


[Integrations](#) [Built-in nodes](#) [Core nodes](#)

MCP Server Trigger node

Use the MCP Server Trigger node to allow n8n to act as a [Model Context Protocol \(MCP\)](#) server, making n8n tools and workflows available to MCP clients.

 **Credentials**

Permanent link

You can find authentication information for this node [here](#).

How the MCP Server Trigger node works

The MCP Server Trigger node acts as an entry point into n8n for MCP clients. It operates by exposing a URL that MCP clients can interact with to access n8n tools.

Unlike conventional [trigger nodes](#), which respond to events and pass their output to the next [connected node](#), the MCP Server Trigger node only connects to and executes [tool](#) nodes. Clients can list the available tools and call individual tools to perform work.

You can expose n8n workflows to clients by attaching them with the [Custom n8n Workflow Tool](#) node.

Server-Sent Events (SSE) support

The MCP Server Trigger node supports [Server-Sent Events \(SSE\)](#), a long-lived transport built on top of HTTP, for connections between clients and the server. It currently doesn't support [standard input/output \(stdio\)](#) transport.

Node parameters

Use these parameters to configure your node.

MCP URL

The MCP Server Trigger node has two **MCP URLs**: test and production. n8n displays the URLs at the top of the node panel.

Select **Test URL** or **Production URL** to toggle which URL n8n displays.

- **Test**: n8n registers a test MCP URL when you select **Listen for Test Event** or **Execute workflow**, if the workflow isn't active. When you call the MCP URL, n8n displays the data in the workflow.
- **Production**: n8n registers a production MCP URL when you activate the workflow. When using the production URL, n8n doesn't display the data in the workflow. You can still view workflow data for a production execution: select the **Executions** tab in the workflow, then select the workflow execution you want to view.

Authentication

You can require authentication for clients connecting to your MCP URL. Choose from these authentication methods:

- Bearer auth
- Header auth

Refer to the [HTTP request credentials](#) for more information on setting up each credential type.

Path

By default, this field contains a randomly generated MCP URL path, to avoid conflicts with other MCP Server Trigger nodes.

You can manually specify a URL path, including adding route parameters. For example, you may need to do this if you use n8n to prototype an API and want consistent endpoint URLs.

The **Path** field can take the following formats:

- `/:variable`
- `/path/:variable`
- `/:variable/path`
- `/:variable1/path/:variable2`
- `/:variable1/:variable2`

Templates and examples

Build an MCP Server with Google Calendar and Custom Functions

by Solomon

[View template details](#)

Build your own N8N Workflows MCP Server

by Jimleuk

[View template details](#)

AI-Powered Telegram Task Manager with MCP Server

by Francis Njenga

[View template details](#)

[Browse MCP Server Trigger integration templates](#), or [search all templates](#)

Integrating with Claude Desktop

You can connect to the MCP Server Trigger node from [Claude Desktop](#) by running a gateway to proxy SSE messages to stdio-based servers.

To do so, add the following to your Claude Desktop configuration:

```
1  {
2    "mcpServers": {
3      "n8n": {
4        "command": "npx",
5        "args": [
6          "mcp-remote",
7          "<MCP_URL>",
8          "--header",
9          "Authorization: Bearer ${AUTH_TOKEN}"
10       ],
11      "env": {
12        "AUTH_TOKEN": "<MCP_BEARER_TOKEN>"
```

```
13     }  
14     }  
15   }  
16 }
```

Be sure to replace the `<MCP_URL>` and `<MCP_BEARER_TOKEN>` placeholders with the values from your MCP Server Trigger node parameters and credentials.

Limitations

Configuring the MCP Server Trigger node with webhook replicas

The MCP Server Trigger node relies on Server-Sent Events (SSE), which require the same server instance to handle persistent connections. This can cause problems when running n8n in **queue mode** depending on your **webhook processor** configuration:

- If you use queue mode with a **single webhook replica**, the MCP Server Trigger node works as expected.
- If you run **multiple webhook replicas**, you need to route all `/mcp*` requests to a single, dedicated webhook replica. Create a separate replica set with one webhook container for MCP requests. Afterward, update your ingress or load balancer configuration to direct all `/mcp*` traffic to that instance.

**Caution when running with multiple webhook replicas**

If you run an MCP Server Trigger node with multiple webhook replicas and don't route all `/mcp*` requests to a single, dedicated webhook replica, your SSE connections will frequently break or fail to reliably deliver events.

Related resources

n8n also provides an [MCP Client Tool](#) node that allows you to connect your n8n AI agents to external tools.

Refer to the [MCP documentation](#) and [MCP specification](#) for more details about the protocol, servers, and clients.

Common issues

Here are some common errors and issues with the MCP Server Trigger node and steps to resolve or troubleshoot them.

Running the MCP Server Trigger node with a reverse proxy

When running n8n behind a reverse proxy like nginx, you may experience problems if the MCP endpoint isn't configured for SSE.

Specifically, you need to disable proxy buffering for the endpoint. Other items you might want to adjust include disabling gzip compression (n8n handles this itself), disabling chunked transfer encoding, and setting the `Connection` to an empty string to remove it from the forwarded headers.

Explicitly disabling these in the MCP endpoint ensures they're not inherited from other places in your nginx configuration.

An example nginx location block for serving MCP traffic with these settings may look like this:

```
1  location /mcp/ {  
2      proxy_http_version      1.1;  
3      proxy_buffering         off;  
4      gzip                    off;  
5      chunked_transfer_encoding off;  
6  
7      proxy_set_header        Connection '';  
8  
9      # The rest of your proxy headers and settings  
10     # . . .  
11 }
```