Model Context Protocol (MCP) Design Notes

Purpose

This document provides a foundational understanding of the Model Context Protocol (MCP), detailing its architecture, core primitives (Resources, Tools, Prompts), and data flow. It serves as a technical reference for implementing agentic interactions within the application.

1. High-Level Concept

Definition: MCP is an open standard protocol that enables AI models to securely access external tools, data, and prompts. It acts as a universal translator between AI applications ("Clients") and external systems ("Servers"). **Goal:** `MCP Server → Context → AI`

2. Architecture

The architecture follows a Client-Host-Server model:

```
Host Application (e.g., IDE/Claude Desktop) [runs MCP Client]
<==[Transport]==> MCP Server
```

- **MCP Host/Agent:** The frontend application the user interacts with (e.g., Cursor, saved in `docs/`).
- **MCP Client:** The internal component in the Host that speaks the MCP protocol (JSON-RPC 2.0).
- **Transport Layer:**
  - **Stdio:** Standard Input/Output (common for local processes).
  - **SSE (Server-Sent Events):** HTTP streaming (common for remote servers).
- **MCP Server:** A lightweight process that exposes specific capabilities (Resources, Tools, Prompts).

3. Core Primitives

The protocol defines three main types of capabilities a server can expose:

A. Resources (Data)

Passive reading of information. - **Analogy:** Like a file system or GET request. - **Use Case:** Reading logs, fetching database schema, viewing file contents. - **Key Operations:** `resources/list`, `resources/read`. - **Updates:** Clients can subscribe to resource updates.

B. Tools (Actions)

Executable functions that can have side effects. - **Analogy:** Function calling or POST request. - **Use Case:** Executing code, creating database entries, searching via API. - **Key Operations:** `tools/list`, `tools/call`.

C. Prompts (Context)

Reusable templates to guide the LLM. - **Analogy:** Saved prompt templates or "slash commands". - **Use Case:** "Analyze this code", "Generate unit tests". - **Key Operations:** `prompts/list`, `prompts/get`.

4. Advanced Capabilities & Flows

Client <-> Server Data Link

1. **Initialization:**
   - Client connects.
   - **Handshake:** Client and Server exchange `initialize` messages to negotiate protocol version and capabilities.
2. **Discovery:**
   - Client requests lists of Tools, Resources, and Prompts.
3. **Execution/Retrieval:**
   - User asks a question.
   - Host (Agent) determines which tool/resource is needed.
   - Client sends request to Server.
   - Server executes and returns result (Text or Image).

Server Features

- **Logging:** Servers can emit log messages (`notifications/message`) to the Client console (Debug, Info, Error).
- **Notifications:** Real-time signals that resources have changed.

Client Responsibilities (Host features)

- **Sampling:** The *Server* can ask the *Client* to generate a completion using the Host's LLM. This allows lightweight servers to utilize the powerful model of the Host.
- **Roots:** The Client informs the Server which directories/files are in scope (e.g., "This is the current project root: `/app`"). This is crucial for security and context.

5. Summary Diagram

```
[User]
```

```
[Host Application (AI Agent)]
   (Decides to use a tool)

[MCP Client]   (JSON-RPC over Stdio/SSE)    [MCP Server]


                                              [External System]
                                              (DB, API, Files)
```

Original Hand-Written Notes (Reference)

```
Model Context Protocol (MCP)

* Definition: MCP Server  Context to AI.
* Architecture Diagram Flow:
* Agent (MCP Host)  MCP Client  [Connection]  MCP Server.


* Primitives: Tools, Resources, Prompts.
* Key Operations:
* List: Discovery.
* Get: Retrieval.
* Sampling: Language model completion.
* Elicitation: Additional information.
* Logging: Server logs.


* Datalink Flow:
* Initialize  Tool Discovery (Primitives)  Tool Execution (Primitives)  Real-time Update (No


* Server Flow:
1. User invokes a prompt with parameters.
2. User selects resources to include.
3. AI processes the request using tools.


* Client Responsibilities:
* Elicitation: Request specific info.
* Roots: Which directory to focus on.
* Sampling: Request LLM completion through client.
```