i-UG Open Source Education for IBM i

Node-Red

LESSON 6: Speaking to the World

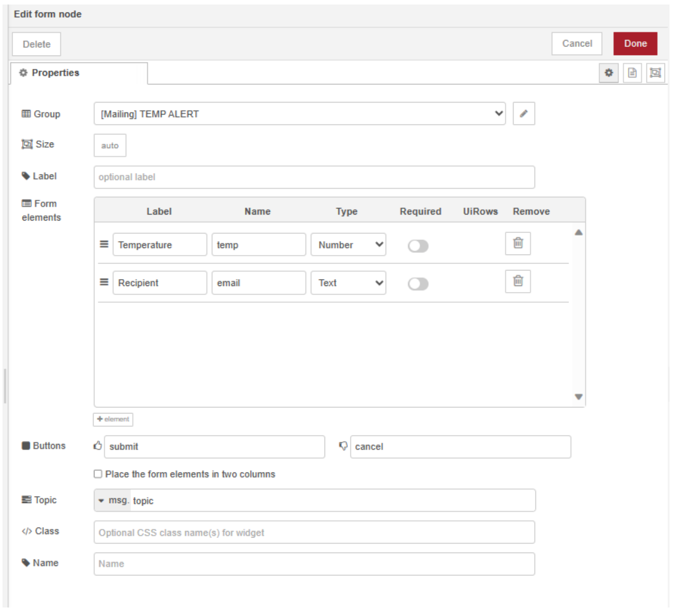
1. e-mail

There are any number of reasons why we may want to advise people of an event as and when it happens. A good example might be of, say, a temperature sensor picking up a reading that is in excess of a threshold that we have set.

We will cover the whole management of such a situation later in the course as this needs to include the management of the Alert and how we reset the Alert, but for now, we'll just concentrate on how we actually get the e-mail out of the system to the specified recipient.

We'll start with adding a Form Node, this will let us input the email we want the warning to be sent to, and choose a temperature threshold, all within the Node-Red UI Dashboard.

Drag in a Form Node:



Edit form node

Group	[Mailing] TEMP ALERT
Size	auto
Label	optional label
Form elements	

Label	Name	Type	Required	UiRows	Remove
Temperature	temp	Number			
Recipient	email	Text			

+ element

Buttons	submit / cancel
	☐ Place the form elements in two columns
Topic	msg. topic
Class	Optional CSS class name(s) for widget
Name	Name

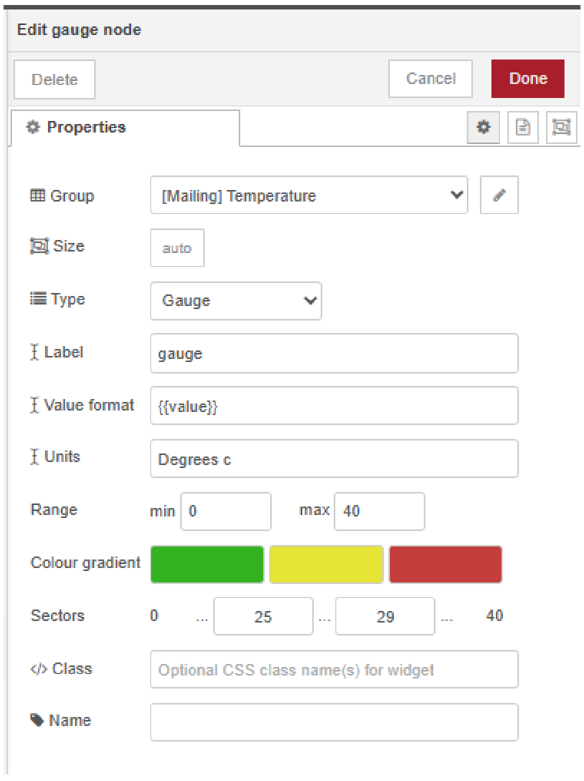
We are going to create two Form elements within the node, one for the temperature - set this to a number type, and one for the recipient email - set this to a text type. Use the *+element* button to add each one.

Unclick the Required button on the Temperature, as we will not need to enforce that here.

Put the form on a tab called 'Mailing' with a name of 'TEMP ALERT'.

The values the user enters in this form will be passed in the msg.payload as subsets, using the name column. So, msg.payload.temp and msg.payload.email

Drag in a Gauge Node and open it: -


Edit gauge node

Delete | Cancel | Done

Properties

Group	[Mailing] Temperature
Size	auto
Type	Gauge
Label	gauge
Value format	{{value}}
Units	Degrees c
Range	min 0 max 40
Colour gradient	🟩 🟨 🟥
Sectors	0 ... 25 ... 29 ... 40
Class	Optional CSS class name(s) for widget
Name	

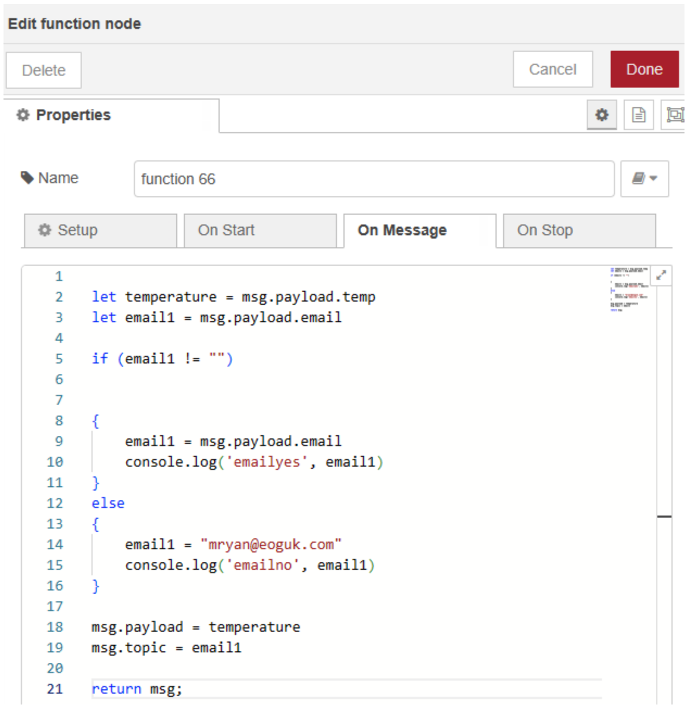
Put this on the same Tab as the Form (Mailing) and name it 'Temperature', as in the example here.

We will set the Range to be between 0 and 40 (Degrees C).

Set the Units to 'Degrees c'

Now, we can colour the Gauge based on the value coming in, so in the Sectors we will put '25' and '30'. That means that for all values between 0 and 25, we will see the Gauge in Green. For values between 25 and 30 we will see it in Yellow and for values over 30 we will see it in Red.

Now drag in a Function Node, the purpose of which is to take the information from the Form Node entries and make it useable to both the Gauge Node and the Email out Node.


Edit function node

Delete | Cancel | Done

Properties

Name: function 66

Setup | On Start | On Message | On Stop

```

let temperature = msg.payload.temp
let email1 = msg.payload.email

if (email1 != "")


{
    email1 = msg.payload.email
    console.log('emailyes', email1)
}
else
{
    email1 = "mryan@eoguk.com"
    console.log('emailno', email1)
}

msg.payload = temperature
msg.topic = email1

return msg;
```

Take a good look at the code.

We are taking the information that we get in from the msg.payload (msg.payload.temp and msg.payload.email) and putting the temperature into one variable and the email recipient entered into another variable called email1.

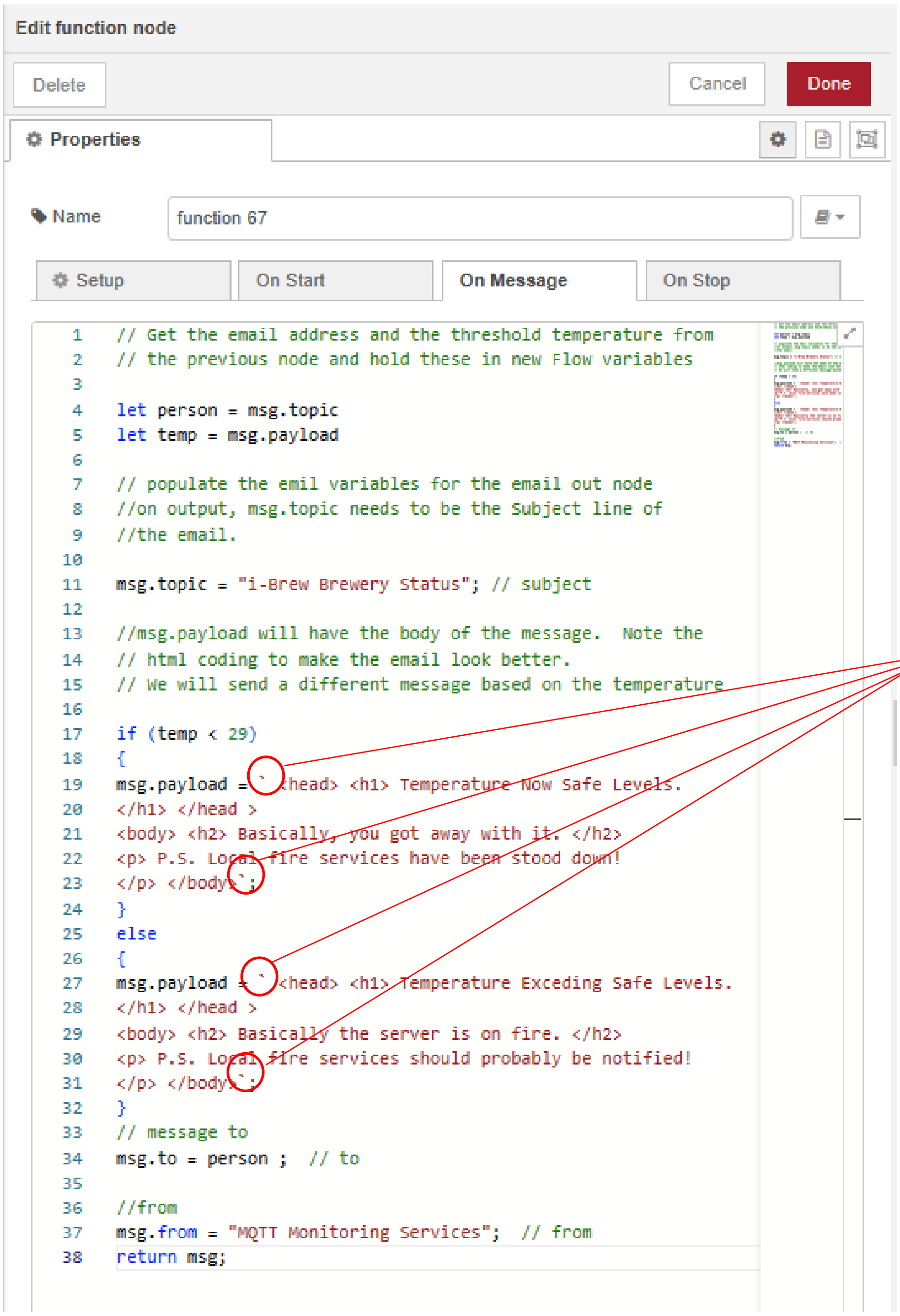
If no email is actually entered (Say, just the temperature), we will apply a default email. N.B. in your code, please use an email that you have access to!

If an email IS entered in the form, we will use that email address and make sure that we keep that email for the life of this Flow deployment. So, if no new email is entered, we will just skip over the replacement of email1.

This is the code you will need to put in (Just as an aside, there are many ways in which this can be achieved using code, and different programmers will have different views. Suffice to say, this will work. If you think of a better way, please go ahead… as long as it works).

In order to build up an actual email message that we can use, we need another Function Node. Pull this in now and we'll set it up: -

This is fairly explanatory. You can see that we are constructing an email – From, To, Subject, Body – and the next Node, the Email out Node), will send this email.



```
// Get the email address and the threshold temperature from
// the previous node and hold these in new Flow variables

let person = msg.topic
let temp = msg.payload

// populate the emil variables for the email out node
//on output, msg.topic needs to be the Subject line of
//the email.

msg.topic = "i-Brew Brewery Status"; // subject

//msg.payload will have the body of the message.  Note the
// html coding to make the email look better.
// We will send a different message based on the temperature

if (temp < 29)
{
msg.payload = `<head> <h1> Temperature Now Safe Levels.
</h1> </head >
<body> <h2> Basically, you got away with it. </h2>
<p> P.S. Local fire services have been stood down!
</p> </body>`;
}
else
{
msg.payload = `<head> <h1> Temperature Exceding Safe Levels.
</h1> </head >
<body> <h2> Basically the server is on fire. </h2>
<p> P.S. Local fire services should probably be notified!
</p> </body>`;
}
// message to
msg.to = person ;  // to

//from
msg.from = "MQTT Monitoring Services";  // from
return msg;
```

The HTML codes (<body>, <h2> etc.) just give the email text a little structure, it is not strictly necessary.

If you have any skill with HTML, then you can play around with the structure of the message and the content. Feel free.

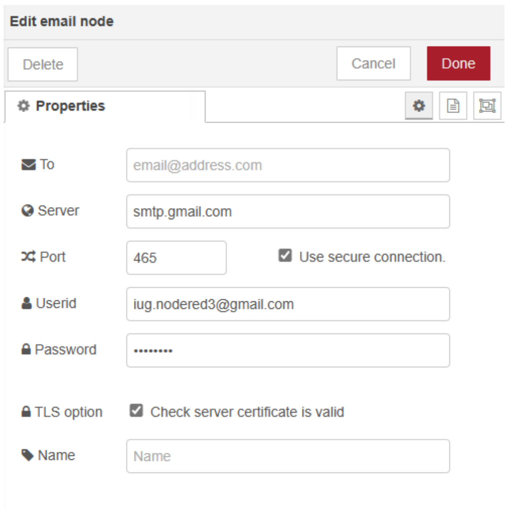
Otherwise, just copy this verbatim and it will work.

Just remember these are `BACKTICKS`

Now drag in an Email out Node and open it: -

The contents of this Node are very specific! Please copy them as shown in this example.

The password will be supplied to you.



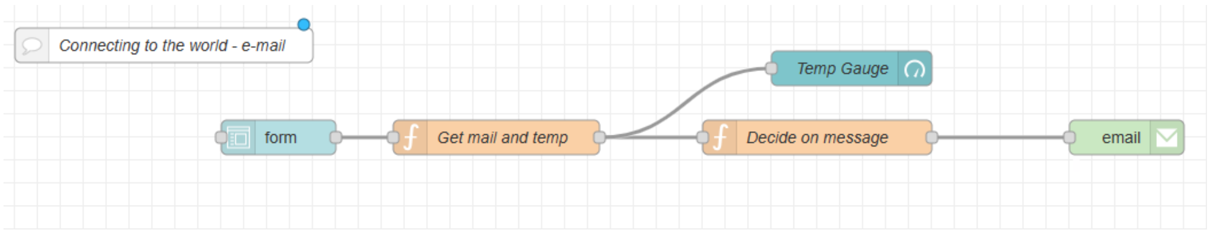
Edit email node

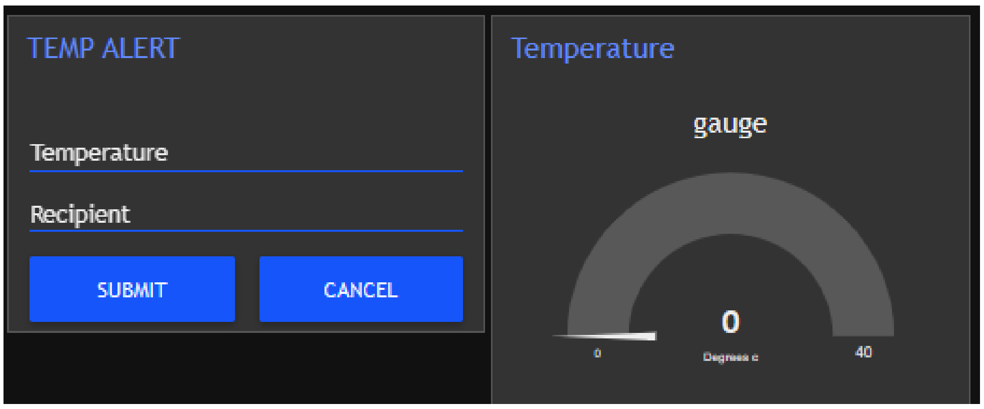
Delete | Cancel | Done

Properties

To	email@address.com
Server	smtp.gmail.com
Port	465 ☑ Use secure connection.
Userid	iug.nodered3@gmail.com
Password	••••••••
TLS option	☑ Check server certificate is valid
Name	Name

Wire these all together and Deploy the Flow. It should look something like this: -





And your dashboard should resemble this.

So now, go to the Form and enter a temperature of, say, 20℃ and your own e-mail address.

After you hit the Submit button, you should find that the Gauge is showing 20 degrees c and depicted in Yellow. You should also receive an e-mail with an advisory stating that the temperature levels are OK.

After a minute or two, just enter a temperature value of 30℃. Don't enter your e-mail address again.. just submit the temperature. You should receive another e-mail telling you that there is a Problem, and the graph should show 30 degrees c and should be in Red.

Try out a few variations. Change the temperature, change the e-mail message, change the threshold….

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