Context Engineering: Optimizing MCP in Claude Code

A practical, example-driven guide for designing prompts and context that help Claude Code use MCP tools efficiently, accurately, and cost-effectively.

1. What is Context Engineering?

Context engineering is the practice of deliberately shaping what information is given to the LLM, when it is given, and how much is provided, so the model can make good decisions about tool usage.

Simple definition: Context engineering is designing the LLM's working memory.

2. Why Context Engineering Matters for MCP

- Claude decides whether and how to call MCP tools based on context.
- Too much context reduces reasoning quality.
- Too little context leads to wrong or missing tool calls.

3. What Counts as Context in Claude Code

- System and project instructions
- Recent conversation messages
- File contents shared with Claude
- MCP tool schemas and descriptions
- Previous MCP tool outputs

4. Before / After Example

Before (Poor Context Engineering)

```
Here are 5,000 lines of logs from today. Please analyze them and tell me which service is failing.
```

Problems: high token usage, slower responses, unclear tool usage.

After (Good Context Engineering)

```
We are investigating production errors. Summary: - Service A: normal - Service B: elevated 5xx errors (35%) - Service C: normal Use fetch_logs(service_name) if you need details.
```

Benefits: fast tool selection, lower cost, higher accuracy.

5. Common Context Engineering Anti-Patterns

- Dumping raw data directly into the prompt
- Making the LLM both reason and execute heavy work
- Overloading system instructions with too many rules
- Repeating the same context in every turn
- Using vague or overly generic MCP tool definitions

6. Best Practices for Optimizing MCP

- Give Claude enough context to decide, not to execute.
- Move bulk data retrieval into MCP tools.
- Pass references and summaries instead of raw payloads.
- Explicitly mention available tools when appropriate.
- Keep project instructions stable and concise.

7. A Simple Mental Model

```
Claude decides WHAT to do. MCP tools decide HOW to do it. Context decides HOW WELL
Claude chooses.
```

Key takeaway: Good context engineering makes MCP feel smart by helping Claude choose the right tool at the right time with the right information.