# Using Advanced Dialog Features in Watson Conversation

Tammy Feb 2, 2018 · 5 min read

Bots are smarter than ever, but sometimes they're not able to show off how smart they really are. Humans say and type in unpredictable ways, that even an expert conversationalist can't always account for. What if a user asks a bot two things at once? Even a human would have to process one request at a time. What if a user is going

Designing a dialog is no easy feat. Luckily there are tricks and tips that make sure your bot sounds as smart as it really is. This post will show you how using the Watson Conversation service (which you can try out for free).

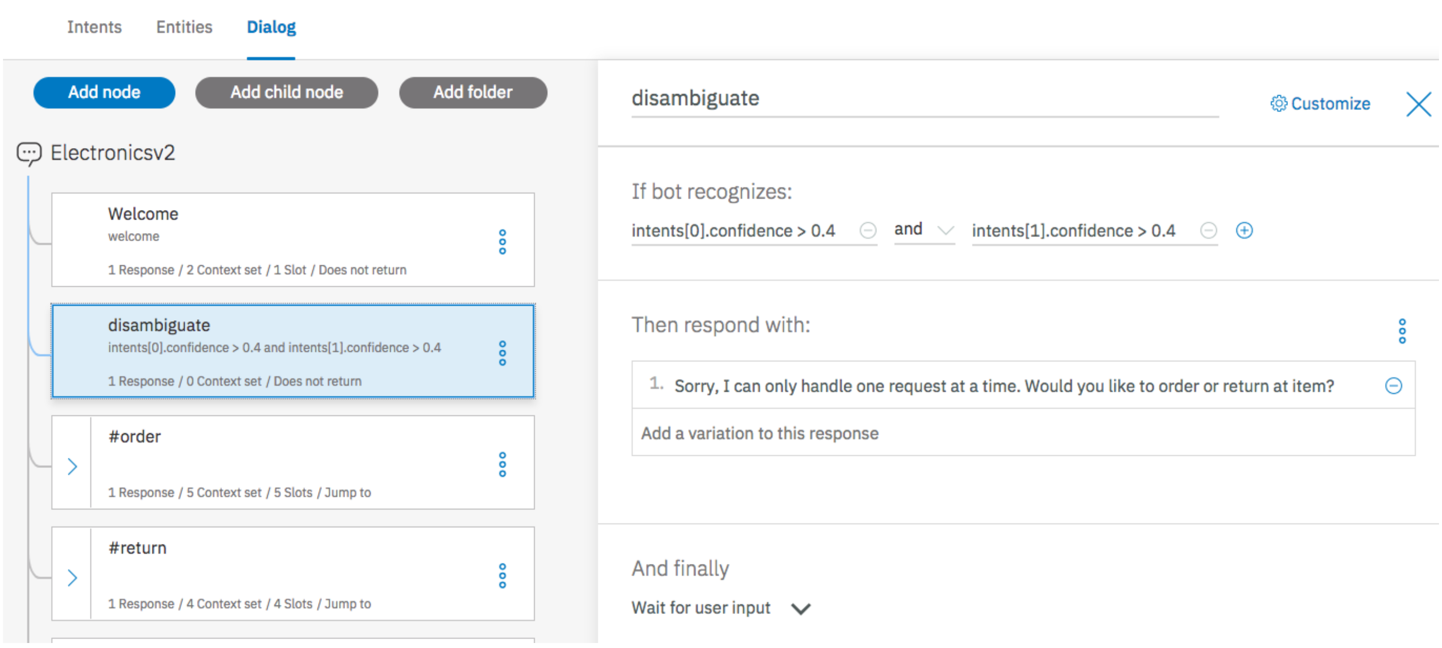
Watch this step-by-step video on how to utilize the more advanced features of Conversation or continue reading through this post.

This tutorial goes over how to handle two intents, manage handlers (how your bot will respond when the user's answer to a prompt is not found), use pattern entities, and add a counter. You can find the workspace json for this tutorial here to follow along.

**Handling two intents**

If a user asks a question or a request that contains two intents, for example, placing and order and making a return, the bot (and humans) will need to take it one at a time. To do this, you want your bot to be able to detect if multiple things are being said, and then clarify which one to process first.

In your dialog tree, create a node to disambiguate between the intents. This node is checking to see if the top two intents returned are higher than 40% confidence (40%

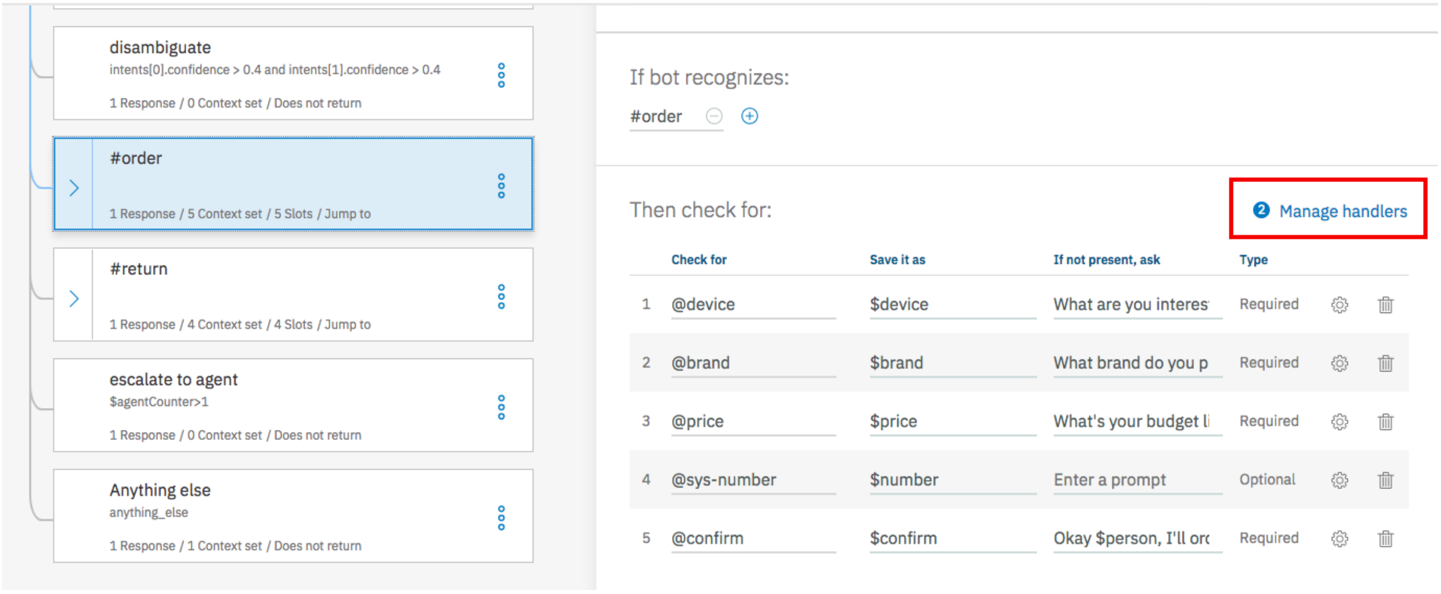

Intents
Entities
Dialog
Add node
Add child node
Add folder
Electronicsv2
Welcome
welcome
1 Response / 2 Context set / 1 Slot / Does not return
disambiguate
intents[0].confidence > 0.4 and intents[1].confidence > 0.4
1 Response / 0 Context set / Does not return
#order
1 Response / 5 Context set / 5 Slots / Jump to
#return
1 Response / 4 Context set / 4 Slots / Jump to
disambiguate
Customize
If bot recognizes:
intents[0].confidence > 0.4
and
intents[1].confidence > 0.4
Then respond with:
1. Sorry, I can only handle one request at a time. Would you like to order or return at item?
Add a variation to this response
And finally
Wait for user input

Disambiguating between top two intents

## Manage handlers

During a conversation, a user may change their mind and switch topics. For example, a user may be placing an order but then decide to return something first instead. With slots, a feature to gather information in a single node, the bot looks for the specific answers. If the specific answers are not found, your bot can look for handlers, or global intents within the node, and respond accordingly. So if a user decides to cancel the order half way through the flow, and a handler is set for a #cancel intent, the bot can stop collecting the necessary information and exit the node.

To add a handler, go to your node, and click 'Manage handlers'. Here, you can add the #cancel intent and the appropriate response.



disambiguate
intents[0].confidence > 0.4 and intents[1].confidence > 0.4
1 Response / 0 Context set / Does not return

#order
1 Response / 5 Context set / 5 Slots / Jump to

#return
1 Response / 4 Context set / 4 Slots / Jump to

escalate to agent
$agentCounter>1
1 Response / 0 Context set / Does not return

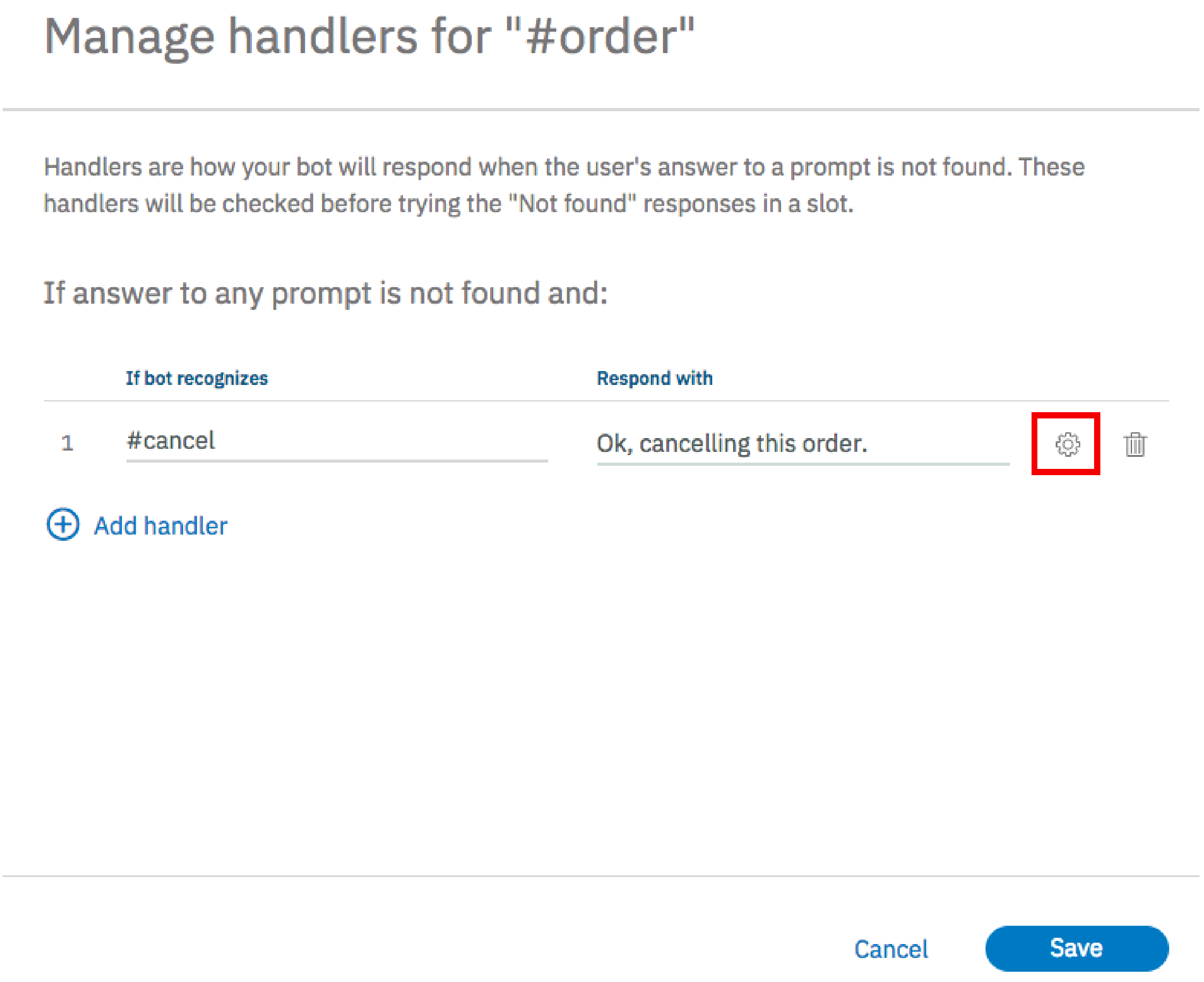
Anything else
anything_else
1 Response / 1 Context set / Does not return

If bot recognizes:
#order

Then check for: Manage handlers

| | Check for | Save it as | If not present, ask | Type |
|---|---|---|---|---|
| 1 | @device | $device | What are you interes | Required |
| 2 | @brand | $brand | What brand do you p | Required |
| 3 | @price | $price | What's your budget li | Required |
| 4 | @sys-number | $number | Enter a prompt | Optional |
| 5 | @confirm | $confirm | Okay $person, I'll orc | Required |

Manage handlers feature



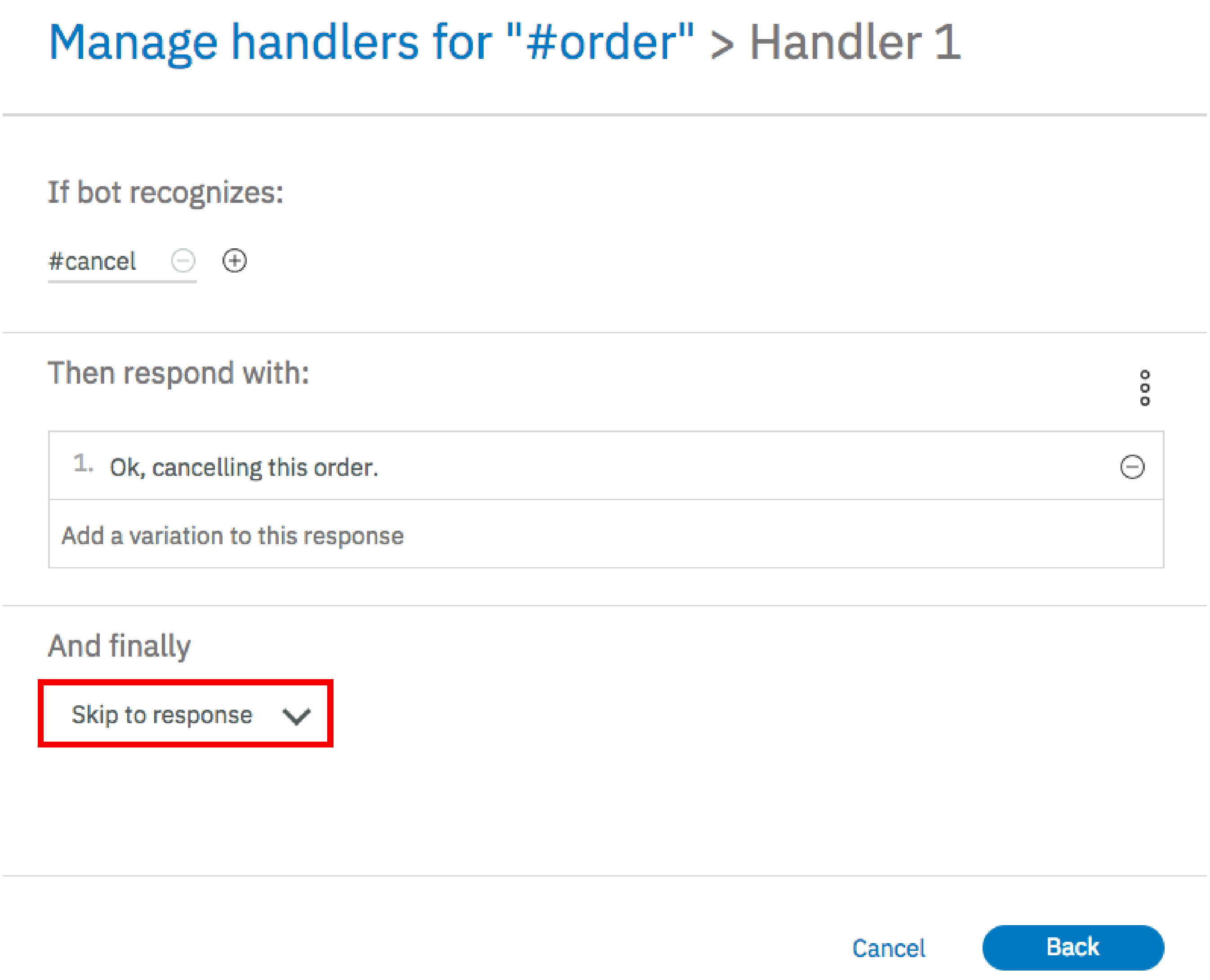
Manage handlers for "#order"

Handlers are how your bot will respond when the user's answer to a prompt is not found. These handlers will be checked before trying the "Not found" responses in a slot.

If answer to any prompt is not found and:

| | If bot recognizes | Respond with |
|---|---|---|
| 1 | #cancel | Ok, cancelling this order. |

Then, click the settings icon, and make sure the bot is skipping to the response. This ensures that the bot will exit the node, instead of collecting more variables.



Manage handlers for "#order" > Handler 1

If bot recognizes:

#cancel

Then respond with:

1. Ok, cancelling this order.

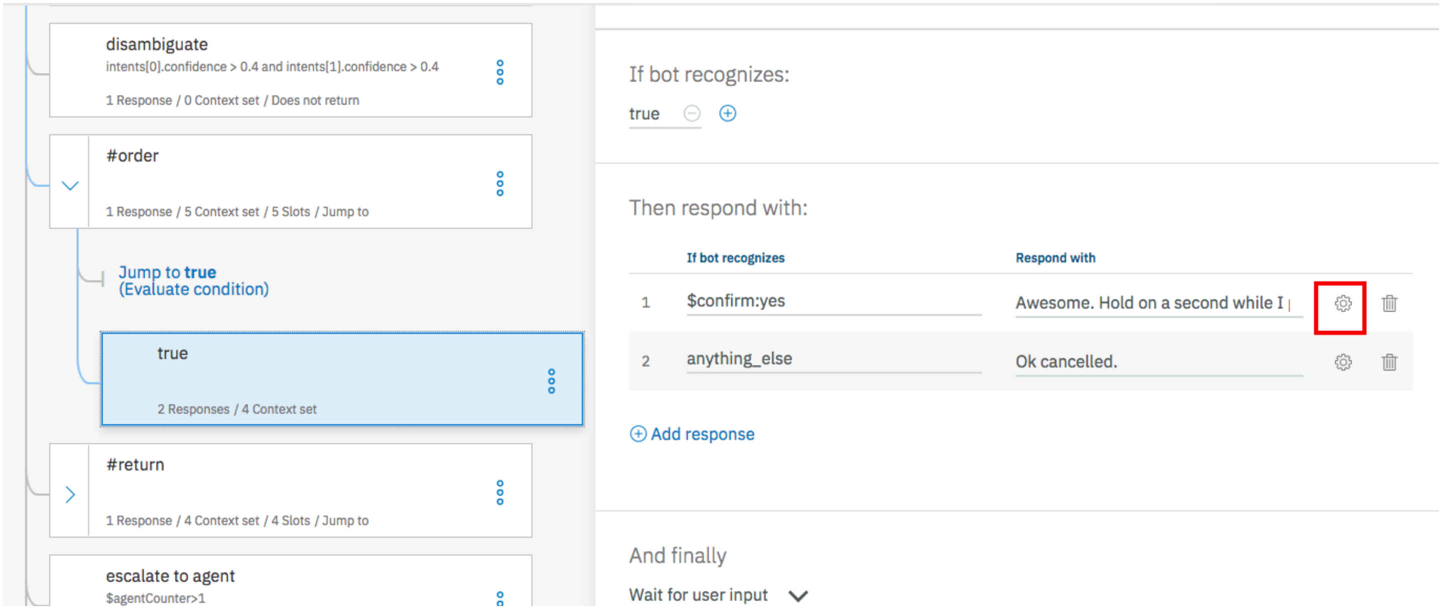
Add a variation to this response

And finally

Skip to response

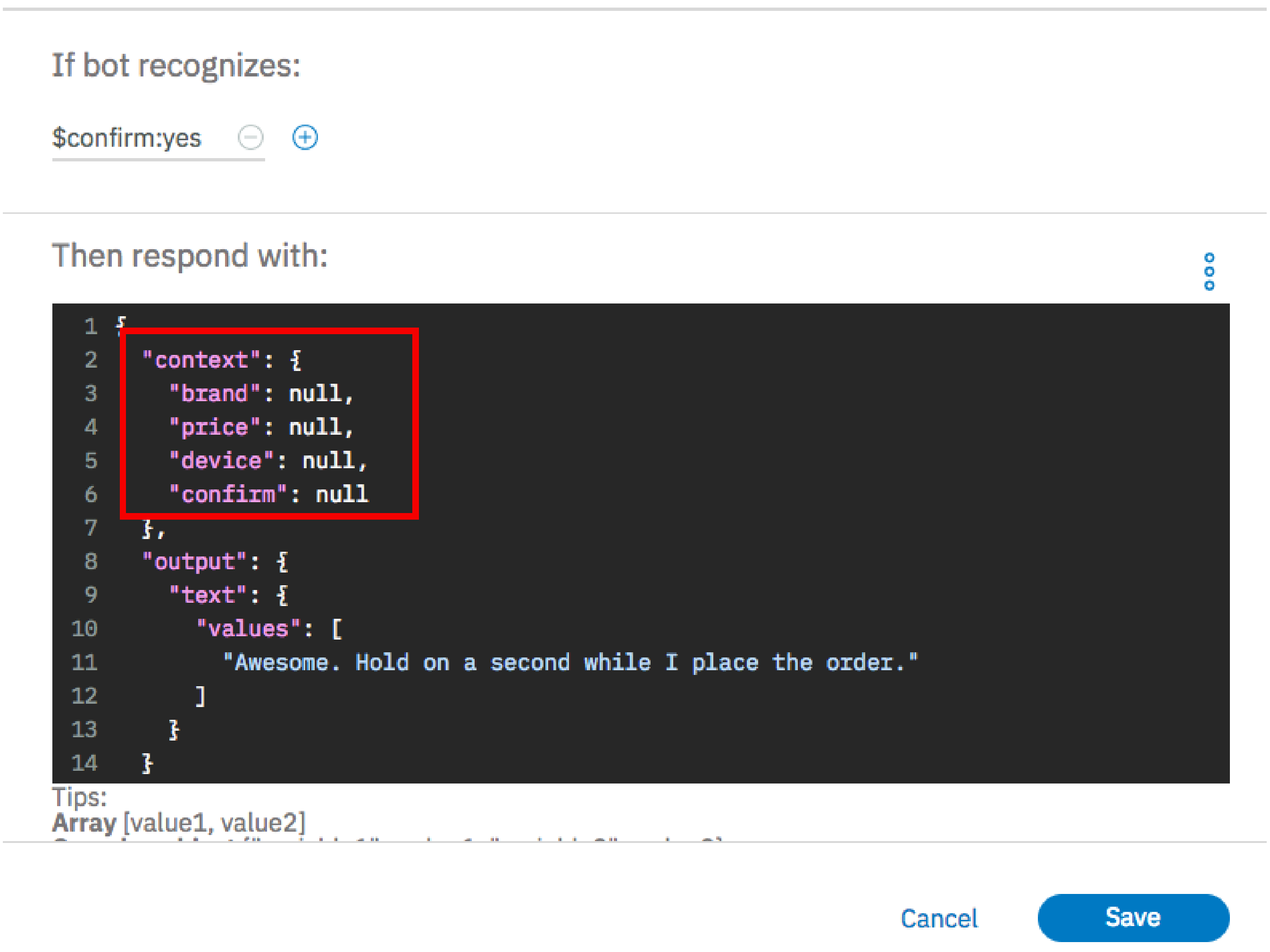
Setting to 'Skip to response'

Finally, you also want to ensure that any variables collected have been reset. This means, if your user was in the process of ordering an Apple phone, the bot has saved those variables `$brand=Apple , $device=phone` and will not collect that information next time, unless otherwise stated. Create a 'Jump to' child node that's set to "true" (so this node is always fired after the #order node).


disambiguate
intents[0].confidence > 0.4 and intents[1].confidence > 0.4
1 Response / 0 Context set / Does not return
#order
1 Response / 5 Context set / 5 Slots / Jump to
Jump to true
(Evaluate condition)
true
2 Responses / 4 Context set
#return
1 Response / 4 Context set / 4 Slots / Jump to
escalate to agent
$agentCounter>1
If bot recognizes:
true
Then respond with:
If bot recognizes
Respond with
1
$confirm:yes
Awesome. Hold on a second while I
2
anything_else
Ok cancelled.
Add response
And finally
Wait for user input

'Jump to' child node set to true

In our example, we've set a confirmation variable, so the bot can confirm with the user at the end of a flow. Open the json editor and set the context variables to "null".



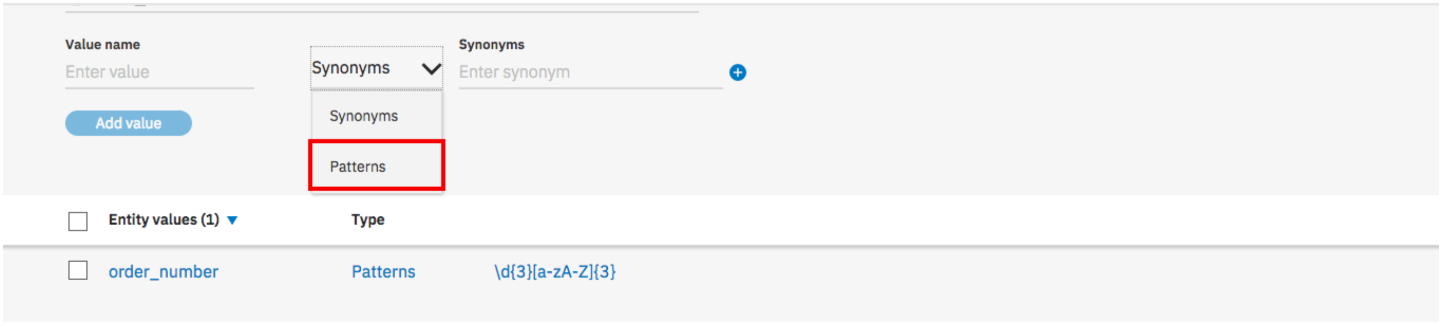
```
{
  "context": {
    "brand": null,
    "price": null,
    "device": null,
    "confirm": null
  },
  "output": {
    "text": {
      "values": [
        "Awesome. Hold on a second while I place the order."
      ]
    }
  }
```

JSON editor (click three buttons on the right side if not open)

### Pattern entities

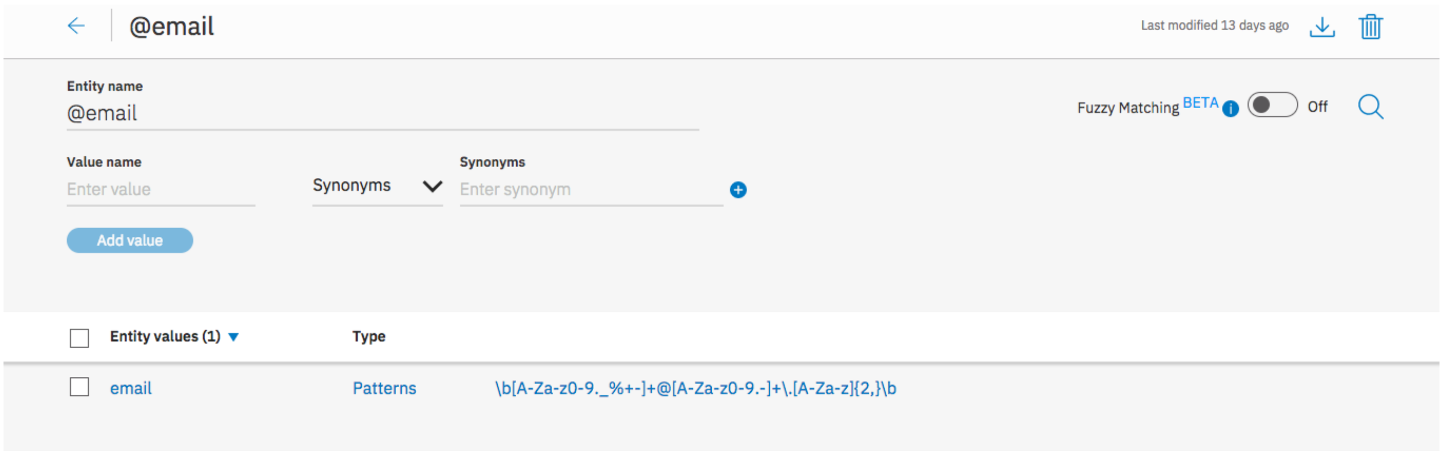
In several use cases, such as this bot, some answers require alpha-numeric text or symbols (i.e. emails, order numbers, etc). Instead of training the values for every possible combination, pattern entities allows you to set the pattern in regular expression and then able to detect that value. In our example, we created a pattern for an order number and email.

Go to the entities section of the tooling, enter the value and choose ‘Patterns’.



| Entity values (1) | Type | |
|---|---|---|
| order_number | Patterns | \d{3}[a-zA-Z]{3} |

Pattern entity for order number



@email

| Entity values (1) | Type | |
|---|---|---|
| email | Patterns | \b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}\b |

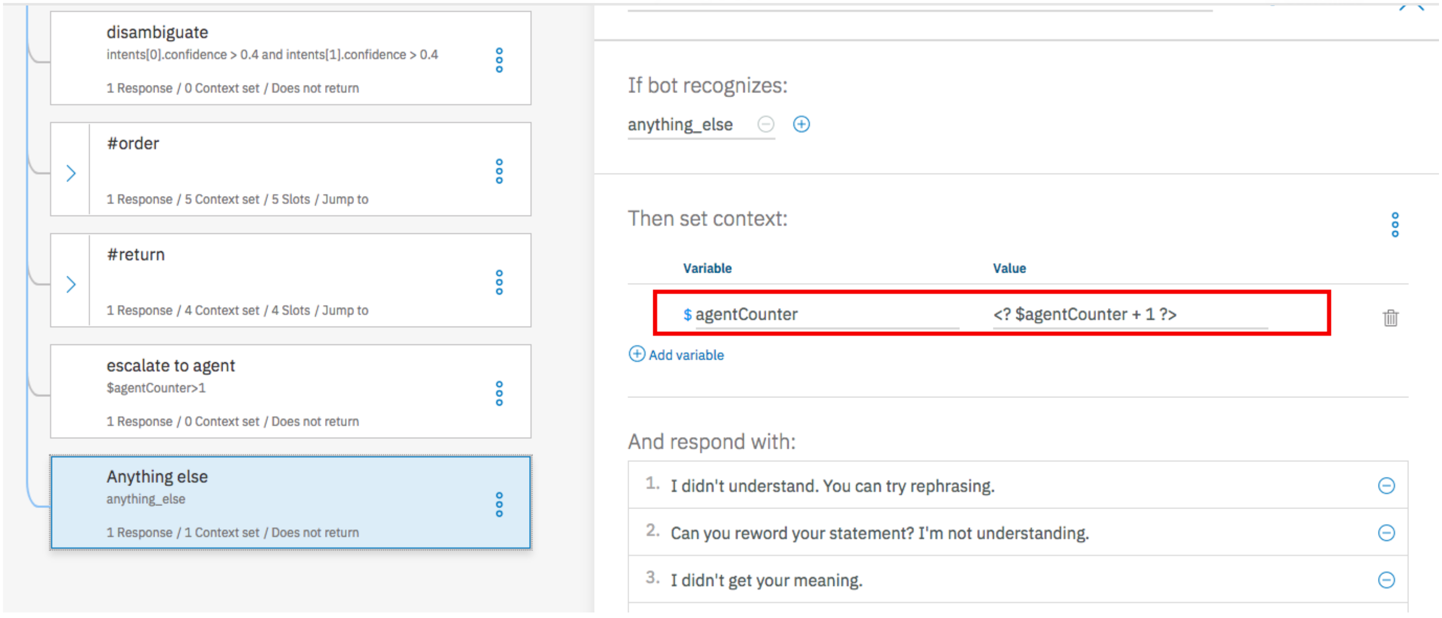
Pattern entity for email

If you'd like to store a pattern entity as a context variable, you want to save the value found rather than the expression. To do so, just add ".literal" syntax after the variable, for example, "@email.literal"

## Counter

Many times, a user will ask or say something completely off topic or the bot simply does not have the capability to understand. Rather than the bot repeatedly respond with the same responses like "I don't understand" or "Can you rephrase," you can have your bot know when it's the right time to pass off to a live agent or end the conversation. Adding a counter tells the bot that if it doesn't understand the user after a certain amount of times, escalate to an agent or end the conversation.

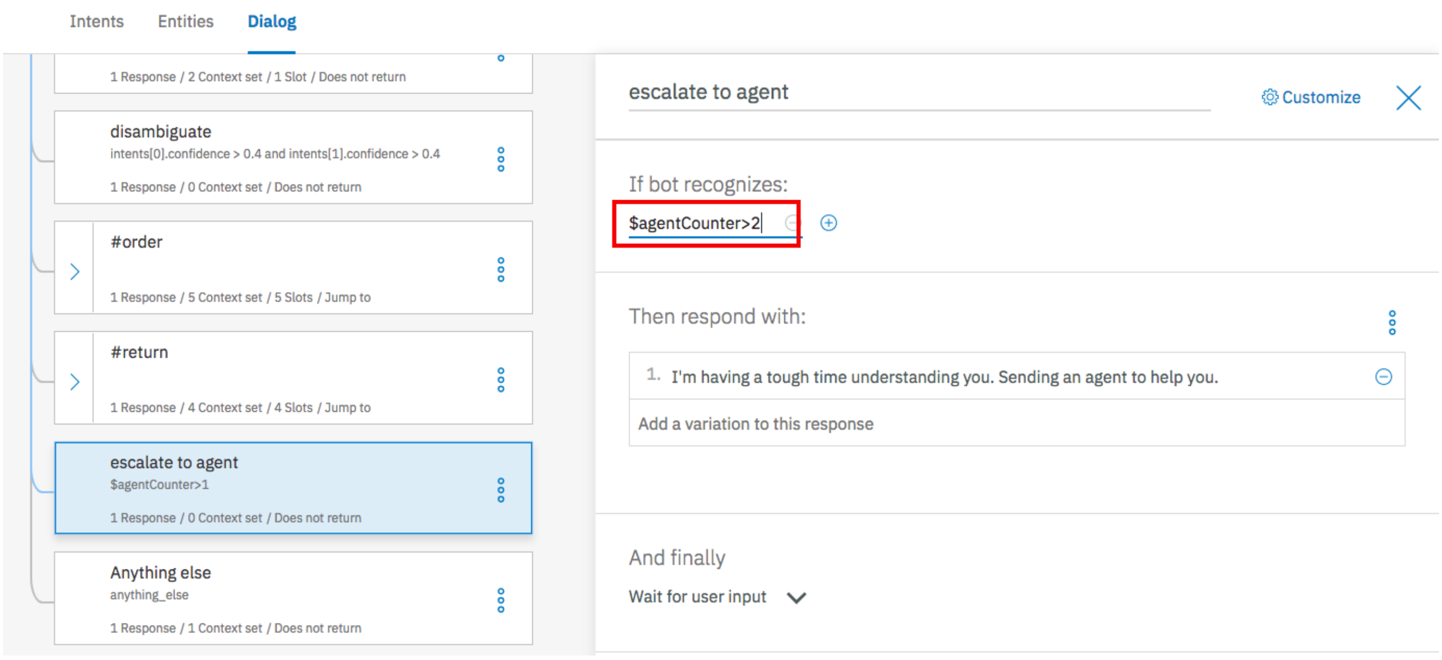
In dialog, create a context variable in the node you want to count from. In the example, we added the "agentCounter" (can be named anything) variable in the Anything else node.



disambiguate
intents[0].confidence > 0.4 and intents[1].confidence > 0.4
1 Response / 0 Context set / Does not return
#order
1 Response / 5 Context set / 5 Slots / Jump to
#return
1 Response / 4 Context set / 4 Slots / Jump to
escalate to agent
$agentCounter>1
1 Response / 0 Context set / Does not return
Anything else
anything_else
1 Response / 1 Context set / Does not return
If bot recognizes:
anything_else
Then set context:
Variable
Value
$ agentCounter
<? $agentCounter + 1 ?>
Add variable
And respond with:
1. I didn't understand. You can try rephrasing.
2. Can you reword your statement? I'm not understanding.
3. I didn't get your meaning.

Setting context variable in Anything else

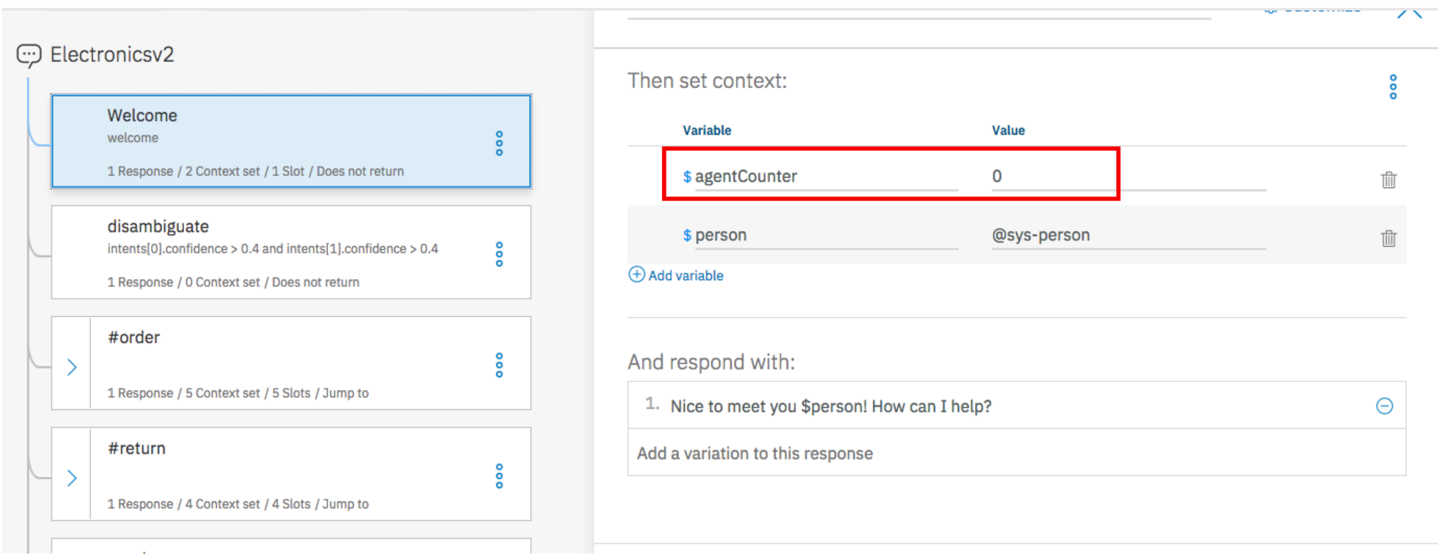
Add a node above (we named it "escalate to agent"), that recognizes if the context variable $agentCounter is above 2, respond accordingly. This number can be set to whatever you'd like.



Intents
Entities
Dialog
1 Response / 2 Context set / 1 Slot / Does not return
disambiguate
intents[0].confidence > 0.4 and intents[1].confidence > 0.4
1 Response / 0 Context set / Does not return
#order
1 Response / 5 Context set / 5 Slots / Jump to
#return
1 Response / 4 Context set / 4 Slots / Jump to
escalate to agent
$agentCounter>1
1 Response / 0 Context set / Does not return
Anything else
anything_else
1 Response / 1 Context set / Does not return
escalate to agent
Customize
If bot recognizes:
$agentCounter>2
Then respond with:
1. I'm having a tough time understanding you. Sending an agent to help you.
Add a variation to this response
And finally
Wait for user input

Escalate to agent node

Finally, we then need to create the counter and set the variable to 0 in the 'Welcome' node.


Electronicsv2
Welcome
welcome
1 Response / 2 Context set / 1 Slot / Does not return
disambiguate
intents[0].confidence > 0.4 and intents[1].confidence > 0.4
1 Response / 0 Context set / Does not return
#order
1 Response / 5 Context set / 5 Slots / Jump to
#return
1 Response / 4 Context set / 4 Slots / Jump to
Then set context:
Add variable
And respond with:
Add a variation to this response

| Variable | Value |
|---|---|
| $ agentCounter | 0 |
| $ person | @sys-person |

1. Nice to meet you $person! How can I help?

Adding agentCounter to Welcome node

While this post reviews how to use these features, make sure to watch the video for more details.

Thanks to Chris Desmarais.

Bots Watson Conversation Watson Editorial Watson Assistant