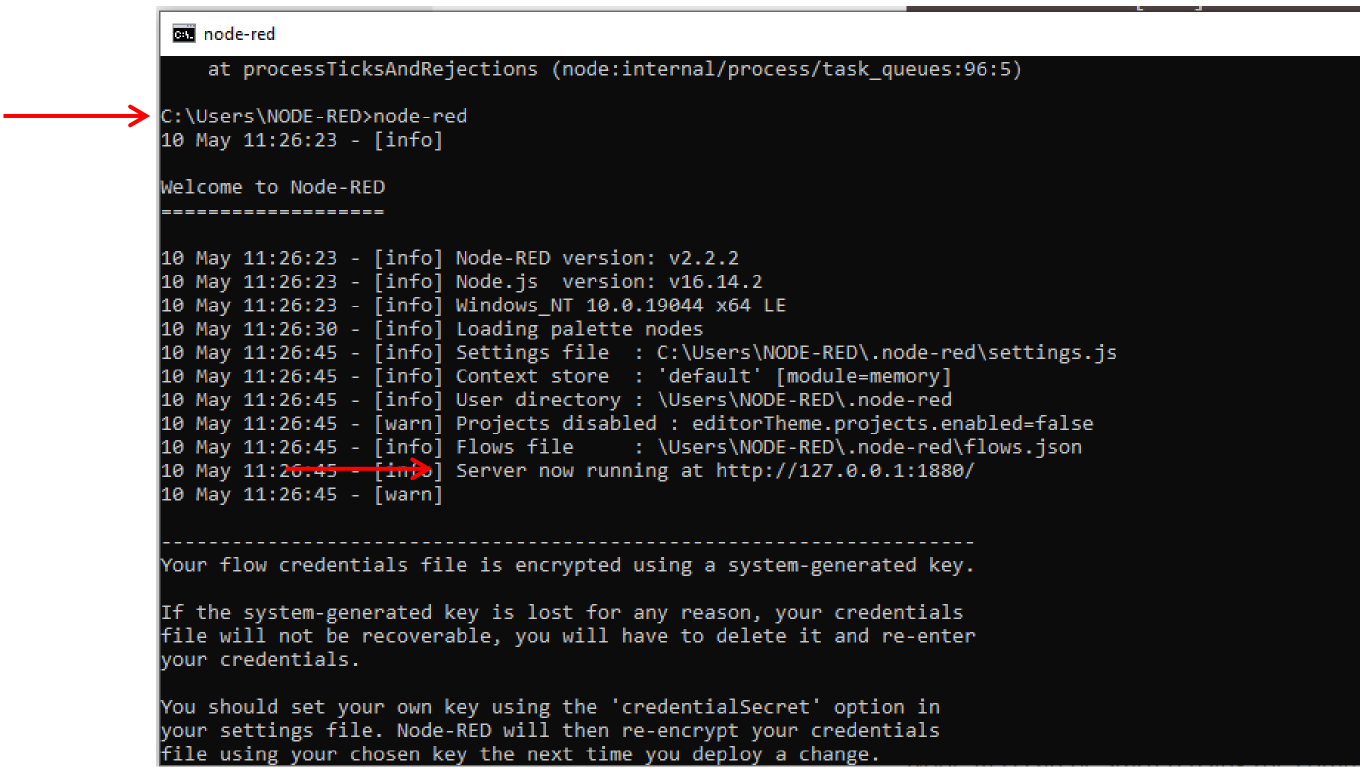
ARRAY METER NODE-RED PROJECT MANUAL

MALAVIKA K.V

NODE-RED INSTALLATION SETUP

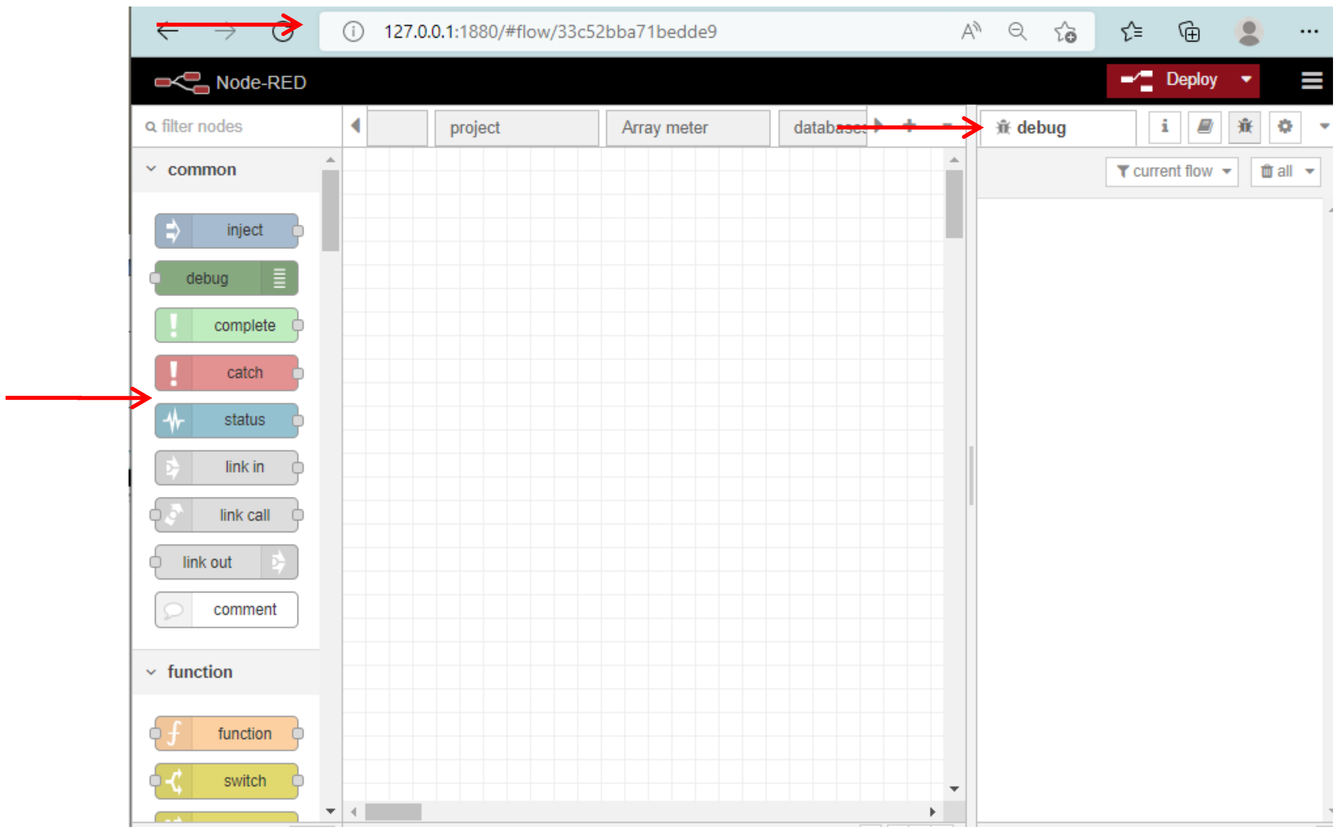
1. First need to install node-red using [Running Node-RED locally : Node-RED (nodered.org)]()
2. Once installed as a global module you can use the node-red command to start Node-RED in your terminal. You can use Ctrl-C or close the terminal window to stop Node-RED.



```
node-red
    at processTicksAndRejections (node:internal/process/task_queues:96:5)

C:\Users\NODE-RED>node-red
10 May 11:26:23 - [info]

Welcome to Node-RED
===================

10 May 11:26:23 - [info] Node-RED version: v2.2.2
10 May 11:26:23 - [info] Node.js  version: v16.14.2
10 May 11:26:23 - [info] Windows_NT 10.0.19044 x64 LE
10 May 11:26:30 - [info] Loading palette nodes
10 May 11:26:45 - [info] Settings file  : C:\Users\NODE-RED\.node-red\settings.js
10 May 11:26:45 - [info] Context store  : 'default' [module=memory]
10 May 11:26:45 - [info] User directory : \Users\NODE-RED\.node-red
10 May 11:26:45 - [warn] Projects disabled : editorTheme.projects.enabled=false
10 May 11:26:45 - [info] Flows file     : \Users\NODE-RED\.node-red\flows.json
10 May 11:26:45 - [info] Server now running at http://127.0.0.1:1880/
10 May 11:26:45 - [warn]

---------------------------------------------------------------------
Your flow credentials file is encrypted using a system-generated key.

If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.

You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.
```

3. 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]().

4. 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

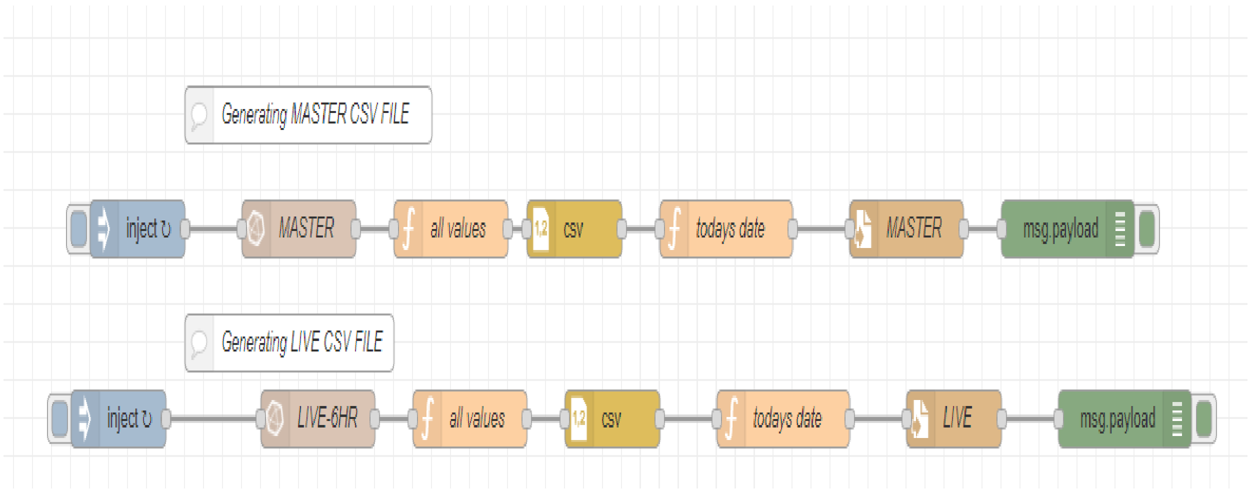
STEP 2:Historical data filling of live meters from their last communicating time to current date timestamp

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

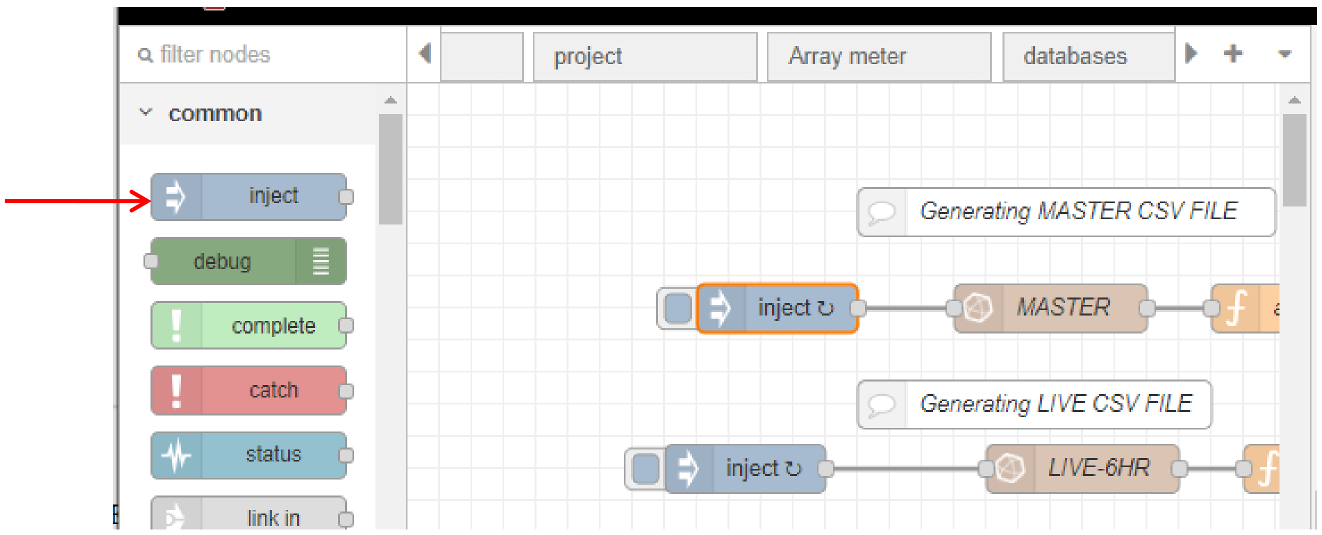
STEP 1:Collecting 24 hr data of meters which are in live from last 6hr(6PM-12AM)on a day and create a csv file with 4 columns that are meter id, timestamp, kwhD_lifetime, kwhR_lifetime

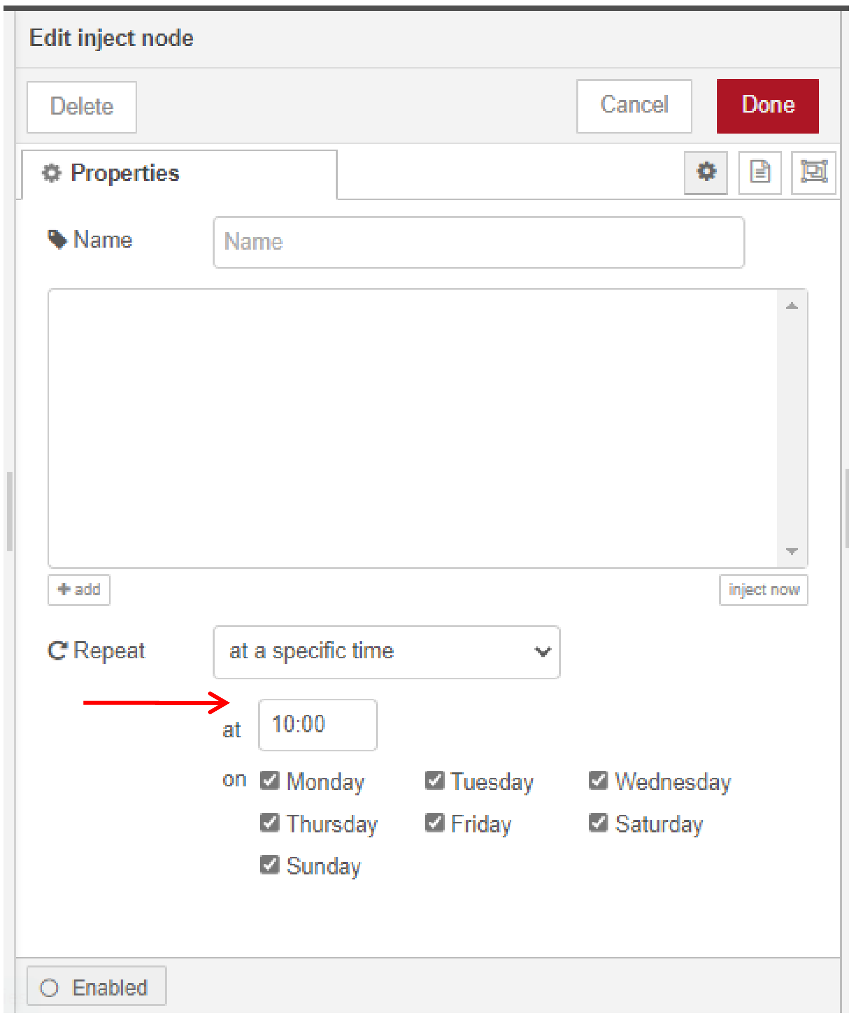
NODE-RED FLOW FOR STEP 1:



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 from the 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.




Edit inject node

Name: Name

Repeat: at a specific time

at 10:00

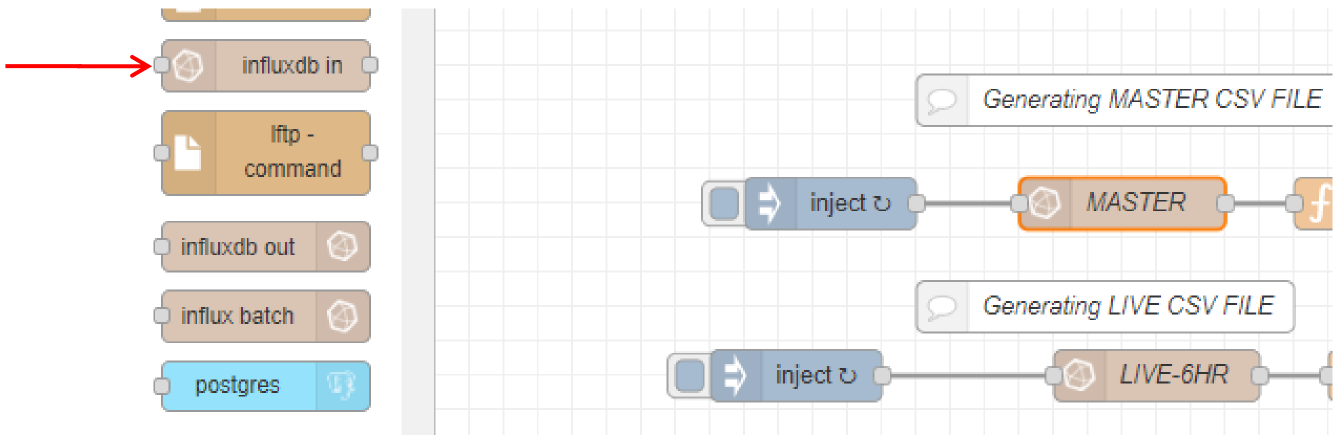
on ☑ Monday ☑ Tuesday ☑ Wednesday ☑ Thursday ☑ Friday ☑ Saturday ☑ Sunday

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.

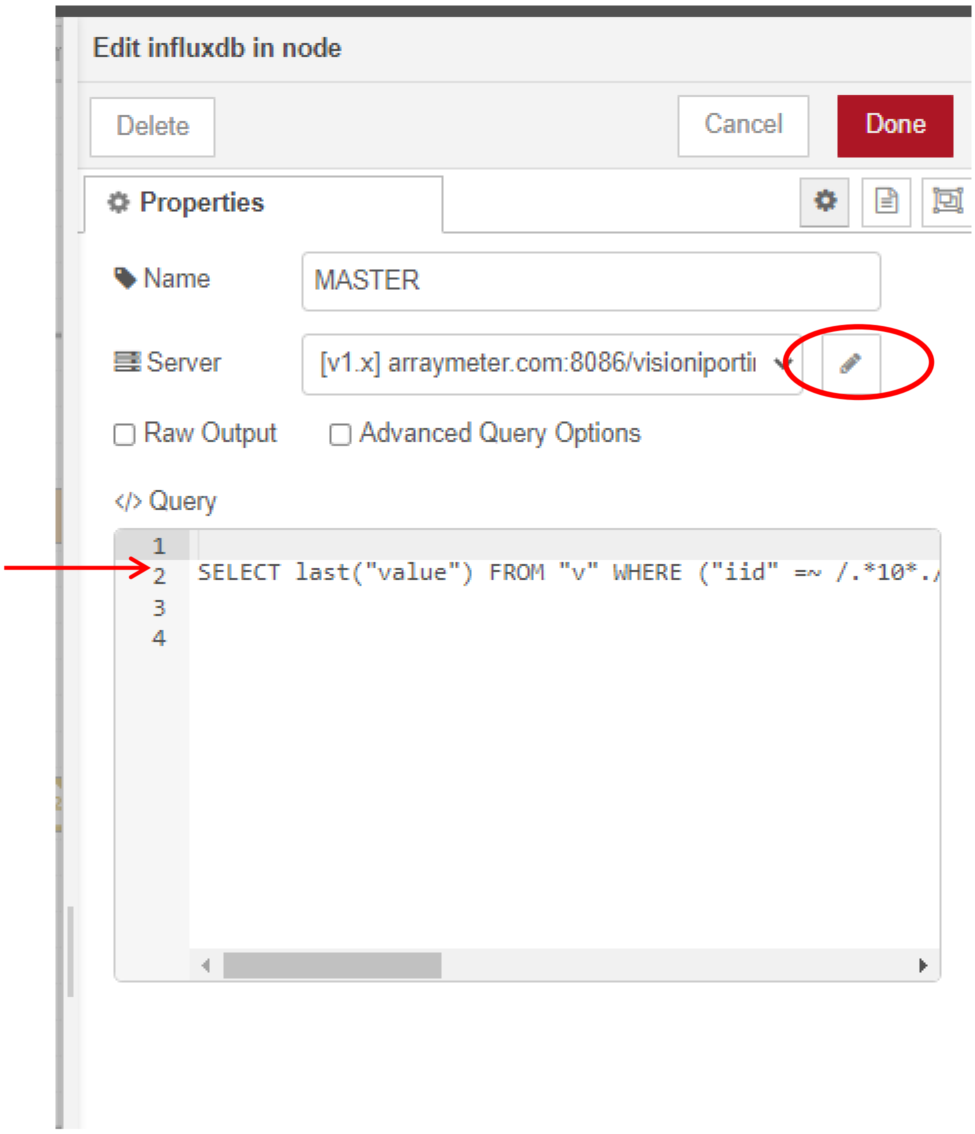
Influxdb in node (node-red-contrib-influxdb (node) - Node-RED (nodered.org))**:**

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



Name: MASTER

Server: [v1.x] arraymeter.com:8086/visioniportii

☐ Raw Output ☐ Advanced Query Options

Query

```
SELECT last("value") FROM "v" WHERE ("iid" =~ /.*10*./
```

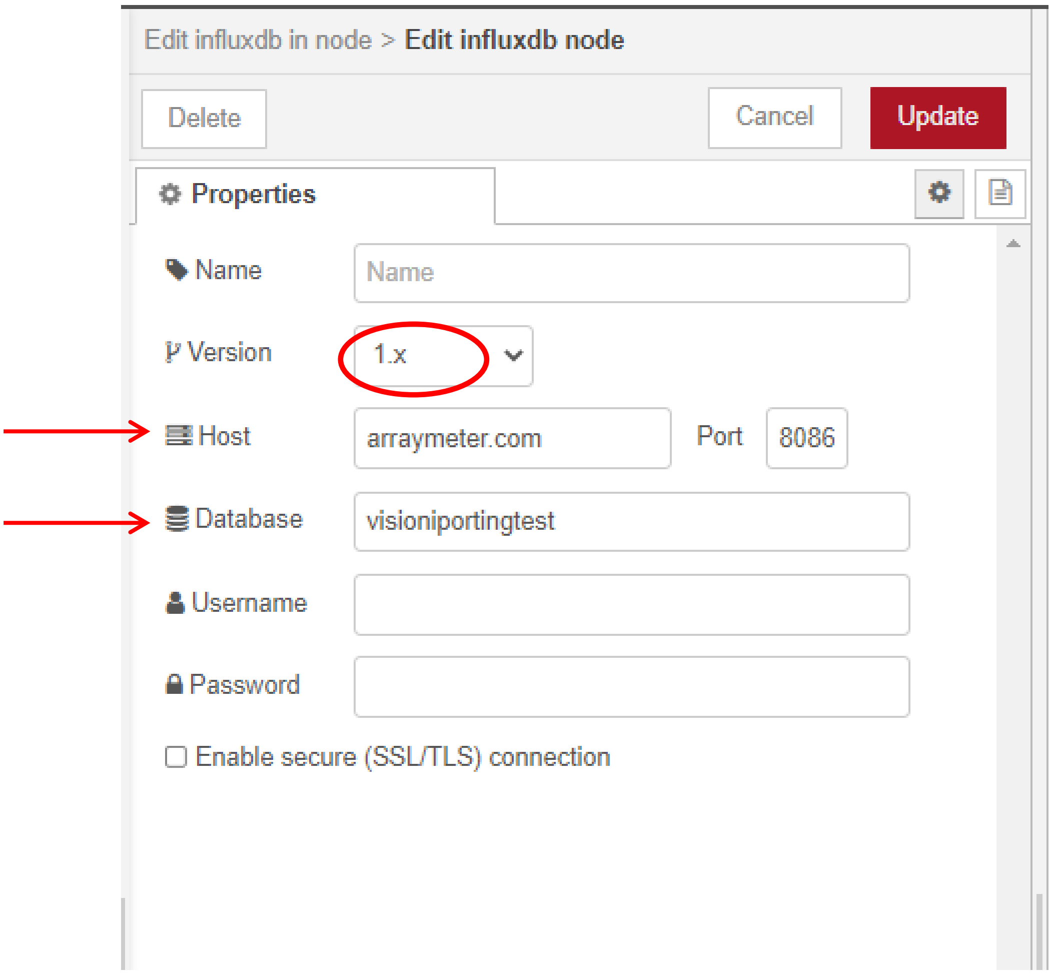
Write the influx query in the query box given

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


Edit influxdb in node > Edit influxdb node

Delete | Cancel | Update

Properties

Name: Name

Version: 1.x

Host: arraymeter.com Port: 8086

Database: visioniportingtest

Username:

Password:

☐ Enable secure (SSL/TLS) connection

Version =1.x

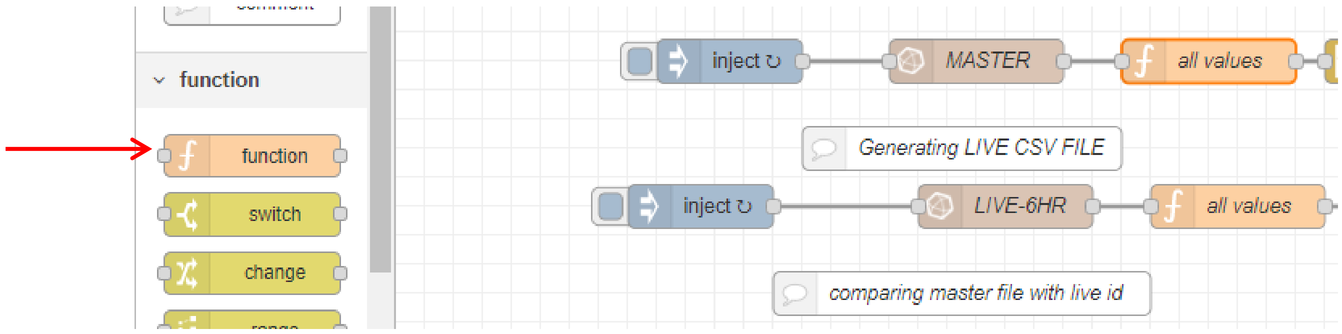
Host = arraymeter.com

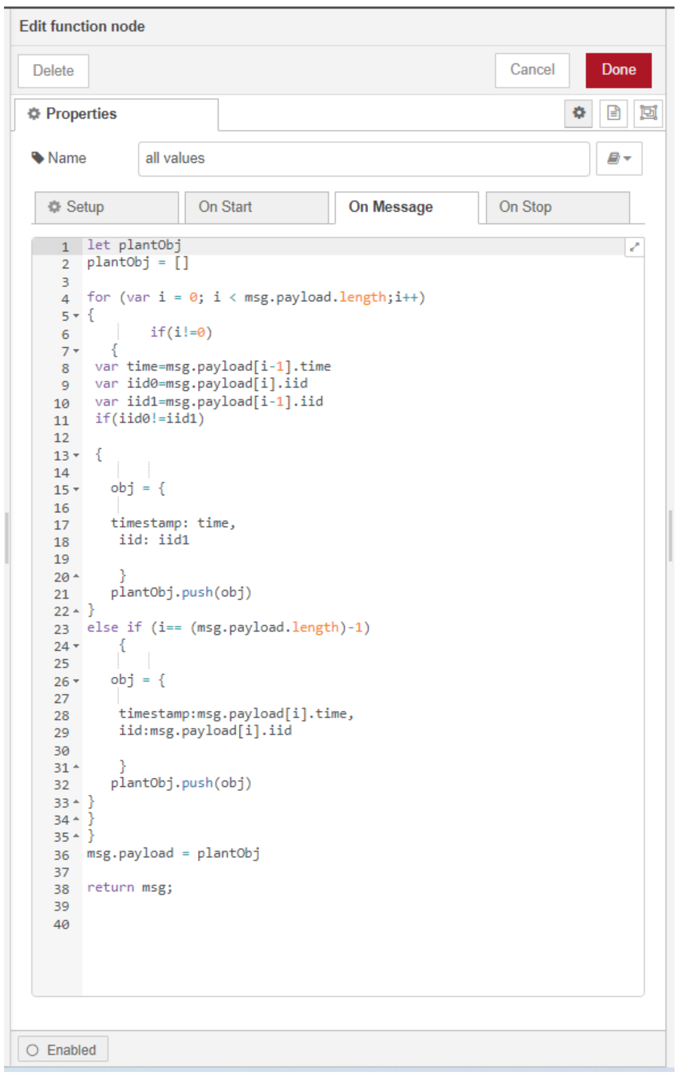
Port = 8086

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.





Name: all values

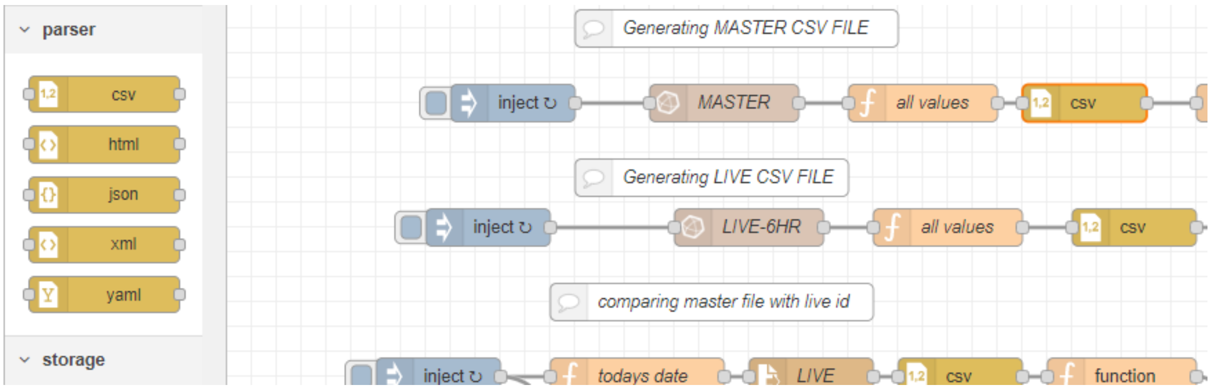
```
let plantObj
plantObj = []

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;
```

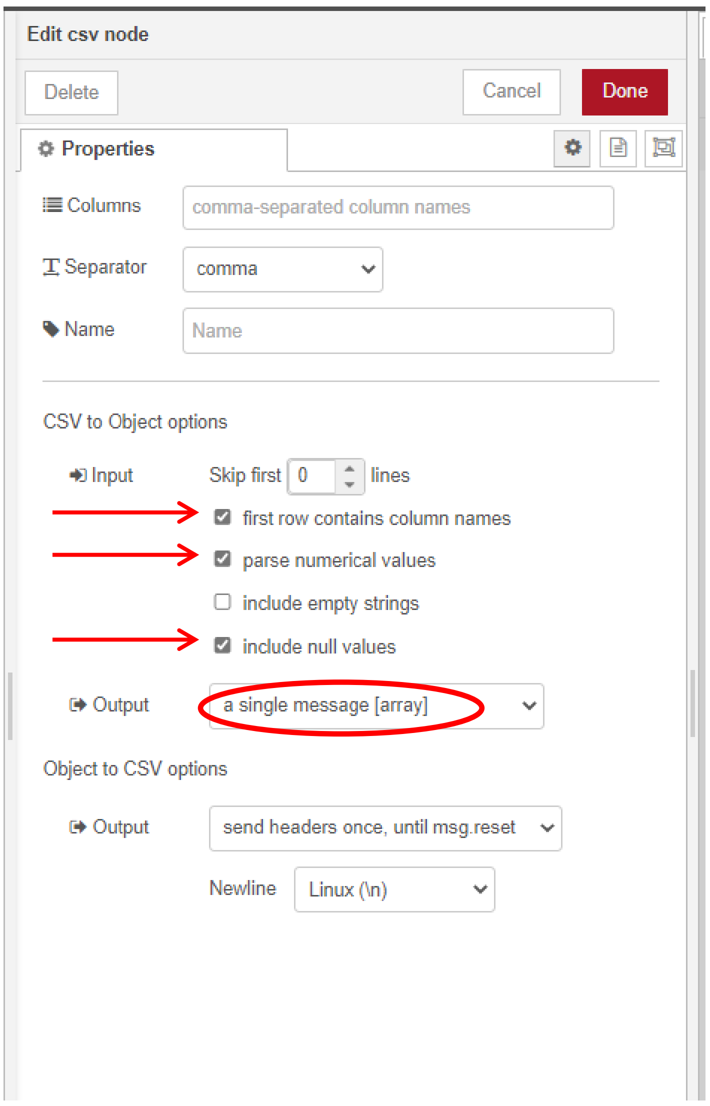
Feed the javascript code on the message box

```
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:





Edit csv node

Columns: comma-separated column names

Separator: comma

Name: Name

CSV to Object options

Input: Skip first 0 lines

- [x] first row contains column names
- [x] parse numerical values
- [ ] include empty strings
- [x] include null values

Output: a single message [array]

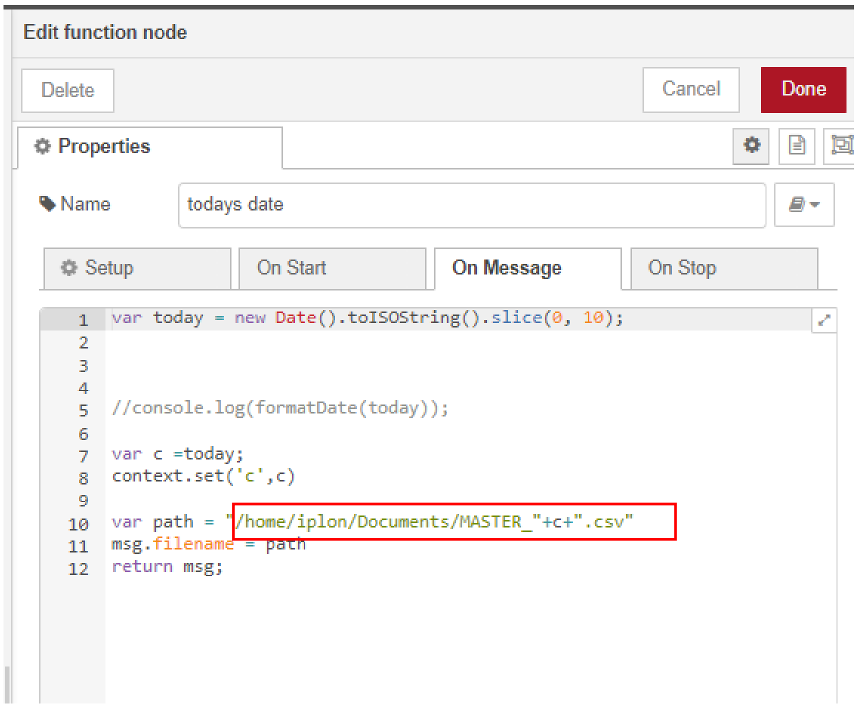
Object to CSV options

Output: send headers once, until msg.reset

Newline: Linux (\n)

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

Function node: To give todays date as file name



Name: todays date

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



//console.log(formatDate(today));

var c =today;
context.set('c',c)

var path = "/home/iplon/Documents/MASTER_"+c+".csv"
msg.filename = path
return msg;
```

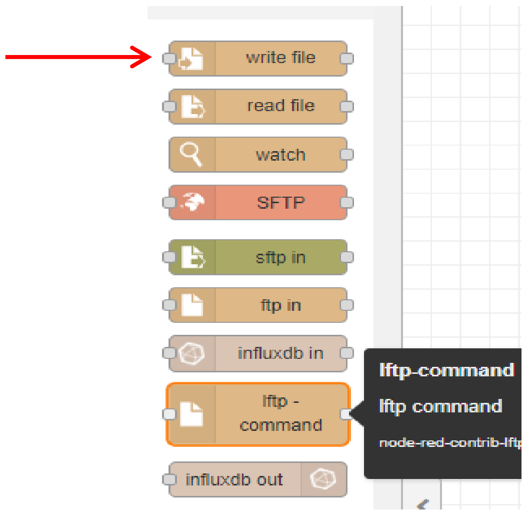
While writing the code in other devices the file path will be different that has to change accordingly

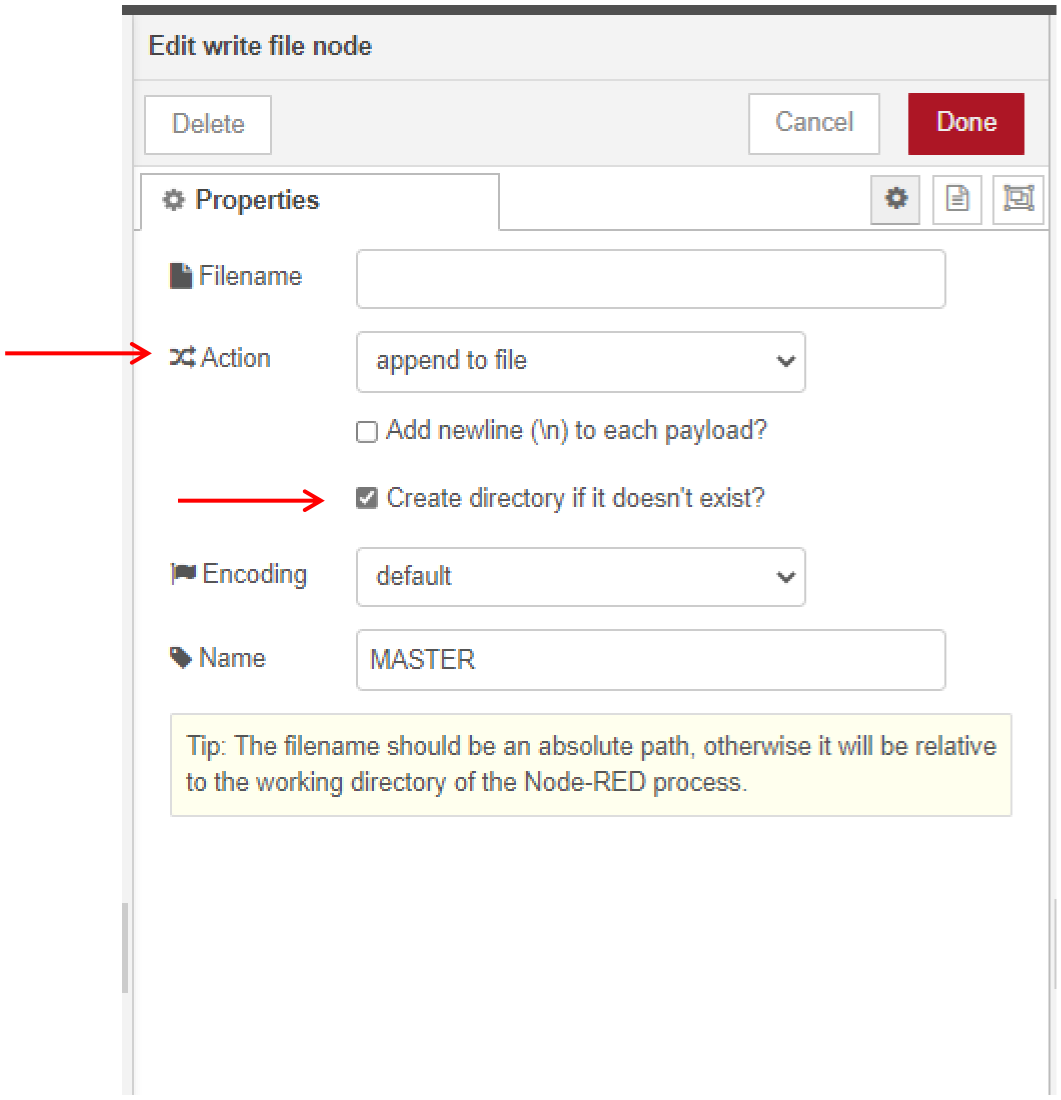
Code:

```
var today = new Date().toISOString().slice(0, 10);
var c =today;
context.set('c',c)

var  path = "/home/iplon/Documents/MASTER_"+c+".csv"
msg.filename = path
return msg;
```

Change the file path according to the device

Write file node:




Edit write file node

Filename

Action: append to file

☐ Add newline (\n) to each payload?

☑ Create directory if it doesn't exist?

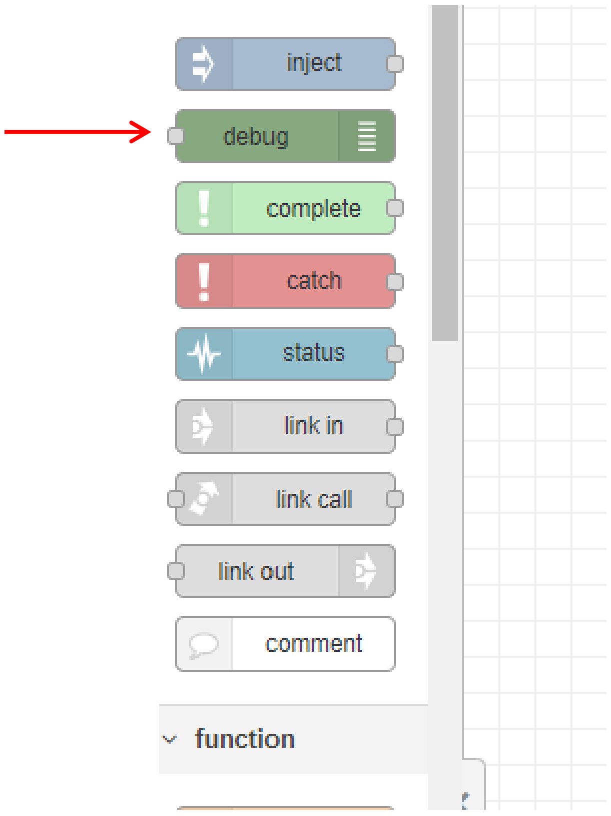
Encoding: default

Name: MASTER

Tip: The filename should be an absolute path, otherwise it will be relative to the working directory of the Node-RED process.

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

Debug node:



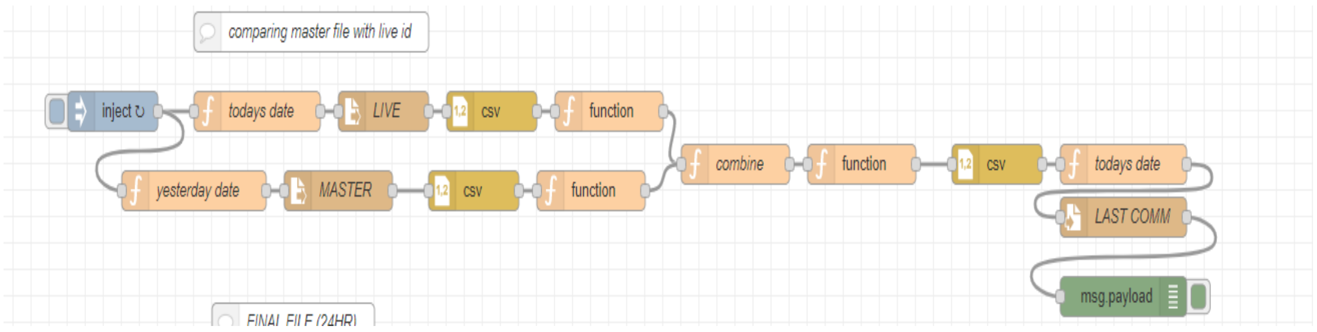
The Debug node causes any message to be displayed in the Debug sidebar. By default, it just displays the payload of the message, but it is possible to display the entire message object.

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

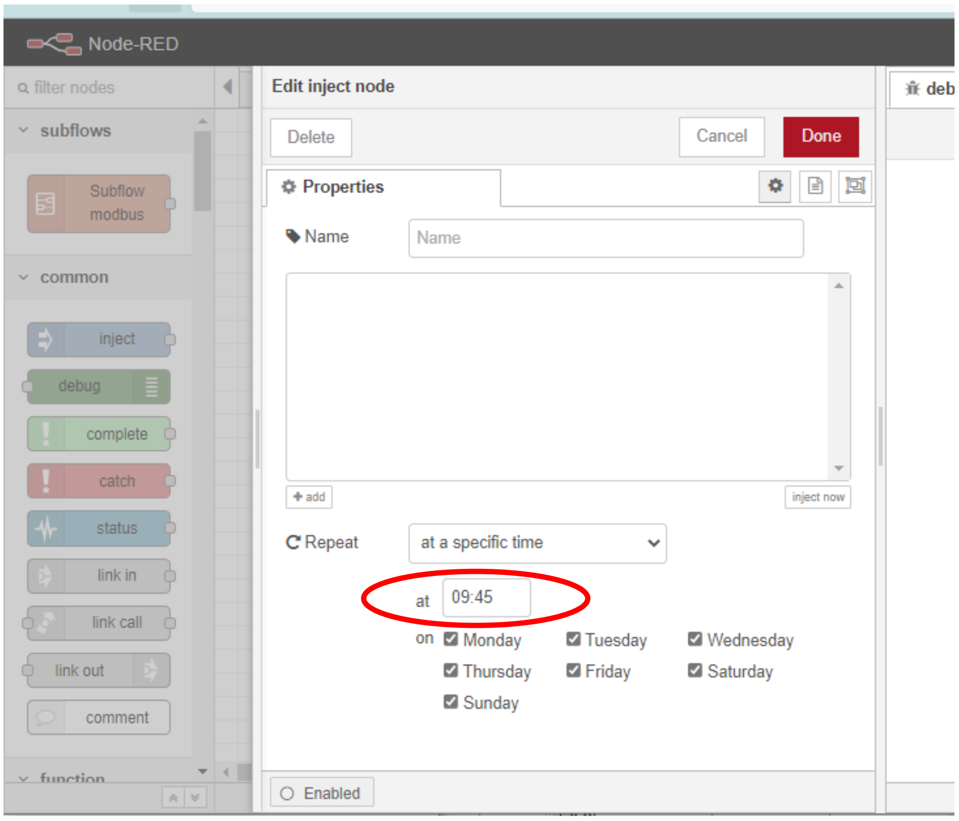
STEP 2:Historical data filling of live meters from their last communicating time to current date timestamp

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



Edit inject node

Name: Name

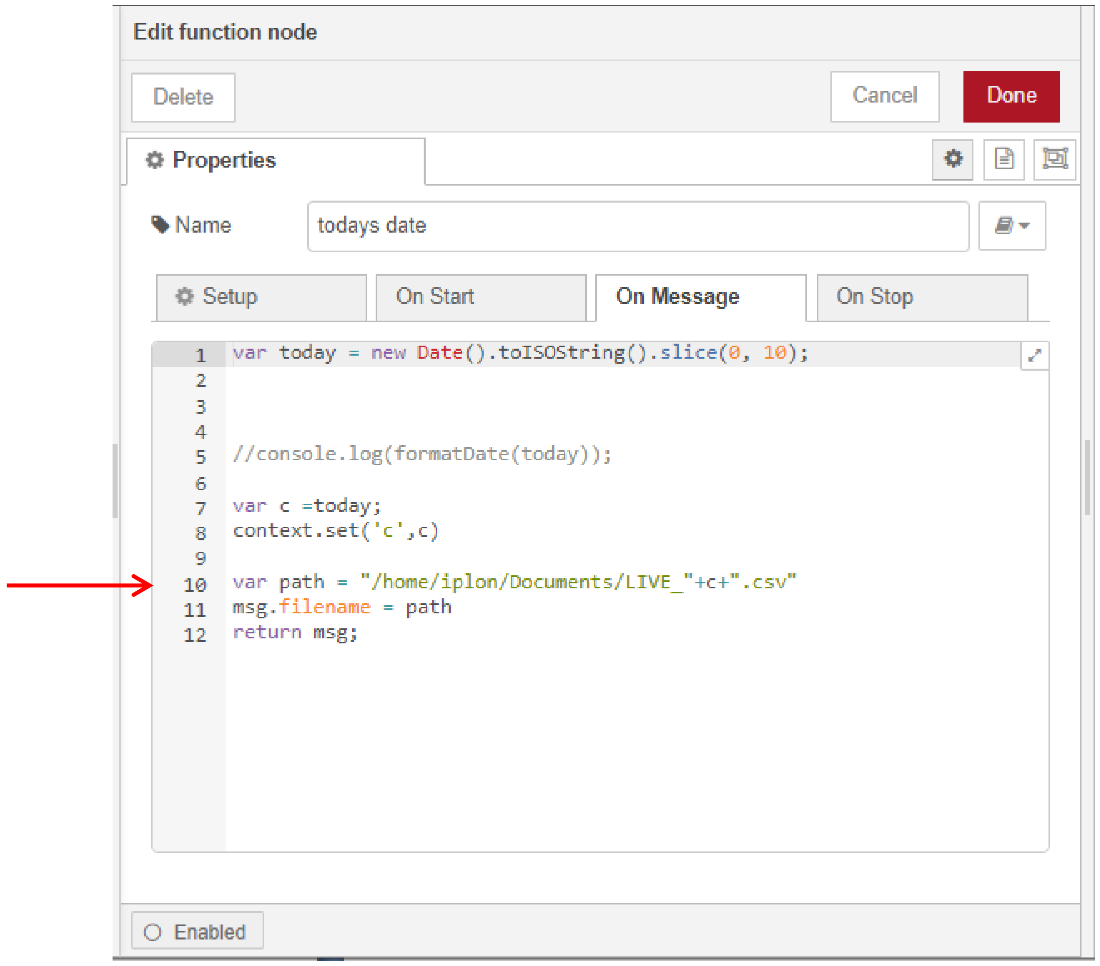
Repeat: at a specific time

at 09:45

on ☑ Monday ☑ Tuesday ☑ Wednesday ☑ Thursday ☑ Friday ☑ Saturday ☑ Sunday

Enabled

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



Name: todays date

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



//console.log(formatDate(today));

var c =today;
context.set('c',c)

var path = "/home/iplon/Documents/LIVE_"+c+".csv"
msg.filename = path
return msg;
```

Code:

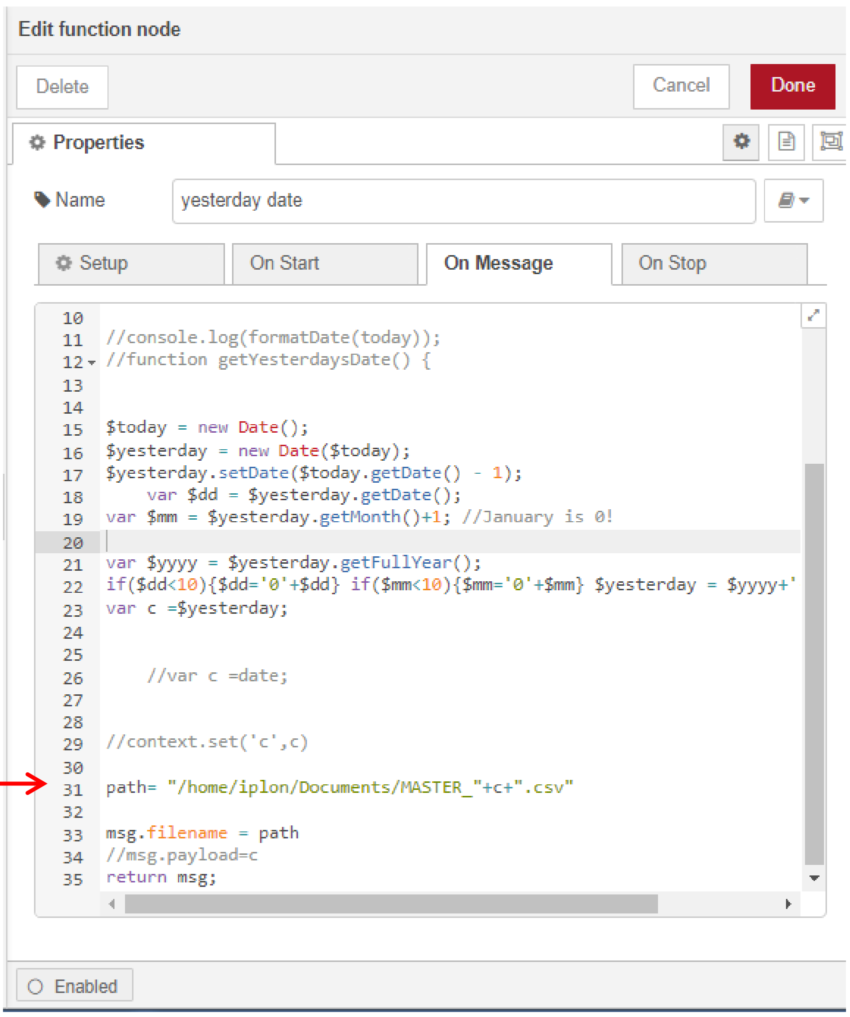
var today = new Date().toISOString().slice(0, 10);

var c =today;
context.set('c',c)

Change the file path according to the device

var path = "/home/iplon/Documents/LIVE_"+c+".csv"
msg.filename = path
return msg;

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



Name: yesterday date

```

//console.log(formatDate(today));
//function getYesterdaysDate() {


$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 = $yyyy+'
var c =$yesterday;


    //var c =date;


//context.set('c',c)

path= "/home/iplon/Documents/MASTER_"+c+".csv"

msg.filename = path
//msg.payload=c
return msg;
```

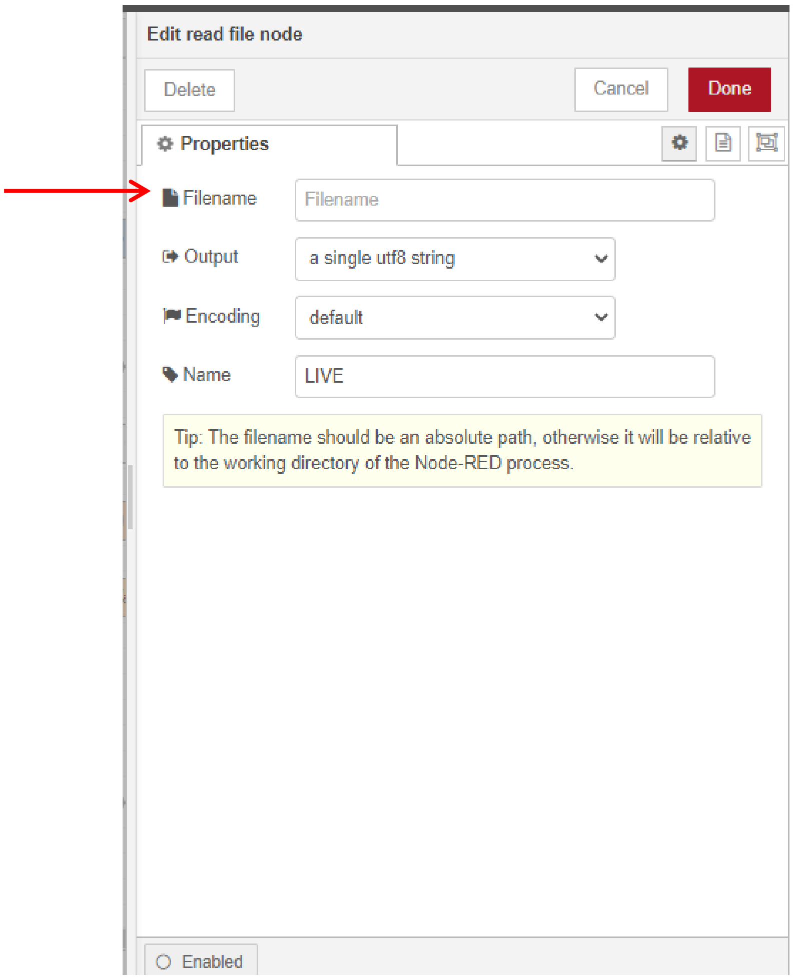
Code :

```
$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;

var c =$yesterday;
path= "/home/iplon/Documents/MASTER_"+c+".csv"

msg.filename = path
//msg.payload=c
return msg;
```

Change the file path according to the device

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



Edit read file node

Filename: Filename

Output: a single utf8 string

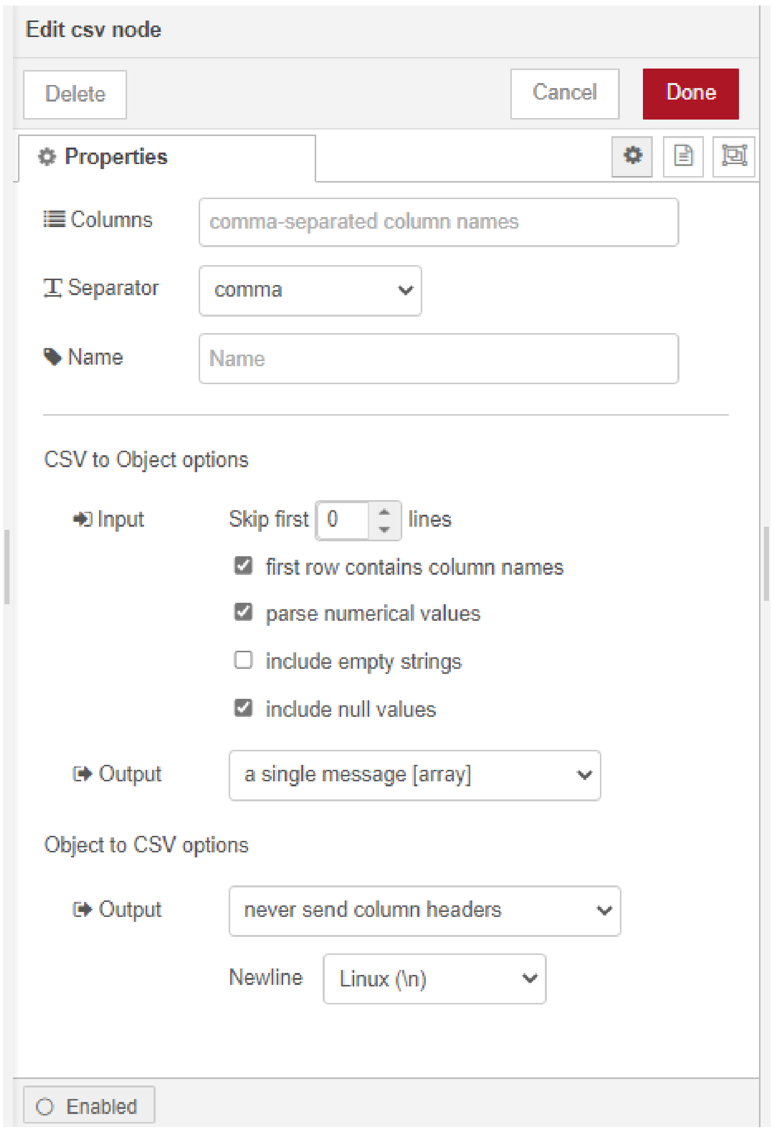
Encoding: default

Name: LIVE

Tip: The filename should be an absolute path, otherwise it will be relative to the working directory of the Node-RED process.

csv node:

configure both the csv nodes same as in the figure



Edit csv node

Columns: comma-separated column names

Separator: comma

Name: Name

CSV to Object options

Input: Skip first 0 lines

- [x] first row contains column names
- [x] parse numerical values
- [ ] include empty strings
- [x] include null values

Output: a single message [array]

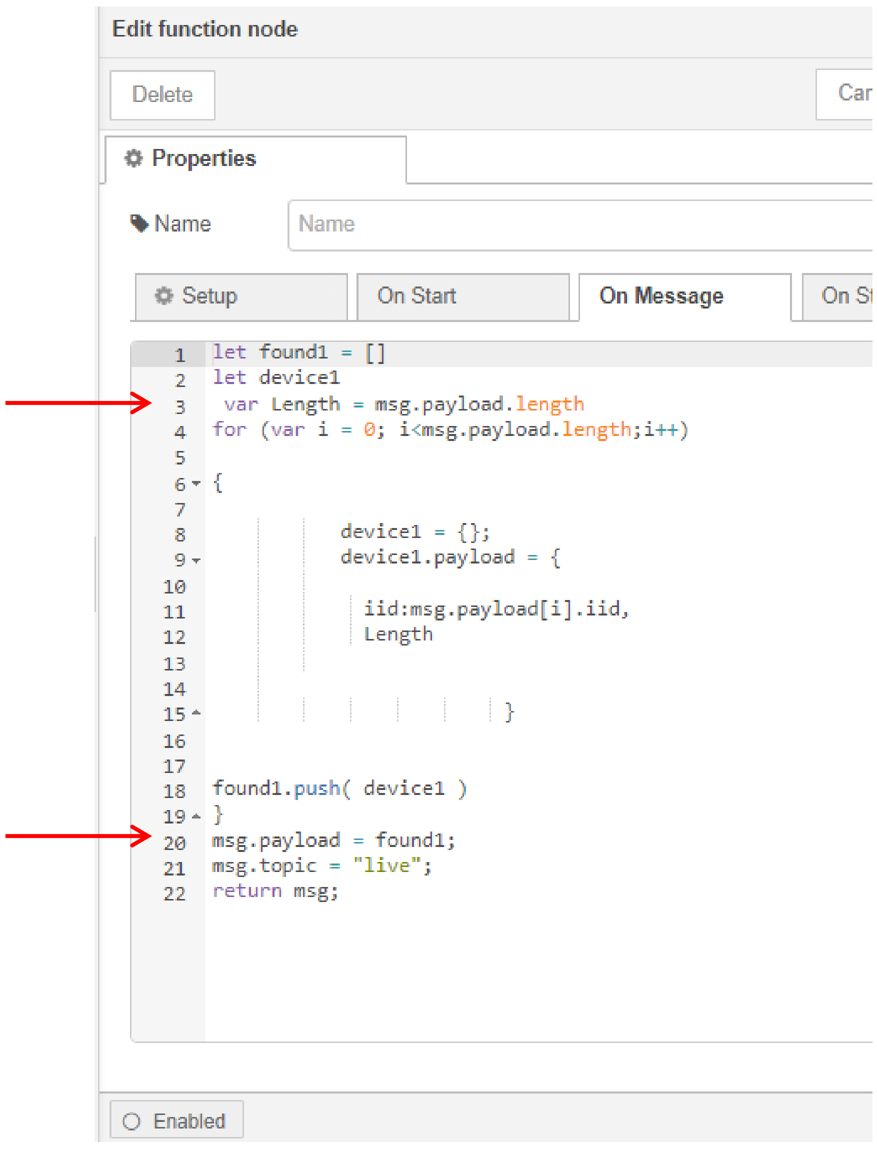
Object to CSV options

Output: never send column headers

Newline: Linux (\n)

Function node:

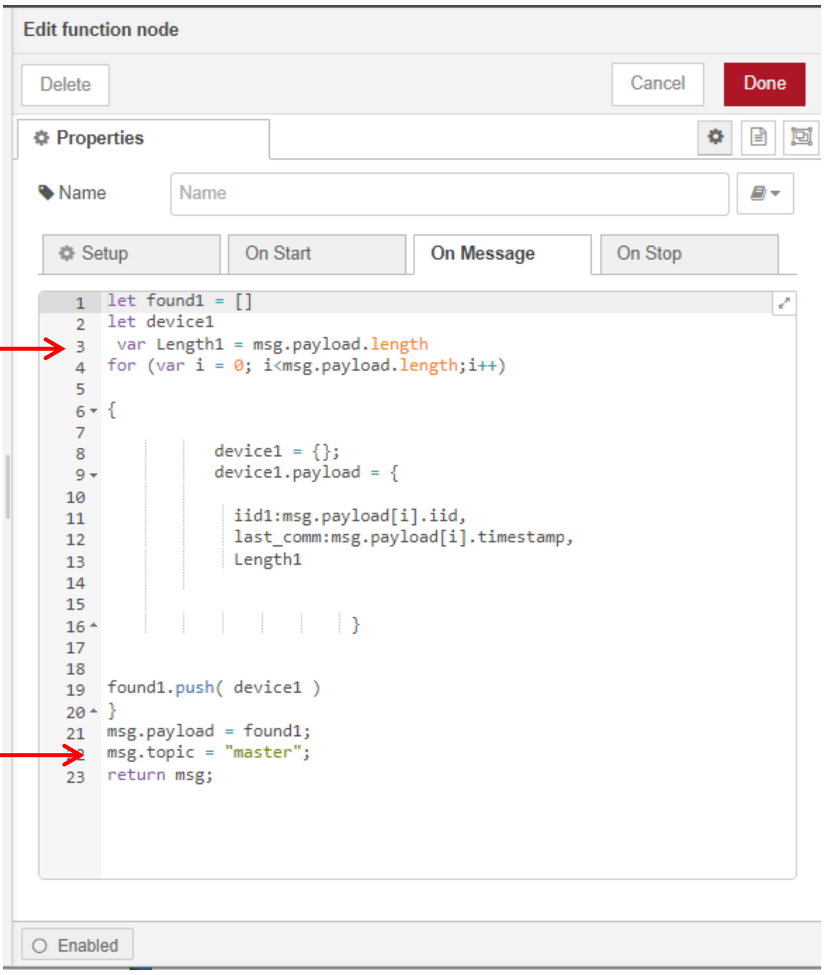
We are using 2 functions node here both for taking length of the array and meters id in the corresponding file and also we are giving topic to both flows to merge both flows



```
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 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;
```

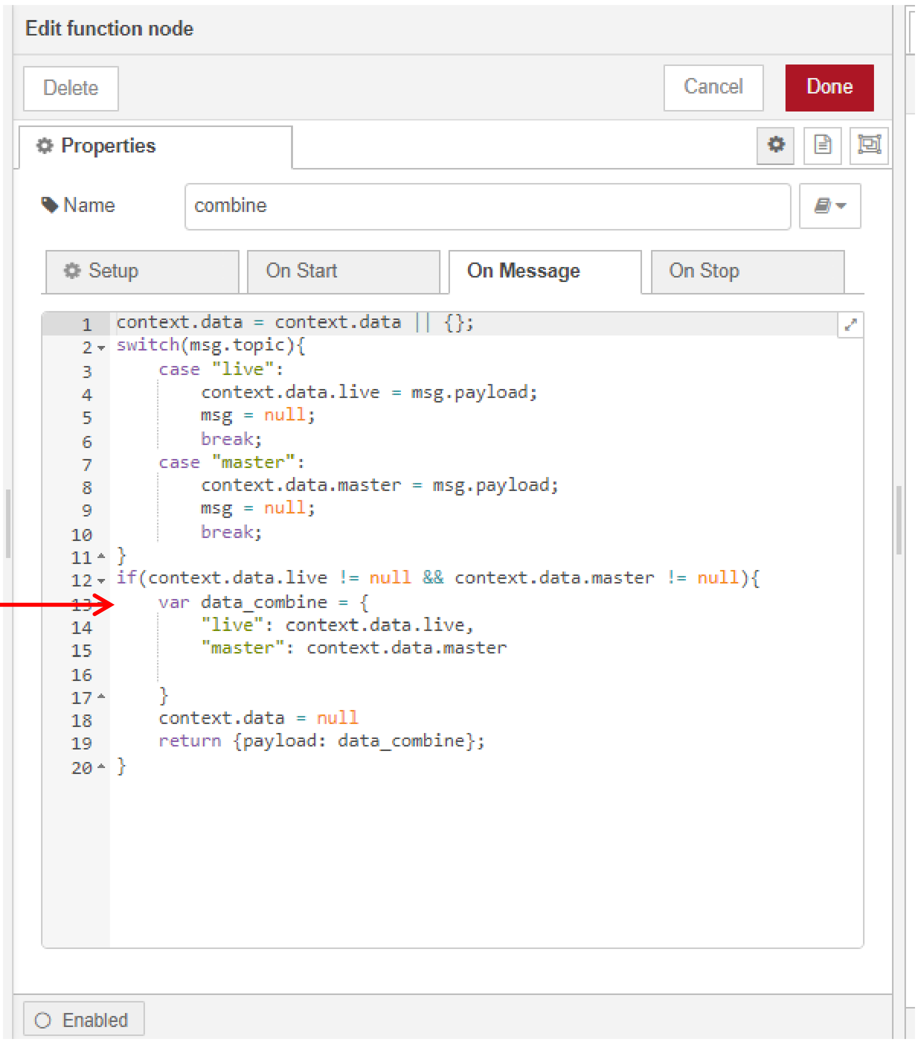


```
let found1 = []
let device1
 var Length1 = msg.payload.length
for (var i = 0; i<msg.payload.length;i++)

{

        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;
```

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):



Name: combine

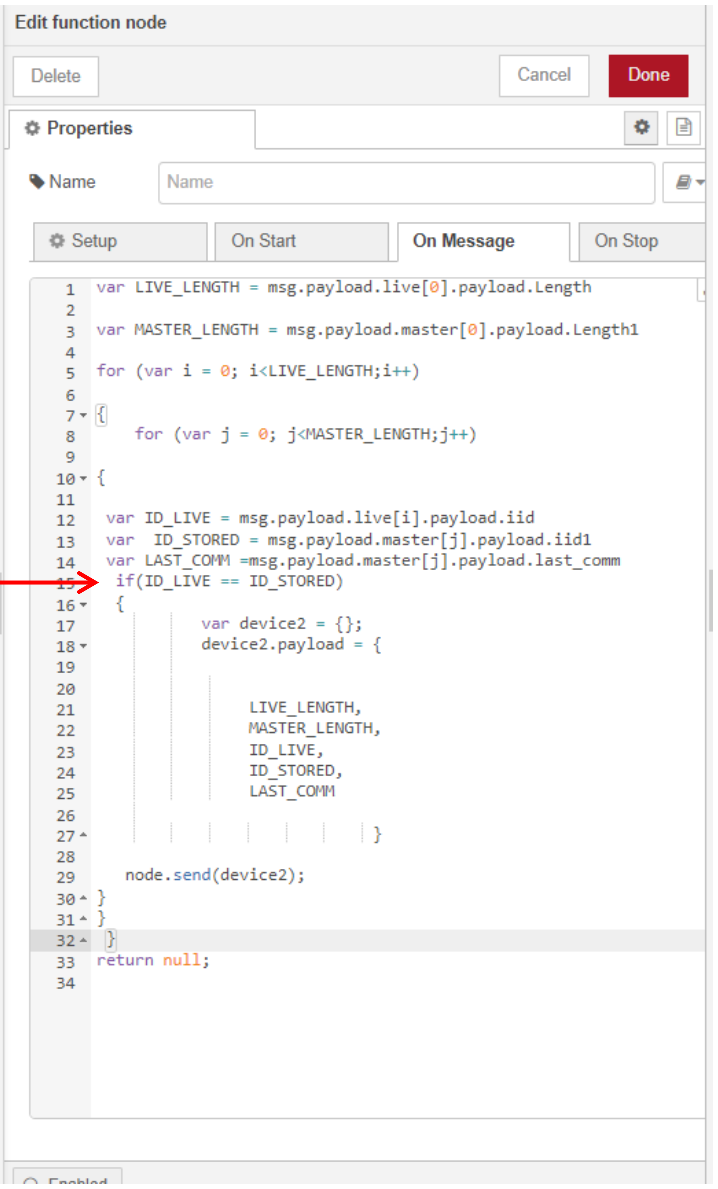
```
context.data = context.data || {};
switch(msg.topic){
    case "live":
        context.data.live = msg.payload;
        msg = null;
        break;
    case "master":
        context.data.master = msg.payload;
        msg = null;
        break;
}
if(context.data.live != null && context.data.master != null){
    var data_combine = {
        "live": context.data.live,
        "master": context.data.master

    }
    context.data = null
    return {payload: data_combine};
}
```

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

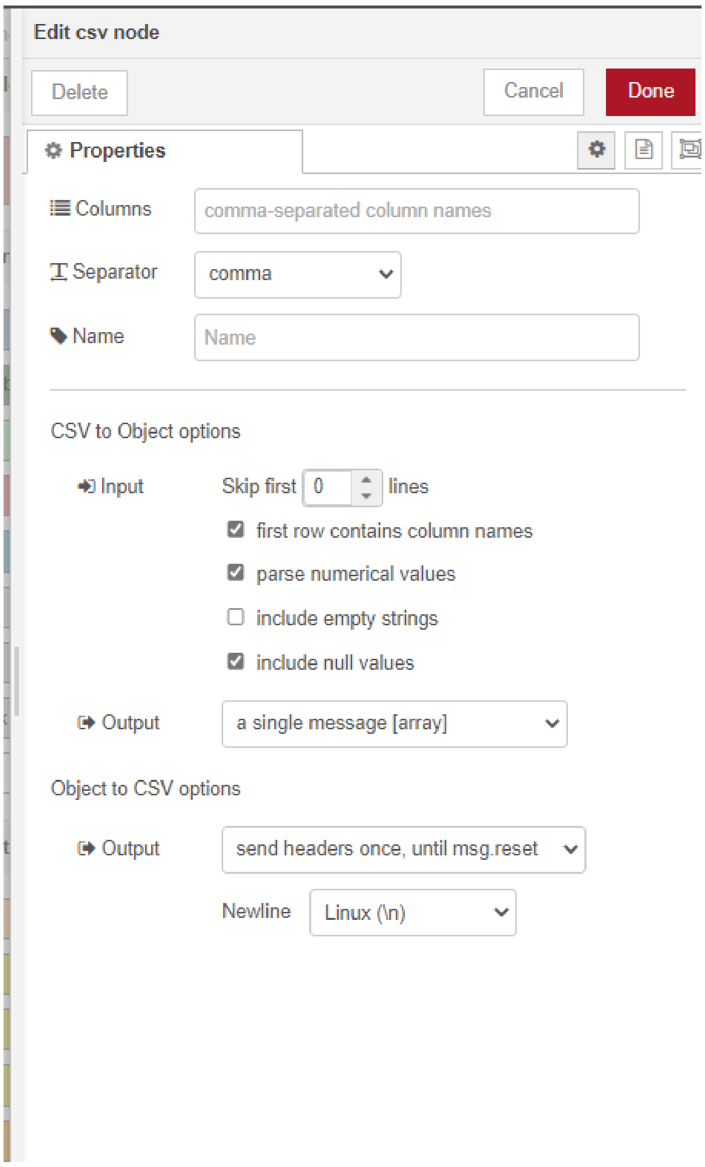


```
var LIVE_LENGTH = msg.payload.live[0].payload.Length

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;

```

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:



Edit csv node

Delete | Cancel | Done

Properties

Columns: comma-separated column names

Separator: comma

Name: Name

CSV to Object options

Input: Skip first 0 lines

- [x] first row contains column names
- [x] parse numerical values
- [ ] include empty strings
- [x] include null values

Output: a single message [array]

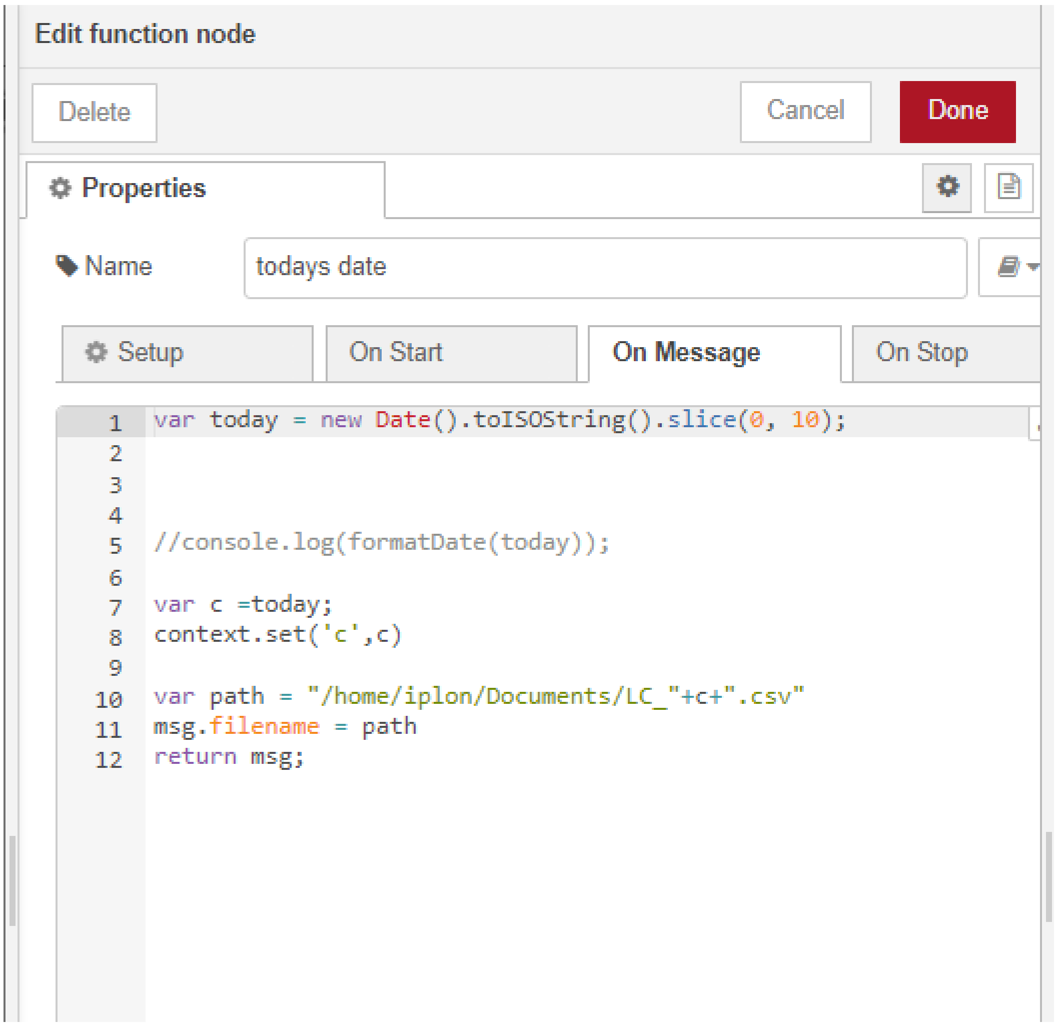
Object to CSV options

Output: send headers once, until msg.reset

Newline: Linux (\n)

Configure csv node same as in the figure

Function node for todays date:



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



//console.log(formatDate(today));

var c =today;
context.set('c',c)

var path = "/home/iplon/Documents/LC_"+c+".csv"
msg.filename = path
return msg;
```

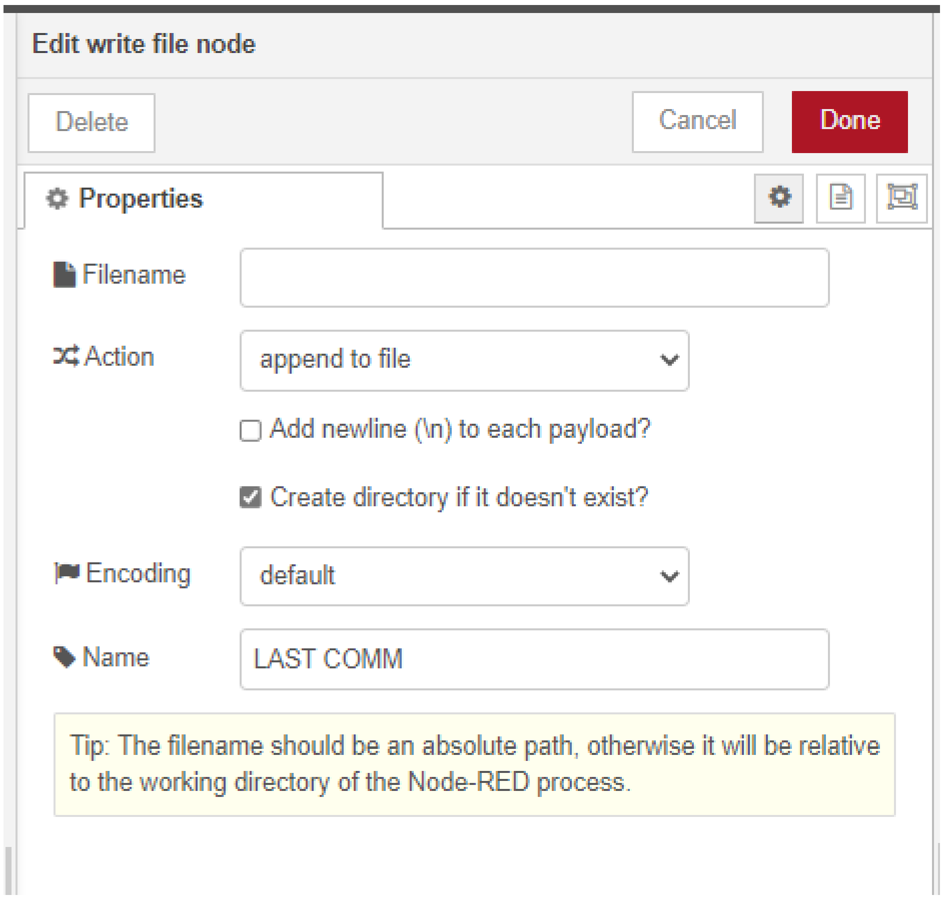
Code:

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

var c =today;

context.set('c',c)

var path = "/home/iplon/Documents/LC_"+c+".csv"

msg.filename = path

return msg;
```

Change the file path according to the device

Write File Node:

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



Edit write file node

Filename:

Action: append to file

☐ Add newline (\n) to each payload?

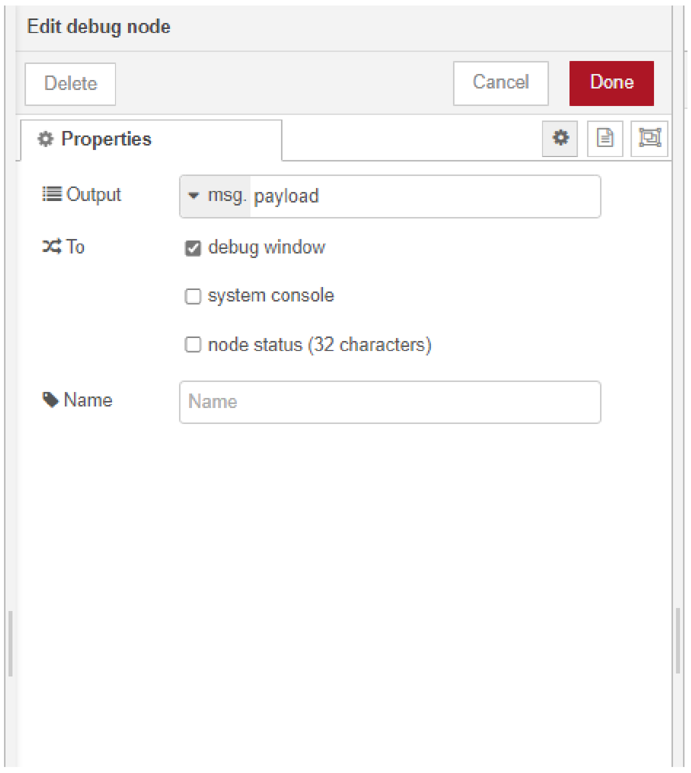
☑ Create directory if it doesn't exist?

Encoding: default

Name: LAST COMM

Tip: The filename should be an absolute path, otherwise it will be relative to the working directory of the Node-RED process.

Debug node:


Edit debug node

Output: msg. payload

To:
- [✓] debug window
- [ ] system console
- [ ] node status (32 characters)

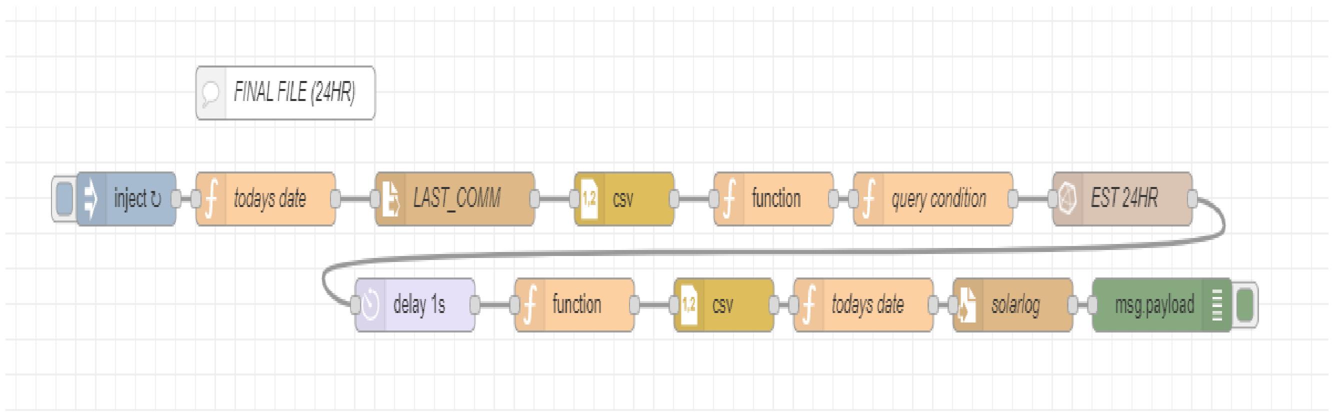
Name: Name

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.

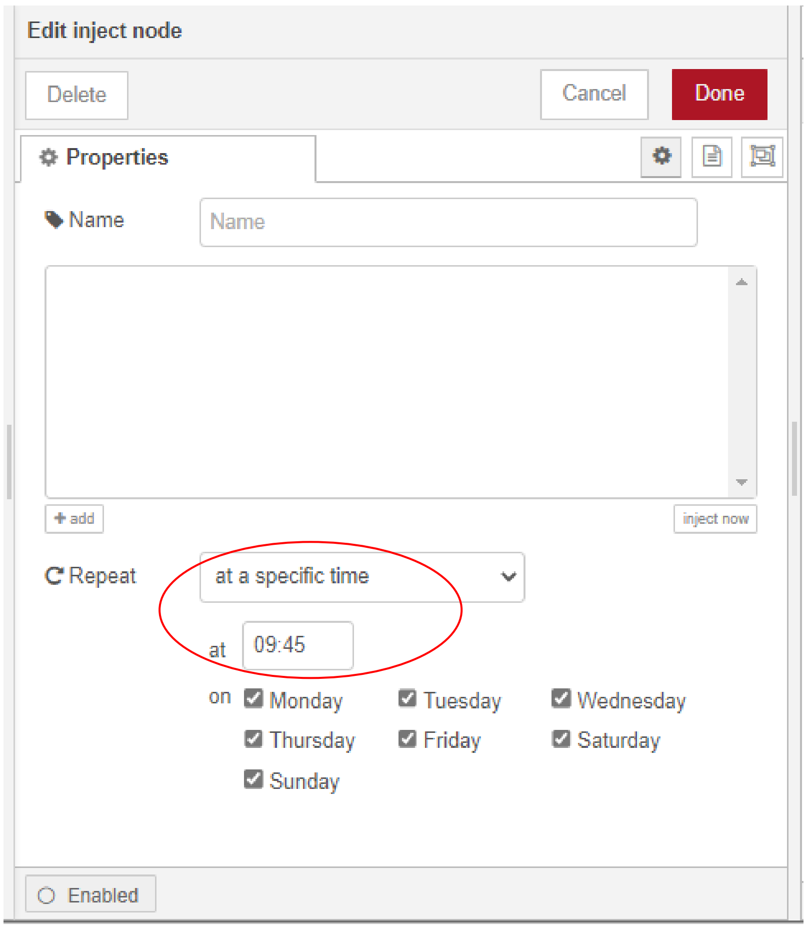
1. Click the Deploy button.With the Debug sidebar tab selected,
2. Click the Inject button to get output if there is no triggerng time set.
3. Check the destination of file what we give as path to check whether the csv file created or not.

NODE-RED FLOW FOR STEP 3

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.



INJECT NODE SETUP



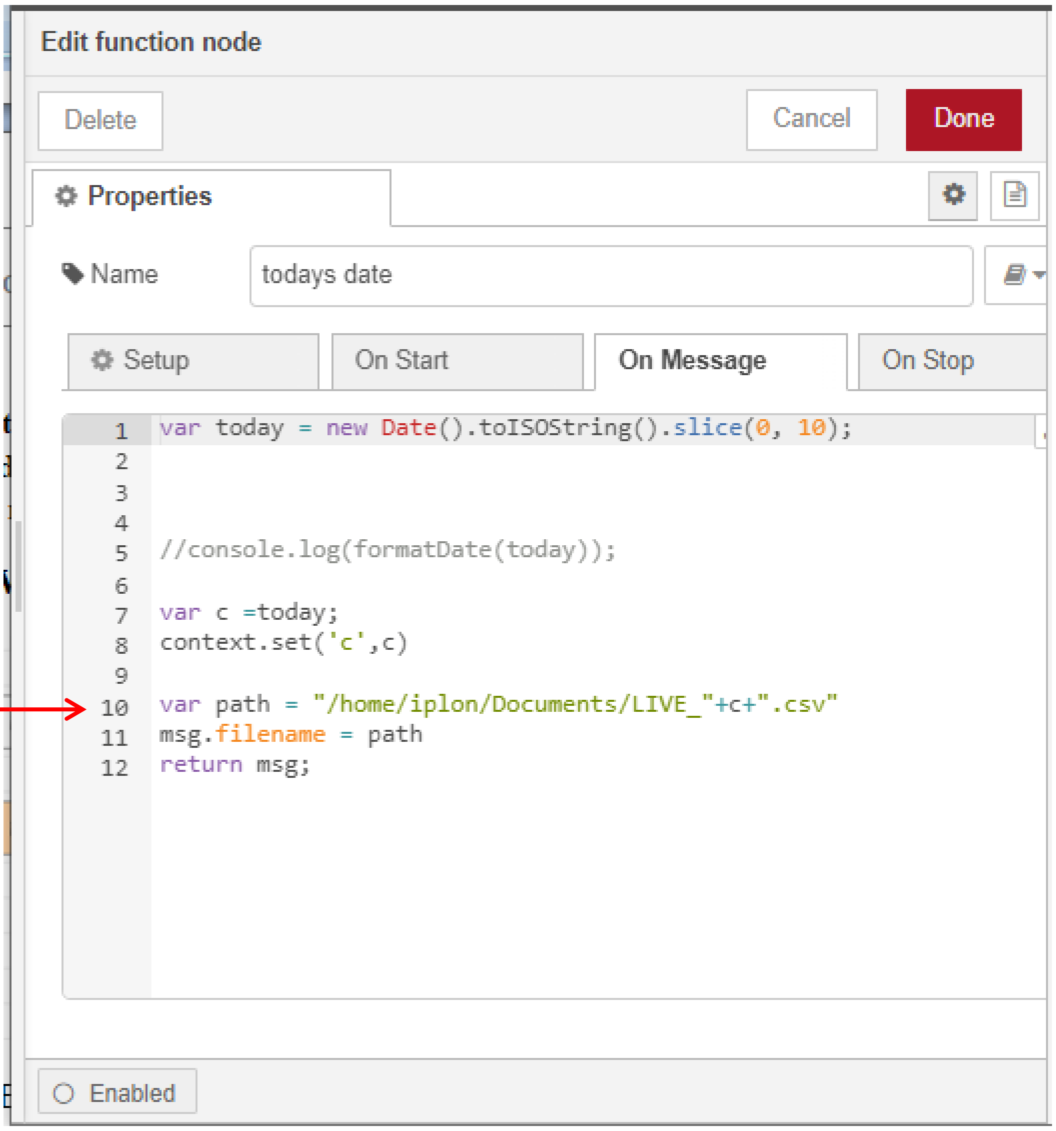
Name: Name

Repeat: at a specific time

at 09:45

on ☑ Monday ☑ Tuesday ☑ Wednesday ☑ Thursday ☑ Friday ☑ Saturday ☑ Sunday

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

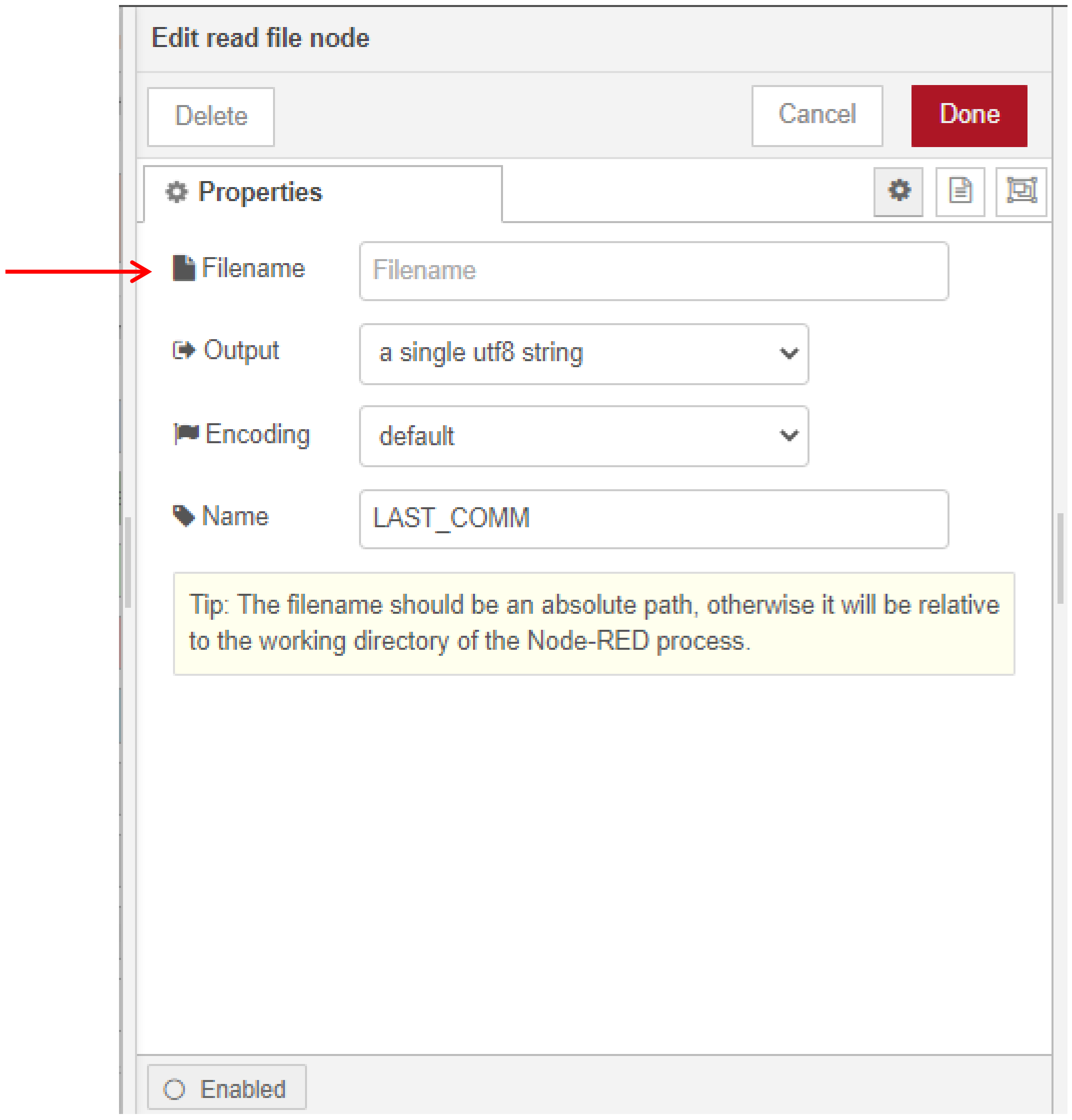


Name: todays date

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



//console.log(formatDate(today));

var c =today;
context.set('c',c)

var path = "/home/iplon/Documents/LIVE_"+c+".csv"
msg.filename = path
return msg;
```

READ FILE NODE:

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



Edit read file node

Delete Cancel Done

Properties

Filename: Filename

Output: a single utf8 string

Encoding: default

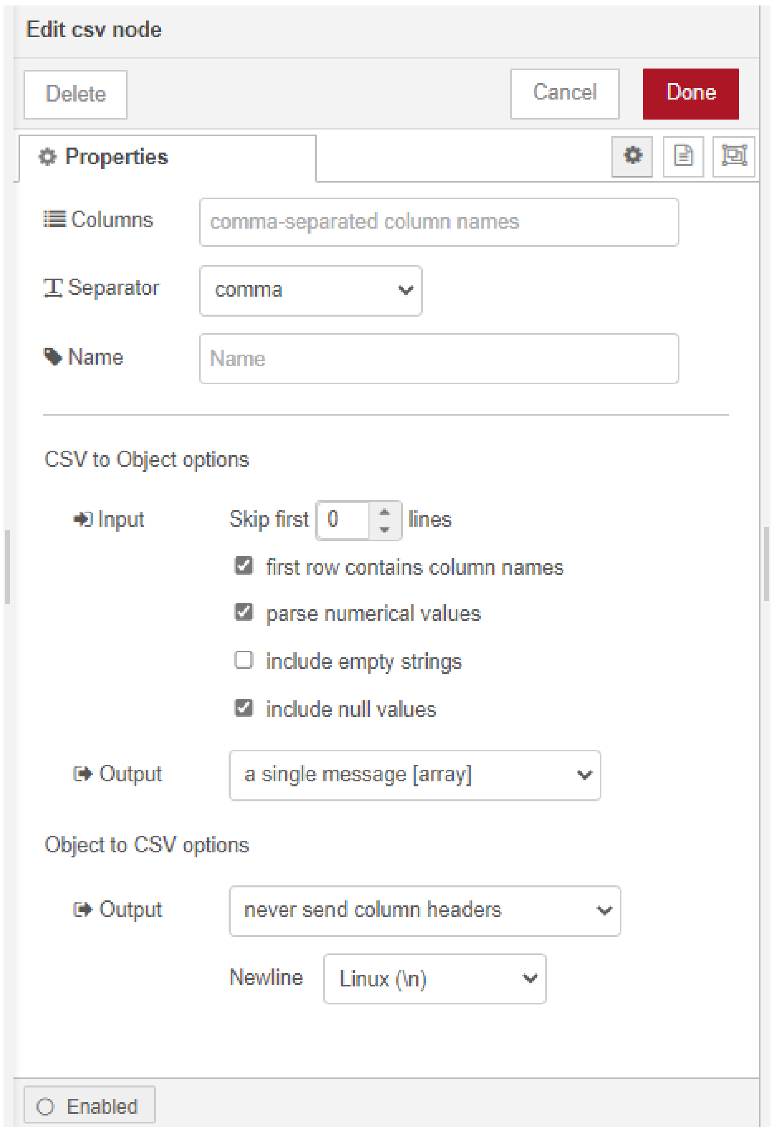
Name: LAST_COMM

Tip: The filename should be an absolute path, otherwise it will be relative to the working directory of the Node-RED process.

Enabled

CSV NODE:

Configure csv node same as in below figure


Edit csv node

Delete Cancel Done

Properties

Columns: comma-separated column names

Separator: comma

Name: Name

CSV to Object options

Input: Skip first 0 lines

- [x] first row contains column names
- [x] parse numerical values
- [ ] include empty strings
- [x] include null values

Output: a single message [array]

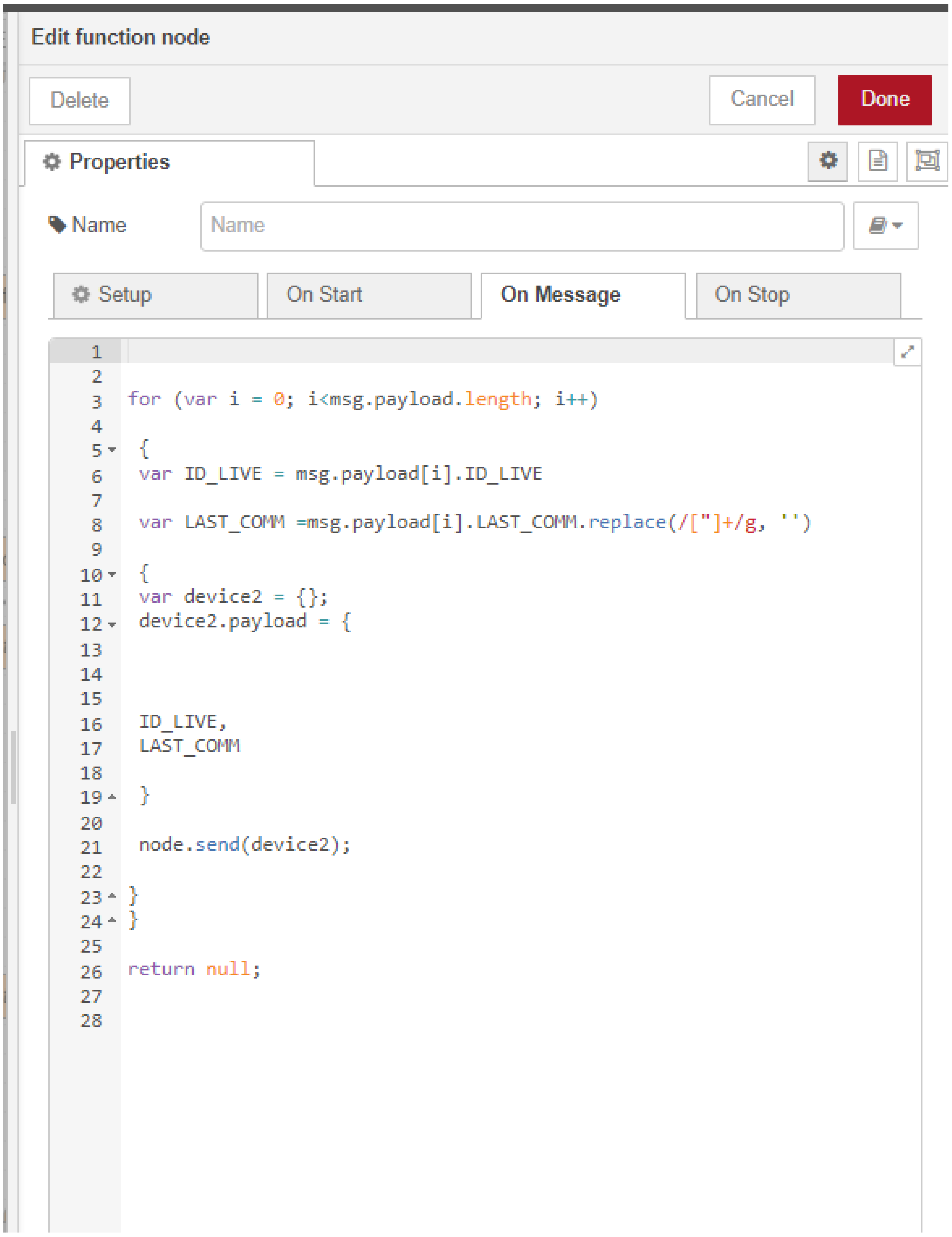
Object to CSV options

Output: never send column headers

Newline: Linux (\n)

Enabled

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



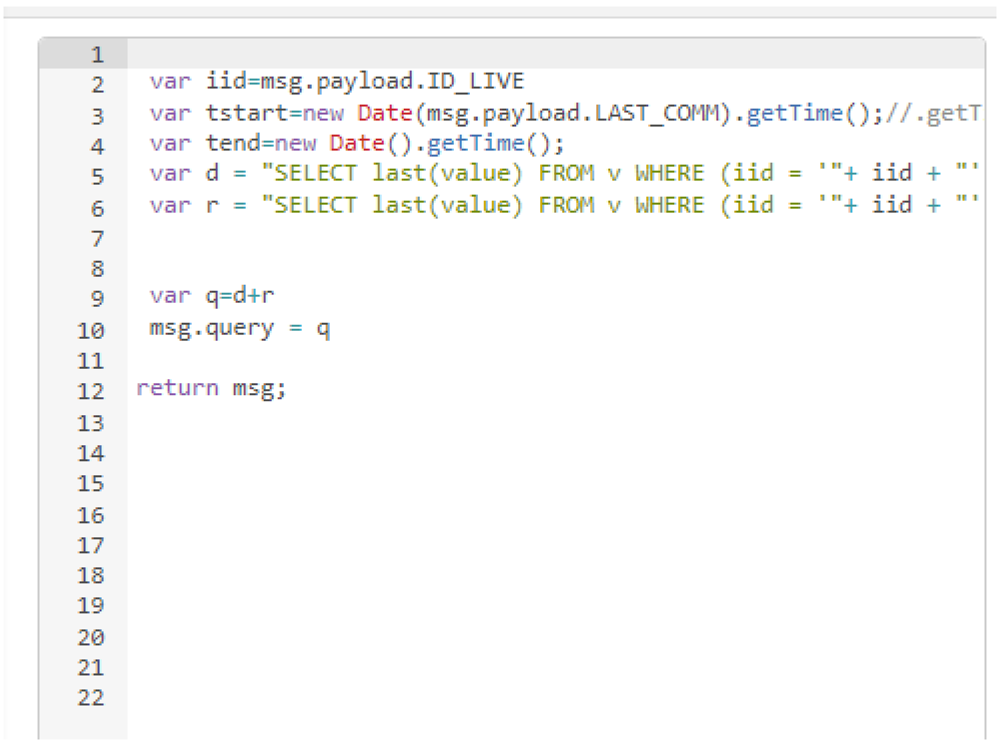
```

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;


```

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.



```
 var iid=msg.payload.ID_LIVE
 var tstart=new Date(msg.payload.LAST_COMM).getTime();//.getT
 var tend=new Date().getTime();
 var d = "SELECT last(value) FROM v WHERE (iid = '"+ iid + "'
 var r = "SELECT last(value) FROM v WHERE (iid = '"+ iid + "'


 var q=d+r
 msg.query = q

return msg;
```

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;
```

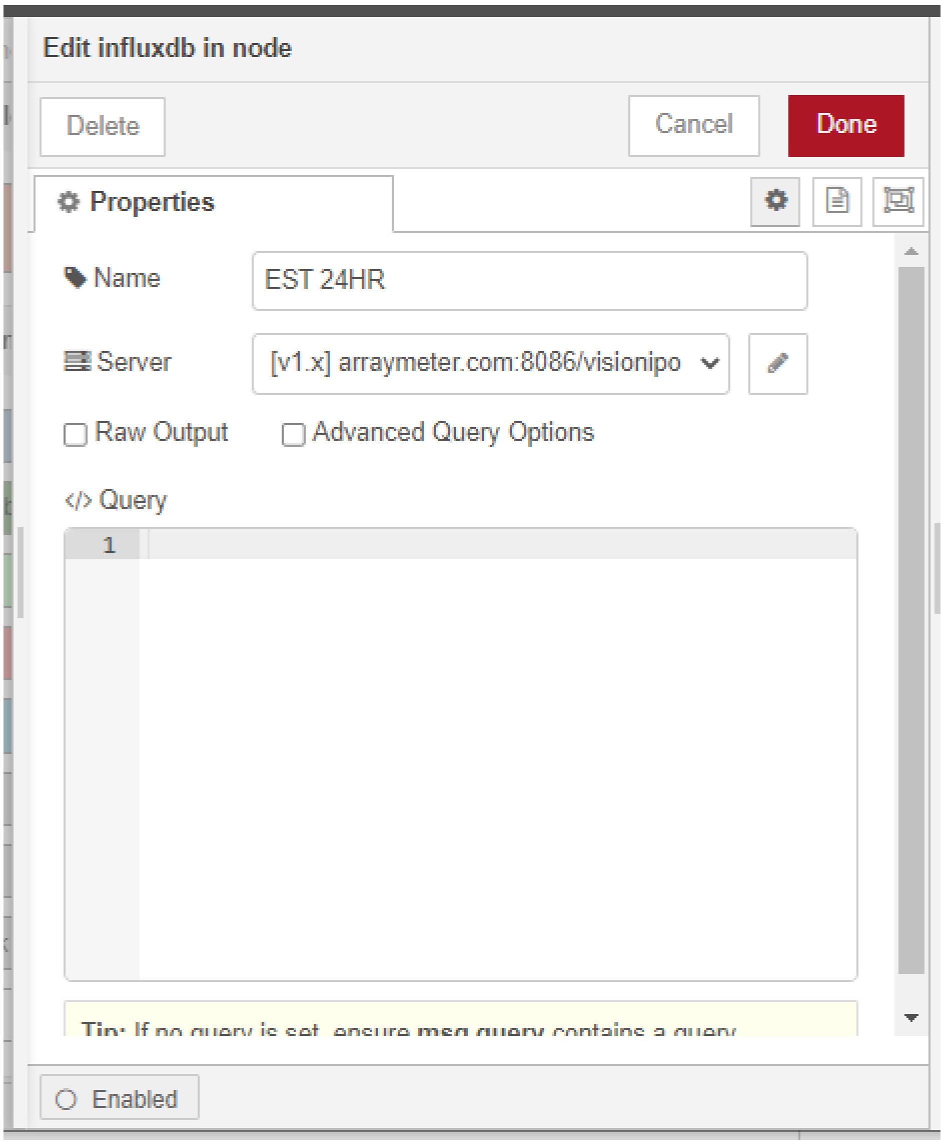
INFLUXDB IN NODE:

- 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



Edit influxdb in node

Name: EST 24HR

Server: [v1.x] arraymeter.com:8086/visionipo

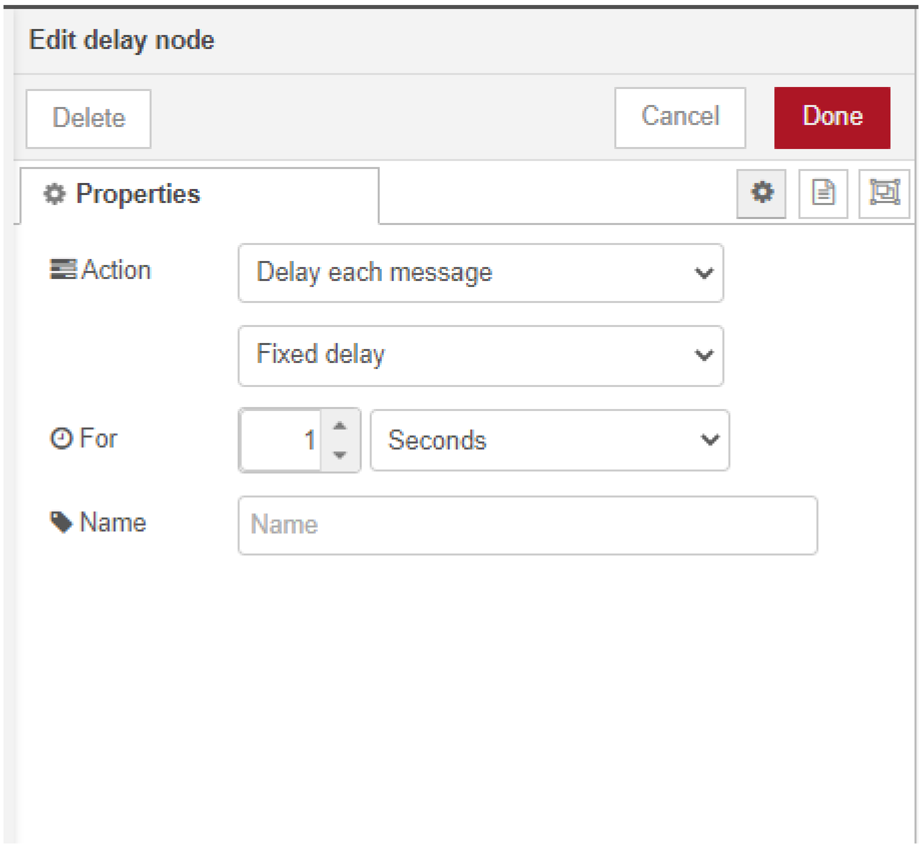
☐ Raw Output ☐ Advanced Query Options

Query

Tip: If no query is set, ensure msg.query contains a query

DELAY NODE:

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


Edit delay node

Action: Delay each message

Fixed delay

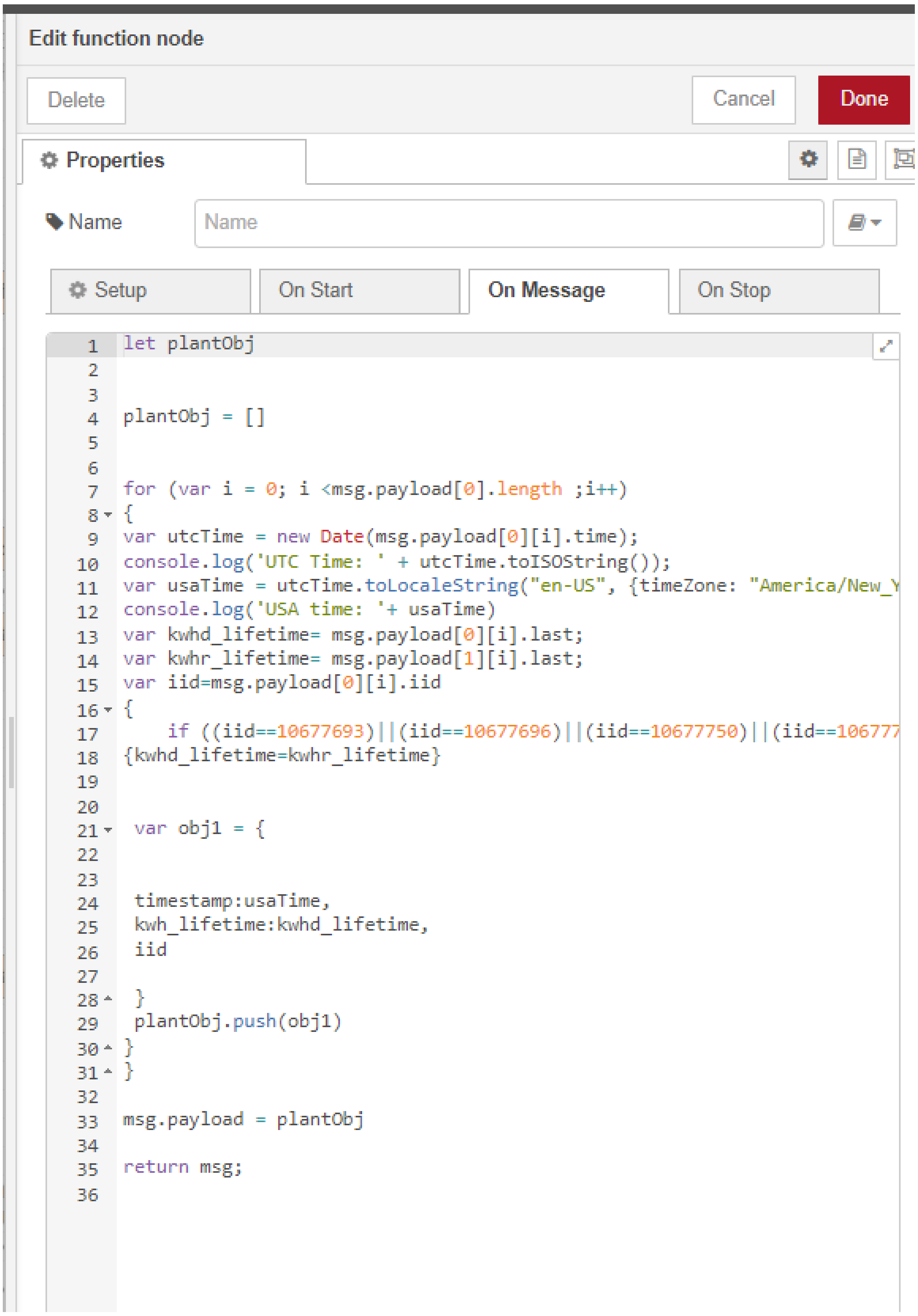
For: 1 Seconds

Name: Name

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

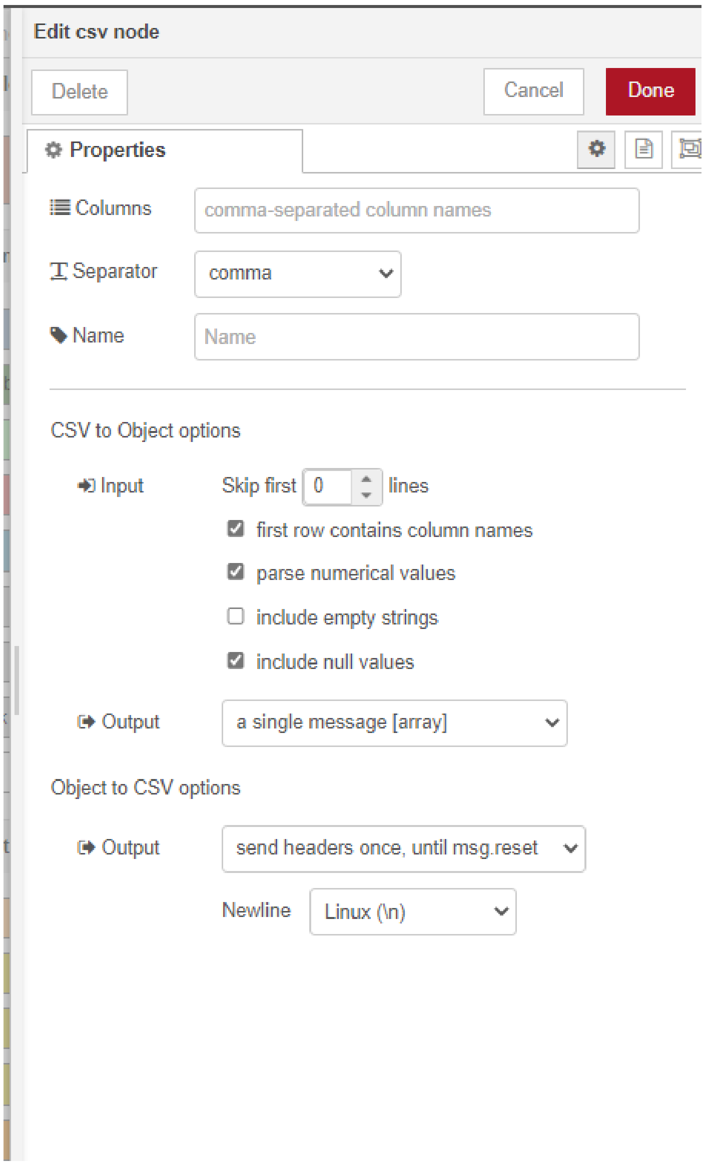


```
let plantObj


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_Y
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==106777
{kwhd_lifetime=kwhr_lifetime}


 var obj1 = {


 timestamp:usaTime,
 kwh_lifetime:kwhd_lifetime,
 iid

 }
 plantObj.push(obj1)
}
}

msg.payload = plantObj

return msg;

```

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:


Edit csv node

Delete Cancel Done

Properties

Columns: comma-separated column names

Separator: comma

Name: Name

CSV to Object options

Input: Skip first 0 lines

- [x] first row contains column names
- [x] parse numerical values
- [ ] include empty strings
- [x] include null values

Output: a single message [array]

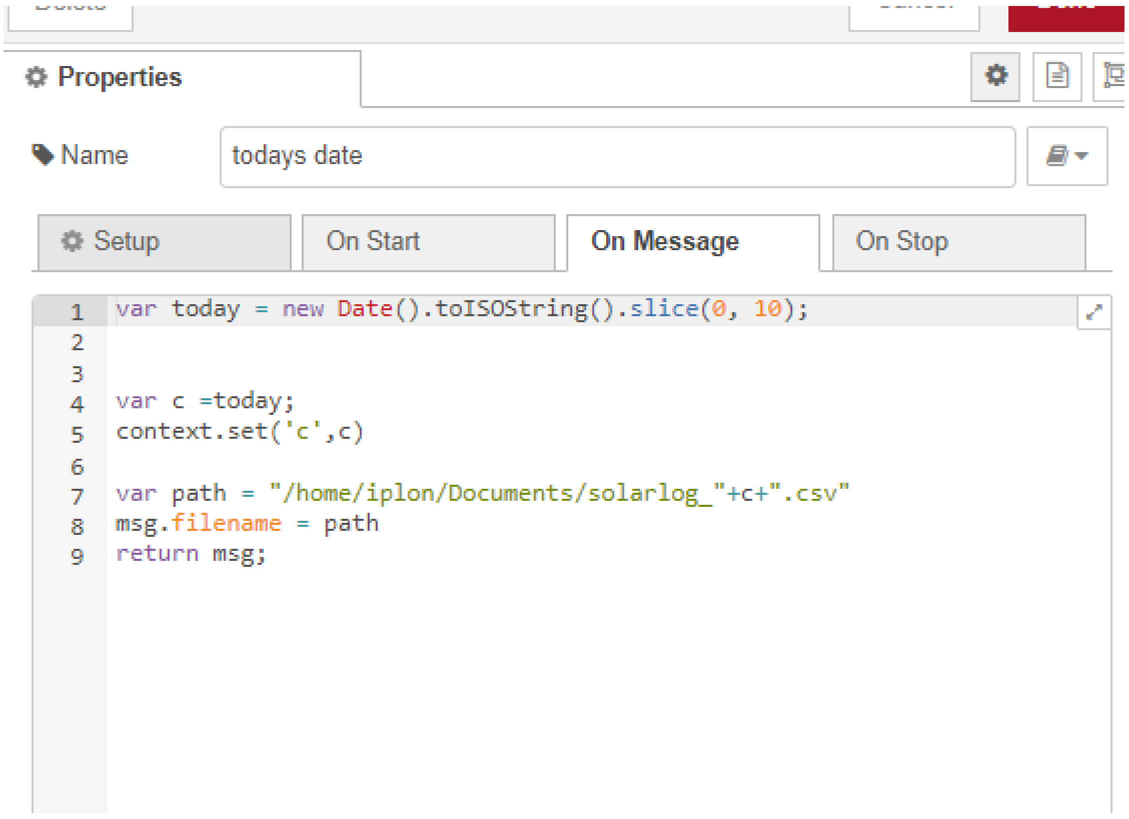
Object to CSV options

Output: send headers once, until msg.reset

Newline: Linux (\n)

Configure csv node same as in the figure

Function node for today’s date:



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


var c =today;
context.set('c',c)

var path = "/home/iplon/Documents/solarlog_"+c+".csv"
msg.filename = path
return msg;
```

Code:

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

var c =today;

context.set('c',c)

Change the file path according to the device

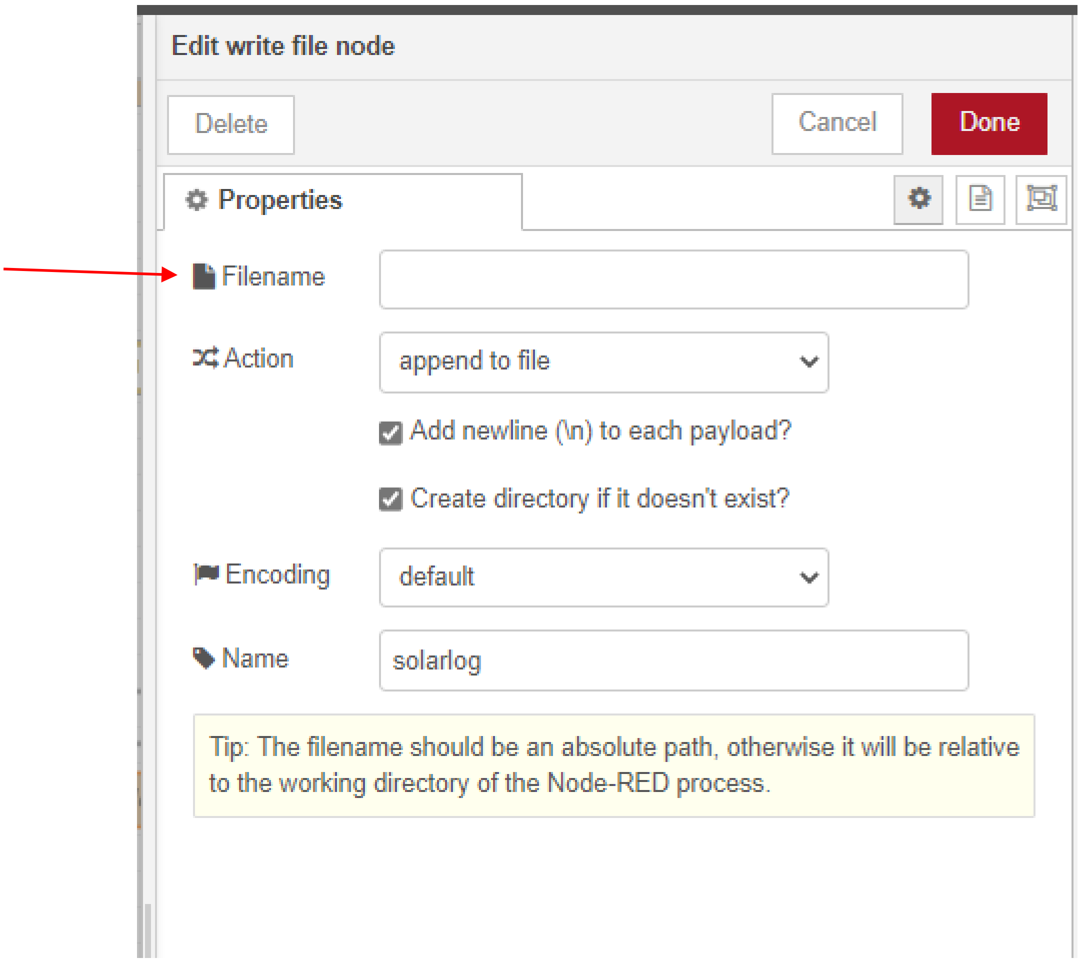
var path = "/home/iplon/Documents/solarlog_"+c+".csv"

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



Filename

Action: append to file

☑ Add newline (\n) to each payload?

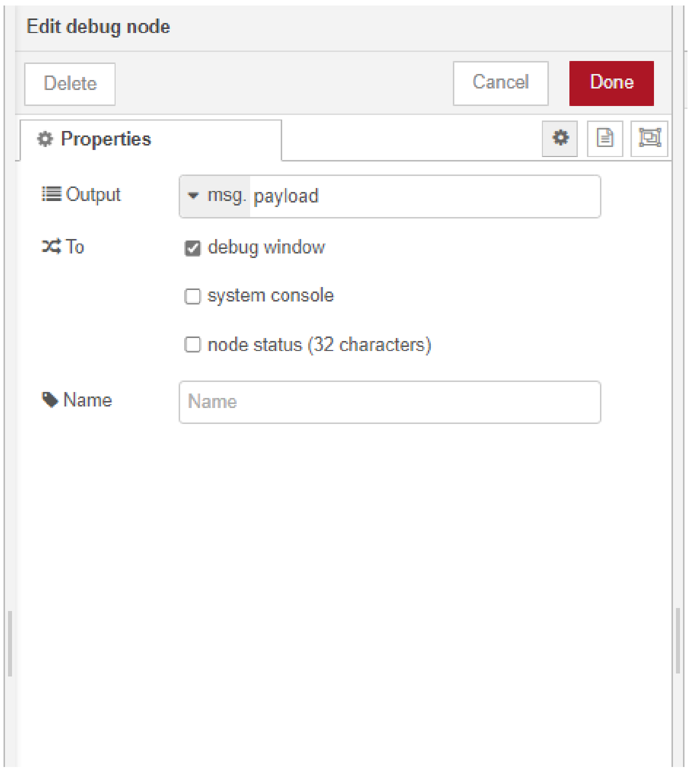
☑ Create directory if it doesn't exist?

Encoding: default

Name: solarlog

Tip: The filename should be an absolute path, otherwise it will be relative to the working directory of the Node-RED process.

Debug node:


Edit debug node

Output: msg. payload

To: [✓ debug window] system console, node status (32 characters)

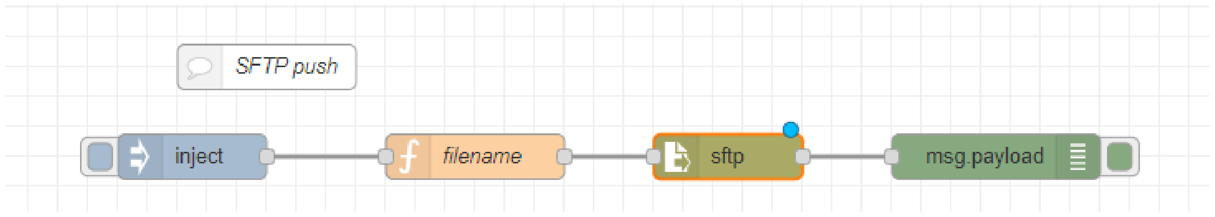
Name: Name

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

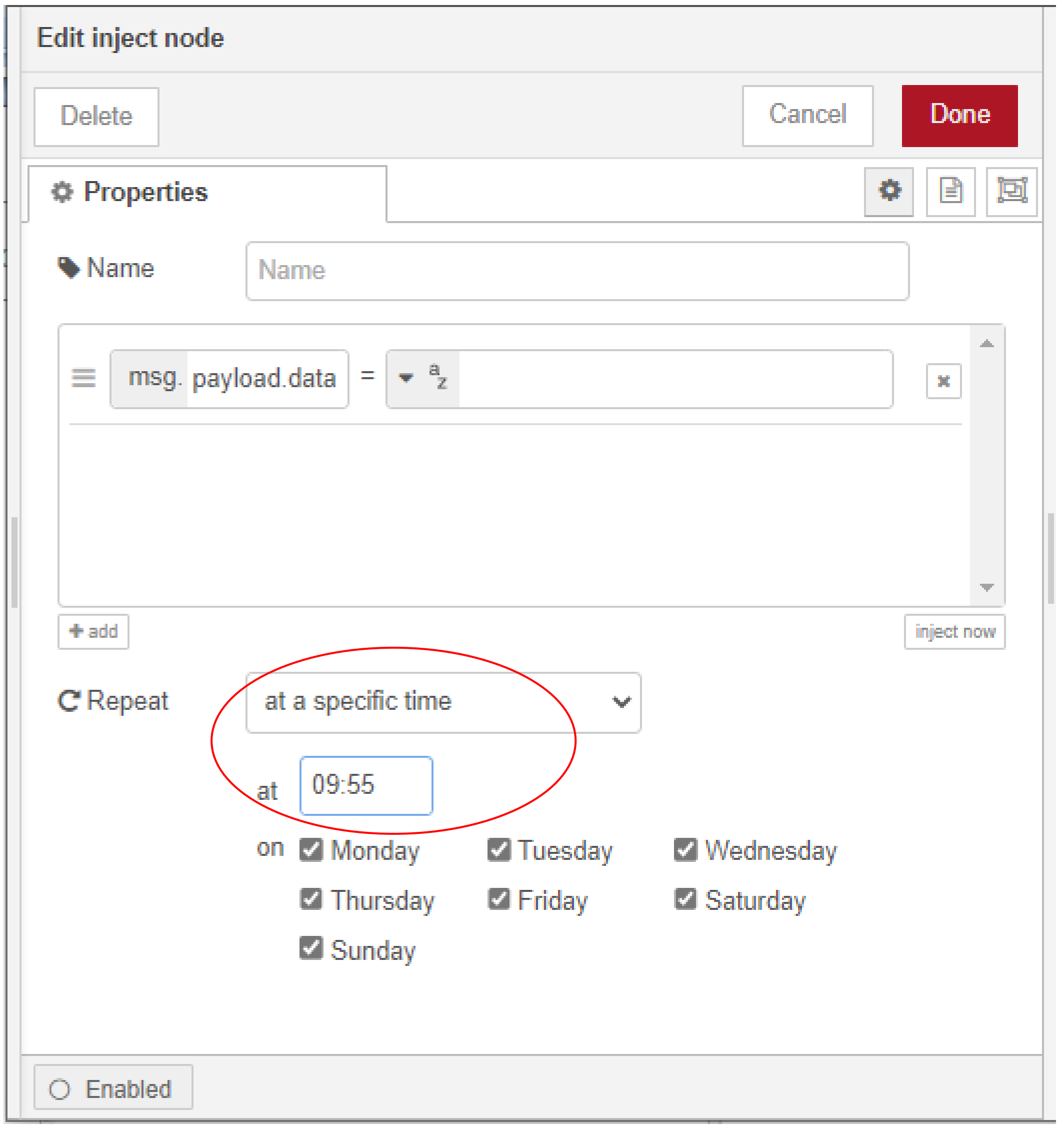
1. Click the Deploy button.With the Debug sidebar tab selected,
2. Click the Inject button to get output if there is no triggerng time set.
3. Check the destination of file what we give as path to check whether the csv file created or not.

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

NODE-RED flow for STEP 4:



INJECT NODE SETUP



Name: Name

msg. payload.data =

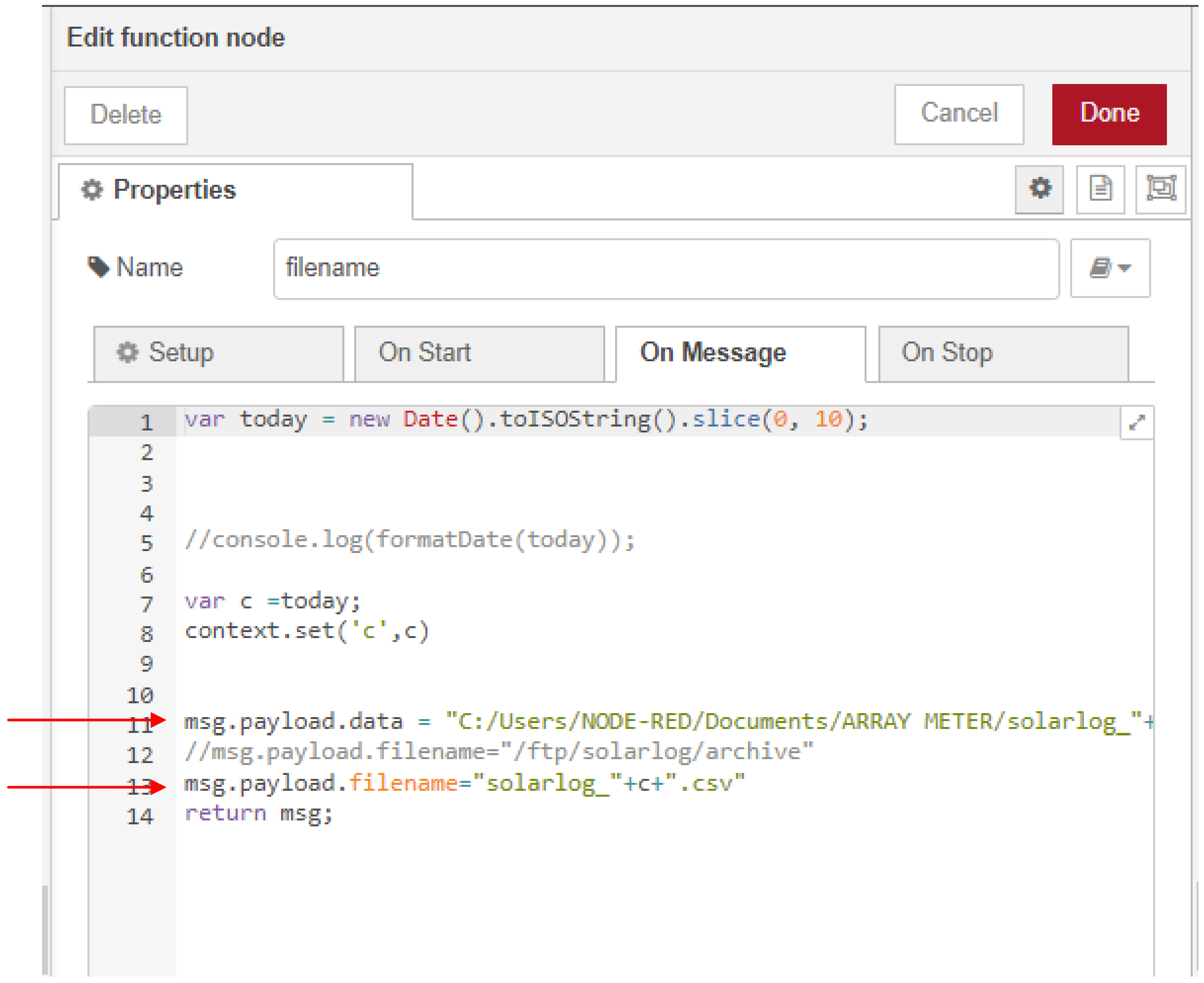
Repeat: at a specific time

at 09:55

on ✓ Monday ✓ Tuesday ✓ Wednesday ✓ Thursday ✓ Friday ✓ Saturday ✓ Sunday

FUNCTION NODE (todays date):

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



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



//console.log(formatDate(today));

var c =today;
context.set('c',c)


msg.payload.data = "C:/Users/NODE-RED/Documents/ARRAY METER/solarlog_"+
//msg.payload.filename="/ftp/solarlog/archive"
msg.payload.filename="solarlog_"+c+".csv"
return msg;
```

Code:

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

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;

Change the file path according to the device

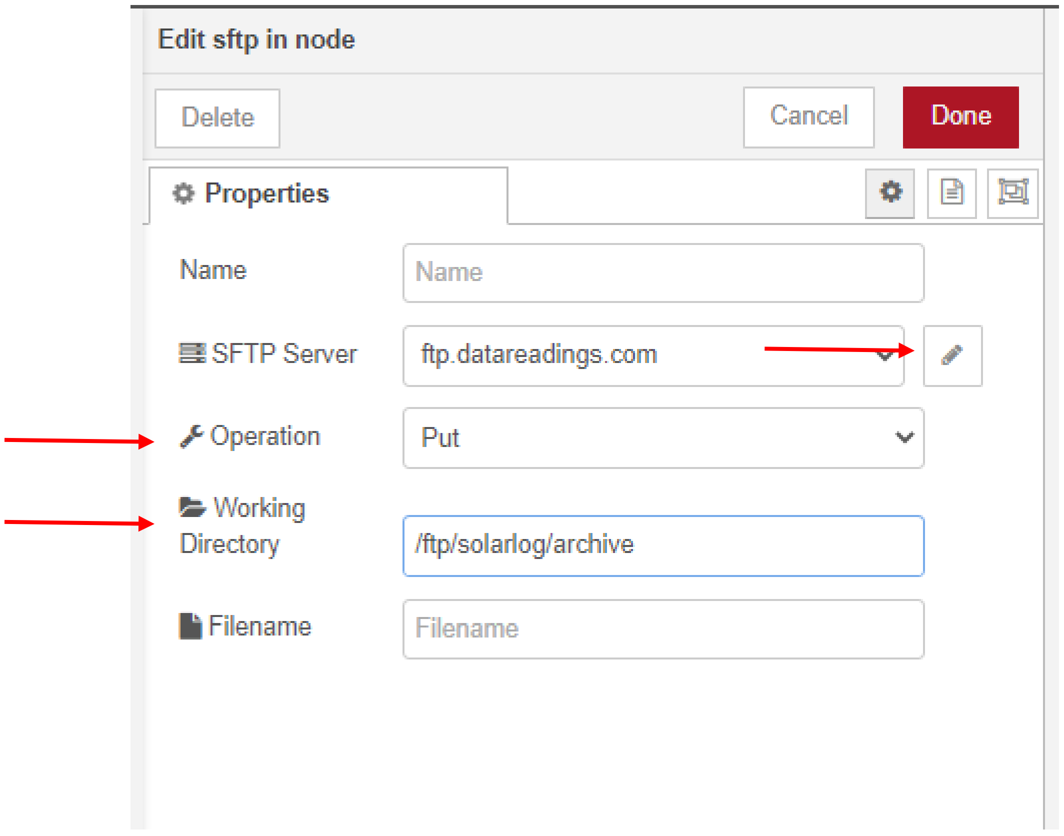
SFTP NODE:

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.



Edit sftp in node

Name: Name

SFTP Server: ftp.datareadings.com

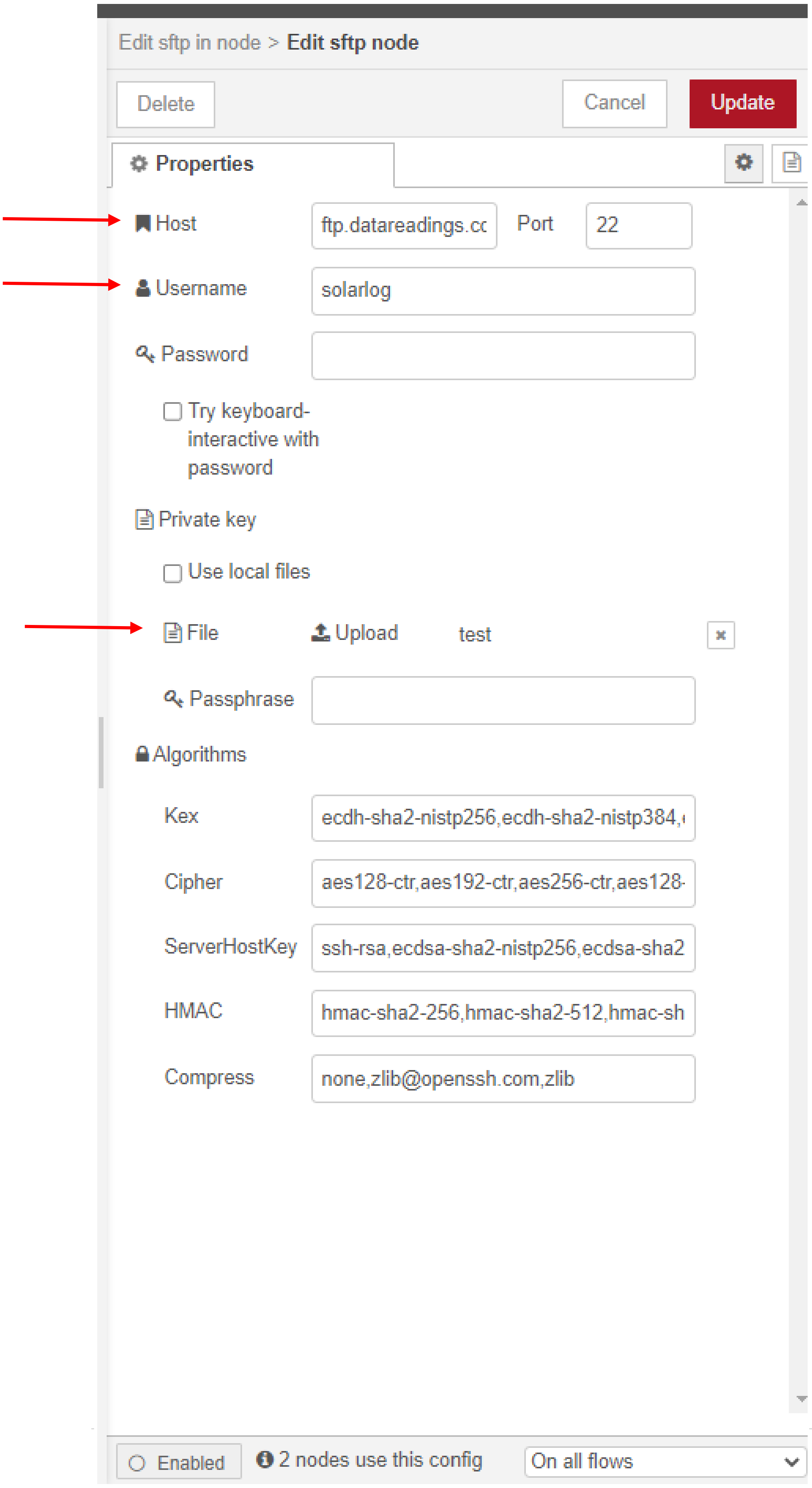
Operation: Put

Working Directory: /ftp/solarlog/archive

Filename: Filename

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



Edit sftp in node > Edit sftp node

Delete Cancel Update

Properties

Host: ftp.datareadings.cc Port: 22

Username: solarlog

Password:

☐ Try keyboard-interactive with password

Private key

☐ Use local files

File: Upload test

Passphrase:

Algorithms

Kex: ecdh-sha2-nistp256,ecdh-sha2-nistp384,

Cipher: aes128-ctr,aes192-ctr,aes256-ctr,aes128-

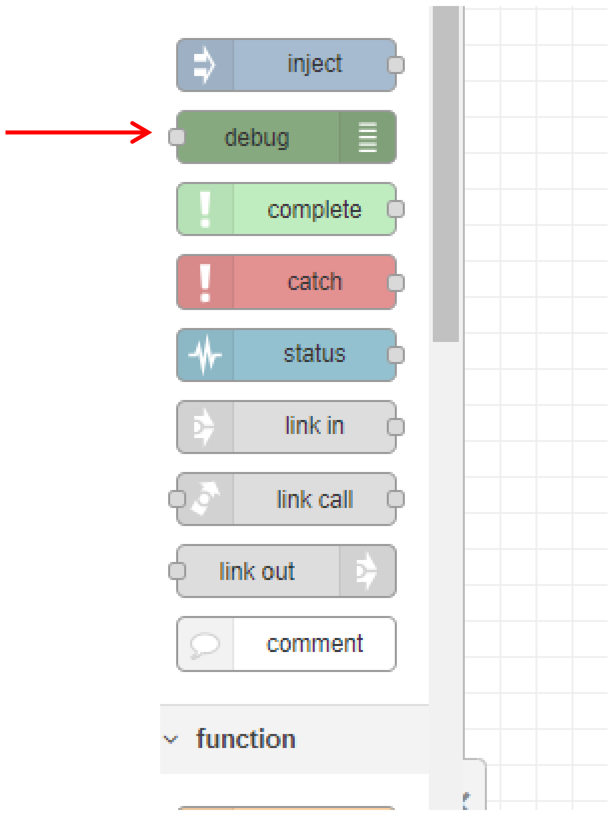
ServerHostKey: ssh-rsa,ecdsa-sha2-nistp256,ecdsa-sha2

HMAC: hmac-sha2-256,hmac-sha2-512,hmac-sh

Compress: none,zlib@openssh.com,zlib

Enabled 2 nodes use this config On all flows

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 triggerng time set.