

**ARRAY METER**

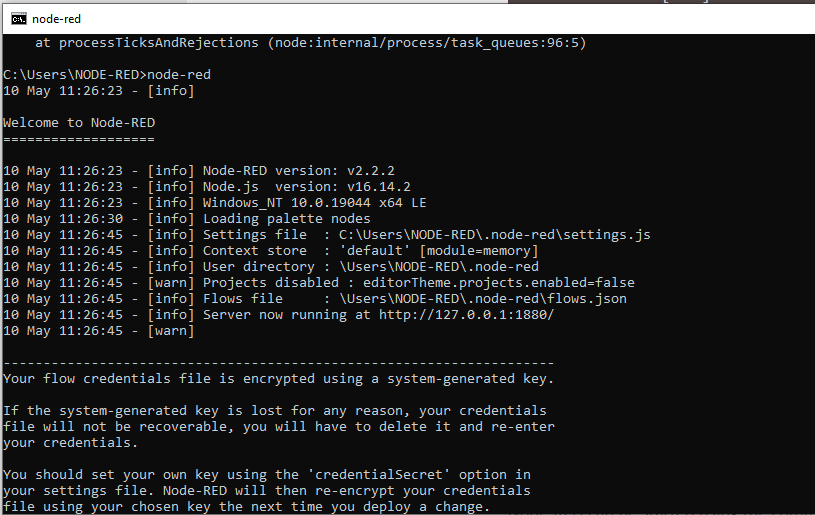
**NODE-RED PROJECT**

**MANUAL**

**MALAVIKA K.V**

**NODE-RED INSTALLATION SETUP**

1. First need to install node-red using [Running Node-RED locally : Node-RED (nodered.org)](https://nodered.org/docs/getting-started/local)
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.



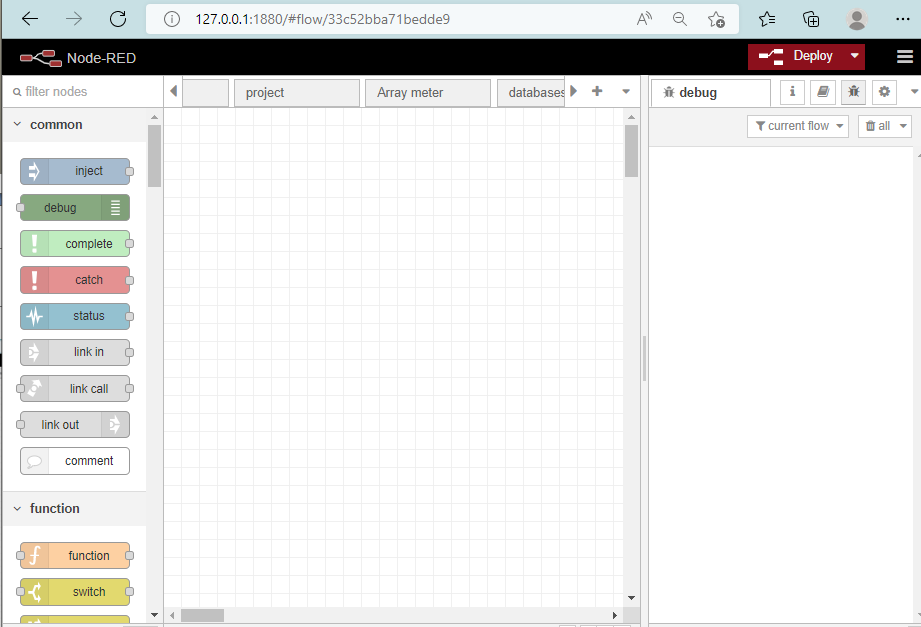
1. Access the editor

With Node-RED [running](https://nodered.org/docs/getting-started), 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](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.

1. 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.



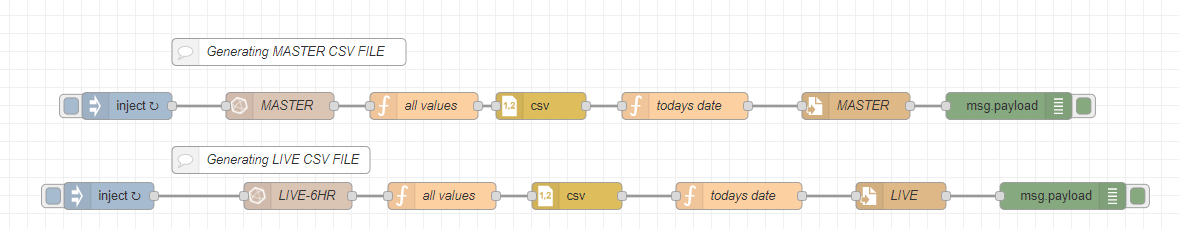
**ARRAY METER PROJECT REQUIREMETS**

STEP 1:Collecting 24 hr data of meters which are in live from last 6hr(6PM-12AM)on a day and create a csv file with 4 columns that  are meter id, timestamp, kwhD\_lifetime, kwhR\_lifetime  
  
STEP 2:Historical data filling of live meters from their last communicating time to current date timestamp  
  
STEP 3:Create a single csv file that having live meter ids and kwhD\_lifetime and kwhR\_lifetime  data from their last communicated time to current date with historical data filled if any meter come in to live after few days of no communication.

STEP 4: SFTP Push to sent this csv file to arraymeter server everyday

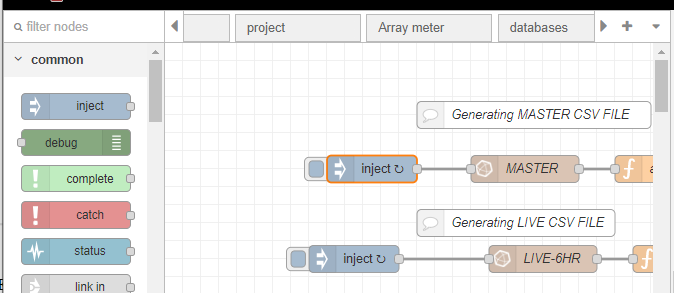
**STEP 1**:Collecting 24 hr data of meters which are in live from last 6hr(6PM-12AM)on a day and create a csv file with 4 columns that  are meter id, timestamp, kwhD\_lifetime, kwhR\_lifetime

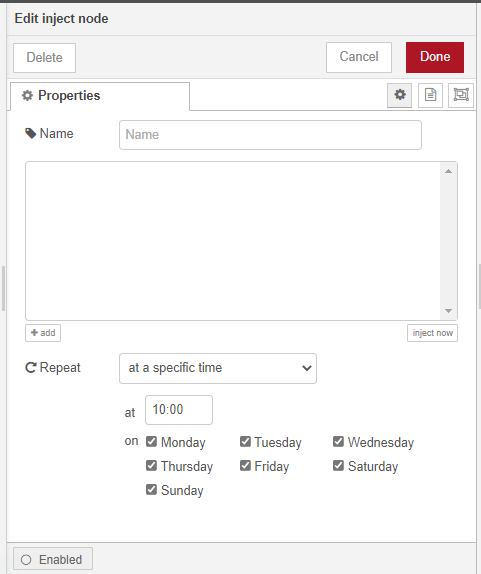
**NODE-RED FLOW FOR STEP 1:**



**Inject Node:**

1. The Inject 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. Drag one onto the [workspace](https://nodered.org/docs/user-guide/editor/workspace/) from the [palette](https://nodered.org/docs/user-guide/editor/palette/).
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](https://nodered.org/docs/user-guide/editor/sidebar/info).



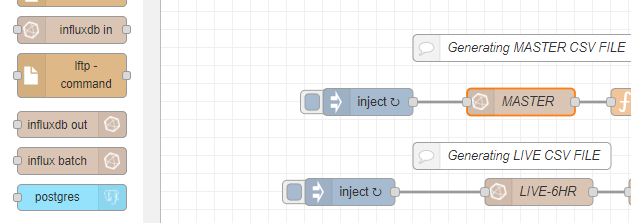


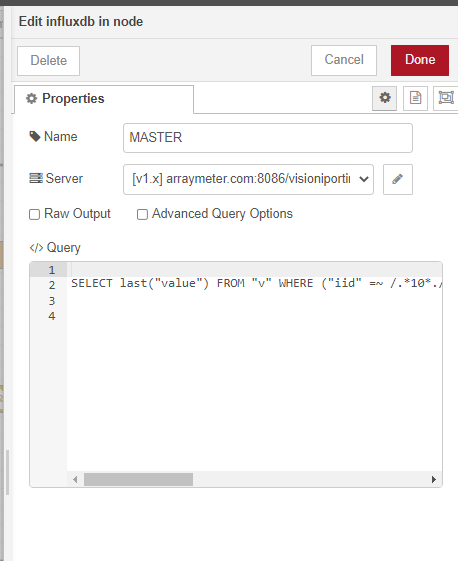
Here we are setting up the triggering time to 10:00 am repeat that will trigger the corresponding flow in thet pre-set time everyday or in selected days.

**Influxdb in node** ([node-red-contrib-influxdb (node) - Node-RED (nodered.org)](https://flows.nodered.org/node/node-red-contrib-influxdb)**):**

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



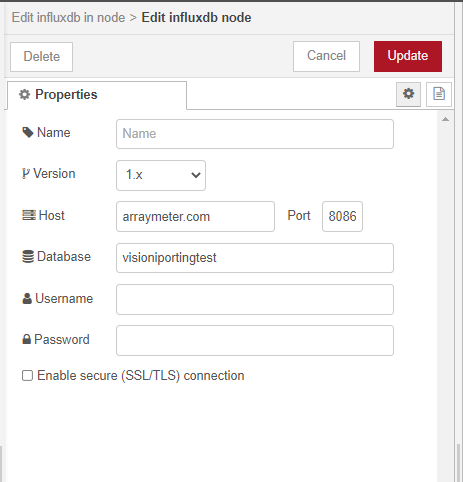
Write the influx query in the query box given

Master Query = SELECT last("value") FROM "v" WHERE ("iid" =~ /.\*10\*./ AND "f" =~ /^kWhD\_lifetime$/) AND time >= now() - 30d GROUP BY time(15m), "p", "b", "d", "f", "iid" fill(none);

Live query =

SELECT last("value") FROM "v" WHERE ("iid" =~ /.\*10\*./ AND "f" =~ /^kWhD\_lifetime$/) AND time >= now() - 6h GROUP BY time(15m), "p", "b", "d", "f", "iid" fill(null);

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



Version =1.x

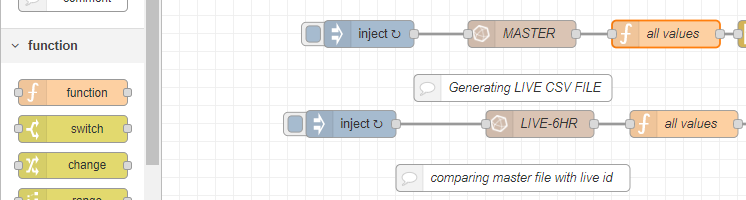
Host = arraymeter.com

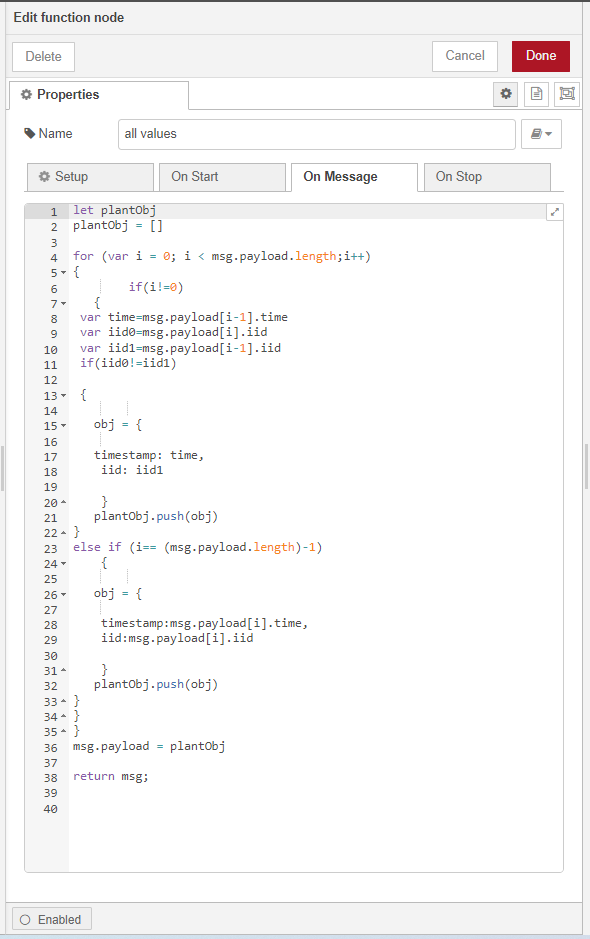
Port = 8086

Database = visioniportingtest(use the same database name used in influx db)

**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 plantObj

plantObj = []

for (var i = 0; i < msg.payload.length;i++)

{

if(i!=0)

{

var time=msg.payload[i-1].time

var iid0=msg.payload[i].iid

var iid1=msg.payload[i-1].iid

if(iid0!=iid1)

{

obj = {

timestamp: time,

iid: iid1

}

plantObj.push(obj)

}

else if (i== (msg.payload.length)-1)

{

obj = {

timestamp:msg.payload[i].time,

iid:msg.payload[i].iid

}

plantObj.push(obj)

}

}

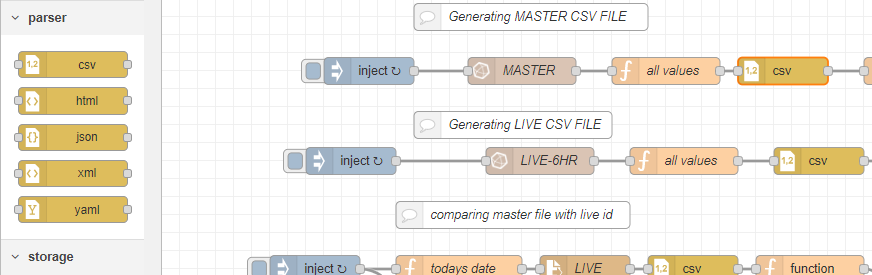
}

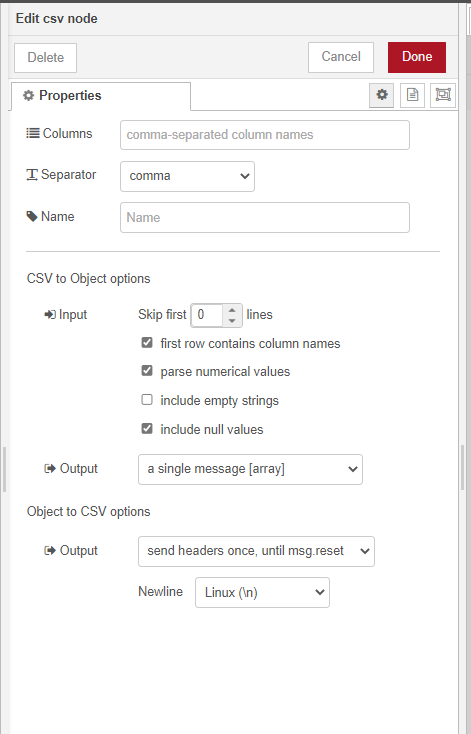
msg.payload = plantObj

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 only the last communicated time of every meter

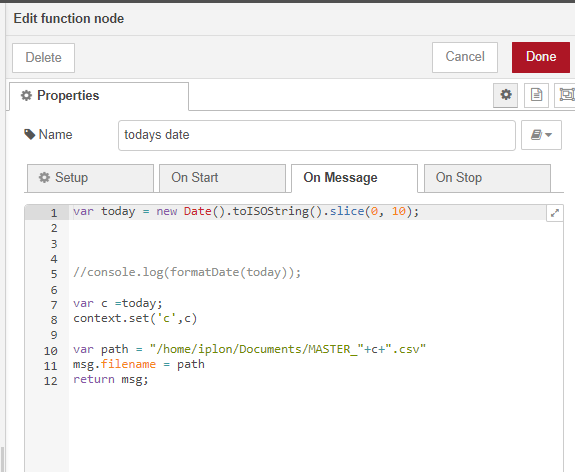
**csv node:**





Tick the boxes like this and take the output as a single message[array]

**Function node: To give todays date as file name**



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

**Code:**

var today = new Date().toISOString().slice(0, 10);

var c =today;

Change the file path according to the device

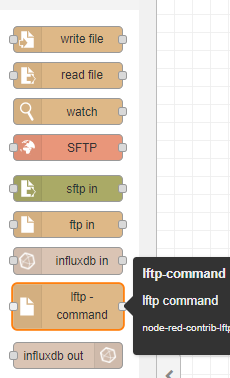
context.set('c',c)

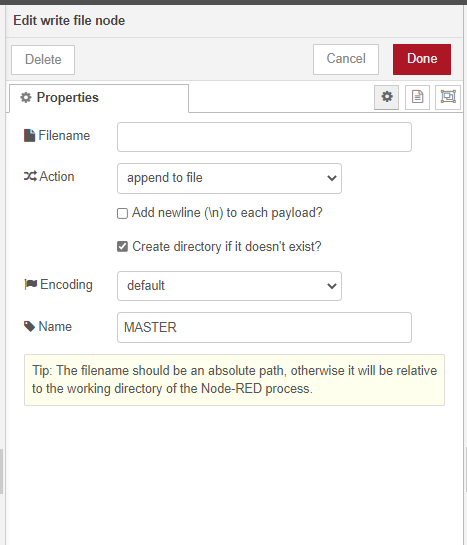
var path = "/home/iplon/Documents/MASTER\_"+c+".csv"

msg.filename = path

return msg;

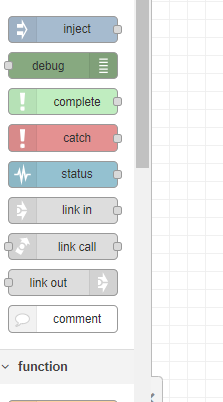
**Write file node:**

****



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

**Debug node:**



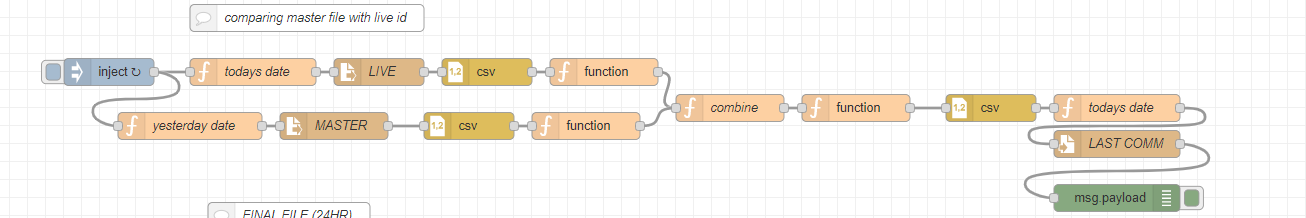
The Debug node causes any message to be displayed in the [Debug sidebar](https://nodered.org/docs/user-guide/editor/sidebar/debug). By default, it just displays the payload of the message, but it is possible to display the entire message object.

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

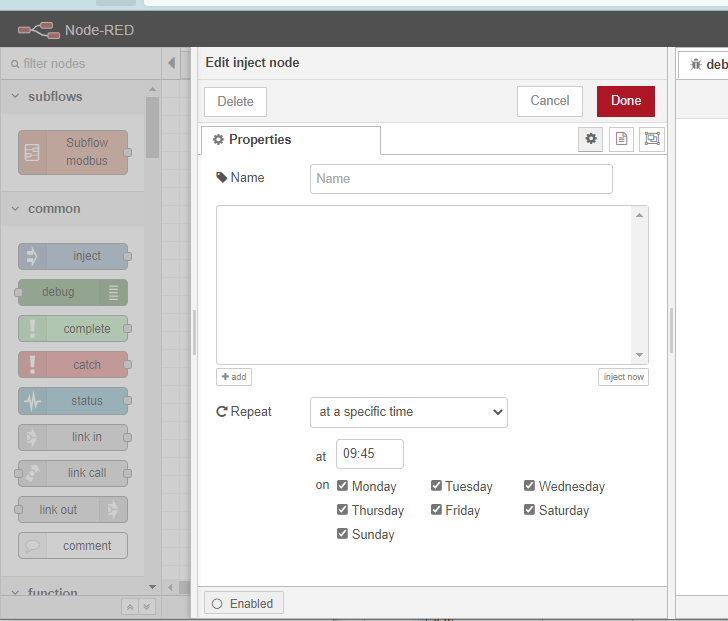
**STEP 2:Historical data filling of live meters from their last communicating time to current date timestamp**

**NODE-RED FLOW FOR STEP 2:**

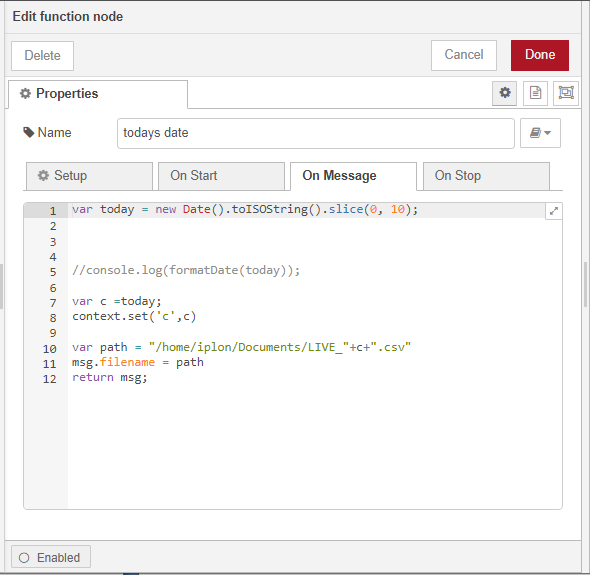
Here we are comparing live csv that collected today with master csv collected yesterday it will validate the both file and take last communicated time of live meter from master csv



INJECT NODE SETUP



1. Inject node Then connected to 2 function node first one (todays date)for giving file path as todays date



Code:

var today = new Date().toISOString().slice(0, 10);

Change the file path according to the device

var c =today;

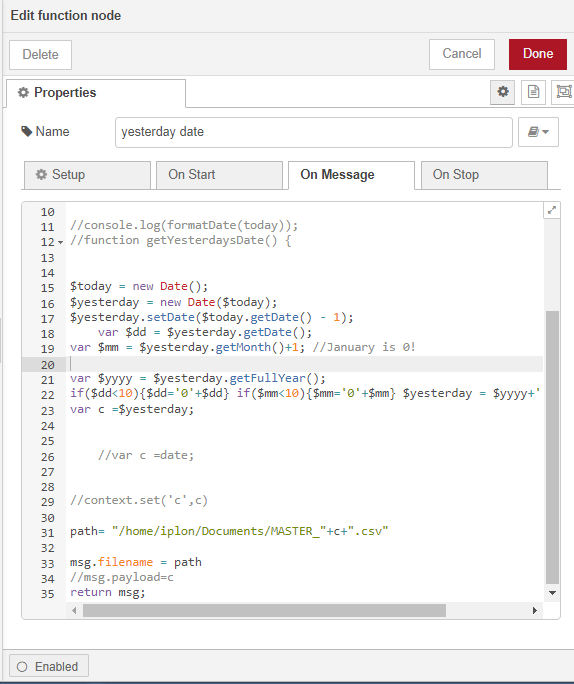
context.set('c',c)

var path = "/home/iplon/Documents/LIVE\_"+c+".csv"

msg.filename = path

return msg;

1. Second function node(yesterdays date)for giving file path as yesterdays date.



Code :

$today = new Date();

$yesterday = new Date($today);

$yesterday.setDate($today.getDate() - 1);

var $dd = $yesterday.getDate();

var $mm = $yesterday.getMonth()+1; //January is 0!

var $yyyy = $yesterday.getFullYear();

if($dd<10){$dd='0'+$dd} if($mm<10){$mm='0'+$mm} $yesterday = $yyyy+'-'+$mm+'-'+$dd;

Change the file path according to the device

var c =$yesterday;

path= "/home/iplon/Documents/MASTER\_"+c+".csv"

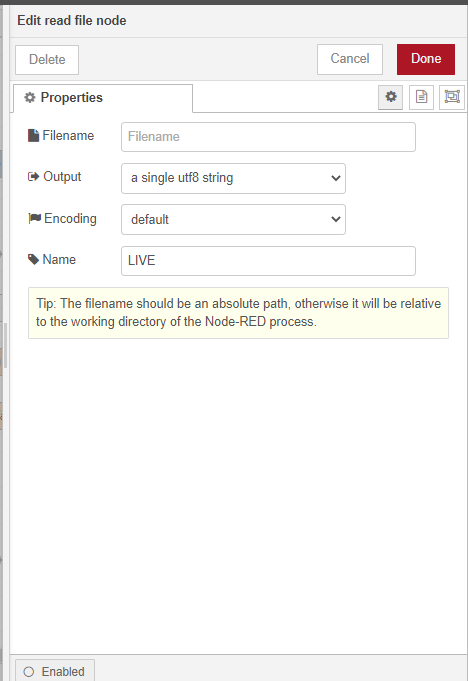
msg.filename = path

//msg.payload=c

return msg;

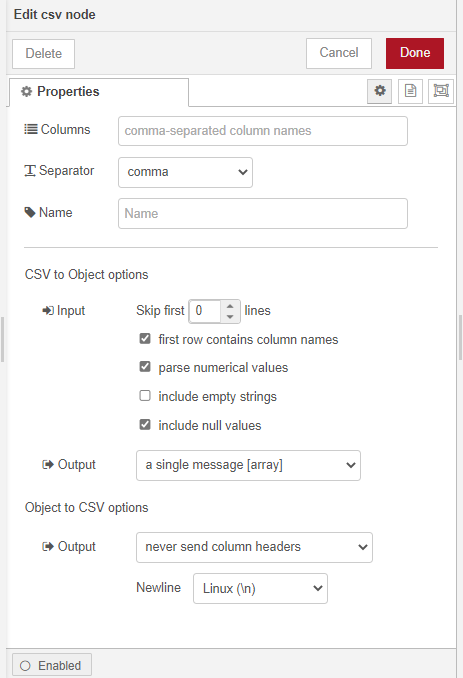
**READ FILE NODE:**

2 read file node are there. In both nodes we are leaving the file name box blank so it will take the path what we are feeding from function node



**csv Node:**

configure both the csv nodes same as in the figure



**Function node:**

We are using 2 functions node here both for taking length of the array and meters id in the corresponding file and also we are giving topic to both flows to merge both flows



Code of first function node(live):

let found1 = []

let device1

var Length = msg.payload.length

for (var i = 0; i<msg.payload.length;i++)

{

device1 = {};

device1.payload = {

iid:msg.payload[i].iid,

Length

}

found1.push( device1 )

}

msg.payload = found1;

msg.topic = "live";

return msg;



Code of second function node(master):

let found1 = []

let device1

var Length1 = msg.payload.length

for (var i = 0; i<msg.payload.length;i++)

{

device1 = {};

device1.payload = {

iid1:msg.payload[i].iid,

last\_comm:msg.payload[i].timestamp,

Length1

}

found1.push( device1 )

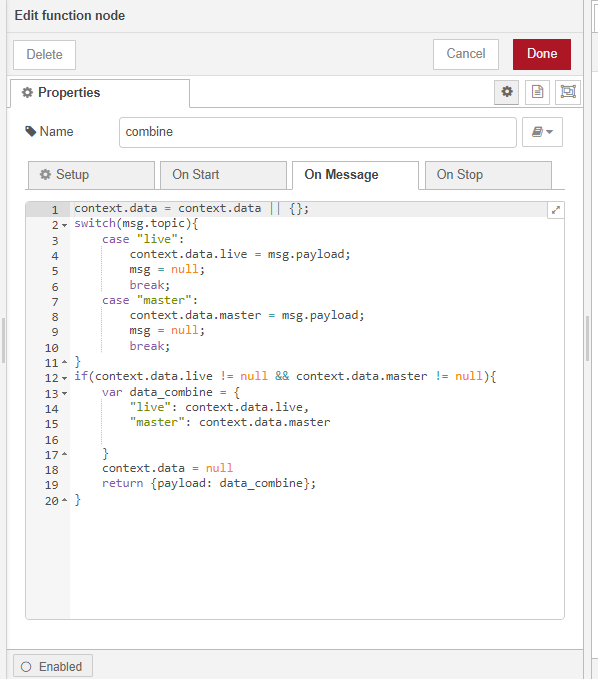
}

msg.payload = found1;

msg.topic = "master";

return msg;

**Function node(for merging both flows):**



Code:

context.data = context.data || {};

switch(msg.topic){

case "live":

context.data.live = msg.payload;

msg = null;

break;

case "master":

context.data.master = msg.payload;

msg = null;

break;

}

if(context.data.live != null && context.data.master != null){

var data\_combine = {

"live": context.data.live,

"master": context.data.master

}

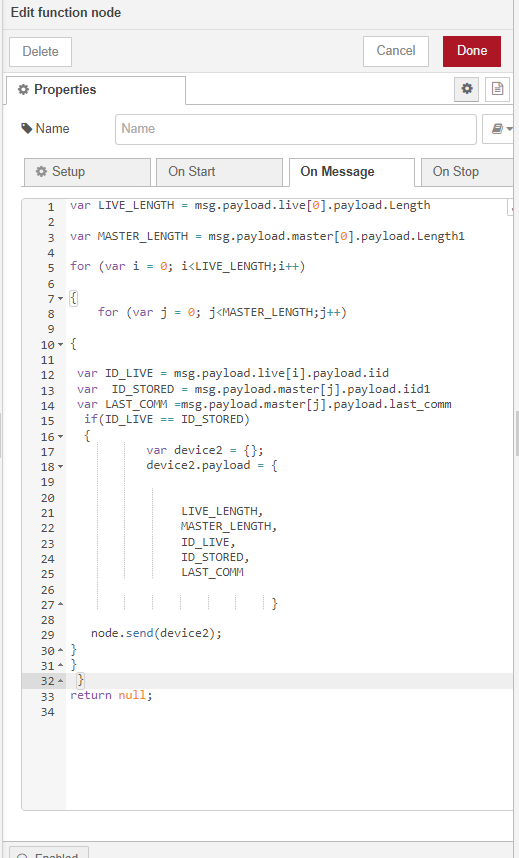
context.data = null

return {payload: data\_combine};

}

**Function node(for comparison)**

For comparing both files and taking last communication timestamp of live meters



Code :

var LIVE\_LENGTH = msg.payload.live[0].payload.Length

var MASTER\_LENGTH = msg.payload.master[0].payload.Length1

for (var i = 0; i<LIVE\_LENGTH;i++)

{

for (var j = 0; j<MASTER\_LENGTH;j++)

{

var ID\_LIVE = msg.payload.live[i].payload.iid

var ID\_STORED = msg.payload.master[j].payload.iid1

var LAST\_COMM =msg.payload.master[j].payload.last\_comm

if(ID\_LIVE == ID\_STORED)

{

var device2 = {};

device2.payload = {

LIVE\_LENGTH,

MASTER\_LENGTH,

ID\_LIVE,

ID\_STORED,

LAST\_COMM

}

node.send(device2);

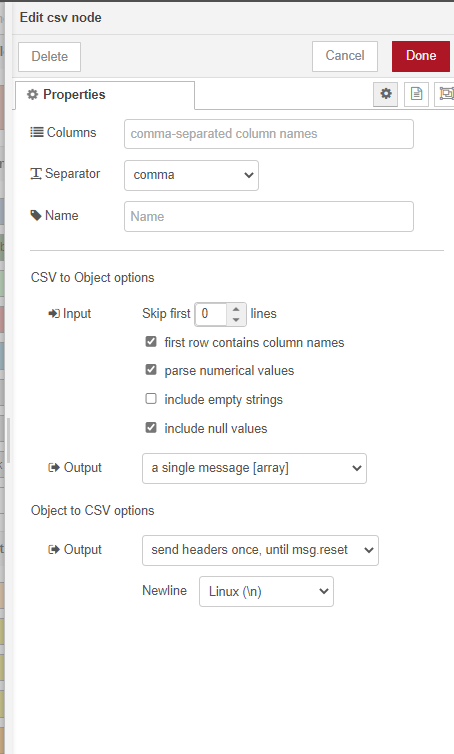
}

}

}

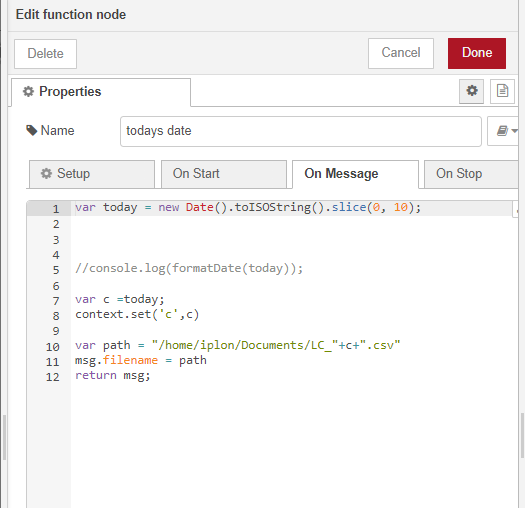
return null;

**csv node:**

****

Configure csv node same as in the figure

**Function node for todays date:**



Code:

var today = new Date().toISOString().slice(0, 10);

var c =today;

Change the file path according to the device

context.set('c',c)

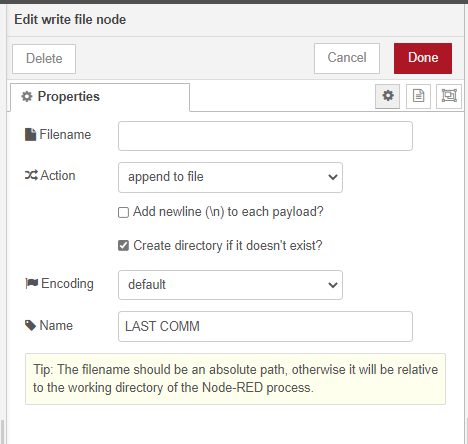
var path = "/home/iplon/Documents/LC\_"+c+".csv"

msg.filename = path

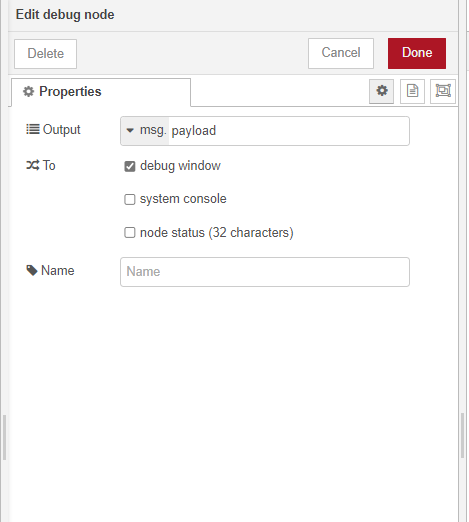
return msg;

**Write File Node:**

Leave the file name box blank so it will take the path what we are feeding from function node



**Debug node:**

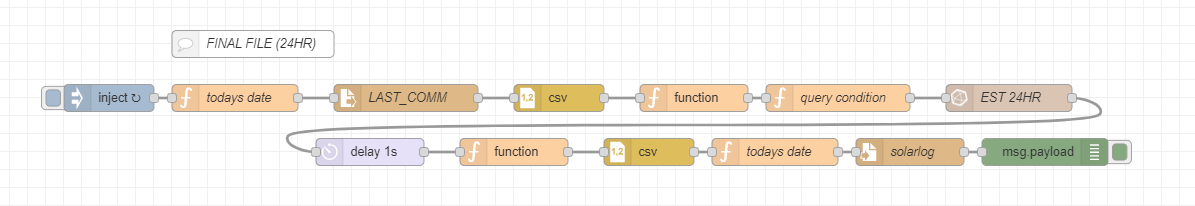


The Debug node causes any message to be displayed in the [Debug sidebar](https://nodered.org/docs/user-guide/editor/sidebar/debug). By default, it just displays the payload of the message, but it is possible to display the entire message object.

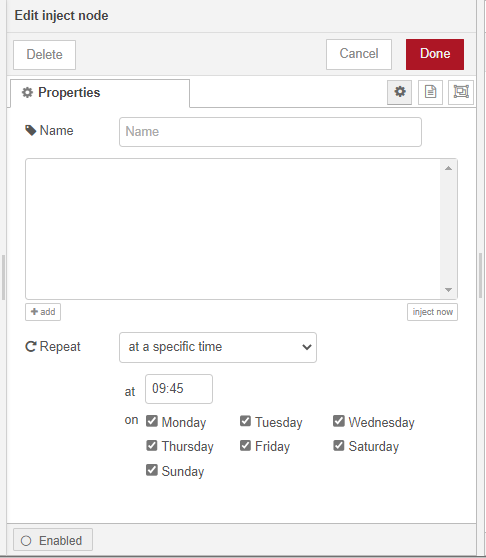
1. Click the Deploy button.With the Debug sidebar tab selected,
2. Click the Inject button to get output if there is no triggerng time set.
3. Check the destination of file what we give as path to check whether the csv file created or not.

**STEP 3: Create a single csv file that having live meter ids and kwhD\_lifetime and kwhR\_lifetime  data from their last communicated time to current date with historical data filled if any meter come in to live after few days of no communication.**

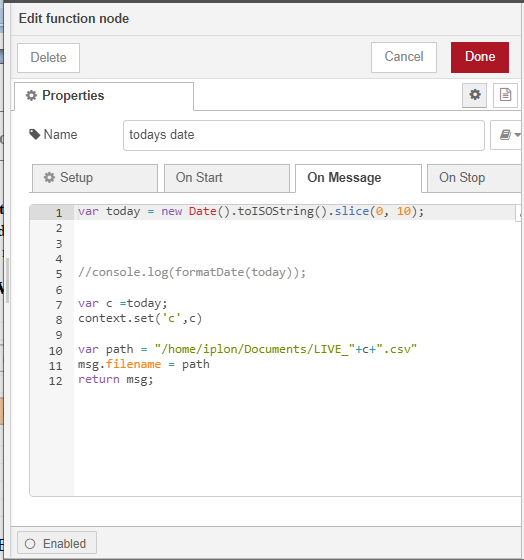
**NODE-RED FLOW FOR STEP 3**

****

INJECT NODE SETUP

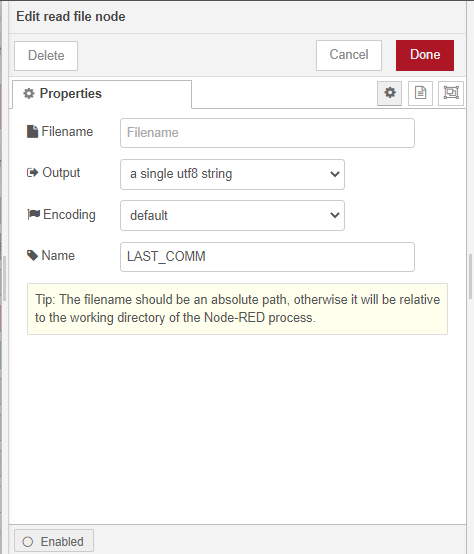
****

1. Inject node Then connected to a function node (todays date)for giving file path as today’s date



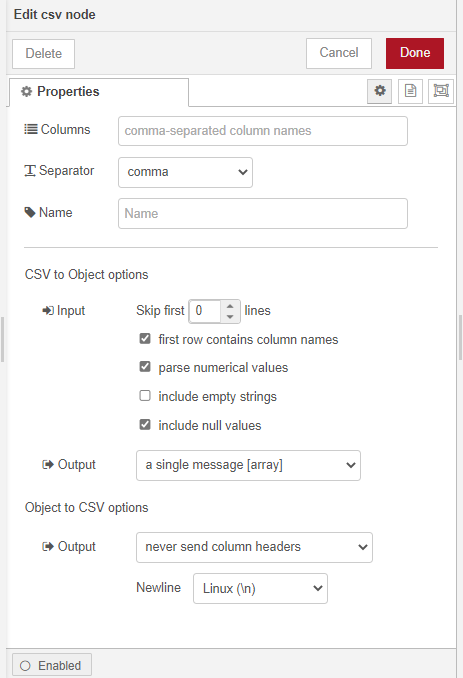
**READ FILE NODE:**

we are leaving the file name box blank so it will take the path what we are feeding from function node

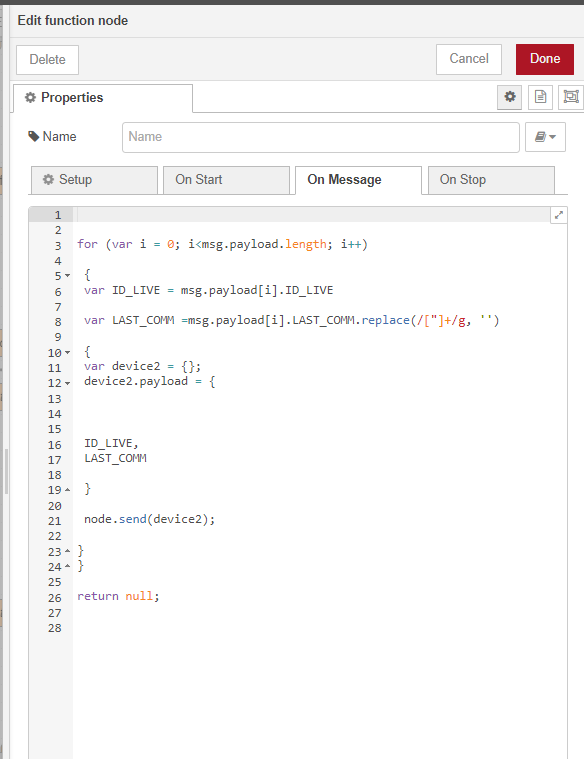


**CSV NODE:**

Configure csv node same as in below figure

****

This csv node output connected to a function node which have javascript code for send an array of output to single messages and also making the last communicate format



Code:

for (var i = 0; i<msg.payload.length; i++)

{

var ID\_LIVE = msg.payload[i].ID\_LIVE

var LAST\_COMM =msg.payload[i].LAST\_COMM.replace(/["]+/g, '')

{

var device2 = {};

device2.payload = {

ID\_LIVE,

LAST\_COMM

}

node.send(device2);

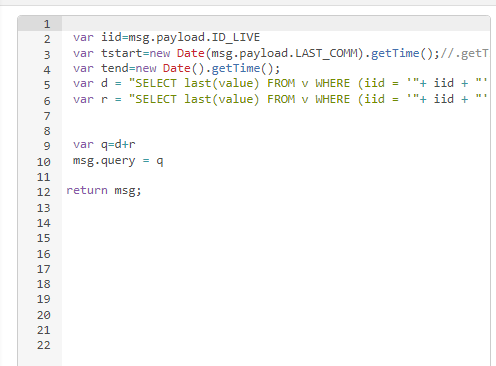
}

}

return null;

**FUNCTION NODE FOR QUERY CONDITIONS**

Function node for query with varying live id and start and end time and query every live id data from their corresponding time to current time.

****

**Code:**

var iid=msg.payload.ID\_LIVE

var tstart=new Date(msg.payload.LAST\_COMM).getTime();//.getTime();

var tend=new Date().getTime();

var d = "SELECT last(value) FROM v WHERE (iid = '"+ iid + "' AND f ='kWhD\_lifetime') AND time >= " + tstart + "ms and time <= " + tend + "ms GROUP BY time(15m),f, iid fill(previous);";

var r = "SELECT last(value) FROM v WHERE (iid = '"+ iid + "' AND f ='kWhR\_lifetime') AND time >= " + tstart + "ms and time <= " + tend + "ms GROUP BY time(15m),f, iid fill(previous);";

var q=d+r

msg.query = q

return msg;

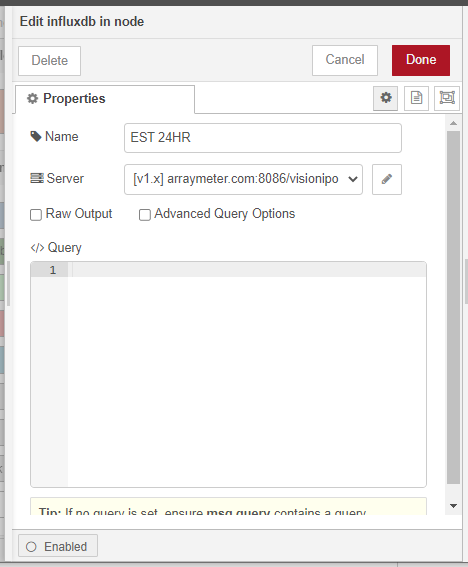
**INFLUXDB IN NODE:**

* Connect function node where query condition code running to this influx db leave the query box inside the influxdb in node blank hen only it will take the function node outputs as query
* Configure the server details

Host: arraymeter.com

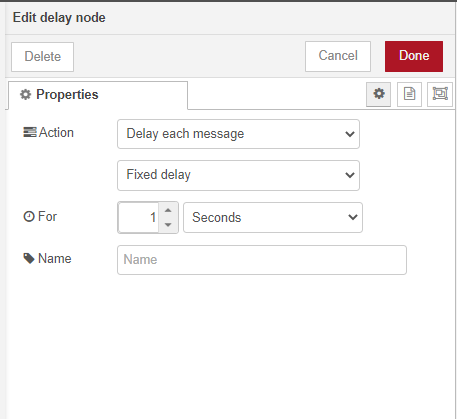
Port:8086

Database: visioniportingtest

****

**DELAY NODE:**

Delay node used here for avoid rush of output messages here giving fixed delay of 1s foe each output messages.

****

**FUNCTION NODE TO CREATE FINAL FILE:**

Here a javascript code used to convert utc time to US timing and an if condition to filter out reversed meter and the values to kwhd\_limetime column

Output set as 3 column cith column names timestamp,kwh\_lifetime,iid

****

**Code:**

let plantObj

plantObj = []

for (var i = 0; i <msg.payload[0].length ;i++)

{

var utcTime = new Date(msg.payload[0][i].time);

console.log('UTC Time: ' + utcTime.toISOString());

var usaTime = utcTime.toLocaleString("en-US", {timeZone: "America/New\_York"});

console.log('USA time: '+ usaTime)

var kwhd\_lifetime= msg.payload[0][i].last;

var kwhr\_lifetime= msg.payload[1][i].last;

var iid=msg.payload[0][i].iid

{

if ((iid==10677693)||(iid==10677696)||(iid==10677750)||(iid==10677719)||(iid==10677695)||(iid==10677720)||(iid==10677694)||(iid==10677689)||(iid==10586404))

{

kwhd\_lifetime=kwhr\_lifetime}

var obj1 = {

timestamp:usaTime,

kwh\_lifetime:kwhd\_lifetime,

iid

}

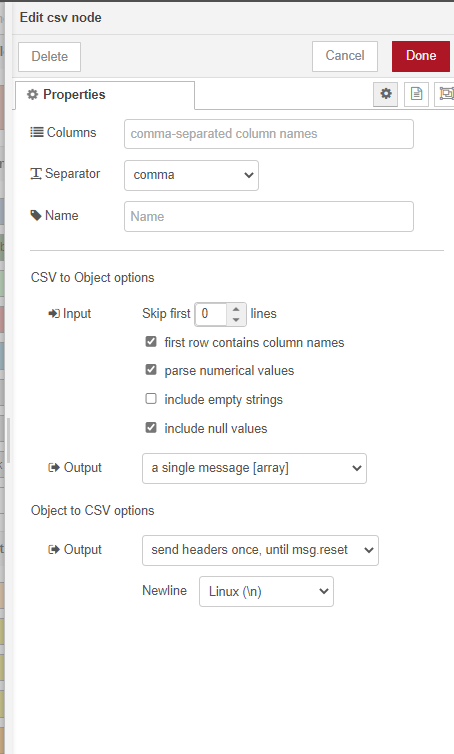
plantObj.push(obj1)

}}

msg.payload = plantObj

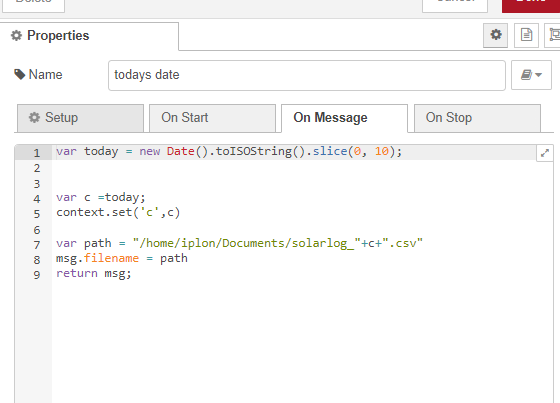
return msg;

**csv node:**

****

Configure csv node same as in the figure

**Function node for today’s date:**



Code:

var today = new Date().toISOString().slice(0, 10);

var c =today;

Change the file path according to the device

context.set('c',c)

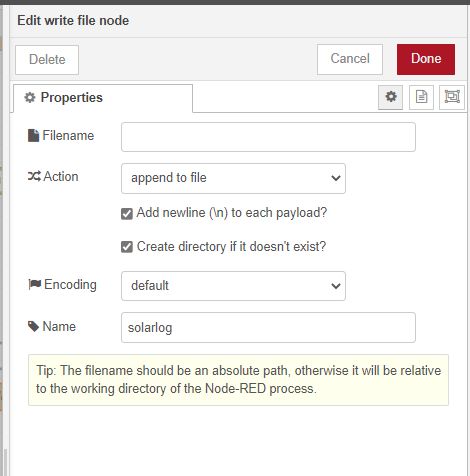
var path = "/home/iplon/Documents/solarlog\_"+c+".csv"

msg.filename = path

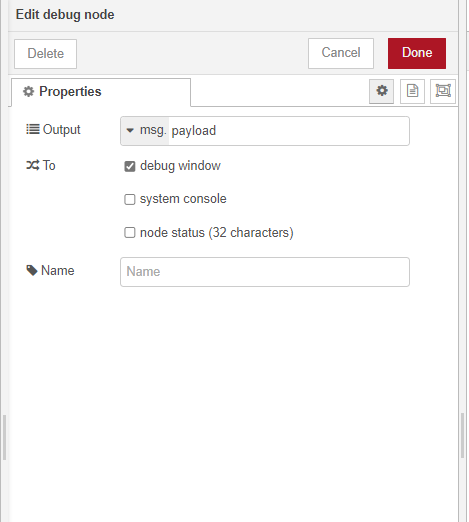
return msg;

**Write File Node:**

Leave the file name box blank so it will take the path what we are feeding from function node



**Debug node:**

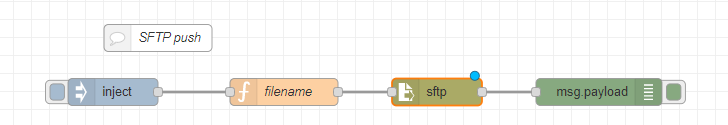


The Debug node causes any message to be displayed in the [Debug sidebar](https://nodered.org/docs/user-guide/editor/sidebar/debug). By default, it just displays the payload of the message, but it is possible to display the entire message object.

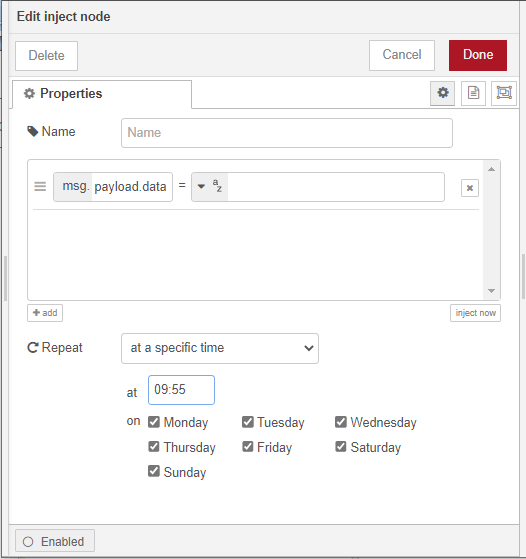
1. Click the Deploy button.With the Debug sidebar tab selected,
2. Click the Inject button to get output if there is no triggerng time set.
3. Check the destination of file what we give as path to check whether the csv file created or not.

**STEP 4: SFTP Push to sent this csv file to arraymeter server everyday**

**NODE-RED flow for STEP 4:**

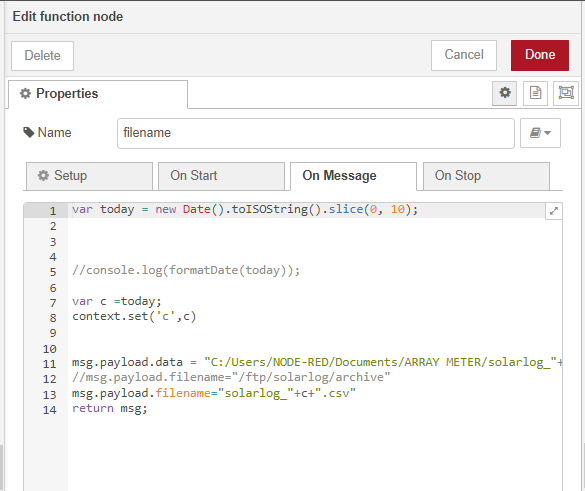
****

**INJECT NODE SETUP**

****

**FUNCTION NODE (todays date):**

Function node with javascript to get final file and give this path to SFTP push node



Code:

var today = new Date().toISOString().slice(0, 10);

Change the file path according to the device

var c =today;

context.set('c',c)

msg.payload.data="C:/Users/NODE-RED/Documents/ARRAY METER/solarlog\_"+c+".csv"

msg.payload.filename="solarlog\_"+c+".csv"

return msg;

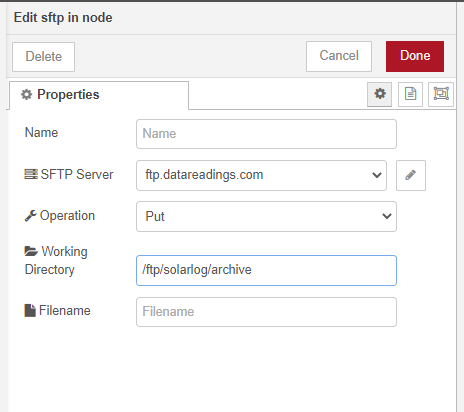
**SFTP NODE:**

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

Set Operation –put

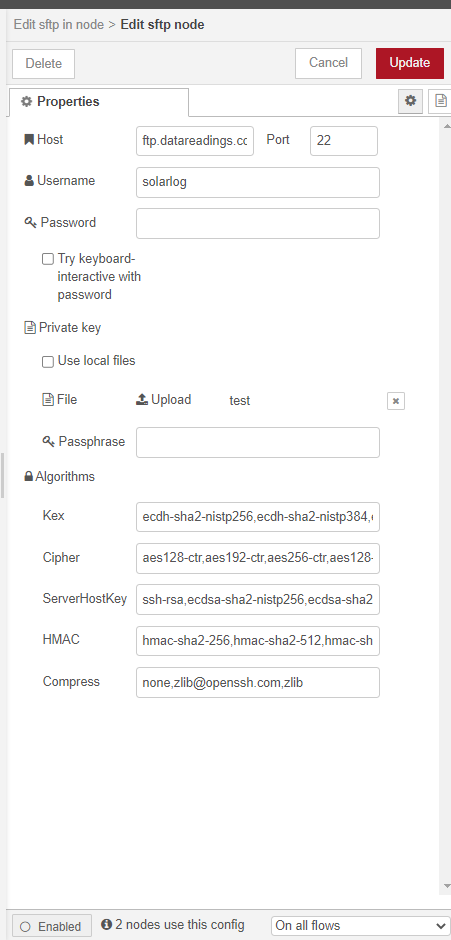
Working directory- /ftp/solarlog/archive

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

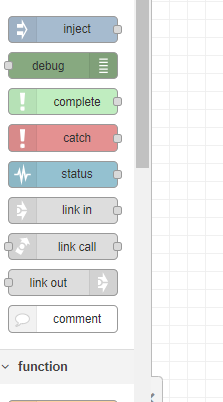


Setup the server configuration as per below figure (click the pencil icon)upload the private key file given for sftp connection in the configuration box for connecting node-red to the server.

Set port always 22 for SFTP , 21 for FTP



**Debug node:**



The Debug node causes any message to be displayed in the [Debug sidebar](https://nodered.org/docs/user-guide/editor/sidebar/debug). 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 triggerng time set.