sphinx-mcp

Release 0.1.0a1

Anirban Basu

Jul 10, 2025

CONTENTS

`sphinx-mcp` is a Sphinx[1] documentation extension for Model Context Protocol (MCP) servers. In this documentation, you will find information on how to install, configure, and use the extension to document MCP servers. You will also find examples of how to use the extension with different MCP servers, such as pymcp[2] and @modelcontextprotocol/server-everything[3].

[1] https://www.sphinx-doc.org/
[2] https://github.com/anirbanbasu/pymcp
[3] https://github.com/modelcontextprotocol/servers/tree/main/src/everything

CHAPTER

1

USAGE

This section covers the installation, configuration and usage of the `sphinx-mcp` extension for Sphinx.

1.1 Installation

You can install `sphinx-mcp` using pip as `pip install sphinx-mcp`. Alternatively, you can add it to your existing project a project management tool, such as uv[4], by calling `uv add sphinx-mcp`. Once installed, you can use it in your Sphinx project by adding it to the `extensions` list in your Sphinx `conf.py` file and add the MCP servers configuration, as shown below.

1.2 Configuration

To document only one MCP server, you can set the `allow_only_one_mcp_server` configuration option to `True` (it defaults to `False` if not specified).

```
extensions = [
    "sphinx_mcp",
]

mcp_config = {
   "mcpServers": {
      "pymcp": {
         "transport": "stdio",
         "command": "python",
         "args": ["-m", "pymcp.server"],
      }
   }
}

allow_only_one_mcp_server = True
```

To document multiple MCP servers, you can specify them in the `mcpServers` dictionary. Remember to set the `allow_only_one_mcp_server` to `False` or not set it at all, as it defaults to `False`. In the fol-

[4] https://docs.astral.sh/uv/

lowing example, two MCP servers are configured: one using the `pymcp` library and another using the `@modelcontextprotocol/server-everything` package, which is not even implemented in Python!

```
extensions = [
    "sphinx_mcp",
]

mcp_config = {
   "mcpServers": {
      "pymcp": {
         # Local stdio server
         "transport": "stdio",
         "command": "python",
         "args": ["-m", "pymcp.server"],
      },
      "everything": {
         # Make sure to run `nvm use --lts` before running this.
         "command": "npx",
         "args": ["-y", "@modelcontextprotocol/server-everything"],
      }
   }
}
```

In both the single and multiple server configurations, only `stdio` transport is illustrated. However, servers with `sse` and `streamable-http` transports can also be configured. For `stdio` transport, the command necessary to run the server must execute successfully as Sphinx will use it to connect to the server. This necessitates having the necessary runtime environment set up, such as having the correct Python version or Node.js version installed.

1.3 Directives

The extension provides the following directives to document MCP servers. Each directive belongs to the domain `mcpdocs`.

`mcpdocs::tools`
: This directive is used to document the tools available in one or more MCP servers. An optional argument can be provided to specify the MCP server to document. If no argument is provided, it defaults to all the servers configured in the `mcp_config` dictionary. If the argument is absent, each tool listed will be prefixed with the server name, followed by two colons, e.g., `pymcp::greet`.

  Arguments:

  - server name (optional)

`mcpdocs::prompts`
: This directive is used to document the prompts available in one or more MCP servers. Similar to the tools directive, an optional argument can be provided to specify the MCP server to document. If no argument is provided, it defaults to all the servers configured in the `mcp_config` dictionary. If the argument is absent, each prompt listed will be prefixed with the server name, followed by two colons, e.g., `pymcp::code_prompt`.

  Arguments:

  - server name (optional)

`mcpdocs::resources`
: This directive is used to document the resources available in one or more MCP servers. An optional argument can be provided to specify the MCP server to document. If no argument is provided, it defaults to all the servers configured in the `mcp_config` dictionary. If the argument

is absent, each resource listed will be prefixed with the server name, followed by two colons, e.g., `pymcp::resource_logo`.

Arguments:

- server name (optional)

`mcpdocs::resource_templates`

This directive is used to document the resource templates available in one or more MCP servers. An optional argument can be provided to specify the MCP server to document. If no argument is provided, it defaults to all the servers configured in the `mcp_config` dictionary. If the argument is absent, each resource template listed will be prefixed with the server name, followed by two colons, e.g., `pymcp::resource_unicode_modulo10`.

Arguments:

- server name (optional)

1.4 Limitations

The extension currently only has the aforementioned directives. It does not contain any roles yet. In addition, the generation of indices for the documented tools, prompts, resources, and resource templates is not implemented yet.

CHAPTER

2

EXAMPLE MCP SERVER: `pymcp`

`pymcp` is an open-source template repository for developing MCP servers in Python, using FastMCP[5]. The project is available on GitHub at pymcp[6].

2.1 Tools

1. **greet**: Greet the caller with a quintessential Hello World message.

   Input schema:

```
{
  "properties": {
    "name": {
      "anyOf": [
        {
          "type": "string"
        },
        {
          "type": "null"
        }
      ],
      "default": null,
      "description": "The optional name to be greeted.",
      "title": "Name"
    }
  },
  "type": "object"
}
```

   Output schema:

```
{
  "properties": {
```

(continues on next page)

[5] http://gofastmcp.com/
[6] https://github.com/anirbanbasu/pymcp

(continued from previous page)

```
    "result": {
      "title": "Result",
      "type": "string"
    }
  },
  "required": [
    "result"
  ],
  "title": "_WrappedResult",
  "type": "object",
  "x-fastmcp-wrap-result": true
}
```

Annotations:

```
{
  "title": null,
  "readOnlyHint": true,
  "destructiveHint": null,
  "idempotentHint": null,
  "openWorldHint": null
}
```

2. **generate_password**: Generate a random password with specified length, optionally including special characters. The password will meet the complexity requirements of at least one lowercase letter, one uppercase letter, and two digits. If special characters are included, it will also contain at least one such character. Until the password meets these requirements, it will keep regenerating. This is a simple example of a tool that can be used to generate passwords. It is not intended for production use.

   Input schema:

```
{
  "properties": {
    "length": {
      "default": 12,
      "description": "The length of the password to generate (between 8 and 64␣
↪characters).",
      "maximum": 64,
      "minimum": 8,
      "title": "Length",
      "type": "integer"
    },
    "use_special_chars": {
      "default": false,
      "description": "Include special characters in the password.",
      "title": "Use Special Chars",
      "type": "boolean"
    }
  },
  "type": "object"
}
```

   Output schema:

```
{
  "properties": {
```

(continues on next page)

(continued from previous page)


```
    "result": {
      "title": "Result",
      "type": "string"
    }
  },
  "required": [
    "result"
  ],
  "title": "_WrappedResult",
  "type": "object",
  "x-fastmcp-wrap-result": true
}
```


Annotations:


```
{
  "title": null,
  "readOnlyHint": true,
  "destructiveHint": null,
  "idempotentHint": null,
  "openWorldHint": null
}
```


3. **permutations**: Calculate the number of ways to choose k items from n items without repetition and with order. If k is not provided, it defaults to n.

   Input schema:


```
{
  "properties": {
    "n": {
      "description": "The number of items to choose from.",
      "minimum": 1,
      "title": "N",
      "type": "integer"
    },
    "k": {
      "anyOf": [
        {
          "minimum": 1,
          "type": "integer"
        },
        {
          "type": "null"
        }
      ],
      "default": null,
      "description": "The optional number of items to choose.",
      "title": "K"
    }
  },
  "required": [
    "n"
  ],
  "type": "object"
}
```


   Output schema:

```
{
  "properties": {
    "result": {
      "title": "Result",
      "type": "integer"
    }
  },
  "required": [
    "result"
  ],
  "title": "_WrappedResult",
  "type": "object",
  "x-fastmcp-wrap-result": true
}
```


Annotations:


```
{
  "title": null,
  "readOnlyHint": true,
  "destructiveHint": null,
  "idempotentHint": null,
  "openWorldHint": null
}
```


4. **pirate_summary**: Summarise the given text in a pirate style. This is an example of a tool that can use LLM sampling to generate a summary.

   Input schema:


```
{
  "properties": {
    "text": {
      "title": "Text",
      "type": "string"
    }
  },
  "required": [
    "text"
  ],
  "type": "object"
}
```


   Output schema:


```
{
  "properties": {
    "result": {
      "title": "Result",
      "type": "string"
    }
  },
  "required": [
    "result"
  ],
  "title": "_WrappedResult",
  "type": "object",
  "x-fastmcp-wrap-result": true
```


(continues on next page)

(continued from previous page)

```
}
```

5. **vonmises_random**: Generate a random number from the von Mises distribution. This is an example of a tool that uses elicitation to obtain the required parameter kappa (κ).

   Input schema:

```
{
  "properties": {
    "mu": {
      "description": "The mean angle mu (\u03bc), expressed in radians between 0␣
↪and 2\u03c0",
      "maximum": 6.283185307179586,
      "minimum": 0,
      "title": "Mu",
      "type": "number"
    }
  },
  "required": [
    "mu"
  ],
  "type": "object"
}
```

   Output schema:

```
{
  "properties": {
    "result": {
      "title": "Result",
      "type": "number"
    }
  },
  "required": [
    "result"
  ],
  "title": "_WrappedResult",
  "type": "object",
  "x-fastmcp-wrap-result": true
}
```

2.2 Prompts

1. **code_prompt**: Get a prompt to write a code snippet in Python based on the specified task.

   Input arguments:

```
[
  {
    "name": "task",
    "description": null,
    "required": true
  }
]
```

2.3 Resources

1. **resource_logo** (data://logo) [text/plain]: Get the base64 encoded PNG logo of PyMCP.
2. **resource_logo_svg** (data://logo_svg) [image/svg+xml]: Get the PyMCP logo as SVG.

2.4 Resource templates

1. **resource_unicode_modulo10**: Computes the modulus 10 of a given number and returns a Unicode character representing the result. The character is chosen based on whether the modulus is odd or even: - For odd modulus, it uses the Unicode character starting from ❶ (U+2776). - For even modulus, it uses the Unicode character starting from ① (U+2460). - If the modulus is 0, it returns the circled zero character ⓪ (U+24EA).

CHAPTER

3

EXAMPLE MCP SERVER: EVERYTHING

`everything` is a Model Context Protocol (MCP) server that attempts to exercise all the features of the MCP protocol. It is not intended to be a useful server, but rather a test server for builders of MCP clients. It implements prompts, tools, resources, sampling, and more to showcase MCP capabilities.

Source: everything[7] as part of the official MCP examples on GitHub.

3.1 Tools

1. **echo**: Echoes back the input

   Input schema:

   ```
   {
     "type": "object",
     "properties": {
       "message": {
         "type": "string",
         "description": "Message to echo"
       }
     },
     "required": [
       "message"
     ],
     "additionalProperties": false,
     "$schema": "http://json-schema.org/draft-07/schema#"
   }
   ```

   Output schema:

   ```
   null
   ```

2. **add**: Adds two numbers

   Input schema:

[7] https://github.com/modelcontextprotocol/servers/tree/main/src/everything

```
{
  "type": "object",
  "properties": {
    "a": {
      "type": "number",
      "description": "First number"
    },
    "b": {
      "type": "number",
      "description": "Second number"
    }
  },
  "required": [
    "a",
    "b"
  ],
  "additionalProperties": false,
  "$schema": "http://json-schema.org/draft-07/schema#"
}
```

Output schema:

```
null
```

3. **printEnv**: Prints all environment variables, helpful for debugging MCP server configuration

   Input schema:

```
{
  "type": "object",
  "properties": {},
  "additionalProperties": false,
  "$schema": "http://json-schema.org/draft-07/schema#"
}
```

   Output schema:

```
null
```

4. **longRunningOperation**: Demonstrates a long running operation with progress updates

   Input schema:

```
{
  "type": "object",
  "properties": {
    "duration": {
      "type": "number",
      "default": 10,
      "description": "Duration of the operation in seconds"
    },
    "steps": {
      "type": "number",
      "default": 5,
      "description": "Number of steps in the operation"
    }
  },
  "additionalProperties": false,
```

(continues on next page)

(continued from previous page)

```
  "$schema": "http://json-schema.org/draft-07/schema#"
}
```

Output schema:

```
null
```

5. **sampleLLM**: Samples from an LLM using MCP's sampling feature

   Input schema:

```
{
  "type": "object",
  "properties": {
    "prompt": {
      "type": "string",
      "description": "The prompt to send to the LLM"
    },
    "maxTokens": {
      "type": "number",
      "default": 100,
      "description": "Maximum number of tokens to generate"
    }
  },
  "required": [
    "prompt"
  ],
  "additionalProperties": false,
  "$schema": "http://json-schema.org/draft-07/schema#"
}
```

   Output schema:

```
null
```

6. **getTinyImage**: Returns the MCP_TINY_IMAGE

   Input schema:

```
{
  "type": "object",
  "properties": {},
  "additionalProperties": false,
  "$schema": "http://json-schema.org/draft-07/schema#"
}
```

   Output schema:

```
null
```

7. **annotatedMessage**: Demonstrates how annotations can be used to provide metadata about content

   Input schema:

```
{
  "type": "object",
  "properties": {
    "messageType": {
```

(continues on next page)

(continued from previous page)

```
      "type": "string",
      "enum": [
        "error",
        "success",
        "debug"
      ],
      "description": "Type of message to demonstrate different annotation␣
↪patterns"
    },
    "includeImage": {
      "type": "boolean",
      "default": false,
      "description": "Whether to include an example image"
    }
  },
  "required": [
    "messageType"
  ],
  "additionalProperties": false,
  "$schema": "http://json-schema.org/draft-07/schema#"
}
```

Output schema:

```
null
```

8. **getResourceReference**: Returns a resource reference that can be used by MCP clients

   Input schema:

```
{
  "type": "object",
  "properties": {
    "resourceId": {
      "type": "number",
      "minimum": 1,
      "maximum": 100,
      "description": "ID of the resource to reference (1-100)"
    }
  },
  "required": [
    "resourceId"
  ],
  "additionalProperties": false,
  "$schema": "http://json-schema.org/draft-07/schema#"
}
```

   Output schema:

```
null
```

3.2 Prompts

1. **simple_prompt**: A prompt without arguments
2. **complex_prompt**: A prompt with arguments

Input arguments:

```
[
  {
    "name": "temperature",
    "description": "Temperature setting",
    "required": true
  },
  {
    "name": "style",
    "description": "Output style",
    "required": false
  }
]
```

3. **resource_prompt**: A prompt that includes an embedded resource reference

   Input arguments:

```
[
  {
    "name": "resourceId",
    "description": "Resource ID to include (1-100)",
    "required": true
  }
]
```

3.3 Resources

1. **Resource 1** (test://static/resource/1) [text/plain]
2. **Resource 2** (test://static/resource/2) [application/octet-stream]
3. **Resource 3** (test://static/resource/3) [text/plain]
4. **Resource 4** (test://static/resource/4) [application/octet-stream]
5. **Resource 5** (test://static/resource/5) [text/plain]
6. **Resource 6** (test://static/resource/6) [application/octet-stream]
7. **Resource 7** (test://static/resource/7) [text/plain]
8. **Resource 8** (test://static/resource/8) [application/octet-stream]
9. **Resource 9** (test://static/resource/9) [text/plain]
10. **Resource 10** (test://static/resource/10) [application/octet-stream]

3.4 Resource templates

1. **Static Resource**: A static resource with a numeric ID