AI Agents in Life Science: Hands on Session 2 - AI agent collaboration with the Model Context Protocol (MCP)

Connect to an existing host (Postman)

In this guide, you'll learn how to connect your running MCP servers to a real-world host application. We'll use **Postman**, which has built-in MCP support, to act as our client. This will allow you to explore your server's tools and resources interactively, just as an AI assistant would.

Prerequisites

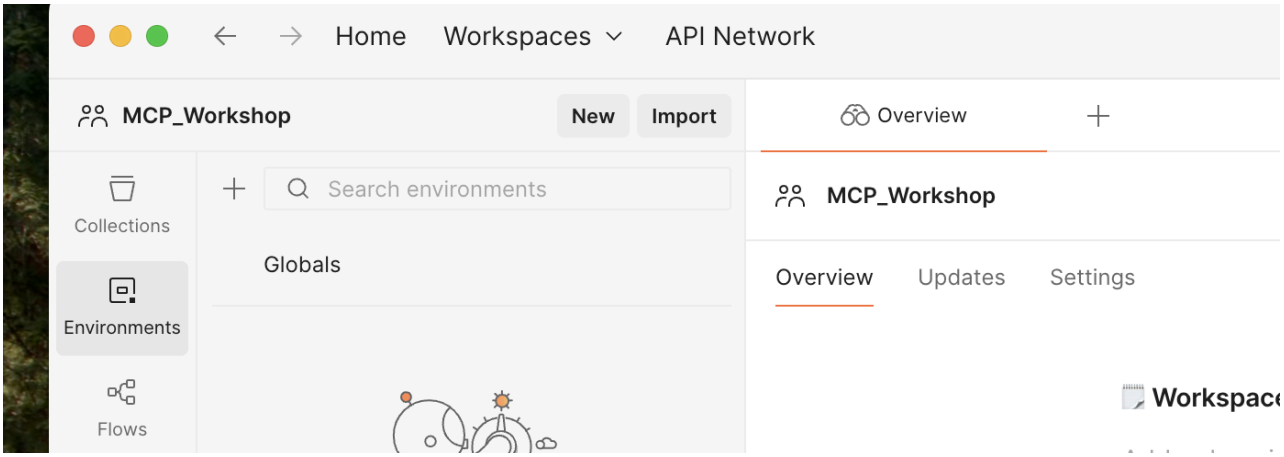
If you haven't already, download and install the Postman desktop app.

Download Link: https://www.postman.com/downloads/

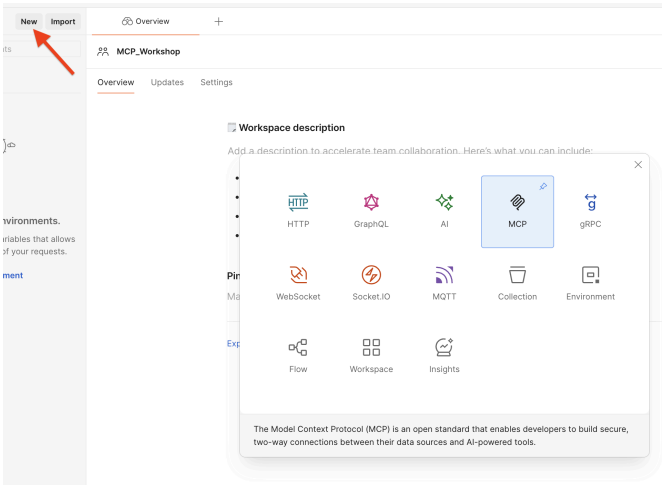
Connect to the Servers

1. Let's connect Postman to the `SciLifeLab_agent_workshop/Section_2_MCP/existing_clients_and_servers/postman_mcp_basic_server.py` file.

2. **Open Postman:** Launch the Postman application and create a workspace if prompted (e.g., `MCP_Workshop`).



3. **Create New MCP Request:** In your workspace, click the **`New`** button.

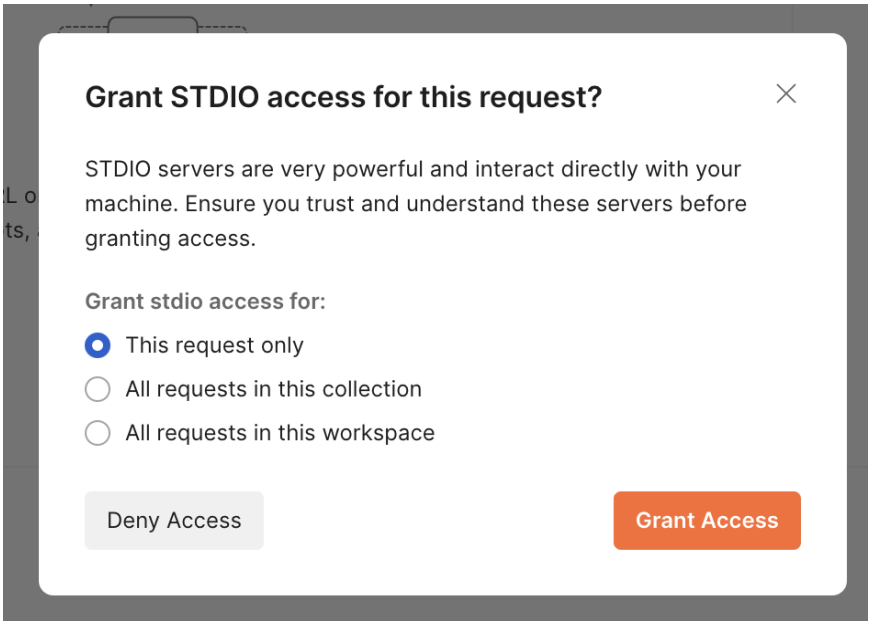


4. **Rename the Request:** Click the default name (e.g., `Untitled Request`) and change it to something descriptive, like `My_MCP_Server`.

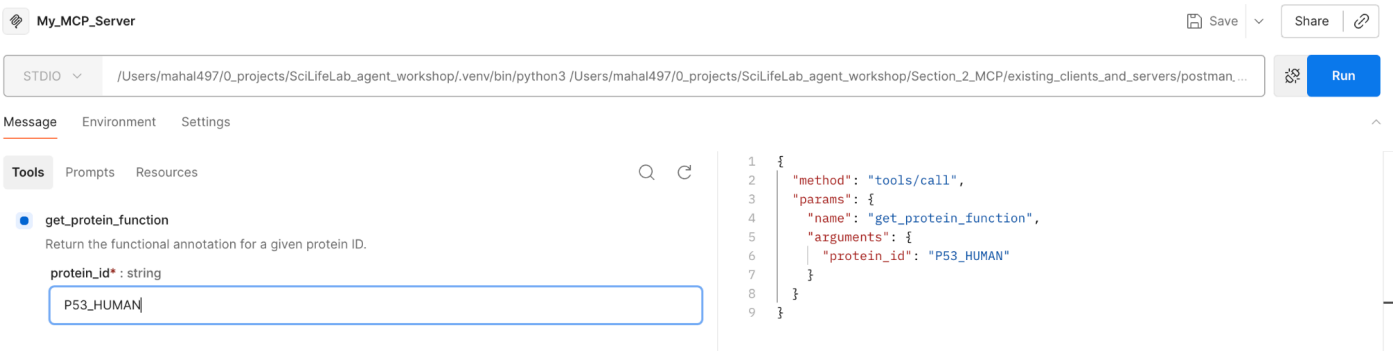
5. **Load Configuration:**
   - Find the configuration file in your workshop materials: `SciLifeLab_agent_workshop/Section_2_MCP/existing_clients_and_servers/postman_mcp_basic_server_config.json`
   - Open this file in a text editor (e.g., VS Code or Notepad).
   - Update `<ADD YOUR FULL PATH HERE>` to match your full local directory path.
   - **Copy its entire contents** to your clipboard.

6. **Paste and Connect:**
   - Return to Postman and paste the copied JSON content into the main configuration window (labeled `Enter command or paste JSON config`). Refer to the image under point **4** for guidance.
   - Click the **`Connect`** button.

7. **Grant Access:** Postman will show a security pop-up asking for permission to connect
   - Select **`This request only`**.
   - Click **`Grant Access`**.



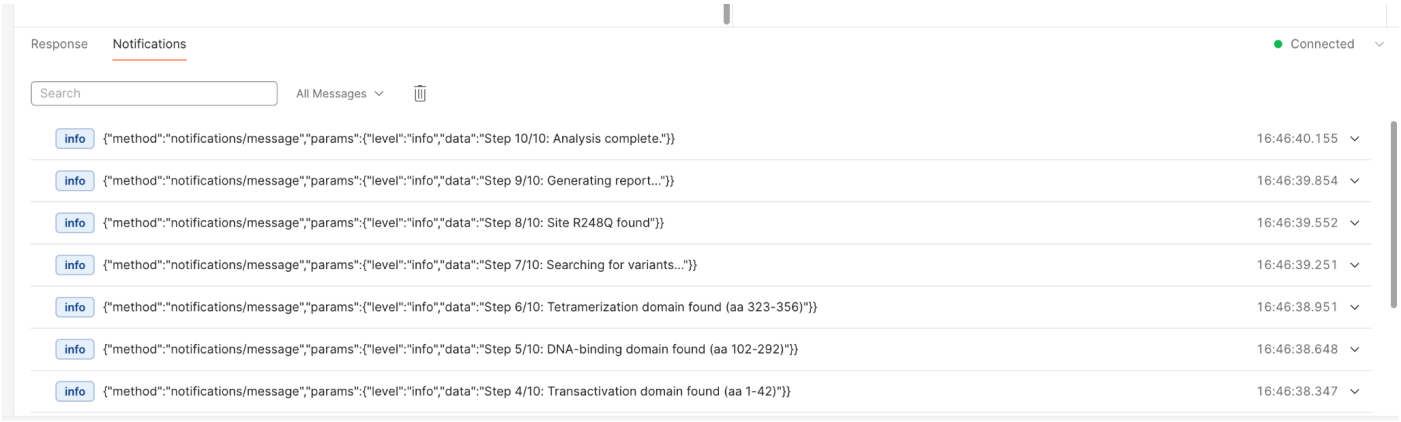
8. **Run a Tool:** You are now connected!
    - Select the **`Tools`** tab in Postman.
    - Click on the `get_protein_function` tool.
    - In the `protein_id*` (required) field, type `P53_HUMAN`.
    - Click the **`RUN`** button.



9. **Observe the Response:** Look at the **`Response`** panel at the bottom of the screen. You should see the JSON response from your server containing the protein's function!

10. **Disconnect:** Once you're done, click the **Disconnect** button at the top.

11. Now we'll use the `SciLifeLab_agent_workshop/Section_2_MCP/existing_clients_and_servers/postman_mcp_advanced_server.py` file for the advanced server. Repeat the process from step **one**, but in step **five**, use the `SciLifeLab_agent_workshop/Section_2_MCP/existing_clients_and_servers/postman_mcp_advanced_server_config.json` file instead.
12. **Explore!** You are now connected to the advanced server.
    - Click the **Tools** tab. Notice all the new tools: `find_protein`, `generate_hypothesis`, `stream_analysis_log`, etc.
    - Click the **Resources** tab. You'll see the resources defined on the server, like `dataset://proteins`.
    - **Try it:** Run the `find_protein` tool with the `protein_name` set to `p53`. Observe the "elicitation" response it returns, asking you to choose between human and mouse.



Workshop Complete

You've successfully used a real-world host application to connect to and interact with your custom MCP servers. This demonstrates how any MCP-compliant application (like an AI assistant) can discover and use the tools you build.

Happy learning!