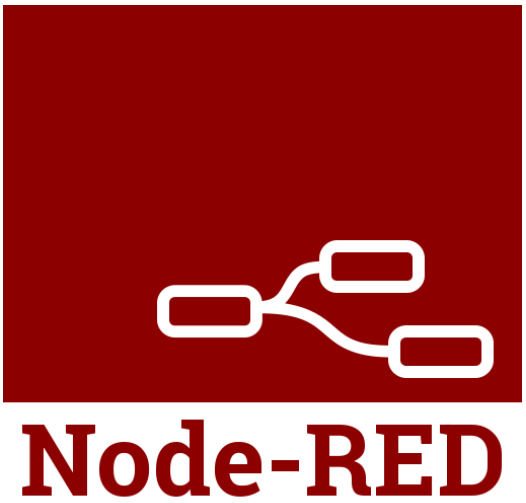
Batch Scheduling Process on Node-RED Integration Guide



Introduction

In this comprehensive guide, we will guide you through the process of setting up a batch processing scheduler using Node-RED. This guide will provide you with step-by-step instructions for installing Node-RED alongside some related Node-RED palettes required in this integration. We will demonstrate the integration, showing you how to automate your workflows.

Please note that this guide is based on the Node-RED version **3.1.3**. Future versions may have different features or requirements, so some steps in this guide might not be applicable.

Prerequisites

For equipments, you will need:

1. Any Windows PC/laptop
2. Network connection

Node-RED Installation

If you already have Node-RED installed, you can skip this step.

The following are instructions on how to install Node-RED on a Windows based operating system. All required download links can be found on the official Node-RED and NodeJS websites.

Node.js

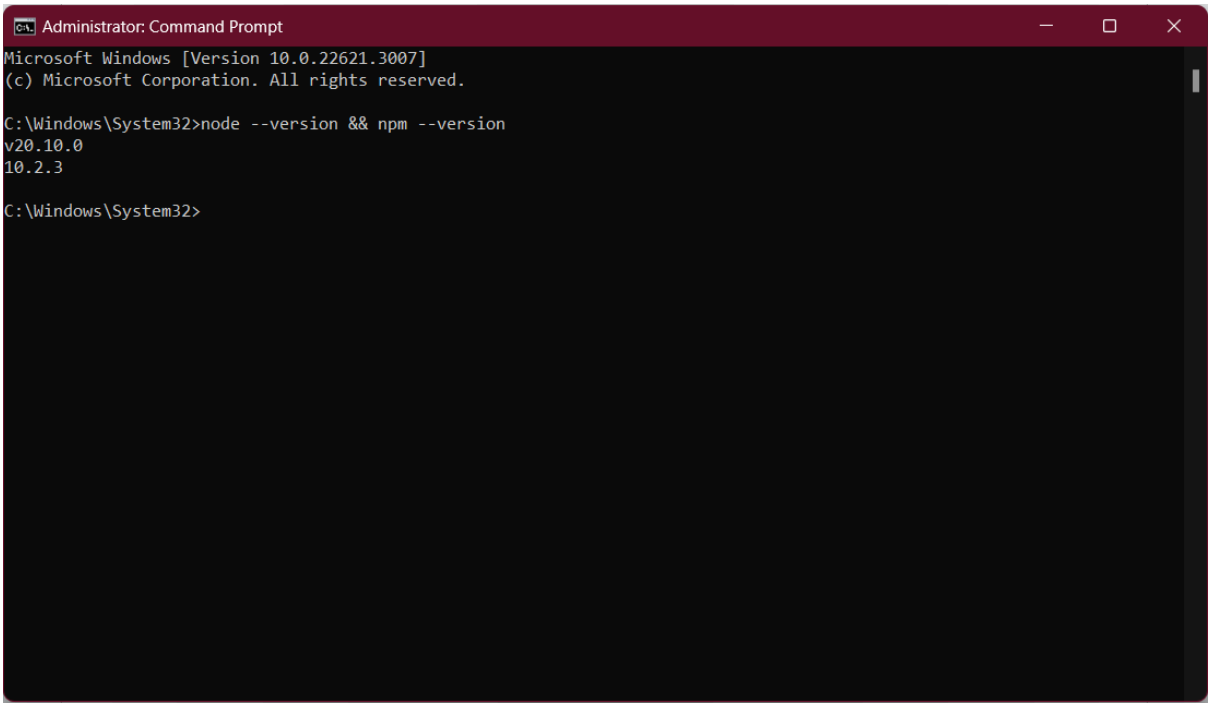
To install Node-RED locally you will need a supported version of Node.js. Node.js is a JavaScript based programming language and must be installed before Node-RED. Node.js can be downloaded from the official Node.js website.

Once the installation is complete, open up a Windows command prompt (CMD), you can open a command prompt by using the search box in your windows taskbar and type cmd and run the application as administrator then once the command prompt has opened, input the following:

```
node --version && npm --version
```

This command checks if both Node.js and NPM are correctly installed, and it will be expected to return a version number.

The output should look something like this:



NPM is a package manager for the JavaScript programming language, this is needed for installing additional Node-RED packages which we will need later. NPM should be installed by default when installing Node.js.

Install Node-RED

After installing Node.js, we can begin to install Node-RED, navigate to CMD and input the following command:

```
npm install -g --unsafe-perm node-red
```

That command will install Node-RED as a global module along with its dependencies.

Please note if you are having trouble with this please contact your System Administrator. Once this command is successfully run Node-RED will begin to install and this can take some time to complete. Once the installation is complete, the command command output should look something like:

```
+ node-red@1.1.0
added 332 packages from 341 contributors in 18.494s
found 0 vulnerabilities
```

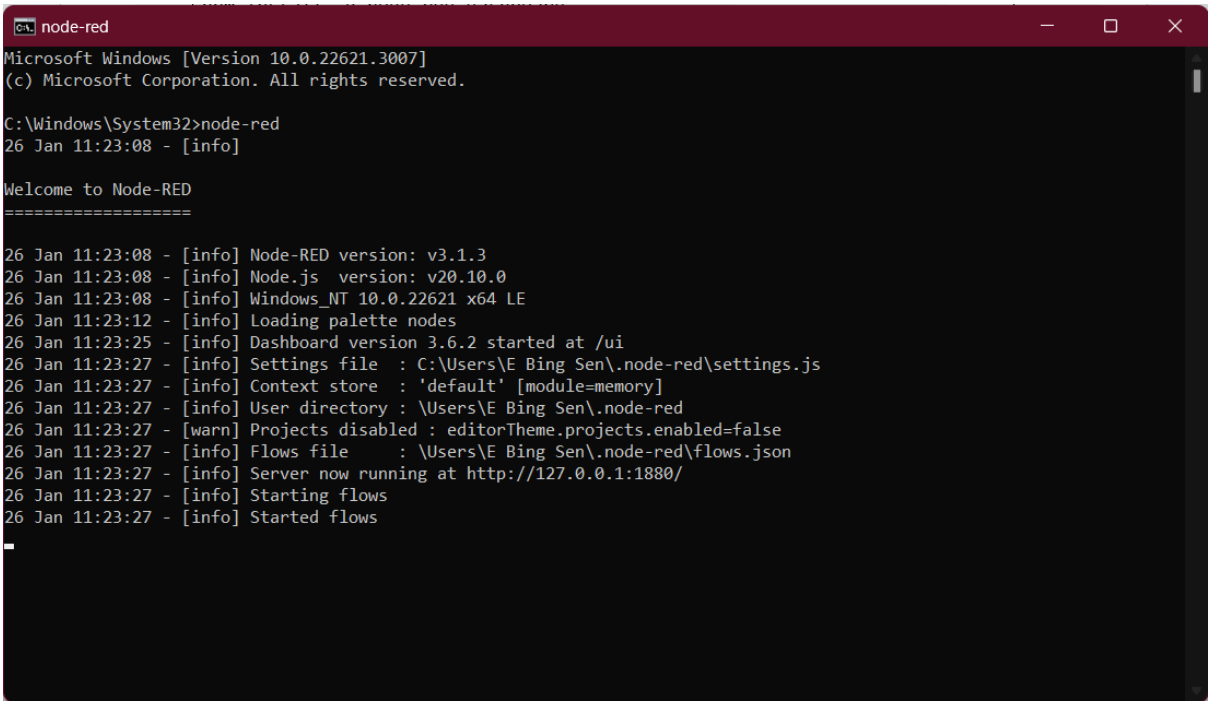
Run Node-RED

If you already know how to run Node-RED, you can skip this step.

You can begin to run your Node-RED web server and install additional packages within Node-RED. You can first do this by navigating to a command prompt and typing in the following command:

```
node-red
```

As we installed Node-RED as a global NPM module it adds the command node-red to your system path which you can then input into a command prompt to start Node-RED:



Once the server has booted up successfully the command prompt will let us know that the server is now running, and what IP address it is on.

You can find it at the line - [info] Server now running at http://127.0.0.1:1880/.

Now you can navigate to your web browser and input the IP address of the Node-RED web server or click on the link above to access the web server as most users will have the same IP address as the IP address of the machine running Node-RED. You have now successfully installed, booted up, and accessed your Node-RED web server.

You can now begin to create your own flows and install additional packages required for your flows.

Additional Packages downloads

The following are instructions on how to install additional packages in Node-RED.

By default, you should have a node palette downloaded called node-red. This module contains 50 nodes that are the basic building blocks for creating flows. All nodes include documentation you can see in the Info sidebar tab when you select a node.

The following additional packages are needed when following this integration guide.

node-red-dashboard

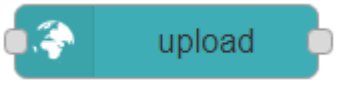
The first package you will need to install is the node-red-dashboard, this module adds nodes which allow us to easily create a live data dashboard. We will be using several nodes in this module for this guide.

To install this module, you have to:

1. Navigate to the hamburger menu on the top left of your screen.
2. Open the Manage palette tab. Alternatively (Alt+Shift+P).
3. Navigate to the Install tab.
4. Insert “node-red-dashboard” in the search bar.
5. Press the Install button.

node-red-contrib-ui-upload

The second package you will need to install is the node-red-contrib-ui-upload, this module adds nodes which allow us to upload a file content by WebSocket (Socket.io) streaming. We will be using the upload node in this guide.

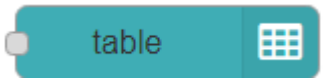


To install this module, you have to:

1. Navigate to the hamburger menu on the top left of your screen.
2. Open the Manage palette tab. Alternatively (Alt+Shift+P).
3. Navigate to the Install tab.
4. Insert “node-red-contrib-ui-upload” in the search bar.
5. Press the Install button.

node-red-node-ui-table

The third package you will need to install is the node-red-node-ui-table, this module adds nodes which allow us to display data as a table. We will be using the table node in this guide.



To install this module, you have to:

1. Navigate to the hamburger menu on the top left of your screen.
2. Open the Manage palette tab. Alternatively (Alt+Shift+P).
3. Navigate to the Install tab.
4. Insert “node-red-node-ui-table” in the search bar.
5. Press the Install button.

node-red-contrib-cron-plus

The fourth package you will need to install is the node-red-contrib-cron-plus, this module adds nodes which allow us to have full dynamic control with a flexible timer/scheduler. We will be using the cron-plus node in this guide.

To install this module, you have to:

1. Navigate to the hamburger menu on the top left of your screen.
2. Open the Manage palette tab. Alternatively (Alt+Shift+P).
3. Navigate to the Install tab.
4. Insert “node-red-contrib-cron-plus” in the search bar.
5. Press the Install button.

node-red-contrib-cpu

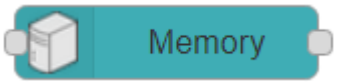
The fifth package you will need to install is the node-red-contrib-cpu, this module adds nodes which allow us to monitor your cpu usage. We will be using the cpu node in this guide.

To install this module, you have to:

1. Navigate to the hamburger menu on the top left of your screen.
2. Open the Manage palette tab. Alternatively (Alt+Shift+P).
3. Navigate to the Install tab.
4. Insert “node-red-contrib-cpu” in the search bar.
5. Press the Install button.

node-red-contrib-os

The last package you will need to install is the node-red-contrib-os, this module adds nodes which allow us to monitor your cpu system information. But we will be only focusing on using the Memory node in this guide.

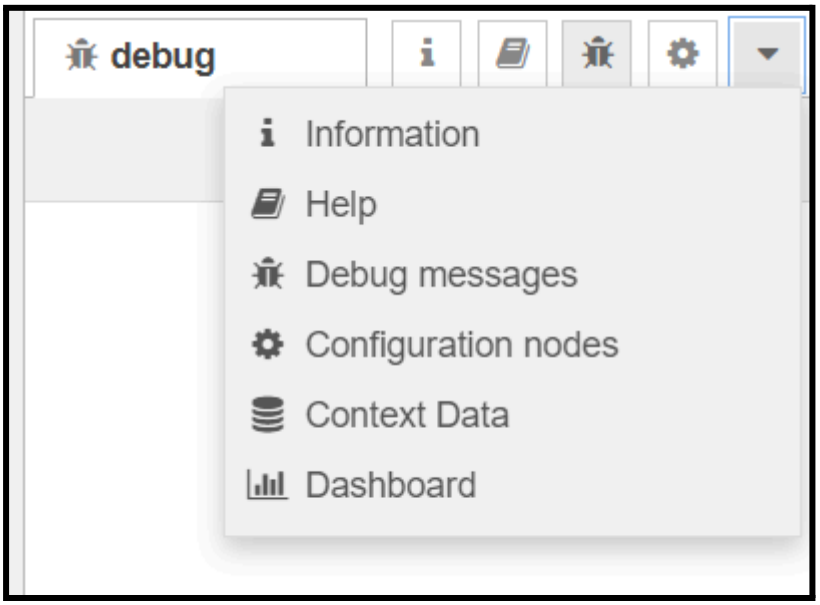


To install this module, you have to:

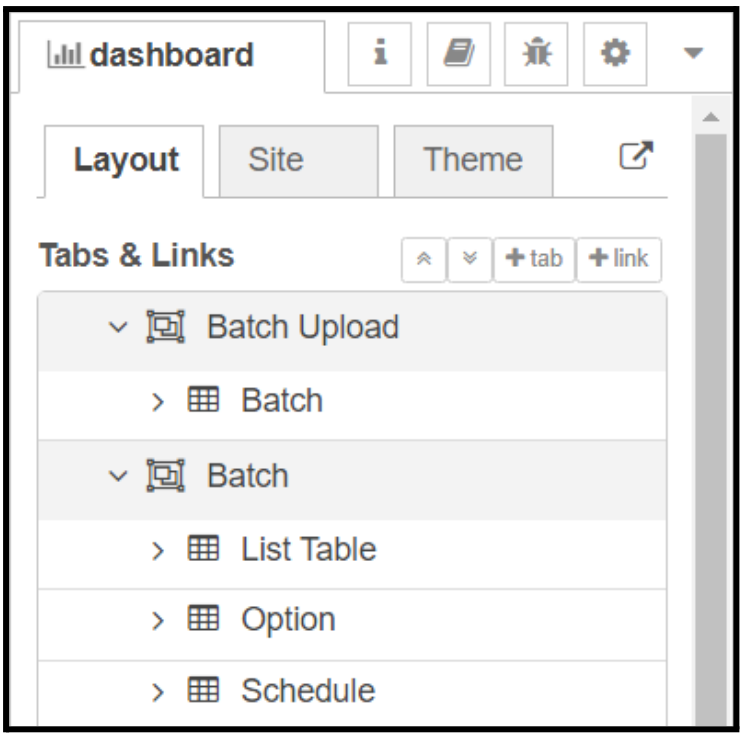
1. Navigate to the hamburger menu on the top left of your screen.
2. Open the Manage palette tab. Alternatively (Alt+Shift+P).
3. Navigate to the Install tab.
4. Insert “node-red-contrib-os” in the search bar.
5. Press the Install button.

Setting up the Dashboard

The following are instructions on setting up the dashboard and creating tabs for the Batch Processing Scheduler which includes Uploading the batch files and Scheduling the uploaded batch files.



You can access the dashboard by clicking on the dropdown button at the sidebar and navigate to the dashboard tab. You will need to create several tabs and groups for the Batch Processing Scheduler as it allows you to allocate any UI Nodes to be in their respective tabs. Example as shown below:

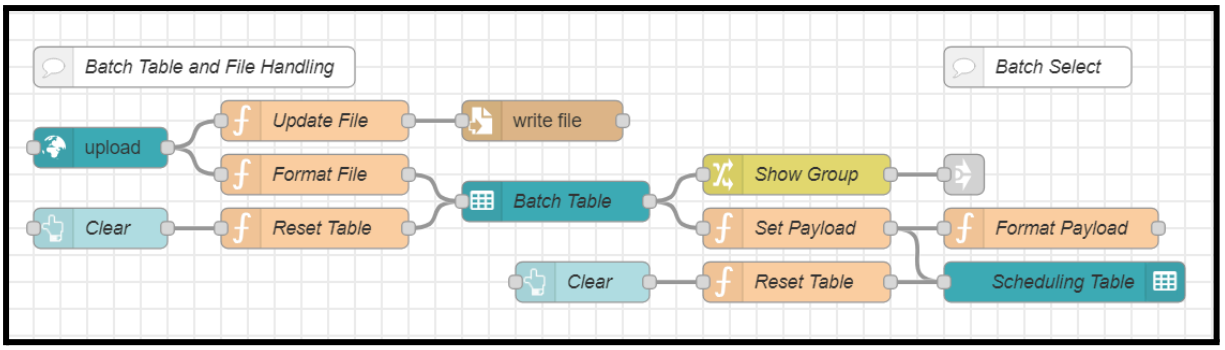


Start Building the Workflow

The following are instructions on how to build the workflow for the Batch Scheduling Process in Node-RED, divided in three different sections.

Batch Table and File Handling

We will start by building the Batch Scheduling Process from the batch file and file handling section first. The flow shown below is an example of the Batch Table and File Handling:



This flow serves two primary purposes:

Uploading and Saving Batch Files:

- Utilises an upload node to help in the batch file upload process.
- Employs two function nodes:
    - Writes and saves the file to the local machine.
    - Formats and display the batch files into the UI table on the Node-RED dashboard.
- Features a 'Clear' button paired with a function node, allowing users to empty the Batch Table for a reset.

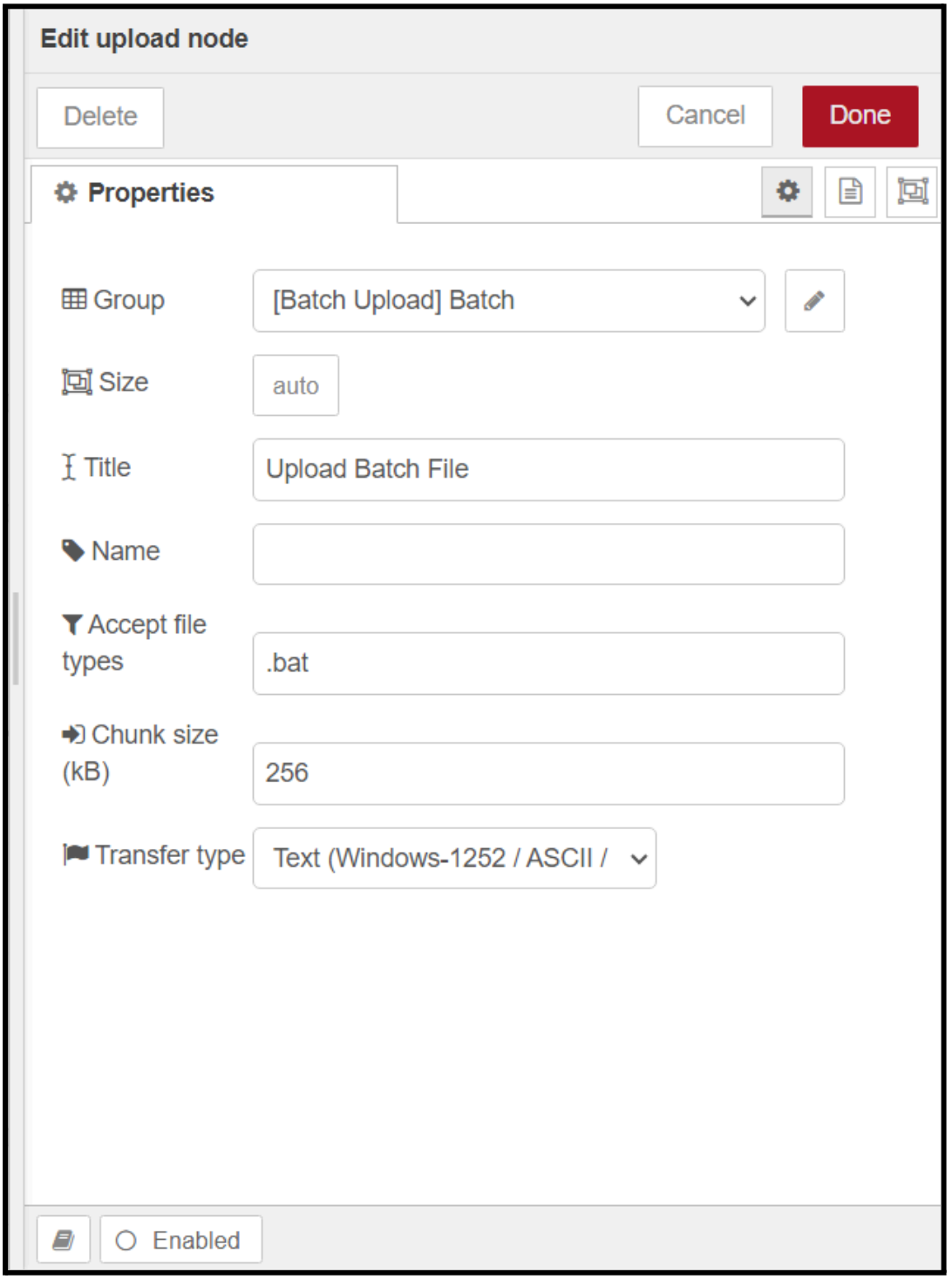
Selecting Batch Files for Scheduling:

- After creating the Batch Table, a function node enables users to choose batch files for scheduling.
- Sends selected files to the second table (Scheduling Table).
- Includes a 'Clear' button paired with another function node to empty the Scheduling Table for a reset.
- Includes a function node that formats file names for local machine access using the file path.

Node Configurations

The following will be the instructions for configuring the nodes added into the workflow. You can double click on the nodes to configure them.

1. **Upload**



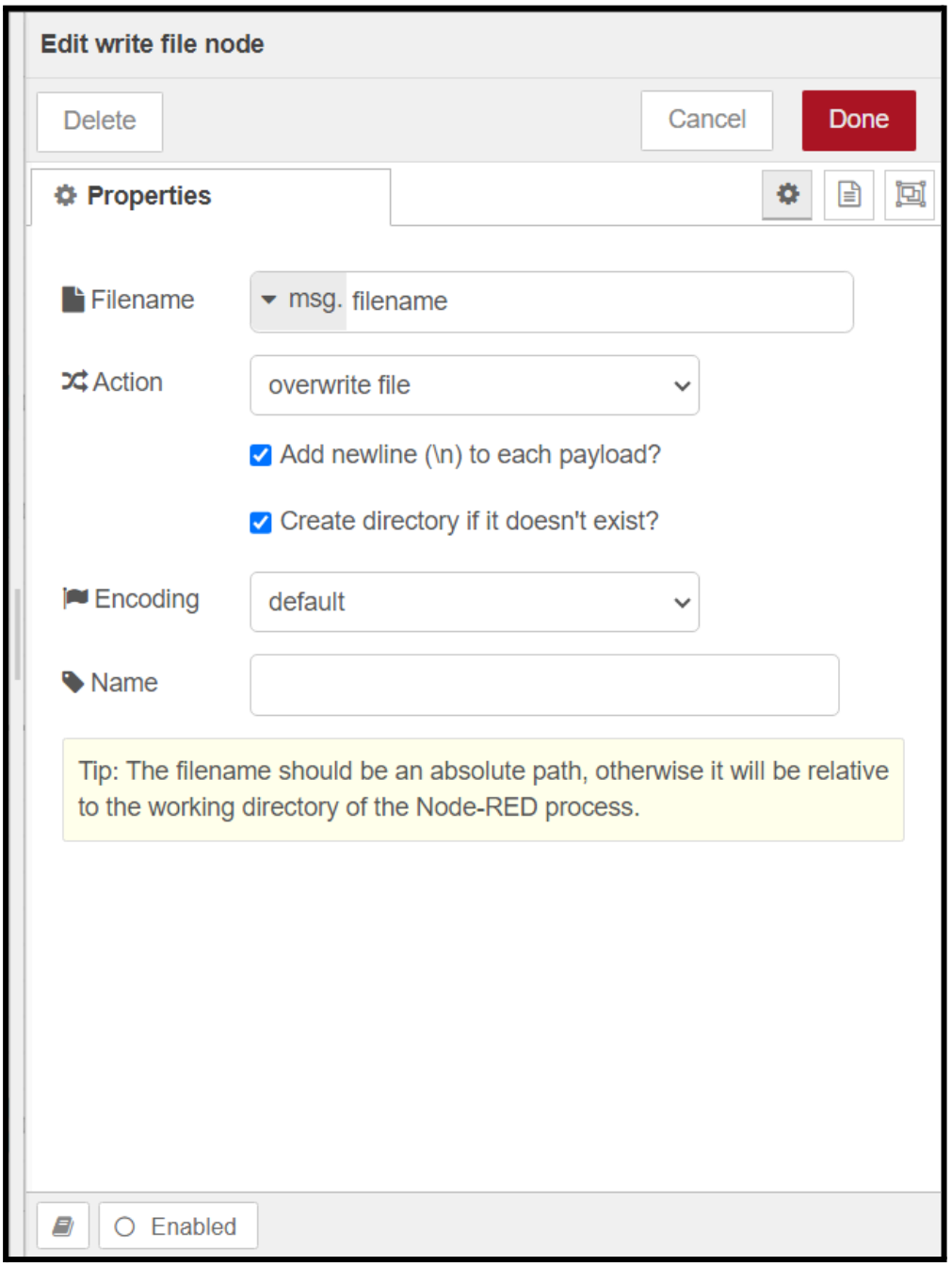
This node allows you to upload files to the Node-RED Dashboard and you want this node to accept only batch file types and set the transfer type to text so it can be displayed in the UI Table.

2. **Function (Update File)**

```
var content = msg.payload;
var filenameMatch = content.match(/^SET FILENAME=(.+)$/m);

if (filenameMatch && filenameMatch.length > 1) {
    msg.filename = filenameMatch[1].trim();
} else {
    node.warn("Filename not found in the batch file content.");
}

msg.filename = 'D:/BatchFiles/' + msg.filename;

return msg;
```

This Update File Function node uses a javascript code to change the filename to your desired file path so that the nodes will be able to access the batch file stored in your desired file path. **Please note** that `'D:/BatchFiles/'` will be the directory of where you will save these files. You should replace this with your preferred path. After making this change, the system will be able to locate and access the batch files in your chosen directory.

3. **Write File**



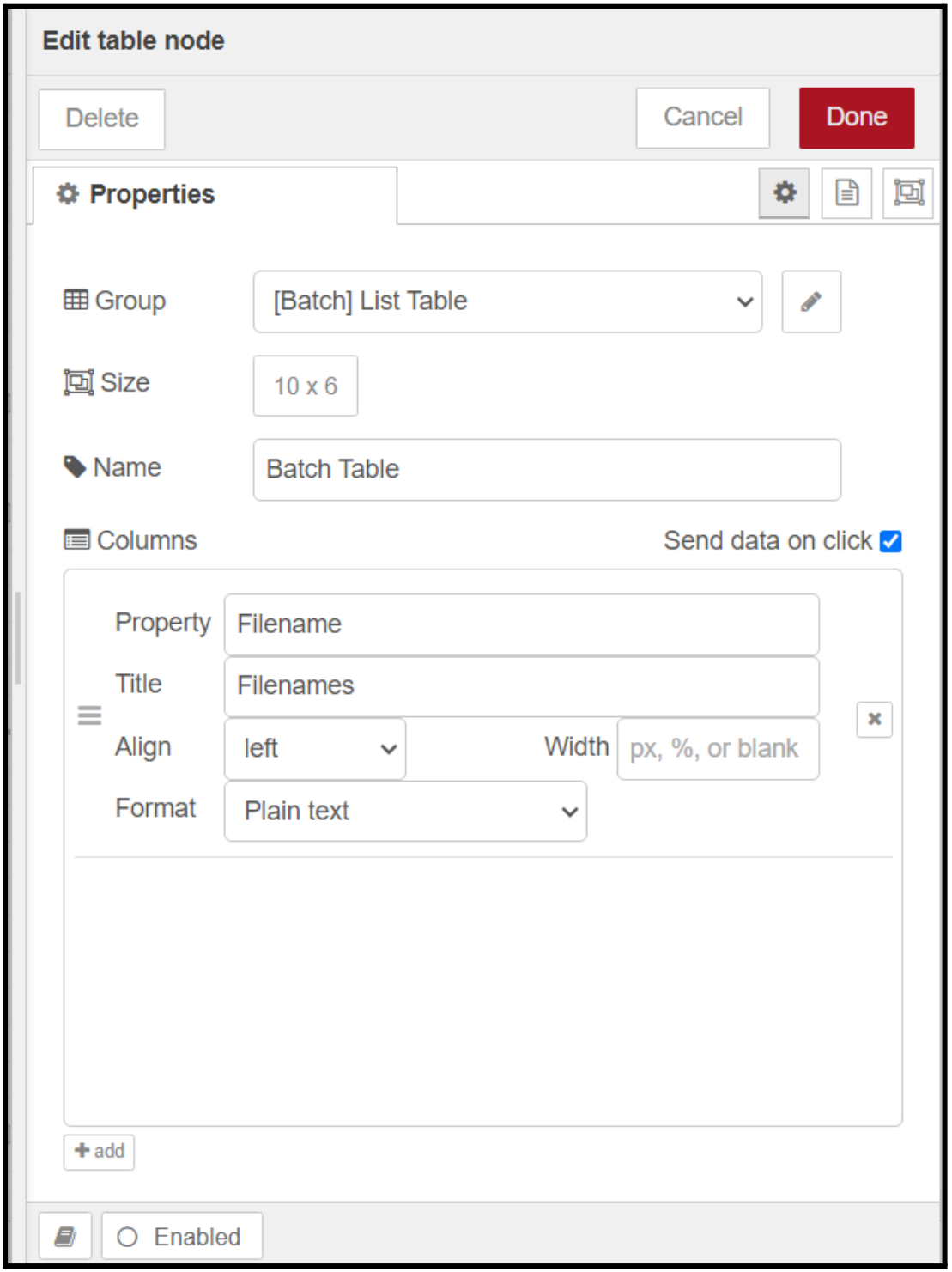
This node is to write msg.payload into a file and you want to change the Filename property to msg.filename as the previous node saves the file path in msg.filename. You also need to change the action to overwrite the file to perform the action in the batch file and check for “Create directory if it doesn’t exist?”.

4. **Function (Format File)**

```
var content = msg.payload;
var filenameMatch = content.match(/^SET FILENAME=(.+)$/m);
var filenames = flow.get("filenames") || [];

if (filenameMatch && filenameMatch.length > 1) {
    var filename = filenameMatch[1].trim();
    filenames.push(filename);
} else {
    node.warn("Filename not found in the batch file content.");
}

var tableData = filenames.map(function (name) {
    return { "Filename": name };
});

flow.set("filenames", filenames);

msg.payload = tableData;
return msg;
```

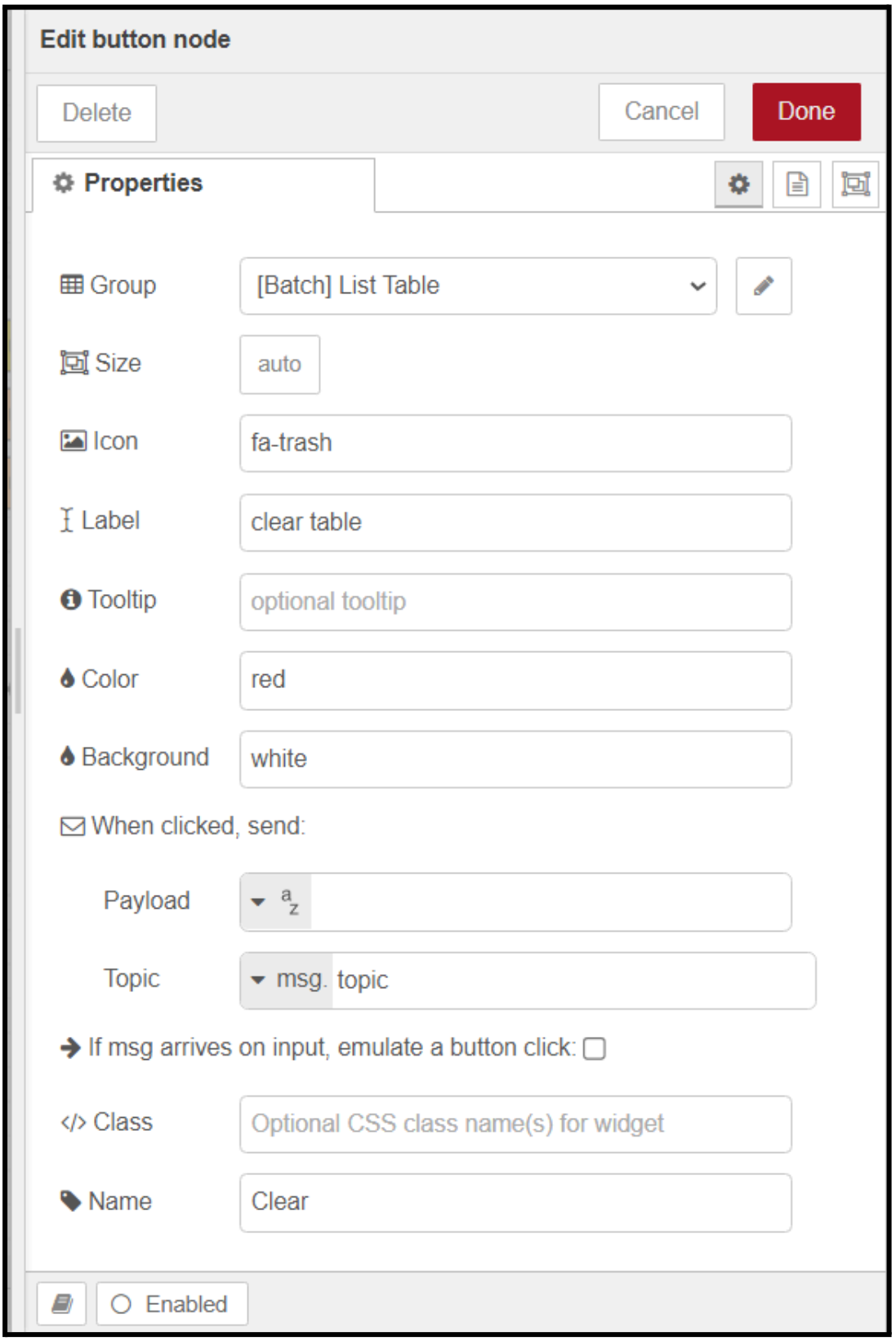
This Format File Function node uses a javascript code to map the batch files uploaded into a format where it can be saved into the UI table node.

5. **Table (Batch Table)**



This table node contains a column for Filenames and has the property of Filename. Please note that the property field should be the same as the variable name in the flow variable. You should also check for the “Send data on click”, so that you can pass data to another node.

6. **Button (Clear)**



This button node can be customised with an icon, colour, background and more.

7. **Function (Reset Table)**

```
flow.set("filenames", []);

msg.payload = {
    "command": "clearData",
    arguments: [],
    returnPromise: true
}
return msg;
```

This Reset Table Function node uses javascript codes to clear the data in the table connected to this node.

8. **Function (Set Payload)**

```
const choose = msg.payload;

let fileRows = flow.get("toSchedule") || [];

fileRows.push(choose);

const tableData2 = fileRows.map(item => item.Selected ?
item.Selected.Filename : item);

flow.set("toSchedule", fileRows);

msg.payload = tableData2;
return msg;
```

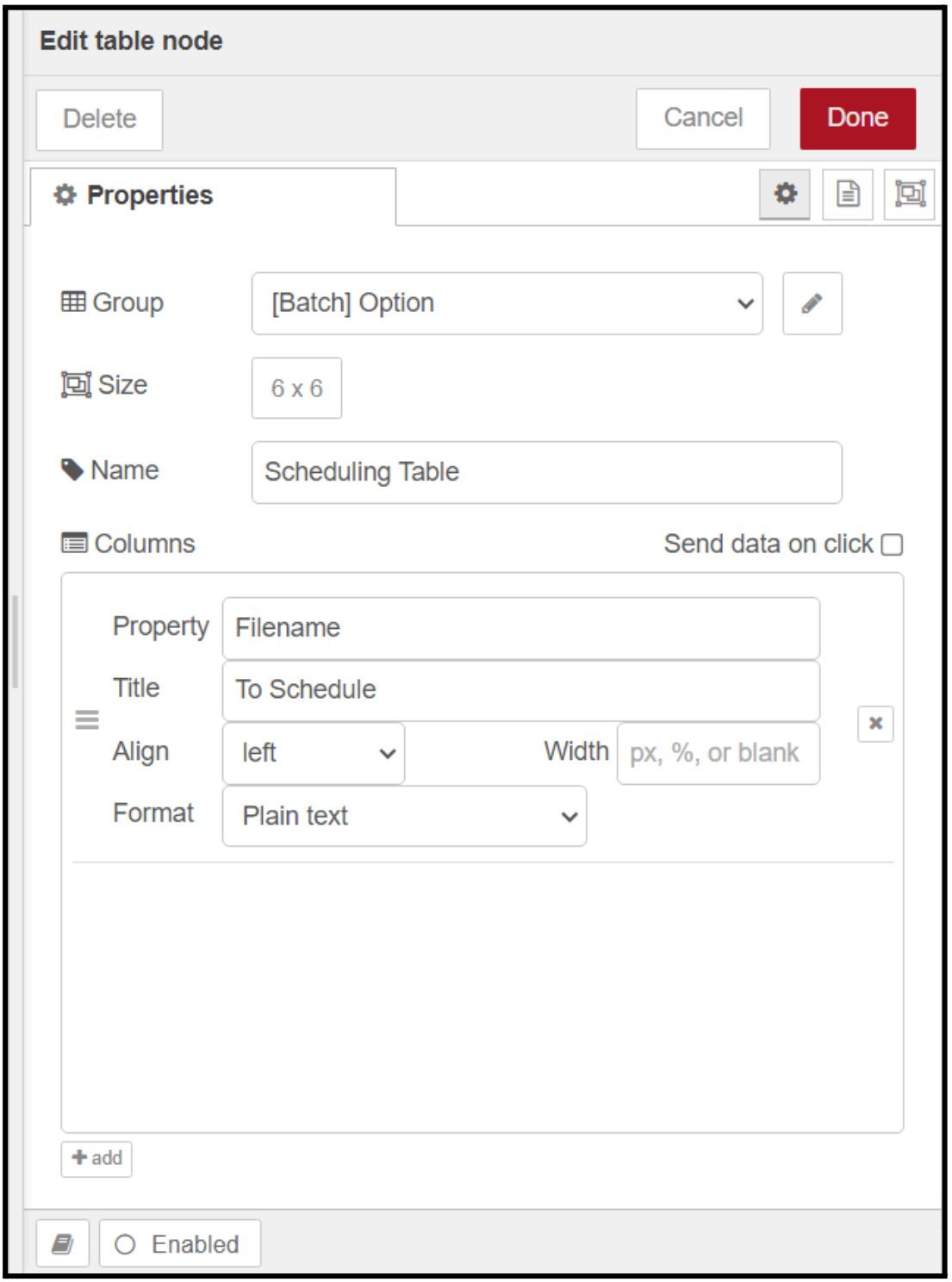
This Set Payload Function node uses javascript codes to format the selected batch files to store it in a new table for scheduling.

9. **Function (Format Payload)**

```
let tableData = msg.payload;

let filenames = [];

for (let i = 0; i < tableData.length; i++) {
    let row = tableData[i];
    filenames.push(row.Filename);
}

msg.payload = filenames.map(file => `D:/BatchFiles/${file}`);

flow.set("Array", msg.payload);

return msg;
```

This Format Payload Function node formats the name of the batch file from the table to obtain the path of the batch files stored previously as you uploaded them. **Please note** that `'D:/BatchFiles/'` will be the directory of where you will save these files. You should replace this with your previously set path. After making this change, the system will be able to access the batch files in your chosen directory.

10. Table (Scheduling Table)

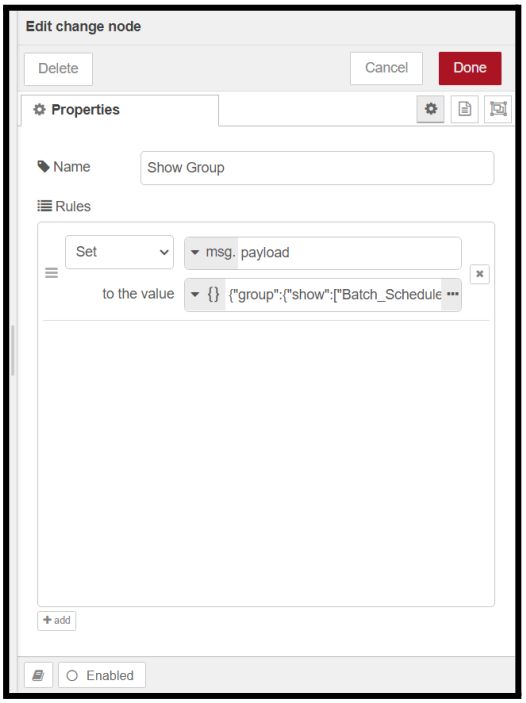


This Table node contains a column for To Schedule which also uses the Property of Filename that is modified in the previous function node, so you will only obtain the batch files that are selected from the Batch Table node. You will also have to add the Clear button and the function node for it to reset the table so you will not have a problem with selecting the wrong batch files to schedule.

Optional Node(s)

The following nodes are optional as they only improve the UI of the software.

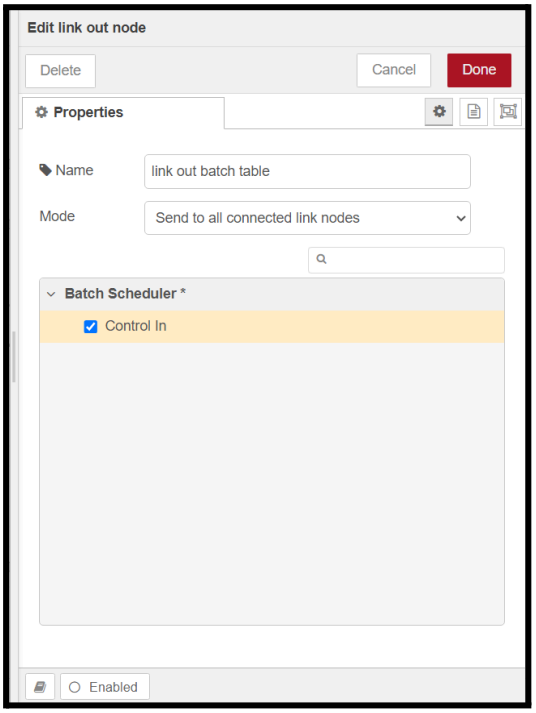
1. **Change (Show Group)**




```
{
    "group": {
        "show": [
            "Batch_Schedule"
        ]
    }
}
```

This change node is to show a new group in the UI Dashboard when a data is clicked in the table. You can click on the ellipses to format json.

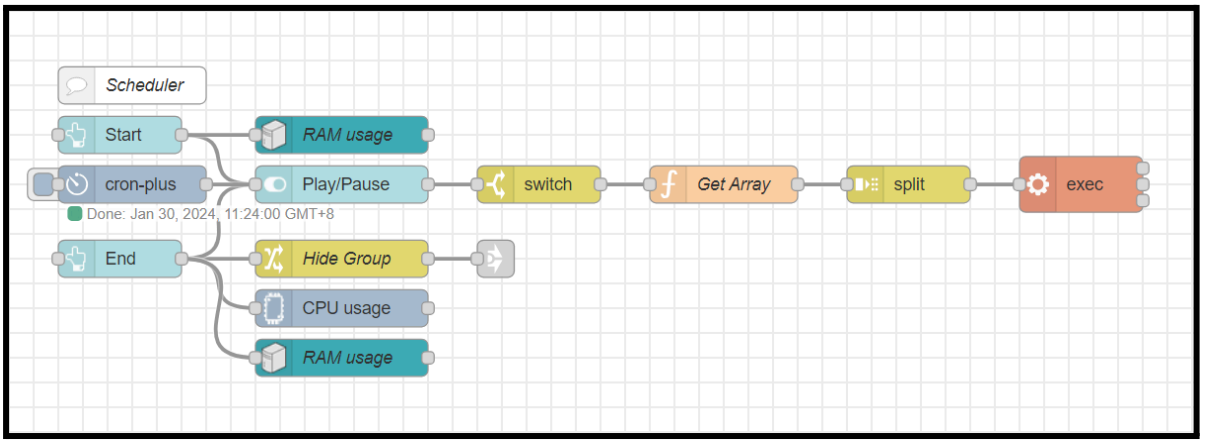
2. **Link-out**



This Link-out node is to connect the flow and make the flow more organised.

Scheduler

We will follow up by building the Batch Scheduling Process from the Scheduler section. The flow shown below is an example of the Scheduler:



This flow has a primary focus on scheduling the selected batch files and is able to control the start and stop the process.

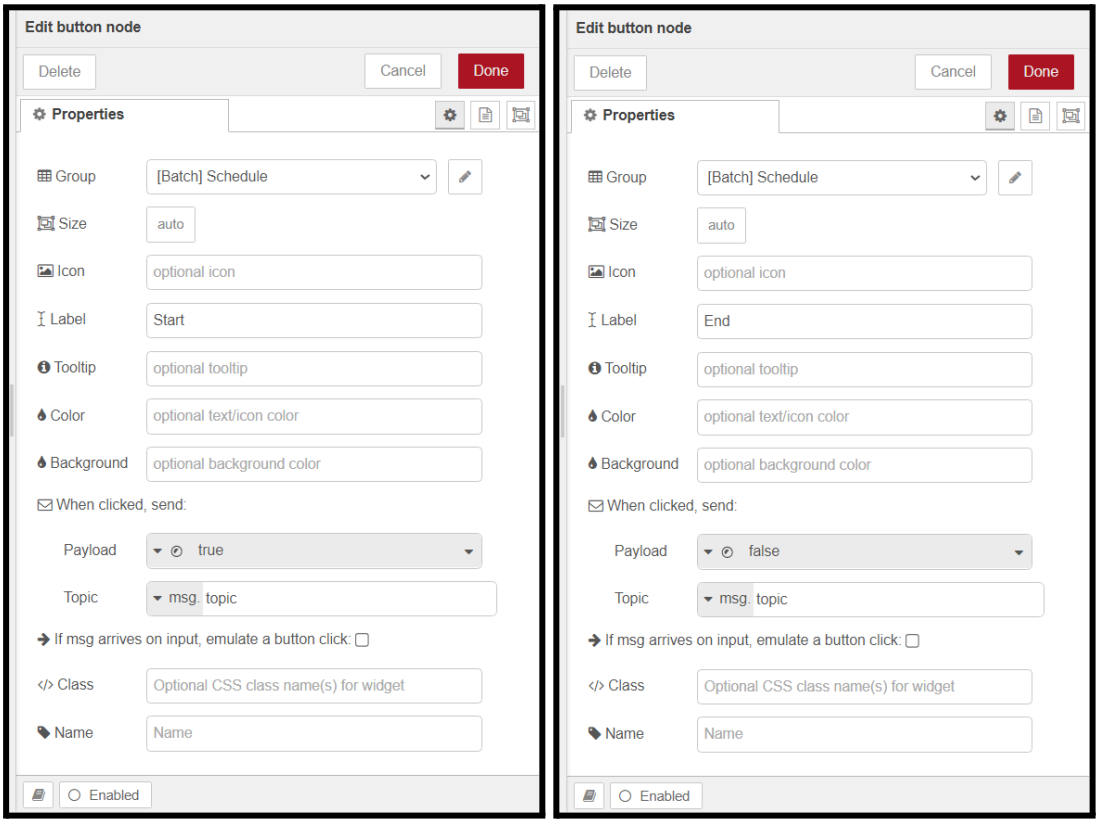
Scheduling:

- Utilises a cron-plus node to help in the scheduling process.
- Employs two button nodes:
    - Start scheduling process
    - End scheduling process
- Features two switch nodes:
    - node-red-dashboard : ui_switch
        - A toggle switch in Node-RED Dashboard for interactive uses.
    - node-red : switch
        - It evaluates the incoming payload and routes the message to the appropriate output.
- Uses a function node to obtain the selected array for scheduling.
- Splitting the array for executing each batch file required for scheduling.

Node Configurations

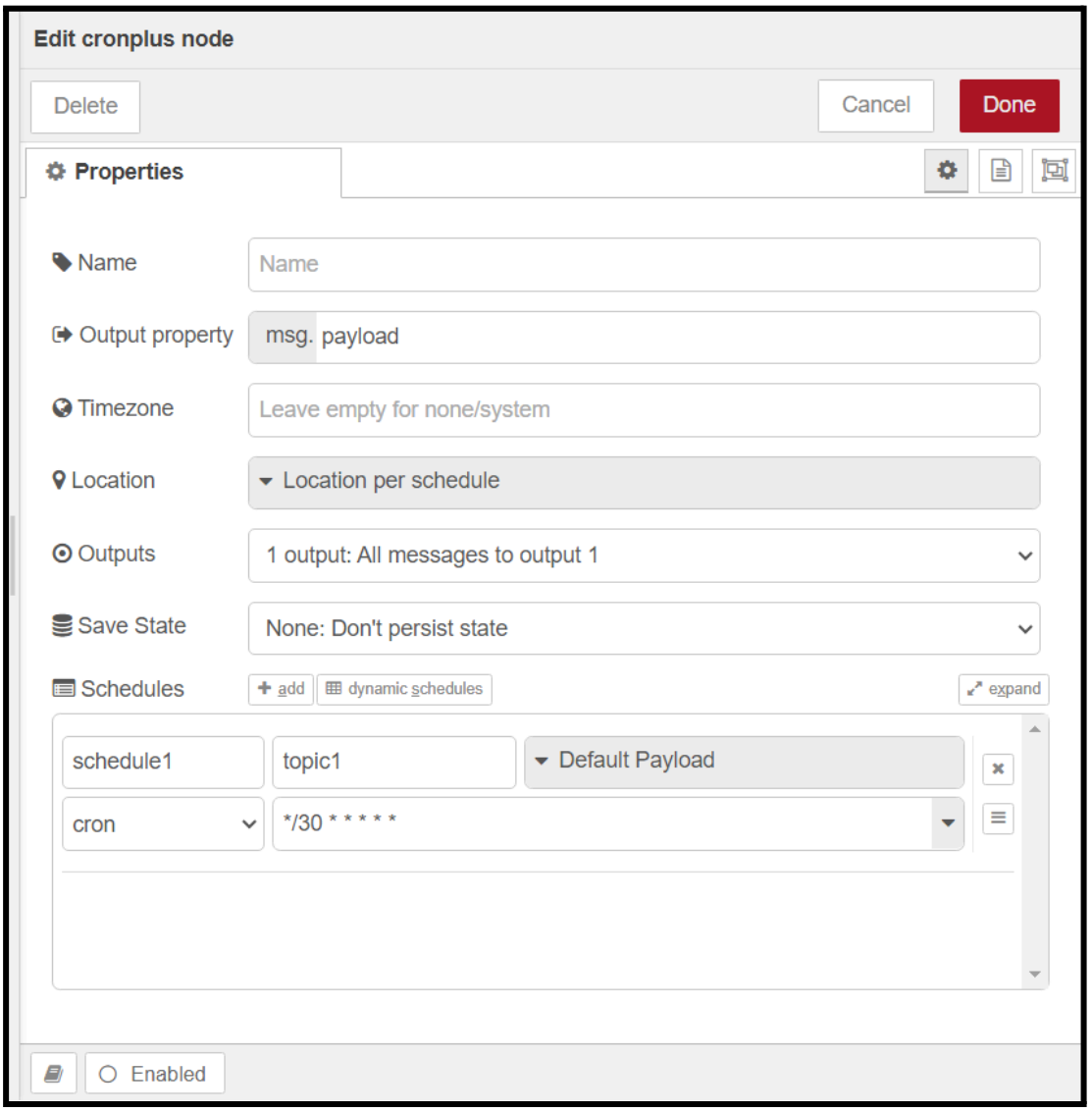
The following will be the instructions for configuring the nodes added into the workflow. You can double click on the nodes to configure them.

1. **Button (Start & End)**



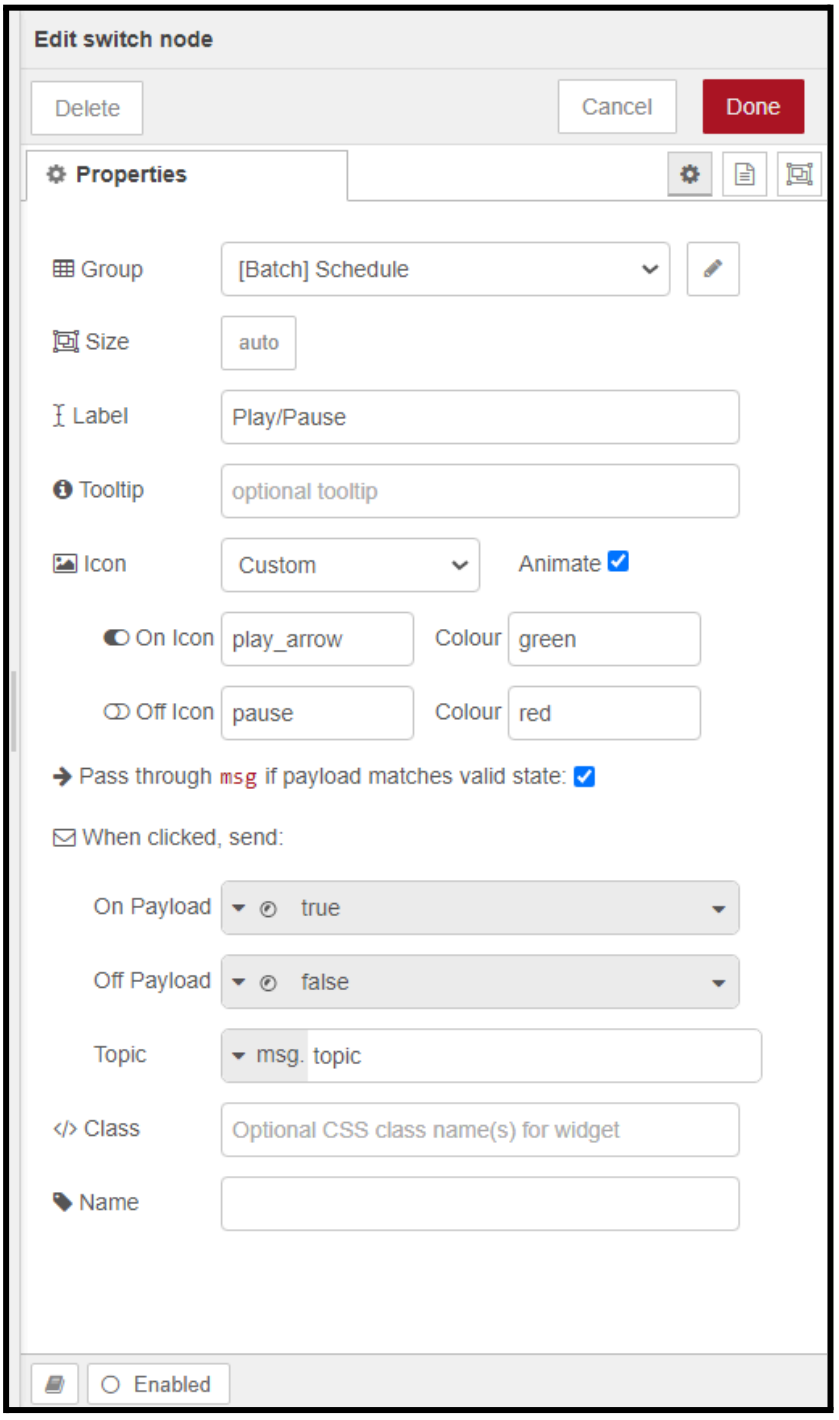
These two button nodes have similar but opposite configurations, one for setting the payload to true and the other setting the payload to false.

2. cronplus



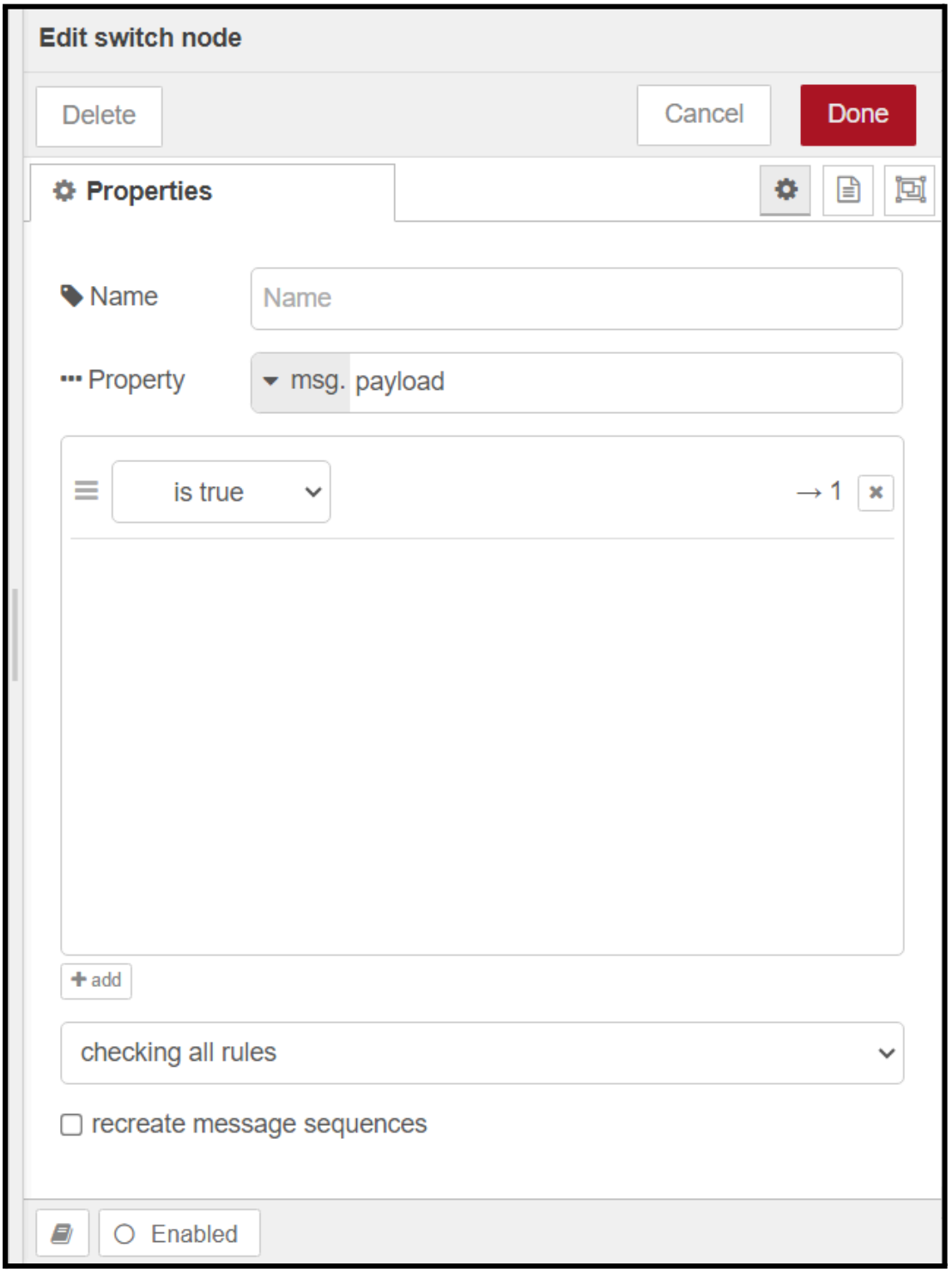
This cronplus node is to schedule the injection of a payload to start a flow and you just need to edit the schedule to "***/30 * * * * ***" for injection at every 30 second mark or you could choose your preferred schedules.

3. **Ui_switch (Play/Pause)**



This (Play/Pause) ui_switch node is only to add a switch to the user interface, no specific configurations to be made. Icon customization is optional.

4. **Switch**



This switch node is to route messages based on their property values and you only need to add for “is true” for value rules.

5. **Function (Get Array)**

```
msg.payload = flow.get("Array");
return msg;
```

This Get Array Function node uses javascript codes to get the array from the flow variable that was selected batch files displayed in the second table.

6. **Split**

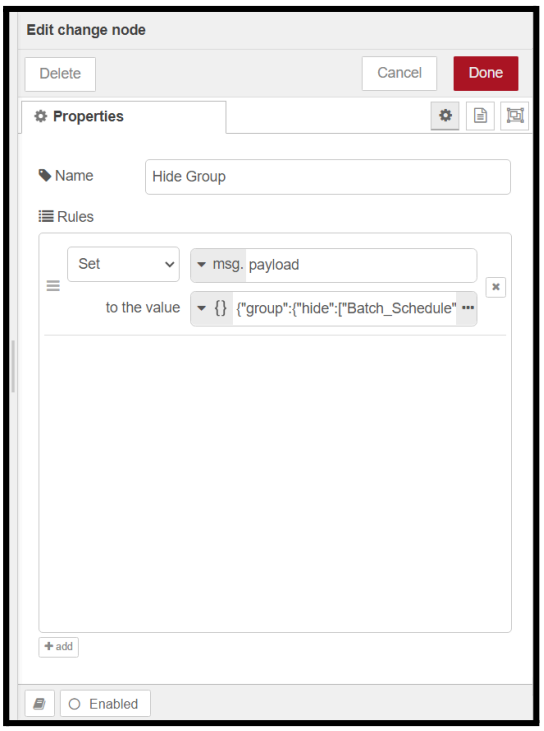
You can simply drag and drop a split node from the node palettes and connect it to your workflow as it will just automatically split the array into separate items after passing through this node.

7. **Exec**

You can simply drag and drop an exec node from the node palettes, connect it to your workflow and check the append msg.payload. This will allow the execution of any batch files sent through by the preceding nodes.

Optional Node(s)

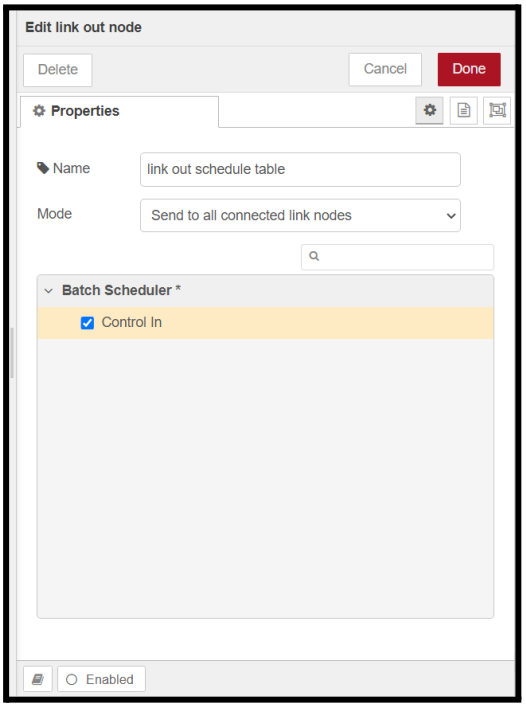
1. Change (Hide Group)




```
{
    "group": {
        "hide": [
            "Batch_Schedule"
        ]
    }
}
```

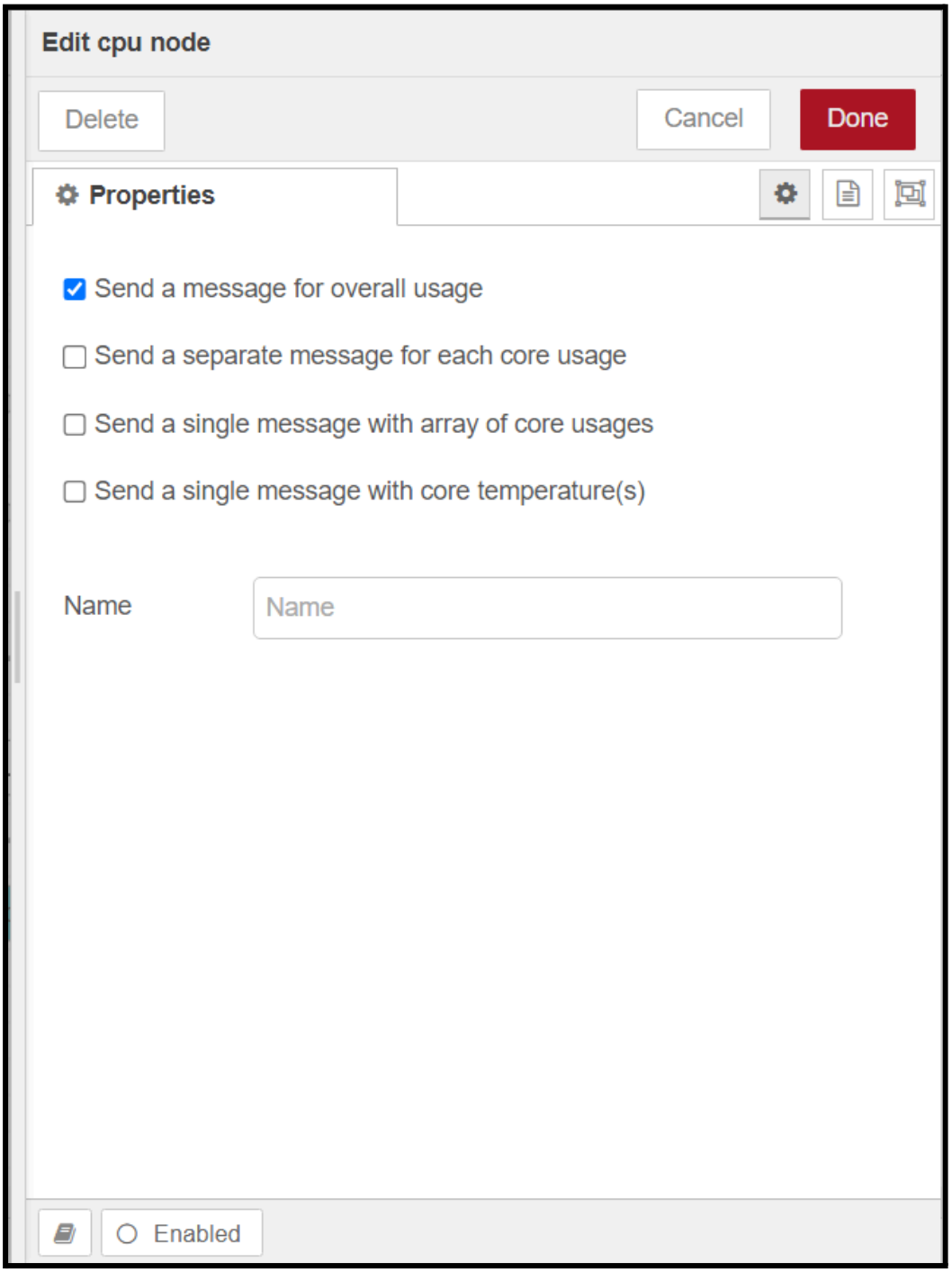
This change node is to hide the group in the UI Dashboard for scheduling buttons. You can click on the ellipses to format json.

2. Link-out



This Link-out node is to connect the flow and make the flow more organised.

3. **CPU (CPU Usage)**



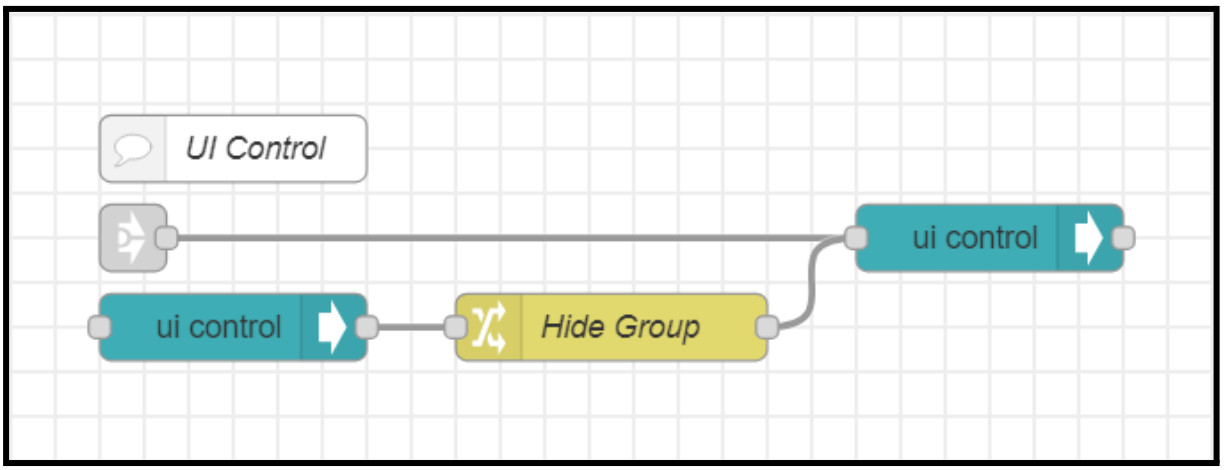
This CPU node monitors the CPU usage and you need to check the “Send a message for overall usage” only so it shows the average usage of your CPU.

4. **Memory (RAM Usage)**

You can simply drag and drop this node into your workflow and connect it to your flow. It will show you an object for totalmem, freemem and memusage which stands for total memory, free memory and memory usage.

UI Control (Optional)

We will complete building the Batch Scheduling Process with the UI Control section. The flow shown below is an example of the Scheduler:



This flow is optional if you did not code for the optional sections above. However, if you're aiming to create software with a better user experience, you can follow along in this section. This flow focuses on enhancing the user interface and user experience of the software.

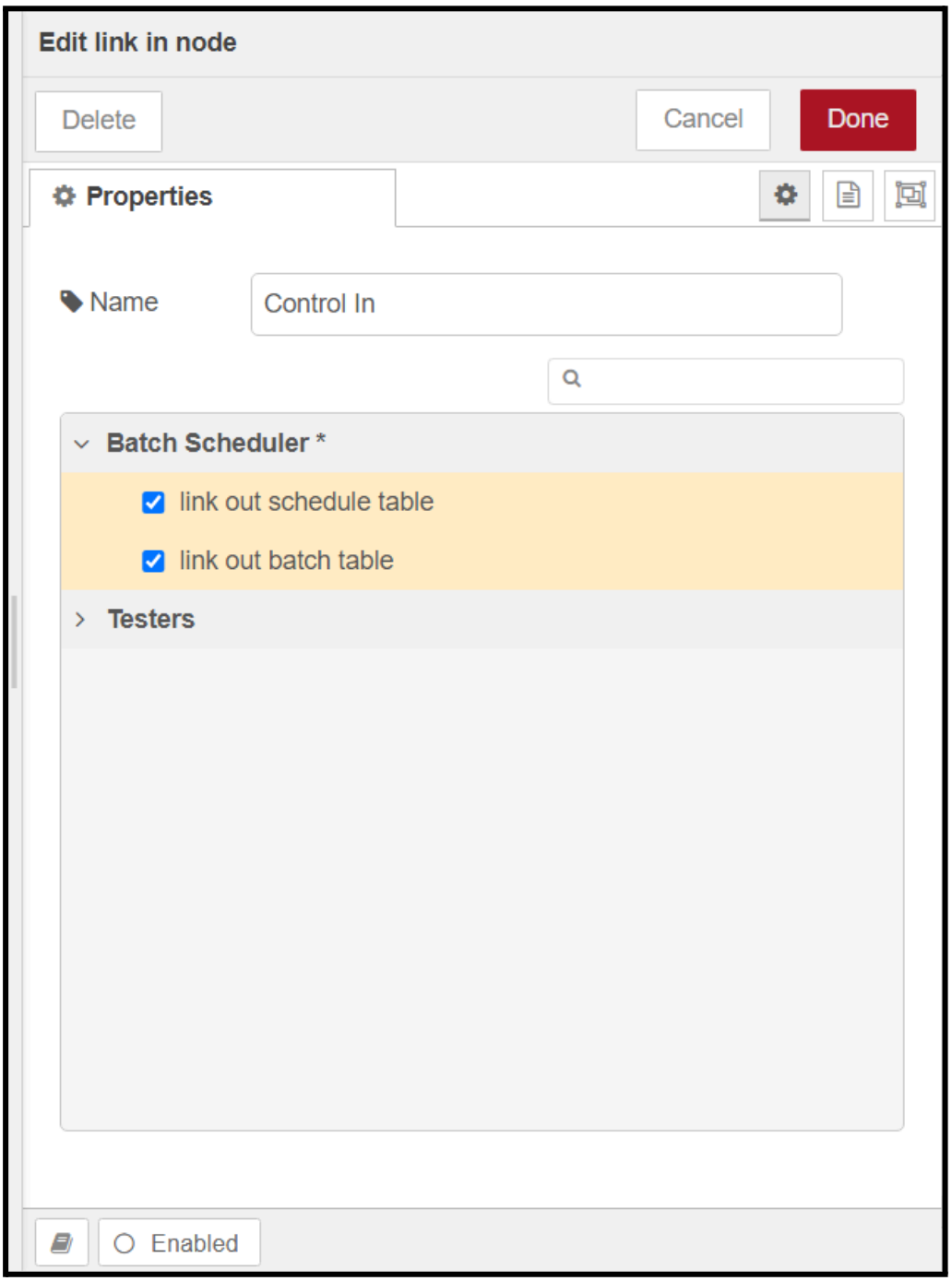
UI Control:

- Utilises a link-in node to enhance the UIUX.
- Employs two ui control nodes:
    - Initialise the start state which hides a group.
    - Update all states when data is sent by the 'hide groups' or 'show groups' commands.
- Features a change function to hide the group.

Node Configurations

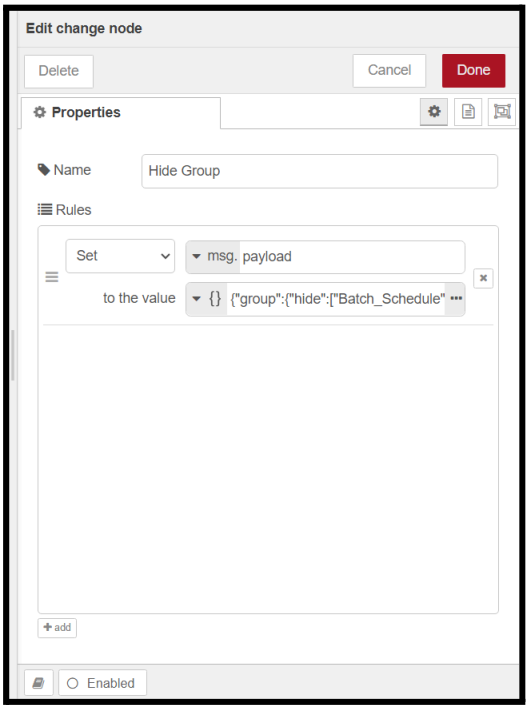
The following will be the instructions for configuring the nodes added into the workflow. You can double click on the nodes to configure them.

1. **Link-in**



This link-in node connects from the link out node and it keeps the flow organised.

2. Change (Hide Group)




```
{
    "group": {
        "hide": [
            "Batch_Schedule"
        ]
    }
}
```

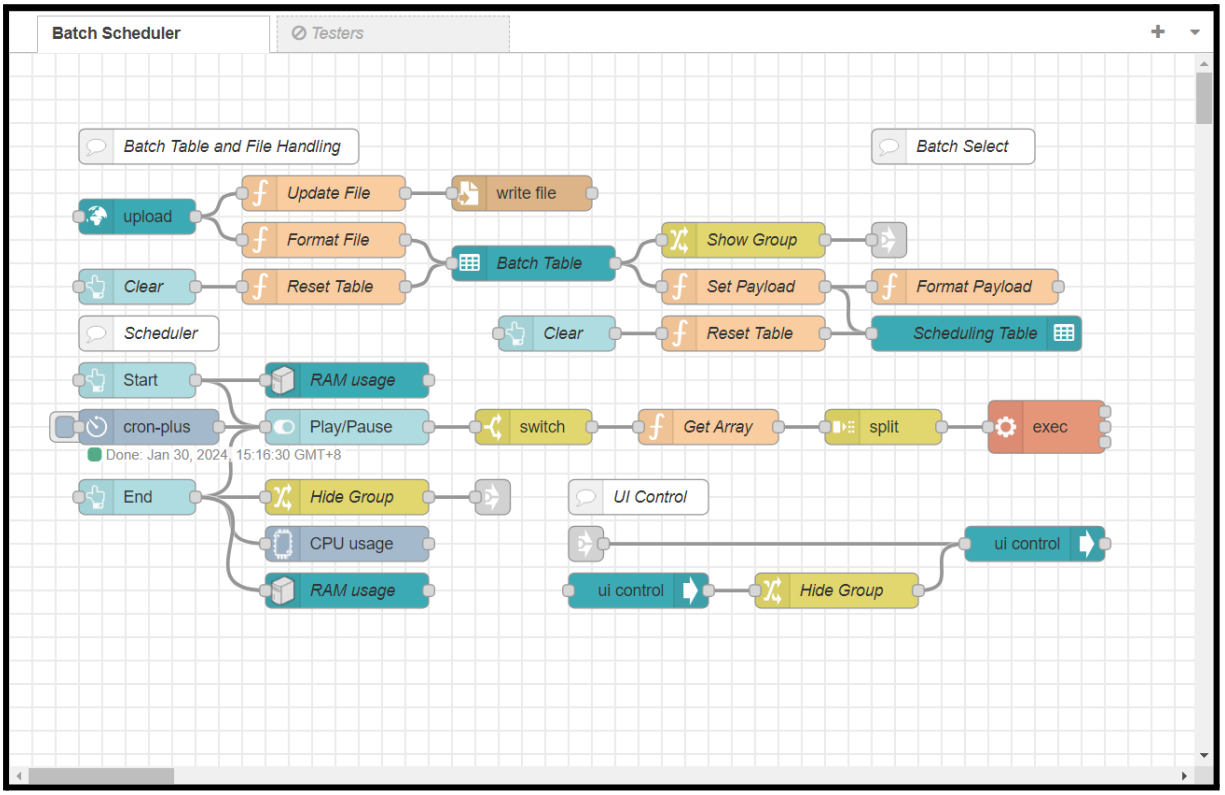
This change node is to hide the group in the UI Dashboard for scheduling buttons. You can click on the ellipses to format json.

3. UI Control

You can simply drag and drop two UI control nodes and connect them as the sample flow above. It should do the trick.

Sample Flow

The sample flow of the Batch Scheduling Process looks as below:

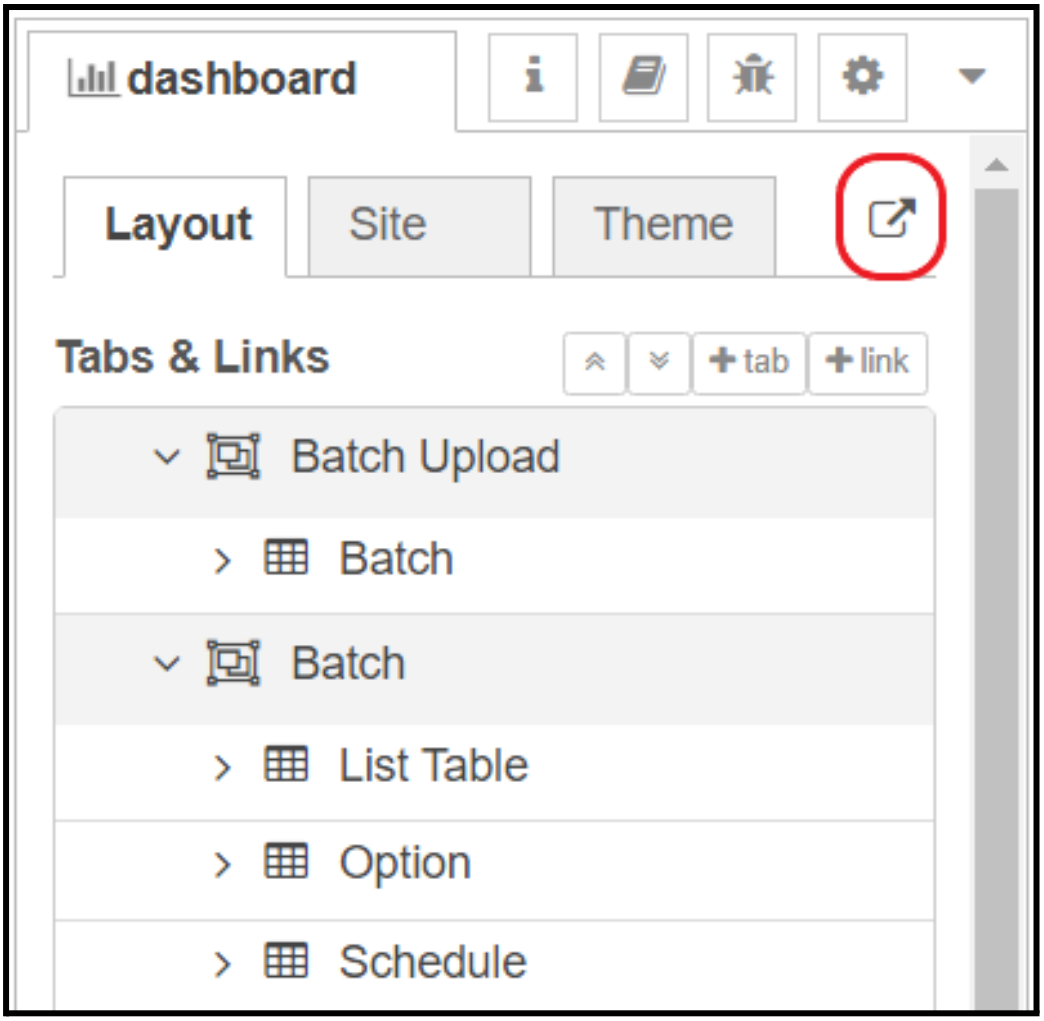


Note : This is just a sample flow for you to build, you can explore ways to improve the dashboard and workflow.

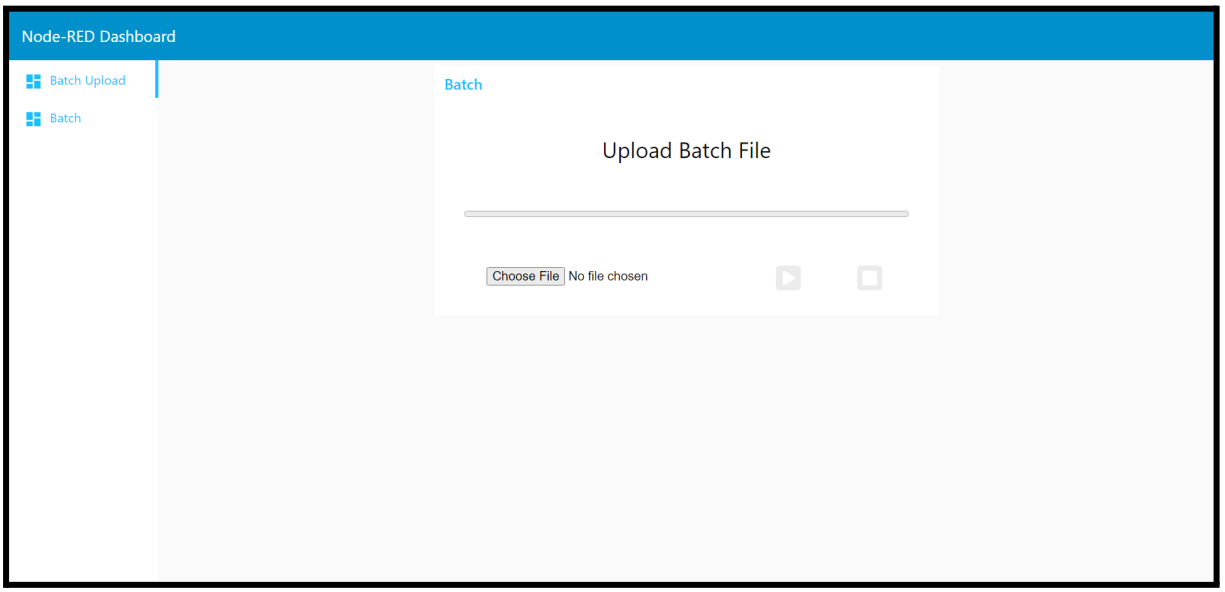
Note : You should always use a debug node to find what causes the error and where the errors come from.

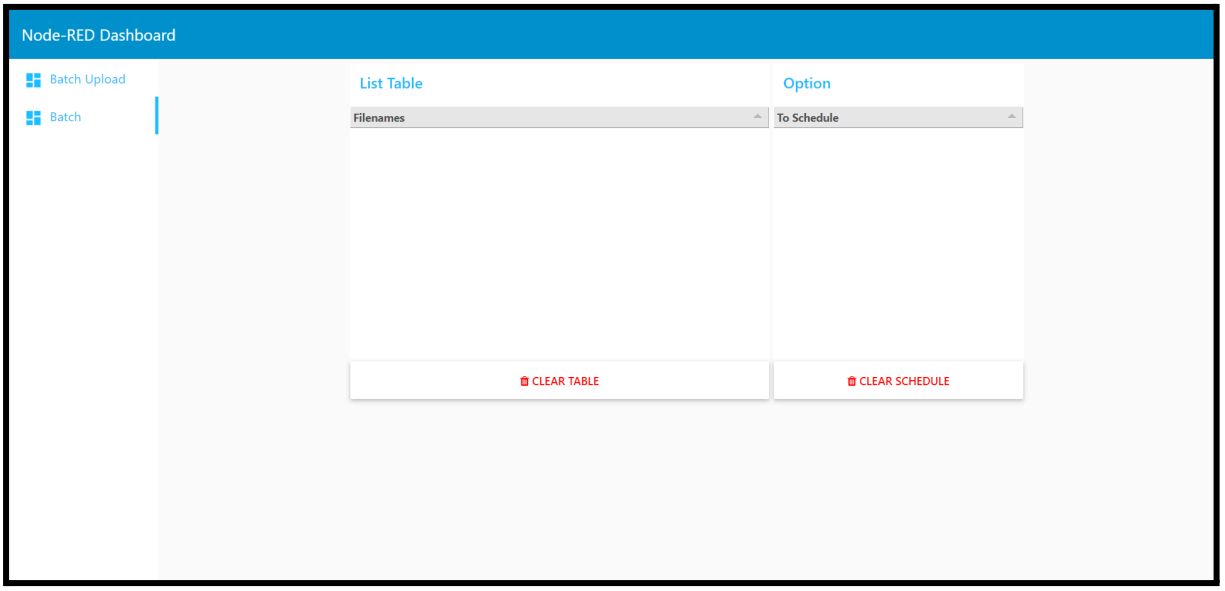
Deploy

Once the flow is successfully built, click on the ‘Deploy’ button to deploy the flow. Afterwards, navigate to the Node-RED Dashboard by clicking the indicated button.



This will bring you to a new tab for the Node-RED Dashboard that looks like this:





Batch Files

The following will be the instructions for configuring the batch files. You will be using them in this Batch Processing Scheduler.

run_edge.bat

```
@echo off
SET FILENAME=run_edge.bat
start msedge
```

Above is an example of a batch file that runs the application Microsoft Edge. When you are creating a batch file, you should always have:

1. *@echo off* to turn off display of commands.
2. *SET FILENAME=yourDesiredFileName.bat* to define your desired filename.
3. *start yourApplication* to launch your application.

The second step is important, especially for the **Nodes Configuration section for Batch Table and File Handling**, you have two function nodes for Update File and Format File that uses the line that compares SET FILENAME to store your filename, allowing easy access within Node-RED.

This line marks the end of this document