Choose node file structure

# Node file structure#

Following best practices and standards in your node structure makes your node easier to maintain. It's helpful if other people need to work with the code.

The file and directory structure of your node depends on:

• Your node's complexity.

• Whether you use node versioning.

• How many nodes you include in the npm package.

## Required files and directories#

Your node must include:

• A package.json file at the root of the project. This is required for any npm module.

package.json

• A nodes directory, containing the code for your node:  
This directory must contain the base file, in the format <node-name>.node.ts. For example, MyNode.node.ts.  
n8n recommends including a codex file, containing metadata for your node. The codex filename must match the node base filename. For example, given a node base file named MyNode.node.ts, the codex name is MyNode.node.json.  
The nodes directory can contain other files and subdirectories, including directories for versions, and node code split across more than one file to create a modular structure.

nodes

• This directory must contain the base file, in the format <node-name>.node.ts. For example, MyNode.node.ts.

<node-name>.node.ts

MyNode.node.ts

• n8n recommends including a codex file, containing metadata for your node. The codex filename must match the node base filename. For example, given a node base file named MyNode.node.ts, the codex name is MyNode.node.json.

MyNode.node.ts

MyNode.node.json

• The nodes directory can contain other files and subdirectories, including directories for versions, and node code split across more than one file to create a modular structure.

nodes

• A credentials directory, containing your credentials code. This code lives in a single credentials file. The filename format is <node-name>.credentials.ts. For example, MyNode.credentials.ts.

credentials

<node-name>.credentials.ts

MyNode.credentials.ts

## Modular structure#

You can choose whether to place all your node's functionality in one file, or split it out into a base file and other modules, which the base file then imports. Unless your node is very simple, it's a best practice to split it out.

A basic pattern is to separate out operations. Refer to the HttpBin starter node for an example of this.

For more complex nodes, n8n recommends a directory structure. Refer to the Airtable node or Microsoft Outlook node as examples.

• actions: a directory containing sub-directories that represent resources.  
Each sub-directory should contain two types of files:  
An index file with resource description (named either <resourceName>.resource.ts or index.ts)  
Files for operations <operationName>.operation.ts. These files should have two exports: description of the operation and an execute function.

actions

• Each sub-directory should contain two types of files:

• An index file with resource description (named either <resourceName>.resource.ts or index.ts)

<resourceName>.resource.ts

index.ts

• Files for operations <operationName>.operation.ts. These files should have two exports: description of the operation and an execute function.

<operationName>.operation.ts

description

execute

• methods: an optional directory dynamic parameters' functions.

methods

• transport: a directory containing the communication implementation.

transport

## Versioning#

If your node has more than one version, and you're using full versioning, this makes the file structure more complex. You need a directory for each version, along with a base file that sets the default version. Refer to Node versioning for more information on working with versions, including types of versioning.

## Decide how many nodes to include in a package#

There are two possible setups when building a node:

• One node in one npm package.

• More than one node in a single npm package.

n8n supports both approaches. If you include more than one node, each node should have its own directory in the nodes directory.

nodes

## A best-practice example for programmatic nodes#

n8n's built-in Airtable node implements a modular structure and versioning, following recommended patterns.