

# Language Translation in Node-RED

## Hands-On Lab



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

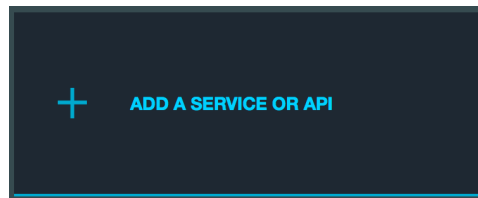


# Add Language Translation in IBM Bluemix

The IBM Watson Language Translation service enables you to translate text from one language to another. These languages are supported:

- The News domain - targeted at news articles and transcripts, it translates English to and from French, Spanish, Portuguese or Arabic.
- The Conversational domain - targeted at conversational colloquialisms, it translates English to and from French, Spanish, Portuguese or Arabic.
- The Patent domain - targeted at technical and legal terminology, it translates Spanish, Portuguese, Chinese, or Korean to English.

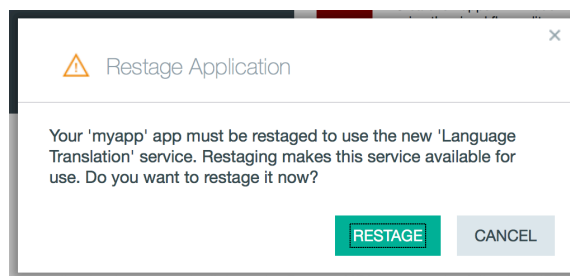
1. Go to the application overview for your Node-RED application in the IBM Bluemix dashboard and click **Add a service or API**.



2. Click the **Language Translation** node under the Watson section. Click on **Create**.




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

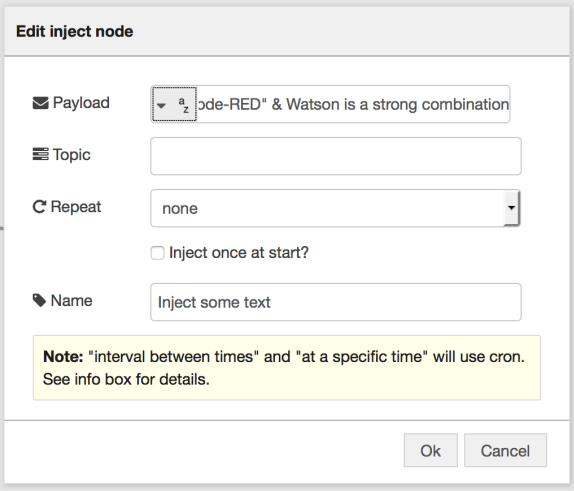


3. When the application has finished restaging, open the Node-RED Flow Editor. If you already have Node-RED open, refresh the page.

# Add Language Translation in Node-RED

In this section, we will add a Language Translation node and translate text input into French. In this example some random text (in English in this case) is injected, translated to French and the result outputted to the Debug tab.

1. Add an  node as shown below (you can use any text for this):




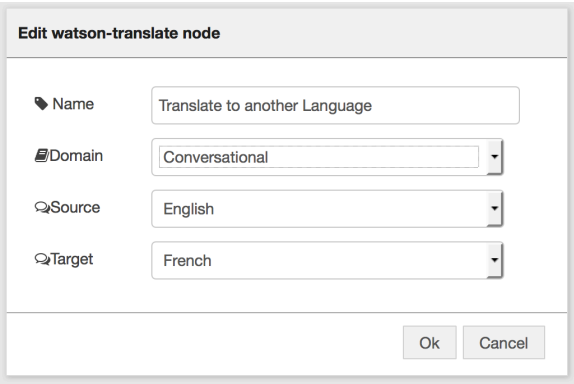
The 'Edit inject node' dialog box contains the following fields:

- Payload:** A text input field containing "Node-RED" & Watson is a strong combination".
- Topic:** An empty text input field.
- Repeat:** A dropdown menu set to "none".
- Inject once at start?:** An unchecked checkbox.
- Name:** A text input field containing "Inject some text".

A note at the bottom states: "Note: 'interval between times' and 'at a specific time' will use cron. See info box for details."

Buttons: Ok, Cancel

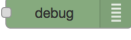
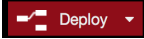
2. Add a  node as shown below. The text in this case is English so select English. Based on your source choose the right domain: News or Conversational.

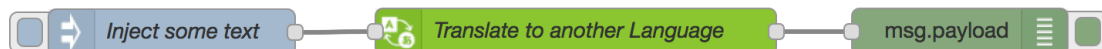


The 'Edit watson-translate node' dialog box contains the following fields:

- Name:** A text input field containing "Translate to another Language".
- Domain:** A dropdown menu set to "Conversational".
- Source:** A dropdown menu set to "English".
- Target:** A dropdown menu set to "French".

Buttons: Ok, Cancel

3. Add a  node. The translated text will be returned in message.payload and displayed in the Debug tab.
4. Connect the nodes together as shown below and click on the red  Deploy in the upper-right of the screen.



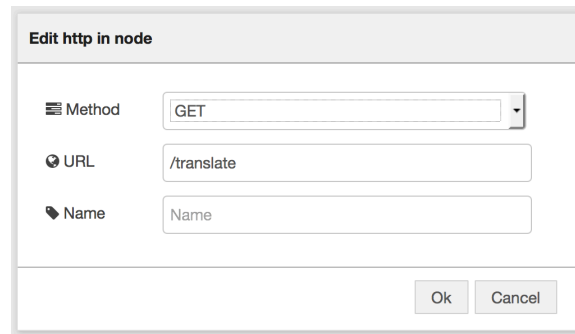
- Click on the tab on the left side of the inject node to start the flow. The output from the debug node will show up in the debug tab in the right-hand pane.



# Creating an Interactive Web UI

In this section, we will create a simple webpage that accepts user input, translates it into French, and then displays it on the webpage.

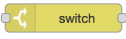
1. Add a  as shown below:

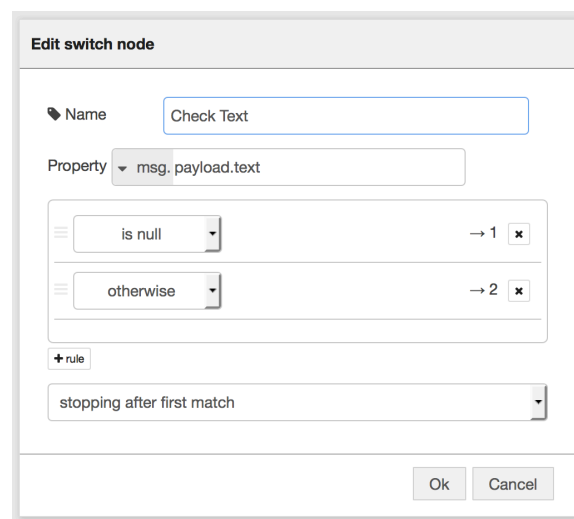


The 'Edit http in node' dialog box shows the following configuration:

- Method: GET
- URL: /translate
- Name: Name

Buttons: Ok, Cancel

2. Add a  node as shown below. This node checks whether the text parameter is passed in.




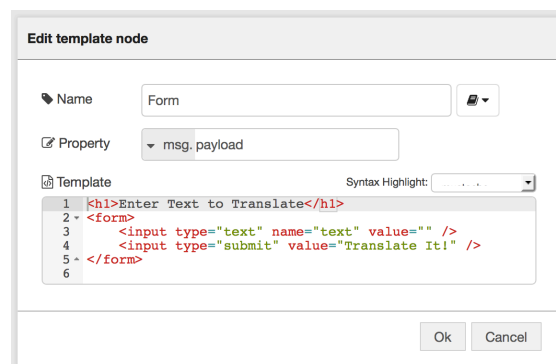
The 'Edit switch node' dialog box shows the following configuration:

- Name: Check Text
- Property: msg.payload.text
- Rule 1: is null → 1
- Rule 2: otherwise → 2
- Stopping after first match

Buttons: Ok, Cancel

If webpage is called with a query parameter named text, the first flow will be processed. Otherwise, the second flow will be processed.

3. For the first flow, add a  node to display a form for the user to input text to translate as shown below.



The 'Edit template node' dialog box shows the following configuration:

- Name: Form
- Property: msg.payload
- Template:


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

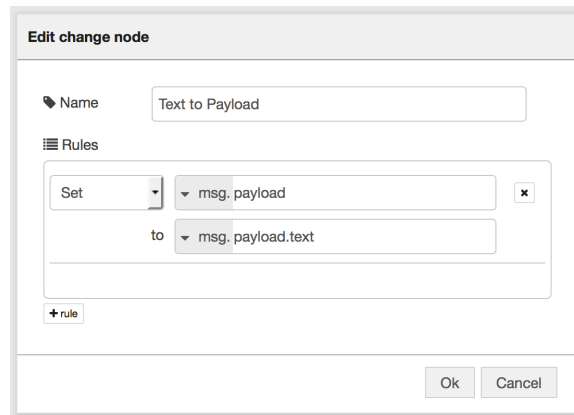
Buttons: Ok, Cancel

4. Add a  node and connect the nodes together as shown below.



This will display the form when the user opens a browser and goes to the application URL, appended by /translate.

5. Next, add a  node to extract the text from the query parameter and put it in the msg.payload for the Language Translation node.




**Edit change node**

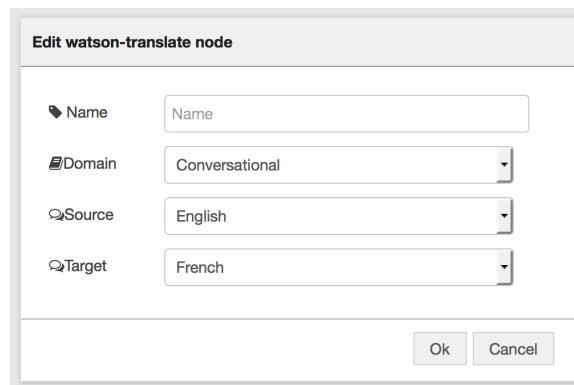
Name: Text to Payload

Rules:

Set msg.payload to msg.payload.text

Ok Cancel

6. Add a  node as shown below. For this example, we'll translate English text into French.



**Edit watson-translate node**

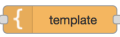
Name: Name

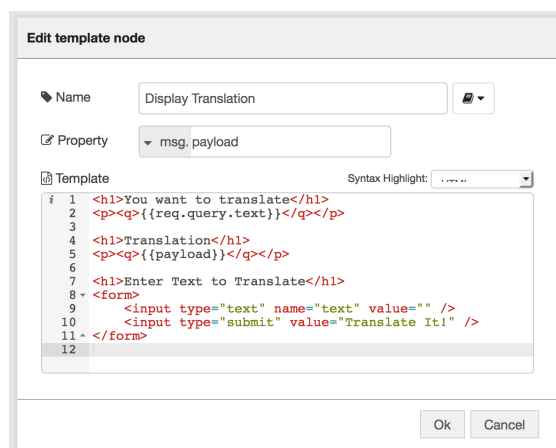
Domain: Conversational

Source: English

Target: French

Ok Cancel

7. Add a  node to display the result along with the form for the user to translate another piece of text.



**Edit template node**


Name: Display Translation

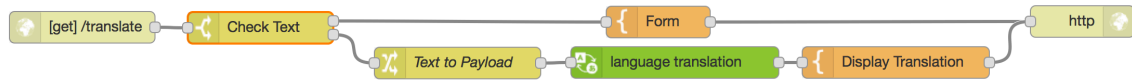
Property: msg.payload

Template:

```
1 <h1>You want to translate</h1>
2 <p><q>{{req.query.text}}</q></p>
3
4 <h1>Translation</h1>
5 <p><q>{{payload}}</q></p>
6
7 <h1>Enter Text to Translate</h1>
8 <form>
9   <input type="text" name="text" value="" />
10  <input type="submit" value="Translate It!" />
11 </form>
12
```

Ok Cancel

8. Connect the nodes together as shown below, and click on  to save the changes.



9. Test the application by visiting the application URL, appended with /translate.

`https://_____.mybluemix.net/translate`