Session 5 part 1: Building a Dialog

IBM Watson Conversation

Lab Instructions

Laurent Vincent

Content

Let's get started 3
1. Overview 3
2. Objectives 3
3. Prerequisites 3
4. Scenario 4
5. What to expect when you are done 5

Building more natural answers 6
6. Update Greeting answers 6
7. Test the sequence of the responses 7

Welcome statement enhancements 8
8. Add contextual details 8
9. Add the welcome node 11
10. Context variable initialisation 13
11. First "skip user input" 15

Human takeover 17
12. Increment the unanswer counter 17
13. Talk to concierge node 18
14. Does the Bot understand the user? 19

Hotel amenities management 23
15. One node and one response per condition 23
16. Single node with multiple conditions / responses 28
17. How to keep the hotel amenity context? (Hotel_hours node) 31
18. How to keep the hotel amenity context? (Hotel_locations node) 36

Find a restaurant! 43
19. Find a restaurant branch 43
20. Return responses node 44
21. Capture restaurant & mealtime nodes 46
22. Request missing information nodes 50
23. Loop management 50
24. Navigation node management 51
25. Test your branch 53

Enhance Find a restaurant branch (optional) 56
26. Initialize context variables 56
27. Avoid infinite loop 57
28. Test the enhancements 63

Let’s get started

1. Overview

The IBM Watson Developer Cloud (WDC) offers a variety of services for developing cognitive applications. Each Watson service provides a Representational State Transfer (REST) Application Programming Interface (API) for interacting with the service. Some services, such as the Speech to Text service, provide additional interfaces.

The Watson Assistant service combines several cognitive techniques to help you build and train a bot - defining intents and entities and crafting dialog to simulate conversation. The system can then be further refined with supplementary technologies to make the system more human-like or to give it a higher chance of returning the right answer. Watson Assistant allows you to deploy a range of bots via many channels, from simple, narrowly focused bots to much more sophisticated, full-blown virtual agents across mobile devices, messaging platforms like Slack, or even through a physical robot.

The **illustrating screenshots** provided in this lab guide could be slightly different from what you see in the Watson Assistant service interface that you are using. If there are colour or wording differences, it is because there have been updates to the service since the lab guide was created.

2. Objectives

Watson Assistant Service provides several options to manage Conditions, and possibility to have several answers to make your bot more human.

In this lab, you will learn:

- how to manage complex conditions
- how to manage multiple answers
- how to manage context variables

3. Prerequisites

Before you start the exercises in this guide, you will need to complete the following prerequisite tasks:

- Session 4 – building a conversation lab Instructions
- The instructor provided you the link to get labs content. You may download each file individually.

Reminder of IBM Cloud URLs per location:

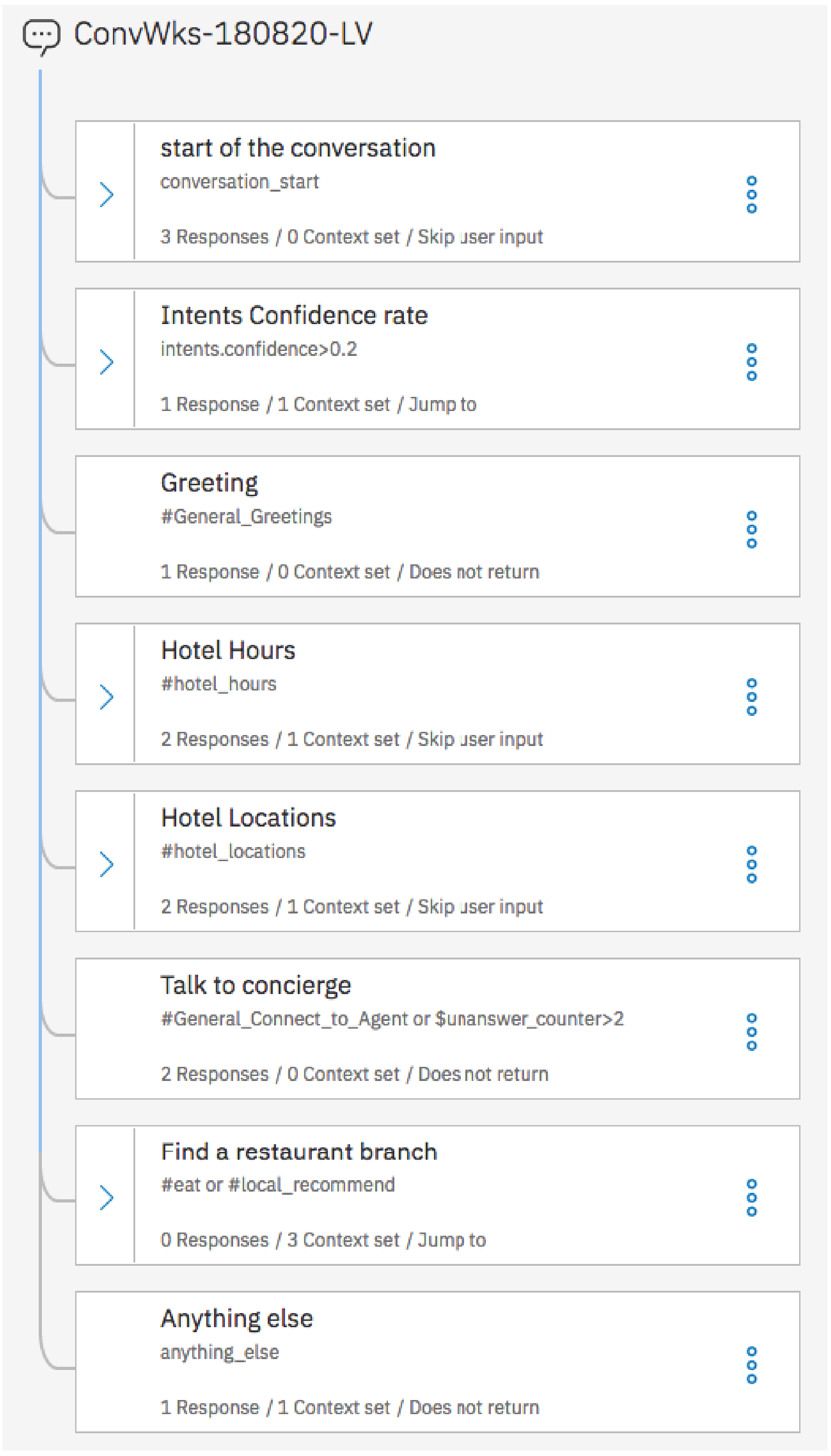
Location	URL
US	https://console.ng.bluemix.net/
UK	https://console.eu-gb.bluemix.net/
Sidney	https://console.au-syd.bluemix.net/
Germany	https://console.eu-de.bluemix.net/

4. Scenario

Use case:	A Hotel Concierge Virtual assistant that is accessed from the guest room and the hotel lobby.
End-users:	Hotel customers

5. What to expect when you are done

At the end of session, you should get a more complex dialog using several conditions and answers in the same node.



ConvWks-180820-LV

start of the conversation
conversation_start
3 Responses / 0 Context set / Skip user input

Intents Confidence rate
intents.confidence>0.2
1 Response / 1 Context set / Jump to

Greeting
#General_Greetings
1 Response / 0 Context set / Does not return

Hotel Hours
#hotel_hours
2 Responses / 1 Context set / Skip user input

Hotel Locations
#hotel_locations
2 Responses / 1 Context set / Skip user input

Talk to concierge
#General_Connect_to_Agent or $unanswer_counter>2
2 Responses / 0 Context set / Does not return

Find a restaurant branch
#eat or #local_recommend
0 Responses / 3 Context set / Jump to

Anything else
anything_else
1 Response / 1 Context set / Does not return

Building more natural answers

6. Update Greeting answers

In a response, you can perform any combination of these functions:

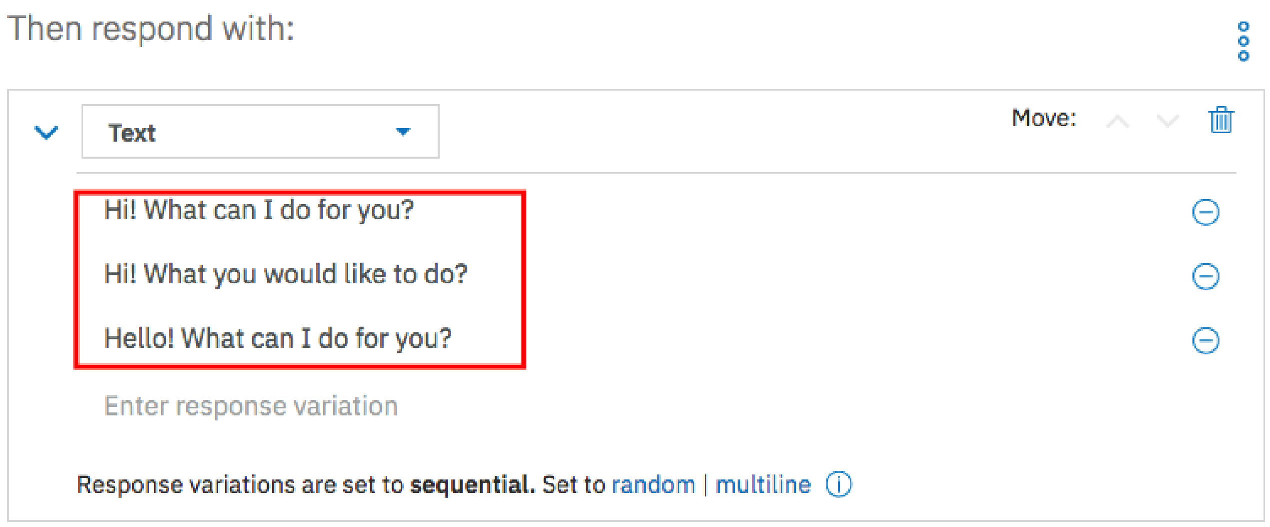
- update the dialog context
- provide a text response to the user
- change the flow of the dialog by using a Jump to action

The three options are always processed in this order, regardless of the order in which you specify them.

Right now, we just want to humanize the bots answer and make it more natural

1. Go back to your conversation service and go to the dialog page.
2. You are going to update the **Greeting** node and add several answers to make it less mechanic and more human. Select Greeting node.
3. Add the following responses:

Field	Value
Watson responses	*Hi! How may I help you?*
Watson responses	*Hi! What you would like to do?*
Watson responses	*Hello! What can I do for you?*


Then respond with:

Text

Hi! What can I do for you?

Hi! What you would like to do?

Hello! What can I do for you?

Enter response variation

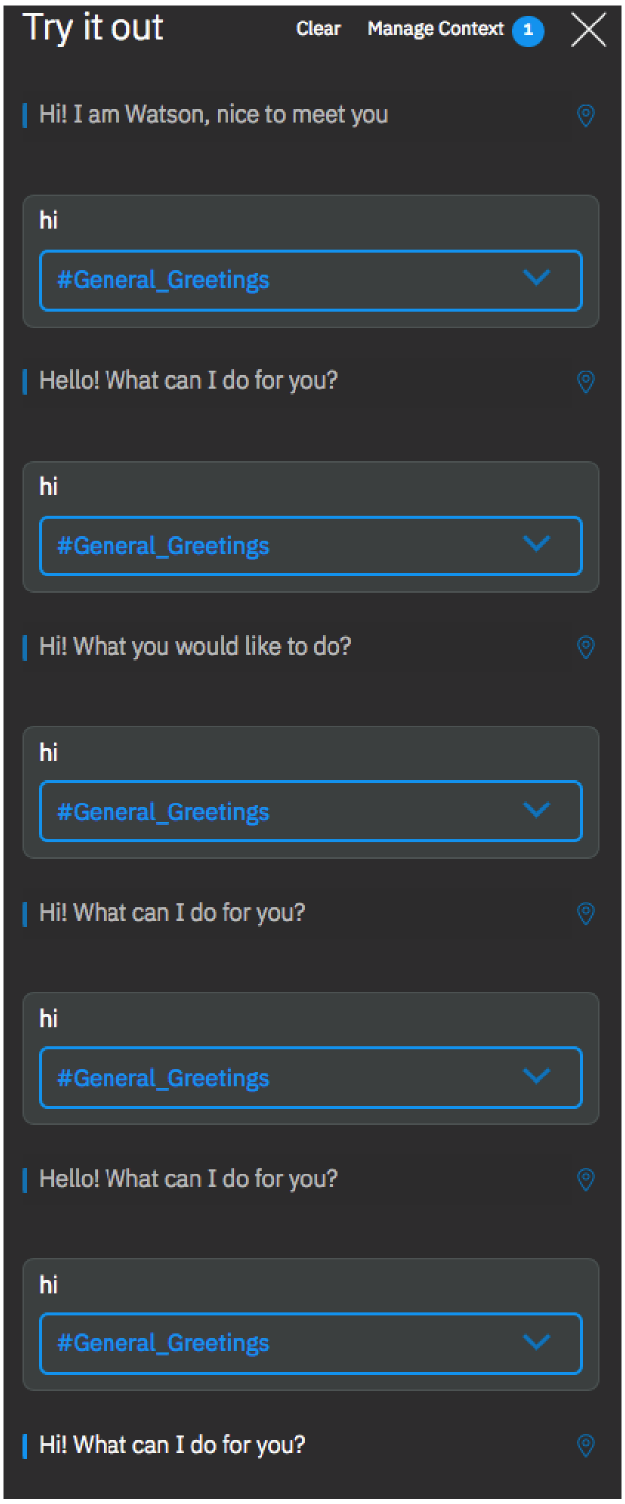
Response variations are set to **sequential.** Set to random | multiline

Now, you could decide if you want to return these responses consecutively in the defined sequence, by keeping the current setting displayed at the bottom of the response frame. We want to update it so the responses are picked in random order.

4. Click on **Set to random**, at the bottom of the response frame.

7. Test the sequence of the responses

1. Open **Try it out** panel
2. Enter *hi* several times and review the Watson's response.



Hi! I am Watson, nice to meet you

hi

#General_Greetings

Hello! What can I do for you?

hi

#General_Greetings

Hi! What you would like to do?

hi

#General_Greetings

Hi! What can I do for you?

hi

#General_Greetings

Hello! What can I do for you?

hi

#General_Greetings

Hi! What can I do for you?

Welcome statement enhancements

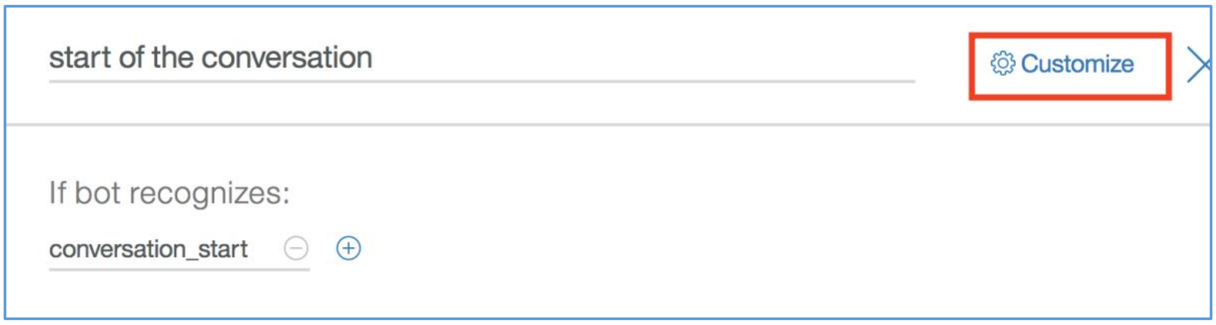
In the first step in the conversation we should:

- Engage the user to continue
- Include information about the role of the solution and its name (if needed)
- Consider using contextual information such as the time or user's name
- Initialize the context variable if required.

8. Add contextual details

1. Go back to your conversation service and go to the dialog page.

   You are going to update the Watson welcome according to the time of the day. You can do it for any other criteria which could make sense for you.
2. Select **start of the conversation** node which is the first node of your dialog
3. Click **Customize** at the top right of the box

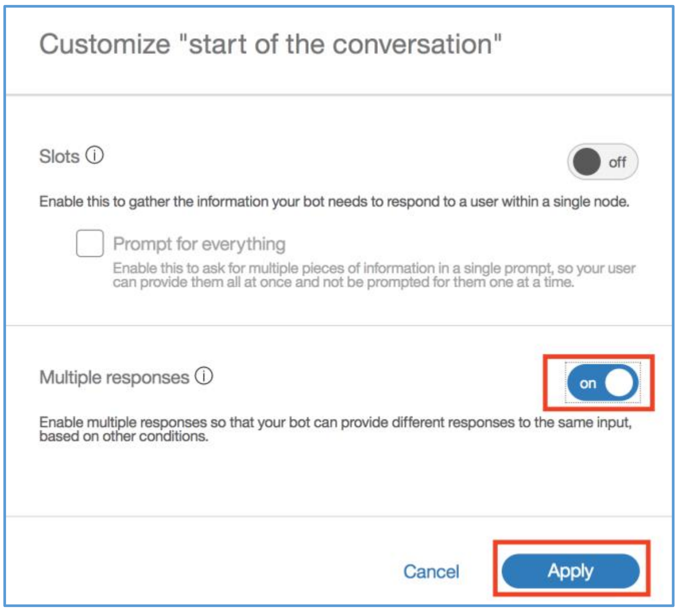


start of the conversation

If bot recognizes:

conversation_start

4. Then enable **Multiple responses**



Customize "start of the conversation"

Slots off

Enable this to gather the information your bot needs to respond to a user within a single node.

Prompt for everything

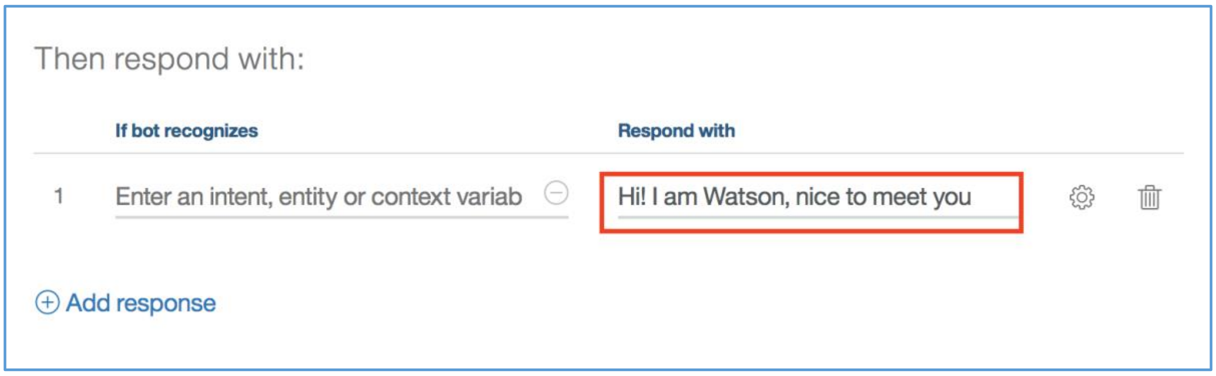
Enable this to ask for multiple pieces of information in a single prompt, so your user can provide them all at once and not be prompted for them one at a time.

Multiple responses on

Enable multiple responses so that your bot can provide different responses to the same input, based on other conditions.

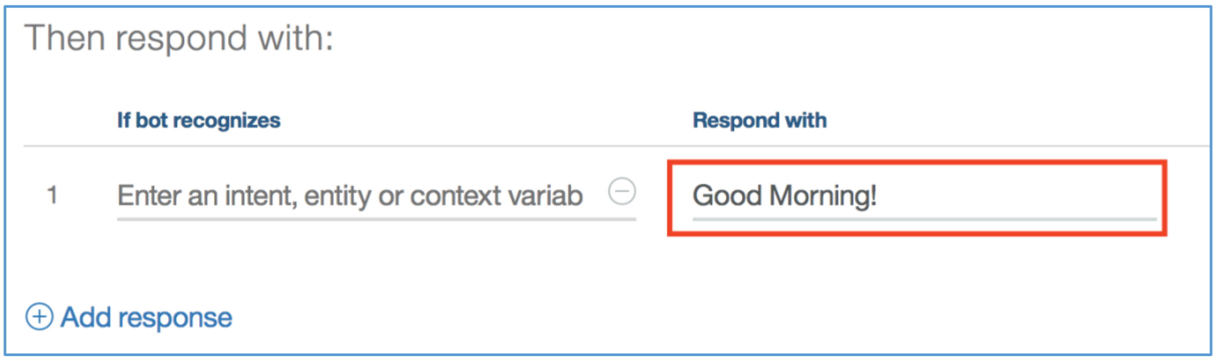
5. Click **Apply**

6. Replace the response Hi! I am Watson,… with *Good Morning!*



Then respond with:

	If bot recognizes	Respond with
1	Enter an intent, entity or context variab	Hi! I am Watson, nice to meet you

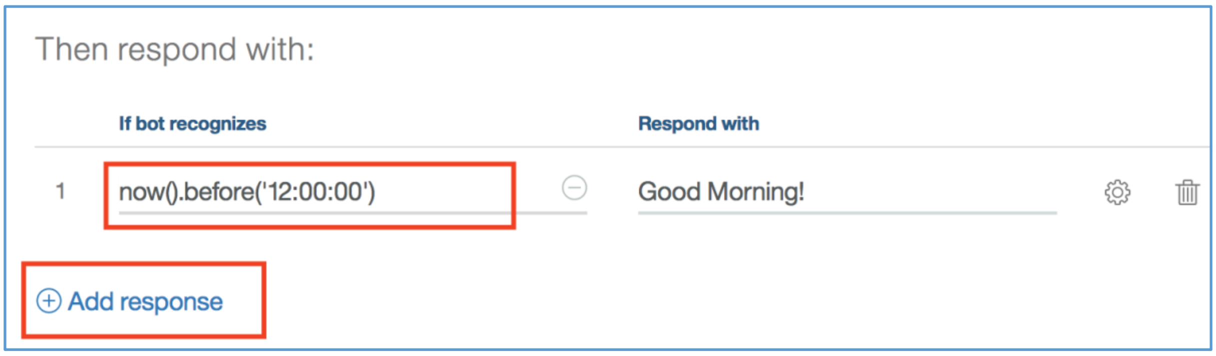


Then respond with:

	If bot recognizes	Respond with
1	Enter an intent, entity or context variab	Good Morning!

7. Click on **Enter an intent, entity or …**
8. Enter the following condition *now().before('12:00:00')*

You should get the result below. You are using the method 'before' to evaluate if the current time is before noon. That's why the answer is *Good Morning*.



Then respond with:

	If bot recognizes	Respond with
1	now().before('12:00:00')	Good Morning!

9. Click on **Add response**

10. Repeat the steps from 6 to 8 with :

    condition : *now().after('12:00:00')*

    response : *Good Afternoon!*

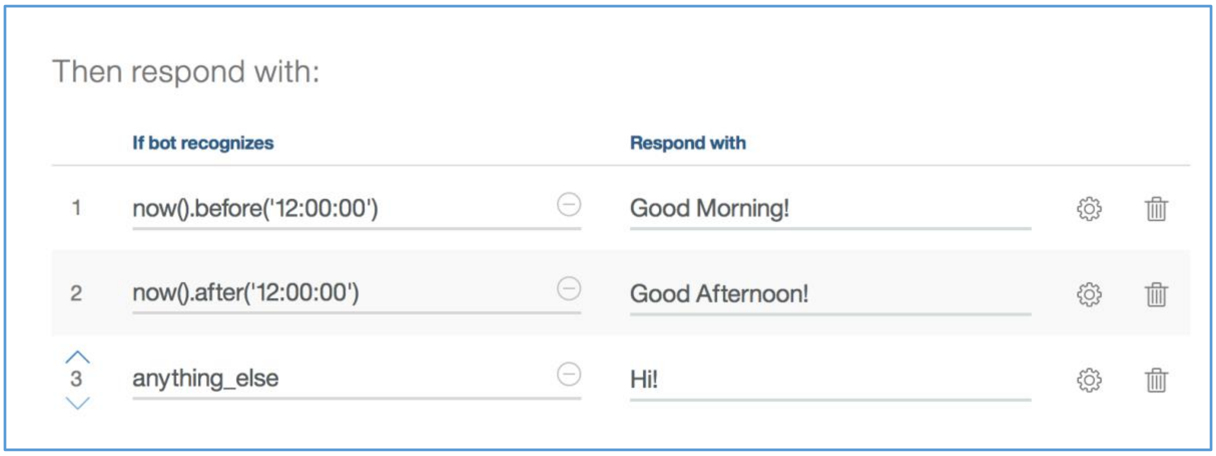
11. Click on **Add response**

12. Repeat the steps from 6 to 12 with :

    condition : *anything_else*

    response : *Hi!*

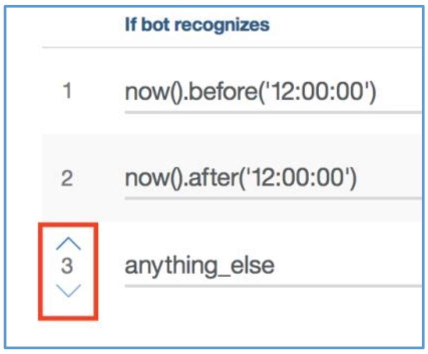
Then you should have:



Then respond with:

	If bot recognizes	Respond with
1	now().before('12:00:00')	Good Morning!
2	now().after('12:00:00')	Good Afternoon!
3	anything_else	Hi!

Note: if you need to move up or down a response, you can use the arrows at the left of each selected row.

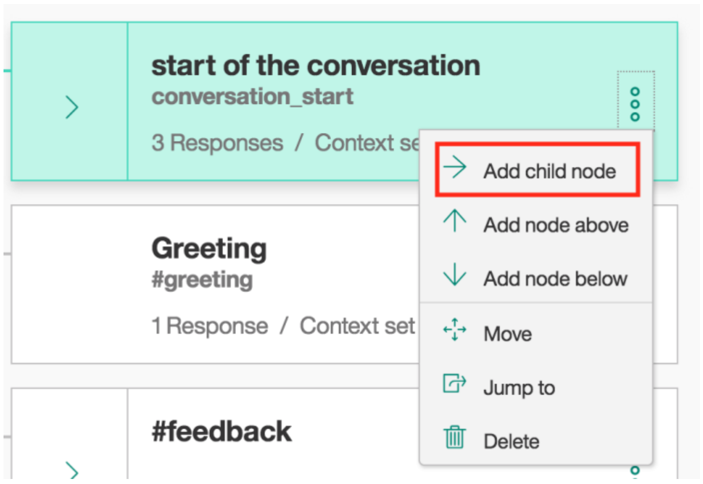


	If bot recognizes
1	now().before('12:00:00')
2	now().after('12:00:00')
3	anything_else

13. Close the **start of the conversation** editor pane.

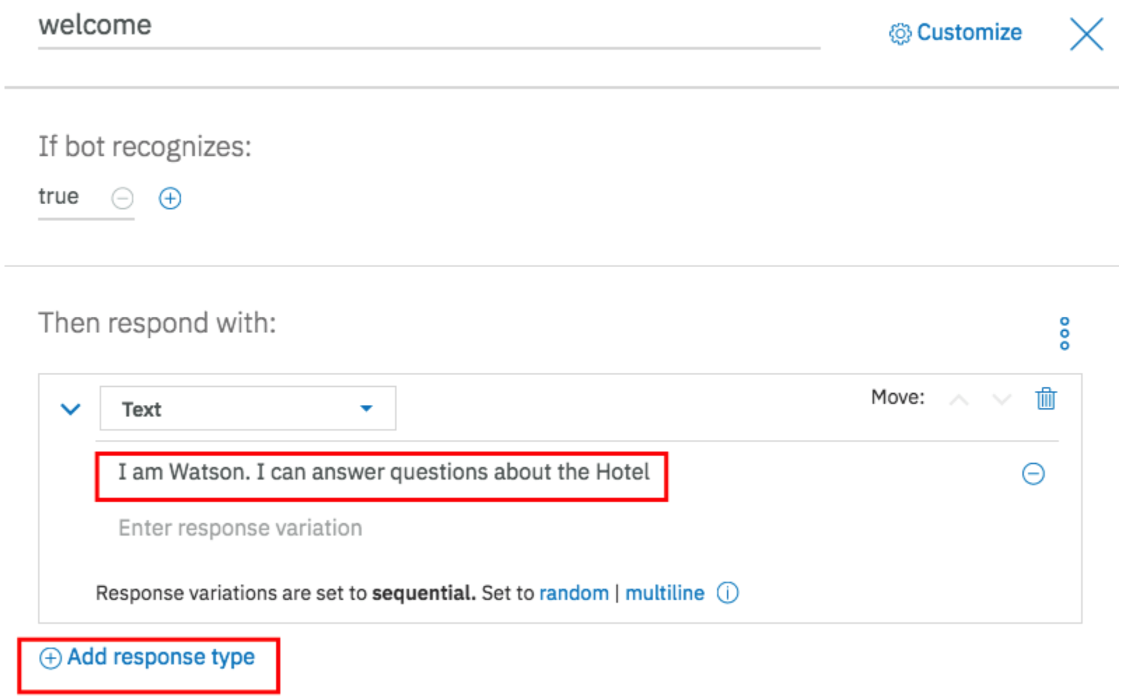
9. Add the welcome node

1. Select the **start of the conversation** node and add a child by clicking on the 3 dots icon and select **add child node**.



2. Fill the node as below

Field	Value
Name of the node	*welcome*
If bot recognizes	*true*
Watson responses	*I am Watson. I can answer questions about the Hotel.*


welcome

If bot recognizes:
true

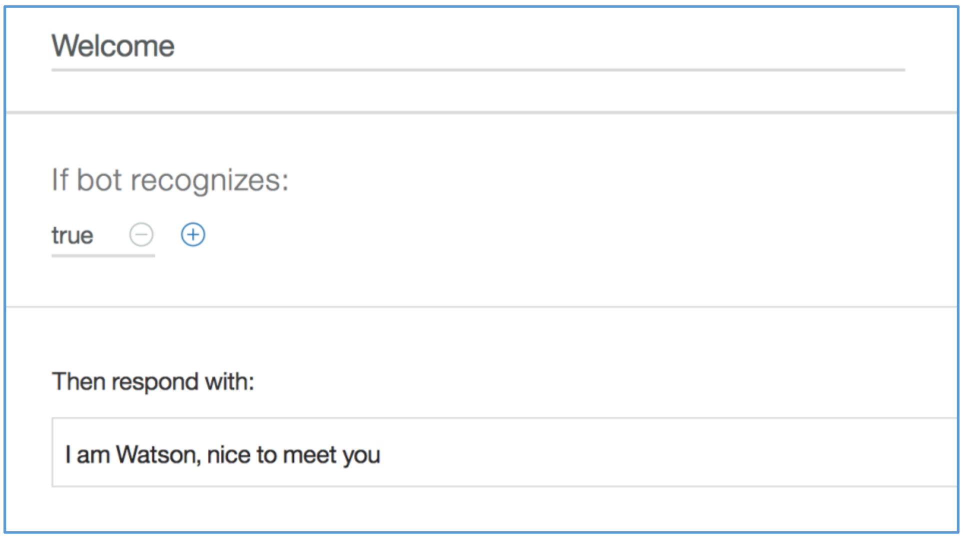
Then respond with:

Text

I am Watson. I can answer questions about the Hotel

Enter response variation

Response variations are set to **sequential.** Set to random | multiline


Welcome

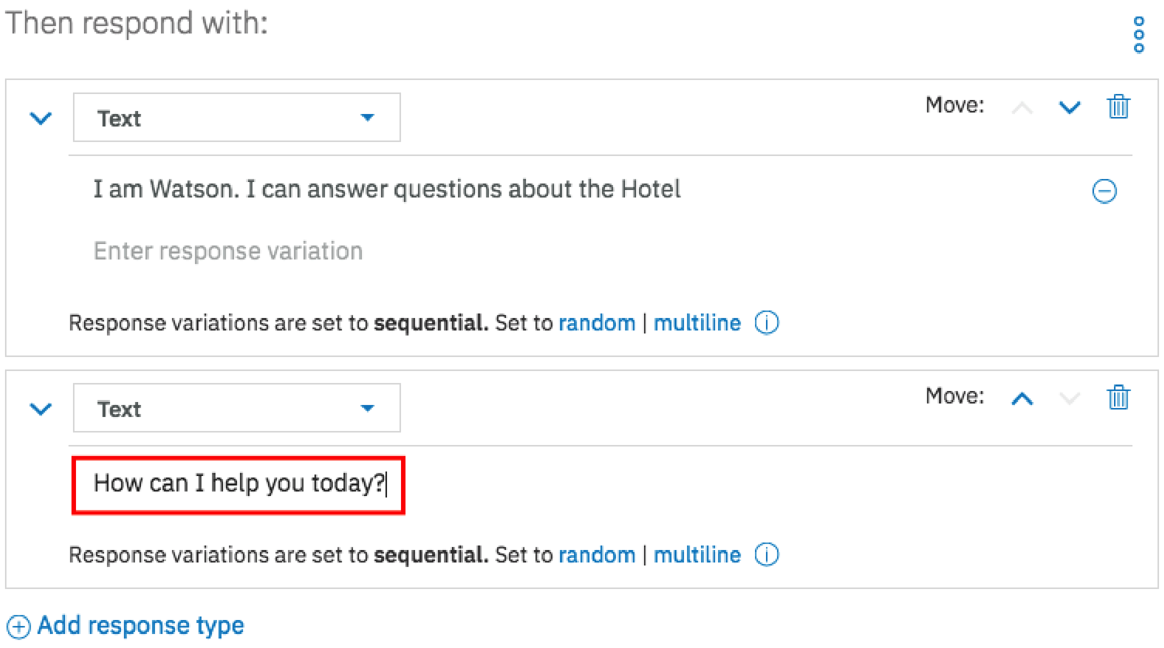
If bot recognizes:

true

Then respond with:

I am Watson, nice to meet you

3. Click **Add response type**, then select **Text** option
4. Fill the new response with node as below *How can I help you today?*


Then respond with:

Text

I am Watson. I can answer questions about the Hotel

Enter response variation

Response variations are set to **sequential.** Set to random | multiline

Text

How can I help you today?

Response variations are set to **sequential.** Set to random | multiline

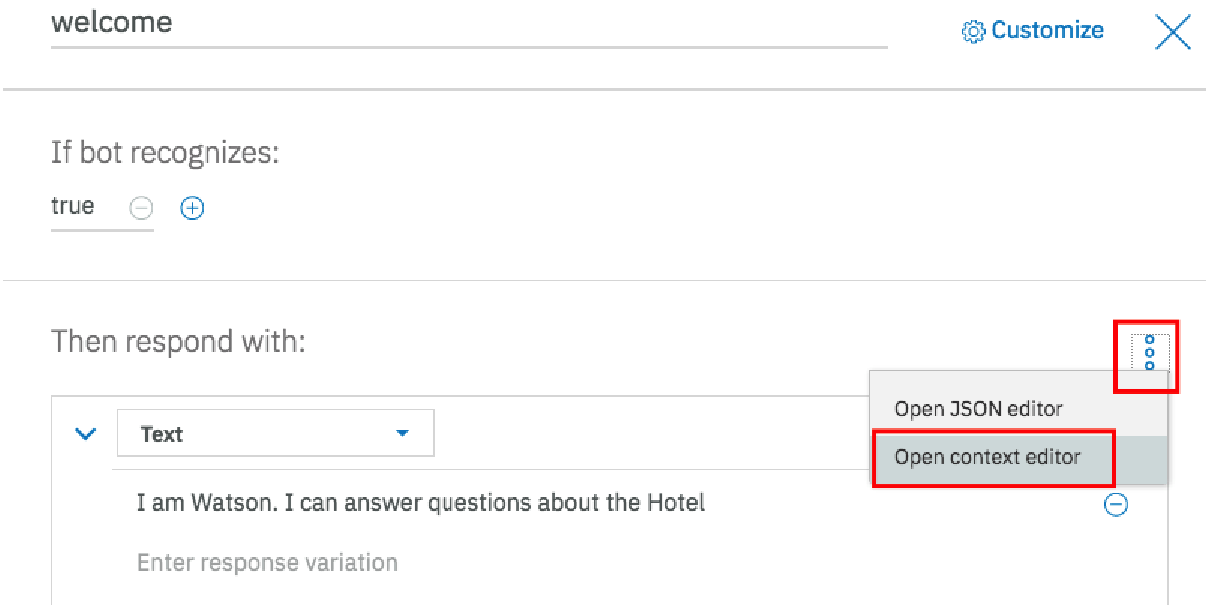
10. Context variable initialisation

We will initialize 3 context variables required for our conversation:

- *$private:{"mycredential":{"user":"<Function user ID>","password":"<Function Password>"}}* which is a placeholder for the credential required to leverage IBM Cloud function from the dialog. In the real implementation the credential must be managed by the Client Application which orchestrate the conversation.
- *$private:{"location":{“city”:”Belgrade”,"latitude":"44.7866","longitude":"20.4489"}}* which is the location of your hotel;
- *$unanswer_counter:0* which will be used to count the number of iteration without any correct answer to the end user. Once it will reach a threshold the bot should request a human action.

All context variable *private* won’t be stored in Watson Assistant logs.

1. Open the **context editor**



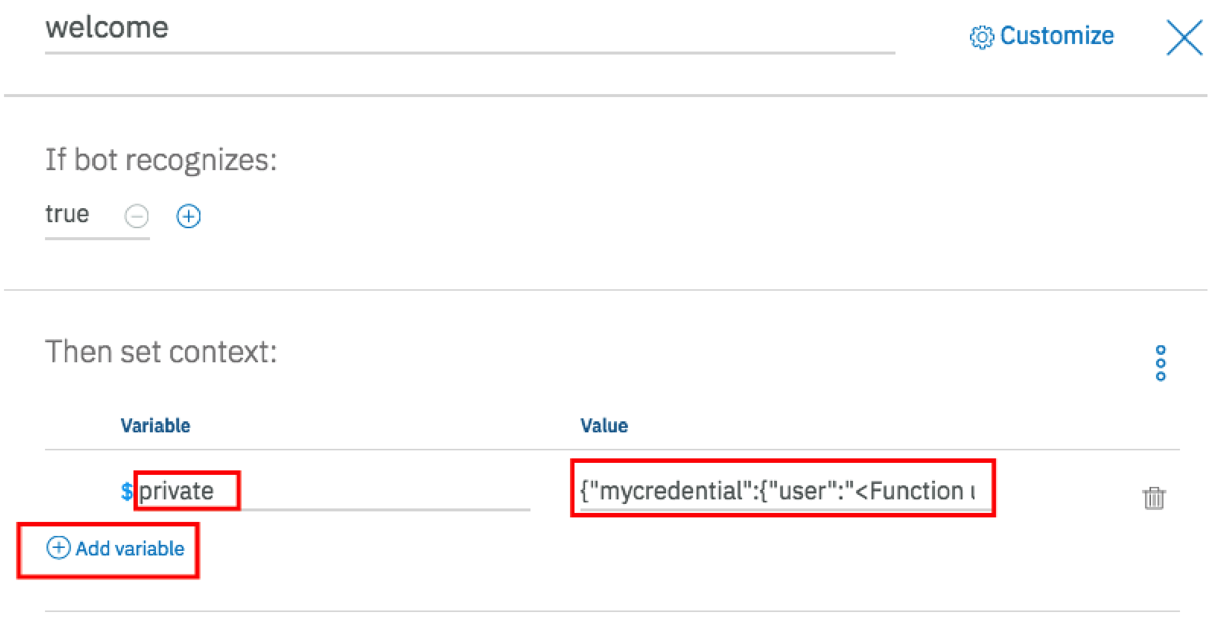
2. Fill it like this

   Context variable : *mycredential*

   Context value :

   {"mycredential":{"user":"<Function user ID>","password":"<Function Password>"},"location":{"city":"Belgrade","latitude":"44.7866","longitude":"20.4489"}}

   User and password must be replaced with your IBM Cloud Functions credentials



welcome

If bot recognizes:

true

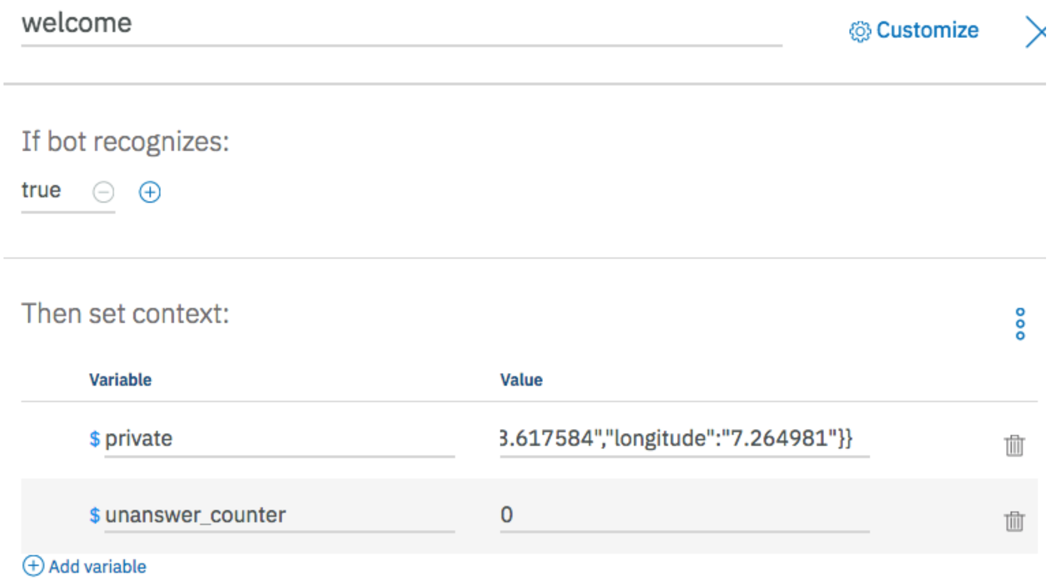
Then set context:

Variable	Value
$private	{"mycredential":{"user":"<Function ι

3. Click **Add variable** (at the bottom)
4. Fill it like this

   Context variable : *unanswer_counter*

   Context value : *0*



welcome

If bot recognizes:

true

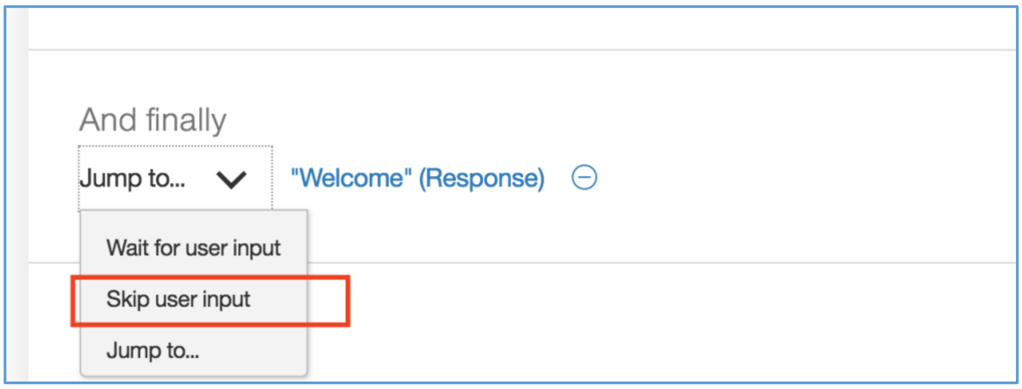
Then set context:

Variable	Value
$private	3.617584","longitude":"7.264981"}}
$unanswer_counter	0

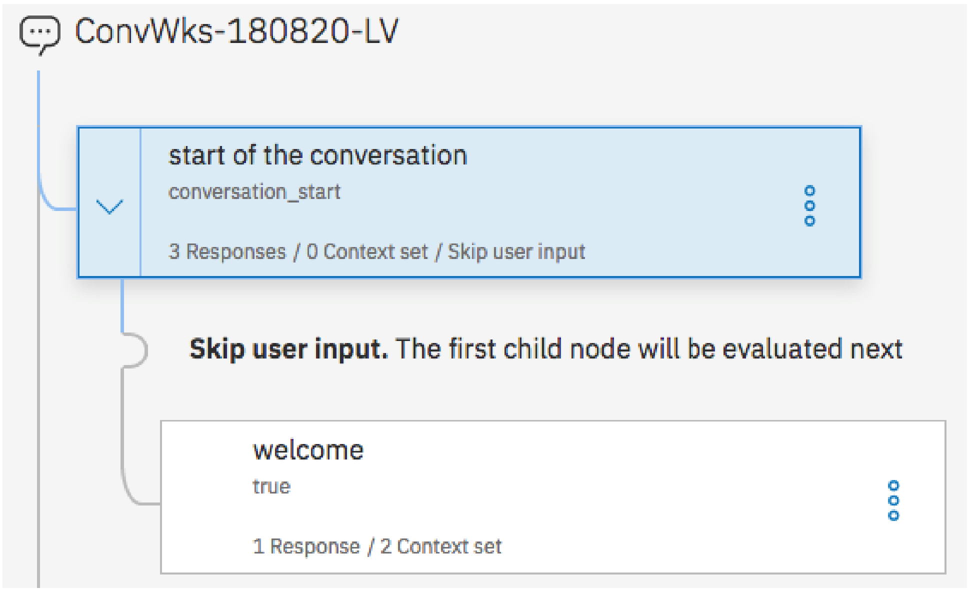
11. First “skip user input”

As this stage, if you are using the try it out, the welcome statement will be only Good afternoon. The bot is waiting a user’s input to execute the child node. As we want to execute this node without waiting, we can use Two options: **Jump to** capability or **Skip the user input** capability. The second should be the more appropriate.

1. Select the **start of the conversation** node and go back to the node editor
2. At the bottom of the page click **Wait for user input** drop box
3. Select **Skip user input** option



Then you should have the following configuration, the jump to link is replaced with the skip the user link :

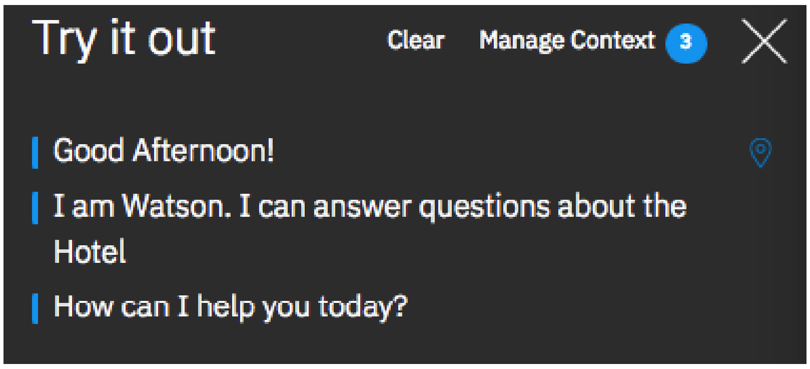

ConvWks-180820-LV

start of the conversation
conversation_start
3 Responses / 0 Context set / Skip user input

Skip user input. The first child node will be evaluated next

welcome
true
1 Response / 2 Context set

4. You are ready to test it. Open **Try it out** panel and Watson should provide you the following welcome message:

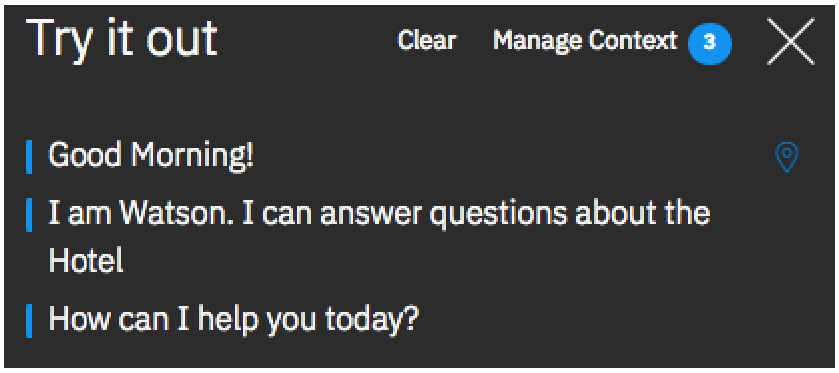


Good Afternoon!

I am Watson. I can answer questions about the Hotel

How can I help you today?

Or



Good Morning!

I am Watson. I can answer questions about the Hotel

How can I help you today?

Human takeover

12. Increment the unanswered counter

The **Anything else** node is only the overall fallback node, if it gets hit several times in a row, it may cause the users frustration. To avoid this, we are going to :

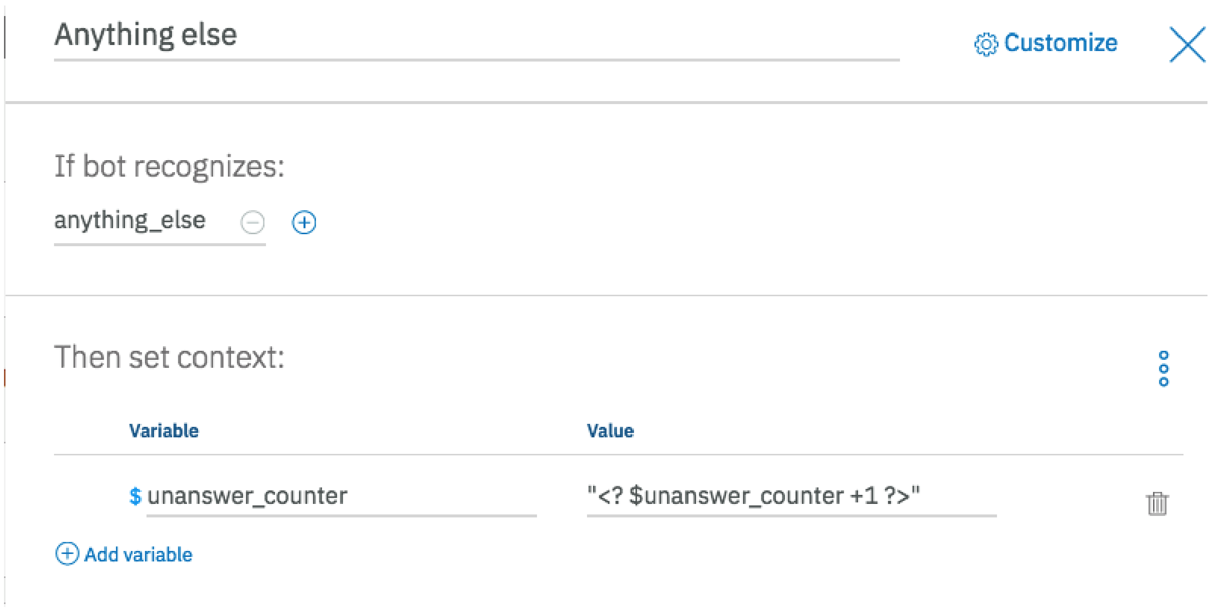
- Increment the unanswer_counter for all hits of Anything else node,
- leverage this counter in a new node to request a human agent action

We can consider 3 iterations without answer is our threshold.

1. Select **Anything else** node to open its editor
2. Open the **context editor**
3. Fill it like this

   Context variable : *unanswer_counter*

   Context value : *"<? $unanswer_counter +1 ?>"*



Anything else

If bot recognizes:

anything_else

Then set context:

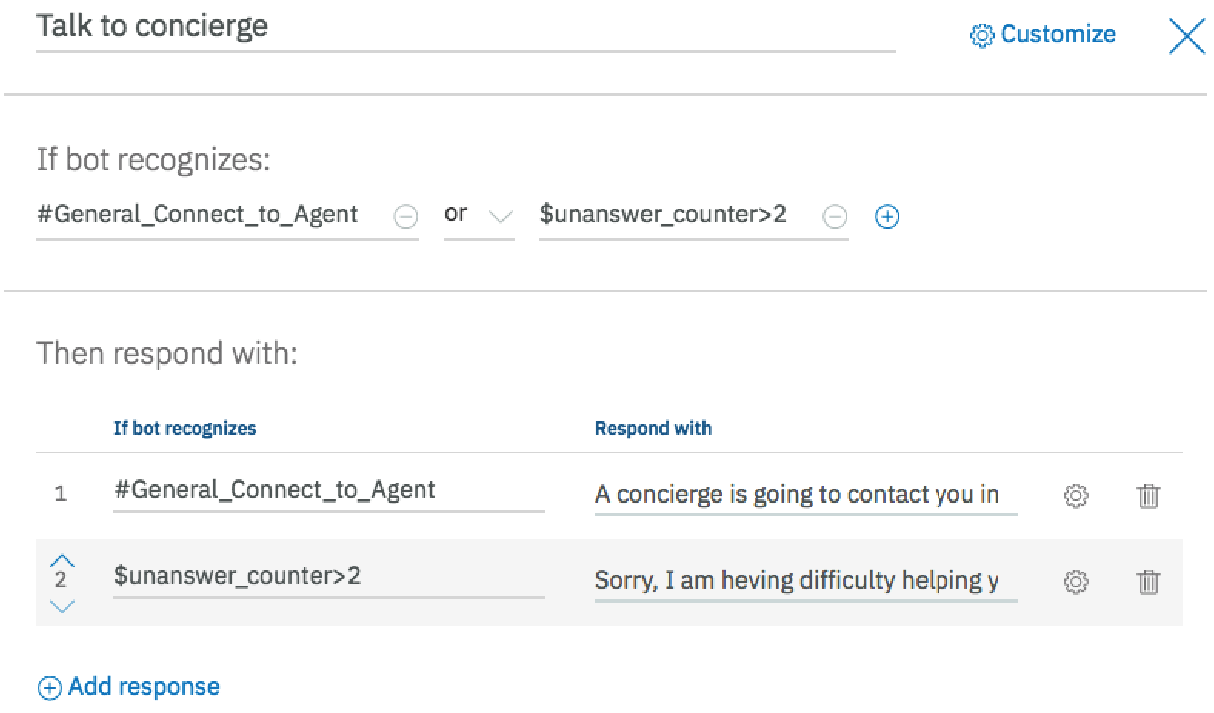
Variable	Value
$ unanswer_counter	"<? $unanswer_counter +1 ?>"

13. Talk to concierge node

1. Add a node above the **Anything else** node
2. To support multiple conditions, click on the **customize** button (top right):

3. Then activate **multiple responses** and click **Apply**:
4. Here are the parameters of your new node

Field	Value
Name of the node	*Talk to concierge*
If bot recognizes	*#General_Connect_to_Agent or $unanswer_counter>2*
Response 1	
Condition	*#General_Connect_to_Agent*
Responses	*A concierge is going to contact you in less than 2 minutes.*
Response 2	
Condition	*$unanswer_counter>2*
Responses	*Sorry, I am having difficulty helping you, a concierge is going to contact you in less than 2 minutes.*


Talk to concierge

Customize

If bot recognizes:

#General_Connect_to_Agent or $unanswer_counter>2

Then respond with:

	If bot recognizes	Respond with
1	#General_Connect_to_Agent	A concierge is going to contact you in
2	$unanswer_counter>2	Sorry, I am heving difficulty helping y

Add response

14. Does the Bot understand the user?

In our case, to manage the difficulty of providing the right answer, we want to contact a human agent, but there could be some useful actions which could help the user to go further.

In the example we have two reasons to contact somebody, a direct request of the user, identified with *#General_Connect_to_Agent*, and the impossibility to provide an answer managed with *$unanswer_counter*.

The counter is set to zero in the Welcome node and incremented in the Anything else node. Since we only want this to trigger with consecutive hits, we need to reset the counter when we do understand. So, we will add a single node just below the **start of the conversation** node that checks the confidence rate and resets if the confidence is above 20%.

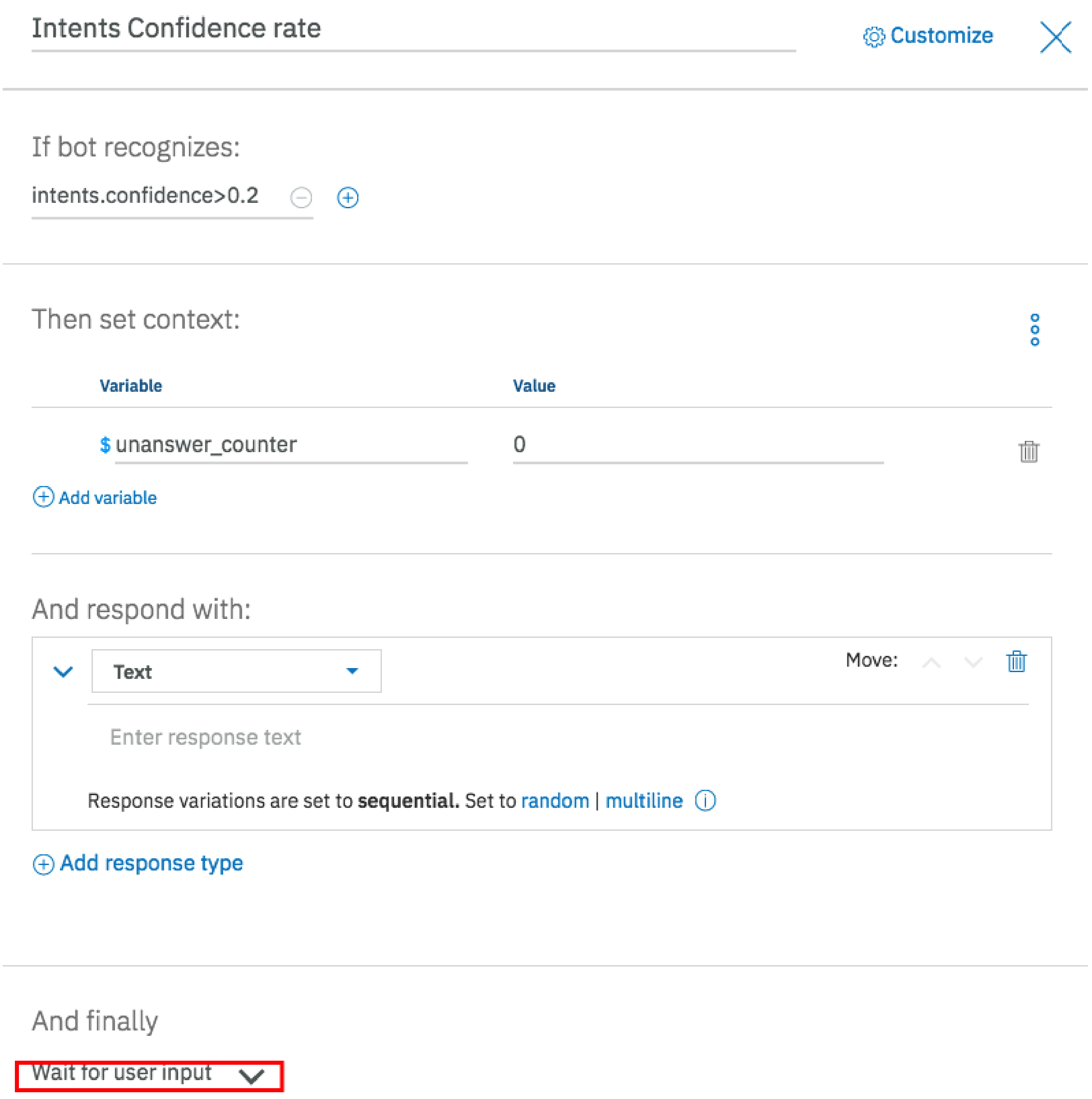
1. Add a node below the **start of the conversation** node
2. Fill the node as below

Field	Value
Name of the node	*Intents Confidence rate*
If bot recognizes	*Intents.confidence>0.2*
Watson responses	

4. Open the **context editor**
5. Fill it like this

   Context variable : *unanswer_counter*

   Context value : *0*


Intents Confidence rate

Customize

If bot recognizes:

intents.confidence>0.2

Then set context:

Variable	Value
$ unanswer_counter	0

Add variable

And respond with:

Text

Move:

Enter response text

Response variations are set to **sequential.** Set to random | multiline

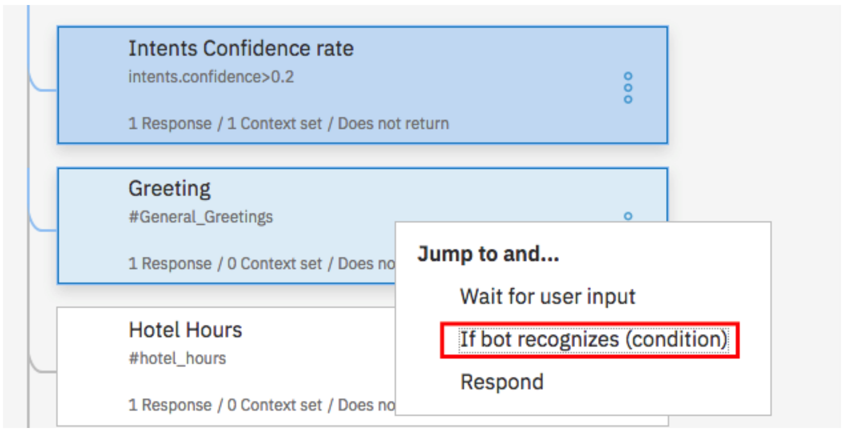
Add response type

And finally

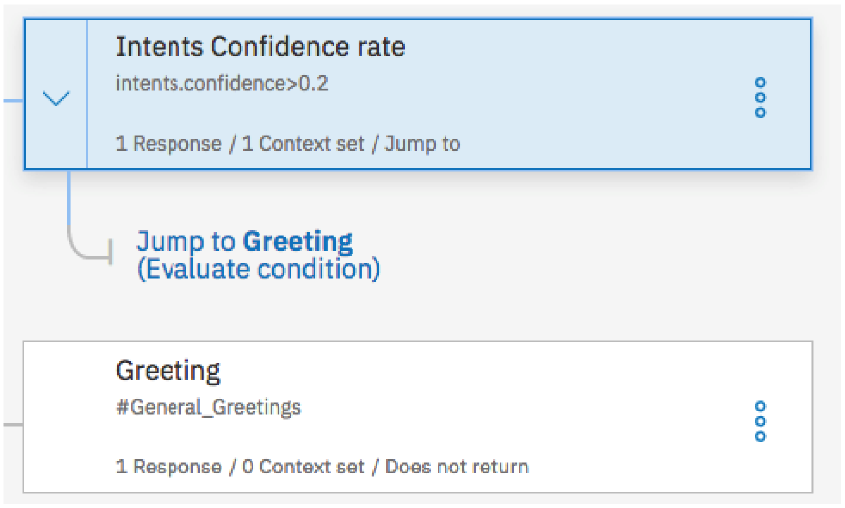
Wait for user input

As we don't expect any interaction with the user, the node must jump tp the next existing node **Greeting**.

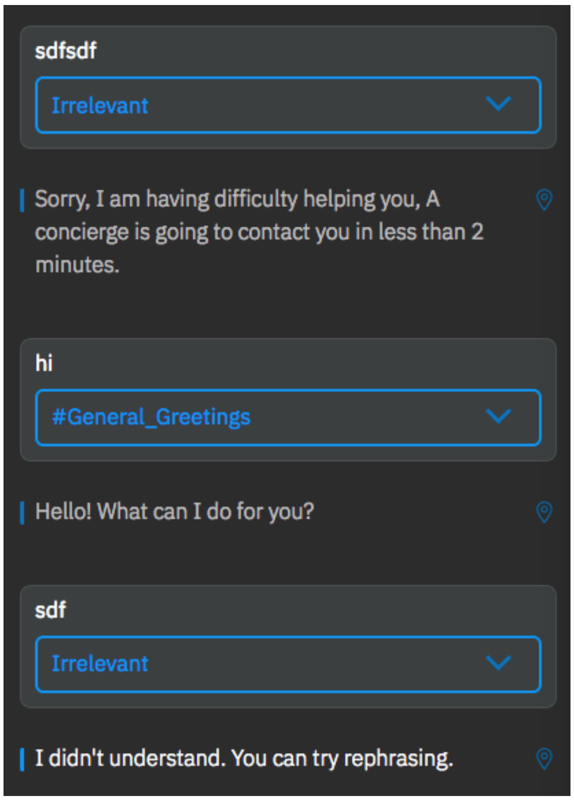
6. Click **Wait for user input** drop box
7. Select **Jump to** option
8. Select **Greeting** node


Intents Confidence rate
intents.confidence>0.2
1 Response / 1 Context set / Does not return
Greeting
#General_Greetings
1 Response / 0 Context set / Does no
Hotel Hours
#hotel_hours
1 Response / 0 Context set / Does no
Jump to and...
Wait for user input
If bot recognizes (condition)
Respond

9. Select **If bot recognizes** option



10. Go to **Try it out** panel to test the behaviour of your bot as shown below.



sdfsdf

Irrelevant

Sorry, I am having difficulty helping you, A concierge is going to contact you in less than 2 minutes.

hi

#General_Greetings

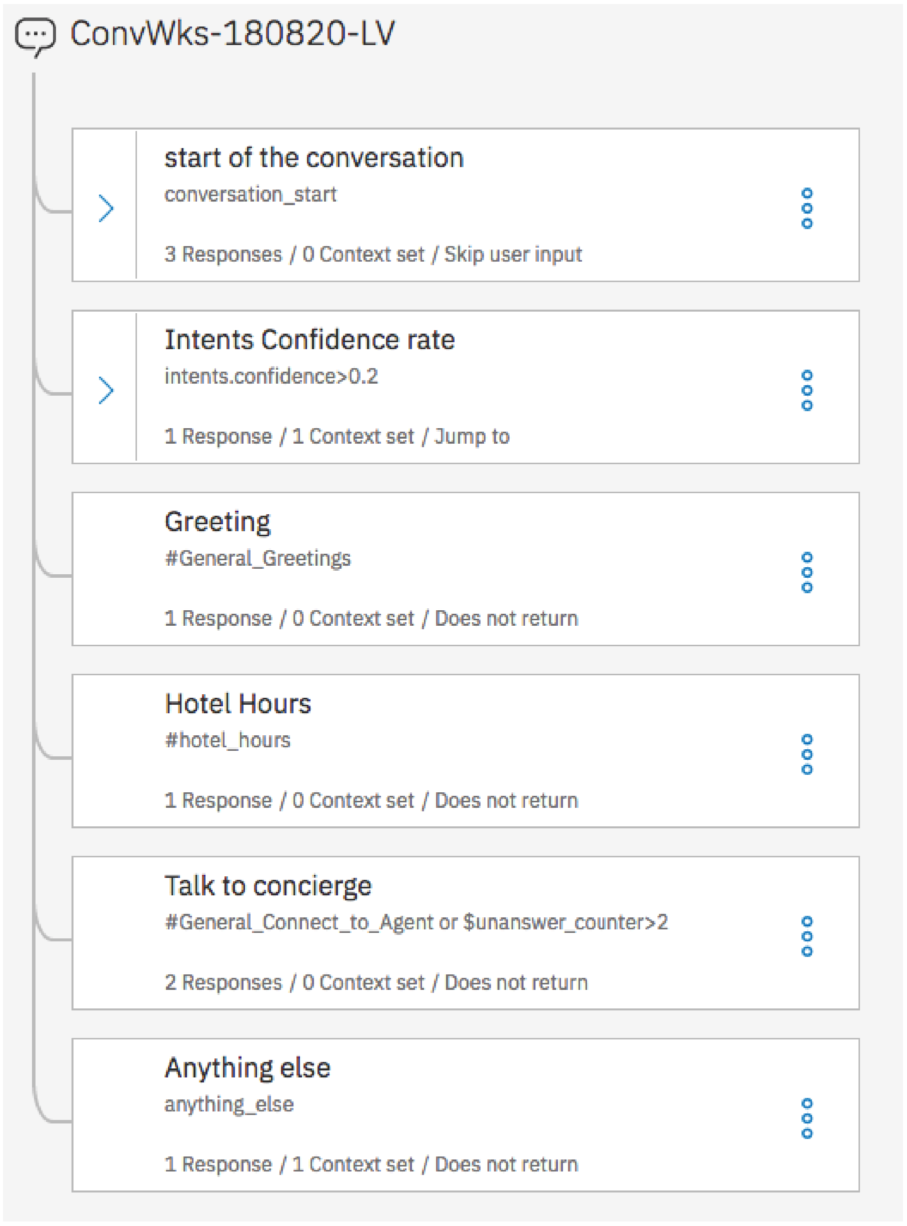
Hello! What can I do for you?

sdf

Irrelevant

I didn't understand. You can try rephrasing.

Your dialog should look like this


ConvWks-180820-LV

start of the conversation
conversation_start
3 Responses / 0 Context set / Skip user input

Intents Confidence rate
intents.confidence>0.2
1 Response / 1 Context set / Jump to

Greeting
#General_Greetings
1 Response / 0 Context set / Does not return

Hotel Hours
#hotel_hours
1 Response / 0 Context set / Does not return

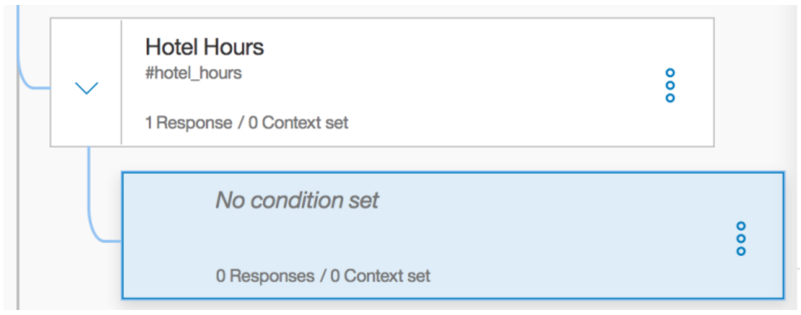
Talk to concierge
#General_Connect_to_Agent or $unanswer_counter>2
2 Responses / 0 Context set / Does not return

Anything else
anything_else
1 Response / 1 Context set / Does not return

15. One node and one response per condition

In this section, you will practice building child nodes using the Jump to function. If you have a difficult time working on your screen due to panel overlap, you can close your **Try it out** panel or increase your browser window size.

1. Click the **Hotel Hours** node to highlight it
2. Click **Add child node**



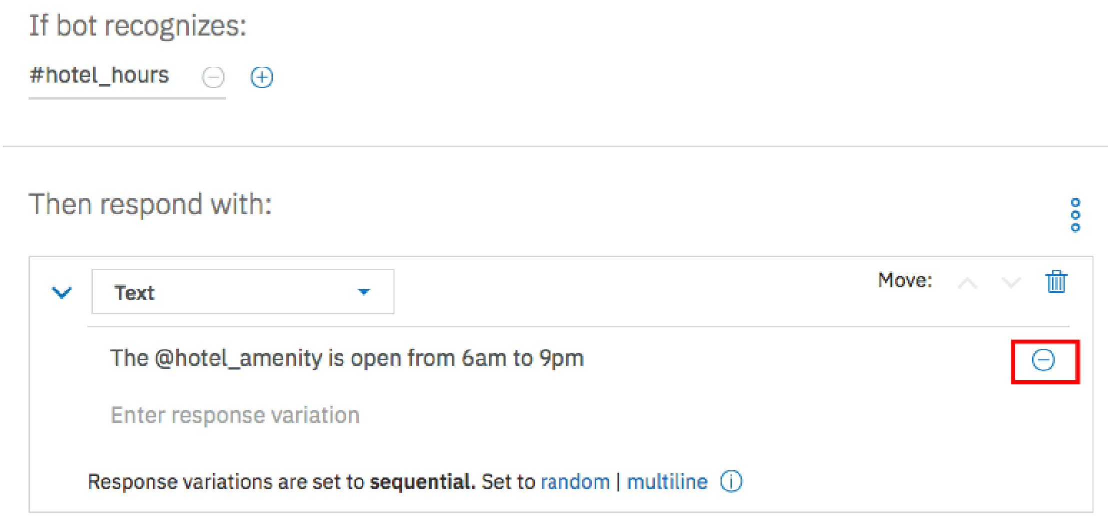
Hotel Hours
#hotel_hours
1 Response / 0 Context set

No condition set
0 Responses / 0 Context set

3. In the editor of the new node, fill it like this:

Field	Value
Name of the node	*Gym Hours*
If bot recognizes	*@hotel_amenity:gym*
Watson responses	*The @hotel_amenity.value is open from 5am to 11pm.*

4. Go back and click the **Hotel Hours** node to highlight it
5. Remove the response by clicking on the minus icon

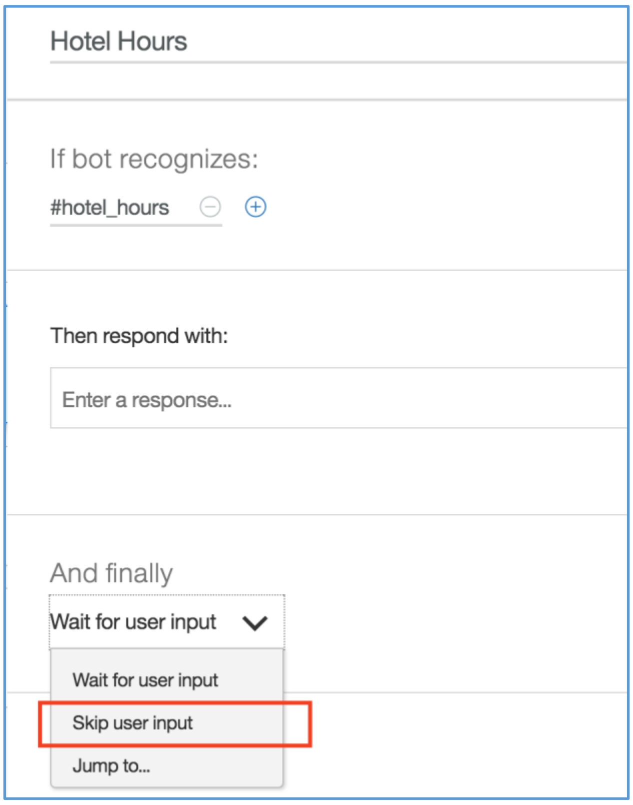


If bot recognizes:
#hotel_hours

Then respond with:

The @hotel_amenity is open from 6am to 9pm
Enter response variation
Response variations are set to **sequential.** Set to random | multiline

6. At the bottom of the page, click **Jump to** drop box and select **Skip user input** option



Hotel Hours

If bot recognizes:

#hotel_hours

Then respond with:

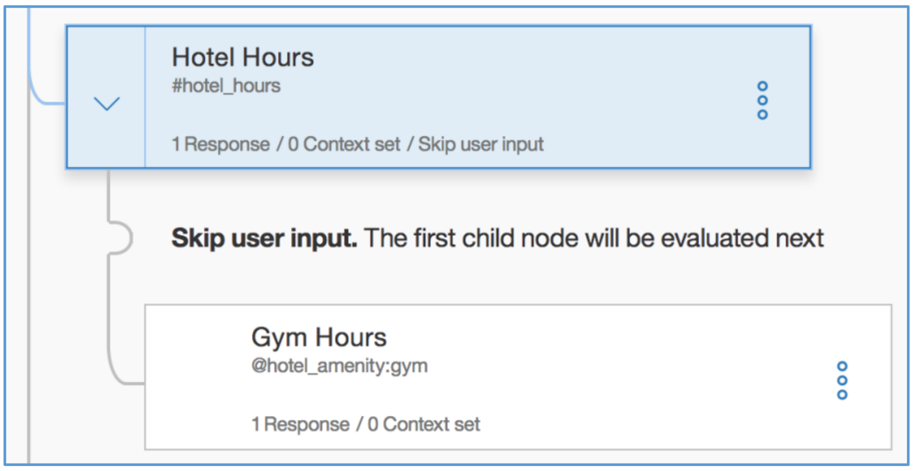
Enter a response...

And finally

Wait for user input

- Wait for user input
- Skip user input
- Jump to...

Then you should get this:



Hotel Hours
#hotel_hours
1 Response / 0 Context set / Skip user input

Skip user input. The first child node will be evaluated next

Gym Hours
@hotel_amenity:gym
1 Response / 0 Context set

7. Repeat the step 1 to 3 to add 4 more children, with the following inputs

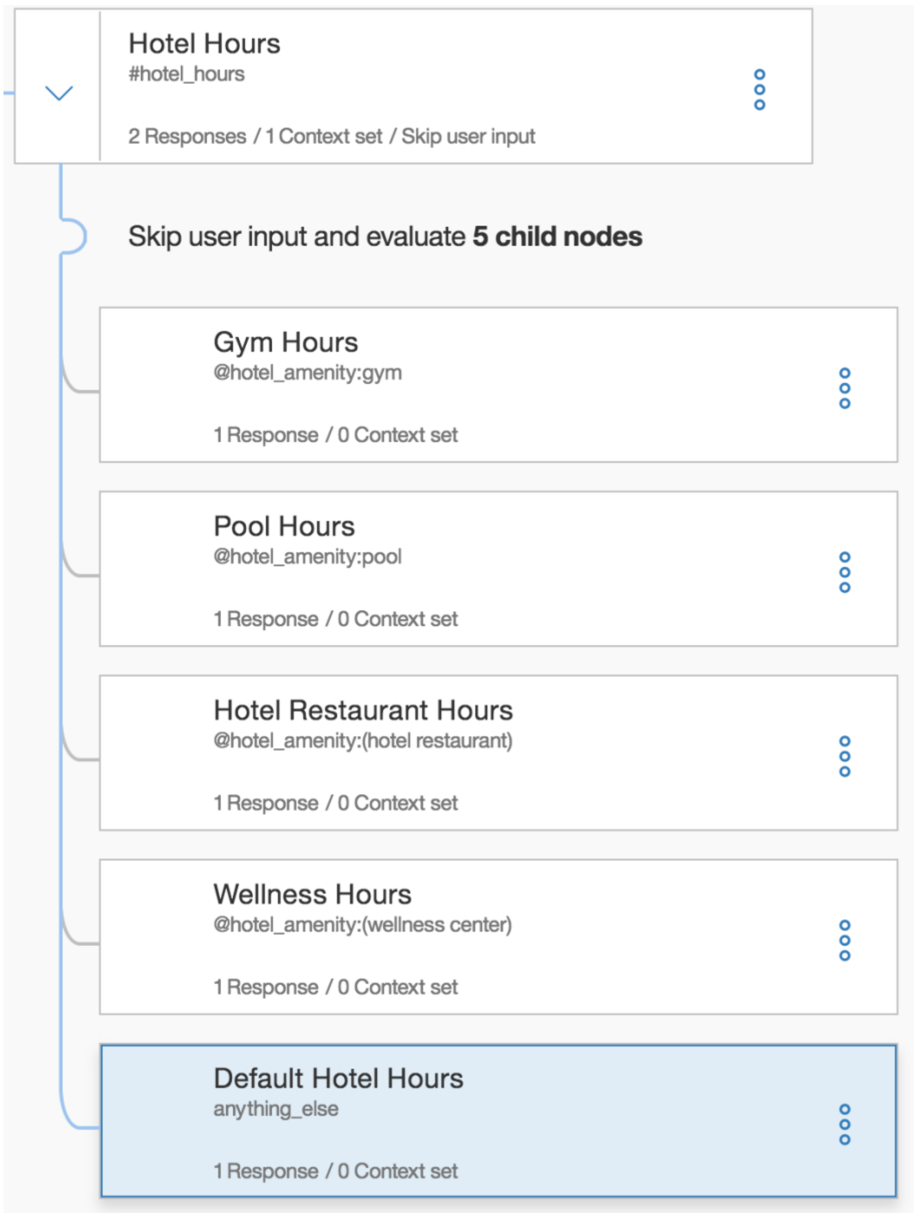
Field	**Value**
Name of the node	*Pool Hours*
If bot recognizes	*@hotel_amenity:pool*
Watson responses	*The @hotel_amenity.value is open from 5am to 8pm.*

Field	**Value**
Name of the node	*Hotel Restaurant Hours*
If bot recognizes	*@hotel_amenity:(hotel restaurant)*
Watson responses	*The @hotel_amenity.value is open from 5am to 2pm and from 6pm to 1am.*

Field	**Value**
Name of the node	*Wellness Hours*
If bot recognizes	*@hotel_amenity:(wellness center)*
Watson responses	*The @hotel_amenity.value is open from 10am to 9pm.*

Field	**Value**
Name of the node	*Default Hotel Hours*
If bot recognizes	*anything_else*
Watson responses	*I understand that you would like to learn more about our hours of operation. You can ask me specifically what hours you are looking for (gym, restaurant, pool, sauna), or you may call the front desk for more information.*

When you are finished, your chat flow should look similar to this:



Hotel Hours
#hotel_hours
2 Responses / 1 Context set / Skip user input

Skip user input and evaluate **5 child nodes**

Gym Hours
@hotel_amenity:gym
1 Response / 0 Context set

Pool Hours
@hotel_amenity:pool
1 Response / 0 Context set

Hotel Restaurant Hours
@hotel_amenity:(hotel restaurant)
1 Response / 0 Context set

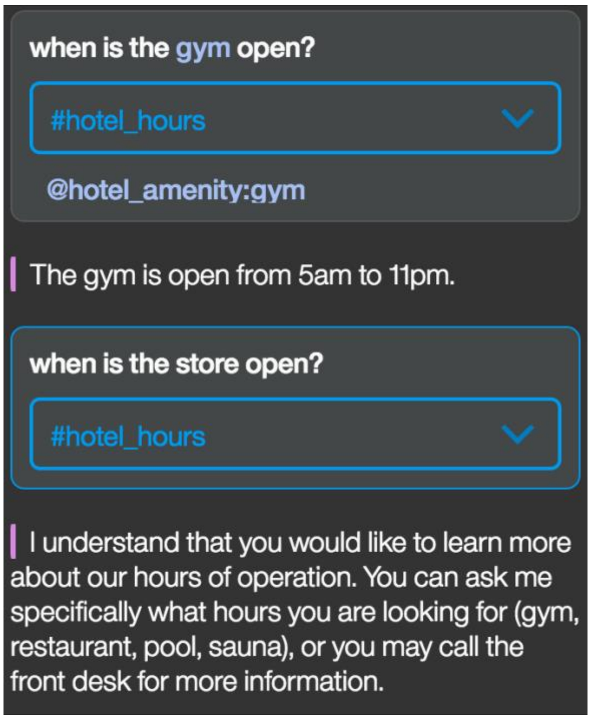
Wellness Hours
@hotel_amenity:(wellness center)
1 Response / 0 Context set

Default Hotel Hours
anything_else
1 Response / 0 Context set

8. Open the **Try it out** panel,
9. Type *when is the gym open?*

   Watson should return the response that is matched to *#hotel_hours* and *@hotel_amenity:gym*
10. Type *when is store open?*

    Watson should return the answer that you put in the **Default Hotel Hours** Response. This is because Watson was able to recognize that the input was related to asking for the hours that a location is open, but it did not recognize the *store* as an entity that was used as a condition in any of the child nodes



when is the gym open?

#hotel_hours

@hotel_amenity:gym

The gym is open from 5am to 11pm.

when is the store open?

#hotel_hours

I understand that you would like to learn more about our hours of operation. You can ask me specifically what hours you are looking for (gym, restaurant, pool, sauna), or you may call the front desk for more information.

16. Single node with multiple conditions / responses

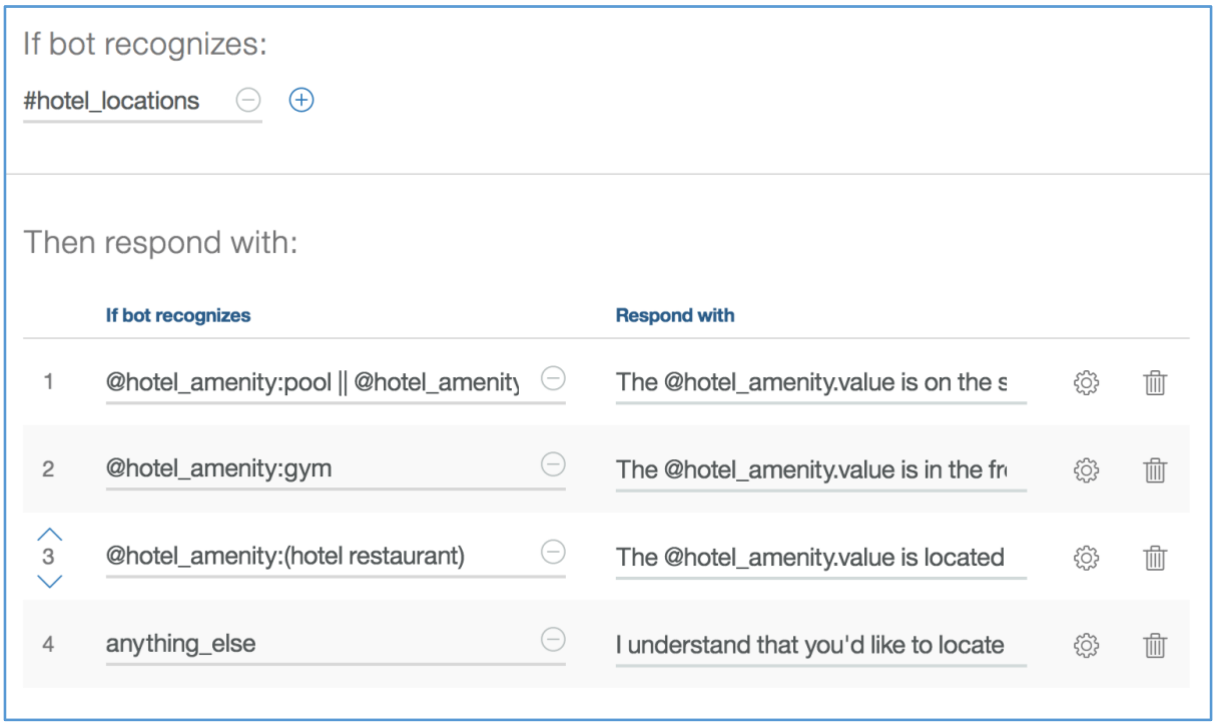
You have created nodes that have one response per condition. It is possible for a single dialog node to have multiple conditions that trigger a different response, called a multiple condition response. This approach enables you to simplify your dialog tree

As you created a such node for **start of conversation**, instead of giving you specific step-by-step instructions for this section, you will be given a list of conditions with the paired responses. There is a screen shot of the final dialog node after the instructions.

1. Create a new dialog node after the **Hotel Hours** node. (This should be a parent node in the main dialog tree)
2. To support multiple conditions, click on the customize button (top right):

3. Then activate **multiple responses** and click **Apply**:
4. Here are the parameters of your new node

Field	**Value**
Name of the node	*Hotel Locations*
If bot recognizes	*#hotel_locations*
Response 1	
Condition	*@hotel_amenity:pool or @hotel_amenity:(wellness center)*
Responses	*The @hotel_amenity.value is on the second floor.*
Response 2	
Condition	*@hotel_amenity:gym*
Responses	*The @hotel_amenity.value is in the front lobby.*
Response 3	
Condition	*@hotel_amenity:(hotel restaurant)*
Responses	*The @hotel_amenity.value is located on the ground floor of the hotel.*
Response 4	
Condition	*anything_else*
Responses	*I understand that you'd like to locate something in the hotel. Please ask me for the specific amenity you are looking for, (gym, restaurant, pool, sauna) or you may call the front desk for directions.*


If bot recognizes:

#hotel_locations

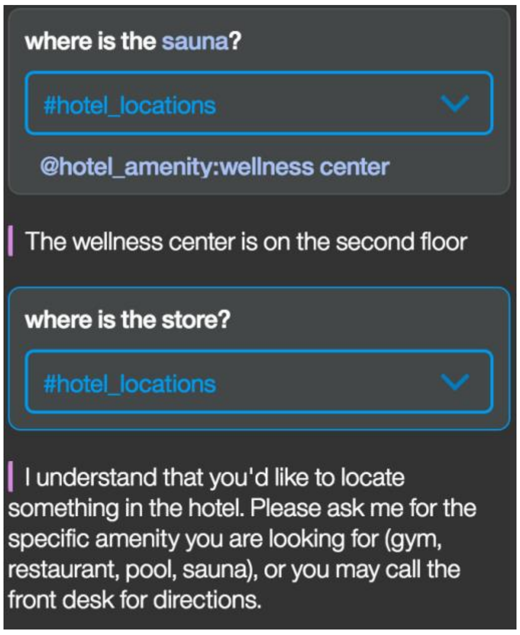
Then respond with:

	If bot recognizes	Respond with
1	@hotel_amenity:pool || @hotel_amenity	The @hotel_amenity.value is on the s
2	@hotel_amenity:gym	The @hotel_amenity.value is in the fr
3	@hotel_amenity:(hotel restaurant)	The @hotel_amenity.value is located
4	anything_else	I understand that you'd like to locate

5. Open the **Try it out** panel
6. Type *where is the sauna?*

   Watson should return *The wellness center is on the second floor*.
7. Type *where is the store?*

   Watson should return the answer that you put without any condition (the last one). This is because Watson was able to recognize that the input was related to asking for location, but it did not recognize the *store* as an entity that was used as a condition in your node


where is the sauna?

#hotel_locations

@hotel_amenity:wellness center

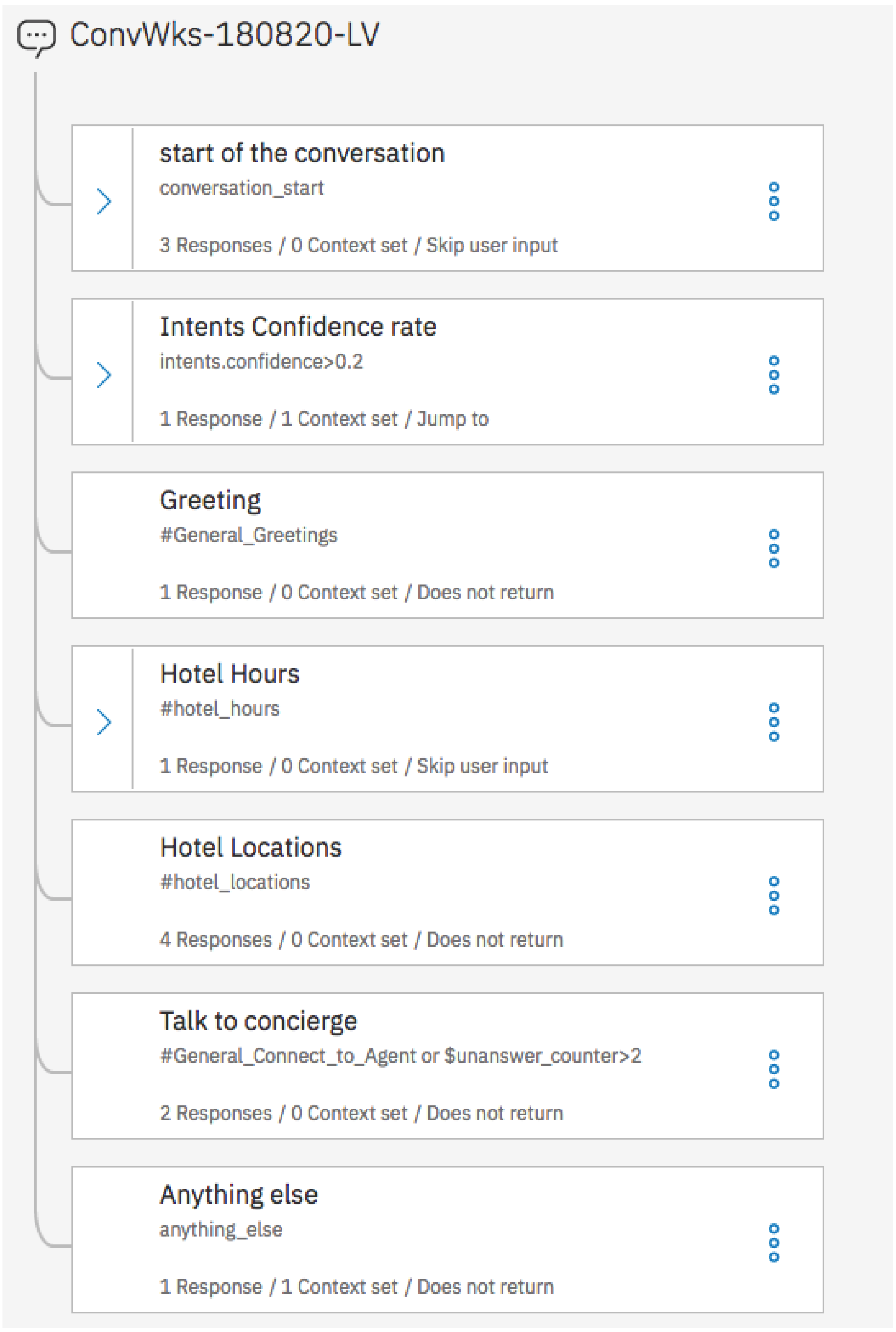
The wellness center is on the second floor

where is the store?

#hotel_locations

I understand that you'd like to locate something in the hotel. Please ask me for the specific amenity you are looking for (gym, restaurant, pool, sauna), or you may call the front desk for directions.

At this stage, your dialog looks like that:



ConvWks-180820-LV

start of the conversation
conversation_start
3 Responses / 0 Context set / Skip user input

Intents Confidence rate
intents.confidence>0.2
1 Response / 1 Context set / Jump to

Greeting
#General_Greetings
1 Response / 0 Context set / Does not return

Hotel Hours
#hotel_hours
1 Response / 0 Context set / Skip user input

Hotel Locations
#hotel_locations
4 Responses / 0 Context set / Does not return

Talk to concierge
#General_Connect_to_Agent or $unanswer_counter>2
2 Responses / 0 Context set / Does not return

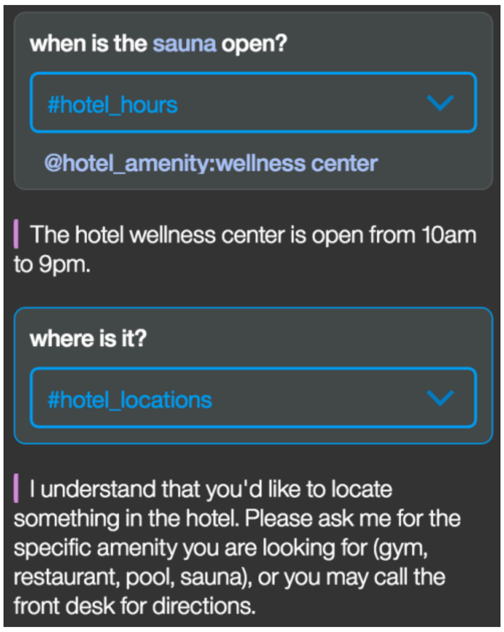
Anything else
anything_else
1 Response / 1 Context set / Does not return

17. How to keep the hotel amenity context? (Hotel_hours node)

1. Open the **Try it out** panel,
2. Type *when is the sauna open?*

   Watson should return *The wellness center is open from…*
3. Type *where is it?*

   In a natural conversation, you should expect that Watson keep in mind that the question is related to the sauna! However, Watson returns the default answer because the entities are not persistent during the conversation. To store information, you must use context variables. This is also the mechanism for passing information between WA and your application that submits the user input.



when is the sauna open?

#hotel_hours

@hotel_amenity:wellness center

The hotel wellness center is open from 10am to 9pm.

where is it?

#hotel_locations

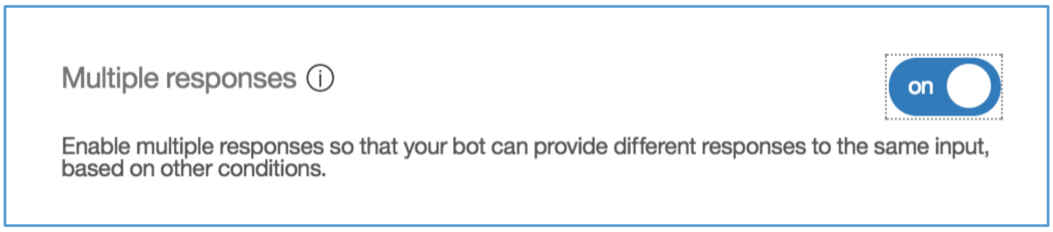
I understand that you'd like to locate something in the hotel. Please ask me for the specific amenity you are looking for (gym, restaurant, pool, sauna), or you may call the front desk for directions.

you are going to use context variable *$hotel_amenity*.

There are two methods to access it:

- Shorthand: *$hotel_amenity*
- Full syntax: *context.hotel_amenity*

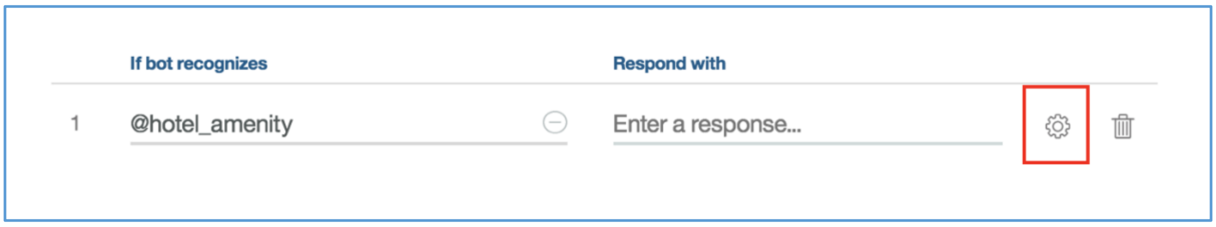
4. Click the **Hotel Hours** node to edit it and click **Customize** button, then enable the **multiple responses** and click **Apply**.



Multiple responses

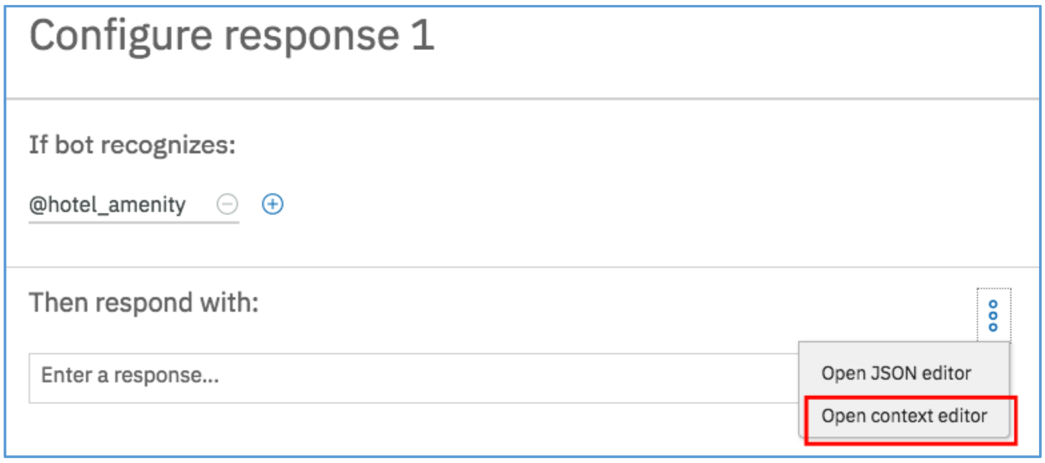
Enable multiple responses so that your bot can provide different responses to the same input, based on other conditions.

5. On the first row, enter *@hotel_amenity* as condition and click on the wheel on the right of condition line



	If bot recognizes	Respond with
1	@hotel_amenity	Enter a response...

6. A new window opens, now **open context editor** (Click on the three dots on the right of condition line)



Configure response 1

If bot recognizes:

@hotel_amenity

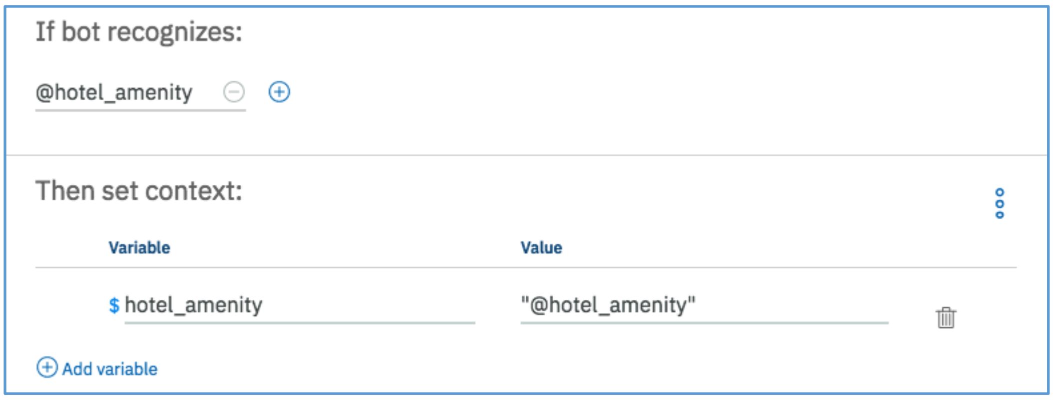
Then respond with:

Enter a response...

7. Fill the context like this

   Variable : *hotel_amenity*

   Value : *"@hotel_amenity"*



If bot recognizes:

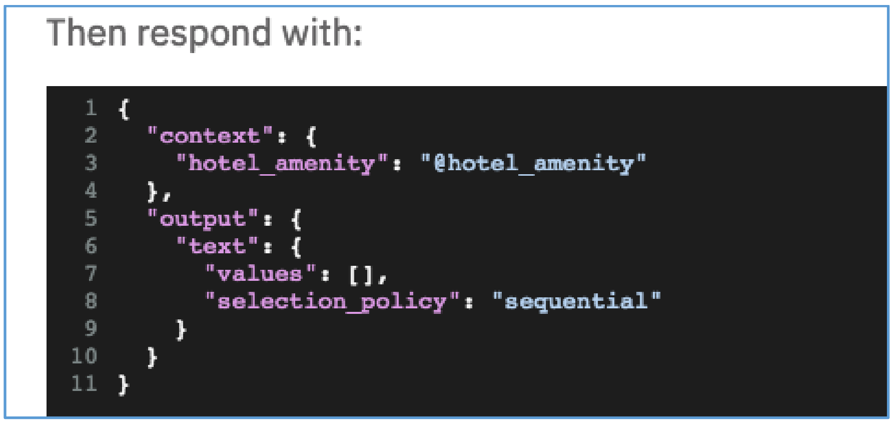
@hotel_amenity

Then set context:

Variable	Value
$ hotel_amenity	"@hotel_amenity"

That's the way to store the entity value of *@hotel_amenity* in the context variable *$hotel_amenity when the @hotel_amenity is captured by Watson.*

You can do the same thing by using the JSON editor. If you open JSON editor you should have:



Then respond with:

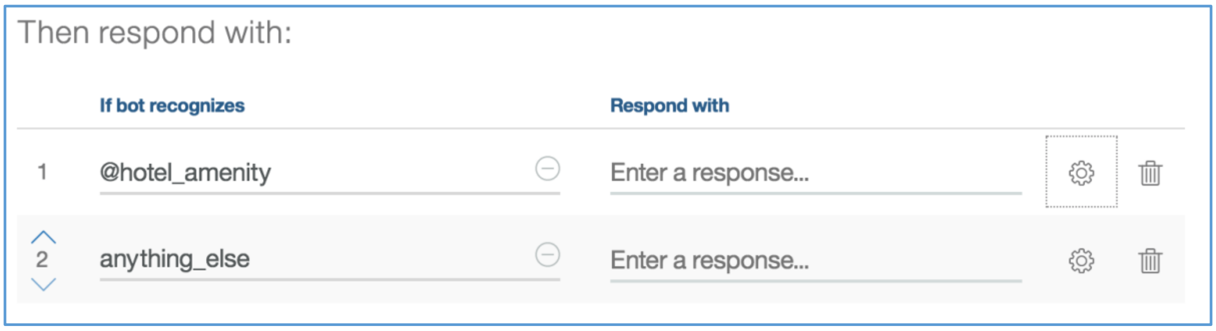
```
{
  "context": {
    "hotel_amenity": "@hotel_amenity"
  },
  "output": {
    "text": {
      "values": [],
      "selection_policy": "sequential"
    }
  }
}
```

8. Click **Save.**
9. Click **Add response** (bottom)

10. Fill the second row like this :

    condition : *anything_else*

    response :



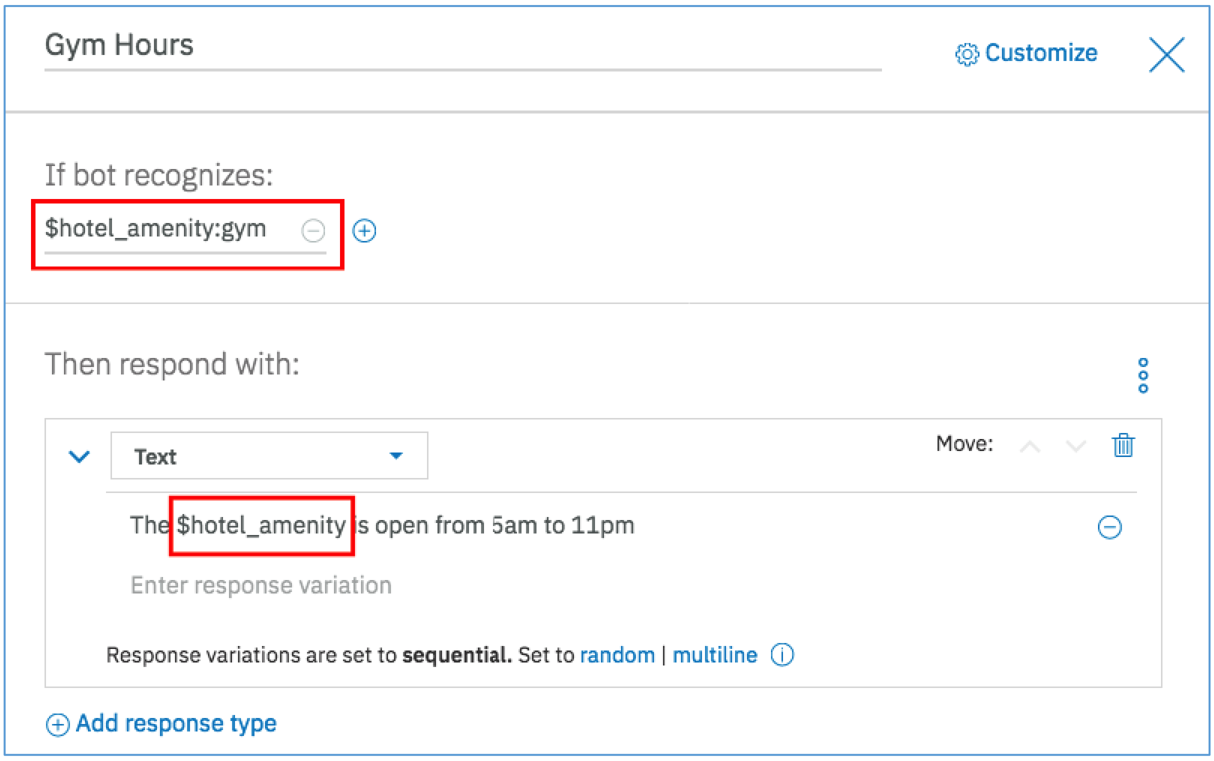
Then respond with:

	If bot recognizes	Respond with
1	@hotel_amenity	Enter a response...
2	anything_else	Enter a response...

11. In each child of **Hotel Hours** node, you must replace

@hotel_amenity	with	*$hotel_amenity*	as condition
@hotel_amenity.value	with	*$hotel_amenity*	in the response

For example with the first node (Gym Hours), you should get the following screen:



Gym Hours

If bot recognizes:

$hotel_amenity:gym

Then respond with:

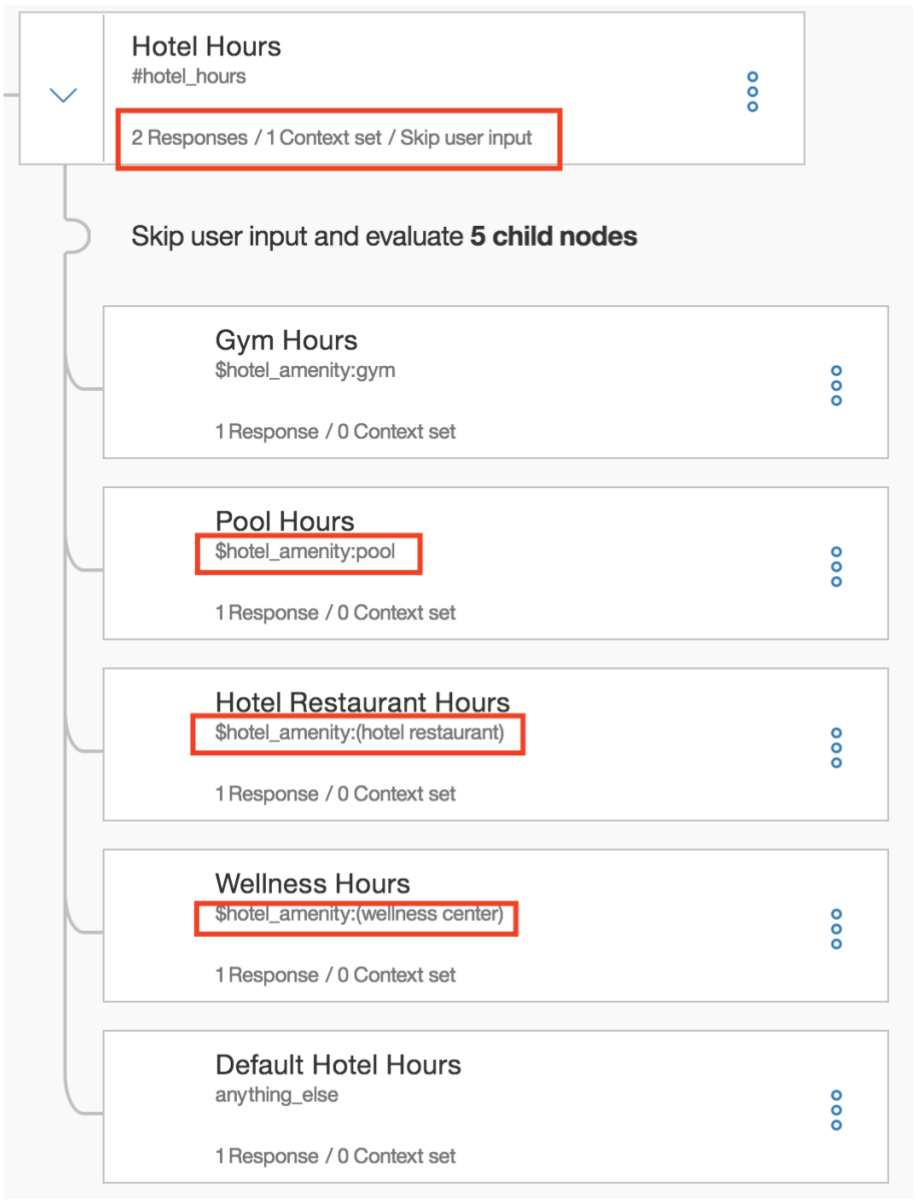
Text

The $hotel_amenity is open from 5am to 11pm

Enter response variation

Response variations are set to **sequential.** Set to random | multiline

Then you should get the dialog below



Hotel Hours
#hotel_hours
2 Responses / 1 Context set / Skip user input

Skip user input and evaluate **5 child nodes**

Gym Hours
$hotel_amenity:gym
1 Response / 0 Context set

Pool Hours
$hotel_amenity:pool
1 Response / 0 Context set

Hotel Restaurant Hours
$hotel_amenity:(hotel restaurant)
1 Response / 0 Context set

Wellness Hours
$hotel_amenity:(wellness center)
1 Response / 0 Context set

Default Hotel Hours
anything_else
1 Response / 0 Context set

The **Hotel Hours** node, now, uses 2 responses and sets one Context. You can test these updates, the behaviour for Watson doesn't change.

18. How to keep the hotel amenity context? (Hotel_locations node)

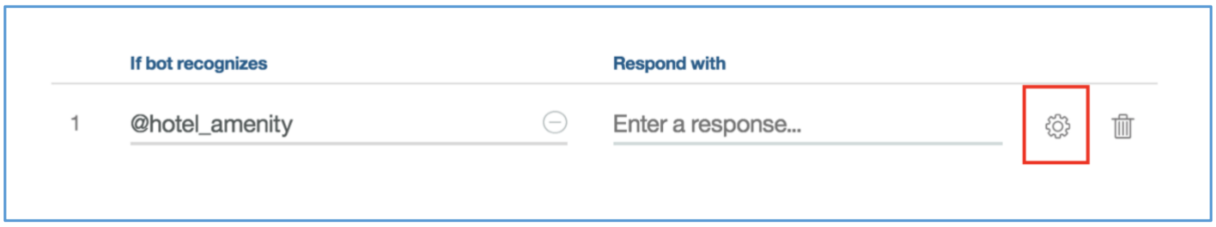
The next step is to update the node Hotel Locations to set the context variable *$hotel_amenity* and manage it correctly.

To set the context variable we duplicate what we did for **Hotel Hours** node.

1. Select **Hotel Locations** node to edit it
2. Update the name to use **Hotel Locations Management**
3. Select **Hotel Locations Management** node and **Add node above** (node menu)
4. Set the node like that: (don't forget to enable **Multiple response** option)

   name: *Hotel Locations*

   condition: *#hotel_locations*
5. On the first row, enter *@hotel_amenity* as condition and click on the wheel on the right of condition line



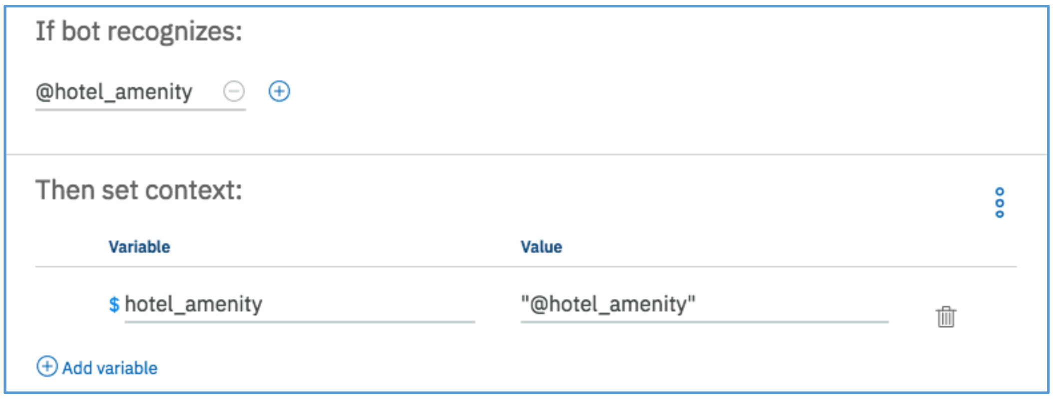
If bot recognizes | Respond with

1 @hotel_amenity | Enter a response...

6. A new window open, **open context editor** (Click on the three dots on the right of condition line)
7. Insert the context definition text below after the first parenthesis

   Variable : *hotel_amenity*

   Value : *"@hotel_amenity"*



If bot recognizes:

@hotel_amenity

Then set context:

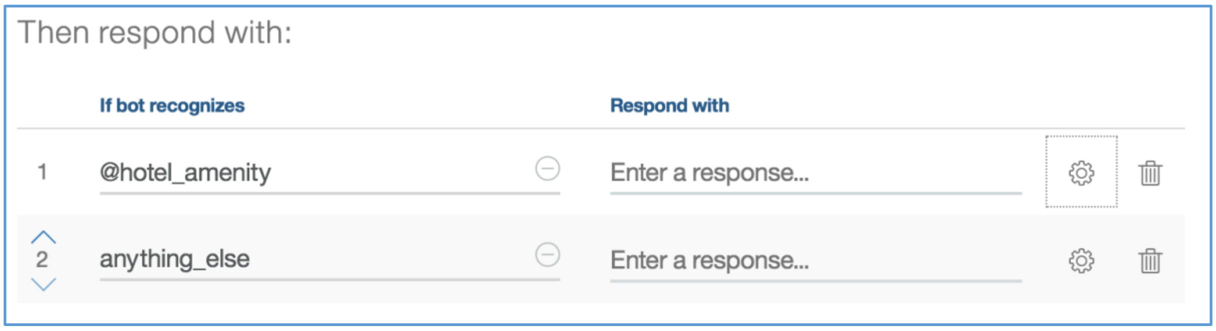
Variable	Value
$ hotel_amenity	"@hotel_amenity"

8. Click **Save**
9. Click **Add response** (bottom)

10. Fill the second row like this :

    condition : *anything_else*

    response :

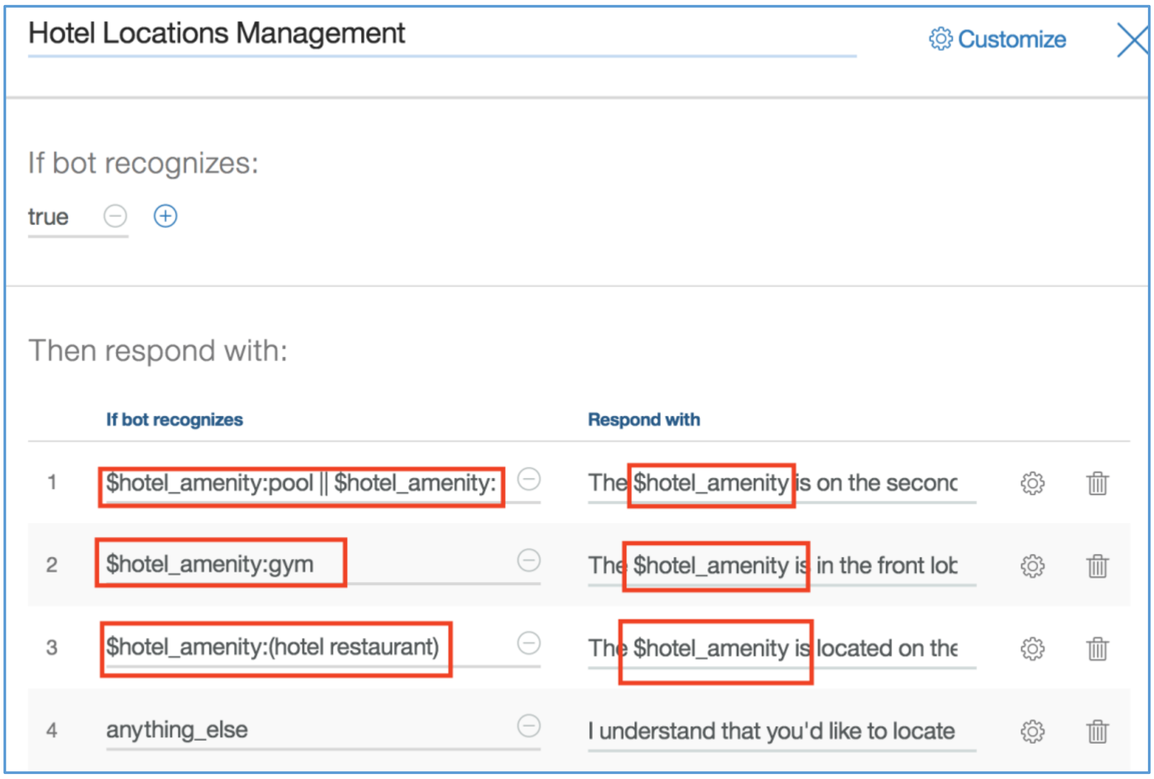


Then respond with:

	If bot recognizes	Respond with
1	@hotel_amenity	Enter a response...
2	anything_else	Enter a response...

11. Select **Hotel Locations Management** node with the four responses to edit it
12. Update the condition, replace *#hotel_locations* with *true*
13. In each response of the node, you must replace

@hotel_amenity	with	*$hotel_amenity*	as condition
@hotel_amenity.value	with	*$hotel_amenity*	in the response



Hotel Locations Management

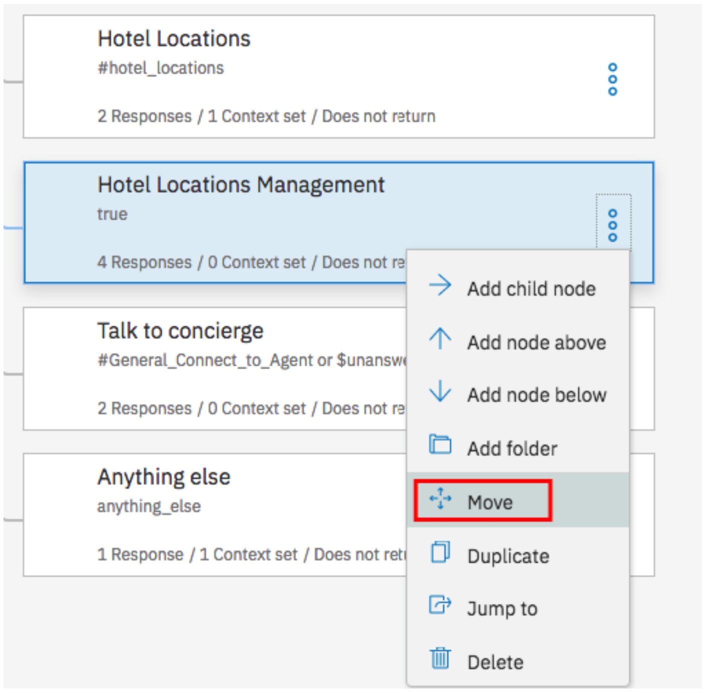
If bot recognizes:

true

Then respond with:

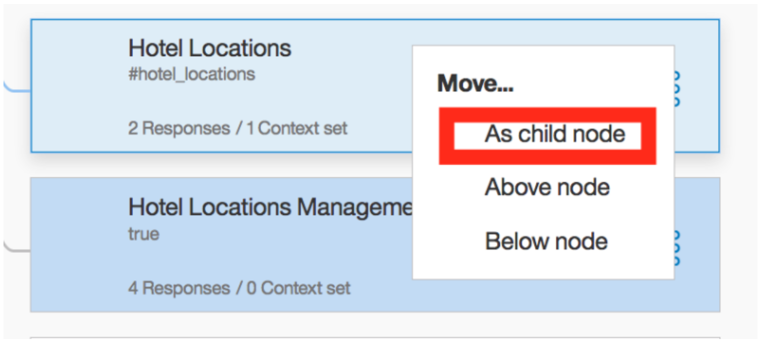
	If bot recognizes	Respond with
1	$hotel_amenity:pool || $hotel_amenity:	The $hotel_amenity is on the seconc
2	$hotel_amenity:gym	The $hotel_amenity is in the front lot
3	$hotel_amenity:(hotel restaurant)	The $hotel_amenity is located on the
4	anything_else	I understand that you'd like to locate

14. On the node menu of **Hotel Locations Management** node, select **Move** option


Hotel Locations
#hotel_locations
2 Responses / 1 Context set / Does not return
Hotel Locations Management
true
4 Responses / 0 Context set / Does not re
Talk to concierge
#General_Connect_to_Agent or $unansw
2 Responses / 0 Context set / Does not re
Anything else
anything_else
1 Response / 1 Context set / Does not ret
Add child node
Add node above
Add node below
Add folder
Move
Duplicate
Jump to
Delete

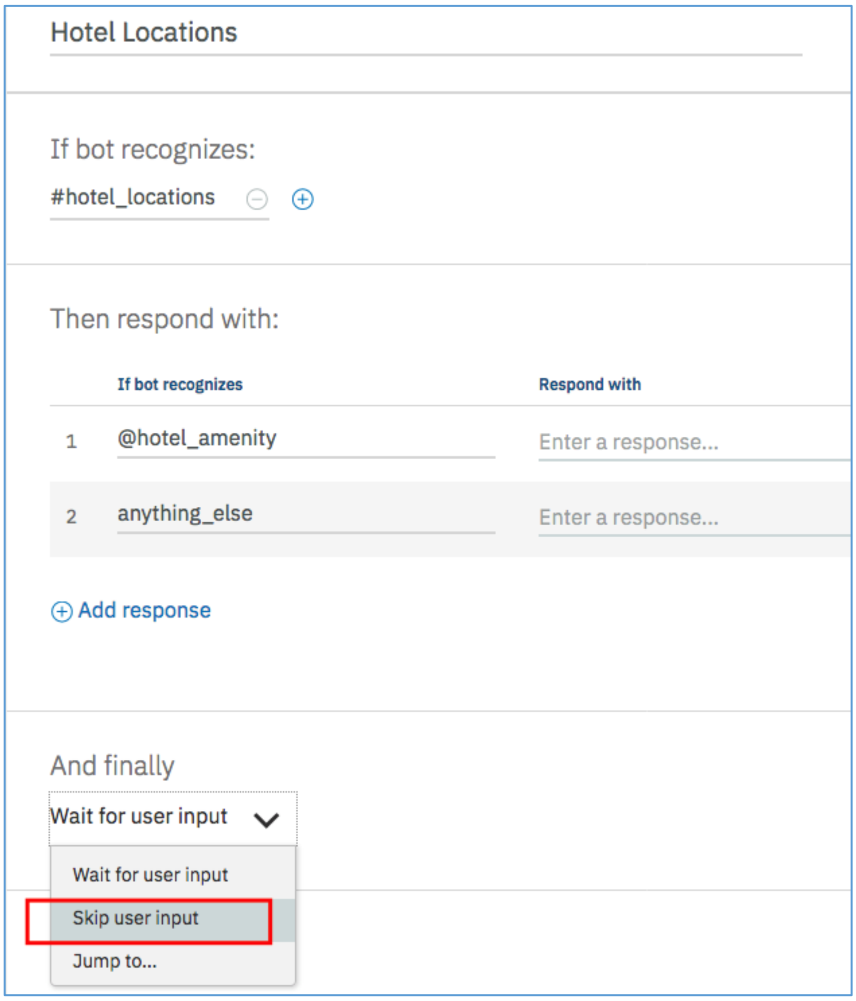
15. Select the **Hotel Locations** node

16. Select **As child node option**



17. Select the **Hotel Locations** node to edit it

18. At the bottom, click **Wait for user input**



Hotel Locations

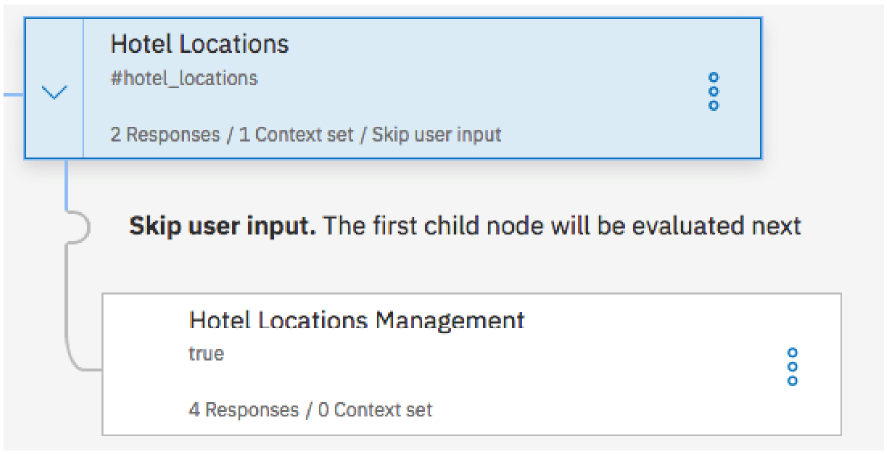
If bot recognizes:

#hotel_locations

Then respond with:

	If bot recognizes	Respond with
1	@hotel_amenity	Enter a response...
2	anything_else	Enter a response...

19. Select **Hotel Locations Management** node
20. Then select **Skip user input** option



Hotel Locations
#hotel_locations
2 Responses / 1 Context set / Skip user input

Skip user input. The first child node will be evaluated next

Hotel Locations Management
true
4 Responses / 0 Context set

Now your bot is ready to manage the entity *$hotel_amenity* in two nodes: **Hotel Location** and **Hotel Hours.**

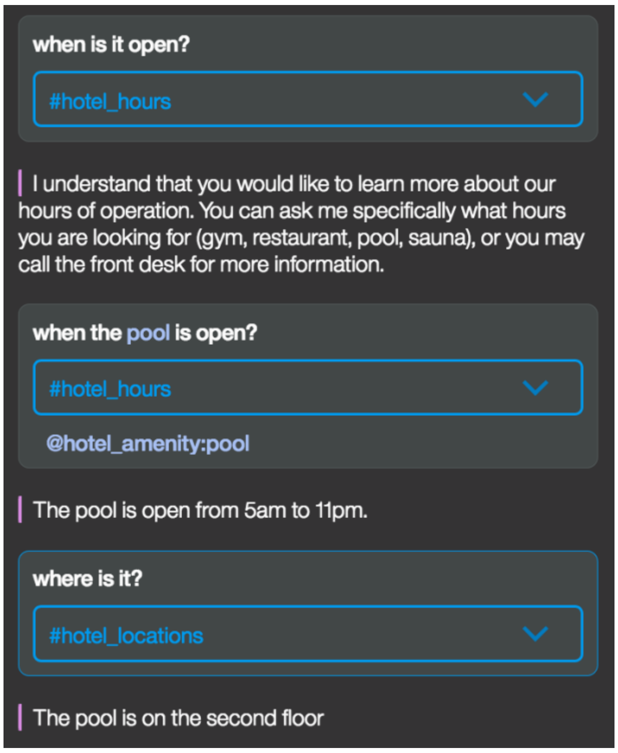
Let's test it.

21. Open the **Try it out** panel, enter successively:

    When is it open?

    when the pool is open?

    where is it?



when is it open?

#hotel_hours

I understand that you would like to learn more about our hours of operation. You can ask me specifically what hours you are looking for (gym, restaurant, pool, sauna), or you may call the front desk for more information.

when the pool is open?

#hotel_hours

@hotel_amenity:pool

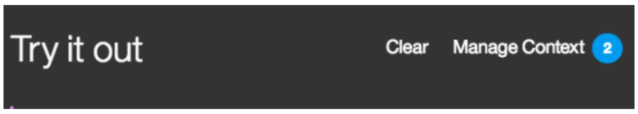
The pool is open from 5am to 11pm.

where is it?

#hotel_locations

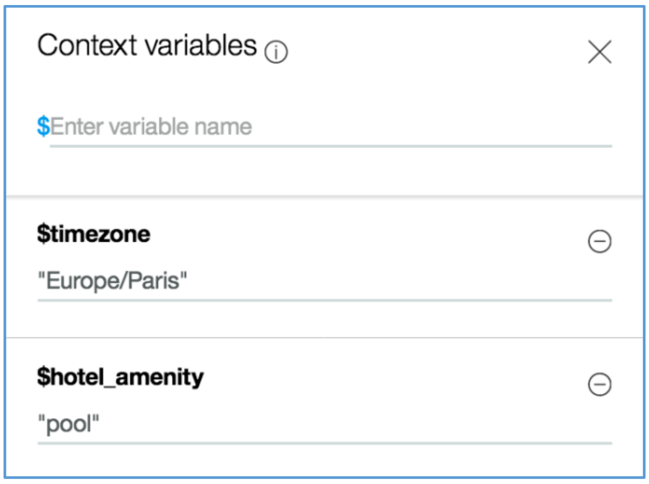
The pool is on the second floor

At the top of the Try it out panel, you can see how many context variables Watson manages.



22. Open the **Manage Context** panel

    The *$hotel_amenity* is correctly set to *pool.*


Context variables

$Enter variable name

$timezone

"Europe/Paris"

$hotel_amenity

"pool"

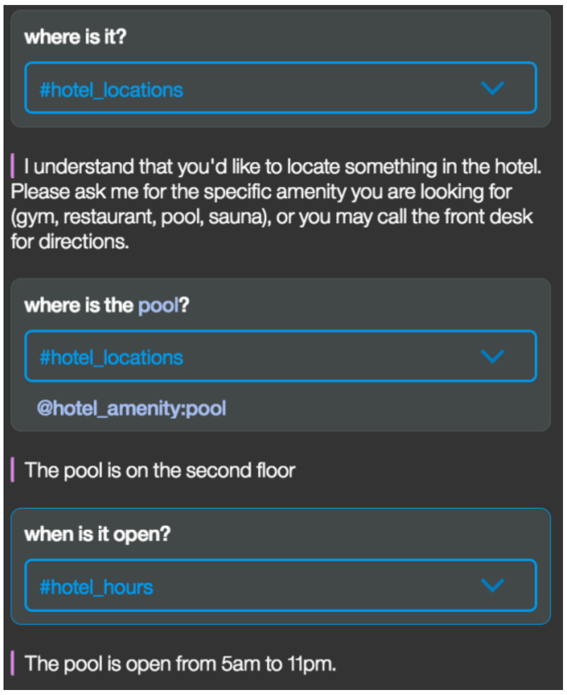
23. Close the **Manage Context** panel and click on **Clear** to reinitialize the context
24. On the **Try it out** panel, enter successively:

    Where is it?

    where is the pool?

    when is it open?

    You should get the following behaviours:


where is it?

#hotel_locations

I understand that you'd like to locate something in the hotel. Please ask me for the specific amenity you are looking for (gym, restaurant, pool, sauna), or you may call the front desk for directions.

where is the pool?

#hotel_locations

@hotel_amenity:pool

The pool is on the second floor

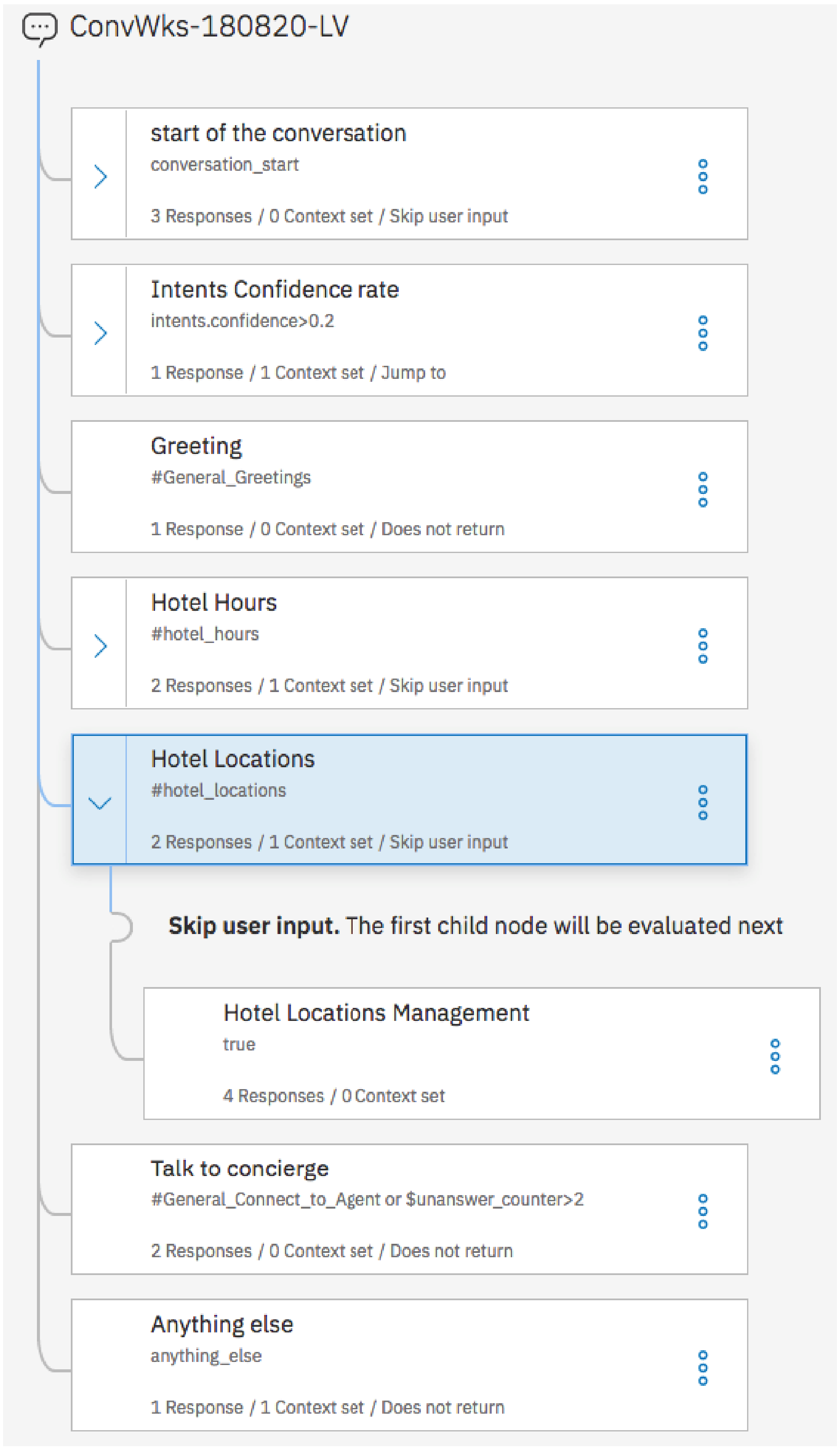
when is it open?

#hotel_hours

The pool is open from 5am to 11pm.

which was expected.

At this stage, your dialog looks like that:



ConvWks-180820-LV

start of the conversation
conversation_start
3 Responses / 0 Context set / Skip user input

Intents Confidence rate
intents.confidence>0.2
1 Response / 1 Context set / Jump to

Greeting
#General_Greetings
1 Response / 0 Context set / Does not return

Hotel Hours
#hotel_hours
2 Responses / 1 Context set / Skip user input

Hotel Locations
#hotel_locations
2 Responses / 1 Context set / Skip user input

Skip user input. The first child node will be evaluated next

Hotel Locations Management
true
4 Responses / 0 Context set

Talk to concierge
#General_Connect_to_Agent or $unanswer_counter>2
2 Responses / 0 Context set / Does not return

Anything else
anything_else
1 Response / 1 Context set / Does not return

Find a restaurant

Let’s find a restaurant for two potential intents:

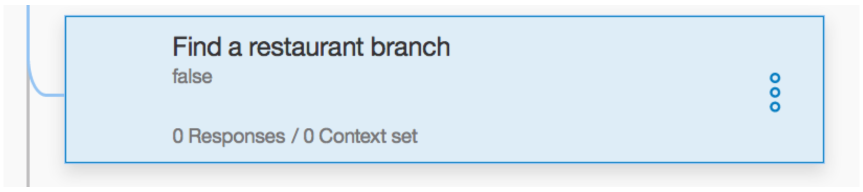
- you want to eat something or
- you are looking for a specific cuisine.

Watson will need 2 pieces of data from you: the type of restaurant and when you want to go to eat.
Sometimes the entity details can be captured in other nodes or the user might just have forgotten to explicitly determine what is their intent (or entity). In such case, you must request more details from the user.
You will manage to provide the right information to the end-users and request the required details to do it.

19. Find a restaurant branch

1. Add a new node above the **Anything else** node of the main tree.
2. Fill the node as below

Field	Value
Name of the node	*Find a restaurant branch*
Triggered by	*false*
Watson responses	Delete it



Find a restaurant branch
false
0 Responses / 0 Context set

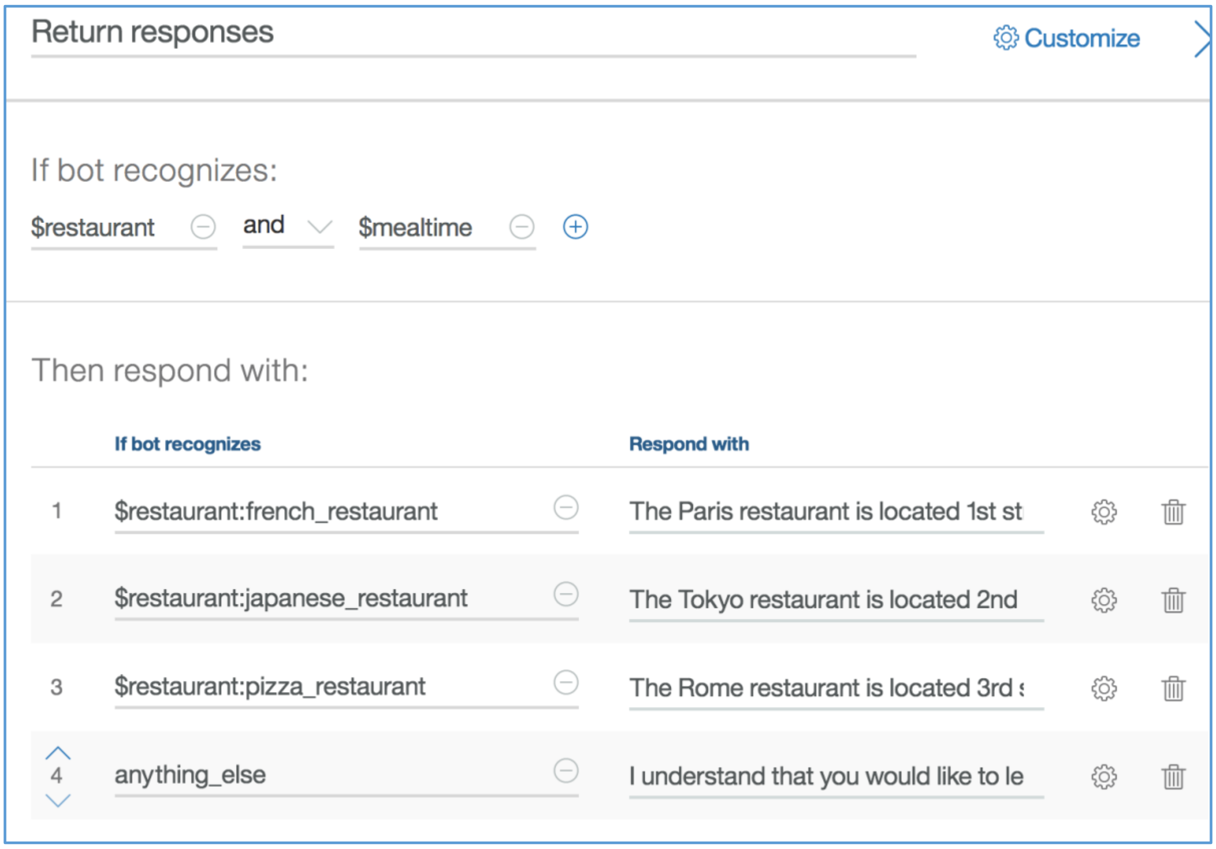
This node is just a placeholder to connect the branch. This is why there isn’t any response and the condition is false.

20. Return responses node

As a follow on, we check if *$restaurant* and *$mealtime* have been stored and finish the information gathering if it has. Initially, of course, we won't have this information, so this node will be skipped.

1. Add a child to **Find a restaurant branch** node and fill that node as below (don't forget to enable **Multiple response** option)

   name: *Return responses*
   condition: *$restaurant and $mealtime*
   resp1 condition: *$restaurant:french_restaurant*
   resp1 response: *The Paris restaurant is located 1st street and open for $mealtime*
   resp2 condition: *$restaurant:japanese_restaurant*
   resp2 response: *The Tokyo restaurant is located 2nd street and open for $mealtime*
   resp3 condition: *$restaurant:pizza_restaurant*
   resp3 response: *The Rome restaurant is located 3rd street and open for $mealtime*
   resp4 condition: *anything_else*
   resp4 response: *I understand that you would like to learn more about local restaurant. you may call the front desk for more information.*



Return responses

If bot recognizes:

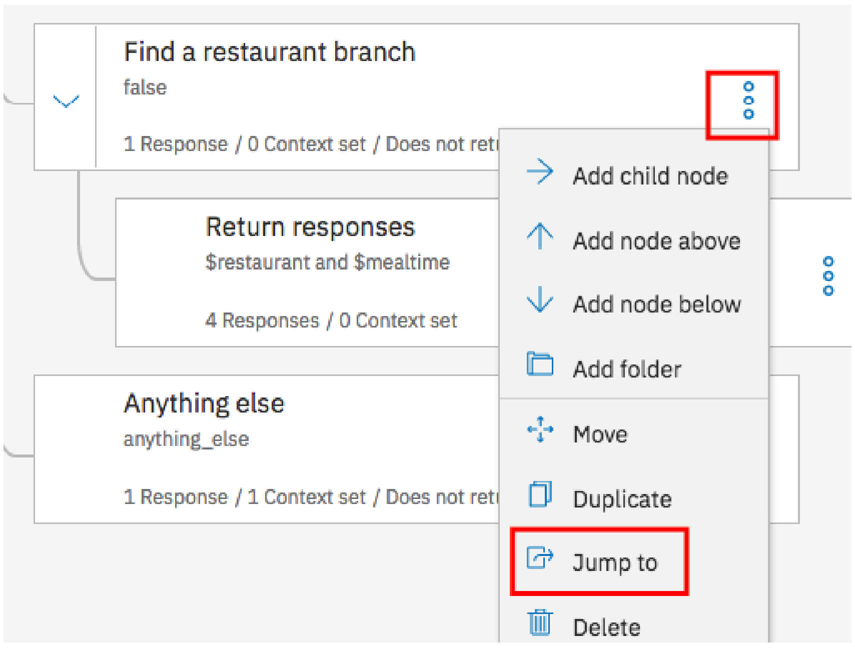
$restaurant and $mealtime

Then respond with:

	If bot recognizes	Respond with
1	$restaurant:french_restaurant	The Paris restaurant is located 1st st
2	$restaurant:japanese_restaurant	The Tokyo restaurant is located 2nd
3	$restaurant:pizza_restaurant	The Rome restaurant is located 3rd s
4	anything_else	I understand that you would like to le

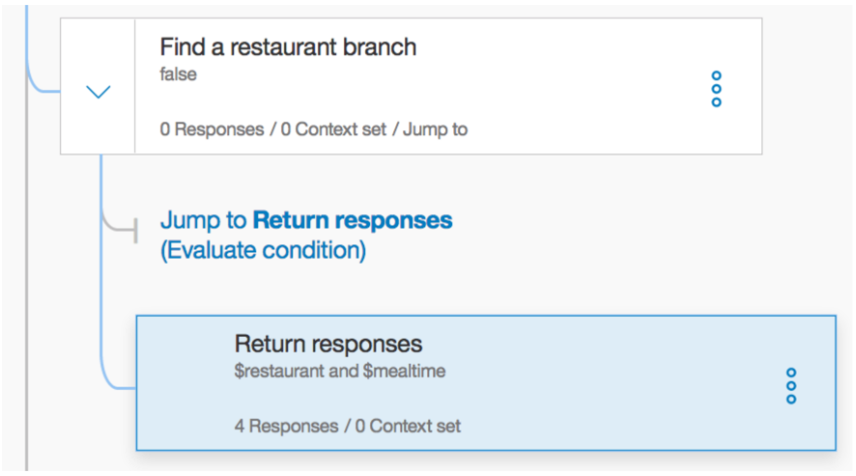
If you want to, you can add more responses for the other type of restaurant, but for this lab it will be enough.

3. As we want to execute this node without waiting for any input from the user, we are using the **Jump to** capability. Select **Find a restaurant branch** node
4. Click **Jump to** on the 3 dots menu.



5. Select the **return responses** node
6. As we want to evaluate the condition select **if bot recognizes (condition)**

   Then you should have:

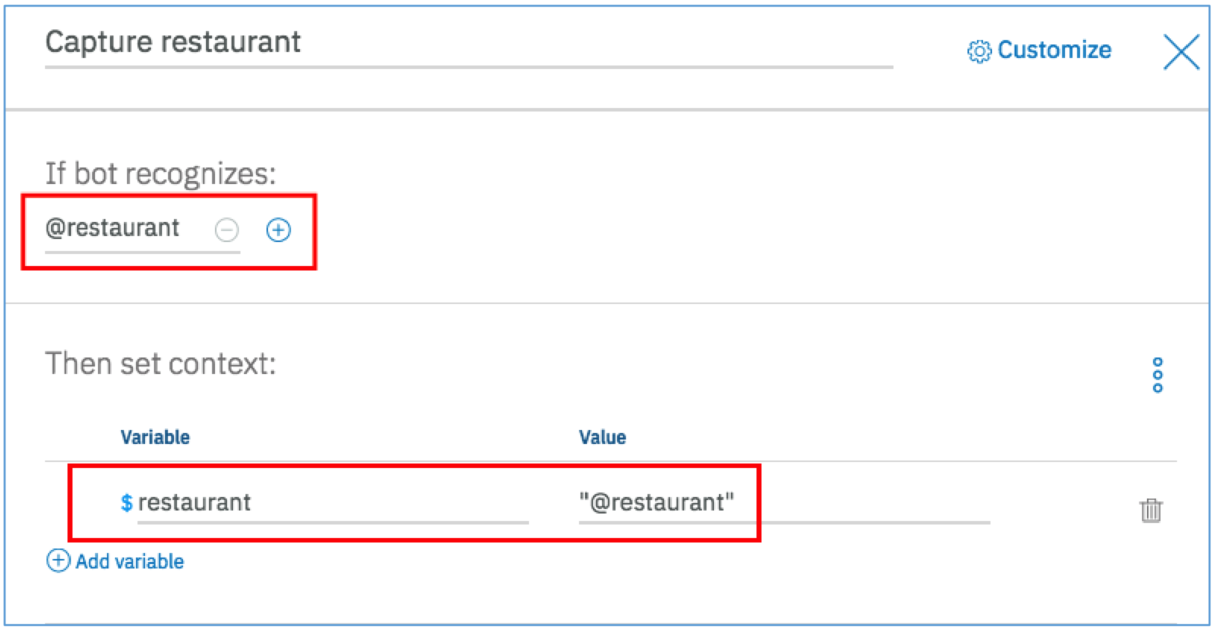

Find a restaurant branch
false
0 Responses / 0 Context set / Jump to
Jump to Return responses
(Evaluate condition)
Return responses
$restaurant and $mealtime
4 Responses / 0 Context set

Next, we will check to see if the user has provided the type of restaurant. If we have it, then we store it in context and continue from the next node.

21. Capture restaurant & mealtime nodes

1. Select **Return responses** node and add a new node after it by clicking **Add node below**.
2. Fill the node as below

name:	*Capture restaurant*
condition:	*@restaurant*
context Variable:	*restaurant*
context Value:	*@restaurant*


Capture restaurant

If bot recognizes:

@restaurant

Then set context:

Variable	Value
$ restaurant	"@restaurant"

If this node is executed, Watson got the required information so we can go back to the first node to display the answer to the user.

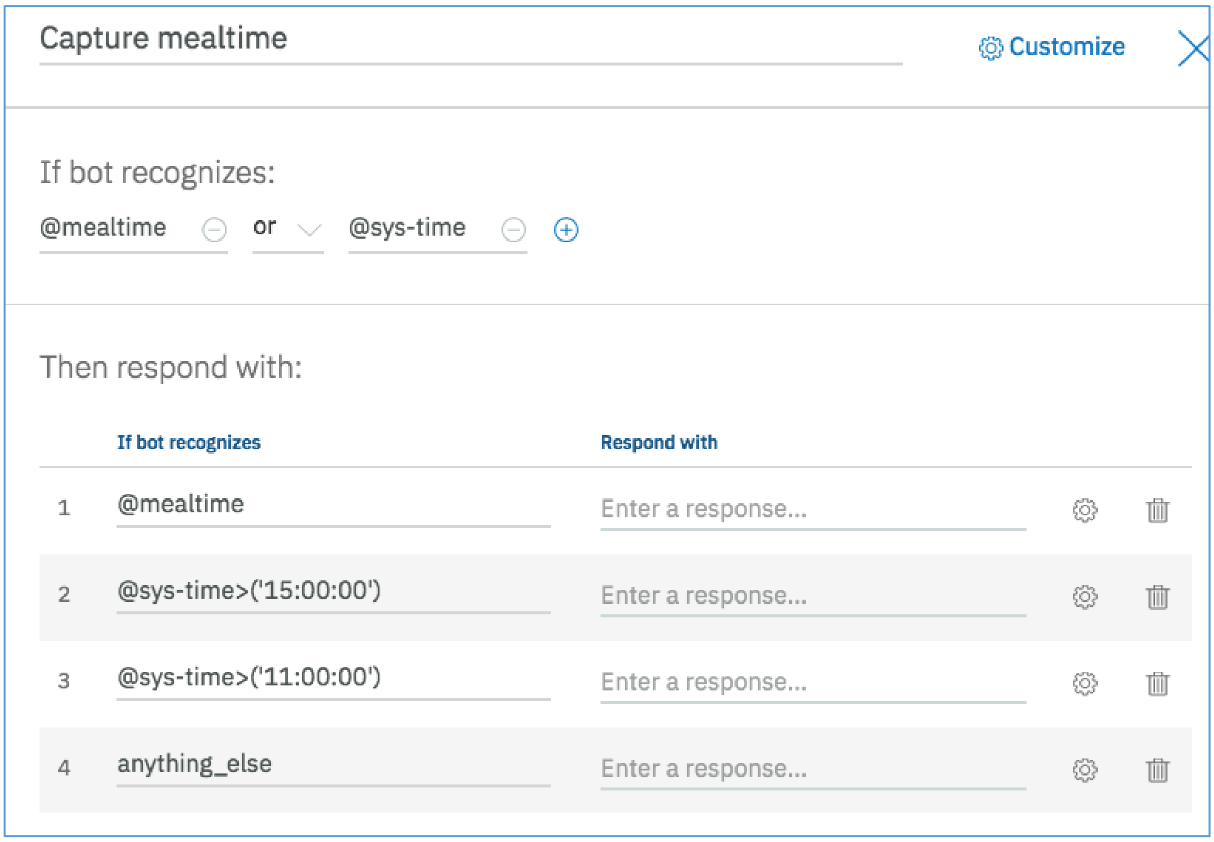
Now, you must capture the mealtime

3. Select **Capture restaurant** node and add a new node after it by clicking **Add node below**.

4. Fill the node as below

name:	*Capture mealtime*
condition:	*@mealtime or @sys-time*
resp1 condition:	*@mealtime*
resp1 context Variable:	*mealtime*
resp1 context Value:	*@mealtime*
resp2 condition:	*@sys-time>('15:00:00')*
resp2 context Variable:	*mealtime*
resp2 context Value:	*dinner*
resp3 condition:	*@sys-time>('11:00:00')*
resp3 context Variable:	*mealtime*
resp3 context Value:	*lunch*
resp4 condition:	*anything_else*
resp4 context Variable:	*mealtime*
resp4 context Value:	*breakfast*

Now you should get:


Capture mealtime

If bot recognizes:

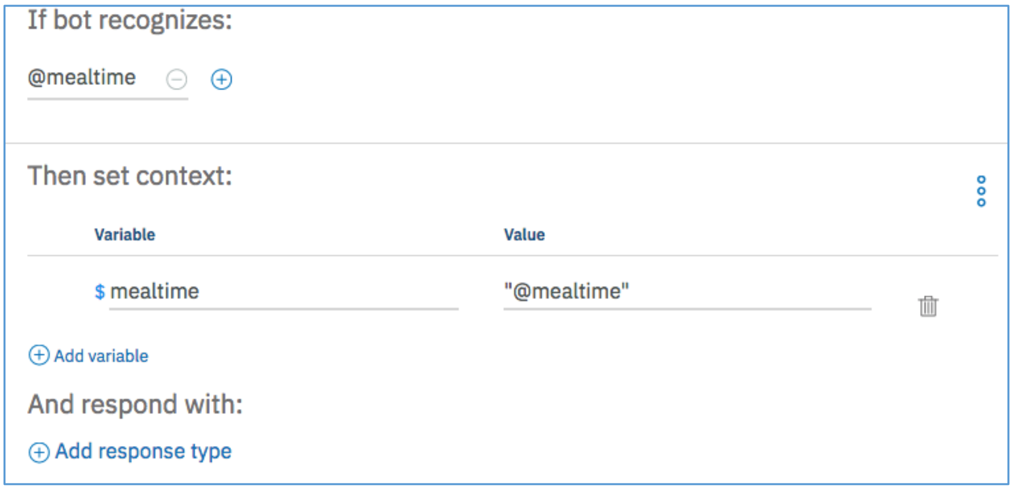
@mealtime or @sys-time

Then respond with:

	If bot recognizes	Respond with
1	@mealtime	Enter a response...
2	@sys-time>('15:00:00')	Enter a response...
3	@sys-time>('11:00:00')	Enter a response...
4	anything_else	Enter a response...

Don’t forget to delete response type option for each row.

When you click on **configure** for each row, you should see:



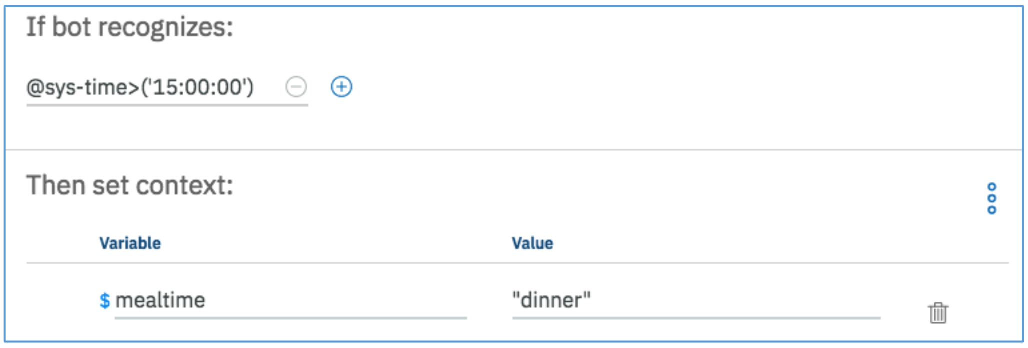
If bot recognizes:

@mealtime

Then set context:

Variable	Value
$ mealtime	"@mealtime"

And respond with:

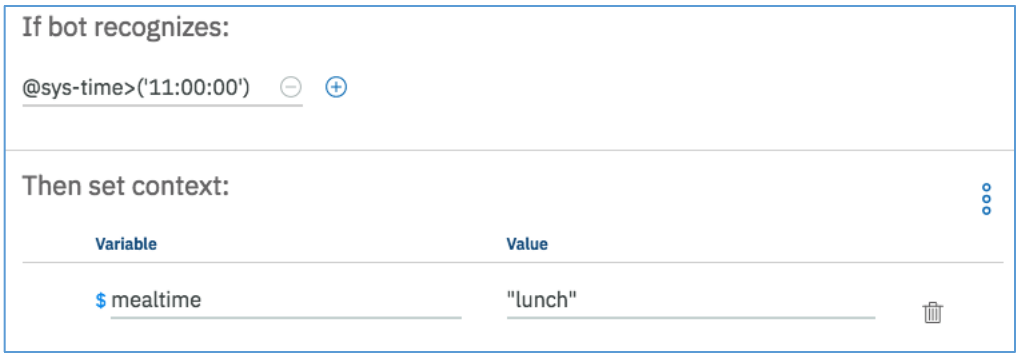


If bot recognizes:

@sys-time>('15:00:00')

Then set context:

Variable	Value
$ mealtime	"dinner"

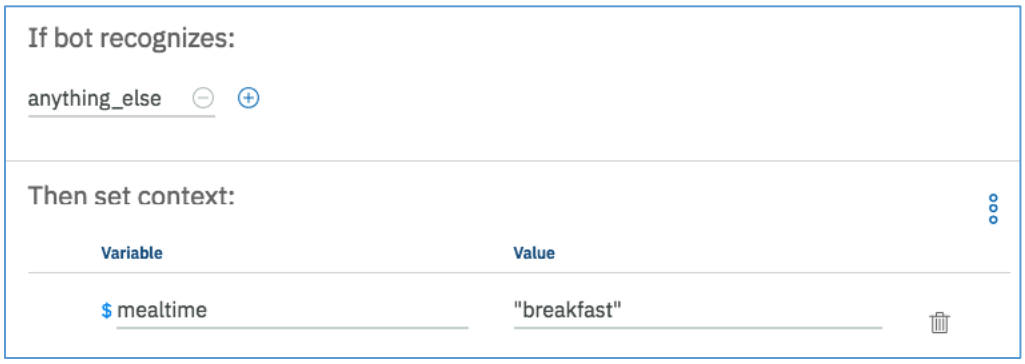


If bot recognizes:

@sys-time>('11:00:00')

Then set context:

Variable	Value
$ mealtime	"lunch"



If bot recognizes:

anything_else

Then set context:

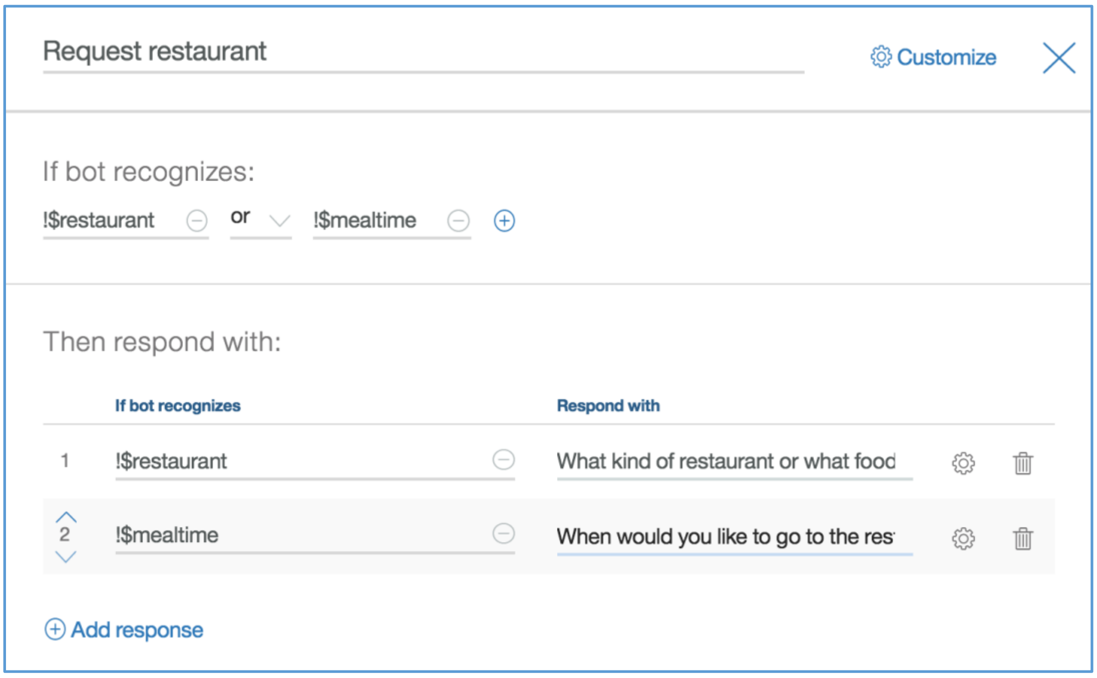
Variable	Value
$ mealtime	"breakfast"

22. Request missing information nodes

Now we check for missing information. If we don't have the right information, then we prompt for it. After prompting, we continue from the user input.

1. Select **Capture mealtime** node and add a new node after it by clicking **Add node below**.
2. Fill the node as below (don't forget to enable **Multiple response** option)

name:	*Request restaurant*
condition:	*!$restaurant or !$mealtime*
resp1 condition:	*!$restaurant*
resp1 response:	*What kind of restaurant or what food would you like?*
Resp2 condition:	*!$mealtime*
Resp2 response:	*When would you like to go to the restaurant?*


Request restaurant

If bot recognizes:

!$restaurant or !$mealtime

Then respond with:

	If bot recognizes	Respond with
1	!$restaurant	What kind of restaurant or what food
2	!$mealtime	When would you like to go to the res

23. Loop management

1. Select **Request Restaurant** node and add a new node after it by clicking **Add node below**.
2. Fill the node as below

name:	*Loop*
condition:	*anything_else*

3. Delete the response type

24. Navigation node management

Now we are going to manage the navigation between nodes of the branch.

As we want to check all required information in one time, we will use **Jump to** as follows:

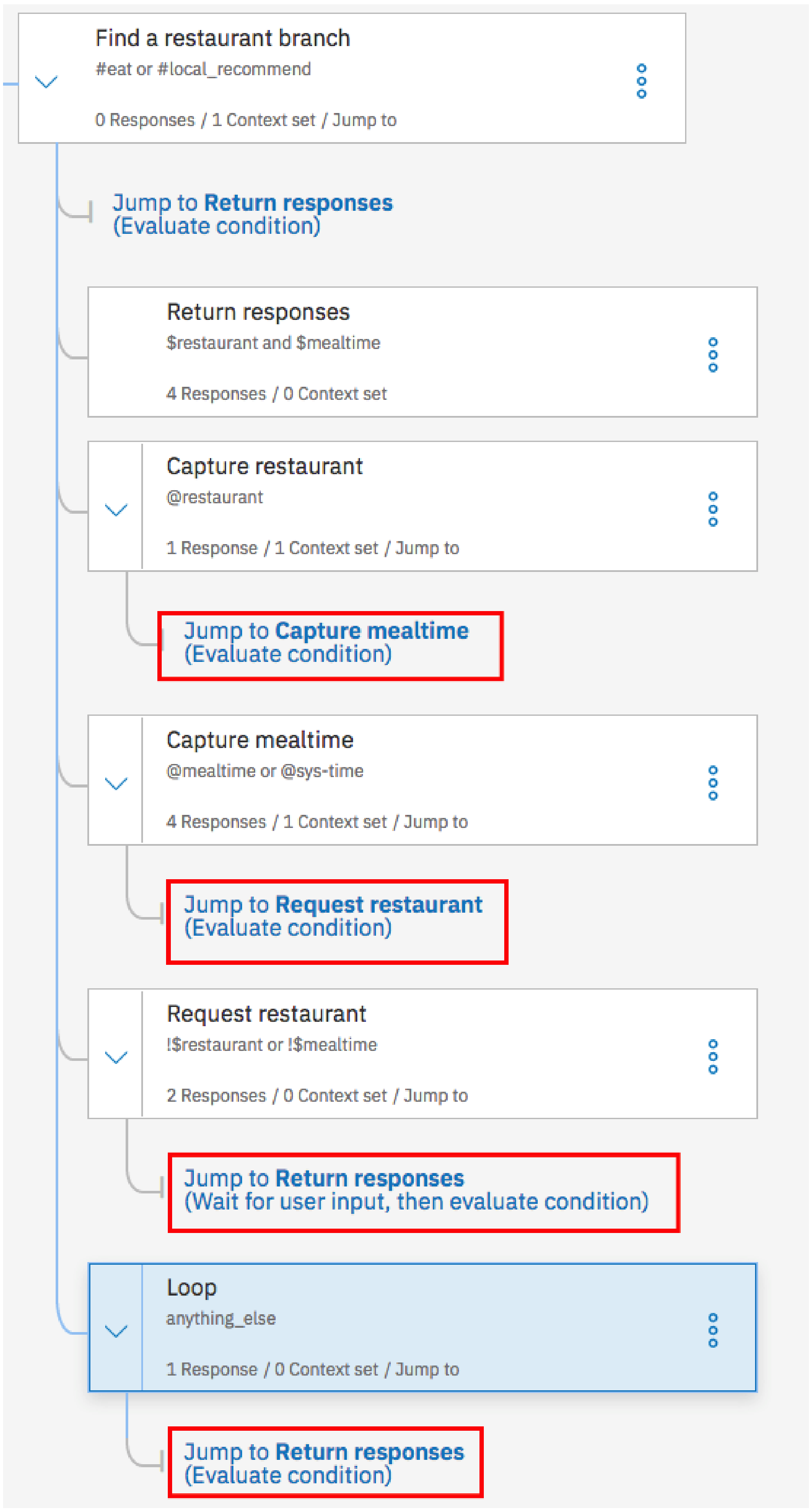
1. We will add **Jump to** and evaluate condition
   - from **Capture restaurant** node, jump to **Capture mealtime** node
   - from **Capture mealtime** node, jump to **Request restaurant** node.

If we don't get all required data, the bot prompts a question, it waits for the answer and returns to the node which will evaluate answer (**Return responses** node).

2. We will add **Jump to** and wait for users input then evaluate condition
   - from **Request restaurant** node, jump to **Return response**s node
3. In the end, we will add **Jump to** and evaluate condition
   - from **Loop** node, jump to **Return response**s node

Note : if bot recognizes (condition) = evaluate condition

Then you should get:



Find a restaurant branch
#eat or #local_recommend
0 Responses / 1 Context set / Jump to

Jump to **Return responses**
(Evaluate condition)

Return responses
$restaurant and $mealtime
4 Responses / 0 Context set

Capture restaurant
@restaurant
1 Response / 1 Context set / Jump to

Jump to **Capture mealtime**
(Evaluate condition)

Capture mealtime
@mealtime or @sys-time
4 Responses / 1 Context set / Jump to

Jump to **Request restaurant**
(Evaluate condition)

Request restaurant
!$restaurant or !$mealtime
2 Responses / 0 Context set / Jump to

Jump to **Return responses**
(Wait for user input, then evaluate condition)

Loop
anything_else
1 Response / 0 Context set / Jump to

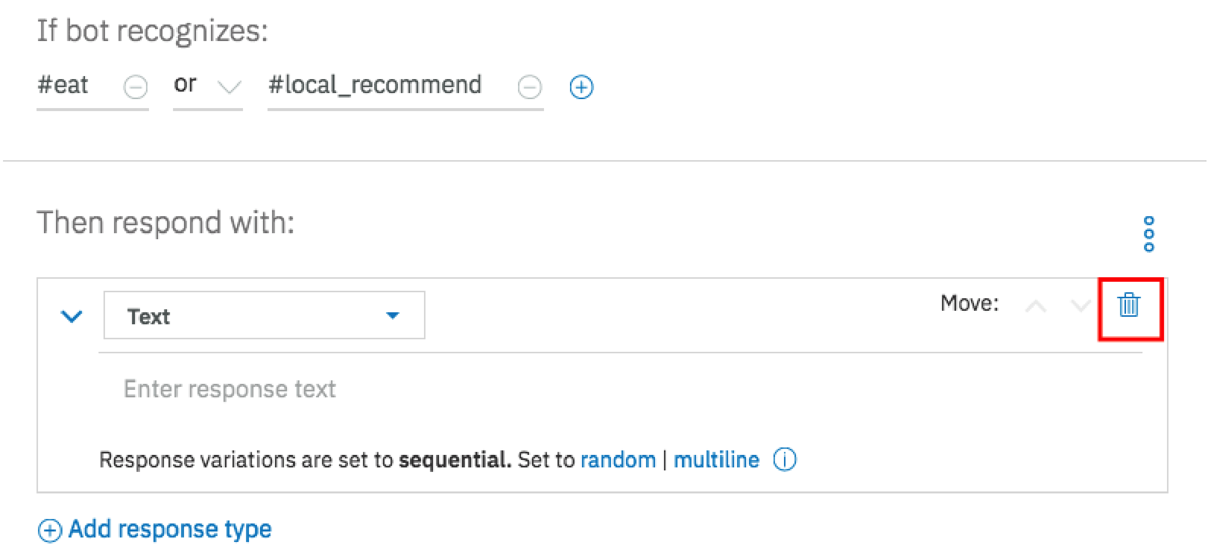
Jump to **Return responses**
(Evaluate condition)

25. Test your branch

I remind you this branch should be leveraged by 2 nodes triggered by the intent *#eat* and *#local_recommend*. Their behaviours can be different.

For the test, we will simplify the dialog.

1. Select **Find a restaurant branch** node,
2. Replace the condition *false* with *#eat* or *#local_recommend*
3. Remove the empty respond by clicking on the **remove** icon


If bot recognizes:

#eat or #local_recommend

Then respond with:

Text

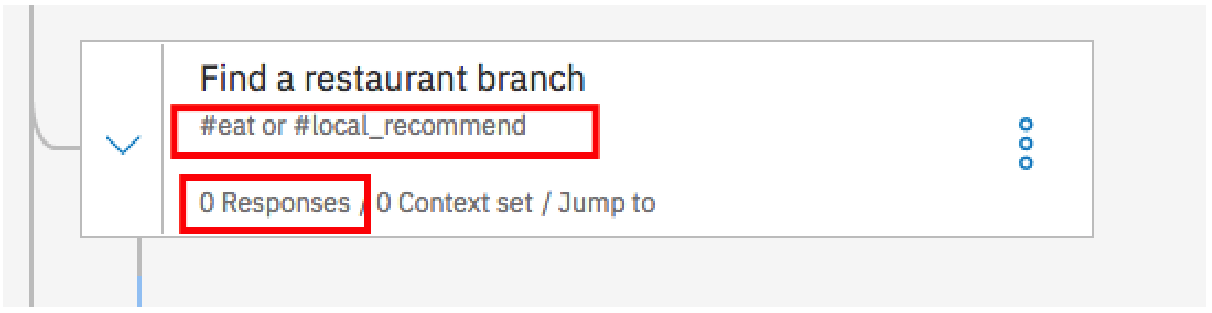
Move:

Enter response text

Response variations are set to **sequential.** Set to random | multiline

Add response type

The we should get:


Find a restaurant branch

#eat or #local_recommend

0 Responses / 0 Context set / Jump to

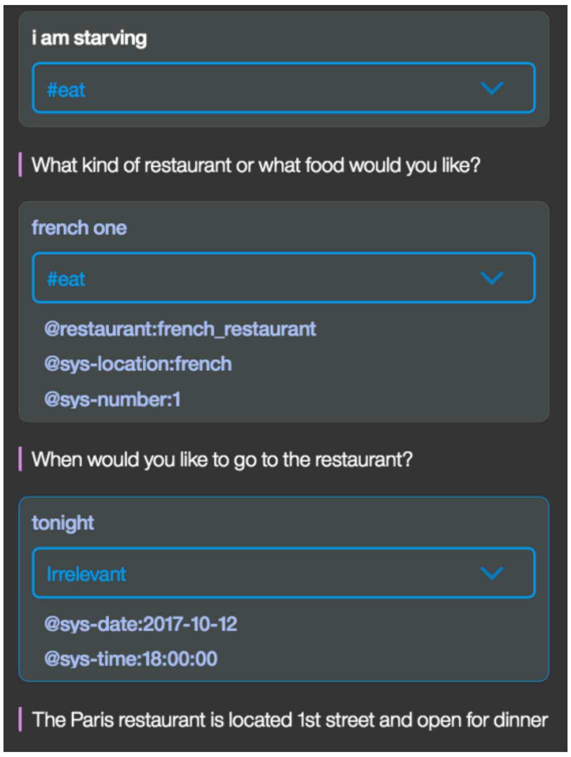
4. Open **Try it out** panel
5. Enter successively

   I am starving

   French one

   tonight

   Watson works as expected and requests the missing information



i am starving

#eat

What kind of restaurant or what food would you like?

french one

#eat

@restaurant:french_restaurant

@sys-location:french

@sys-number:1

When would you like to go to the restaurant?

tonight

Irrelevant

@sys-date:2017-10-12

@sys-time:18:00:00

The Paris restaurant is located 1st street and open for dinner

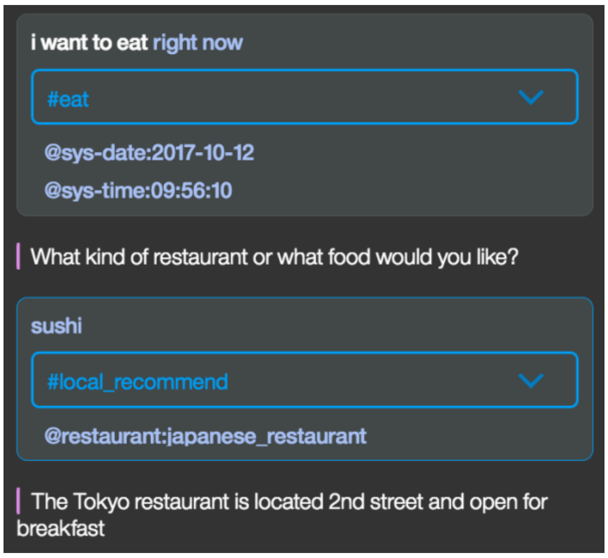
6. Click **Clear**

7. Enter successively

   I want to eat right now

   sushi

   Watson determine when you want to eat and requests the missing information



i want to eat right now

#eat

@sys-date:2017-10-12

@sys-time:09:56:10

What kind of restaurant or what food would you like?

sushi

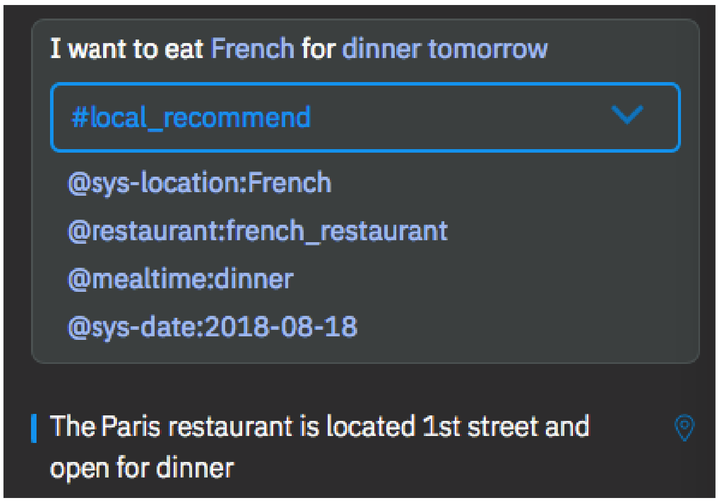
#local_recommend

@restaurant:japanese_restaurant

The Tokyo restaurant is located 2nd street and open for breakfast

8. Click **Clear**

   Enter *I want to eat French for dinner tomorrow*



I want to eat French for dinner tomorrow

#local_recommend

@sys-location:French

@restaurant:french_restaurant

@mealtime:dinner

@sys-date:2018-08-18

The Paris restaurant is located 1st street and open for dinner

You can continue your test. Don’t forget to click clear between each test to reinitialize the context variable.

Enhance Find a restaurant branch

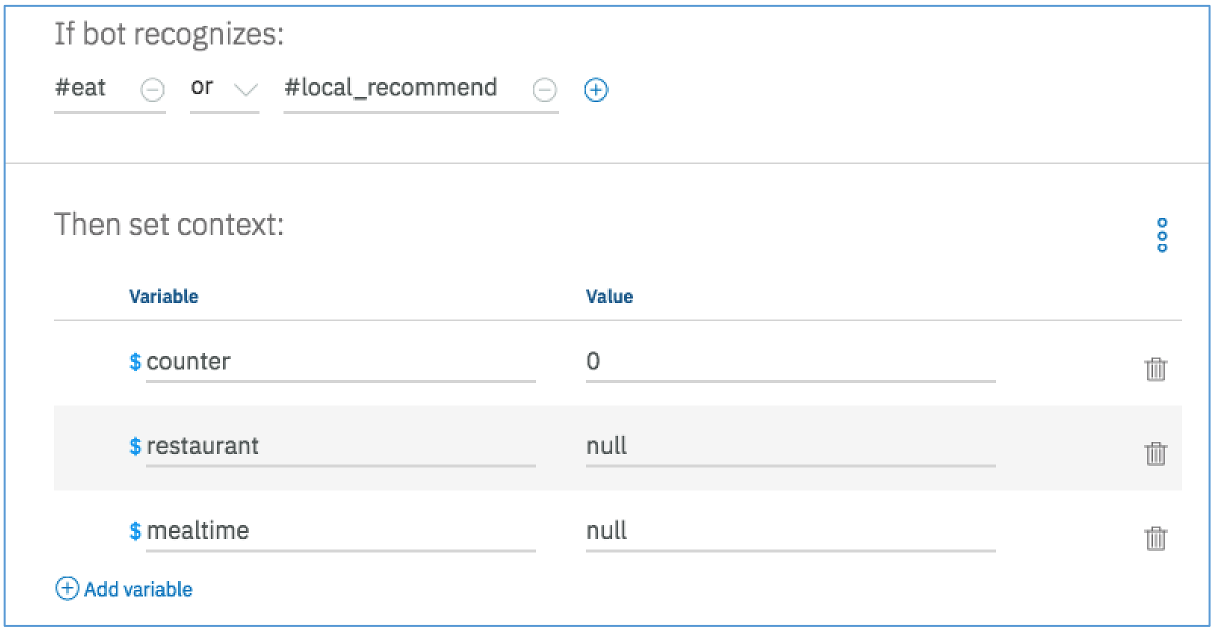
(optional)

If your user looks for a restaurant several times during the conversation the *$restaurant*, and *$mealtime* must be reinitiated to take into account his new choice. It can be done by the application or by Watson Assistant. In this lab, we are going to reinitiate the context variable in the Dialog.
Also, if your user doesn't provide the right information, Watson will repeat the question again and again. So, you want to use a counter context variable to stop the loop.

26. Initialize context variables

1. Select the node **Find a restaurant branch** to edit it
2. Fill the context variables like this

context Variable:	*counter*
context Value:	*0*
context Variable:	*restaurant*
context Value:	*null*
context Variable:	*mealtime*
context Value:	*null*


If bot recognizes:

#eat or #local_recommend

Then set context:

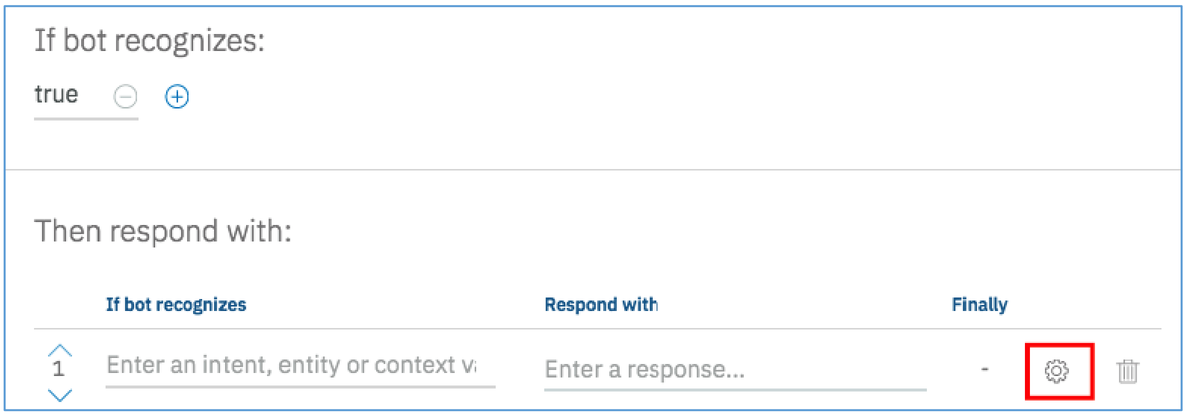
Variable	Value
$ counter	0
$ restaurant	null
$ mealtime	null

27. Avoid infinite loop

1. Select **Return Responses** node and add a new node below by clicking **Add node below**.
2. Fill the node as below

   name: *Counter Management*

   condition: *true*
3. Click **Customize** and switch on **Multiple responses**
4. On the first row , click **Edit response**



If bot recognizes:

true

Then respond with:

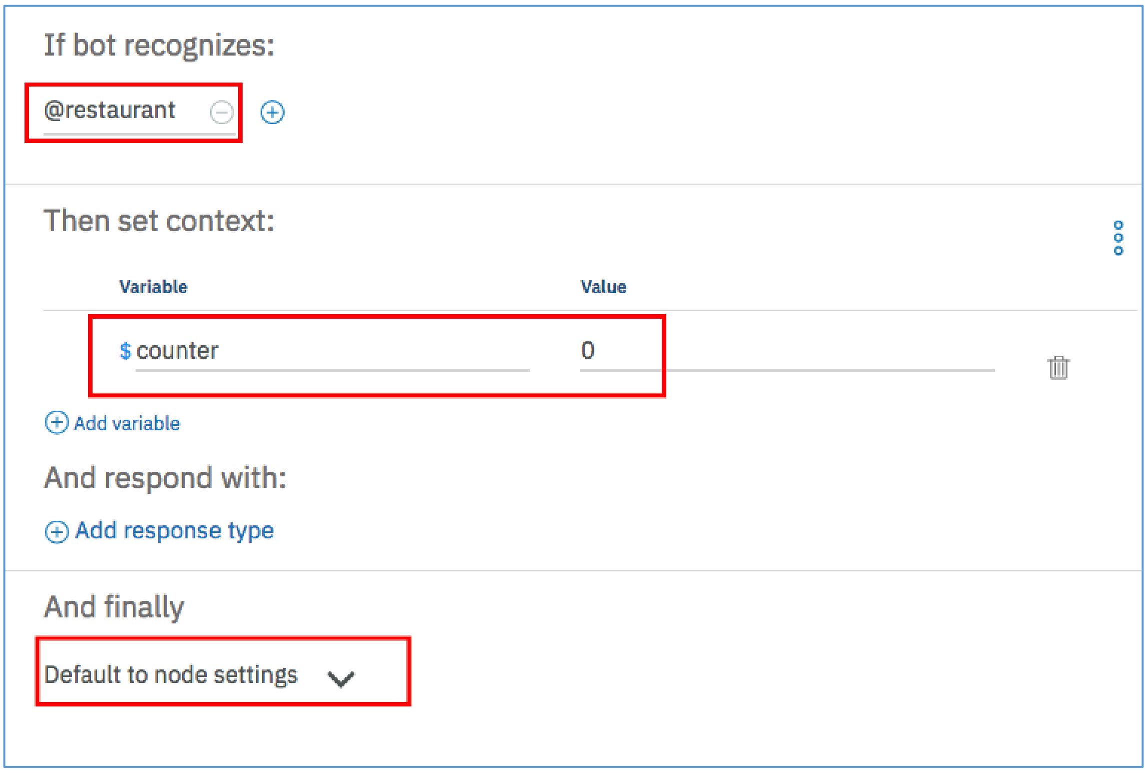
	If bot recognizes	Respond with	Finally
1	Enter an intent, entity or context vi	Enter a response...	-

5. Fill the node as below to reinitialize the counter when we get an expected input.

   resp1 condition: *@restaurant*

   resp1 context Variable: *counter*

   resp1 context Value: *0*



If bot recognizes:

@restaurant

Then set context:

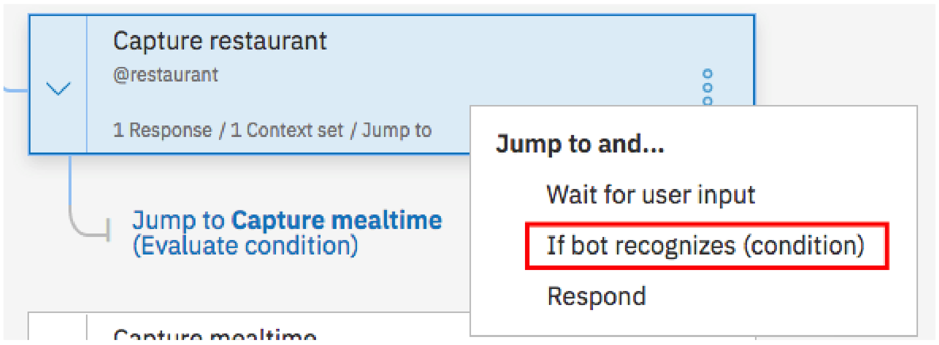
Variable	Value
$ counter	0

And respond with:

And finally

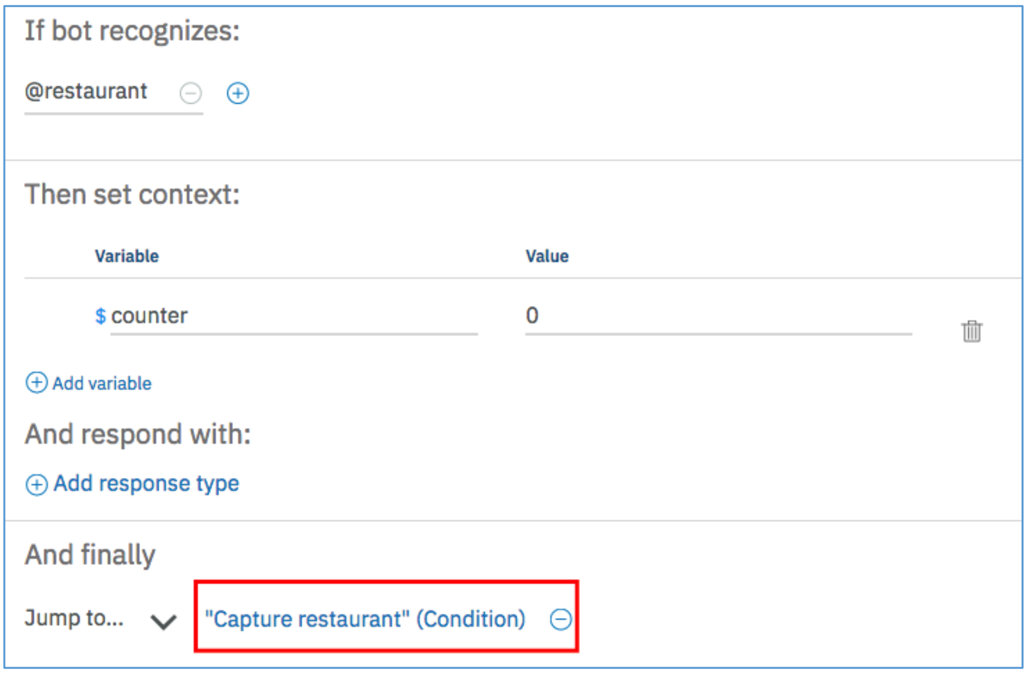
Default to node settings

6. At the bottom, click **Default to node settings**
7. Select jump to…option then select **Capture restaurant** node


Capture restaurant
@restaurant
1 Response / 1 Context set / Jump to
Jump to Capture mealtime (Evaluate condition)
Jump to and...
Wait for user input
If bot recognizes (condition)
Respond
Capture mealtime

8. Select **condition** option

Then we should get


If bot recognizes:

@restaurant

Then set context:

Variable	Value
$ counter	0

Add variable

And respond with:

Add response type

And finally

Jump to... "Capture restaurant" (Condition)

9. Click **Save**

10. Repeat the steps 4 to 9 for the 3 other responses with the following parameters:

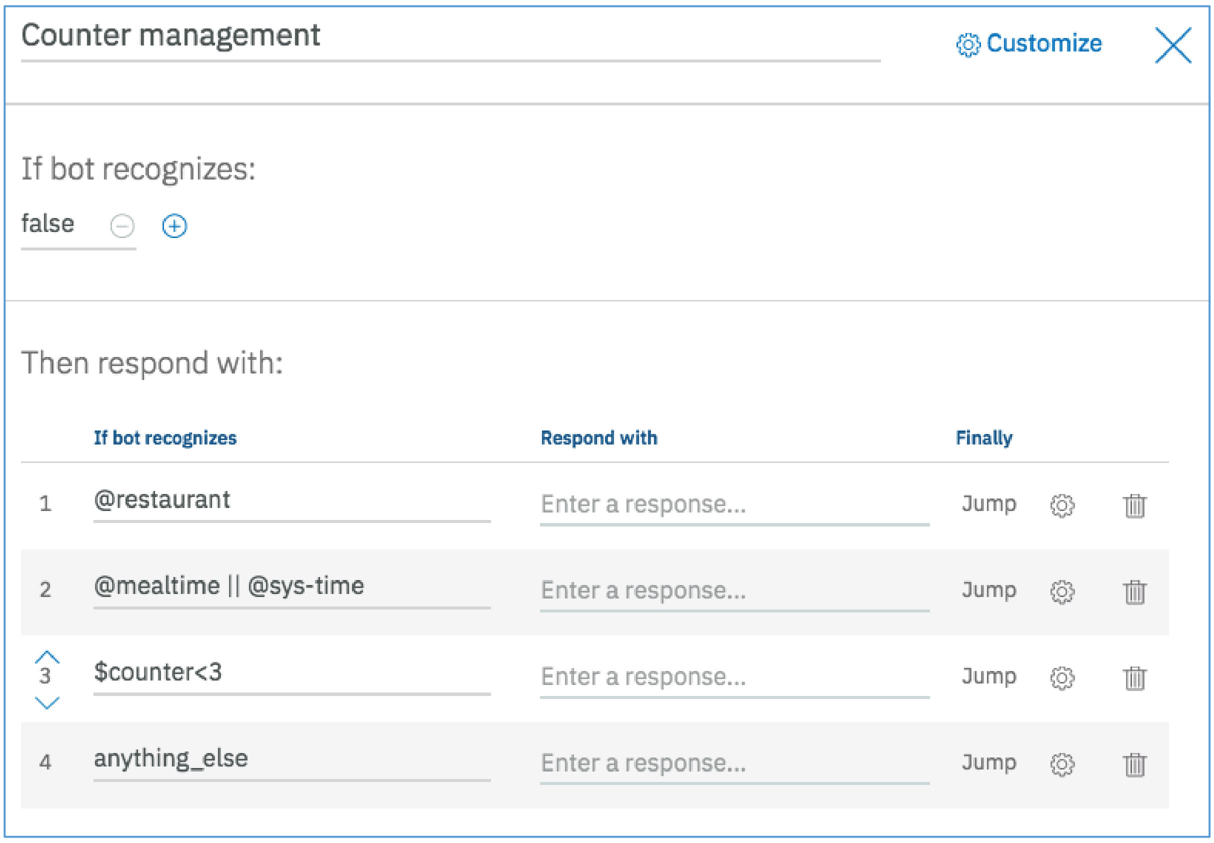
    to reinitialize the counter when we get an expected input.

    resp2 condition: *@mealtime || @sys-time*
    resp2 context Variable: *counter*
    resp2 context Value: *0*
    resp2 jump to condition node: *Capture mealtime*

    to increment the counter when we don't get any expected input.
    resp3 condition: *$counter<3*
    resp3 context Variable: *counter*
    resp3 context Value: *"<? context.counter +1 ?>"*
    resp3 jump to condition node: *Request restaurant*

    to close the loop after too much iterations.
    resp4 condition: *anything_else*
    resp4 context Variable: *mealtime*
    resp4 context Value: *"nodef"*
    resp4 context Variable: *restaurant*
    resp4 context Value: *"nodef"*
    resp3 jump to condition node: *Return responses*

Now you should get:


Counter management

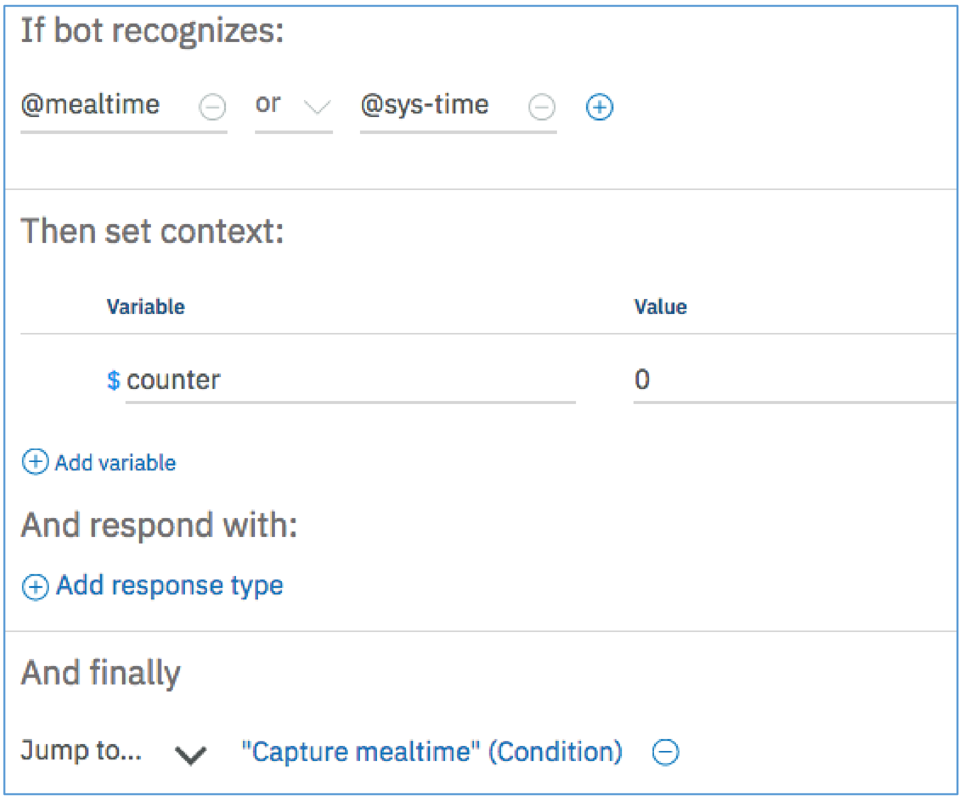
If bot recognizes:

false

Then respond with:

	If bot recognizes	Respond with	Finally
1	@restaurant	Enter a response...	Jump
2	@mealtime || @sys-time	Enter a response...	Jump
3	$counter<3	Enter a response...	Jump
4	anything_else	Enter a response...	Jump

For the 3 last responses you will get



If bot recognizes:

@mealtime or @sys-time

Then set context:

Variable	Value
$ counter	0

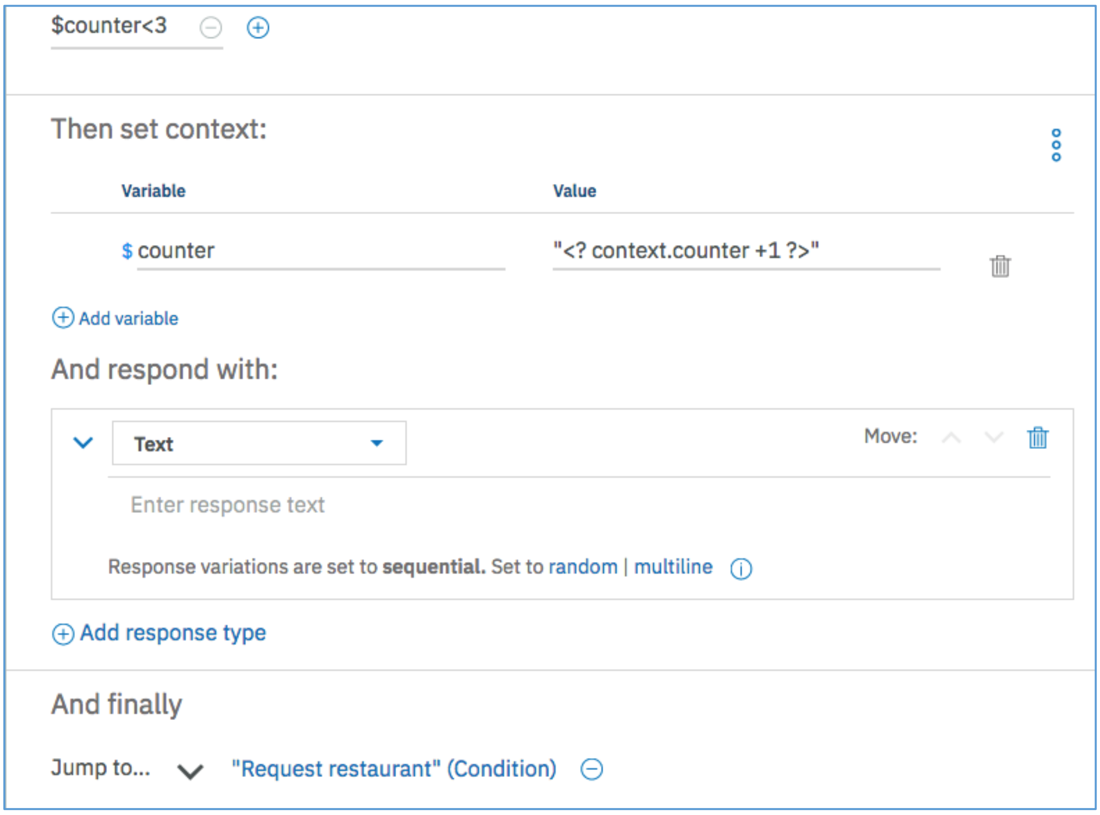
Add variable

And respond with:

Add response type

And finally

Jump to... "Capture mealtime" (Condition)



$counter<3

Then set context:

Variable	Value
$ counter	"<? context.counter +1 ?>"

Add variable

And respond with:

Text

Move:

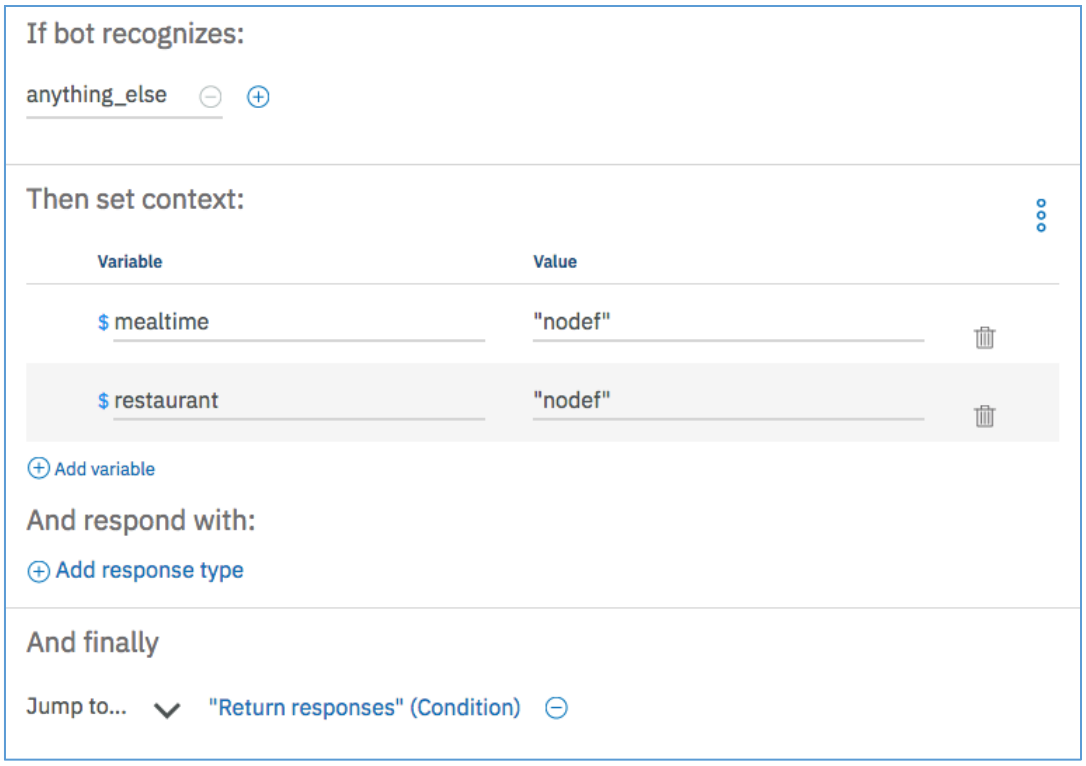
Enter response text

Response variations are set to **sequential.** Set to random | multiline

Add response type

And finally

Jump to... "Request restaurant" (Condition)


If bot recognizes:

anything_else

Then set context:

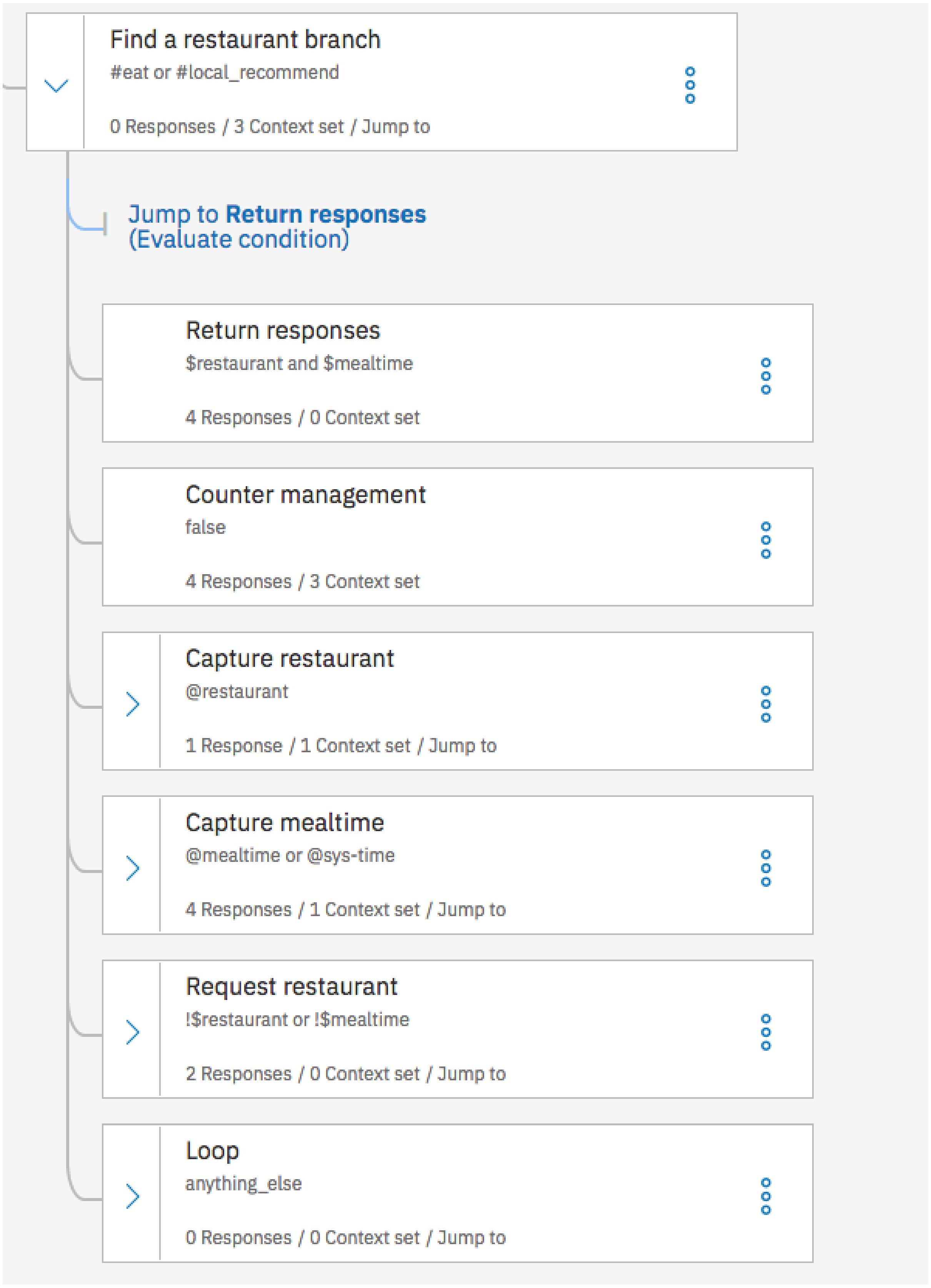
Variable	Value
$ mealtime	"nodef"
$ restaurant	"nodef"

And respond with:

And finally

Jump to... "Return responses" (Condition)

Now you should get the following branch.



Find a restaurant branch
#eat or #local_recommend
0 Responses / 3 Context set / Jump to

Jump to **Return responses**
(Evaluate condition)

Return responses
$restaurant and $mealtime
4 Responses / 0 Context set

Counter management
false
4 Responses / 3 Context set

Capture restaurant
@restaurant
1 Response / 1 Context set / Jump to

Capture mealtime
@mealtime or @sys-time
4 Responses / 1 Context set / Jump to

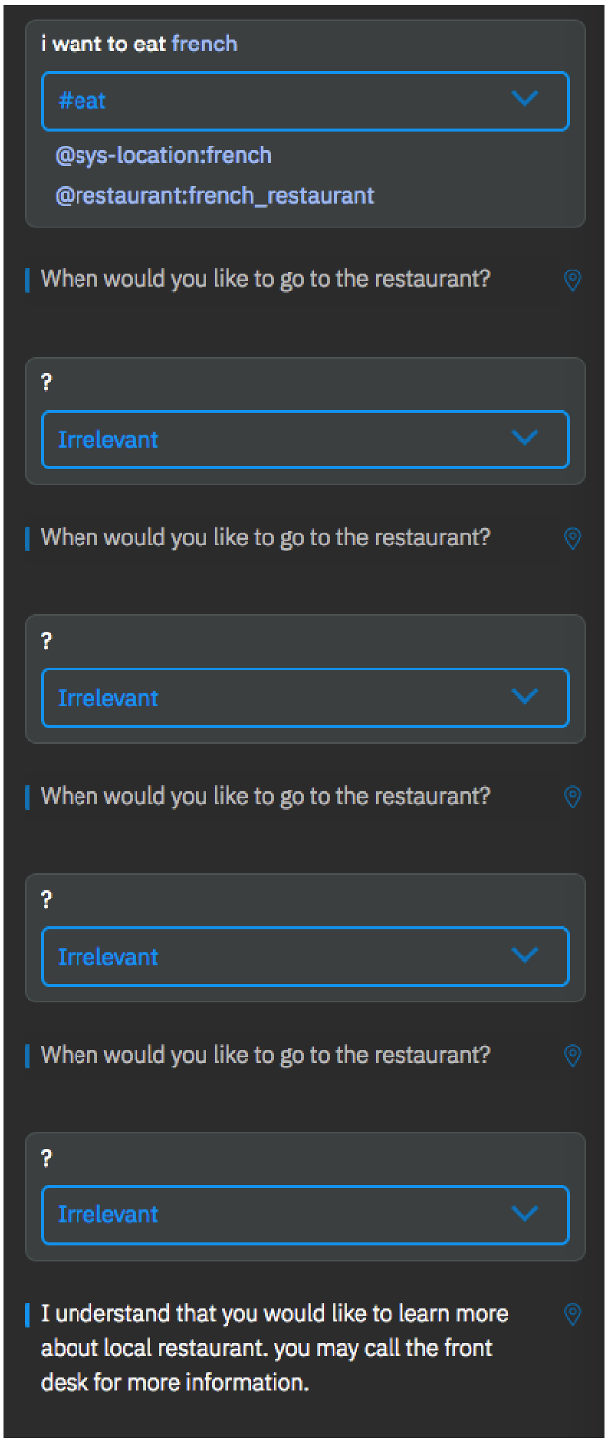
Request restaurant
!$restaurant or !$mealtime
2 Responses / 0 Context set / Jump to

Loop
anything_else
0 Responses / 0 Context set / Jump to

28. Test the enhancements

1. Open **Try it out** panel
2. Enter successively
   I want to eat french
   ?
   ?
   ?
   ?

Watson determines when you want to eat and requests the missing information



i want to eat french

#eat

@sys-location:french

@restaurant:french_restaurant

When would you like to go to the restaurant?

?

Irrelevant

When would you like to go to the restaurant?

?

Irrelevant

When would you like to go to the restaurant?

?

Irrelevant

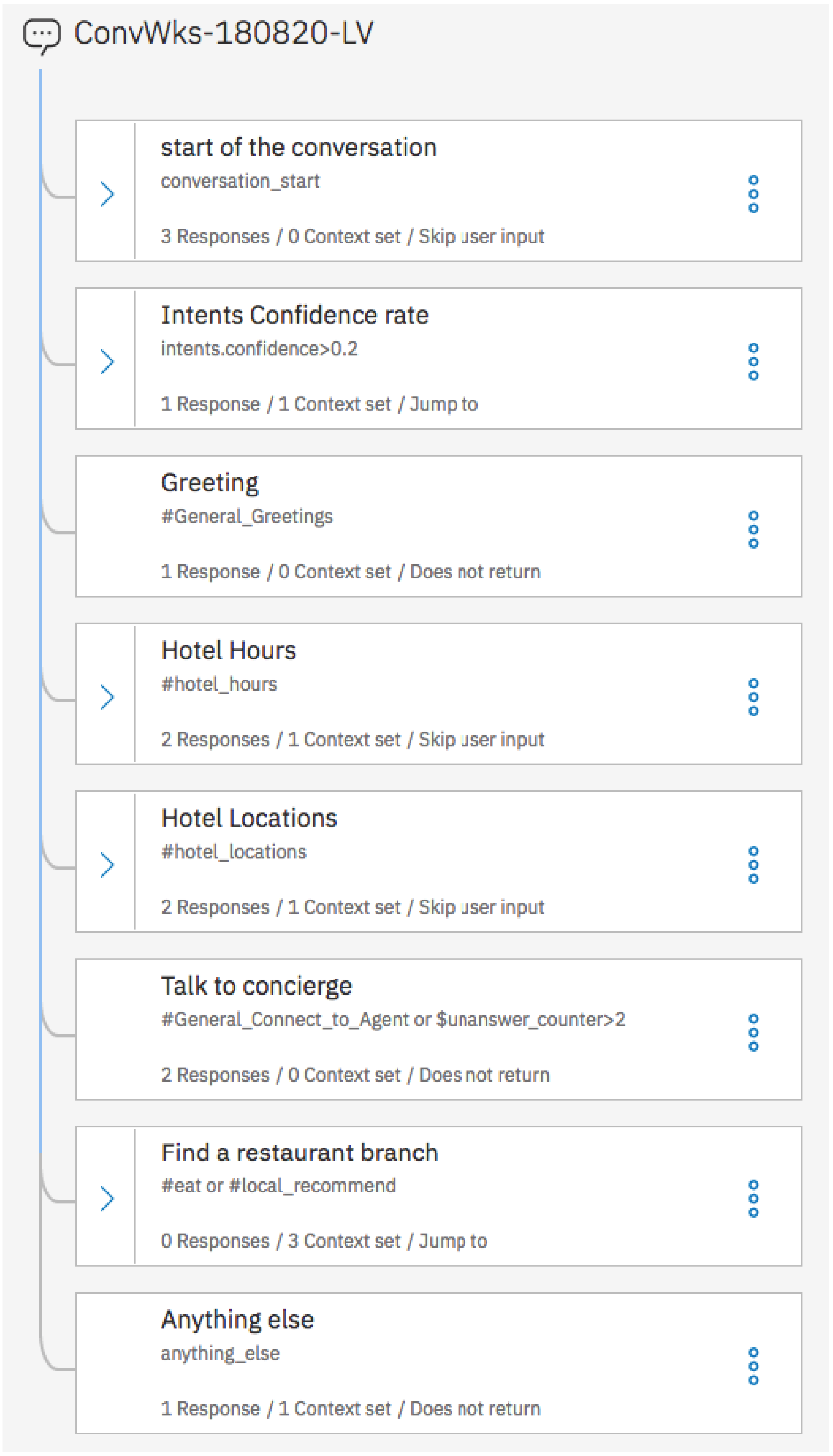
When would you like to go to the restaurant?

?

Irrelevant

I understand that you would like to learn more about local restaurant. you may call the front desk for more information.

The final dialog:



ConvWks-180820-LV

start of the conversation
conversation_start
3 Responses / 0 Context set / Skip user input

Intents Confidence rate
intents.confidence>0.2
1 Response / 1 Context set / Jump to

Greeting
#General_Greetings
1 Response / 0 Context set / Does not return

Hotel Hours
#hotel_hours
2 Responses / 1 Context set / Skip user input

Hotel Locations
#hotel_locations
2 Responses / 1 Context set / Skip user input

Talk to concierge
#General_Connect_to_Agent or $unanswer_counter>2
2 Responses / 0 Context set / Does not return

Find a restaurant branch
#eat or #local_recommend
0 Responses / 3 Context set / Jump to

Anything else
anything_else
1 Response / 1 Context set / Does not return