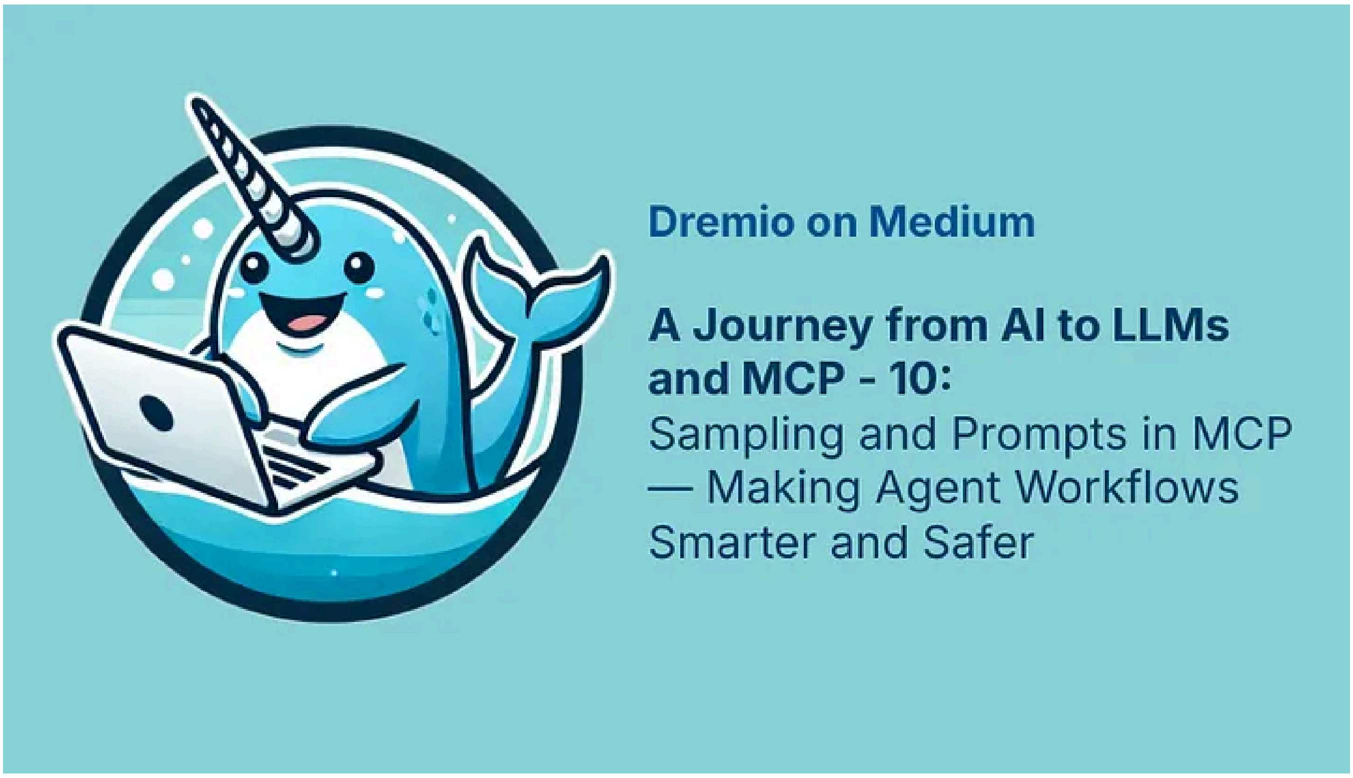
Data, Analytics & A... · Follow publication

A Journey from AI to LLMs and MCP — 10 — Sampling and Prompts in MCP — Making Agent Workflows Smarter and Safer

4 min read · May 20, 2025

Alex Merced Follow

Listen Share More


Dremio on Medium
A Journey from AI to LLMs and MCP - 10:
Sampling and Prompts in MCP — Making Agent Workflows Smarter and Safer

Free Resources

- Free Apache Iceberg Course
- Free Copy of "Apache Iceberg: The Definitive Guide"
- 2025 Apache Iceberg Architecture Guide

- How to Join the Iceberg Community
- Iceberg Lakehouse Engineering Video Playlist
- Ultimate Apache Iceberg Resource Guide

We've now seen how the Model Context Protocol (MCP) allows LLMs to read resources and call tools — giving them access to both data and action.

But what if your MCP server needs the LLM to make a decision?

What if it needs to:

- Analyze a file before running a tool?
- Draft a message for approval?
- Ask the model to choose between options?

That's where Sampling comes in.

And what if you want to give the user — or the LLM — reusable, structured prompt templates for common workflows?

That's where Prompts come in.

In this final post of the series, we'll explore:

- How sampling allows servers to request completions from LLMs
- How prompts enable reusable, guided AI interactions
- Best practices for both features
- Real-world use cases that combine everything we've covered so far

What Is Sampling in MCP?

Sampling is the ability for an MCP server to ask the host to run an LLM completion — on behalf of a tool, prompt, or workflow.

It lets your server say:

> *"Hey, LLM, here's a prompt and some context. Please respond."*

Why is this useful?

- You can generate intermediate reasoning steps
- Let the model propose actions before executing them
- Create more natural multi-turn agent workflows
- Maintain human-in-the-loop approval and visibility

Sampling Flow

Here’s the typical lifecycle:

1. The server sends a `sampling/createMessage` request
2. The host (Claude Desktop, etc.) can review or modify the prompt
3. The host runs the LLM completion
4. The result is sent back to the server

> *This architecture puts control and visibility in the hands of the user, even when the agent logic runs server-side.*

Message Format

Here’s an example `sampling/createMessage` request:

```
{
  "messages": [
    {
      "role": "user",
      "content": {
        "type": "text",
        "text": "Please summarize this log file."
      }
    }
  ],
  "systemPrompt": "You are a helpful developer assistant.",
  "includeContext": "thisServer",
  "maxTokens": 300
}
```

The host chooses which model to use, what context to include, and whether to show the prompt to the user for confirmation.

Response:

```
{
  "model": "claude-3-sonnet",
  "role": "assistant",
  "content": {
    "type": "text",
    "text": "The log file contains several timeout errors and warnings related
  }
}
```

Now the server can act on that response — log it, return it as tool output, or chain it into another step.

Best Practices for Sampling

Best Practice Why It Matters

- Use clear system prompts Guides model behavior contextually
- Limit tokens Prevent runaway completions
- Structure responses Enables downstream parsing (e.g. JSON, bullets)
- Include only relevant context Keep prompts focused and cost-effective
- Respect user control The host mediates the actual LLM call

What Are Prompts in MCP?

Prompts are reusable, structured templates that servers can expose to clients.

Think of them like slash commands or predefined workflows:

- Pre-filled with helpful defaults
- Accept arguments (e.g. “project name”, “file path”)
- Optionally include embedded resources
- Surface in the client UI

Prompts help users and LLMs collaborate efficiently by standardizing useful tasks.

Prompt Structure

Prompts have:

- A name (identifier)
- A description (for discovery)
- A list of arguments (optional)
- A template for generating messages

Example:

```
{
  "name": "explain-code",
  "description": "Explain how this code works",
  "arguments": [
    {
      "name": "language",
      "description": "Programming language",
      "required": true
    },
    {
      "name": "code",
      "description": "The code to analyze",
      "required": true
    }
  ]
}
```

Clients use:

- `prompts/list` to discover prompts
- `prompts/get` to resolve a prompt and arguments into messages

Dynamic Prompt Example

A server might expose:

```
{
  "name": "analyze-logs",
  "description": "Summarize recent logs and detect anomalies",
  "arguments": [
    {
      "name": "timeframe",
      "required": true
    }
  ]
}
```

When the user (or LLM) runs it with:

```
{
  "timeframe": "1h"
}
```

The resolved prompt could include:

- A message like: `“Please summarize the following logs from the past hour.”`
- An embedded resource (e.g. `logs://recent?timeframe=1h`)
- Output ready for sampling

Sampling + Prompts = Dynamic Workflows

Combining prompts + sampling + tools unlocks real agent behavior.

Example Workflow:

- **User selects prompt:** “Analyze logs and suggest next steps”
- Server resolves the prompt and calls sampling/createMessage
- **LLM returns:** “The logs show repeated auth failures. Suggest checking OAuth config.”
- Server calls tools/call to run check_auth_config
- LLM reviews the result and writes a summary

All controlled via:

- Standardized MCP messages
- User-visible approvals

- Modular server logic

Security and Control

Why These Matter for AI Agents

Wrapping It All Together

Over this 10-part series, we've explored the full landscape of AI agent development using MCP:

- LLMs and how they work
- Fine-tuning, prompting, and RAG
- Agent frameworks and limitations

- MCP's architecture and interoperability
- Resources and tools
- Prompts and sampling

MCP gives us standardized, modular building blocks for creating AI agents that are:

- Portable across environments
- Decoupled from model providers
- Secure, observable, and controlled

AI · Model Context Protocol · Machine Learning

Follow

Published in Data, Analytics & AI with Dremio

375 followers · Last published Oct 14, 2025

The Intelligent Lakehouse Platform

Follow

Written by Alex Merced

633 followers · 14 following

I'm a tech, development and data enthusiast who has a lot to say. You can find all my blogs, videos and podcasts at AlexMerced.com

No responses yet

Bgerby

What are your thoughts?

More from Alex Merced and Data, Analytics & AI with Dremio

In Data, Analytics & AI with Dremio by Alex Merced

Building a Basic MCP Server with Python

Free Resources

Apr 4 163 18

In Data, Analytics & AI with Dremio by Alex Merced

The State of Apache Iceberg v4 — October 2025 Edition

Get Data Lakehouse Books:

Oct 14 1

In Data, Analytics & AI with Dremio by Alex Merced

2025 Guide to Architecting an Iceberg Lakehouse

Blog: What is a Data Lakehouse and a Table Format?

Dec 10, 2024 96

Alex Merced

The 2025 & 2026 Ultimate Guide to the Data Lakehouse and the Data Lakehouse Ecosystem

Join the Data Lakehouse Community

Sep 23 👏 5

See all from Alex Merced

See all from Data, Analytics & AI with Dremio

Recommended from Medium

Riccardo Tartaglia

5 Essential MCP Servers Every Developer Should Know

I’ve been experimenting with Model Context Protocol servers for a few months now, and I have to say, they’ve changed the way I work.

Oct 11 👏 22 💬 3

Vandana Maurya

📚 Confusion Matrix — Complete Beginner to Advanced Guide

Here, I am sharing easy-to-read, beginner-friendly, complete note on Confusion Matrix you can go through just before your interview or…

Apr 29

In Towards AI by Ahmed Boulahia

Best Open-Source Embedding Models for RAG

High-Performance Open-Source Embedding Models for RAG Pipelines, Multilingual NLP, and Arabic Text

✦ Oct 14 201

In Coding Nexus by Civil Learning

MarkItDown: Convert Anything into Markdown — the Smart Way to Feed LLMs

You know that feeling when you're trying to feed a PDF or a Word document into an LLM, and it just doesn't understand what's going on...

✦ 6d ago 👏 125 💬 1

Matteo Ferruccio Andreoni

How to Fine-Tune LLMs Locally: The Complete LoRA Guide

Master LoRA Fine-Tuning on Apple Silicon with minimal memory usage. From setup to deployment in one comprehensive tutorial.

3d ago 👏 87

Alex Suzuki

Cretaceous Software Engineering — Why I don’t use AI to write code.

👋 Hi! I’m a Cretaceous Software Engineer.

5d ago 👏 641 💬 28

See more recommendations