Specification

ℹ️ **Protocol Revision**: 2024-11-05

Model Context Protocol (MCP) is an open protocol that enables seamless integration between LLM applications and external data sources and tools. Whether you're building an AI-powered IDE, enhancing a chat interface, or creating custom AI workflows, MCP provides a standardized way to connect LLMs with the context they need.

This specification defines the authoritative protocol requirements, based on the TypeScript schema in schema.ts.

For implementation guides and examples, visit modelcontextprotocol.io.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Overview

MCP provides a standardized way for applications to:

- Share contextual information with language models
- Expose tools and capabilities to AI systems
- Build composable integrations and workflows

The protocol uses JSON-RPC 2.0 messages to establish communication between:

- **Hosts**: LLM applications that initiate connections
- **Clients**: Connectors within the host application
- **Servers**: Services that provide context and capabilities

MCP takes some inspiration from the Language Server Protocol, which standardizes how to add support for programming languages across a whole ecosystem of development tools. In a similar way, MCP standardizes how to integrate additional context and tools into the ecosystem of AI applications.

Key Details

Base Protocol

- JSON-RPC message format
- Stateful connections
- Server and client capability negotiation

Features

Servers offer any of the following features to clients:

- **Resources**: Context and data, for the user or the AI model to use
- **Prompts**: Templated messages and workflows for users
- **Tools**: Functions for the AI model to execute

Clients may offer the following feature to servers:

- **Sampling**: Server-initiated agentic behaviors and recursive LLM interactions

Additional Utilities

- Configuration
- Progress tracking
- Cancellation
- Error reporting
- Logging

Security and Trust & Safety

The Model Context Protocol enables powerful capabilities through arbitrary data access and code execution paths. With this power comes important security and trust considerations that all implementors must carefully address.

Key Principles

1. **User Consent and Control**
   - Users must explicitly consent to and understand all data access and operations
   - Users must retain control over what data is shared and what actions are taken
   - Implementors should provide clear UIs for reviewing and authorizing activities
2. **Data Privacy**
   - Hosts must obtain explicit user consent before exposing user data to servers
   - Hosts must not transmit resource data elsewhere without user consent
   - User data should be protected with appropriate access controls
3. **Tool Safety**
   - Tools represent arbitrary code execution and must be treated with appropriate caution
   - Hosts must obtain explicit user consent before invoking any tool
   - Users should understand what each tool does before authorizing its use
4. **LLM Sampling Controls**
   - Users must explicitly approve any LLM sampling requests
   - Users should control:
     - Whether sampling occurs at all
     - The actual prompt that will be sent
     - What results the server can see
   - The protocol intentionally limits server visibility into prompts

Implementation Guidelines

While MCP itself cannot enforce these security principles at the protocol level, implementors **SHOULD**:

1. Build robust consent and authorization flows into their applications
2. Provide clear documentation of security implications
3. Implement appropriate access controls and data protections
4. Follow security best practices in their integrations
5. Consider privacy implications in their feature designs

Learn More

Explore the detailed specification for each protocol component:

- Architecture
- Base Protocol
- Server Features
- Client Features
- Contributing