

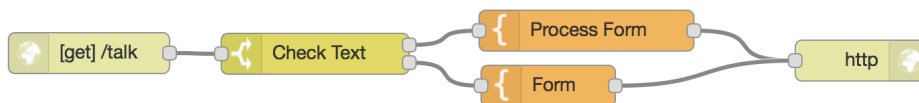
# Text to Speech in Node-RED

## Hands-On Lab

JeanCarl Bisson | [jbisson@us.ibm.com](mailto:jbisson@us.ibm.com) | [@dothewww](#)



Add a web endpoint to convert text to audio  
(see *Add Text to Speech in Node-RED*)



Create a webpage to input text and play audio  
(see *Creating an Interactive Web UI*)



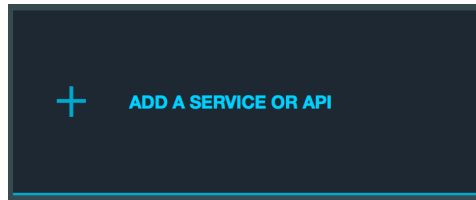
A digital copy of this lab and code snippets can be found at:  
<http://ibm.biz/node-red-text-to-speech>



# Add Text to Speech Service in IBM Bluemix

The Text to Speech node in Node-RED requires API credentials. In this section, we will create and bind the Text to Speech IBM Watson service to the Node-RED application.

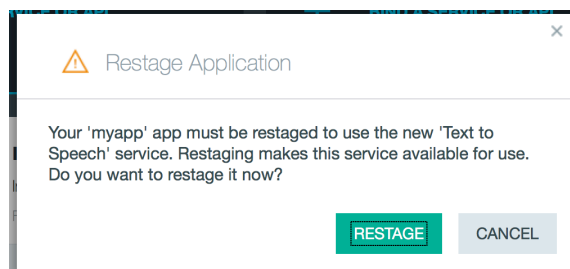
1. Go to the application overview for the Node-RED application in the IBM Bluemix dashboard and click **Add a service or API**.



2. Click the **Text to Speech** node under the Watson section. You can use the default values, or customize the name. Click on **Create**.




3. IBM Bluemix will prompt to restage the application. Click on **Restage**. The application will restart and include the new service credentials in the environment.

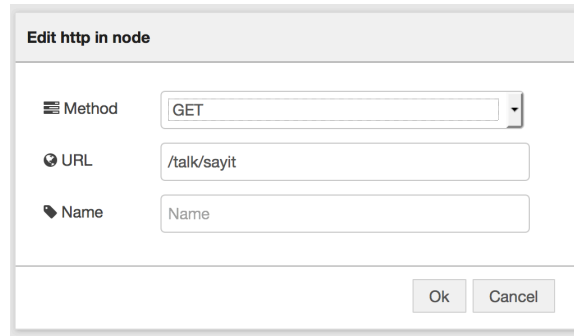


4. When the application has finished restaging, open the Node-RED Flow Editor. If Node-RED is already open, refresh the page.

# Add Text to Speech in Node-RED

In this section, we will use the IBM Watson Text to Speech service to produce a .wav audio file from input text through a simple web endpoint generated using a Node-RED flow.

1. Add a  node as shown below to collect the incoming speech request:




**Edit http in node**

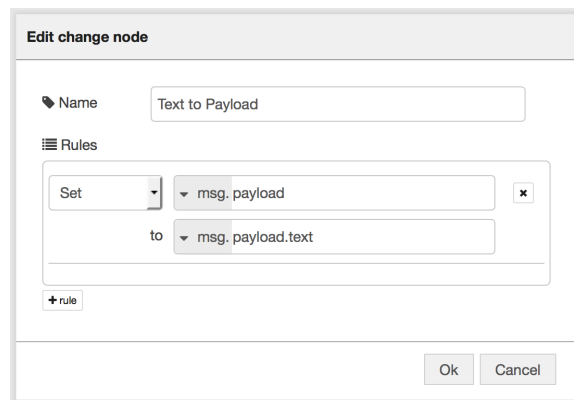
Method: GET

URL: /talk/sayit

Name: Name

Ok Cancel

2. Add a  node as shown below to extract the query parameter msg.payload.text and set it as the msg.payload. When invoked with query parameters such as ?text=Hello, the text will be placed into the msg.payload object.



**Edit change node**


Name: Text to Payload

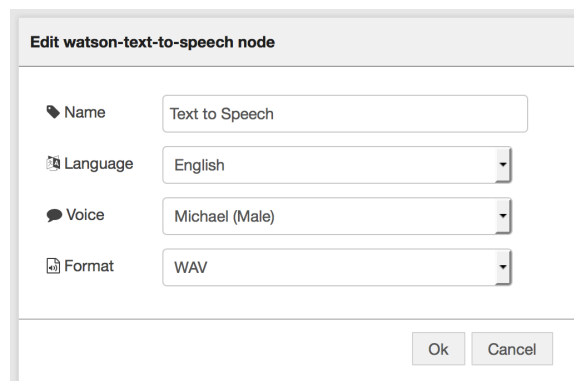
Rules:

Set msg.payload to msg.payload.text

+ rule

Ok Cancel

3. Now add a  node as shown below. This node will generate the binary wav stream content and put it in the msg.speech property.



**Edit watson-text-to-speech node**

Name: Text to Speech


Language: English

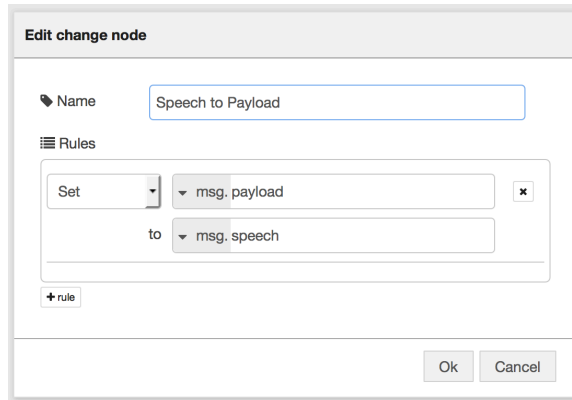
Voice: Michael (Male)

Format: WAV


Ok Cancel

You can change the language, the voice, or the format of the audio that is returned.

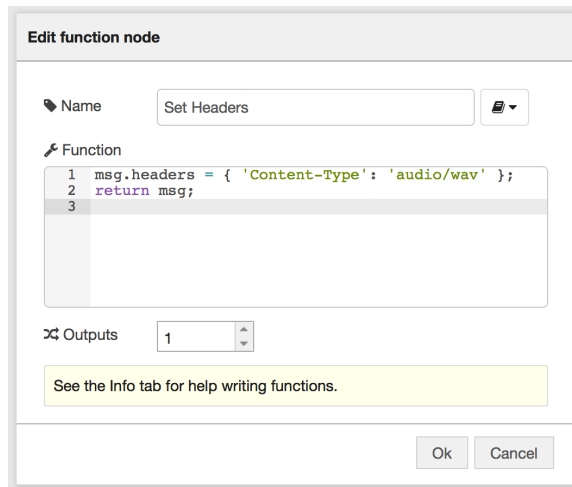
4. Add another  **change** node as shown below to extract the msg.speech and place it in msg.payload.



The 'Edit change node' dialog shows the name 'Speech to Payload'. Under the 'Rules' section, there is a 'Set' dropdown, a field containing 'msg.payload', and a 'to' dropdown containing 'msg.speech'. There is a '+ rule' button at the bottom left and 'Ok' and 'Cancel' buttons at the bottom right.

5. Add a  **function** node as shown below to add HTTP headers to the response:

```
msg.headers = { 'Content-Type': 'audio/wav' };  
return msg;
```




The 'Edit function node' dialog shows the name 'Set Headers'. The 'Function' field contains the code: 


```
1 msg.headers = { 'Content-Type': 'audio/wav' };  
2 return msg;  
3
```

. There is an 'Outputs' dropdown set to '1' and a yellow note that says 'See the Info tab for help writing functions.' at the bottom. 'Ok' and 'Cancel' buttons are at the bottom right.

We set the HTTP response headers by setting the msg.headers to the Content Type of audio/wav. This is required in order to let browsers know that this is an audio file and not HTML.

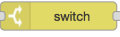
6. Finally, add a  **http response** node. This node will simply return what's in msg.payload to the HTTP response. The completed flow should look like the flow shown below:

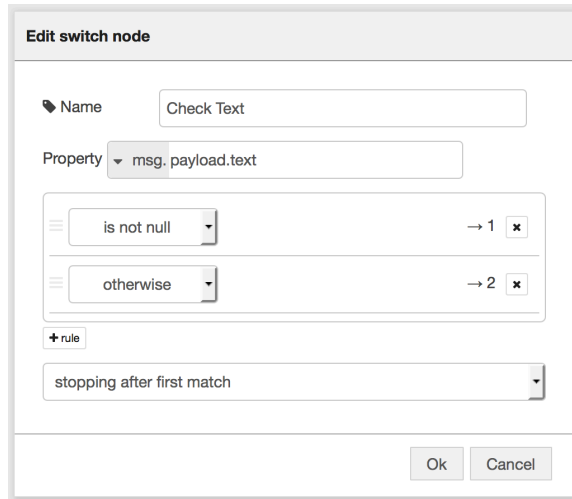


7. Click on  **Deploy** to save the changes. Test the flow by opening the application URL, appended with /talk/sayit?text=Hello as shown below:


`https://_____.mybluemix.net/talk/sayit?text=Hello`

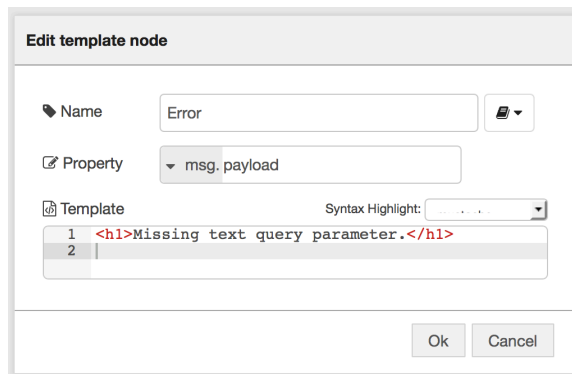
This should prompt you to save a file. Depending on how your browser is configured, it may download the audio file or play it within the browser. Either way, play it and you should hear the text.

8. This flow has a caveat, however. The flow will fail when the text query parameter is not set. Add a  switch node as shown below to check if the text query parameter is present.



The 'Edit switch node' dialog shows a node named 'Check Text'. The property is set to 'msg.payload.text'. There are two rules: 'is not null' pointing to output 1, and 'otherwise' pointing to output 2. The 'stopping after first match' option is selected.

9. You'll notice that adding the second otherwise rule has created a second output handle for the switch node. Add a  template node with the error message shown below.

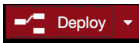


The 'Edit template node' dialog shows a node named 'Error'. The property is 'msg.payload'. The template text is: 

```
1 <h1>Missing text query parameter.</h1>
2
```

10. Connect the nodes together as shown below:



11. Click on  to save the changes. Test the flow by opening a browser and going to the application URL, appended with the following endpoints:

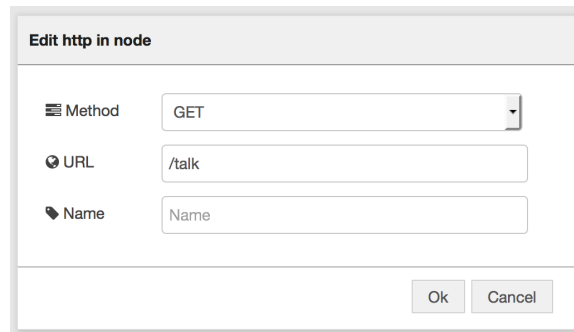
`https://_____.mybluemix.net/talk/sayit?text=Hello`  
`https://_____.mybluemix.net/talk/sayit`

(plays the audio with Hello)  
(displays error message)

# Creating an Interactive Web UI

In this section we will create a simple webpage that displays an input textbox for the user to enter text. When the form is submitted, the text will be played via audio.

1. Add a  node with the following:



**Edit http in node**

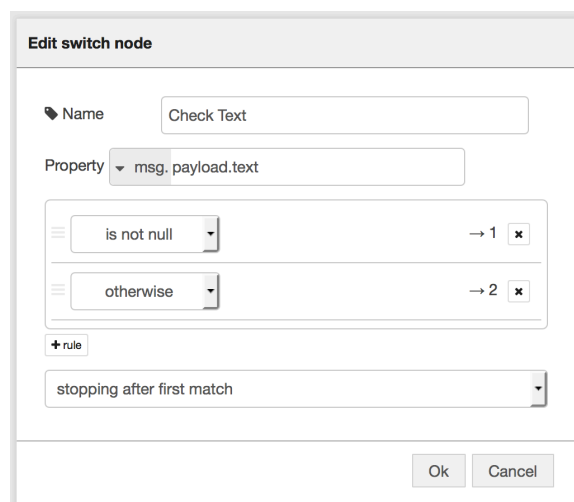
Method: GET

URL: /talk

Name: Name

Ok Cancel

2. Add a  node to branch the flow depending whether text input is submitted.



**Edit switch node**

Name: Check Text

Property: msg.payload.text

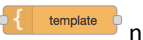
Rule 1: is not null → 1

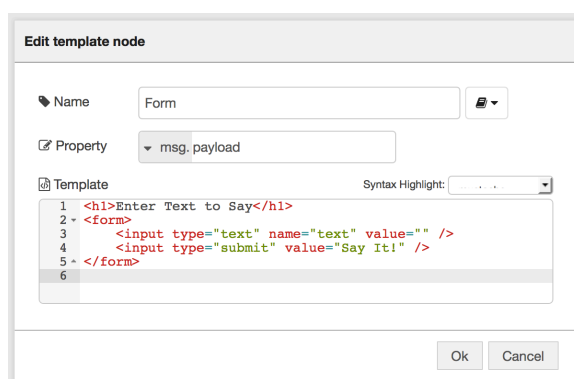
Rule 2: otherwise → 2

stopping after first match

Ok Cancel

When the form is submitted and the text parameter is present, proceed with the first flow. Otherwise, the blank form (flow #2) will be used.

3. Add a  node with the HTML shown below:



**Edit template node**

Name: Form

Property: msg.payload


Template:

```
1 <h1>Enter Text to Say</h1>
2 <form>
3   <input type="text" name="text" value="" />
4   <input type="submit" value="Say It!" />
5 </form>
6
```

Ok Cancel



Get the code:  
[ibm.biz/Bd4vtc](https://ibm.biz/Bd4vtc)

4. Add a  template node with the HTML shown below:

Edit template node

Name:

Property:

Template

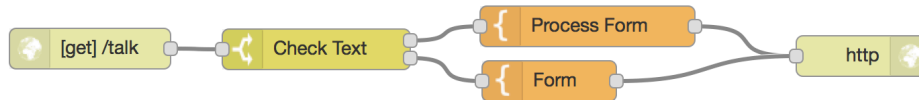
```
1 <h1>You Want To Say</h1>
2 <p><q>{{payload.text}}</q></p>
3
4 <h1>Hear It</h1>
5 <audio controls autoplay>
6   <source src="/talk/sayit?text={{payload.text}}" type="audio/wav">
7   Your browser does not support the audio element.
8 </audio>
9
10 <h1>Enter Text to Say</h1>
11 <form>
12   <input type="text" name="text" value="{{payload.text}}" />
13   <input type="submit" value="Say It!" />
14 </form>
15
```


Ok Cancel



Get the code:  
[ibm.biz/Bd4vtx](https://ibm.biz/Bd4vtx)

5. Add a  and connect the nodes together as follows:



6. Click  Deploy to save the changes. Test the flow by opening the application URL in the browser, appended with /talk.

[https://\\_\\_\\_\\_\\_.mybluemix.net/talk](https://_____.mybluemix.net/talk)