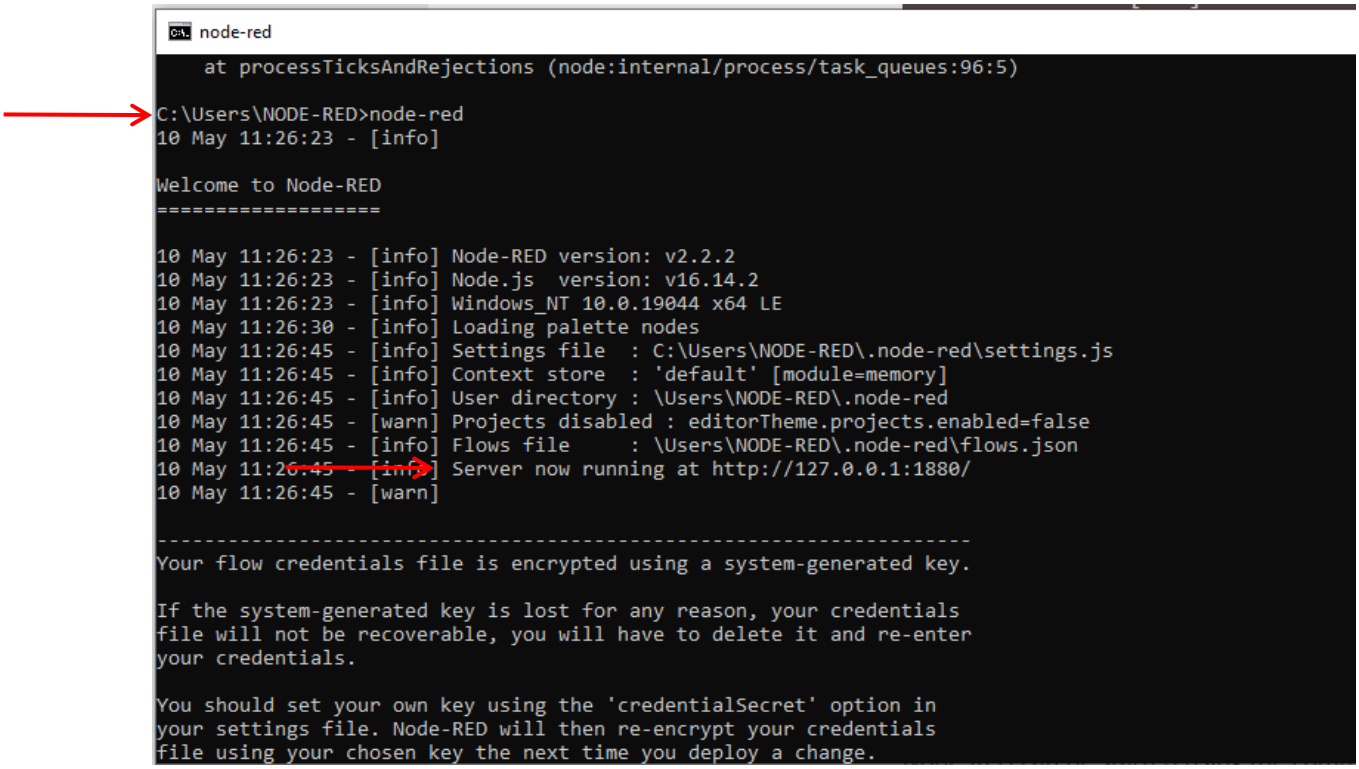
iPLON®
The Infranet Company

FTP-PUSH
NODE-RED PROJECT
MANUAL

MALAVIKA K.V

NODE-RED INSTALLATION SETUP

1. First need to install node-red using [Running Node-RED locally : Node-RED (nodered.org)]
2. Once installed as a global module you can use the node-red command to start Node-RED in your terminal. You can use Ctrl-C or close the terminal window to stop Node-RED.

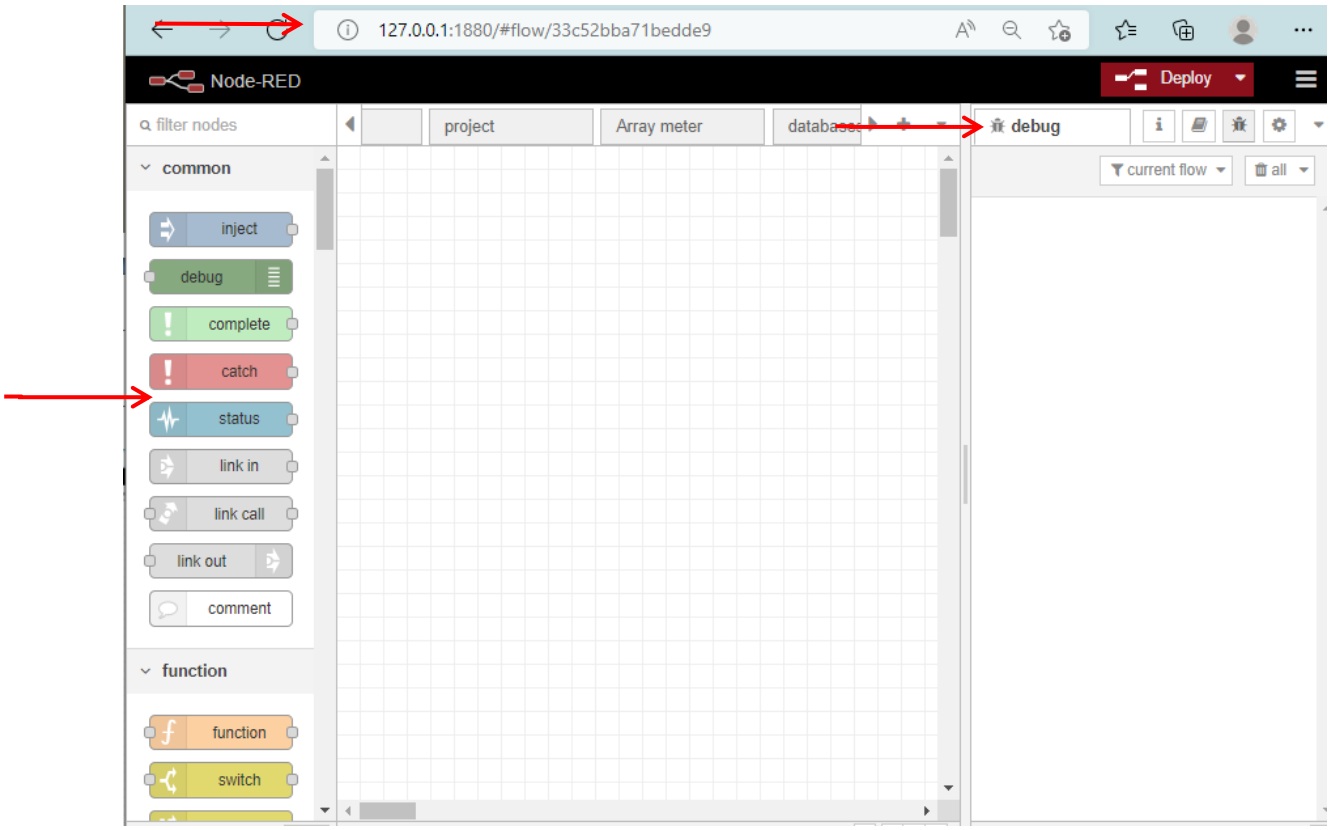


3. Installation using docker command:

   # docker run -it -p 1880:1880 -v node_red_data:/data --name mynodered nodered/node-red

4. Access the editor with Node-RED running, open the editor in a web browser.If you are using a browser on the same computer that is running Node-RED, you can access it with the url: http://localhost:1880.If you are using a browser on another computer, you will need to use the ip address of the computer running Node-RED: http://<ip-address>:1880.

5. A node-red workspace will open on the left side of workspace there will nodes to work on and in right side debug window to show outputs.



FTP PUSH PROJECT REQUIREMETS

STEP 1: Collect every 15 min data of every block device,control room PAC,PDC,CUF,SOLAR_RADIATION,SOLAR_RADIATION_TILT,AIR AMP,MODULE_TEMP,WIND_SPEED,WIND_DIRECTION,HUMDITIY tags from medak and sircilla site and save this as csv file in local system for one full day with a file name of " [plant name]_[todays date".csv [eg: Medak_2022_12_22.csv].

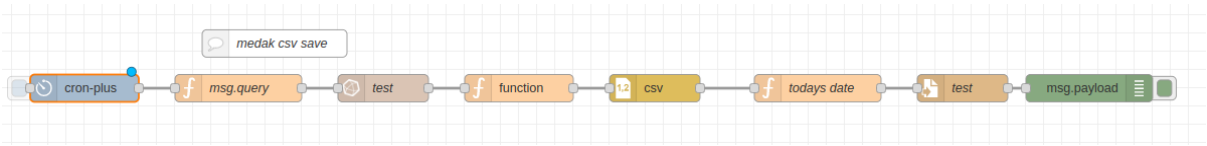
STEP 2: Sent this csv file to a server in every 15 min through ftp push.

STEP 3: Compare local file path and ftp directory path to check any unsent files due to internet connection loss if there is any sent that to ftp server.

NODE-RED FLOW FOR STEP 1:

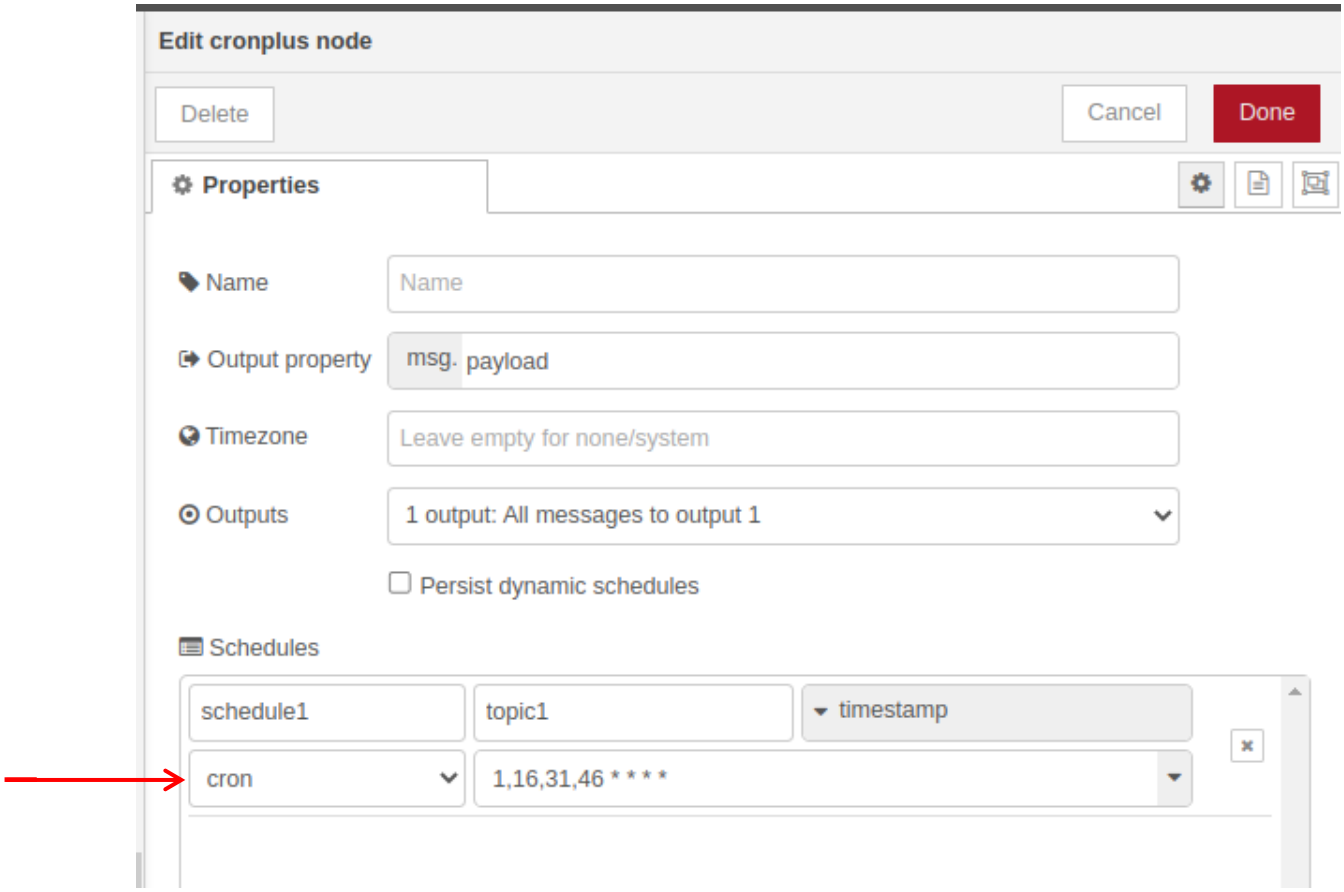
Collect every 15 min data of every block device,control room PAC,PDC,CUF,SOLAR_RADIATION,SOLAR_RADIATION_TILT,AIR_AMP,MODULE_TEMP,WIND_SPEED,WIND_DIRECTION,HUMDITIY tags from medak and sircilla site and save this as csv file in local system for one full day with a file name of "[plant name]_[todays date]".csv [eg: Medak_2022_12_22.csv].

NODE-RED FLOW FOR STEP 1:



Cron plus Node:

1. The cron plus node allows you to inject messages into a flow, either by clicking the button on the node, or setting a time interval between injects.
2. You have to install this node package externally (node-red-contrib-cron-plus)from manage palette feature(click 3 lines symbol in the top right corner in the node-red workspace)
3. Select the newly added Inject node to see information about its properties and a description of what it does in the Information sidebar pane.



Here we are setting up the triggering time to every 1st ,16th,31st,46th min repeat that will trigger the corresponding flow in the pre-set time everyday.

Function node : Here we used function node to give msg.query output to influxdb in node. If we need to query for a variable start and stop time we can use javascript logic to set up start and stop time here we need midnight timestamp[12.00AM] of that day as start time and triggering time as end timestamp

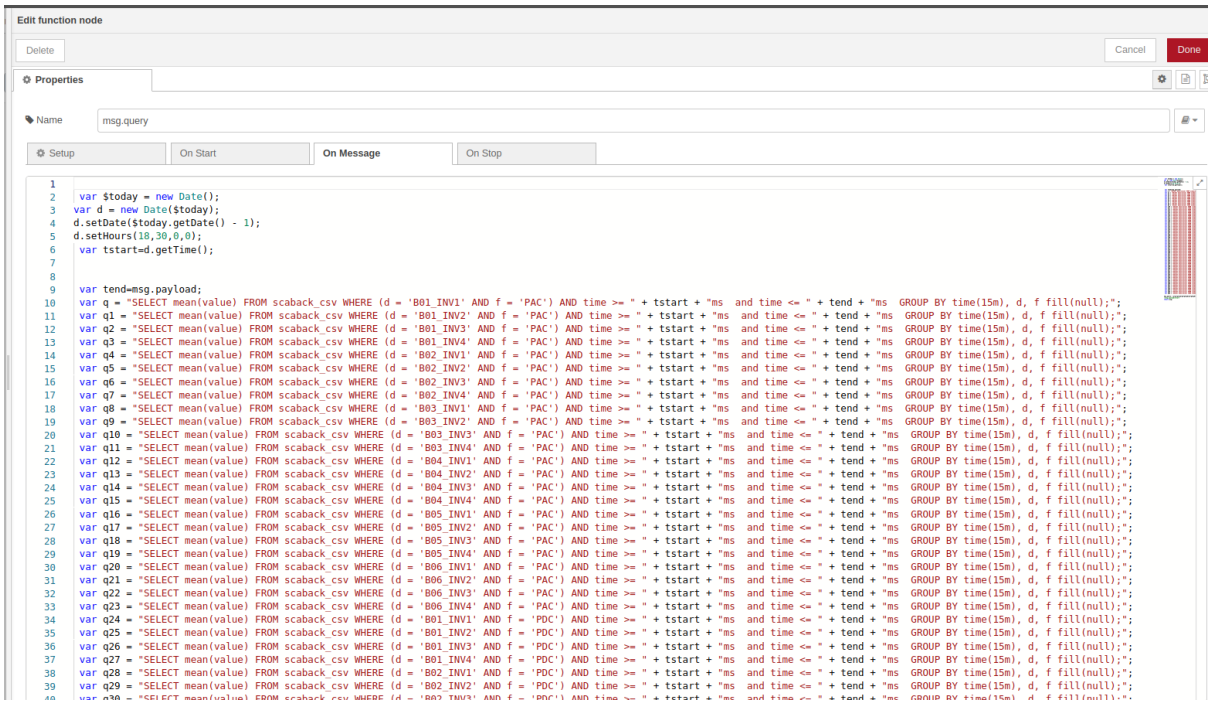
```
var $today = new Date();

var d = new Date($today);

d.setDate($today.getDate() - 1);
d.setHours(18,30,0,0);

var tstart=d.getTime();
var tend=msg.payload;
```

Then add query of different devices we want one by one in double quotes

Eg: var q = "SELECT mean(value) FROM scaback_csv WHERE (d = 'B01_INV1' AND f = 'PAC') AND time >= " + tstart + "ms and time <= " + tend + "ms GROUP BY time(15m), d, f fill(null);";

after add all devices like this add all the query to msg.query

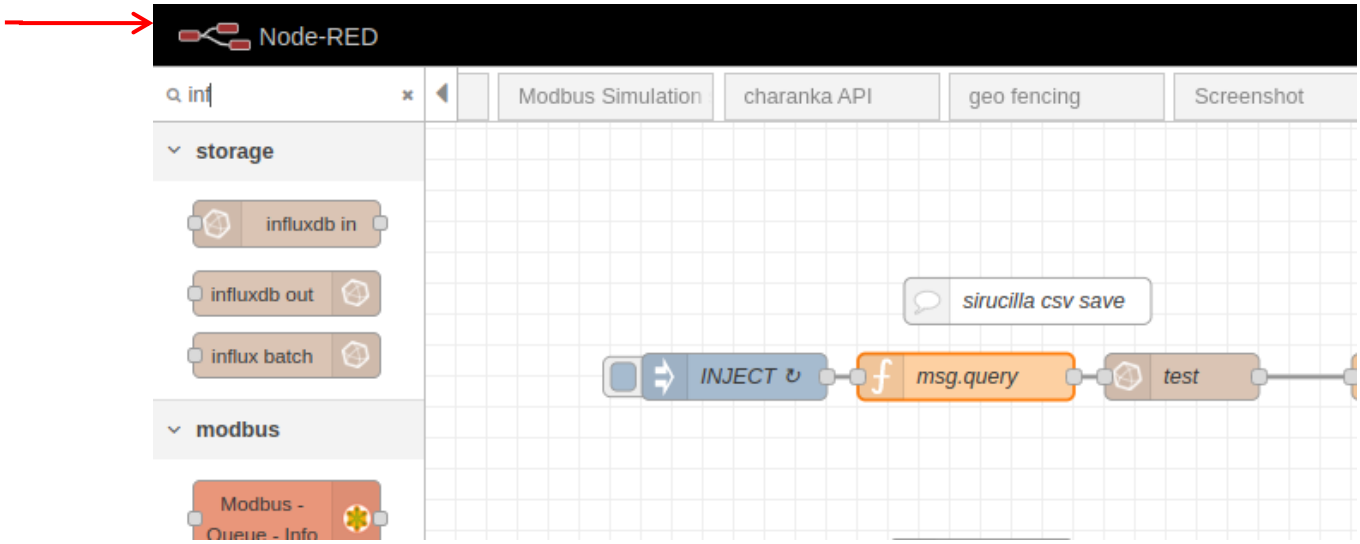
msg.query=q1+q2+…….+q81 like that



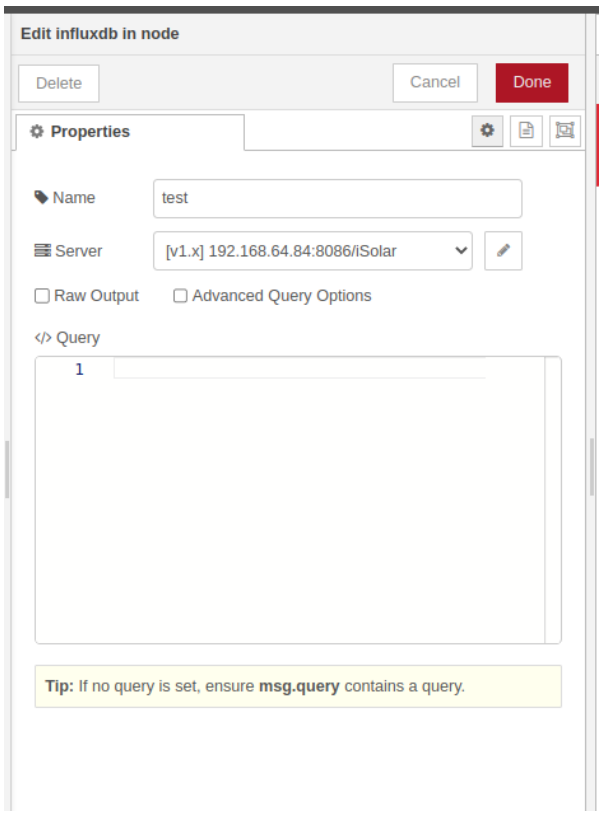
Influxdb in node (node-red-contrib-influxdb (node) - Node-RED (nodered.org)**):**

To access this node we have to install the node package(node-red-contrib-influxdb)from manage pallette feature(click 3 lines symbol in the top right corner in the node-red workspace)

Nodes to query data from an influxdb time series database. Supports InfluxDb versions 1.x to 2.0.

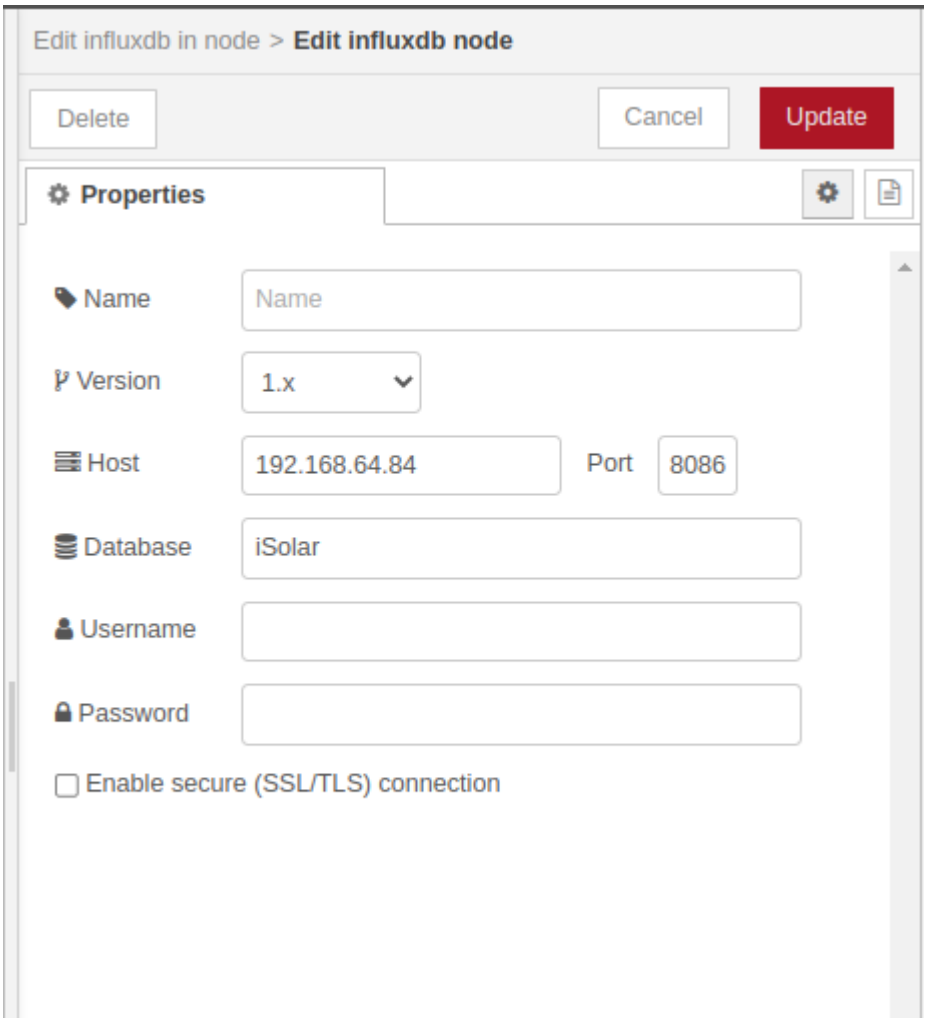


double click on the influx db node



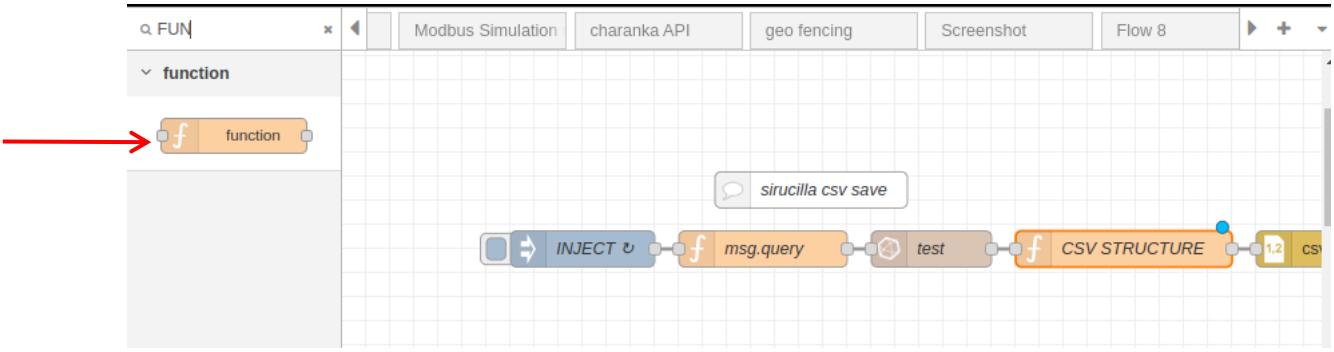
keep the query box blank if you are giving msg.query through a function node

Then click on the pencil icon to add details of the new server

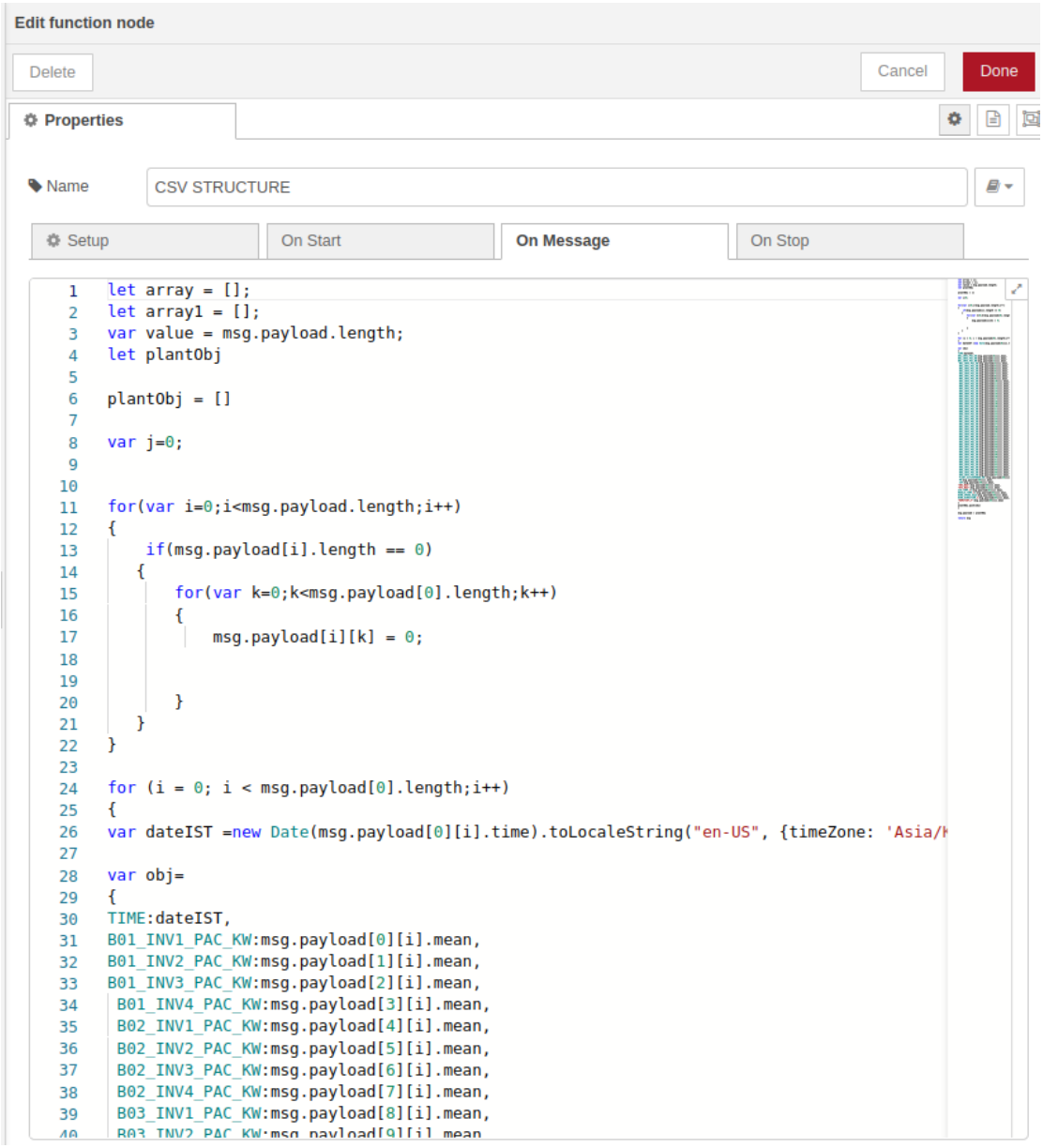


Function node: The Function node allows JavaScript code to be run against the messages that are passed through it.

The message is passed in as an object called msg. By convention it will have a msg.payload property containing the body of the message. Other nodes may attach their own properties to the message, and they should be described in their documentation.



Feed the javascript code on the message box

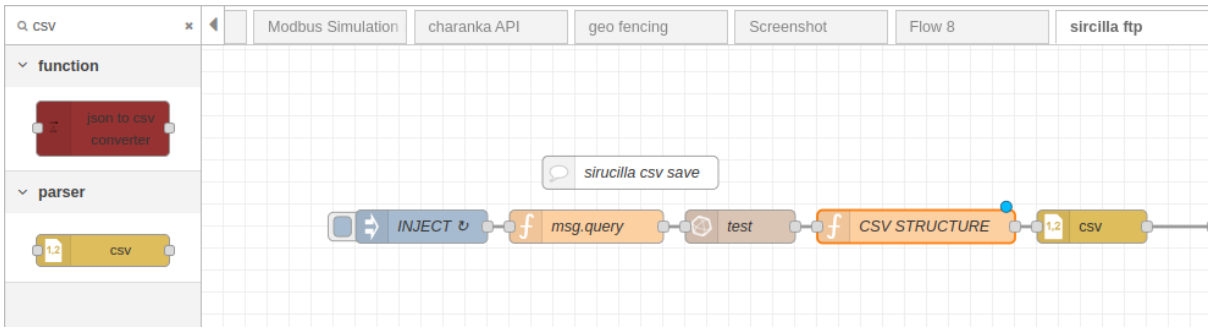


```
let array = [];

let array1 = [];
var value = msg.payload.length;
let plantObj

plantObj = []

var j=0;
for(var i=0;i<msg.payload.length;i++)
{
if(msg.payload[i].length == 0)
{
for(var k=0;k<msg.payload[0].length;k++)
{
msg.payload[i][k] = 0;
}
}
}
for (i = 0; i < msg.payload[0].length;i++)
{
var dateIST =new Date(msg.payload[0][i].time).toLocaleString("en-US", {timeZone: 'Asia/Kolkata'});

var obj=
{
TIME:dateIST,
B01_INV1_PAC_KW:msg.payload[0][i].mean,
B01_INV2_PAC_KW:msg.payload[1][i].mean,
B01_INV3_PAC_KW:msg.payload[2][i].mean,
B01_INV4_PAC_KW:msg.payload[3][i].mean,
B02_INV1_PAC_KW:msg.payload[4][i].mean,
B02_INV2_PAC_KW:msg.payload[5][i].mean,
B02_INV3_PAC_KW:msg.payload[6][i].mean,
B02_INV4_PAC_KW:msg.payload[7][i].mean,
B03_INV1_PAC_KW:msg.payload[8][i].mean,
B03_INV2_PAC_KW:msg.payload[9][i].mean,
B03_INV3_PAC_KW:msg.payload[10][i].mean,
B03_INV4_PAC_KW:msg.payload[11][i].mean,
B04_INV1_PAC_KW:msg.payload[12][i].mean,
B04_INV2_PAC_KW:msg.payload[13][i].mean,
B04_INV3_PAC_KW:msg.payload[14][i].mean,
B04_INV4_PAC_KW:msg.payload[15][i].mean,
B05_INV1_PAC_KW:msg.payload[16][i].mean,
B05_INV2_PAC_KW:msg.payload[17][i].mean,
B05_INV3_PAC_KW:msg.payload[18][i].mean,
B05_INV4_PAC_KW:msg.payload[19][i].mean,
B06_INV1_PAC_KW:msg.payload[20][i].mean,
```

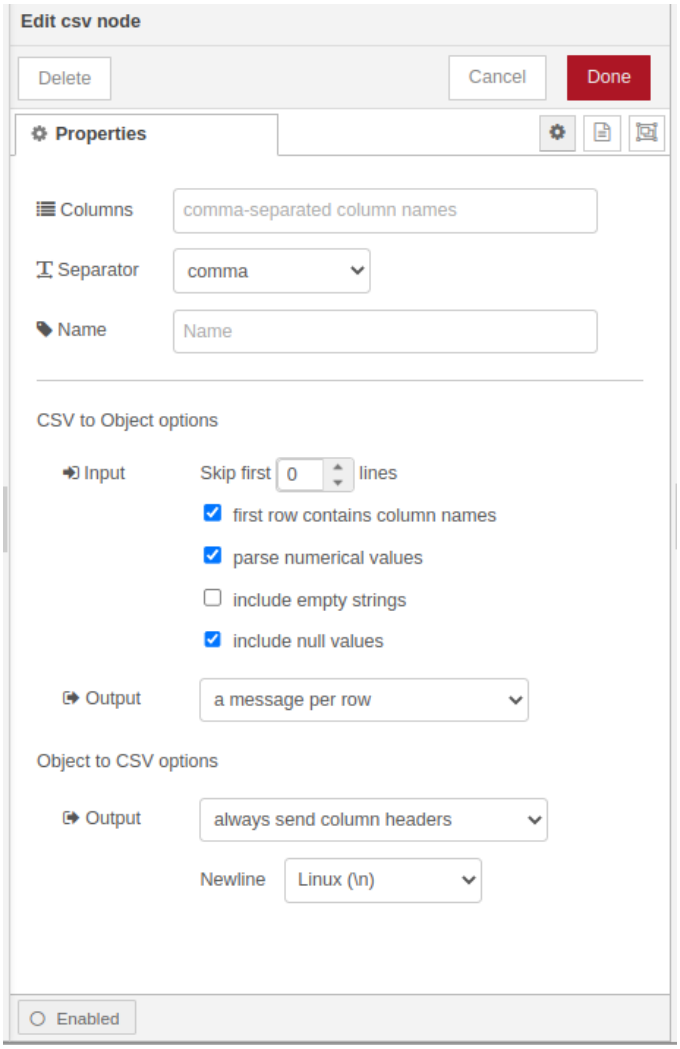
```
B06_INV2_PAC_KW:msg.payload[21][i].mean,
B06_INV3_PAC_KW:msg.payload[22][i].mean,
B06_INV4_PAC_KW:msg.payload[23][i].mean,
B01_INV1_PDC_KW:msg.payload[24][i].mean,
B01_INV2_PDC_KW:msg.payload[25][i].mean,
B01_INV3_PDC_KW:msg.payload[26][i].mean,
B01_INV4_PDC_KW:msg.payload[27][i].mean,
B02_INV1_PDC_KW:msg.payload[28][i].mean,
B02_INV2_PDC_KW:msg.payload[29][i].mean,
B02_INV3_PDC_KW:msg.payload[30][i].mean,
B02_INV4_PDC_KW:msg.payload[31][i].mean,
B03_INV1_PDC_KW:msg.payload[32][i].mean,
B03_INV2_PDC_KW:msg.payload[33][i].mean,
B03_INV3_PDC_KW:msg.payload[34][i].mean,
B03_INV4_PDC_KW:msg.payload[35][i].mean,
B04_INV1_PDC_KW:msg.payload[36][i].mean,
B04_INV2_PDC_KW:msg.payload[37][i].mean,
B04_INV3_PDC_KW:msg.payload[38][i].mean,
B04_INV4_PDC_KW:msg.payload[39][i].mean,
B05_INV1_PDC_KW:msg.payload[40][i].mean,
B05_INV2_PDC_KW:msg.payload[41][i].mean,
B05_INV3_PDC_KW:msg.payload[42][i].mean,
B05_INV4_PDC_KW:msg.payload[43][i].mean,
B06_INV1_PDC_KW:msg.payload[44][i].mean,
B06_INV2_PDC_KW:msg.payload[45][i].mean,
B06_INV3_PDC_KW:msg.payload[46][i].mean,
B06_INV4_PDC_KW:msg.payload[47][i].mean,
PLANT_ACTIVEPOWER_KW:(msg.payload[48][i].mean)*1000,
PR:msg.payload[49][i].mean,
CUF:msg.payload[50][i].mean,
"GHI_Wm2":msg.payload[60][i].mean,
"GTI_Wm2":msg.payload[61][i].mean,
AIR_TEMP_ºC:msg.payload[62][i].mean,
MODULE_TEMP_ºC:msg.payload[63][i].mean,
WIND_SPEED_Kmph:msg.payload[64][i].mean,
WIND_DIRECTION_º:msg.payload[65][i].mean,
"HUMITIDY_%":msg.payload[66][i].mean
}
plantObj.push(obj)
}
msg.payload = plantObj

return msg
```

This javascript code is used to check every message in the array of output getting from influx output using a for loop function and to take make a csv struture as per requirement

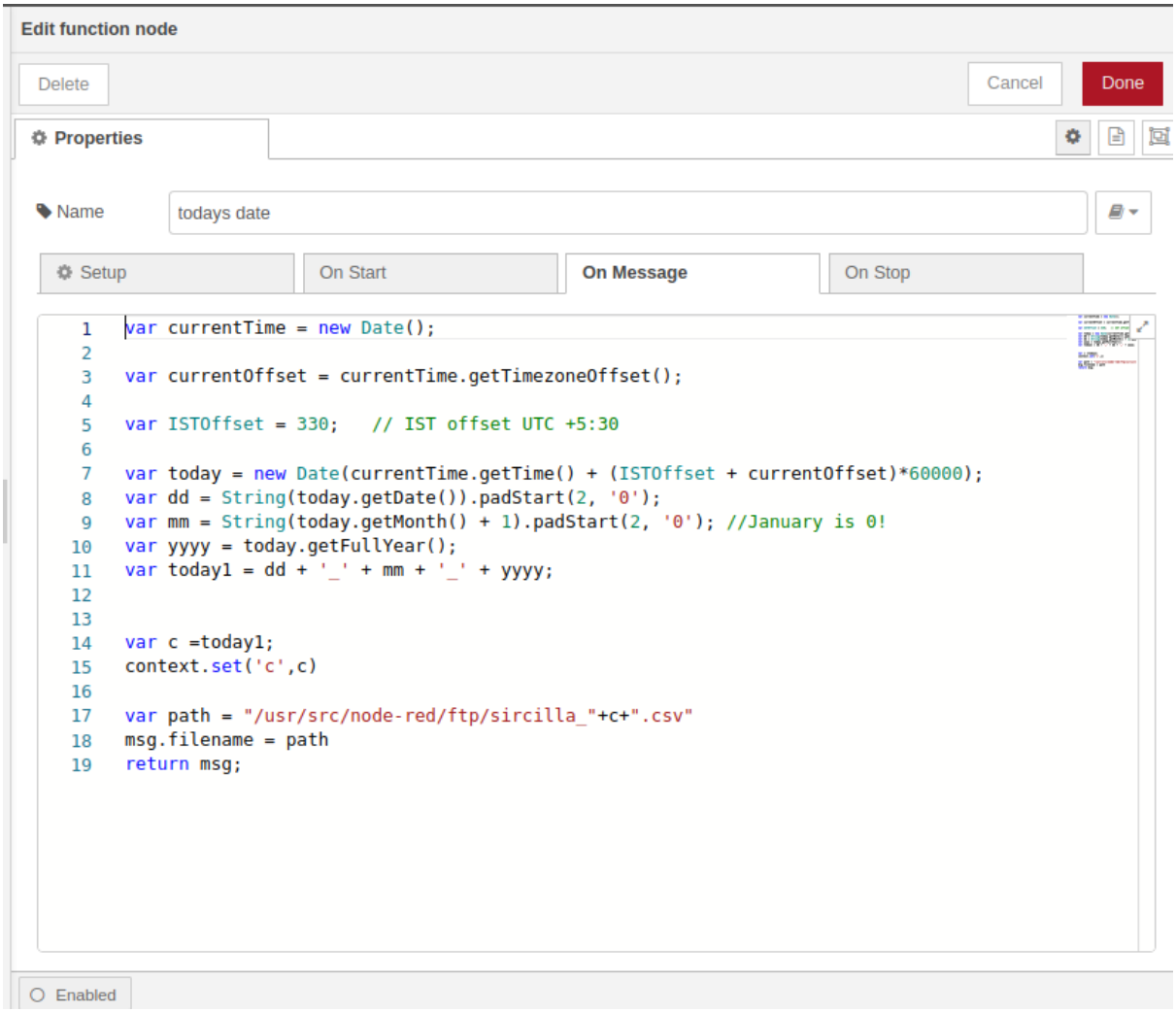
csv node:



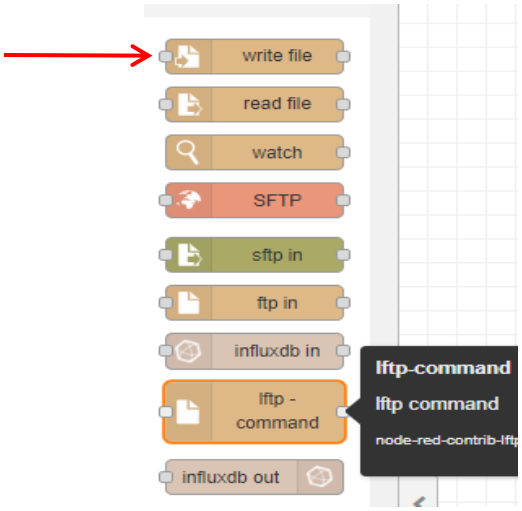
Tick the boxes like this and take the output as a message per row

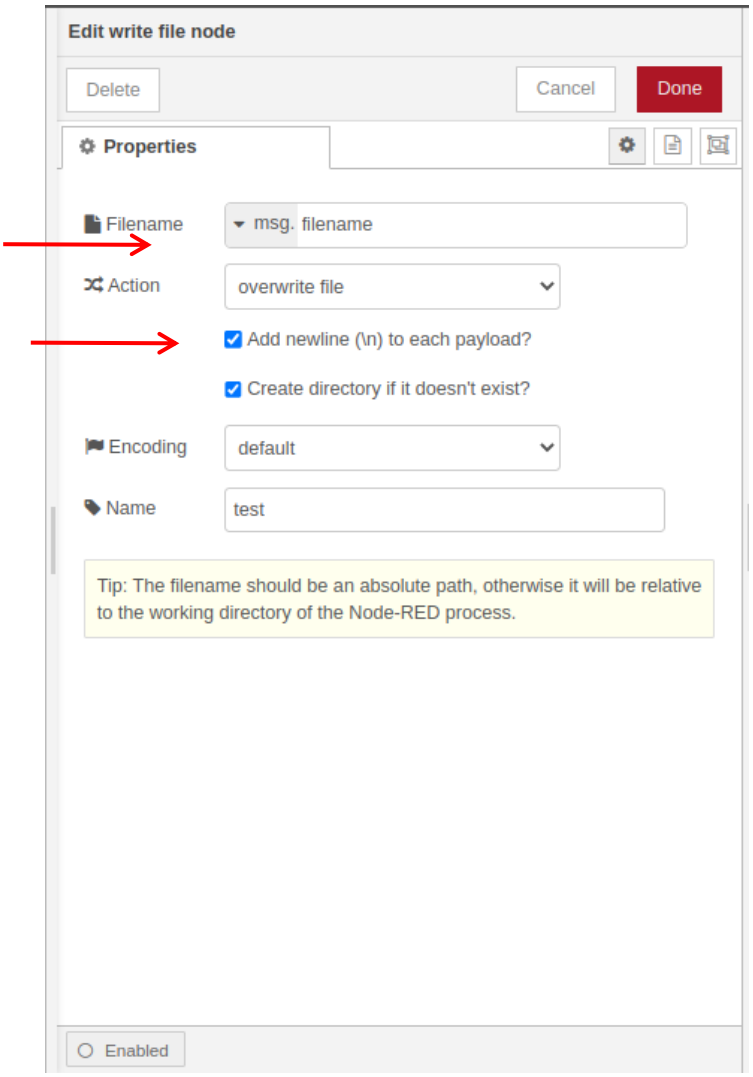


While writing the code in other devices the file path will be different that has to change accordingly.



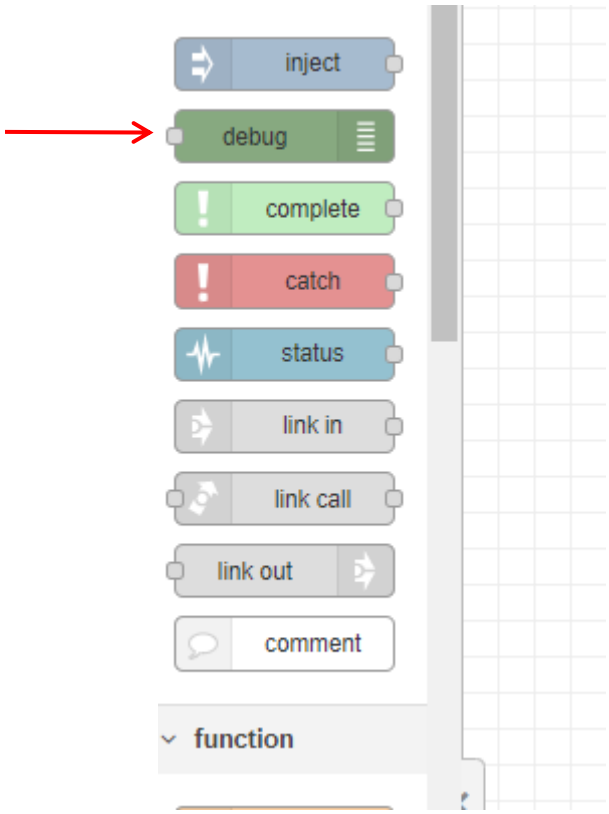
Write file node:





No need to fill the file name box because we already giving it through function node

Debug node:

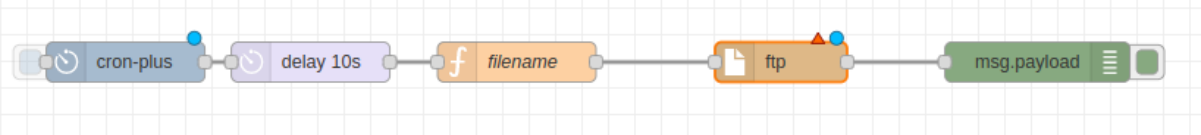


The Debug node causes any message to be displayed in the Debug sidebar. By default, it just displays the payload of the message, but it is possible to display the entire message object.

1 Click the Deploy button. With the Debug sidebar tab selected,
2 Click the Inject button to get output if there is no triggering time set.
3 Check the destination of file what we give as path to check whether the csv file created or not

STEP 2: Sent this csv file to a server in every 15 min through ftp push.

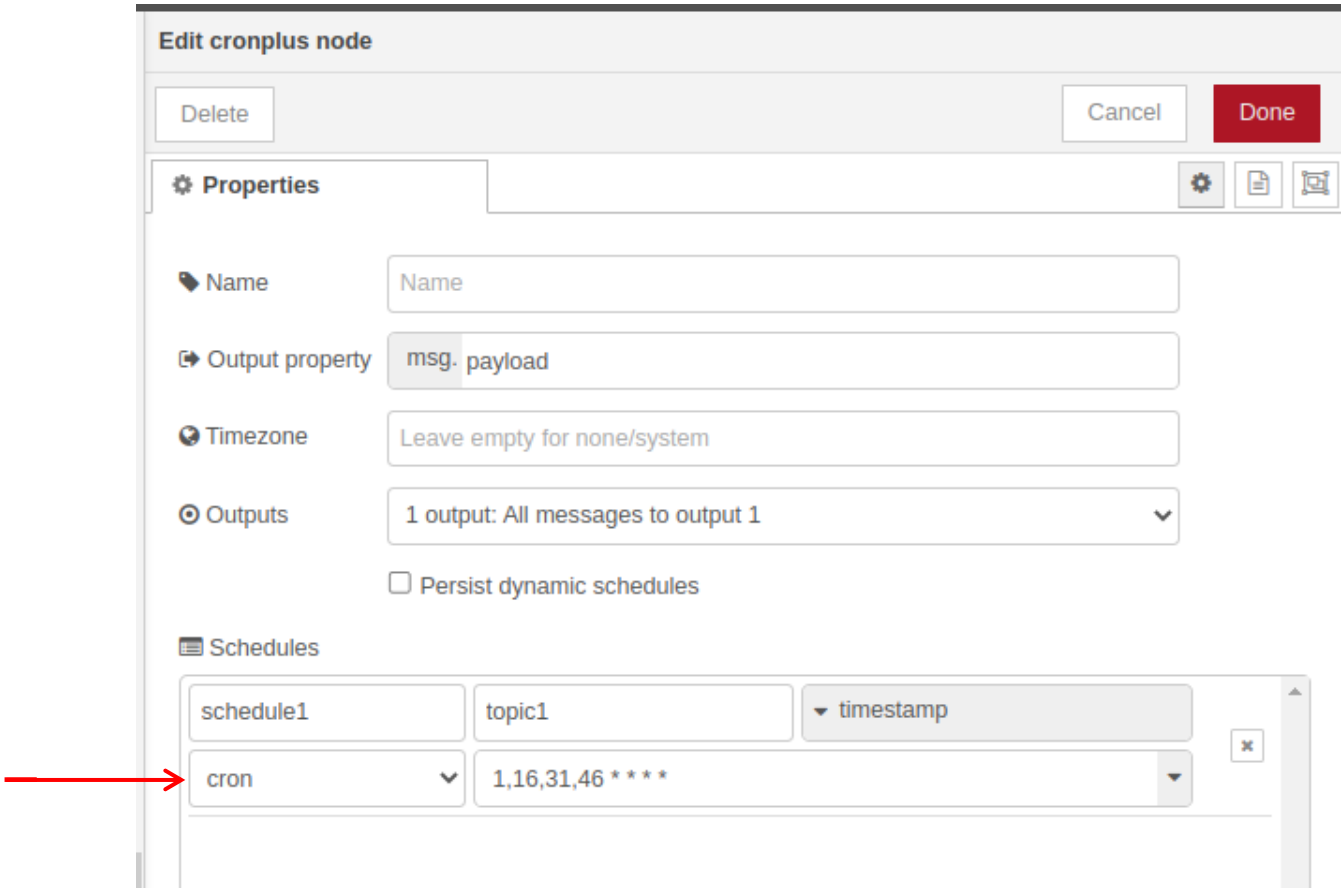
NODE-RED FLOW FOR STEP 2:



Here we are sending the csv file that we saved using the first flow through ftp node

Cron plus Node:

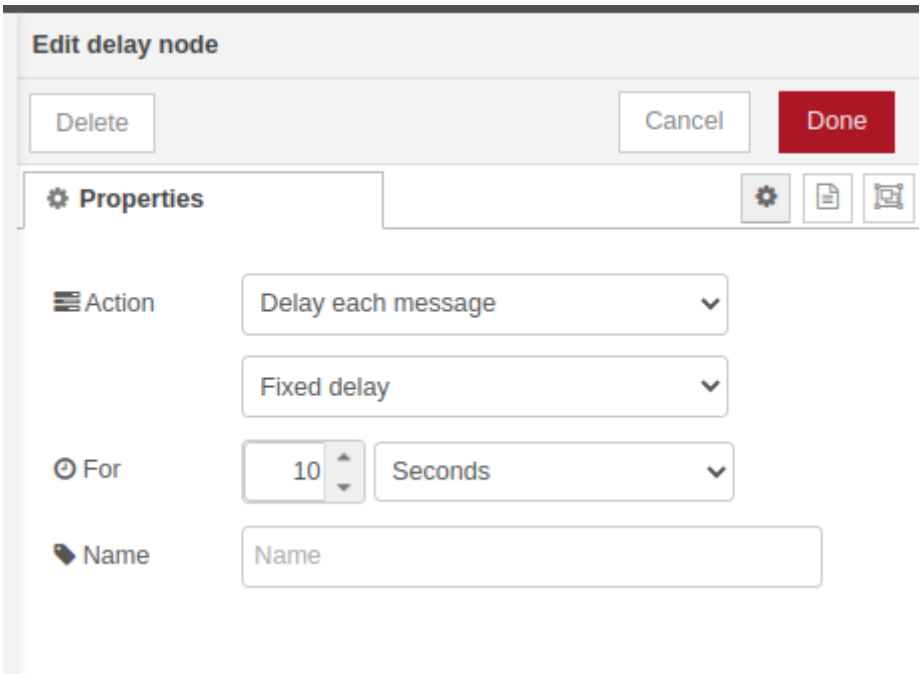
4 The cron plus node allows you to inject messages into a flow, either by clicking the button on the node, or setting a time interval between injects.
5 You have to install this node package externally (node-red-contrib-cron-plus)from manage palette feature(click 3 lines symbol in the top right corner in the node-red workspace)
6 Select the newly added Inject node to see information about its properties and a description of what it does in the Information sidebar pane.



Here we are setting up the triggering time to every 1st ,16th,31st,46th min repeat that will trigger the corresponding flow in the pre-set time everyday.

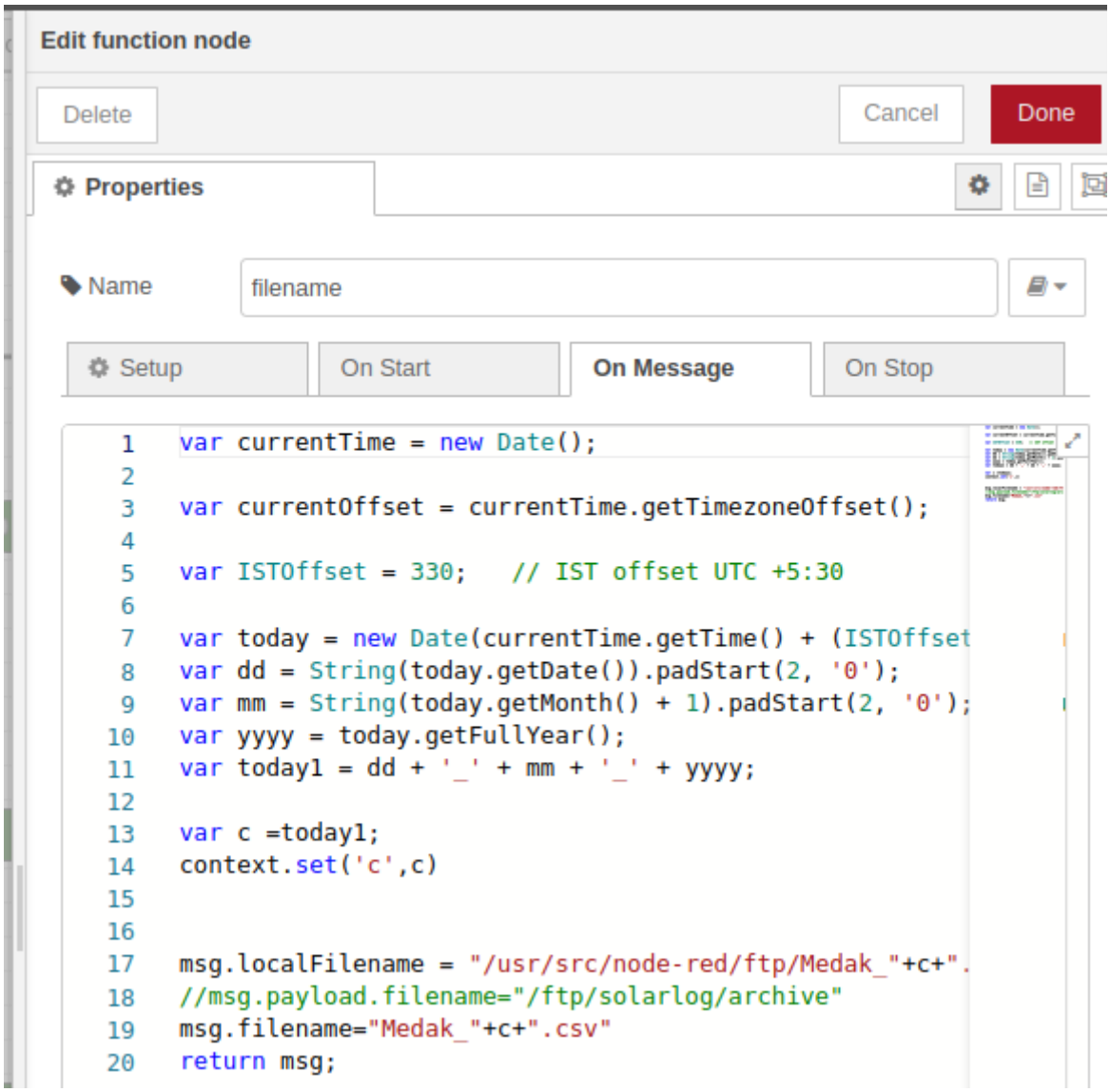
Delay Node:

Cron plus node Then connected to delay node avoid triggering conflit because we after used same triggering in first flow [delay node help us to sent csv file only after it updated in timing]from first flow



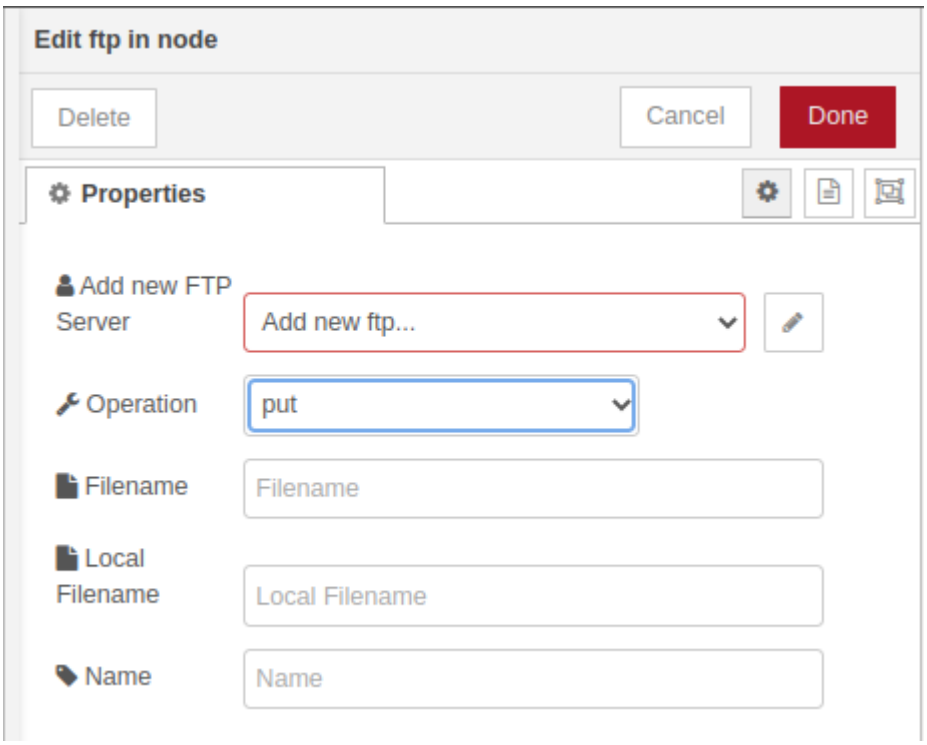
Function node:

Delay node connected to a function node (todays date)for giving file path as today's date or read the same csv file we saved in todays date.



ftp in Node:

Install the node by installing node package(node-red-contrib-ftp)drag and drop the node



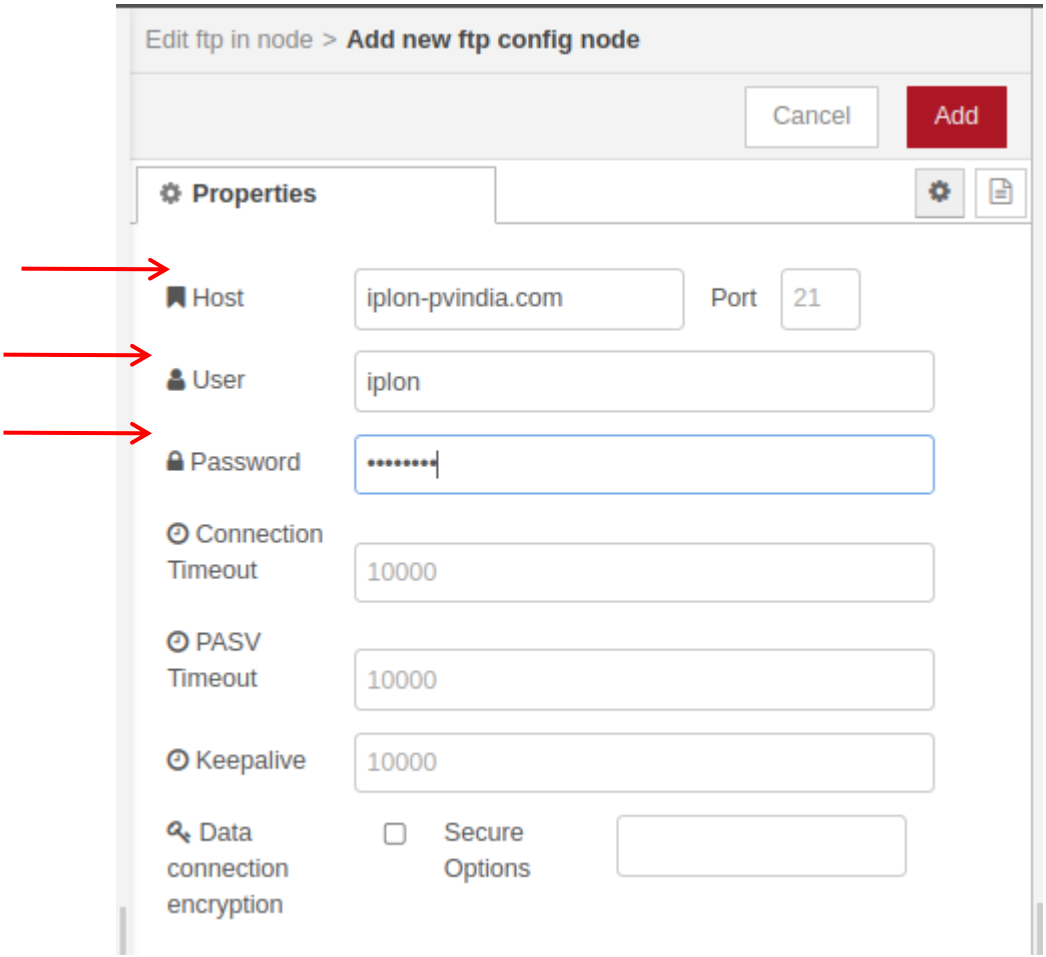
Set Operation –put

Filename - leave this box blank so it will take filename from the function node connected behind it.

Local Filename-leave this box blank so it will take filename from the function node connected behind it.

Setup the server configuration as per below figure (click the pencil icon)enter the username and password to authorize the server.

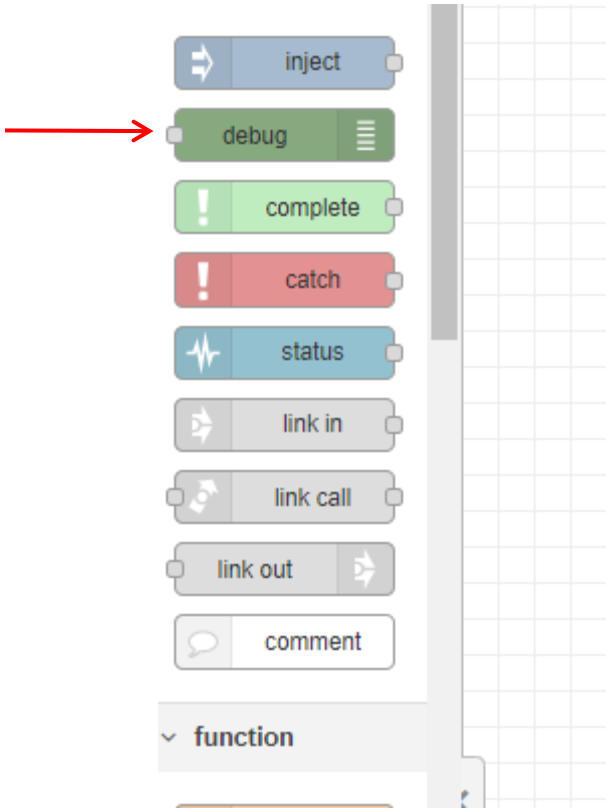
Set port always 21 for FTP



Debug node:

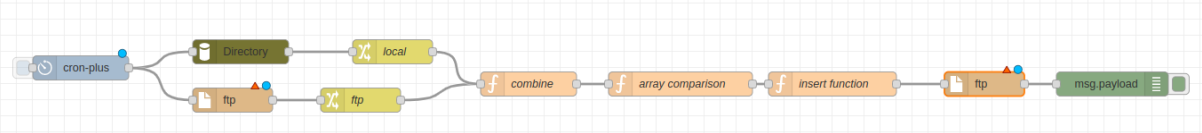
The Debug node causes any message to be displayed in the [Debug sidebar](). By default, it just displays the payload of the message, but it is possible to display the entire message object.

1 Click the Deploy button. With the Debug sidebar tab selected,
2 Click the Inject button to get output if there is no triggering time set.


inject
debug
complete
catch
status
link in
link call
link out
comment
function

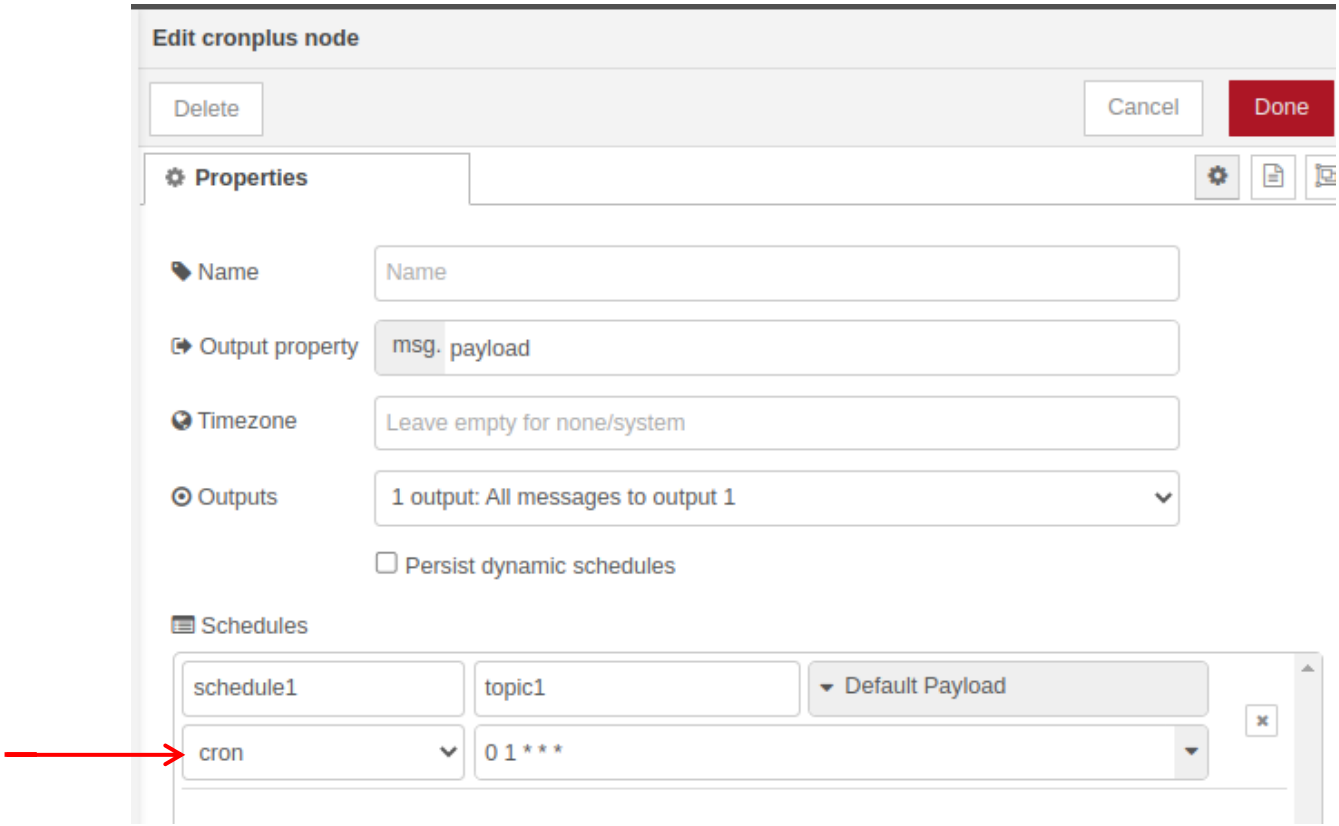
NODE-RED FLOW FOE STEP-3

STEP 3: Compare local file path and ftp directory path to check any unsent files due to internet connection loss if there is any sent that to ftp server.



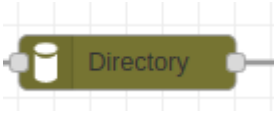
Cron plus Node:

1. The cron plus node allows you to inject messages into a flow, either by clicking the button on the node, or setting a time interval between injects.
2. You have to install this node package externally (node-red-contrib-cron-plus)from manage palette feature(click 3 lines symbol in the top right corner in the node-red workspace)
3. Select the newly added Inject node to see information about its properties and a description of what it does in the Information sidebar pane.

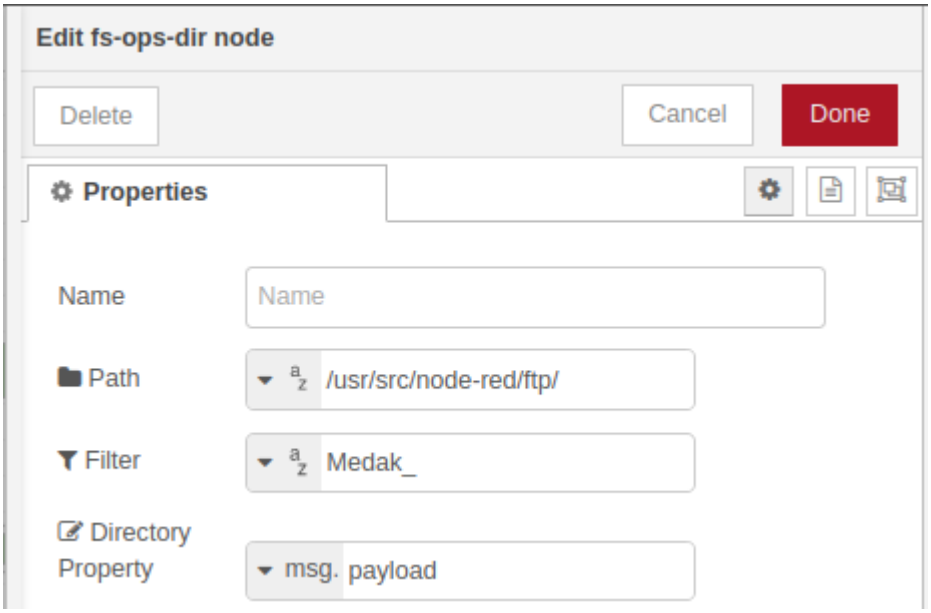


Here we are setting up the triggering time to every 1min it will trigger the corresponding flow in the pre-set time everyday. Connecting this cron plus node to bothnodes (fs-ops-dir and ftp in) so that both will trigger at same time

Fs-ops-dir Node:



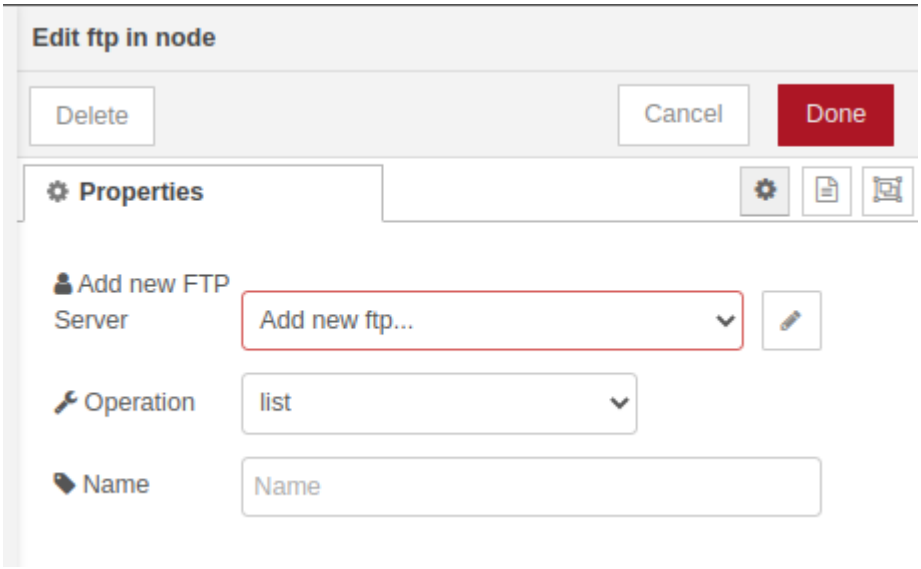
Here we are going to compare 2 directories one directory is in local server where the file are saved from flow 1 next directory is in ftp push server so first we have to list out files in local server directory for that we are using fs-ops-dir node we can install this using external node package (node-red-contrib-fs-ops)



inside this node we have to give path of the directory from where we have to list aout the files if there is any filter we can give that also for better listing in this our file format is like Medak_20_12_2011.csv so that I am filtering it with “Medak_” then set directory property =msg.payload

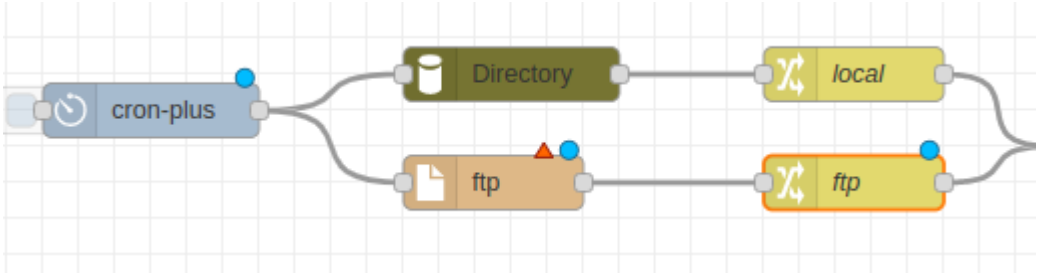
FTP in node:

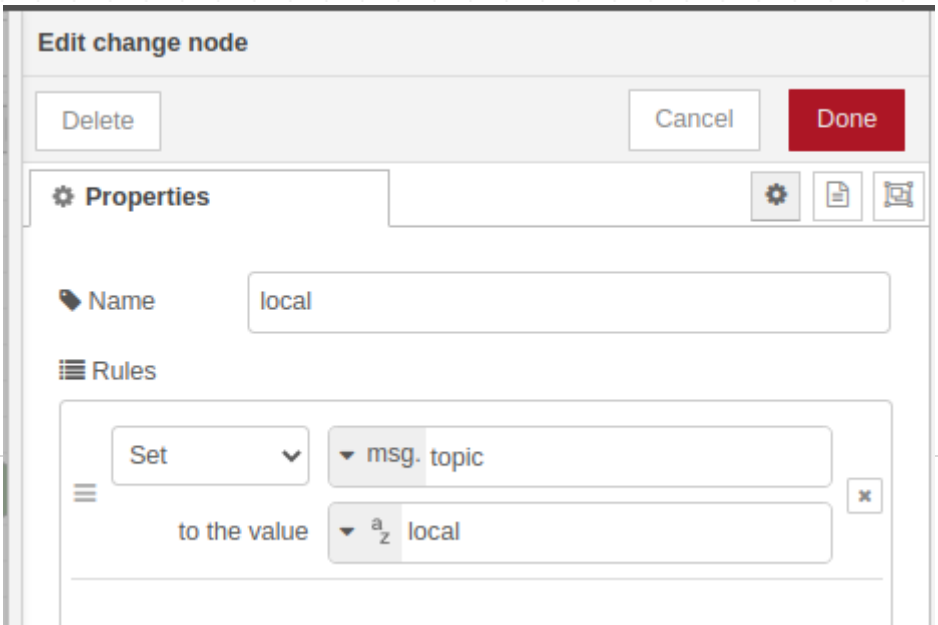
Parellel to fs-ops-dir node we are setting ftp in node also for listing files in the ftp push server for that we have to give ftp server configurations and set operation to list



Change Node:

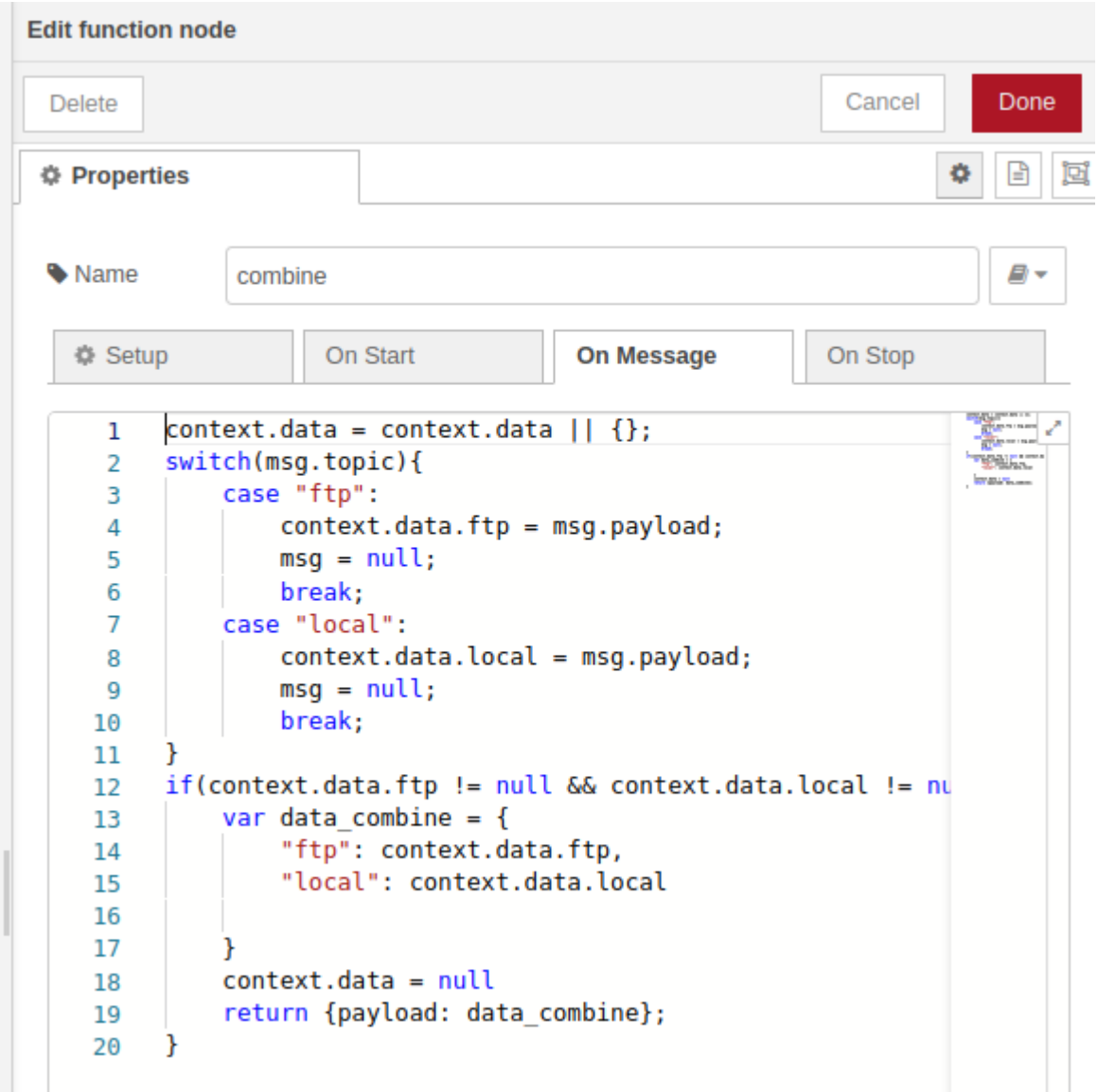
we are connecting both fs-ops-dir and ftp in node to change node to set topic for both outputs





There you have to set msg.topic to any name according to your flow [use the same in combine logic]

Function node for combine outputs from both flows:



Inside function we are writing logic to combine outputs from both flows and merge it to one array of output

combine output will be look like this

Function node to compare both arrays:

- In this logic we are first defining length of both array
- Defining 2 empty array to push files names from the array of outputs and one more empty array to save filtered file names
- Normally if there is any unsent files are there,The array length won't be same(local files number will be more than files in ftp server) in that case only we have to sent the unsent file to ftp server so that we are setting a if condition to check both array length if that is not equal to only it will go inside to take out the unsent file
- If the array lengths are not equal it will go inside a for loop to read array one by one and push the file names to new array we created.
- Then it compare filenames in both array by using if condition if it is equal it go inside a condition object and set flag =1
- so that flag=0 means filename which is not equal or not included so we can push this filtered filename to the 3rd array we create and take out this array only to payload.

Edit function node

Delete | Cancel | Done

Properties

Name: array comparison

Setup | On Start | On Message | On Stop

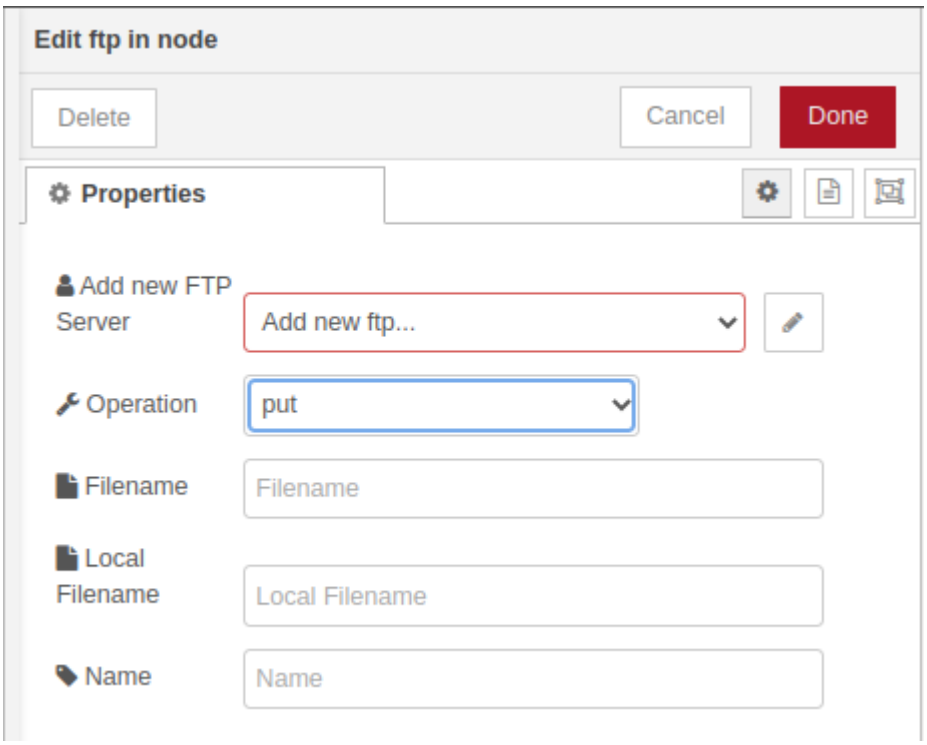
```
1  var local_LENGTH = msg.payload.local.length
2
3  var ftp_LENGTH = msg.payload.ftp.length
4
5  let local = [];
6  let ftp = [];
7
8  let BALANCE = [];
9  if (local_LENGTH != ftp_LENGTH) {
10     for (var i = 0; i < local_LENGTH; i++) {
11         local.push(msg.payload.local[i]);
12
13     }
14     for (var j = 0; j < ftp_LENGTH; j++) {
15
16         ftp.push(msg.payload.ftp[j].name);
17
18     }
19     for (var k = 0; k < local_LENGTH; k++) {
20         var flag = 0;
21         for (var l = 0; l < ftp_LENGTH; l++) {
22             if (local[k] === ftp[l]) {
23
24                 ftp.splice(l, 1);
25                 l--;
26                 flag = 1;
27
28             }
29         }
30
31         if (flag == 0) {
32             var device2 = {};
```

Function node for insert filename and path for ftp in

The output from array comparison function node is again giving to one more function node where we will mention the directory path and filename as a inputs for ftp in node.

```
msg.payload.data = "/usr/src/node-red/ftp/" + msg.payload.files_retrived
//flow.set("path_test",  msg.payload.path);
msg.payload.filename = msg.payload.files_retrived
return msg;

```

ftp in node:



Set Operation –put

Filename - leave this box blank so it will take filename from the function node connected behind it.

Local Filename-leave this box blank so it will take filename from the function node connected behind it.

Setup the server configuration as per below figure (click the pencil icon)enter the username and password to authorize the server.

Set port always 21 for FTP

Debug node:

The Debug node causes any message to be displayed in the Debug sidebar. By default, it just displays the payload of the message, but it is possible to display the entire message object.

3 Click the Deploy button. With the Debug sidebar tab selected,
4 Click the Inject button to get output if there is no triggering time set.