Model context protocol (MCP)

The Model context protocol (aka MCP) is a way to provide tools and context to the LLM. From the MCP docs:

> MCP is an open protocol that standardizes how applications provide context to LLMs. Think of MCP like a USB-C port for AI applications. Just as USB-C provides a standardized way to connect your devices to various peripherals and accessories, MCP provides a standardized way to connect AI models to different data sources and tools.

The Agents SDK has support for MCP. This enables you to use a wide range of MCP servers to provide tools to your Agents.

MCP servers

Currently, the MCP spec defines two kinds of servers, based on the transport mechanism they use:

1. **stdio** servers run as a subprocess of your application. You can think of them as running "locally".
2. **HTTP over SSE** servers run remotely. You connect to them via a URL.

You can use the `MCPServerStdio` and `MCPServerSse` classes to connect to these servers.

For example, this is how you'd use the official MCP filesystem server.

```
async with MCPServerStdio(
    params={
        "command": "npx",
        "args": ["-y", "@modelcontextprotocol/server-filesystem", samples_dir],
    }
) as server:
    tools = await server.list_tools()
```

Using MCP servers

MCP servers can be added to Agents. The Agents SDK will call `list_tools()` on the MCP servers each time the Agent is run. This makes the LLM aware of the MCP server's tools. When the LLM calls a tool from an MCP server, the SDK calls `call_tool()` on that server.

```
agent=Agent(
    name="Assistant",
    instructions="Use the tools to achieve the task",
```

```
    mcp_servers=[mcp_server_1, mcp_server_2]
)
```

Caching

Every time an Agent runs, it calls `list_tools()` on the MCP server. This can be a latency hit, especially if the server is a remote server. To automatically cache the list of tools, you can pass `cache_tools_list=True` to both `MCPServerStdio` and `MCPServerSse`. You should only do this if you're certain the tool list will not change.

If you want to invalidate the cache, you can call `invalidate_tools_cache()` on the servers.

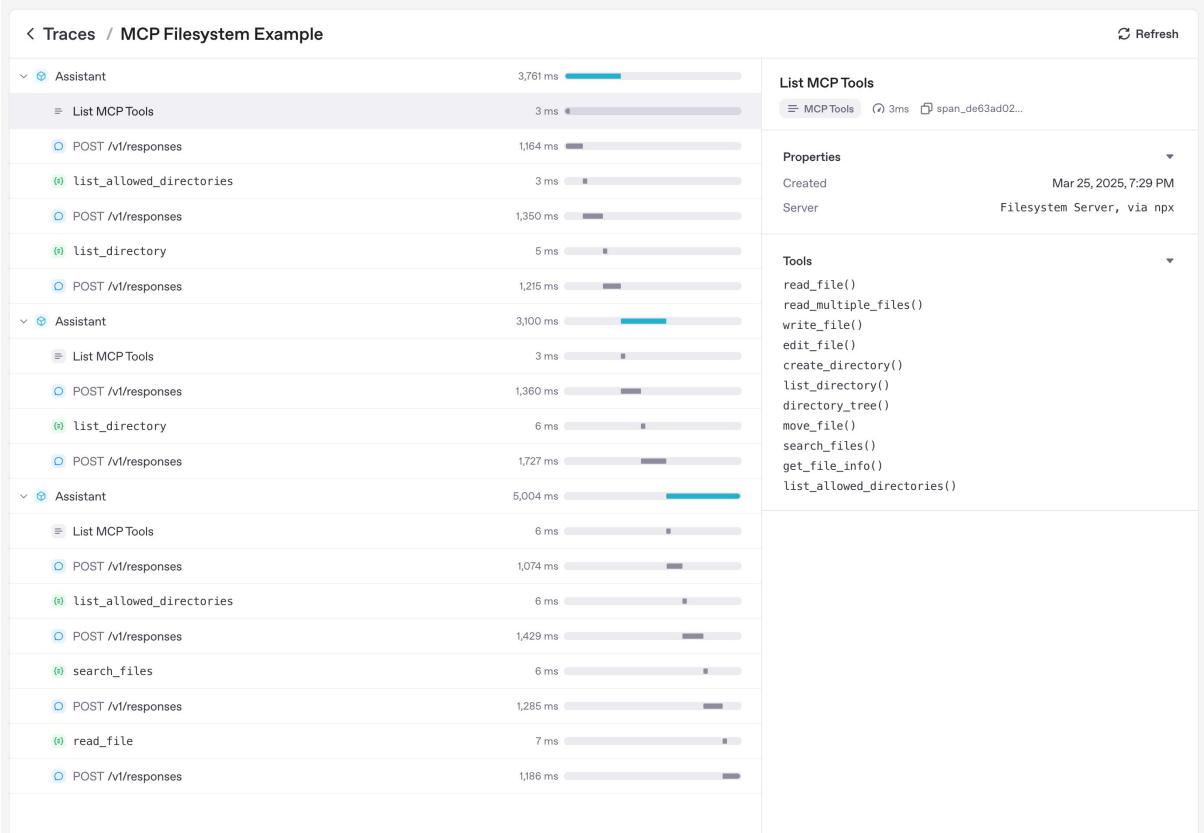
End-to-end examples

View complete working examples at examples/mcp.

Tracing

Tracing automatically captures MCP operations, including:

1. Calls to the MCP server to list tools
2. MCP-related info on function calls


Traces / MCP Filesystem Example
Refresh
Assistant 3,761 ms
List MCP Tools 3 ms
POST /v1/responses 1,164 ms
list_allowed_directories 3 ms
POST /v1/responses 1,350 ms
list_directory 5 ms
POST /v1/responses 1,215 ms
Assistant 3,100 ms
List MCP Tools 3 ms
POST /v1/responses 1,360 ms
list_directory 6 ms
POST /v1/responses 1,727 ms
Assistant 5,004 ms
List MCP Tools 6 ms
POST /v1/responses 1,074 ms
list_allowed_directories 6 ms
POST /v1/responses 1,429 ms
search_files 6 ms
POST /v1/responses 1,285 ms
read_file 7 ms
POST /v1/responses 1,186 ms
List MCP Tools
MCP Tools 3ms span_de63ad02...
Properties
Created Mar 25, 2025, 7:29 PM
Server Filesystem Server, via npx
Tools
read_file()
read_multiple_files()
write_file()
edit_file()
create_directory()
list_directory()
directory_tree()
move_file()
search_files()
get_file_info()
list_allowed_directories()