Page: https://modelcontextprotocol.io/introduction

Get Started

Tutorials

Concepts

Development

Introduction

Get started with the Model Context Protocol (MCP)

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.

Page: https://modelcontextprotocol.io/

Get Started

Tutorials

Concepts

Development

Introduction

Get started with the Model Context Protocol (MCP)

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.

Page: https://modelcontextprotocol.io/tutorials/building-a-client

Get Started

Tutorials

Concepts

Development

Building MCP clients

Learn how to build your first client in MCP

Page: https://modelcontextprotocol.io/quickstart

Quickstart

Get started with building your first MCP server and connecting it to a host

Page: https://modelcontextprotocol.io/docs/tools/inspector

Inspector

In-depth guide to using the MCP Inspector for testing and debugging Model Context Protocol servers

Page: https://modelcontextprotocol.io/docs/concepts/prompts

Prompts

Create reusable prompt templates and workflows

Prompts enable servers to define reusable prompt templates and workflows that clients can easily surface to users and LLMs. They provide a powerful way to standardize and share common LLM interactions.

Prompts are designed to be user-controlled, meaning they are exposed from servers to clients with the intention of the user being able to explicitly select them for use.

Page: https://modelcontextprotocol.io/tutorials/building-mcp-with-llms

Building MCP with LLMs

Speed up your MCP development using LLMs such as Claude!

Page: https://modelcontextprotocol.io/docs/concepts/transports

Transports

Page: https://modelcontextprotocol.io/docs/concepts/sampling

Sampling

Let your servers request completions from LLMs

Sampling is a powerful MCP feature that allows servers to request LLM completions through the client, enabling sophisticated agentic behaviors while maintaining security and privacy.

This feature of MCP is not yet supported in the Claude Desktop client.

Page: https://modelcontextprotocol.io/clients

Clients

A list of applications that support MCP integrations

This page provides an overview of applications that support the Model Context Protocol (MCP). Each client may support different MCP features, allowing for varying levels of integration with MCP servers.

Page: https://modelcontextprotocol.io/development/contributing

Contributing

How to participate in Model Context Protocol development

We welcome contributions from the community! Please review our contributing guidelines for details on how to submit changes.

All contributors must adhere to our Code of Conduct.

For questions and discussions, please use GitHub Discussions.

Was this page helpful?

Page: https://modelcontextprotocol.io/docs/concepts/tools

Tools

Enable LLMs to perform actions through your server

Tools are a powerful primitive in the Model Context Protocol (MCP) that enable servers to expose executable functionality to clients. Through tools, LLMs can interact with external systems, perform computations, and take actions in the real world.

Tools are designed to be model-controlled, meaning that tools are exposed from servers to clients with the intention of the AI model being able to automatically invoke them (with a human in the loop to grant approval).

Page: https://modelcontextprotocol.io/examples

Examples

A list of example servers and implementations

Page: https://modelcontextprotocol.io/docs/concepts/roots

Roots

Understanding roots in MCP

Roots are a concept in MCP that define the boundaries where servers can operate. They provide a way for clients to inform servers about relevant resources and their locations.

Page: https://modelcontextprotocol.io/building-mcp-with-llms

Introduction

Get started with the Model Context Protocol (MCP)

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.

Page: https://modelcontextprotocol.io/docs/concepts/architecture

Core architecture

Understand how MCP connects clients, servers, and LLMs

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.

Page: https://modelcontextprotocol.io/docs/concepts/resources

Resources

Expose data and content from your servers to LLMs

Resources are a core primitive in the Model Context Protocol (MCP) that allow servers to expose data and content that can be read by clients and used as context for LLM interactions.

Resources are designed to be application-controlled, meaning that the client application can decide how and when they should be used.

Different MCP clients may handle resources differently. For example:

Server authors should be prepared to handle any of these interaction patterns when implementing resource support. In order to expose data to models automatically, server authors should use a model-controlled primitive such as Tools.

Page: https://modelcontextprotocol.io/docs/tools/debugging

Get Started

Tutorials

Concepts

Development

Debugging

A comprehensive guide to debugging Model Context Protocol (MCP) integrations

Effective debugging is essential when developing MCP servers or integrating them with applications. This guide covers the debugging tools and approaches available in the MCP ecosystem.

This guide is for macOS. Guides for other platforms are coming soon.

Page: https://modelcontextprotocol.io/development/roadmap

Get Started

Tutorials

Concepts

Development

Roadmap

Our plans for evolving Model Context Protocol (H1 2025)

The Model Context Protocol is rapidly evolving. This page outlines our current thinking on key priorities and future direction for the first half of 2025, though these may change significantly as the project develops.

We encourage community participation! Each section links to relevant discussions where you can learn more and contribute your thoughts.