must follow a **standard set of rules and message formats** defined by the MCP Protocol. This ensures:
  + **Interoperability**: The server can communicate seamlessly with any MCP Client and Host.
  + **Consistency**: Requests and responses follow a predictable structure.
  + **Security and Reliability**: Protocol rules often include authentication, error handling, and data validation mechanisms.

**The protocol used for the communication between MCP host and MCP server is Model Context Protocol**

# COMMUNICATION BETWEEN COMPONENTS OF MCP



* **User Prompt Initiation:** The user enters a natural language query in any Copilot-enabled interface (e.g., IDE, chat app, web portal).
* **Prompt Reception by MCP Host:** The MCP Host receives the prompt and forwards it to the MCP Server.
* **Tool Discovery by MCP Server:** The MCP Server identifies the tools (skills, plugins, APIs) available for the current user and context.
* **Prompt + Tool List Sent to LLM:** The MCP Host sends the user prompt along with the list of available tools to the LLM.
* **Tool Selection by LLM:** The LLM analyzes the prompt and tool metadata to determine which tool(s) to invoke.
* **Tool Invocation Request to MCP Server:** The MCP Host requests the MCP Server to invoke the selected tool(s) with the required parameters.
* **Tool Execution and Result Return** The tool executes (e.g., API call, function execution) and returns the result to the MCP Server.
* **Result Sent to LLM for Interpretation:** The MCP Server sends the tool result back to the LLM.
* **LLM Generates Final Response:** The LLM uses the tool result and original prompt to generate a natural language response.
* **Response Displayed to User:** The final response is shown to the user in the interface they initiated the query from.