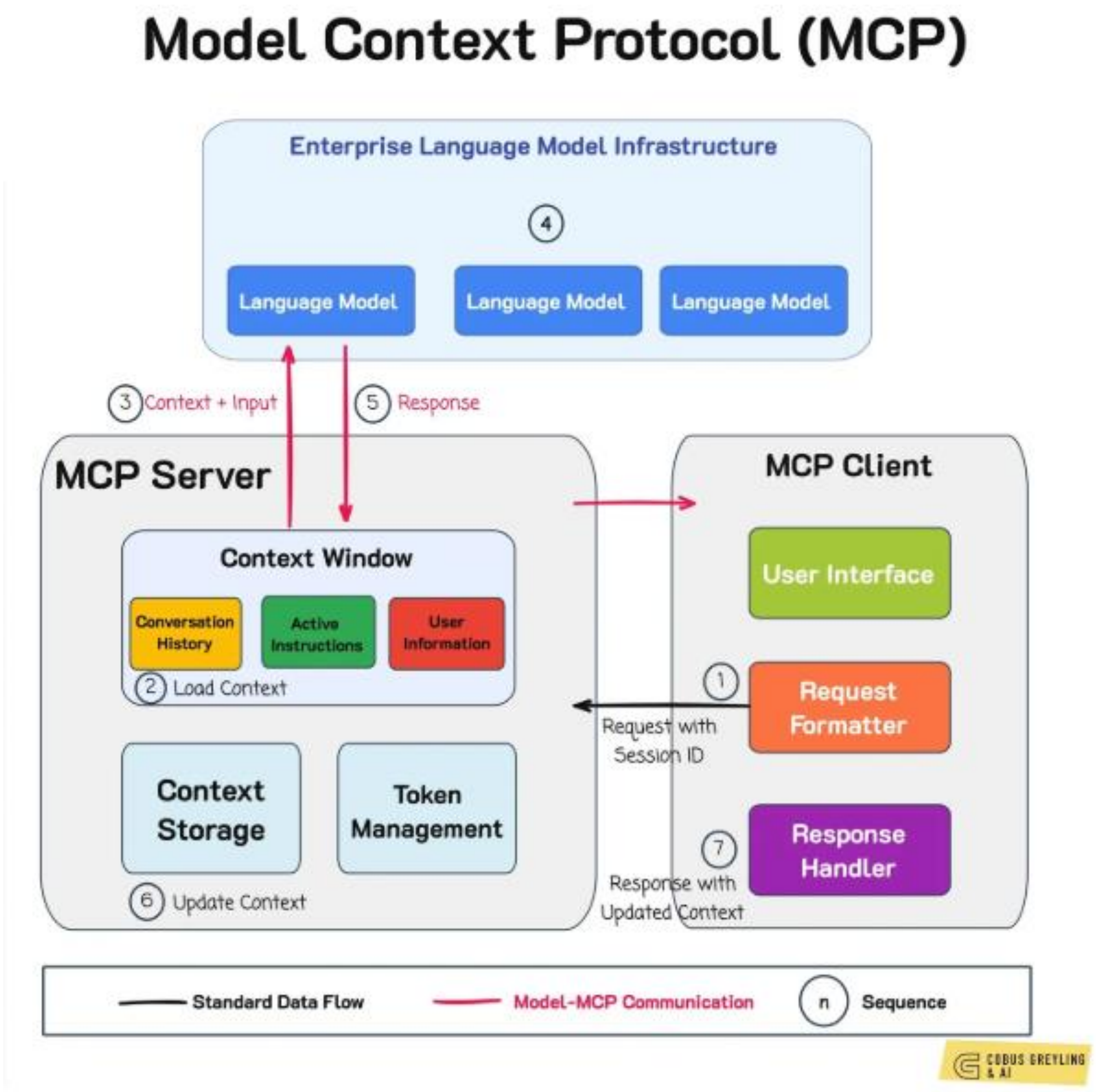
Model Context Protocol (MCP): The Universal Connector for AI Systems

What is MCP?



The picture above is taken from an on Medium by cobusgreyling

Think of Model Context Protocol (MCP) like a universal phone charger. Before USB-C, everyone had different chargers for different devices. It was confusing and inefficient. MCP does for AI what USB-C did for electronics - it creates a standard way for AI systems to connect and share information.

Why do we need it?

When companies use AI, they often have multiple AI systems that need to work together. Without a standard way to connect them:

- Information gets lost between systems
- Engineers waste time building custom connections
- Companies struggle to keep track of what information is being shared
- It's hard to enforce security and privacy rules

How does MCP work?

MCP creates a standard format for sharing "context" - the information AI systems need to do their job. It's like a standard envelope for sending information with clearly labeled sections:

- **The actual information** (like customer data or company guidelines)
- **Where the information came from** (like a database or user input)
- **Important details about the information** (like how recent it is or how sensitive it is)
- **Rules for handling the information** (like how long to keep it or who can see it)

How does it help in real systems?

Imagine a customer service AI system:

1. A customer asks a question about their bill
2. The first AI figures out what the customer wants (billing help)
3. A second AI finds relevant company policies about billing
4. A third AI writes a helpful response to the customer

Without MCP, each AI would need to start from scratch. With MCP:

- The first AI packages up what it learned about the customer's issue
- It sends this package to the second AI using the standard MCP format
- The second AI adds information about relevant policies to the package
- It sends the enhanced package to the third AI
- The third AI has all the context it needs to write a helpful response

Benefits for companies

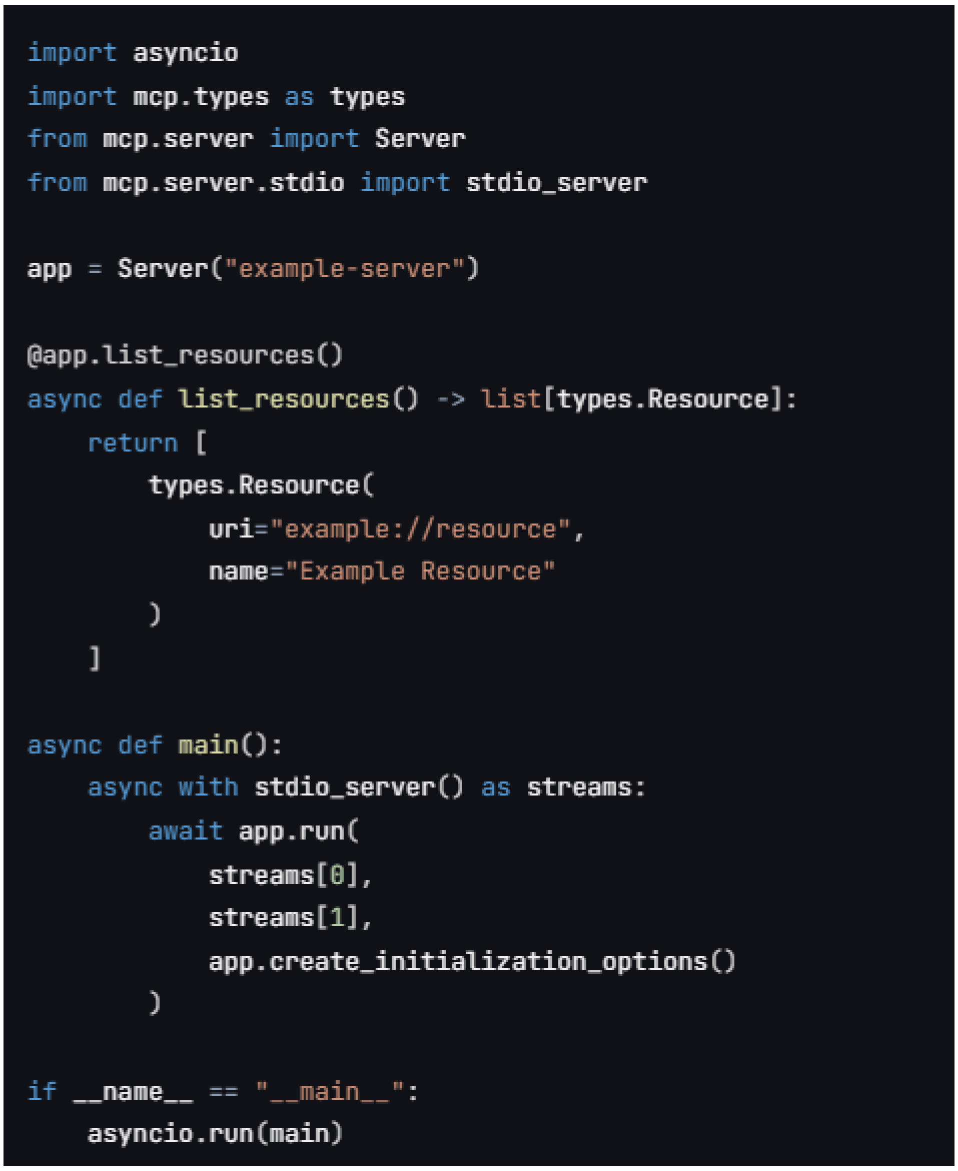
Companies using MCP see several benefits:

- **Faster development**: Engineers don't need to build custom connections
- **Better performance**: Systems don't waste time reprocessing information
- **Easier compliance**: There's a clear record of what information was shared
- **Better customer experience**: AI systems have the context they need to be helpful

In simple terms

MCP is like a universal language that lets AI systems talk to each other effectively. It ensures that when information moves between systems, nothing important gets lost in translation. This makes AI systems work better together, which makes them more useful and reliable for businesses and customers.

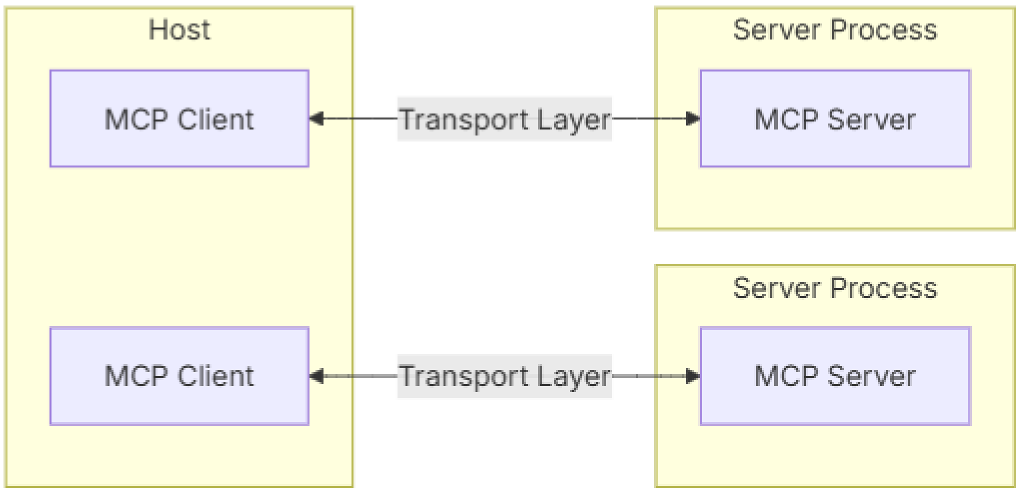
Here is a small implementation



```python
import asyncio
import mcp.types as types
from mcp.server import Server
from mcp.server.stdio import stdio_server

app = Server("example-server")

@app.list_resources()
async def list_resources() -> list[types.Resource]:
    return [
        types.Resource(
            uri="example://resource",
            name="Example Resource"
        )
    ]

async def main():
    async with stdio_server() as streams:
        await app.run(
            streams[0],
            streams[1],
            app.create_initialization_options()
        )

if __name__ == "__main__":
    asyncio.run(main)
```

Core Architecture

The Model Context Protocol (MCP) is built on a flexible, extensible architecture that enables seamless communication between LLM applications and integrations. This document covers the core architectural components and concepts



Core Components

Protocol Layer

The protocol layer handles message framing, request/response linking, and high-level communication patterns.

Key Classes include **Protocol, Client, Server**

Transport Layer

The transport layer handles the actual communication between clients and servers. MCP supports multiple transport mechanisms including **Stdio Transport and HTTP with SSE Transport.**

All transports use JSON-RPC 2.0 to exchange messages