Model Context Protocol (MCP) Message Types

This document summarizes the request and response message types defined in the Model Context Protocol schema.

Core Message Structure

All MCP messages follow the JSON-RPC 2.0 specification with these basic structures:

- **Request**: Contains `jsonrpc`, `method`, `params`, and `id` fields
- **Response**: Contains `jsonrpc`, `result`, and `id` fields
- **Error Response**: Contains `jsonrpc`, `error`, and `id` fields
- **Notification**: Contains `jsonrpc` and `method` fields (no `id` as no response is expected)

Client Requests

Method	Description	Key Parameters
`initialize`	First request sent by client to server	`capabilities`, `clientInfo`, `protocolVersion`
`ping`	Check if server is alive	None
`resources/list`	Request list of available resources	`cursor` (optional)
`resources/templates/list`	Request list of resource templates	`cursor` (optional)
`resources/read`	Read a specific resource	`uri`
`resources/subscribe`	Subscribe to resource updates	`uri`
`resources/unsubscribe`	Unsubscribe from resource updates	`uri`
`prompts/list`	Request list of available prompts	`cursor` (optional)
`prompts/get`	Get a specific prompt	`name`, `arguments` (optional)
`tools/list`	Request list of available tools	`cursor` (optional)
`tools/call`	Call a specific tool	`name`, `arguments` (optional)
`logging/setLevel`	Set logging level	`level`
`completion/complete`	Request completion options	`argument`, `ref`

Server Requests

Method	Description	Key Parameters
`ping`	Check if client is alive	None
`sampling/createMessage`	Request LLM sampling via client	`messages`, `maxTokens`, various optional parameters
`roots/list`	Request list of root URIs from client	None

Client Notifications

Method	Description	Key Parameters
`notifications/initialized`	Sent after initialization completes	None
`notifications/cancelled`	Cancel a previous request	`requestId`, `reason` (optional)
`notifications/progress`	Progress update for long-running operation	`progress`, `progressToken`, `total` (optional)
`notifications/roots/list_changed`	Inform server that roots list changed	None

Server Notifications

Method	Description	Key Parameters
`notifications/cancelled`	Cancel a previous request	`requestId`, `reason` (optional)
`notifications/progress`	Progress update for long-running operation	`progress`, `progressToken`, `total` (optional)
`notifications/resources/list_changed`	Resource list has changed	None
`notifications/resources/updated`	A resource has been updated	`uri`
`notifications/prompts/list_changed`	Prompt list has changed	None
`notifications/tools/list_changed`	Tool list has changed	None
`notifications/message`	Log message	`data`, `level`, `logger` (optional)

Response Types

Request Method	Response Result Type	Key Fields
`initialize`	`InitializeResult`	`capabilities`, `protocolVersion`, `serverInfo`, `instructions` (optional)
`resources/list`	`ListResourcesResult`	`resources`, `nextCursor` (optional)
`resources/templates/list`	`ListResourceTemplatesResult`	`resourceTemplates`, `nextCursor` (optional)
`resources/read`	`ReadResourceResult`	`contents`
`prompts/list`	`ListPromptsResult`	`prompts`, `nextCursor` (optional)
`prompts/get`	`GetPromptResult`	`messages`, `description` (optional)
`tools/list`	`ListToolsResult`	`tools`, `nextCursor` (optional)
`tools/call`	`CallToolResult`	`content`, `isError` (optional)
`completion/complete`	`CompleteResult`	`completion` (contains `values`, `hasMore`, `total`)
`sampling/createMessage`	`CreateMessageResult`	`content`, `model`, `role`, `stopReason` (optional)
`roots/list`	`ListRootsResult`	`roots`

Content Types

The protocol supports several content types that can be exchanged:

- **TextContent**: Plain text with optional annotations
- **ImageContent**: Base64-encoded image data with MIME type
- **EmbeddedResource**: Resource content embedded in a message
- **TextResourceContents**: Text-based resource content
- **BlobResourceContents**: Binary resource content (base64-encoded)

Error Handling

Error responses follow the JSON-RPC 2.0 specification with these standard error codes:

- `-32700`: Parse error
- `-32600`: Invalid request
- `-32601`: Method not found

- `-32602`: Invalid params
- `-32603`: Internal error

Additional application-specific error codes may be defined by implementations.

Appendix: Request and Response Examples

Client to Server Examples

Initialize

Request:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "initialize",
  "params": {
    "capabilities": {
      "sampling": {},
      "roots": {
        "listChanged": true
      }
    },
    "clientInfo": {
      "name": "ExampleClient",
      "version": "1.0.0"
    },
    "protocolVersion": "0.1.0"
  }
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "result": {
    "capabilities": {
      "resources": {
        "listChanged": true,
        "subscribe": true
      },
      "prompts": {
        "listChanged": true
      },
      "tools": {
        "listChanged": true
      },
      "logging": {}
    },
    "serverInfo": {
      "name": "ExampleServer",
      "version": "1.0.0"
    },
    "protocolVersion": "0.1.0",
    "instructions": "This server provides access to project resources and
tools."
  }
}
```

Ping

Request:

```
{
  "jsonrpc": "2.0",
  "id": 2,
  "method": "ping",
  "params": {}
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "id": 2,
  "result": {}
}
```

Resources/List

Request:

```
{
  "jsonrpc": "2.0",
  "id": 3,
  "method": "resources/list",
  "params": {
    "cursor": "optional-pagination-token"
  }
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "id": 3,
  "result": {
    "resources": [
      {
        "name": "Project README",
        "uri": "file:///project/README.md",
        "mimeType": "text/markdown",
        "description": "Project overview and documentation"
      },
      {
        "name": "Configuration",
        "uri": "file:///project/config.json",
        "mimeType": "application/json",
        "description": "Project configuration file"
      }
    ],
    "nextCursor": "next-page-token"
  }
}
```

Resources/Templates/List

Request:

```
{
  "jsonrpc": "2.0",
  "id": 4,
  "method": "resources/templates/list",
  "params": {}
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "id": 4,
  "result": {
    "resourceTemplates": [
      {
        "name": "Source Files",
        "uriTemplate": "file:///project/src/{filename}.{ext}",
        "description": "Source code files in the project"
      },
      {
        "name": "Test Files",
        "uriTemplate": "file:///project/tests/{filename}_test.{ext}",
        "description": "Test files for the project"
      }
    ]
  }
}
```

Resources/Read

Request:

```
  {
    "jsonrpc": "2.0",
    "id": 5,
    "method": "resources/read",
    "params": {
      "uri": "file:///project/README.md"
    }
  }
```

Response:

```
  {
    "jsonrpc": "2.0",
    "id": 5,
    "result": {
      "contents": [
        {
          "uri": "file:///project/README.md",
          "mimeType": "text/markdown",
          "text": "# Project Title\n\nThis is a sample project README file."
        }
      ]
    }
  }
```

Resources/Subscribe

Request:

```
  {
    "jsonrpc": "2.0",
    "id": 6,
    "method": "resources/subscribe",
    "params": {
      "uri": "file:///project/config.json"
  }
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "id": 6,
  "result": {}
}
```

Resources/Unsubscribe

Request:

```
{
  "jsonrpc": "2.0",
  "id": 7,
  "method": "resources/unsubscribe",
  "params": {
    "uri": "file:///project/config.json"
  }
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "id": 7,
  "result": {}
}
```

Prompts/List

Request:

```
{
  "jsonrpc": "2.0",
  "id": 8,
  "method": "prompts/list",
  "params": {}
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "id": 8,
  "result": {
    "prompts": [
      {
        "name": "code-review",
        "description": "Review code for issues and improvements",
        "arguments": [
          {
            "name": "language",
            "description": "Programming language",
            "required": true
          },
          {
            "name": "code",
            "description": "Code to review",
            "required": true
          }
        ]
      },
      {
        "name": "explain-code",
        "description": "Explain what code does",
        "arguments": [
          {
            "name": "code",
            "description": "Code to explain",
            "required": true
          }
        ]
      }
    ]
  }
}
```

Prompts/Get

Request:

```
{
  "jsonrpc": "2.0",
  "id": 9,
  "method": "prompts/get",
  "params": {
    "name": "code-review",
    "arguments": {
      "language": "python",
      "code": "def hello():\n    print('Hello, world!')"
    }
  }
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "id": 9,
  "result": {
    "description": "Code review for Python code",
    "messages": [
      {
        "role": "user",
        "content": {
          "type": "text",
          "text": "Please review this Python code:\n\ndef hello():\n
print('Hello, world!')"
        }
      }
    ]
  }
}
```

Tools/List

Request:

```
{
  "jsonrpc": "2.0",
  "id": 10,
  "method": "tools/list",
  "params": {}
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "id": 10,
  "result": {
    "tools": [
      {
        "name": "search-code",
        "description": "Search for code patterns in the project",
        "inputSchema": {
          "type": "object",
          "properties": {
            "pattern": {
              "type": "string"
            },
            "fileType": {
              "type": "string"
            }
          },
          "required": ["pattern"]
        }
      },
      {
        "name": "run-tests",
        "description": "Run project tests",
        "inputSchema": {
          "type": "object",
          "properties": {
            "testPath": {
              "type": "string"
            }
          }
        }
      }
    ]
  }
}
```

Tools/Call

Request:

```
{
  "jsonrpc": "2.0",
  "id": 11,
  "method": "tools/call",
  "params": {
    "name": "search-code",
    "arguments": {
      "pattern": "function",
      "fileType": "js"
    }
  }
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "id": 11,
  "result": {
    "content": [
      {
        "type": "text",
        "text": "Found 3 matches in 2 files:\n- src/main.js:10: function
initialize()\n- src/main.js:25: function cleanup()\n- src/utils.js:5:
function formatDate(date)"
      }
    ]
  }
}
```

Logging/SetLevel

Request:

```
{
  "jsonrpc": "2.0",
  "id": 12,
  "method": "logging/setLevel",
  "params": {
    "level": "info"
  }
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "id": 12,
  "result": {}
}
```

Completion/Complete

Request:

```
{
  "jsonrpc": "2.0",
  "id": 13,
  "method": "completion/complete",
  "params": {
    "argument": {
      "name": "language",
      "value": "py"
    },
    "ref": {
      "type": "ref/prompt",
      "name": "code-review"
    }
  }
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "id": 13,
  "result": {
    "completion": {
      "values": ["python", "python3"],
      "hasMore": false,
      "total": 2
    }
  }
}
```

Server to Client Examples

Ping

Request:

```
{
  "jsonrpc": "2.0",
  "id": 14,
  "method": "ping",
  "params": {}
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "id": 14,
  "result": {}
}
```

Sampling/CreateMessage

Request:

```
{
  "jsonrpc": "2.0",
  "id": 15,
  "method": "sampling/createMessage",
  "params": {
    "messages": [
      {
        "role": "user",
        "content": {
          "type": "text",
          "text": "What is the purpose of this code?"
        }
      },
      {
        "role": "assistant",
        "content": {
          "type": "text",
          "text": "This code appears to be a simple function that prints a
greeting."
        }
      },
      {
        "role": "user",
        "content": {
          "type": "text",
          "text": "Can you suggest improvements?"
        }
      }
    ],
    "maxTokens": 1000,
    "temperature": 0.7,
    "systemPrompt": "You are a helpful coding assistant."
  }
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "id": 15,
  "result": {
    "content": {
      "type": "text",
      "text": "Here are some improvements for the code:\n\n1. Add a
docstring to explain what the function does\n2. Make the greeting
customizable by accepting a name parameter\n3. Consider returning the
greeting instead of printing it directly\n\nHere's an improved
version:\n\n```python\ndef hello(name=\"World\"):\n    \"\"\"Return a
```

```
personalized greeting message.\"\"\"\n    return f\"Hello, {name}!\"\n\n#
To use it:\nprint(hello())  # Prints: Hello, World!\nprint(hello(\"User\"))
# Prints: Hello, User!\n```\n\nThis makes the function more flexible and
reusable."
    },
    "model": "gpt-4-turbo",
    "role": "assistant",
    "stopReason": "stop"
  }
}
```

Roots/List

Request:

```
{
  "jsonrpc": "2.0",
  "id": 16,
  "method": "roots/list",
  "params": {}
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "id": 16,
  "result": {
    "roots": [
      {
        "name": "Main Project",
        "uri": "file:///project"
      },
      {
        "name": "Documentation",
        "uri": "file:///docs"
      }
    ]
  }
}
```