MCP Dev Summit Bengaluru

Beyond Tools and Resources:
A Deep Dive into MCP Sampling for Agentic Features

SMARTBEAR

Kevin Rohan Vaz

Senior Software Engineer @ SmartBear

AGENDA

- Problem Statement
- Introduction to Sampling
- Walkthrough of Sampling Capabilities & Internals
- Sampling Patterns Demo
- Challenges in adopting Sampling
- Q&A

Problem Statement

“Trying to build AI powered MCP tools?”

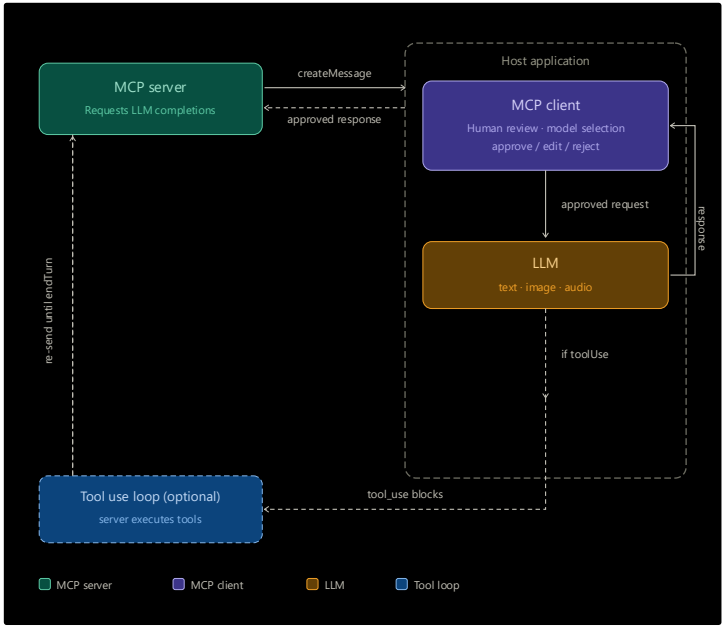
- **API key sprawl** Every server needs its own LLM credentials, billing, and rate limit management - multiplied across every tool you build.
- **No user control** LLM calls happen silently inside the server. Users can't see, edit, or reject what gets sent to the model.

"Every MCP server that needs AI has to solve the same problem - where does the LLM come from?"

“Sampling solves this by letting the server borrow the client's LLM — with the user always in control.”

Introduction to Sampling

- Client-side LLM access via MCP
- No LLM needed on the server - Build AI features without server-side models.
- Server-initiated requests - Server asks the client to call the LLM.
- Multimodal input support - Works with text, images, and audio.



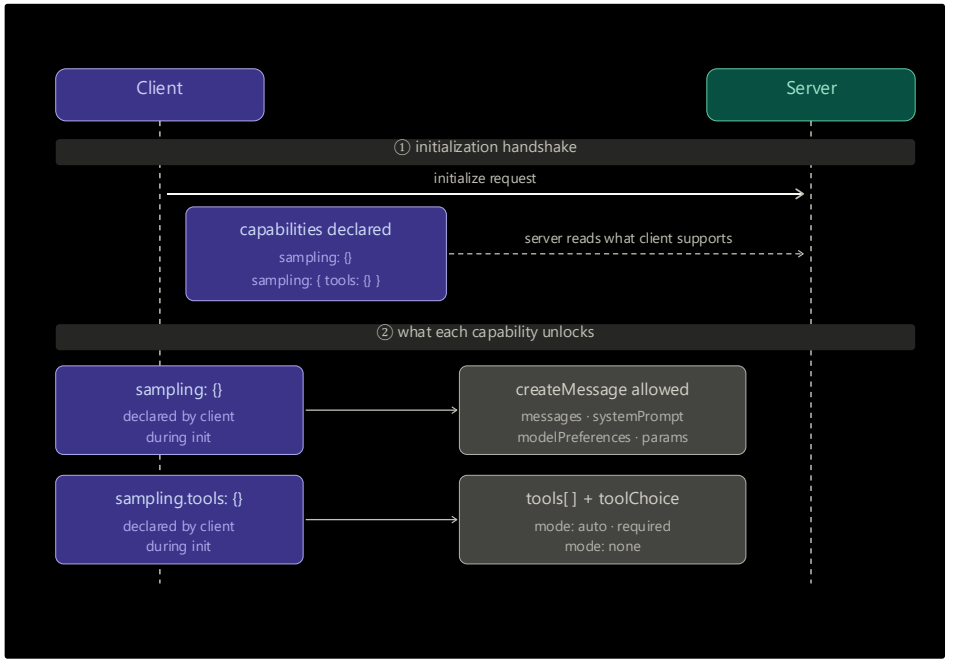
Sampling Capabilities

- **Capability priorities**
  - **Cost** - How important is minimizing costs?
  - **Speed** - How important is low latency?
  - **Intelligence** - How important are advanced capabilities?
- **Model Selection** – specify preferred models for the feature
- **Model Inference parameters** – temperature & max tokens
- **Tool Use for building agents & parallel tool execution**

```
{
  "hints": [
      { "name": "claude-3-sonnet" },  // Prefer Sonnet-class models
      { "name": "claude" }             // Fall back to any Claude model
  ],
  "costPriority": 0.3,                // Cost is less important
  "speedPriority": 0.8,               // Speed is very important
  "intelligencePriority": 0.5         // Moderate capability needs
}
```

Sampling Internals – Client-Side flow

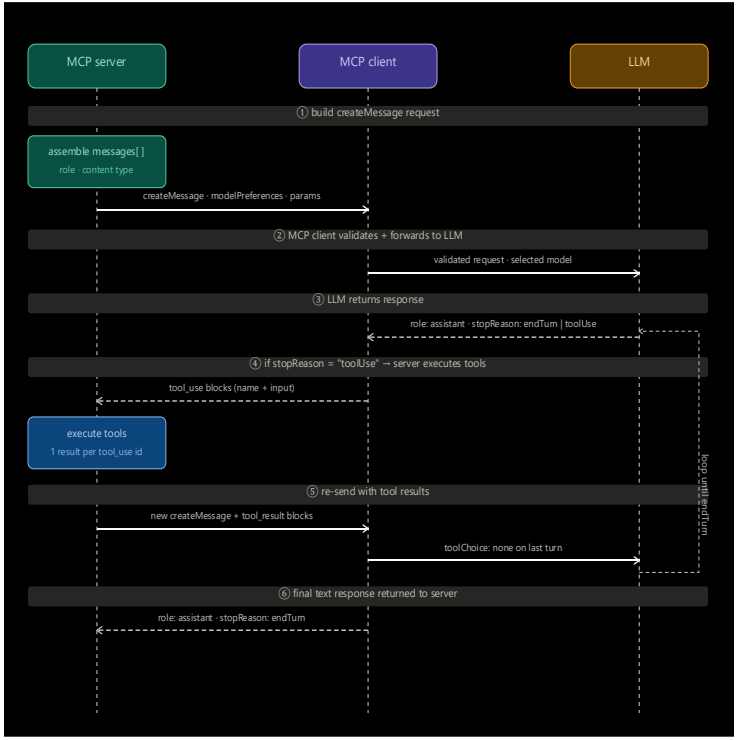
- **MCP Client bridges the server ↔ LLM** – Forwards requests to accessible models.
- **MCP Client declares sampling support** – Sends sampling capability jsonrpc message on server initialization.



Sampling Internals – Server-Side flow

- Tool trigger sends jsonrpc to client – Client handles the sampling request.
- Server owns tool execution – Manages tools and the agent loop.

```
{
  ...
  "method": "sampling/createMessage",
  "params": {
    "messages": [
      {
        "role": "user",
        "content": { "type": "text", "text": "Is it going to rain in Bengaluru?" }
      }
    ],
    "tools": [ { "name": "get_weather", "description": "...", "inputSchema": {...} } ],
    "systemPrompt": "You are a helpful weather assistant",
    "modelPreferences": {
      "hints": [{ "name": "claude-3-sonnet" }],
      "speedPriority": 0.8,
      "intelligencePriority": 0.5
    },
    "maxTokens": 300
  }
}
```



Sampling Patterns - Demo

- **Workflow (Single Turn flow)** –
  Example use cases:-
  - Summarizing a pdf.
  - Explaining errors in a screenshot.
  - Generating new data for missing inputs.
- **Agentic (Multi Turn flow)** –
  Example use cases:-
  - Solving questions which requires using multiple tools like ”Who is the most recent Turing award winner in the field of AI?”.
  - Setting up contract testing in a new project till tests are passing and published.
- **Hybrid (Mix of Workflows and Agents)**

DEMO

Challenges in adopting Sampling

- **Limited host app support** - Not all clients implement sampling.
- **No access to host tools or agents** - Must implement operations independently.
- **Human-in-the-loop adds latency** - Approval steps slow down automated flows.
- **No guaranteed model availability** - Client picks the model, not the server.
- **Capability gaps across clients** - Tool support varies by implementation.
- **Complex multi-turn state management** - Server must track full message history.
- **No standardised error handling** - Rejection and failure behaviour differs.

Q&A

QR code to slides and code examples

MCP Dev Summit Bengaluru