i-UG Open Source Education for IBM i

Node-Red

LESSON 5: Engaging with the UI

1. The Node-RED UI - Dashboard

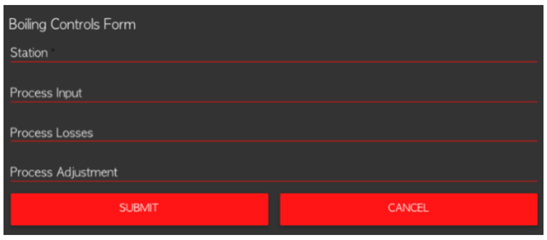
As mentioned before, the Dashboard functions as the UI (User Interface) of Node-Red, and can be used as an easier to navigate and operate interface for using your flows. It can also work as a quick visual indicator of how your flow or work process is running.

As you have seen, you can have several types of dashboard node, all of which can be found by searching the library for what you are looking for on Node-Red's site, however many come with the **node-red-dashboard** package.

This lesson will cover how we can output to the UI, rather than the previous lesson where we looked into using input from the UI. However, one of the components we will need to make this work will be to use an input Node to provide flexibility.

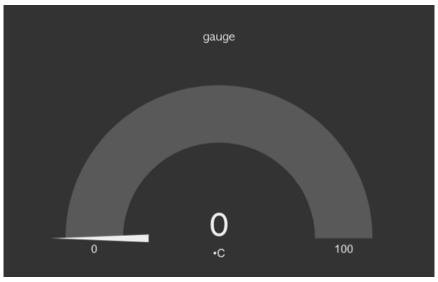
Some examples we can look into now are:

The form node can be useful for inputting data from the Dashboard, which, with the correct coding within a function node, can change and edit the parameters of your task.


Boiling Controls Form
Station
Process Input
Process Losses
Process Adjustment
SUBMIT
CANCEL

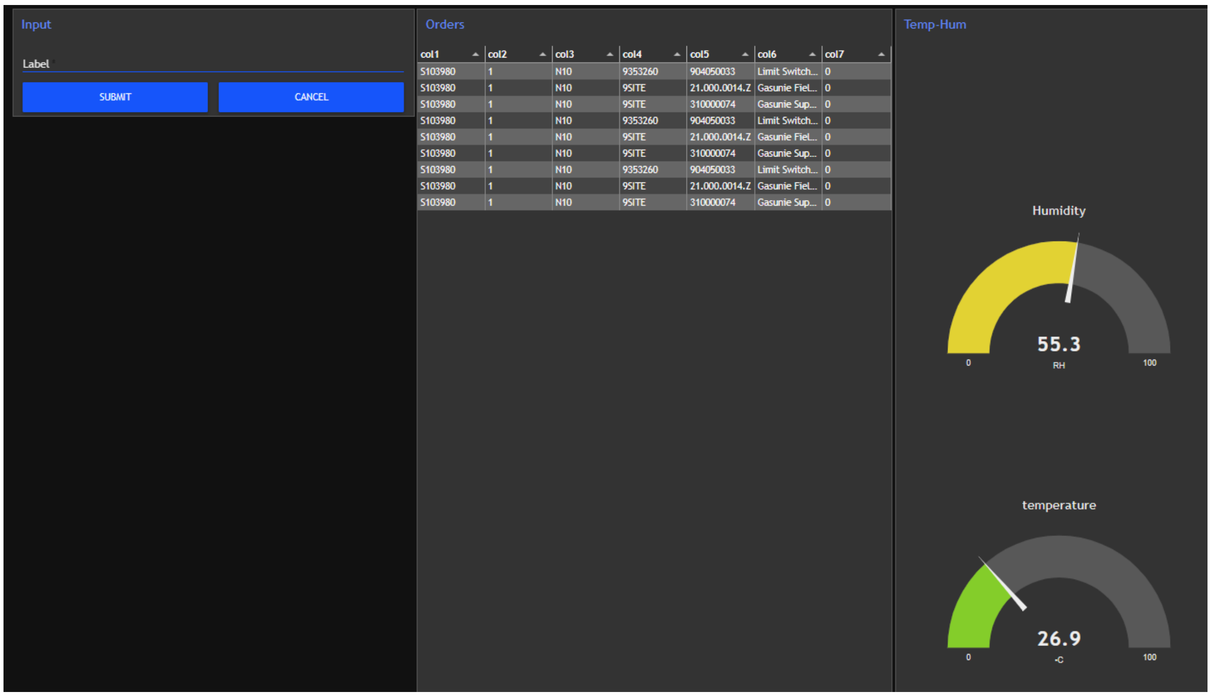
A great and easy way to take information and present it in a table on the UI

The gauge node is a great visual indicator for progress or item amounts, such as fluid levels or temperature read-outs. They can be edited to fit your exact needs.



Positioning

When you add a dashboard node such as a gauge, you can decide where in the dashboard it will appear by first deciding what group it will be in. We will use a Group called Tables for the first example. We can then position the widgets where we want them on the Group Page, although, Node-RED already does quite a good job of positioning them. It can look like this: -



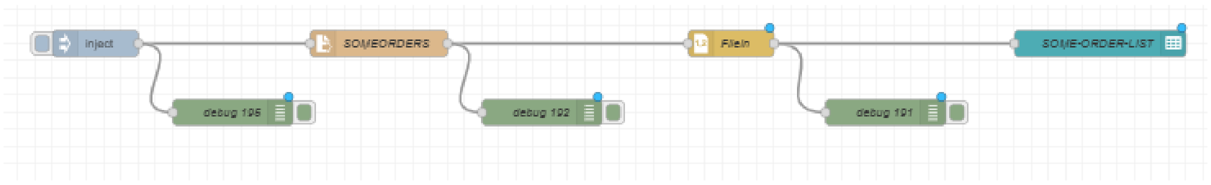
col1	col2	col3	col4	col5	col6	col7
S103980	1	N10	9353260	904050033	Limit Switch...	0
S103980	1	N10	9SITE	21.000.0014.Z	Gasunie Fiel...	0
S103980	1	N10	9SITE	310000074	Gasunie Sup...	0
S103980	1	N10	9353260	904050033	Limit Switch...	0
S103980	1	N10	9SITE	21.000.0014.Z	Gasunie Fiel...	0
S103980	1	N10	9SITE	310000074	Gasunie Sup...	0
S103980	1	N10	9353260	904050033	Limit Switch...	0
S103980	1	N10	9SITE	21.000.0014.Z	Gasunie Fiel...	0
S103980	1	N10	9SITE	310000074	Gasunie Sup...	0

2. Table

So, let's open a new Flow and drag in some Nodes. We are going to present the contents of a CSV table up on to the Node-RED UI, so we'll need more than just the Table Node.

Drag in the Table Node, but also drag in the Inject Node, the Read File Node and the CSV Node. As with all of these Flows, when you are first building them, it is prudent to have a Debug Node or two, just to see that you are actually getting what you think you are getting.

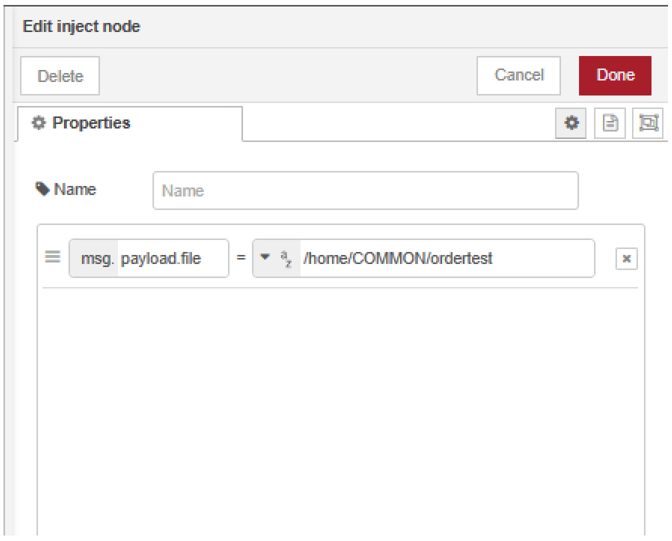
Link them together as in the Flow example below.



We will use the Inject Node to pass in the URL of the file we want to present. This will allow the Read File Node to go to the absolute address of the file and bring it in.

As the data is read, the Node outputs this as a 'JSON String', which is to say, it looks a bit like JSON but isn't. Unfortunately the Table Node is looking for nice, clean JSON or JSON Arrays, so we use the CSV Node to do this work for us and then send that JSON Array to the Table Node for display.

So, in the Inject Node, we will set it to send the URL in a subsection of the payload part of the msg, which we will call .file . There is a reason for this as we will see as we progress on this lesson. So, open the Input Node and key the following: -



Name: Name

msg. payload.file = /home/COMMON/ordertest

We have a COMMON directory on the server so we can all use the same resource.

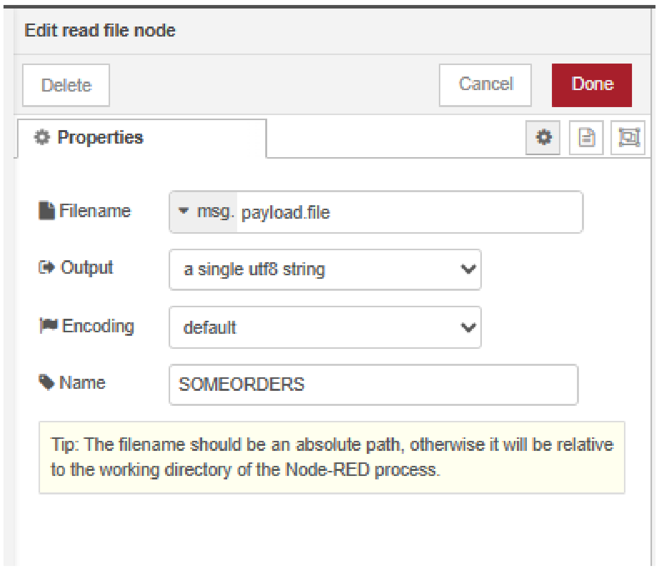
So, add .file to the payload part of the msg. (msg.payload.file)

Change the entry from a timestamp to a string and enter : -

/home/COMMON/ordertest

Now let's look at the Read File Node. Open the Node and set it up like this: -

It is happy to receive the URL from out Inject Node, although you can just key the URL in here. We are using the Inject Node for now as we want to control this better later.



Edit read file node

Filename: msg. payload.file
Output: a single utf8 string
Encoding: default
Name: SOMEORDERS

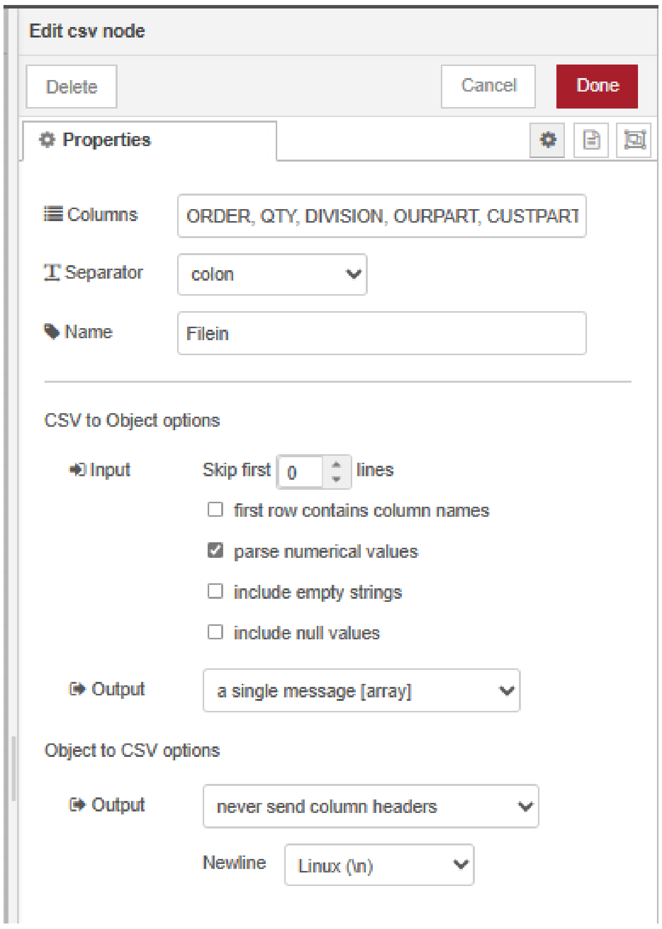
Tip: The filename should be an absolute path, otherwise it will be relative to the working directory of the Node-RED process.

Now, we will tell it to get the URL from the **payload.file** part of the **msg**. Remember, this is what we passed the URL with in the Inject Node.

We'll keep the other entries as defaults but we'll also give it a name.

As we are not satisfied with returning just a string, we will now use the CSV node to change this into a JSON object (Actually, an Array of Objects).

So, open the CSV Node and set it up like this: -



Edit csv node

Columns: ORDER, QTY, DIVISION, OURPART, CUSTPART
Separator: colon
Name: Filein

CSV to Object options

Input: Skip first 0 lines
- ☐ first row contains column names
- ☑ parse numerical values
- ☐ include empty strings
- ☐ include null values

Output: a single message [array]

Object to CSV options

Output: never send column headers
Newline: Linux (\n)

We should be diligent and give it proper column headers. You can just skip this, but column headers make it MUCH more readable. You can make up what you like but I've used : -

ORDER, QTY, DIVISION, OURPART, CUSTPART, DESC

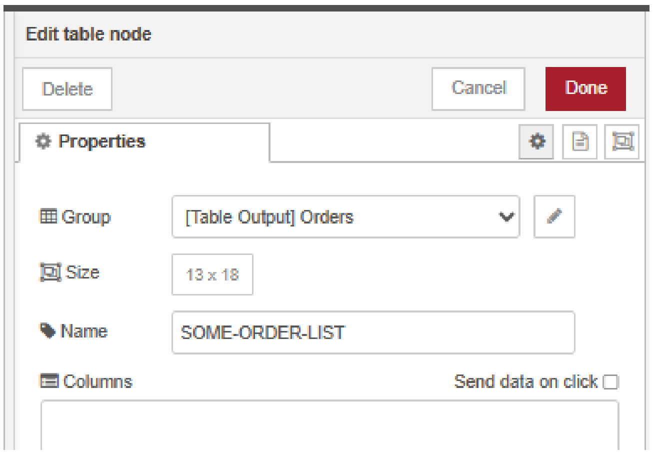
You can just make up 6 headers.

Our CSV file separates its columns using a colon, so we need to select this as our separator.

Fill the rest in like this. We will send all the information from the file as a large JSON Array (Output).

Set the rest as shown.

Using the same technique as you have done in earlier lessons, position the Table on the UI: -



Edit table node

Delete | Cancel | Done

Properties

Group: [Table Output] Orders

Size: 13 x 18

Name: SOME-ORDER-LIST

Columns | Send data on click ☐

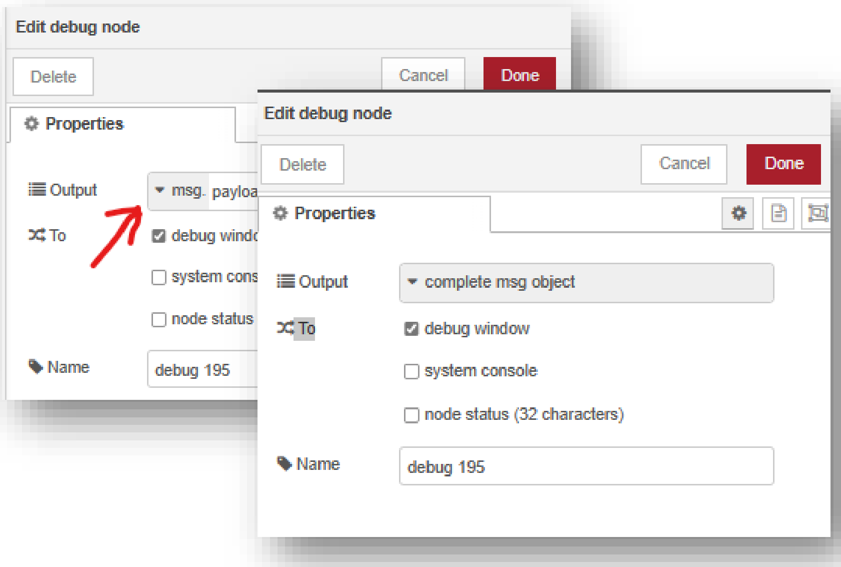
Weve set up a Tab called Table Output and this is the Orders Table. Use the Edit button to add or change this as required, also the edit button on the following panel if you are editing.

As usual, give it a name for documentation. The Node will work out the rest.

Put in the Debug Nodes like on the example above, make sure they are all wired together and then Deploy.

Tip: In the Debug Node, it defaults to just showing the payload part of the msg. You can do other debugging if you change it to show different parts of the msg (topic, file etc.) or you can ask it to show everything that it has in the msg – including the payload. This can be very useful.

Sometimes, when you just want to see what a single value is, you can get this to appear beneath the Debug Node. We will use this later.



Edit debug node

Delete | Cancel | Done

Properties

Output: msg. payload

To: ☑ debug window, ☐ system console, ☐ node status

Name: debug 195

Edit debug node

Delete | Cancel | Done

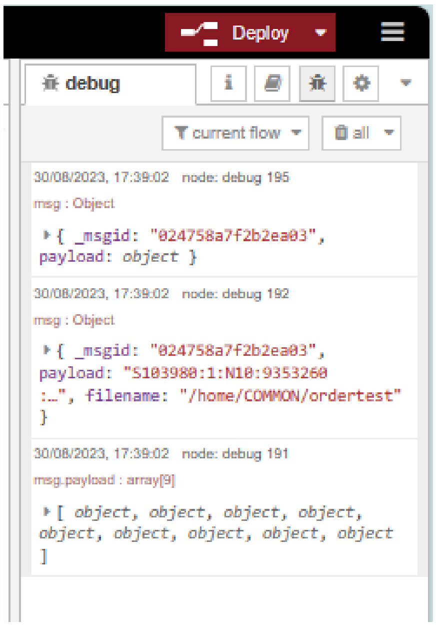
Properties

Output: complete msg object

To: ☑ debug window, ☐ system console, ☐ node status (32 characters)

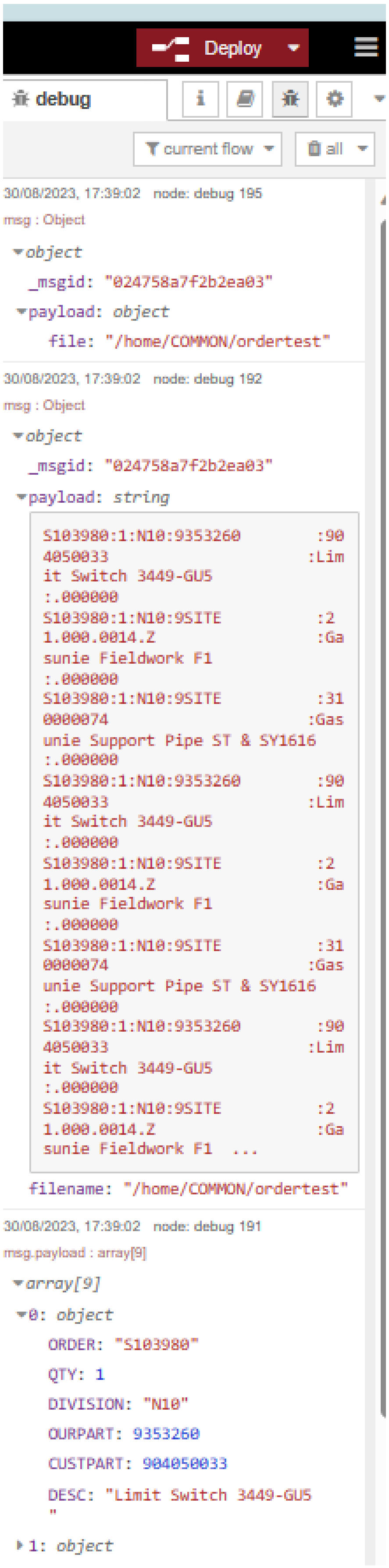
Name: debug 195

Make sure all are linked together, Deploy the Flow then hit the Inject Button. The Debug panel should look like this: -



```
30/08/2023, 17:39:02   node: debug 195
msg : Object
▸{ _msgid: "024758a7f2b2ea03",
payload: object }
30/08/2023, 17:39:02   node: debug 192
msg : Object
▸{ _msgid: "024758a7f2b2ea03",
payload: "S103980:1:N10:9353260
:…", filename: "/home/COMMON/ordertest"
}
30/08/2023, 17:39:02   node: debug 191
msg.payload : array[9]
▸[ object, object, object, object,
object, object, object, object, object
]
```

Click on the Arrows to see just what is coming through…



```
30/08/2023, 17:39:02   node: debug 195
msg : Object
▾object
  _msgid: "024758a7f2b2ea03"
 ▾payload: object
    file: "/home/COMMON/ordertest"
30/08/2023, 17:39:02   node: debug 192
msg : Object
▾object
  _msgid: "024758a7f2b2ea03"
 ▾payload: string
    S103980:1:N10:9353260         :90
    4050033                      :Lim
    it Switch 3449-GU5
    :.000000
    S103980:1:N10:9SITE           :2
    1.000.0014.Z                  :Ga
    sunie Fieldwork F1
    :.000000
    S103980:1:N10:9SITE           :31
    0000074                      :Gas
    unie Support Pipe ST & SY1616
    :.000000
    S103980:1:N10:9353260         :90
    4050033                      :Lim
    it Switch 3449-GU5
    :.000000
    S103980:1:N10:9SITE           :2
    1.000.0014.Z                  :Ga
    sunie Fieldwork F1
    :.000000
    S103980:1:N10:9SITE           :31
    0000074                      :Gas
    unie Support Pipe ST & SY1616
    :.000000
    S103980:1:N10:9353260         :90
    4050033                      :Lim
    it Switch 3449-GU5
    :.000000
    S103980:1:N10:9SITE           :2
    1.000.0014.Z                  :Ga
    sunie Fieldwork F1  ...
  filename: "/home/COMMON/ordertest"
30/08/2023, 17:39:02   node: debug 191
msg.payload : array[9]
▾array[9]
 ▾0: object
    ORDER: "S103980"
    QTY: 1
    DIVISION: "N10"
    OURPART: 9353260
    CUSTPART: 904050033
    DESC: "Limit Switch 3449-GU5
    "
 ▸1: object
```

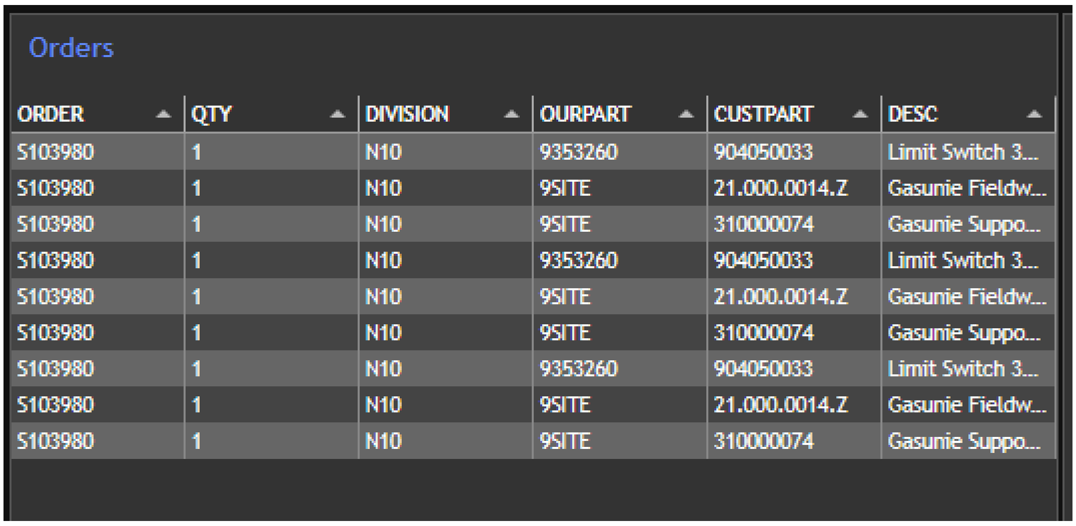
The first Debug output shows that we have /home/COMMON/ordertest as the name of The URL to get the file from

The second Debug shows the contents of the CSV file that has been returned. You can see That this is a String.

Finally, the third Debug shows the data Laid out as a JSON Array. Note the Column Headers you put in. That makes it all very readable

Finally, let's switch over to the UI (http://172.16.1.10:188nn/ui)

We should be seeing this: -



Orders

ORDER	QTY	DIVISION	OURPART	CUSTPART	DESC
S103980	1	N10	9353260	904050033	Limit Switch 3...
S103980	1	N10	9SITE	21.000.0014.Z	Gasunie Fieldw...
S103980	1	N10	9SITE	310000074	Gasunie Suppo...
S103980	1	N10	9353260	904050033	Limit Switch 3...
S103980	1	N10	9SITE	21.000.0014.Z	Gasunie Fieldw...
S103980	1	N10	9SITE	310000074	Gasunie Suppo...
S103980	1	N10	9353260	904050033	Limit Switch 3...
S103980	1	N10	9SITE	21.000.0014.Z	Gasunie Fieldw...
S103980	1	N10	9SITE	310000074	Gasunie Suppo...

Now, if you change the Inject Node to send - home/COMMON/ordertest2 , then you will see different data appearing in the UI as this is a different CSV file.

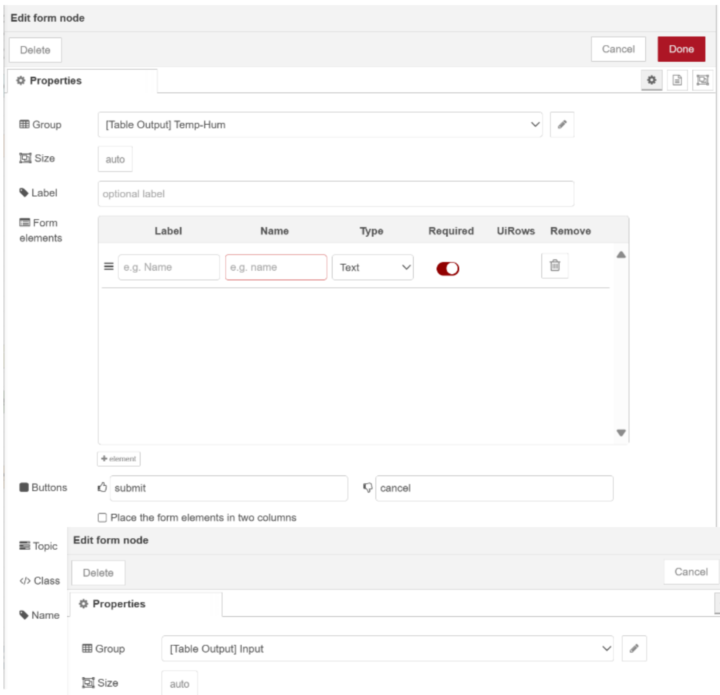
Try it. Remember to re-Deploy after you make the change.

Pretty cool!

3. Forms

So, now let’s make the ability to change the name of the CSV file by using a Form Node. The Form Node allows you to create an input area on the UI so we can type in the URL and change the data without having to Deploy the Flow every time a change is made.

Drag in a Form Node and open it: -



Edit form node

Delete | Cancel | Done

Properties

Group: [Table Output] Temp-Hum
Size: auto
Label: optional label
Form elements:

Label	Name	Type	Required	UiRows	Remove
e.g. Name	e.g. name	Text	●		🗑

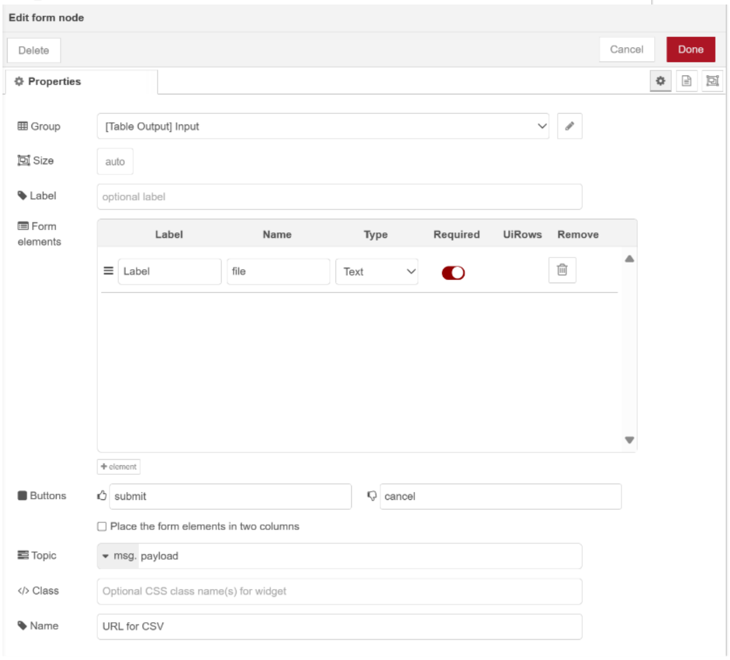
+ element

Buttons: submit | cancel

☐ Place the form elements in two columns

Topic
Class
Name

For this exercise, we wil just have one field, or element, but you can have many elements passed at the same time by clicking the + element box. Each element can be described as types such as Date, Time, Number, e-mail etc. and the Node will ensure that the format entered by the user is appropriate. We will just use one element described as Text.



Edit form node

Delete | Cancel | Done

Properties

Group: [Table Output] Input
Size: auto
Label: optional label
Form elements:

Label	Name	Type	Required	UiRows	Remove
Label	file	Text	●		🗑

+ element

Buttons: submit | cancel

☐ Place the form elements in two columns

Topic: msg. payload
Class: Optional CSS class name(s) for widget
Name: URL for CSV

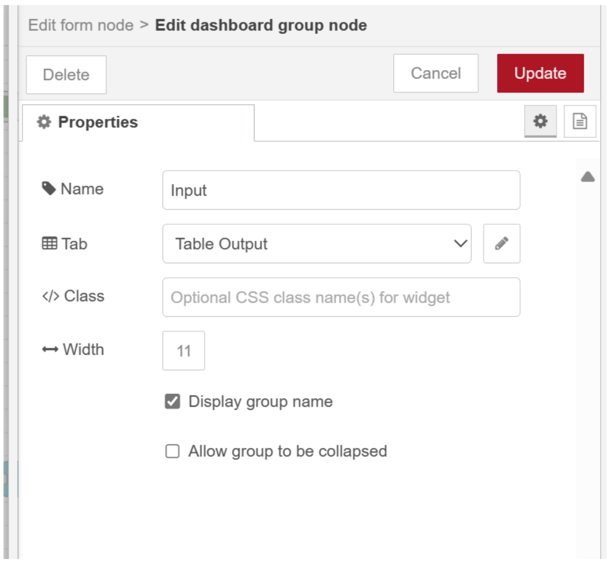
Set your first Form Node like this.

Don’t forget to give it a name.

We’ll just leave the size as Auto for now, but in future, you may need to specify a size in order to make your UI look more tidy.

The Buttons we’ll use will be titled Submit and Cancel. You can change the text here in future.

To make sure that it appears on the correct tab on the UI (next to the Table we created), Click on the Group edit button: -


Edit form node > **Edit dashboard group node**

Delete | Cancel | Update

Properties

Name	Input
Tab	Table Output
Class	Optional CSS class name(s) for widget
Width	11

☑ Display group name

☐ Allow group to be collapsed

Select the Table Output from the drop-down list.

Give this a Name – Input

Leave the rest at defaults.

Link this node to the Read File Node and Deploy.

Now, go to the UI and enter in the first CSV URL - home/COMMON/ordertest and Submit it.

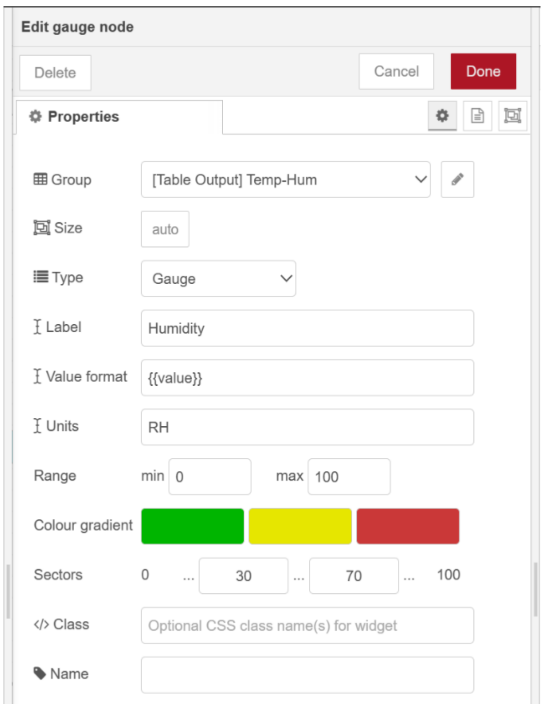
This will pass the URL to the File Read Node and the data will refresh in the Table. This is the first table we looked at when we passed the URL from the Inject Node. We are now bypassing that node.

Now change it to the second URL - home/COMMON/ordertest2 and Submit. Watch the Table change.

4. Gauges Stage 1. Manual Control

Showing Gauges on the UI can be very powerful, and these can be a very powerful indicator of certain conditions.

So, let's start by dragging in a Gauge Node and Open it: -


Edit gauge node

Delete Cancel Done

Properties

Group	[Table Output] Temp-Hum
Size	auto
Type	Gauge
Label	Humidity
Value format	{{value}}
Units	RH
Range	min 0 max 100
Colour gradient	🟩 🟨 🟥
Sectors	0 ... 30 ... 70 ... 100
Class	Optional CSS class name(s) for widget
Name	

As with the previous UI Nodes, we will place this on the same page as the Table and Form. If you like, you can set up a new Tab and a new Group, so that the Gauges appear on a different Tab in the UI.

You can select the type of Gauge you would like, but as we are measuring Temp. and Humidity, they are not really appropriate. You can experiment with the different Gauges.

We'll set the Units for this one to RH and set a range between 1 and 100.

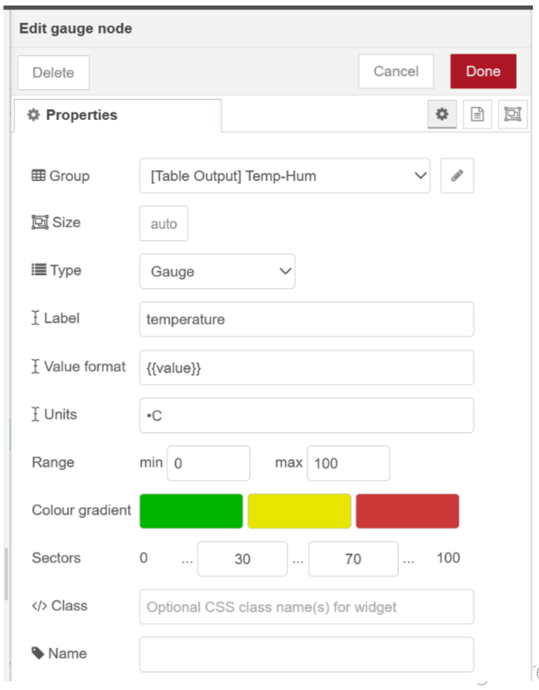
A really useful feature is that we can change the colour based on the actual humidity (or temp. etc.).

Set the Sectors as we have here.

We'll leave the name blank as the default is to use the Label. Humidity would seem to be a good Name. You could override this and use 'Humidity Gauge' to make it more readable.

We need a new Gauge node that we can set up for temperature. You can simply drag in a new Gauge node, or you can copy the current Gauge Node and modify it slightly. Let's try this second option. Just Copy and Paste as usual (Cntrl. 'C' and Cntrl. 'V' will work just fine).

Open this Node and set it like this:


Edit gauge node

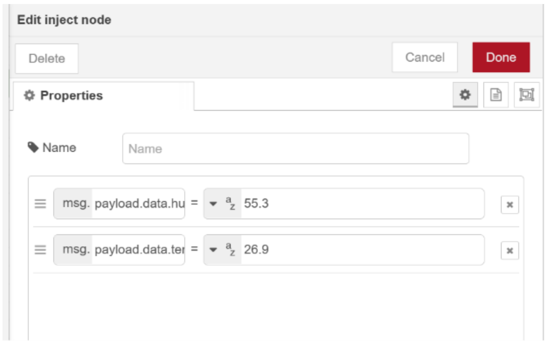
Delete Cancel Done

Properties

Group	[Table Output] Temp-Hum
Size	auto
Type	Gauge
Label	temperature
Value format	{{value}}
Units	•C
Range	min 0 max 100
Colour gradient	🟩 🟨 🟥
Sectors	0 ... 30 ... 70 ... 100
Class	Optional CSS class name(s) for widget
Name	

The Gauge is looking for a string or a number In the msg.payload. There are many ways in which we can do this so lets explore just one of these ways.

We need to drag in a new Inject Node and send two elements in the payload, one for humidity and one for temperature: -


Edit inject node

Delete · Cancel · Done

Properties

Name: Name

msg. payload.data.hu = 55.3

msg. payload.data.te = 26.9

Set the first msg to payload.data.humidity

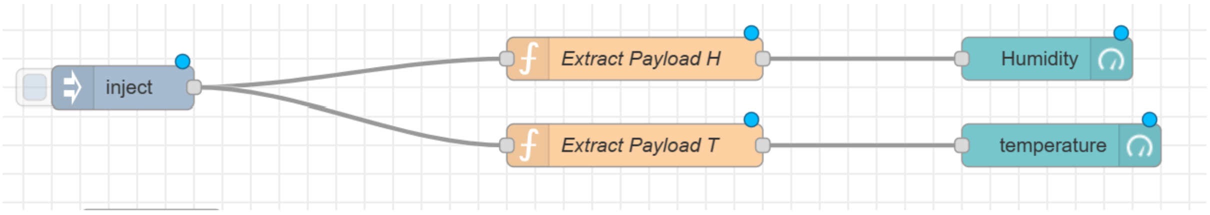
Set the second msg to payload.data.temperature_c

It doesn't really matter what you call these, but you can see the granularity that you can go to.

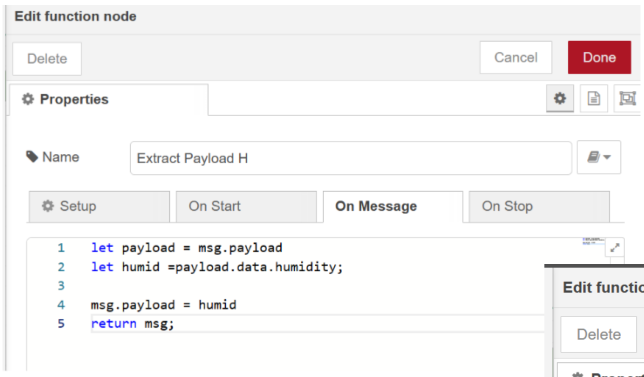
For now, set the msg values to strings and use the numbers we have shown.

Now, drag in two Function Nodes. We will split the Inject into two streams in the Flow and service one Gauge from one Function Node. The Flow should look something like this: -

N.B. If you look at the GitHub answers, please limit this section to Gauges Stage 1 group Only



Set the Function Nodes to contain a small amount of code as below: -


Edit function node

Delete · Cancel · Done

Properties

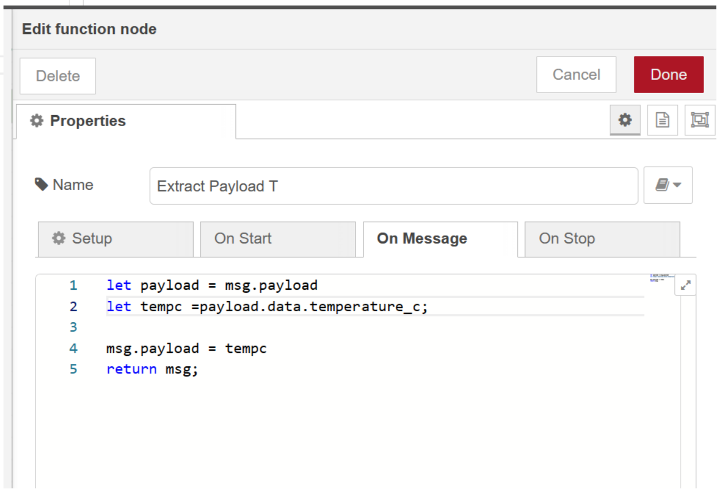
Name: Extract Payload H

Setup · On Start · On Message · On Stop

```
let payload = msg.payload
let humid =payload.data.humidity;

msg.payload = humid
return msg;
```

The Humidity Function Node


Edit function node

Delete · Cancel · Done

Properties

Name: Extract Payload T

Setup · On Start · On Message · On Stop

```
let payload = msg.payload
let tempc =payload.data.temperature_c;

msg.payload = tempc
return msg;
```

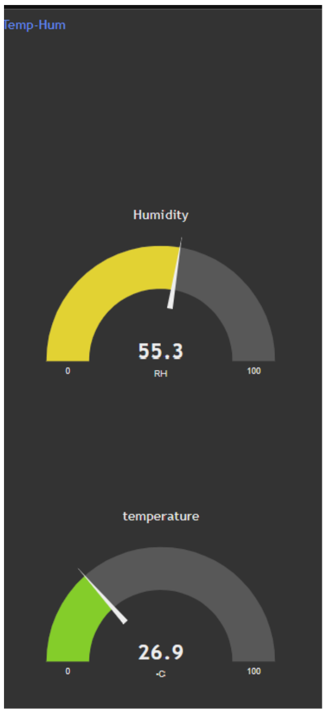
The Temperature Humidity Node

As you can see, we are extracting the section of the Inject Node in each Function Node and sending on the value to the correct Gauge.

A little long winded but you can see how the Function Node is a valuable intermediary between Nodes and – with the correct code – it can achieve just about everything.

Make sure they are all linked together an then Deploy and hit the Inject Node.

In the UI Dashboard, the gauges will look something like this: -



Just by way of experiment, let's look again at the UI and the gauges. You will note that the Humidity almost certainly is shown in Yellow and the temperature gauge should be in the Green.

To test the colour changing on the gauges, go back to the original Inject Node in the Flow and change the values to be sent. If you put the values between 30 and 70 it should go Yellow, if the values are set to over 70, the gauge should present in Red. Very useful visual warning indicator.

Whilst you are at it, if you click on the colour blocks in the Gauge Nodes, you will see that you can change the colour scheme to whatever you like. Try it. Remember to Deploy at each change

5. Gauges - Stage 2. Real Time Feed

So, whilst we can see that we can send values to a Gauge Node and it will display prettily, what we really need to do is to link this to a Real-time sensor device that is actually measuring Temperature and Humidity and sending Real-Time date over to us.

Well, you guessed it, we will be using an MQTT Broker to do this, just like with the lights. We'll see that in a later lesson.

And with that, the lesson is concluded. If you didn't manage to make one of the flows work then you can copy one of the flows from our GitHub repository here, and see where you went wrong. These functions, although they seem simple, are the backbone of many useful applications of Node-Red for major and small businesses.

https://github.com/JoshRyanEOG/i-UG_Education_Node-Red