

nerding-io / n8n-nodes-mcpPublic

n8n custom node for MCP

MIT license

2.1k stars372 forksBranchesTagsActivity

Star

Notifications

<> CodeIssues47Pull requests13ActionsProjectsSecurityInsights

main22 Branches22 Tags

Go to fileGo to fileCode...

actions-userBump version to 0.1.28 [skip ci]0358161 · last month

.github	init: test - unit	3 months ago
.vscode	init	4 months ago
__tests__	init: test - unit	3 months ago
assets	docs: readme - videos	3 months ago
credentials	feat: add HTTP Streamable transport, deprecate SSE, and up...	last month
nodes/McpClient	bug: Zod returning in Json schema (#124)	last month
.editorconfig	init	4 months ago
.eslintrc.js	bug: publish - lint i/o types	4 months ago
.eslintrc.prepublish.js	init	4 months ago
.gitignore	init	4 months ago
.npmignore	init: sse - creds, connection, ci/cd	3 months ago
.nvmrc	add: ai - tool support	3 months ago
.prettierrc.js	init	4 months ago
CODE_OF_CONDUCT.md	init	4 months ago
LICENSE.md	init	4 months ago
README.md	B/http streamable (#118)	last month
gulpfile.js	init	4 months ago
index.js	B/http streamable (#118)	last month
jest.config.js	init: test - unit	3 months ago
package-lock.json	Bump version to 0.1.28 [skip ci]	last month
package.json	Bump version to 0.1.28 [skip ci]	last month
pnpm-lock.yaml	build(deps): bump @modelcontextprotocol/sdk from 1.5.0 t...	2 months ago
tsconfig.json	init: test - unit	3 months ago

READMECode of conductMIT license

MseePverified

n8n-nodes-mcp-client

Important Note: The Server-Sent Events (SSE) transport is deprecated and replaced by the new HTTP Streamable transport. SSE remains available for legacy compatibility, but HTTP Streamable is now the recommended method for all new implementations.

This is an n8n community node that lets you interact with Model Context Protocol (MCP) servers in your n8n workflows.

MCP is a protocol that enables AI models to interact with external tools and data sources in a standardized way. This node allows you to connect to MCP servers, access resources, execute tools, and use prompts.

[n8n](#) is a [fair-code licensed](#) workflow automation platform.

[Installation](#) [Credentials](#) [Environment Variables](#) [Operations Using as a Tool](#) [Compatibility Resources](#)

Getting Started

Official Quickstart Video:

Community Videos

Shoutout to all the creators of the following n8n community videos that are great resources for learning how to use this node:

- [Is MCP the Future of N8N AI Agents? \(Fully Tested!\)](#)
- [Connect N8N AI Agents to EVERYTHING using MCP?](#)
- [Build an AI Agent That Can Use Any Tool \(MCP in n8n Tutorial\)](#)
- [The NEW N8N MCP is an Absolute Game-Changer \(Brave Search MCP\)](#)
- [MCP & n8n Automation: The Ultimate Guide for MCP AI Agents \(2025\)](#)
- [REVOLUÇÃO na criação de AGENTES no N8N com o MCP Server!!!](#) (Portuguese)

If you have a great video that you'd like to share, please let me know and I'll add it to the list!

Interested a deeper dive into MCP?

Check out my YouTube Series [MCP Explained](#) for more information about the Model Context Protocol.

Security Assessment



N N Nodes

by nerding-io

Service Connectors

MseeP.ai

Audited

Installation

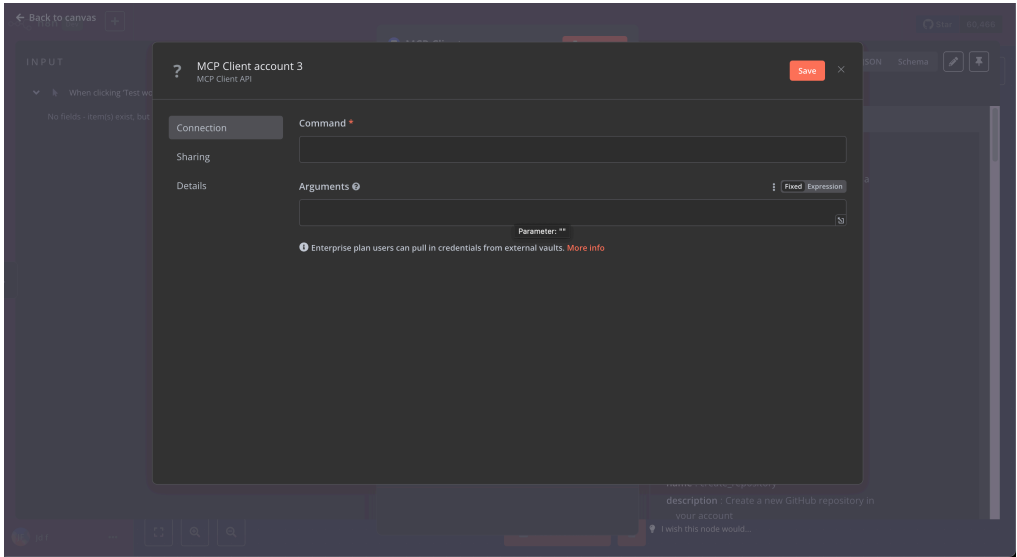
Follow the [installation guide](#) in the n8n community nodes documentation.

Also pay attention to Environment Variables for [using tools in AI Agents](#). It's mandatory to set the `N8N_COMMUNITY_PACKAGES_ALLOW_TOOL_USAGE` environment variable to `true` if you want to use the MCP Client node as a tool in AI Agents.

Credentials

The MCP Client node supports three types of credentials to connect to an MCP server:

Command-line Based Transport (STDIO)



- **Command:** The command to start the MCP server
- **Arguments:** Optional arguments to pass to the server command
- **Environment Variables:** Variables to pass to the server in NAME=VALUE format

HTTP Streamable Transport (Recommended)

- **HTTP Streamable URL:** The HTTP endpoint that supports streaming responses (e.g., <http://localhost:3001/stream>)
- **Additional Headers:** Optional headers to send with requests (format: name:value, one per line)

HTTP Streamable is the recommended and modern method for all new integrations, providing better efficiency and flexibility compared to SSE.

Example: Using a Local MCP Server with HTTP Streamable

This example shows how to connect to a locally running MCP server using HTTP Streamable:

1. Start a local MCP server that supports HTTP Streamable:

```
npx @modelcontextprotocol/server-example-streamable
```

2. Configure MCP Client credentials:

- In the node settings, select **Connection Type:** `HTTP Streamable`
- Create new credentials of type **MCP Client (HTTP Streamable) API**
- Set **HTTP Streamable URL:** `http://localhost:3001/stream`
- Add any required headers for authentication

3. Create a workflow using the MCP Client node:

- Add an MCP Client node
- Set the Connection Type to `HTTP Streamable`
- Select your HTTP Streamable credentials
- Execute the workflow to see the results

Server-Sent Events (SSE) Transport (Deprecated, still available for legacy use)

- **SSE URL:** The URL of the SSE endpoint (default: <http://localhost:3001/sse>)

- **Messages Post Endpoint:** Optional custom endpoint for posting messages if different from the SSE URL
- **Additional Headers:** Optional headers to send with requests (format: name:value, one per line)

Deprecated: SSE is deprecated and will not receive further updates, but remains available for legacy compatibility. For new projects, use HTTP Streamable.

Example: Using a Local MCP Server with SSE (legacy)

This example shows how to connect to a locally running MCP server using Server-Sent Events (SSE):

1. Start a local MCP server that supports SSE:

```
npx @modelcontextprotocol/server-example-sse
```



2. Configure MCP Client credentials:
 - In the node settings, select **Connection Type**: Server-Sent Events (SSE)
 - Create new credentials of type **MCP Client (SSE) API**
 - Set **SSE URL**: `http://localhost:3001/sse`
 - Add any required headers for authentication
3. Create a workflow using the MCP Client node:
 - Add an MCP Client node
 - Set the Connection Type to Server-Sent Events (SSE)
 - Select your SSE credentials
 - Execute the workflow to see the results

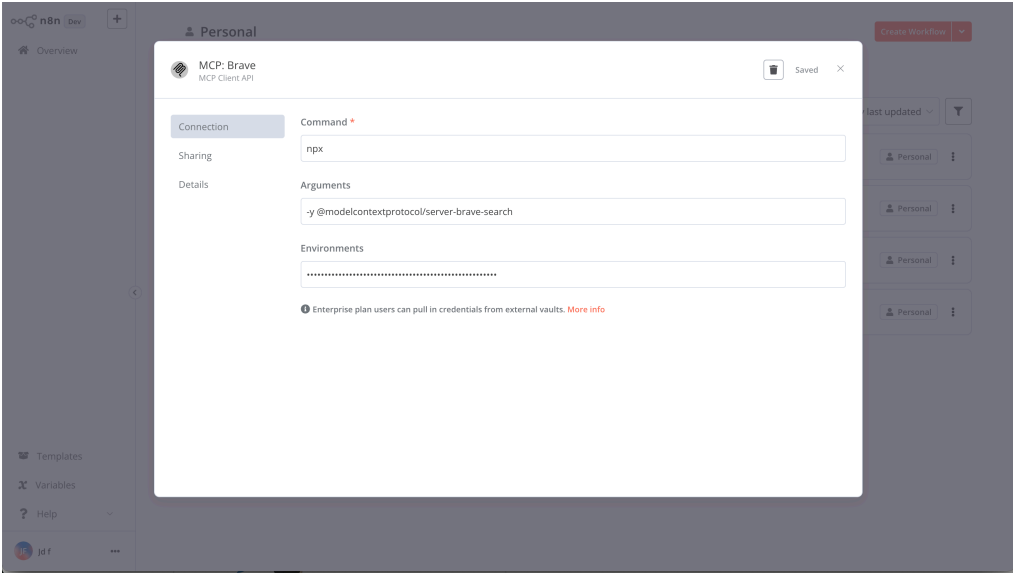
Note: For new projects, HTTP Streamable is strongly recommended.

Environment Variables

The MCP Client node supports passing environment variables to MCP servers using the command-line based transport in two ways:

1. Using the Credentials UI

You can add environment variables directly in the credentials configuration:



This method is useful for individual setups and testing. The values are stored securely as credentials in n8n.

2. Using Docker Environment Variables

For Docker deployments, you can pass environment variables directly to your MCP servers by prefixing them with `MCP_`:

```
version: '3'

services:
  n8n:
    image: n8nio/n8n
    environment:
      - MCP_BRAVE_API_KEY=your-api-key-here
      - MCP_OPENAI_API_KEY=your-openai-key-here
      - MCP_CUSTOM_SETTING=some-value
    # other configuration...
```



These environment variables will be automatically passed to your MCP servers when they are executed.

Example: Using Brave Search MCP Server

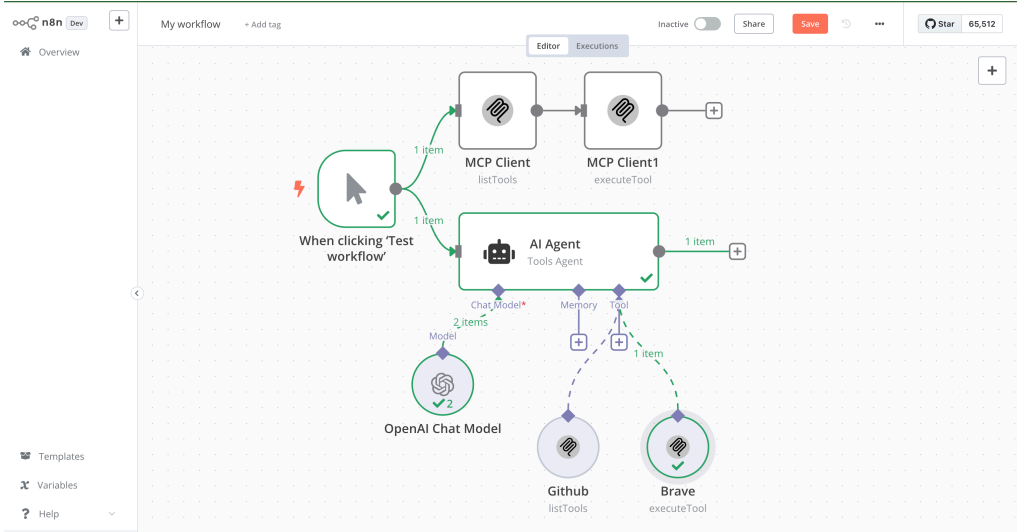
This example shows how to set up and use the Brave Search MCP server:

1. Install the Brave Search MCP server:

```
npm install -g @modelcontextprotocol/server-brave-search
```



2. Configure MCP Client credentials:
 - **Command**: `npx`
 - **Arguments**: `-y @modelcontextprotocol/server-brave-search`
 - **Environment Variables**: `BRAVE_API_KEY=your-api-key` Add a variables (space comma or newline separated)
3. Create a workflow that uses the MCP Client node:
 - Add an MCP Client node
 - Select the "List Tools" operation to see available search tools
 - Add another MCP Client node
 - Select the "Execute Tool" operation
 - Choose the "brave_search" tool
 - Set Parameters to: `{"query": "latest AI news"}`



The node will execute the search and return the results in the output.

Example: Multi-Server Setup with AI Agent

This example demonstrates how to set up multiple MCP servers in a production environment and use them with an AI agent:

1. Configure your docker-compose.yml file:

```
version: '3'

services:
  n8n:
    image: n8nio/n8n
    environment:
      # MCP server environment variables
      - MCP_BRAVE_API_KEY=your-brave-api-key
      - MCP_OPENAI_API_KEY=your-openai-key
      - MCP_SERPER_API_KEY=your-serper-key
      - MCP_WEATHER_API_KEY=your-weather-api-key

      # Enable community nodes as tools
      - N8N_COMMUNITY_PACKAGES_ALLOW_TOOL_USAGE=true
    ports:
      - "5678:5678"
    volumes:
      - ~/.n8n:/home/node/.n8n
```

2. Create multiple MCP Client credentials in n8n:

Brave Search Credentials:

- Command: npx
- Arguments: -y @modelcontextprotocol/server-brave-search

OpenAI Tools Credentials:

- Command: npx
- Arguments: -y @modelcontextprotocol/server-openai

Web Search Credentials:

- Command: npx
- Arguments: -y @modelcontextprotocol/server-serper

Weather API Credentials:

- Command: npx
- Arguments: -y @modelcontextprotocol/server-weather

3. Create an AI Agent workflow:

- Add an AI Agent node
- Enable MCP Client as a tool
- Configure different MCP Client nodes with different credentials
- Create a prompt that uses multiple data sources



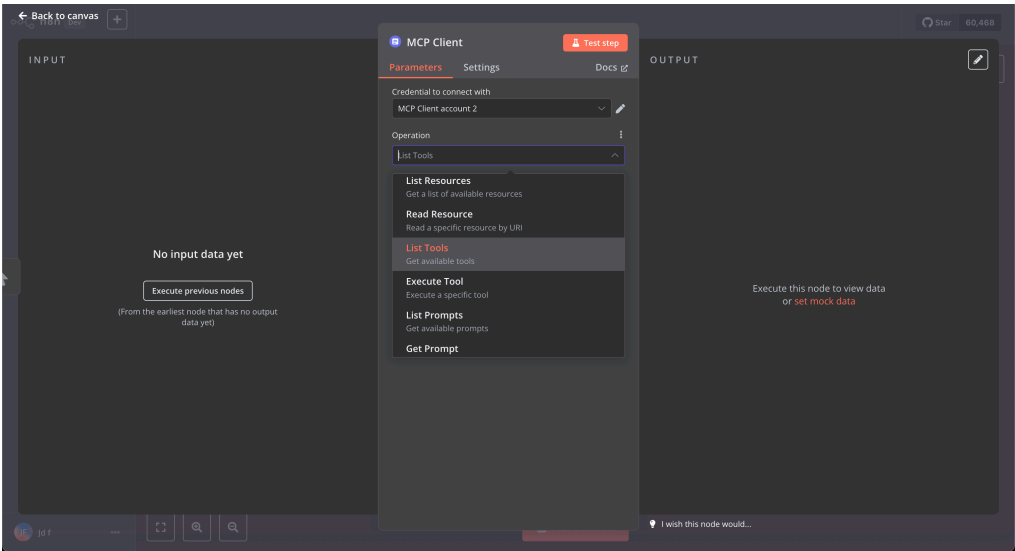
Example AI Agent prompt:

I need you to help me plan a trip. First, search for popular destinations in {destination_country}. Then, check the current weather in the top 3 cities. Finally, find some recent news about travel restrictions for these places.

With this setup, the AI agent can use multiple MCP tools across different servers, all using environment variables configured in your Docker deployment.

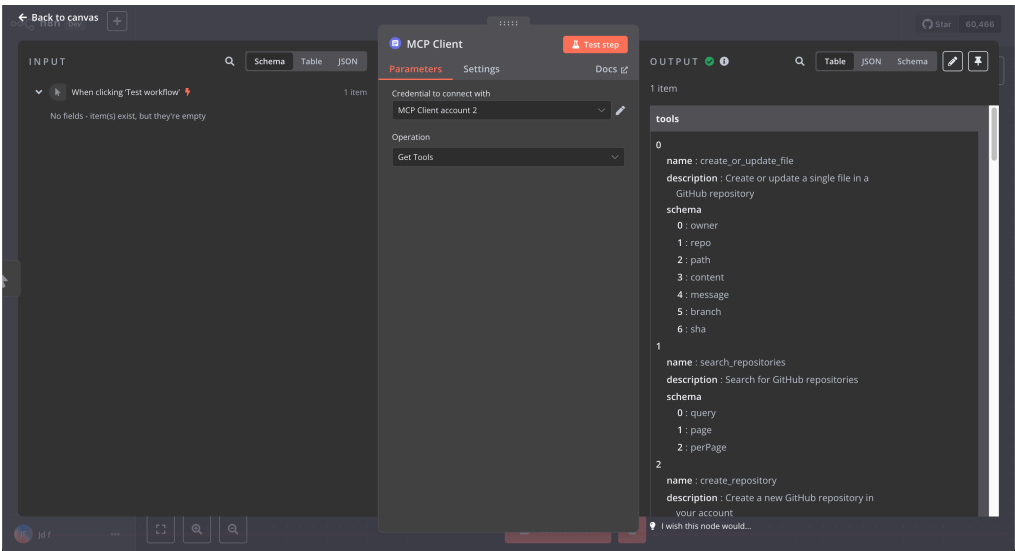
Operations

The MCP Client node supports the following operations:



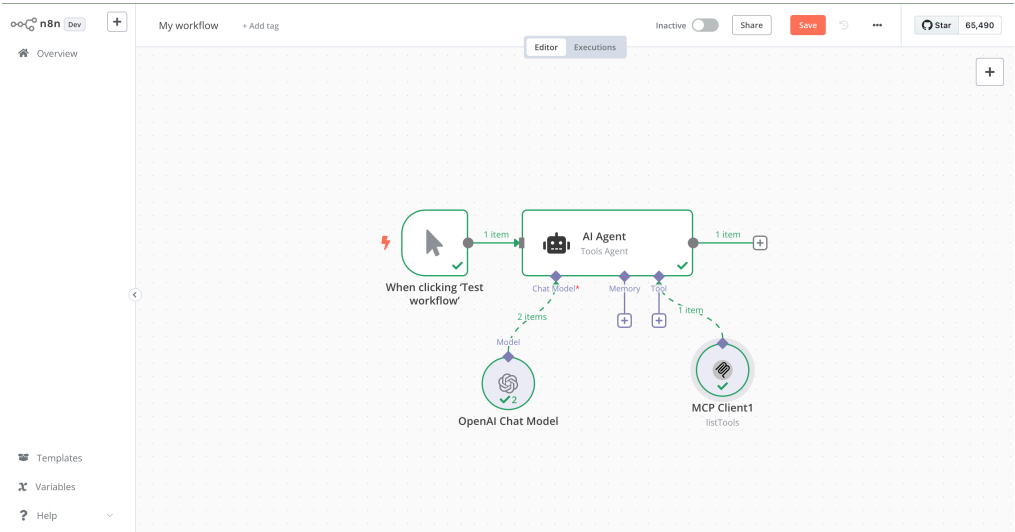
- Execute Tool - Execute a specific tool with parameters
- Get Prompt - Get a specific prompt template
- List Prompts - Get a list of available prompts
- List Resources - Get a list of available resources from the MCP server
- List Tools - Get a list of available tools
- Read Resource - Read a specific resource by URI

Example: List Tools Operation



The List Tools operation returns all available tools from the MCP server, including their names, descriptions, and parameter schemas.

Example: Execute Tool Operation



The Execute Tool operation allows you to execute a specific tool with parameters. Make sure to select the tool you want to execute from the dropdown menu.

Using as a Tool

This node can be used as a tool in n8n AI Agents. To enable community nodes as tools, you need to set the `N8N_COMMUNITY_PACKAGES_ALLOW_TOOL_USAGE` environment variable to `true`.

Setting the Environment Variable

If you're using a bash/zsh shell:

```
export N8N_COMMUNITY_PACKAGES_ALLOW_TOOL_USAGE=true
n8n start
```

If you're using Docker: Add to your docker-compose.yml file:

```
environment:
  - N8N_COMMUNITY_PACKAGES_ALLOW_TOOL_USAGE=true
```

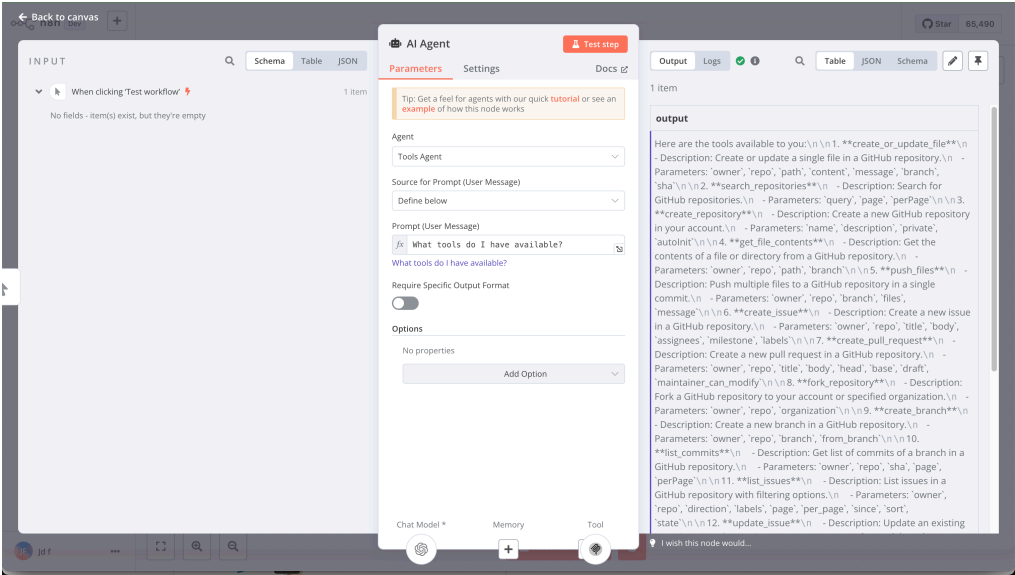
If you're using the desktop app: Create a `.env` file in the n8n directory:

N8N_COMMUNITY_PACKAGES_ALLOW_TOOL_USAGE=true

If you want to set it permanently on Mac/Linux: Add to your ~/.zshrc or ~/.bash_profile :

```
export N8N_COMMUNITY_PACKAGES_ALLOW_TOOL_USAGE=true
```

Example of an AI Agent workflow results:



After setting this environment variable and restarting n8n, your MCP Client node will be available as a tool in AI Agent nodes.

Compatibility

- Requires n8n version 1.0.0 or later
- Compatible with MCP Protocol version 1.0.0 or later
- Supports both STDIO and SSE transports for connecting to MCP servers

Releases 21

📦 Release v0.1.28 Latest
on May 24

+ 20 releases

Packages

No packages published

Contributors 12



Languages

• TypeScript 88.4% • JavaScript 11.6%