**P roject Report**

**Name** : SURESH SURYANATH MAURYA

**Title** : Intelligent Customer Help Desk With Smart Document Understanding

**Category:**Artificial Intelligence

Table of contents

1. INTRODUCTION
   1. Overview
   2. Purpose
2. LITERATURE SURVEY
   1. Existing problem
   2. Proposed solution
3. THEORITICAL ANALYSIS
   1. Block diagram
   2. Hardware / Software designing 4 EXPERIMENTAL INVESTIGATIONS 5 FLOWCHART
4. RESULT
5. ADVANTAGES & DISADVANTAGES
6. APPLICATIONS
7. CONCLUSION
8. FUTURE SCOPE
9. BIBILOGRAPHY APPENDIX
10. Source code
11. Reference

INTRODUCTION

* 1. Overview: We will be able to write an application that leverages multiple Watson AI Services (Discovery , Assistant, Cloud function and Node Red). By the end of the project, we’ll learn best practices of combining Watson services, and how they can build interactive information retrieval systems with Discovery + Assistant.
     + Project Requirements: Python, IBM Cloud, IBM Watson
     + Functional Requirements: IBM cloud
     + Technical Requirements: AI,ML,WATSON AI,PYTHON
     + Software Requirements: Watson assistant, Watson discovery.
     + Project Deliverables: Smartinternz Intership
     + Project Member: Sumith Pevekar
     + Project Duration:19 days
  2. Purpose:

The typical customer care chatbot can answer simple questions, such as store locations and hours, directions, and maybe even making appointments. When a question falls outside of the scope of the pre-determined question set, the option is typically to tell the customer the question isn’t valid or offer to speak to a real person.

In this project, there will be another option. If the customer question is about the operation of a device, the application shall pass the question onto Watson Discovery Service, which has been pre-loaded with the device’s owners manual. So now, instead of “Would you like to speak to a customer representative?” we can return relevant sections of the owners manual to help solve our customers’ problems.

To take it a step further, the project shall use the Smart Document Understanding feature of Watson Discovery to train it on what text in the owners manual is important and what is not. This will improve the answers returned from the queries.

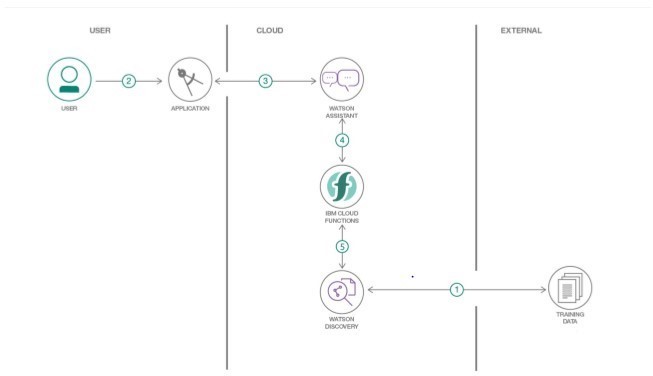
* + 1. Scope of Work
       - Create a customer care dialog skill in Watson Assistant
       - Use Smart Document Understanding to build an enhanced Watson Discovery collection
       - Create an IBM Cloud Functions web action that allows Watson Assistant to post queries to Watson Discovery
       - Build a web application with integration to all these services & deploy the same on IBM Cloud Platform
  1. LITERATURE SURVEY

1. Existing problem:

Generally Chatbots means getting input from users and getting only response questions and for some questions the output from bot will be like “try again”, “I don’t understand”, “will you repeat again”, and so on… and directs customer to customer agent but a good customer Chatbot should minimize involvement of customer agent to chat with customer to clarify his/her doubts. So to achieve this we should include an virtual agent in chatbot so that it will take care of real involvement of customer agent and customer can clariﬁes his doubts with fast chatbots.

1. Proposed solution:

For the above problem to get solved we have to put an virtual agent in chatbot so it can understand the queries that are posted by customers. The virtual agent should trained from some insight records based company background so it can answer queries based on the product or related to company. In this project I used Watson Discovery to achieve the above solution. And later including Assistant and Discovery on Node-RED

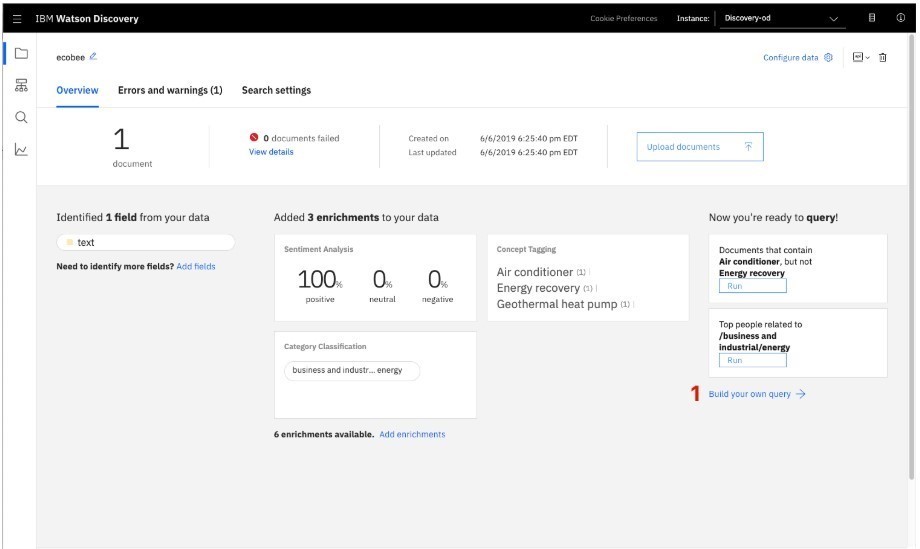
1. THEORITICAL ANALYSIS
   1. Block/Flow Diagram
      1. The document is annotated using Watson Discovery SDU
      2. The user interacts with the backend server via the app UI. The frontend app UI is a chatbot that engages the user in a conversation.
      3. Dialog between the user and backend server is coordinated using a Watson Assistant dialog skill.
      4. If the user asks a product operation question, a search query is passed to a predeﬁned IBM Cloud Functions action.
      5. The Cloud Functions action will query the Watson Discovery service and return the results.
   2. Hardware / Software designing:
      1. Create IBM Cloud services
      2. Conﬁgure Watson Discovery
      3. Create IBM Cloud Functions action
      4. Conﬁgure Watson Assistant
      5. Create ﬂow and conﬁgure node
      6. Deploy and run Node Red app.

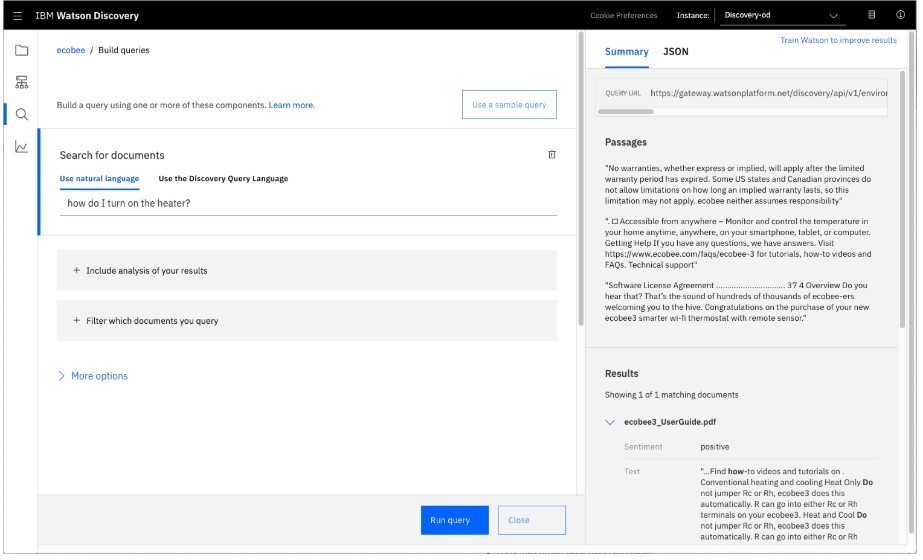
4.EXPERIMENTAL INVESTIGATIONS

1. Create IBM Cloud services Create the following services:
   * Watson Discovery
   * Watson Assistant
   * Node Red
2. Conﬁgure Watson Discovery Import the document

Launch the Watson Discovery tool and create a new data collection by selecting the Upload your own data option. Give the data collection a unique name. When prompted, select and upload the ecobee3\_UserGuide.pdf ﬁle located in the data directory of your local repo.

The Ecobee is a popular residential thermostat that has a wiﬁ interface and multiple conﬁguration options.

Before applying SDU to our document, lets do some simple queries on the data so that we can compare it to results found after applying SDU.

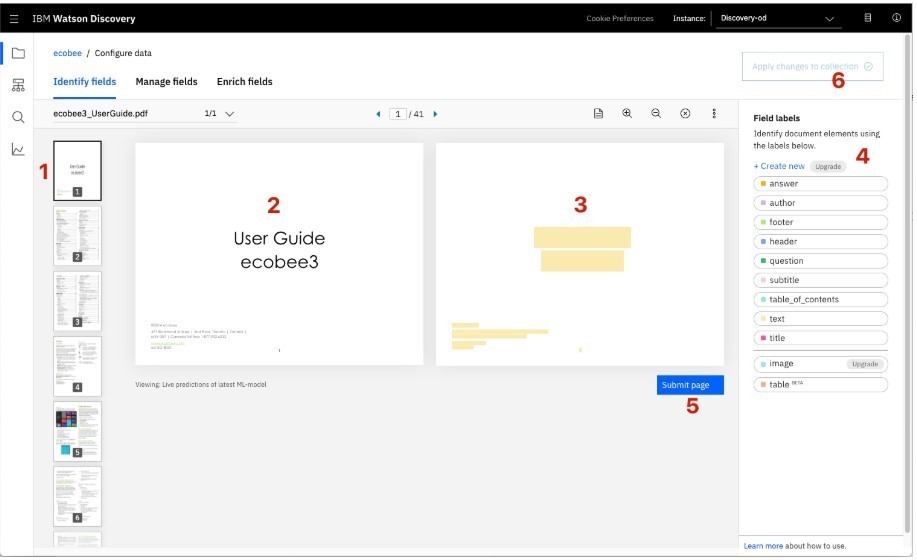
Click the Build your own query [1] button.

Enter queries related to the operation of the thermostat and view the results. As you will see, the results are not very useful, and in some cases, not even related to the question.

Annotate with SDU

Now let's apply SDU to our document to see if we can generate some better query responses.From the Discovery collection panel, click the Conﬁgure data button (located in the top right corner) to start the SDU process.

Here is the layout of the Identify ﬁelds tab of the SDU annotation panel:



The goal is to annotate all of the pages in the document so Discovery can learn what text is important, and what text can be ignored.

1. is the list of pages in the manual. As each is processed, a green check mark will appear on the page.
2. is the current page being annotated.
3. is where you select text and assign it a label.
4. is the list of labels you can assign to the page text. Click [5] to submit the page to Discovery.

Click [6] when you have completed the annotation process.

As you go though the annotations one page at a time, Discovery is learning and should start automatically updating the upcoming pages. Once you get to a page that is already correctly annotated, you can stop, or simply click Submit [5] to acknowledge it is correct. The more pages you annotate, the better the model will be trained.

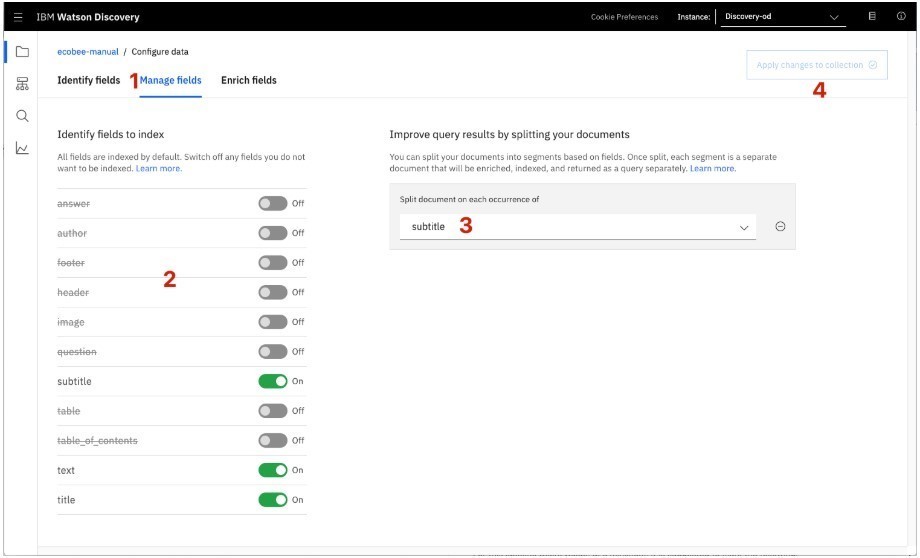
For this speciﬁc owner's manual, at a minimum, it is suggested to mark the following: The main title page as title

The table of contents (shown in the ﬁrst few pages) as table\_of\_contents All headers and sub-headers (typed in light green text) as a subtitle

All page numbers as footers

All warranty and licensing infomation (located in the last few pages) as a footer All other text should be marked as text.

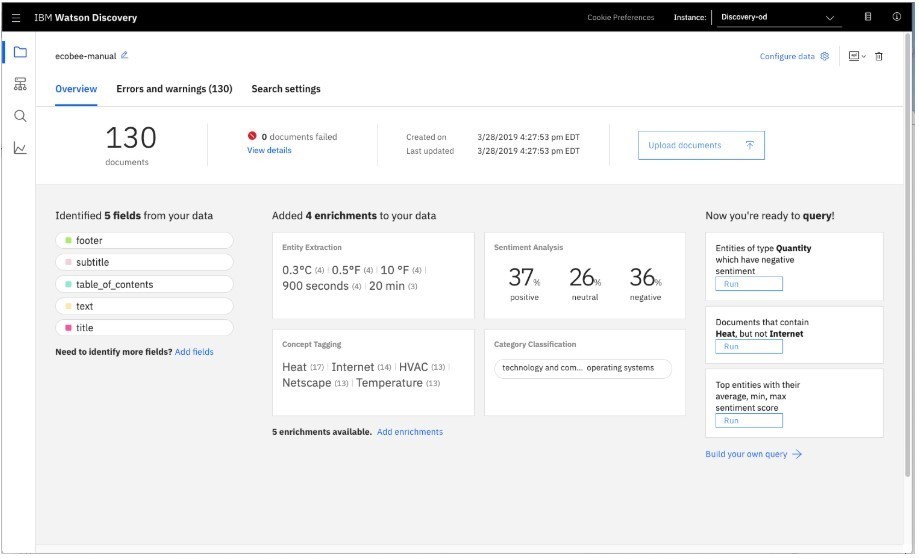
Once you click the Apply changes to collection button [6], you will be asked to reload the

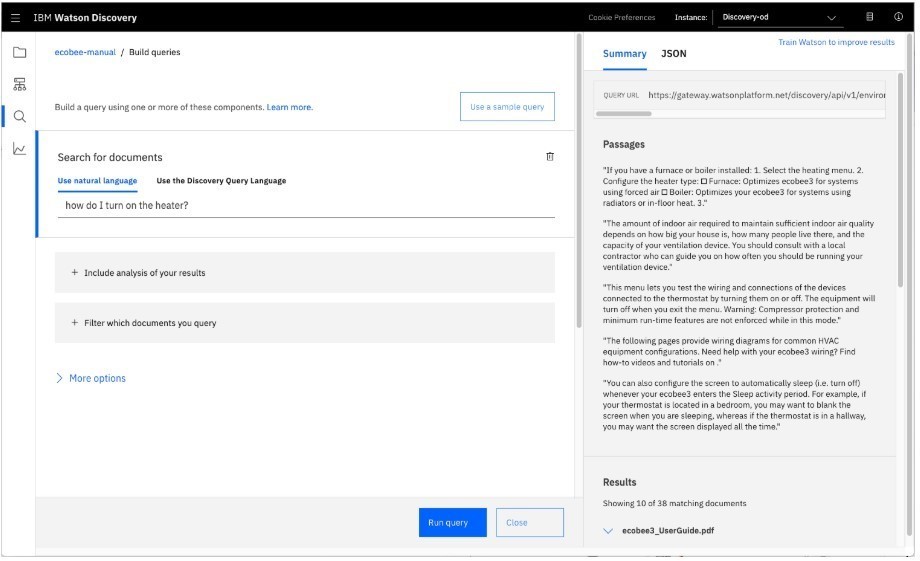
document. Choose the same owner's manual .pdf document as before. Next, click on the Manage ﬁelds [1] tab.

1. Here is where you tell Discovery which ﬁelds to ignore. Using the on/off buttons, turn off all labels except subtitles and text.
2. is telling Discovery to split the document apart, based on subtitle. Click [4] to submit your changes.

Once again, you will be asked to reload the document.

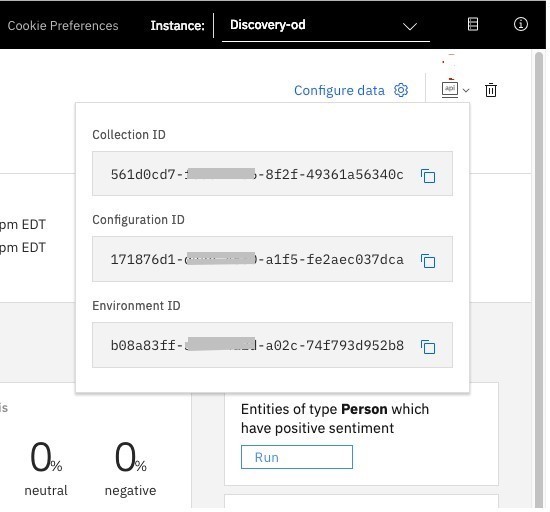
Now, as a result of splitting the document apart, your collection will look very different:



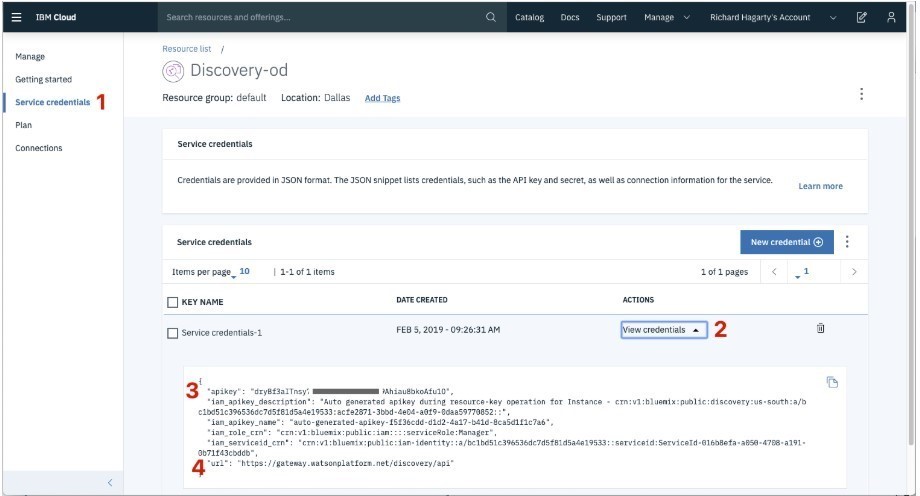
Return to the query panel (click Build your own query) and see how much better the results are.

Store credentials for future use

In upcoming steps, you will need to provide the credentials to access your Discovery collection. The values can be found in the following locations.

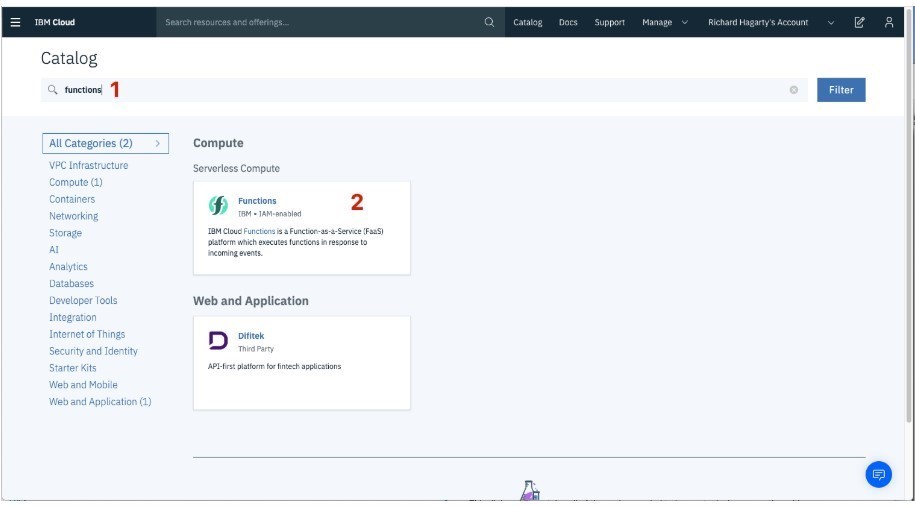
The Collection ID and Environment ID values can be found by clicking the dropdown button [1] located at the top right side of your collection panel:

For credentials, return to the main panel of your Discovery service, and click the Service credentials [1] tab:

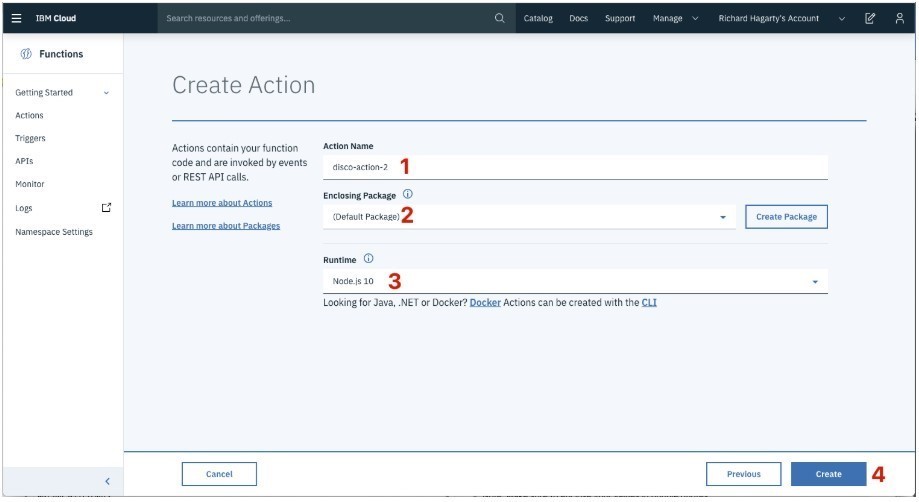


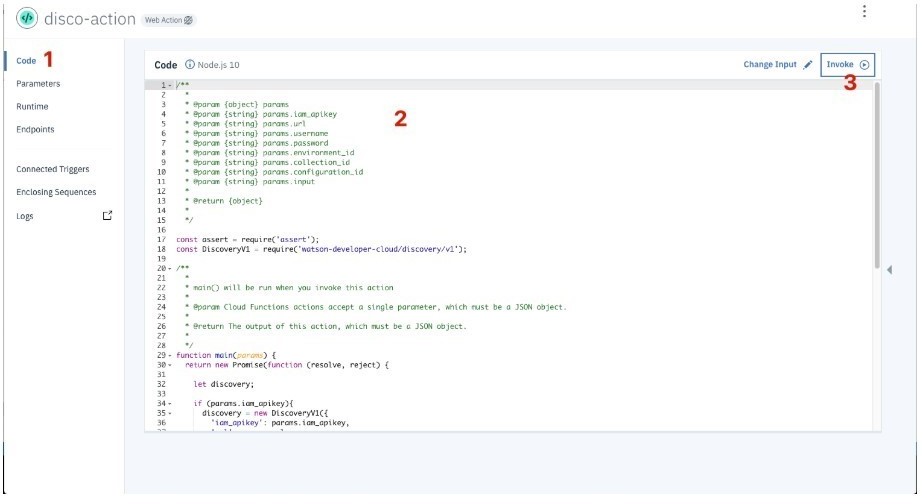
Click the View credentials [2] drop-down menu to view the IAM apikey [3] and URL endpoint [4] for your service.

1. Create IBM Cloud Functions action

Now let's create the web action that will make queries against our Discovery collection. Start the IBM Cloud Functions service by selecting Create Resource from the IBM Cloud dashboard. Enter functions as the ﬁlter [1], then select the Functions card [2]:

From the Functions main panel, click on the Actions tab. Then click on Create. From the Create panel, select the Create Action option.

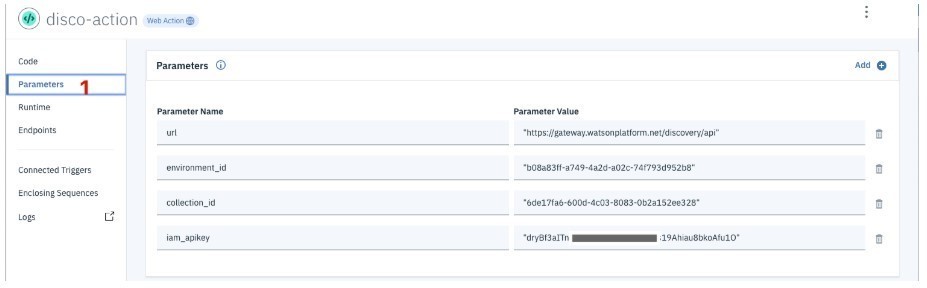
On the Create Action panel, provide a unique Action Name [1], keep the default package [2], and select the Node.js 10 [3] runtime. Click the Create button [4] to create the action.

Once your action is created, click on the Code tab [1]:

In the code editor window [2], cut and paste in the code from the disco-action.js ﬁle found in the actions directory of your local repo. The code is pretty straight-forward - it simply connects to

the Discovery service, makes a query against the collection, then returns the response.

If you press the Invoke button [3], it will fail due to credentials not being deﬁned yet. We'll do this next.

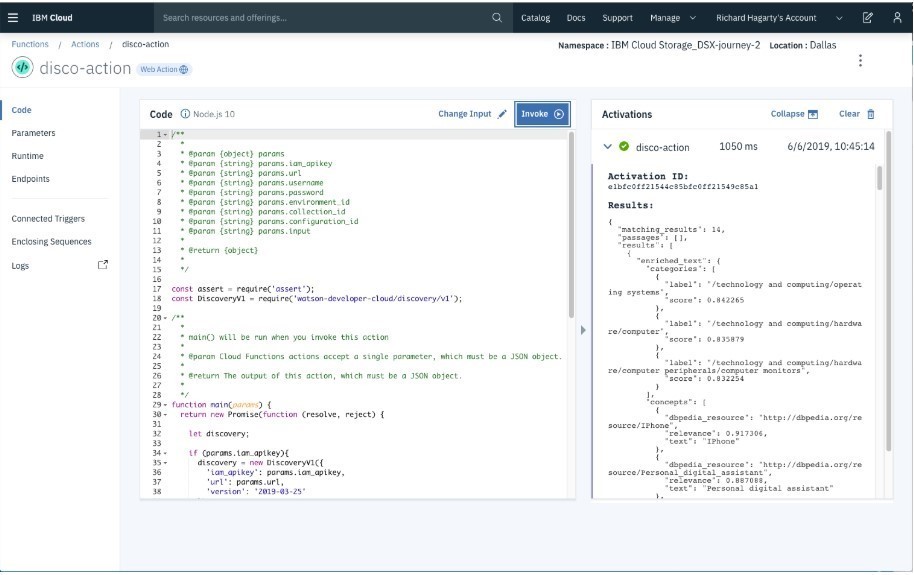
Select the Parameters tab [1]:

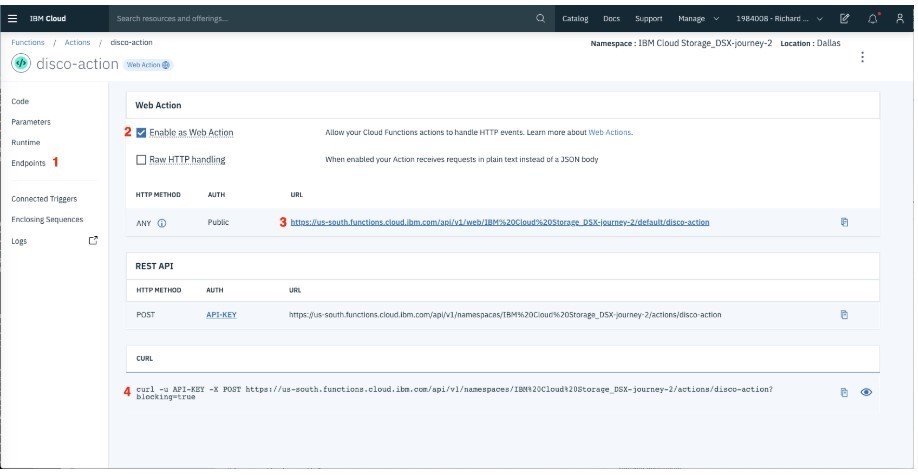
Add the following keys:

* + url
  + environment\_id
  + collection\_id
  + iam\_apikey

For values, please use the values associated with the Discovery service you created in the previous step.

Now that the credentials are set, return to the Code panel and press the Invoke button again. Now you should see actual results returned from the Discovery service:



Next, go to the Endpoints panel [1]:

Click the checkbox for Enable as Web Action [2]. This will generate a public endpoint URL [3].

Take note of the URL value [3], as this will be needed by Watson Assistant in a future step.

To verify you have entered the correct Discovery parameters, execute the provied curl command [4]. If it fails, re-check your parameter values.

1. Conﬁgure Watson Assistant

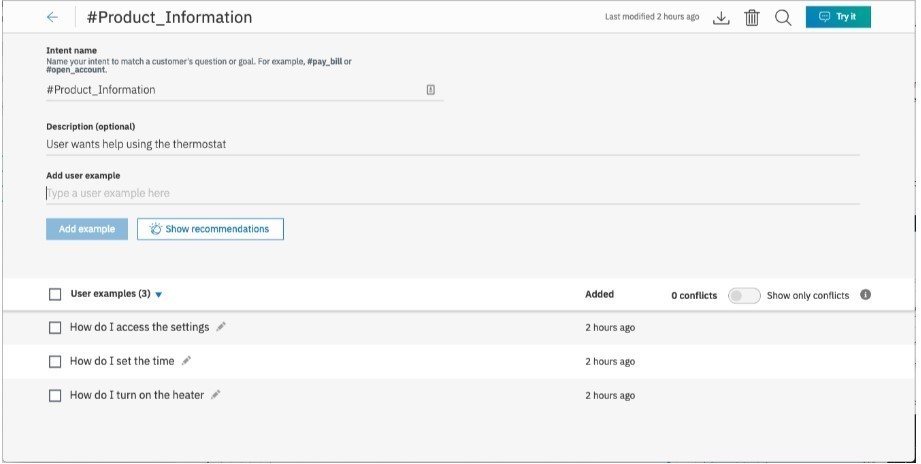
Launch the Watson Assistant tool and create a new dialog skill. Select the Use sample skill option as your starting point.This dialog skill contains all of the nodes needed to have a typical call center conversation with a user.

Add new intent

The default customer care dialog does not have a way to deal with any questions involving outside resources, so we will need to add this.

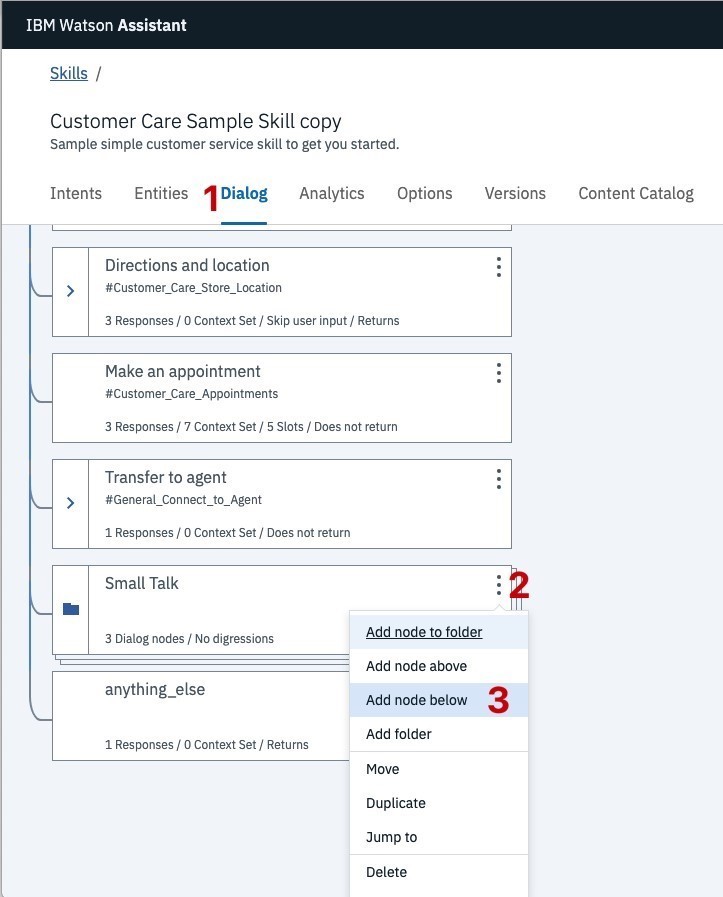
Create a new intent that can detect when the user is asking about operating the Ecobee thermostat.

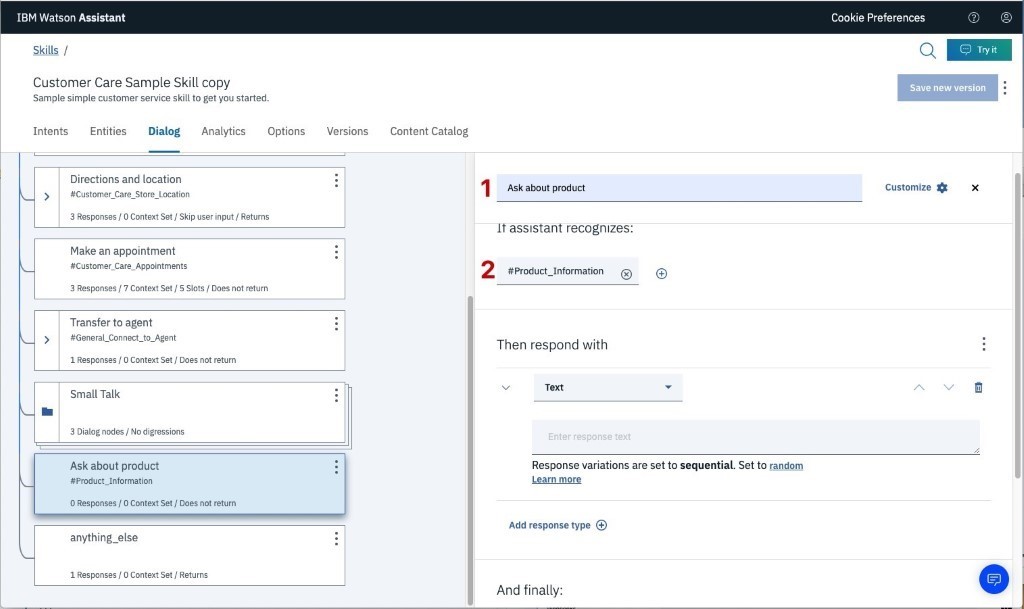
From the Customer Care Sample Skill panel, select the Intents tab. Click the Create intent button.

Name the intent #Product\_Information, and at a minimum, enter the following example questions to be associated with it.

Create new dialog node

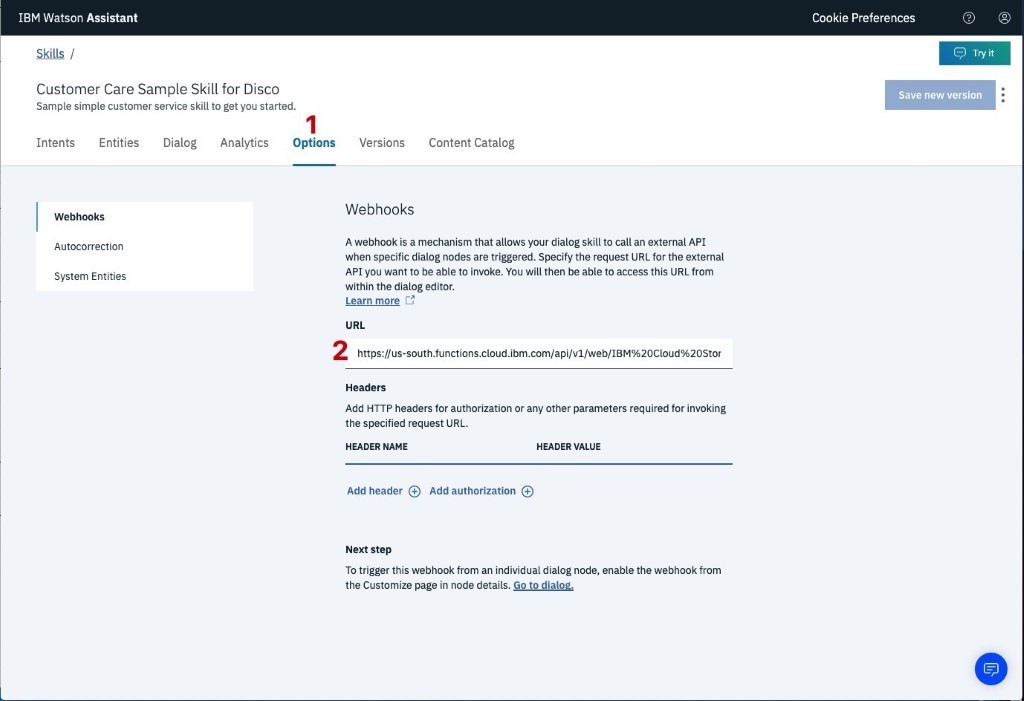
Now we need to add a node to handle our intent. Click on the Dialog [1] tab, then click on the drop down menu for the Small Talk node [2], and select the Add node below [3] option.



Name the node "Ask about product" [1] and assign it our new intent [2].

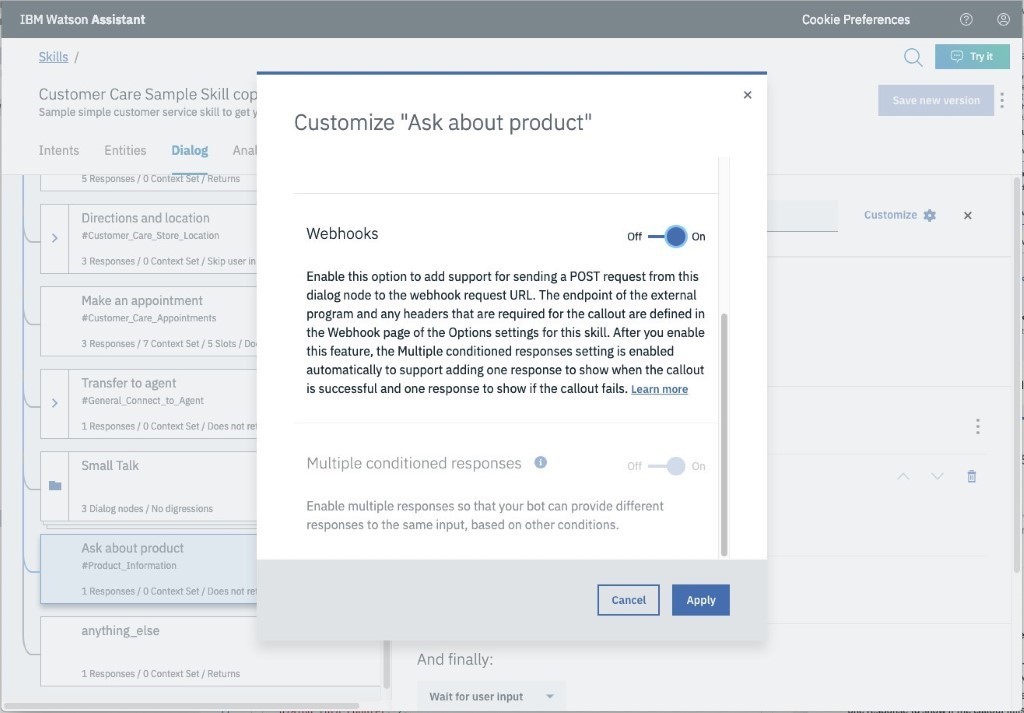
This means that if Watson Assistant recognizes a user input such as "how do I set the time?", it

will direct the conversation to this node. Enable webhook from Assistant

Set up access to our WebHook for the IBM Cloud Functions action you created in Step #4. Select the Options tab [1]:

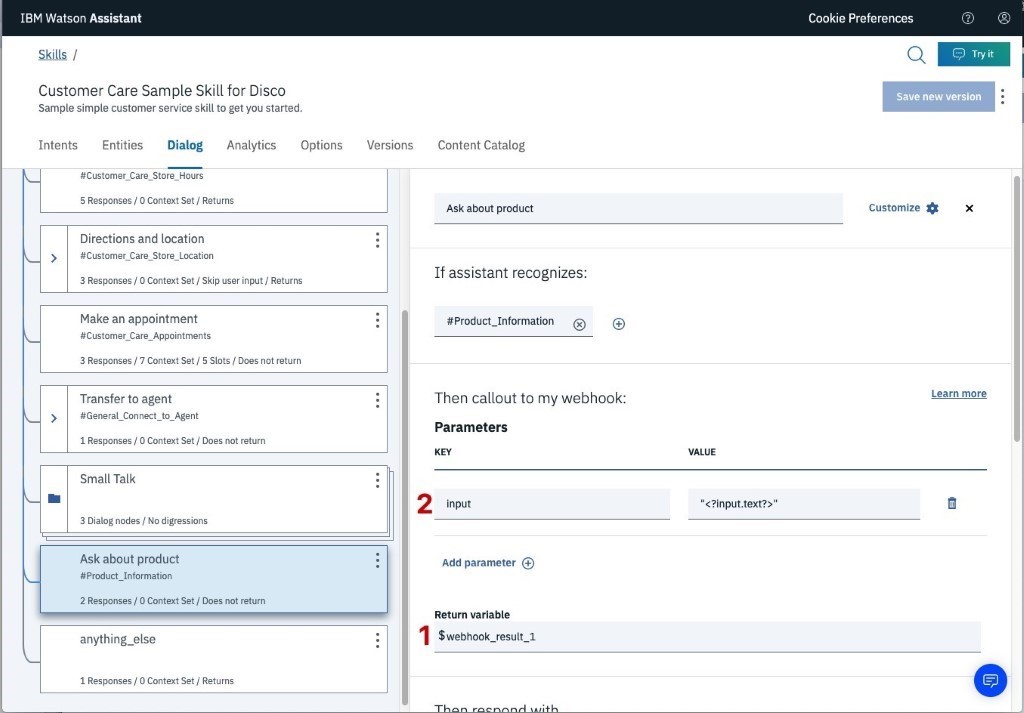
Enter the public URL endpoint for your action [2].

Return to the Dialog tab, and click on the Ask about product node. From the details panel for the node, click on Customize, and enable Webhooks for this node:



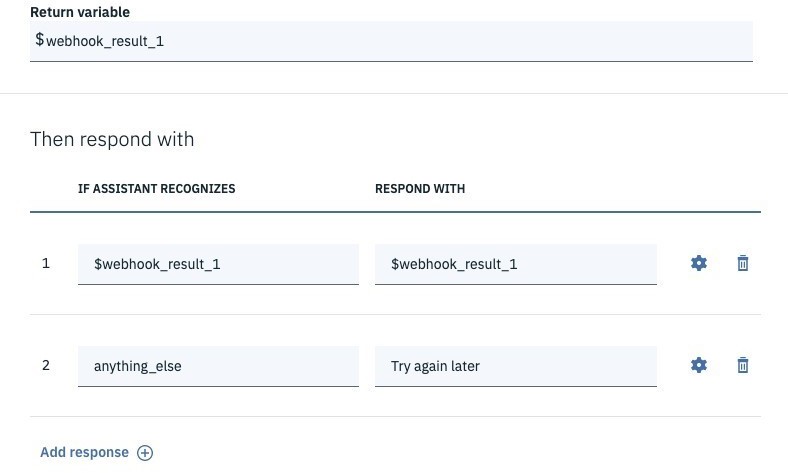
Click Apply.

The dialog node should have a Return variable [1] set automatically to $webhook\_result\_1. This is the variable name you can use to access the result from the Discovery service query.

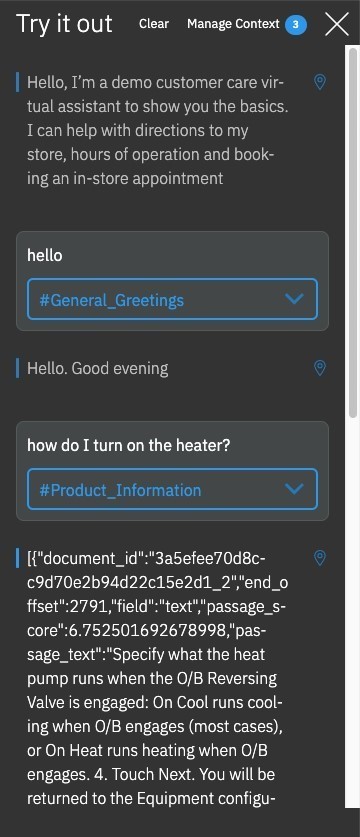


You will also need to pass in the users question via the parameter input [2]. The key needs to be set to the value: "<?input.text?>"

If you fail to do this, Discovery will return results based on a blank query. Optionally, you can add these responses to aid in debugging:



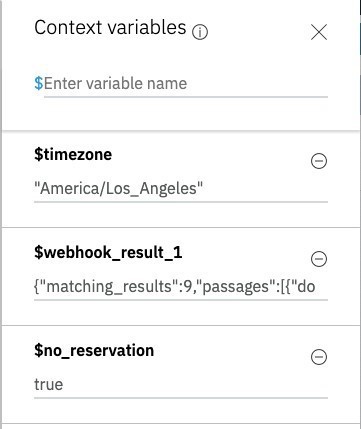
Test in Assistant Tooling

From the Dialog panel, click the Try it button located at the top right side of the panel. Enter some user input:

Note that the input "how do I turn on the heater?" has triggered our Ask about product dialog node, which is indicated by the #Product\_Information response.

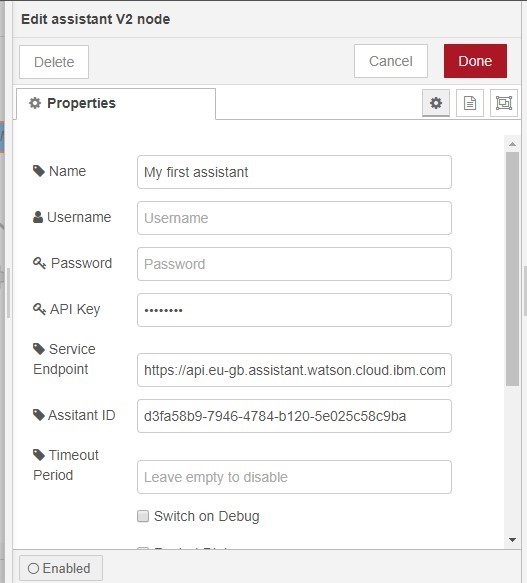
And because we speciﬁed that $webhook\_result\_1.passages be the response, that value is displayed also.

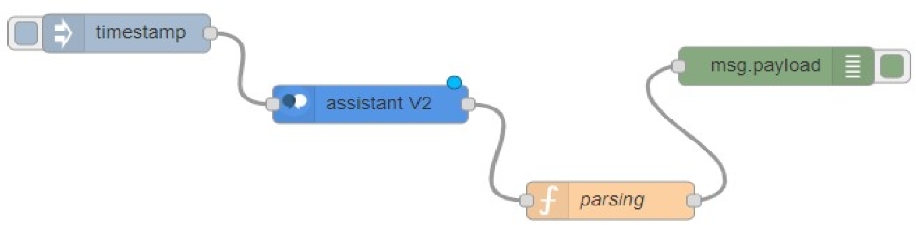
You can also verify that the call was successfully completed by clicking on the Manage Context button at the top right. The response from the Discovery query will be stored in the

$webhook\_result\_1 variable:

1. Create ﬂow and conﬁgure node:

Integration of watson assistant in Node-RED

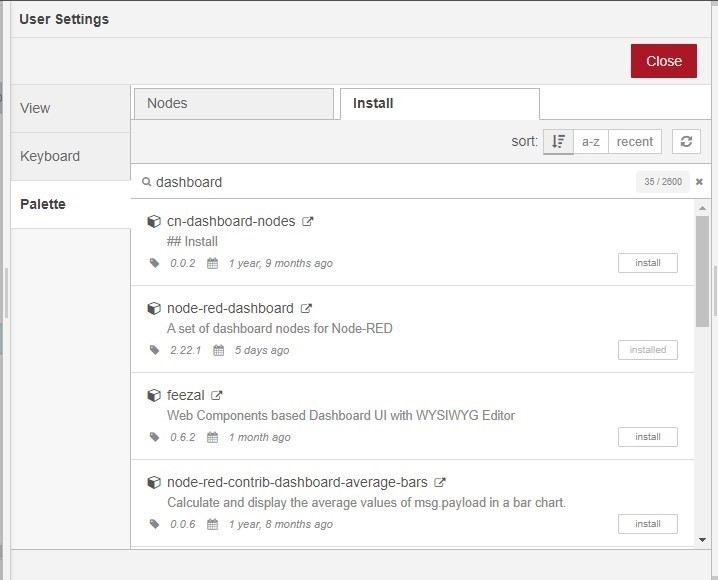
* + Double-click on the Watson assistant node
  + Give a name to your node and enter the username, password and workspace id of your Watson assistant service
  + After entering all the information click on Done
  + Drag inject node on to the ﬂow from the Input section
  + Drag Debug on to the ﬂow from the output section
  + Double-click on the inject node
  + Select the payload as a string
  + Enter a sample input to be sent to the assistant service and click on done
  + Connect the nodes as shown below and click on Deploy



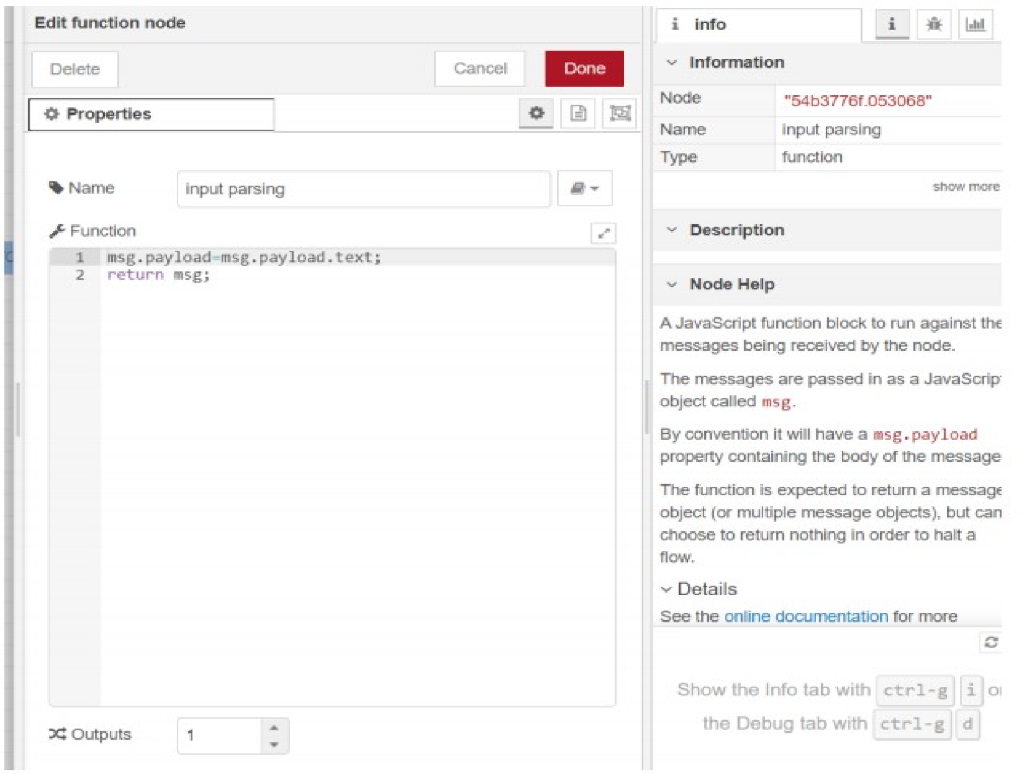
* + Open Debug window as shown below
  + Click on the button to send input text to the assistant node
  + Observe the output from the assistant service node
  + The Bot output is located inside “output.text"
  + Drag the function node to parse the JSON data and get the bot response
  + Double click on the function node and enter the JSON parsing code as shown below and click on done ● Connect the nodes as shown below and click on Deploy
  + Re-inject the ﬂow and observe the parsed output
  + For creating a web application

UI we need “dashboard“ nodes which should be installed manually.

* + Go to navigation pane and click on manage palette



* + Click on install
  + Search for “node-red-dashboard” and click on install and again click on install on the prompt
  + The following message indicates dashboard nodes are installed, close the manage palette
  + Search for “Form” node and drag on to the ﬂow
  + Doube click on the “form” node to conﬁgure
  + Click on the edit button to add the “Group” name and “Tab” name
  + Click on the edit button to add tab name to web application
  + Give sample tab name and click on add do the same thing for the group
  + Give the label as “Enter your input”, Name as “text” and click on Done
  + Drag a function node, double-click on it and enter the input parsing code as shown below

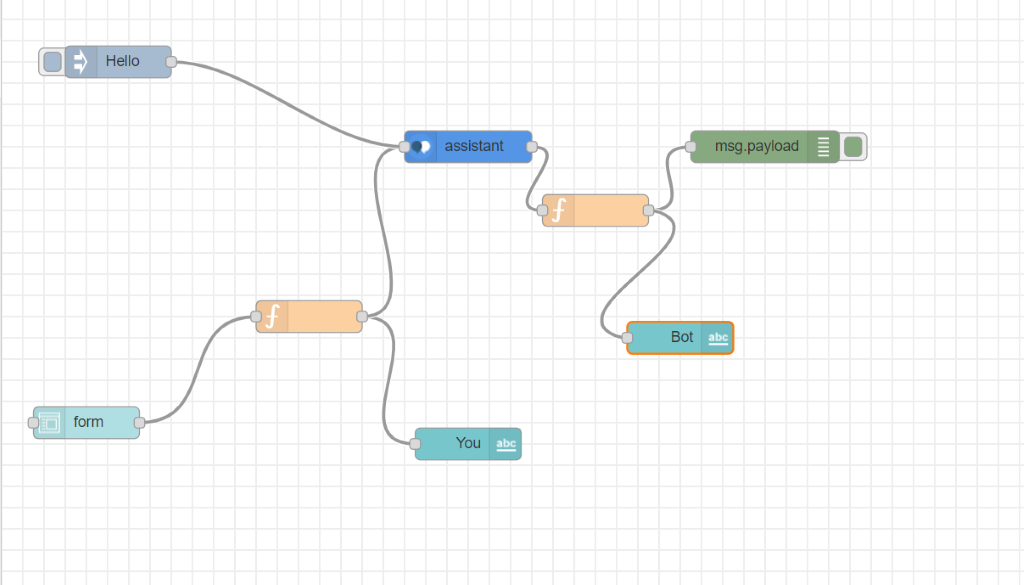


* + Click on done
  + Connect the form output to the input of the function node and output of the function to input of assistant node
  + Search for “text” node from the “dashboard” section
  + Drag two “text” nodes on to the ﬂow
  + Double click on the ﬁrst text node, change the label as “You” and click on Done
  + Double click on the second text node, change the label as “Bot” and click on Done
  + Connect the output of “input parsing” function node to “ You” text node and output of “Parsing” function node to the input of “Bot” text node
  + Click on Deploy

1. FLOWCHART

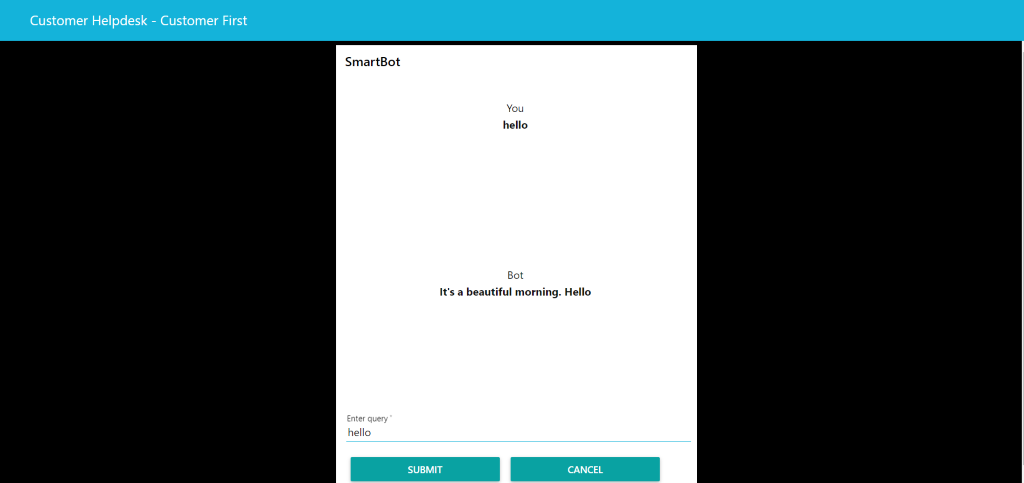
First, go to manage pallete and install dashboard. Now, Create the ﬂow with the help of following nodes:

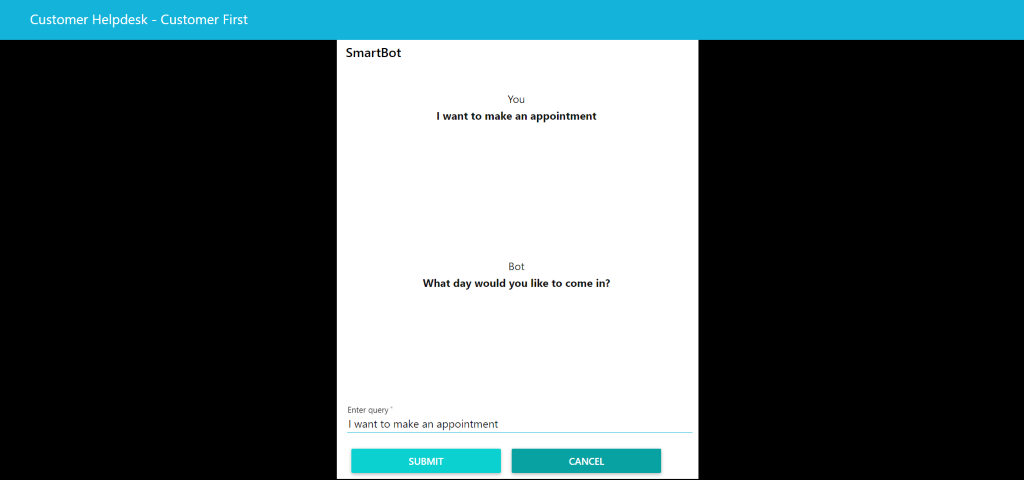
* Inject
* Assistant
* Debug
* Function Ui\_Form
* Ui\_Text



1. RESULTS

Finally our Node-RED dash board integrates all the components and displays it in the dashboard UI by typing URL: [https:/ smart-internz.eu-gb.mybluemix.net/ui/](https://smart-internz.eu-gb.mybluemix.net/ui/)





1. ADVANTAGES & DISADVANTAGES

Advantages:

* Campanies can deploy chatbots to rectiﬁy simple and general human queries .
* No need to divert calls to customer agent and customer agent can look on other works.
* Reduces human workforce .
* Cost eﬃcient . Disadvantages:
* Some times chatbot can mislead customers
* Giving same answer for different sentiments.

1. APPLICATIONS

* Chatbots can be deploy on popular social media platforms like facebook,slack,telegram.
* Chatbots can be used to book appointments, cabs with the help of some steps.

1. CONCLUSION

After completing all the above procedures we have successfully created Intelligent helpdesk smart chartbot using Watson assistant, Watson discovery, Node-RED and Cloud-functions.

1. FUTURE SCOPE

Seamlessly automate tasks. Address customer requests across channels like digital and voice. Guide employees through internal processes. Best of all: allow your teams to focus on higher value work.Also watson studio text to speech and speech to text services could be included to access the chatbot handsfree.

1. BIBILOGRAPHY

APPENDIX

# Source Code

1. **Cloud Function**

/\*\*

\*

* + @param {object} params
  + @param {string} params.iam\_apikey
  + @param {string} params.url
  + @param {string} params.username
  + @param {string} params.password
  + @param {string} params.environment\_id
  + @param {string} params.collection\_id
  + @param {string} params.conﬁguration\_id
  + @param {string} params.input

\*

* + @return {object}

\*

\*/

const assert = require('assert');

const DiscoveryV1 = require('watson-developer-cloud/discovery/v1');

/\*\*

\*

* + main() will be run when you invoke this action

\*

* + @param Cloud Functions actions accept a single parameter, which must be a JSON object.

\*

* + @return The output of this action, which must be a JSON object.

\*

\*/

function main(params) {

return new Promise(function (resolve, reject) {

let discovery;

if (params.iam\_apikey){ discovery = new DiscoveryV1({

'iam\_apikey': params.iam\_apikey, 'url': params.url,

'version': '2019-03-25'

});

}

else {

discovery = new DiscoveryV1({ 'username': params.username, 'password': params.password, 'url': params.url,

'version': '2019-03-25'

});

}

discovery.query({

'environment\_id': params.environment\_id, 'collection\_id': params.collection\_id, 'natural\_language\_query': params.input, 'passages': true,

'count': 3,

'passages\_count': 3

}, function(err, data) { if (err) {

return reject(err);

}

return resolve(data);

});

});

}

# Node Red (ﬂow.json)

[

{

"id": "a130917f.9242",

"type": "tab",

"label": "Flow 1", "disabled": false, "info": ""

},

{

"id": "8a86fd0e.3af4e",

"type": "tab",

"label": "Flow 1", "disabled": true, "info": ""

},

{

"id": "86f63600.4b1868",

"type": "ui\_base", "theme": {

"name": "theme-custom", "lightTheme": {

"default": "#0094CE", "baseColor": "#0094CE",

"baseFont": "-apple-system,BlinkMacSystemFont,Segoe

UI,Roboto,Oxygen-Sans,Ubuntu,Cantarell,Helvetica Neue,sans-serif", "edited": true,

"reset": false

},

"darkTheme": { "default": "#097479",

"baseColor": "#097479",

"baseFont": "-apple-system,BlinkMacSystemFont,Segoe UI,Roboto,Oxygen-Sans,Ubuntu,Cantarell,Helvetica Neue,sans-serif",

"edited": false

},

"customTheme": {

"name": "Untitled Theme 1", "default": "#4B7930", "baseColor": "#09a2a2",

"baseFont": "-apple-system,BlinkMacSystemFont,Segoe

UI,Roboto,Oxygen-Sans,Ubuntu,Cantarell,Helvetica Neue,sans-serif", "reset": false

},

"themeState": { "base-color": {

"default": "#4B7930", "value": "#09a2a2", "edited": true

},

"page-titlebar-backgroundColor": { "value": "#14b3da",

"edited": true

},

"page-backgroundColor": { "value": "#000000",

"edited": true

},

"page-sidebar-backgroundColor": { "value": "#ffffff",

"edited": false

},

"group-textColor": { "value": "#000000",

"edited": true

},

"group-borderColor": { "value": "#ffffff", "edited": false

},

"group-backgroundColor": { "value": "#ffffff",

"edited": false

},

"widget-textColor": { "value": "#111111",

"edited": false

},

"widget-backgroundColor": { "value": "#09a2a2", "edited": false

},

"widget-borderColor": { "value": "#ffffff", "edited": false

},

"base-font": {

"value": "-apple-system,BlinkMacSystemFont,Segoe UI,Roboto,Oxygen-Sans,Ubuntu,Cantarell,Helvetica Neue,sans-serif"

}

},

"angularTheme": { "primary": "indigo",

"accents": "blue",

"warn": "red", "background": "grey"

}

},

"site": {

"name": "Node-RED Dashboard", "hideToolbar": "false", "allowSwipe": "false", "lockMenu": "false", "allowTempTheme": "true", "dateFormat": "DD/MM/YYYY", "sizes": {

"sx": 48,

"sy": 48,

"gx": 6,

"gy": 6,

"cx": 6,

"cy": 14,

"px": 0,

"py": 0

}

}

},

{

"id": "72d2a248.15bb4c",

"type": "ui\_group",

"z": "",

"name": "SmartBot",

"tab": "78b79302.4ea95c",

"order": 1,

"disp": true,

"width": 10, "collapse": false

},

{

"id": "78b79302.4ea95c",

"type": "ui\_tab",

"z": "",

"name": "Customer Helpdesk - Customer First", "icon": "dashboard",

"order": 1, "disabled": false, "hidden": false

},

{

"id": "d9af7ee8.b414c",

"type": "ui\_spacer",

"name": "spacer",

"group": "72d2a248.15bb4c", "order": 3,

"width": 10,

"height": 1

},

{

"id": "ec4df304.14614",

"type": "ui\_spacer",

"name": "spacer",

"group": "72d2a248.15bb4c", "order": 6,

"width": 10,

"height": 1

},

{

"id": "1100ea37.babbc6",

"type": "inject",

"z": "8a86fd0e.3af4e",

"name": "",

"topic": "",

"payload": "Hello Node-RED!", "payloadType": "str",

"repeat": "",

"crontab": "", "once": false, "onceDelay": "", "x": 180,

"y": 100,

"wires": [ [

"5152dc00.ba16c4"

]

]

},

{

"id": "336284b6.78302c",

"type": "debug",

"z": "8a86fd0e.3af4e",

"name": "", "active": true, "tosidebar": true, "console": false, "tostatus": false,

"complete": "payload", "targetType": "msg", "x": 830,

"y": 100,

"wires": []

},

{

"id": "e6285788.ae2808",

"type": "function",

"z": "8a86fd0e.3af4e",

"name": "",

"func": "msg.payload=msg.payload.text;\nreturn msg;", "outputs": 1,

"noerr": 0,

"x": 210,

"y": 300,

"wires": [ [

"5152dc00.ba16c4", "ab0f1f71.88677"

]

]

},

{

"id": "ab0f1f71.88677",

"type": "ui\_text",

"z": "8a86fd0e.3af4e", "group": "72d2a248.15bb4c", "order": 1,

"width": 10,

"height": 3, "name": "",

"label": "You",

"format": "{{msg.payload}}", "layout": "col-center",

"x": 410,

"y": 400,

"wires": []

},

{

"id": "804741be.6cd0b",

"type": "ui\_text",

"z": "8a86fd0e.3af4e", "group": "72d2a248.15bb4c", "order": 4,

"width": 10,

"height": 4, "name": "",

"label": "Bot",

"format": "{{msg.payload}}", "layout": "col-center",

"x": 890,

"y": 320,

"wires": []

},

{

"id": "d11001f4.df857",

"type": "function",

"z": "8a86fd0e.3af4e",

"name": "",

"func": "msg.payload = msg.payload.output.generic[0].text;\nreturn msg;",

"outputs": 1,

"noerr": 0,

"x": 630,

"y": 240,

"wires": [ [

"336284b6.78302c",

"804741be.6cd0b"

]

]

},

{

"id": "5152dc00.ba16c4", "type": "watson-assistant-v2", "z": "8a86fd0e.3af4e",

"name": "Assistant", "service-endpoint": "

[https:/ api.eu-gb.assistant.watson.cloud.ibm.com/instances/614b7370-71e9-4d94-a573-f6711](https://api.eu-gb.assistant.watson.cloud.ibm.com/instances/614b7370-71e9-4d94-a573-f67113d28833) [3d28833](https://api.eu-gb.assistant.watson.cloud.ibm.com/instances/614b7370-71e9-4d94-a573-f67113d28833)",

"assistant\_id": "299bebed-895e-4e40-a609-5df64630874a", "debug": true,

"restart": false, "return\_context": false, "alternate\_intents": false, "multisession": false, "timeout": "",

"optout-learning": false, "x": 400,

"y": 180,

"wires": [ [

"d11001f4.df857"

]

]

},

{

"id": "697c4c48.f0ad84",

"type": "inject",

"z": "a130917f.9242",

"name": "",

"topic": "",

"payload": "Hello", "payloadType": "str", "repeat": "",

"crontab": "", "once": false,

"onceDelay": 0.1,

"x": 110,

"y": 60,

"wires": [ [

"3759f513.6f0c3a"

]

]

},

{

"id": "7be4793d.61d558",

"type": "function",

"z": "a130917f.9242",

"name": "",

"func": "msg.payload=msg.payload.text;\nreturn msg;", "outputs": 1,

"noerr": 0,

"x": 290,

"y": 300,

"wires": [ [

"5f9a1230.3e329c", "3759f513.6f0c3a"

]

]

},

{

"id": "5f9a1230.3e329c",

"type": "ui\_text",

"z": "a130917f.9242",

"group": "72d2a248.15bb4c", "order": 2,

"width": 10,

"height": 2, "name": "",

"label": "You",

"format": "{{msg.payload}}", "layout": "col-center",

"x": 440,

"y": 420,

"wires": []

},

{

"id": "3759f513.6f0c3a",

"type": "watson-conversation-v1", "z": "a130917f.9242",

"name": "",

"workspaceid": "299bebed-895e-4e40-a609-5df64630874a", "multiuser": false,

"context": true,

"empty-payload": false, "service-endpoint": "

[https:/ api.eu-gb.assistant.watson.cloud.ibm.com/instances/614b7370-71e9-4d94-a573-f6711](https://api.eu-gb.assistant.watson.cloud.ibm.com/instances/614b7370-71e9-4d94-a573-f67113d28833) [3d28833](https://api.eu-gb.assistant.watson.cloud.ibm.com/instances/614b7370-71e9-4d94-a573-f67113d28833)",

"timeout": "",

"optout-learning": false, "x": 440,

"y": 140,

"wires": [ [

"39624a94.712cd6"

]

]

},

{

"id": "873d46a4.616a38",

"type": "debug",

"z": "a130917f.9242",

"name": "", "active": true, "tosidebar": true, "console": false, "tostatus": false,

"complete": "false", "x": 720,

"y": 140,

"wires": []

},

{

"id": "2e0c3f21.ee4f7",

"type": "ui\_text",

"z": "a130917f.9242",

"group": "72d2a248.15bb4c", "order": 5,

"width": 10,

"height": 4, "name": "",

"label": "Bot",

"format": "{{msg.payload}}", "layout": "col-center",

"x": 640,

"y": 320,

"wires": []

},

{

"id": "39624a94.712cd6",

"type": "function",

"z": "a130917f.9242",

"name": "",

"func": "msg.payload = msg.payload.output.generic[0].text;\nreturn msg;", "outputs": 1,

"noerr": 0,

"x": 560,

"y": 200,

"wires": [ [

"873d46a4.616a38",

"2e0c3f21.ee4f7"

]

]

},

{

"id": "98debd48.5c9aa",

"type": "ui\_form",

"z": "a130917f.9242",

"name": "",

"label": "",

"group": "72d2a248.15bb4c",

"order": 7,

"width": 10,

"height": 2,

"options": [

{

"label": "Enter query", "value": "text",

"type": "text", "required": true, "rows": null

}

],

"formValue": {

"text": ""

},

"payload": "",

"submit": "submit",

"cancel": "cancel",

"topic": "", "x": 80,

"y": 400,

"wires": [ [

"7be4793d.61d558"

]

]

}

]

References:

1. https:/ [www.ibm.com/cloud/architecture/tutorials/cognitive\_discovery](http://www.ibm.com/cloud/architecture/tutorials/cognitive_discovery)
2. https:/ cloud.ibm.com/docs/assistant?topic=assistant-getting-started
3. https:/ developer.ibm.com/recipes/tutorials/how-to-create-a-watson-chatbot-on-nodered

/

1. https:/ github.com/IBM/watson-discovery-sdu-with-assistant