

IBM Watson

# Building Conversational Solutions

## Building a Dialog

Aco Vidović



## Agenda

- Dialog & node overview
- Folders
- Dialog Turn
- Digressions
- Node Conditions
- Node responses
- Slots
- JSON editor
- IBM Cloud Function

# Getting started / anything\_else

Intents Entities Dialog

Add node

test

Welcome  
welcome

1 Response / Context set

Anything else  
anything\_else

1 Response / Context set

The screenshot shows the IBM Watson Dialog interface. At the top, there are tabs for 'Intents', 'Entities', and 'Dialog', with 'Dialog' being the active tab. Below the tabs is a green button labeled 'Add node'. The main area displays two nodes. The first node is labeled 'test' and contains the text 'Welcome' and 'welcome', with a note below stating '1 Response / Context set'. The second node is labeled 'Anything else' and contains the text 'anything\_else', also with a note below stating '1 Response / Context set'. Each node has a small icon with three dots in the bottom right corner.

# Dialog refresh

Watson Conversation / ConvWks-170601-LV / Build

Intents Entities Dialog

Add node Add child node

ConvWks-170601-LV

Node

start of the conversation  
conversation\_start

3 Responses / 0 Context set / Jump to

Trigger / condition

Jump to welcome (Evaluate responses)

welcome

1 Response / 0 Context set

Greeting  
#greeting

1 Response / 0 Context set

Eat\_Contextual  
#eat

0 Responses / 1 Context set / Jump to

start of the conversation

If bot recognizes:  
conversation\_start

Then respond with:

now().before('12:00:00')

1. Good Morning!

Response

Add a variation to this response

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

1. Good Afternoon!

The screenshot shows the IBM Watson Conversation interface. On the left, there's a sidebar with icons for Intents, Entities, and Dialog. Below it, a tree view of a dialog flow under 'ConvWks-170601-LV'. A specific node, 'start of the conversation' (triggered by 'conversation\_start'), is highlighted with a red box and labeled 'Node'. A red arrow points from this node to the 'Trigger / condition' section on the right. The 'Trigger / condition' section contains the expression 'now().before('12:00:00')'. Below it, a response is listed: '1. Good Morning!' with a red arrow pointing to it and the word 'Response' nearby. Another response section below it has the expression 'now().after('12:00:00')' and the text '1. Good Afternoon!'. The top right of the interface shows standard navigation icons.

# Node description

Name this node... **Node Name:** Name your node in this field.

If bot recognizes:

Enter an intent, entity or context variable...  **Condition:** Enter the main condition in this field.

Then respond with:

**Advanced editor menu:** Delete a response or open the Json editor for the response field.

Text

Move:   

Enter response text **Response:** Enter a response for the condition in this field.

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

And finally

Wait for user input 

**Next action menu:** Drop down menu to choose which action Watson will take next after it returns a response to the user.

## Customize "Greeting"

Slots 



Enable this to gather the information your virtual assistant 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 



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

## Customize "Greeting"

Default digressions settings apply to this node 

Digressions can come into this node 

Allow digressions into this node

 Users can digress to this node from other dialog flows.



Return after digression

 After this dialog flow is processed, return to the dialog flow that was previously in progress.

Name this node...

[Customize](#)

If bot recognizes:

Enter an intent, entity or context variable...

### Slots Frame

Then check for:

[Manage handlers](#)

Check for	Save it as	If not present, ask	Type	
1 Enter entity or intent	Enter context variable	Enter a prompt	Optional	

[+ Add slot](#)

### Multi responses frame

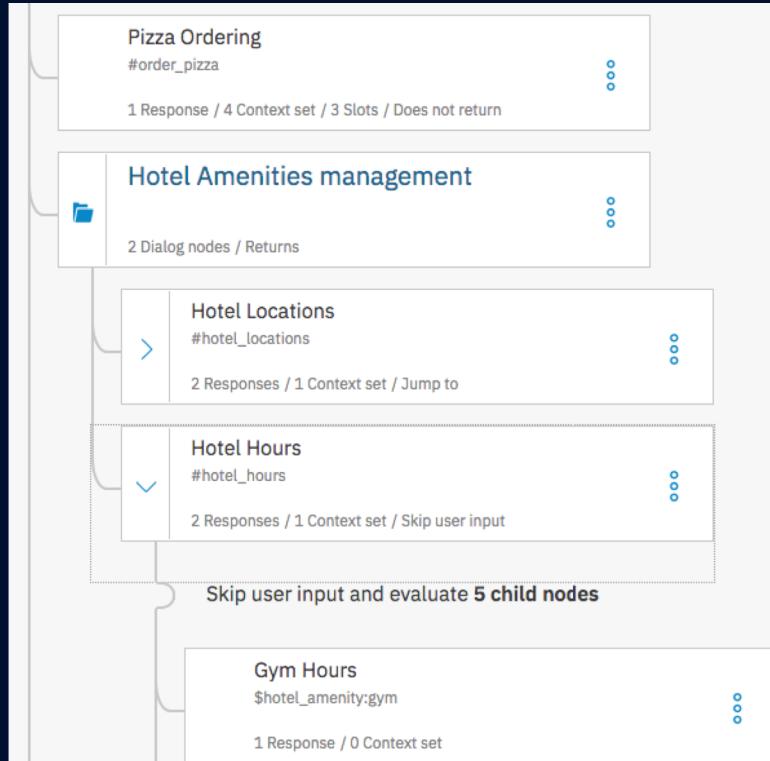
Then respond with:

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

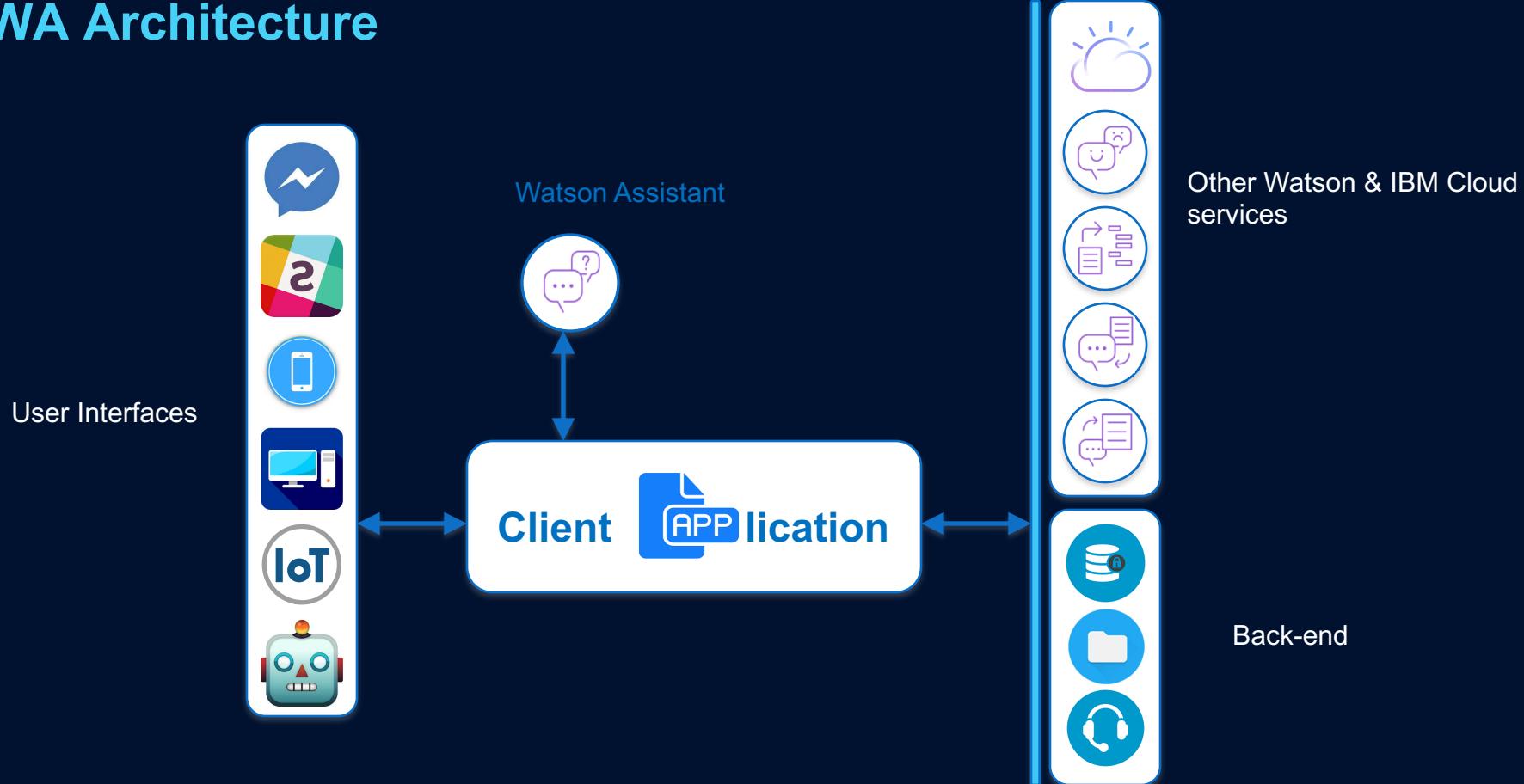
[+ Add response](#)

## Folders

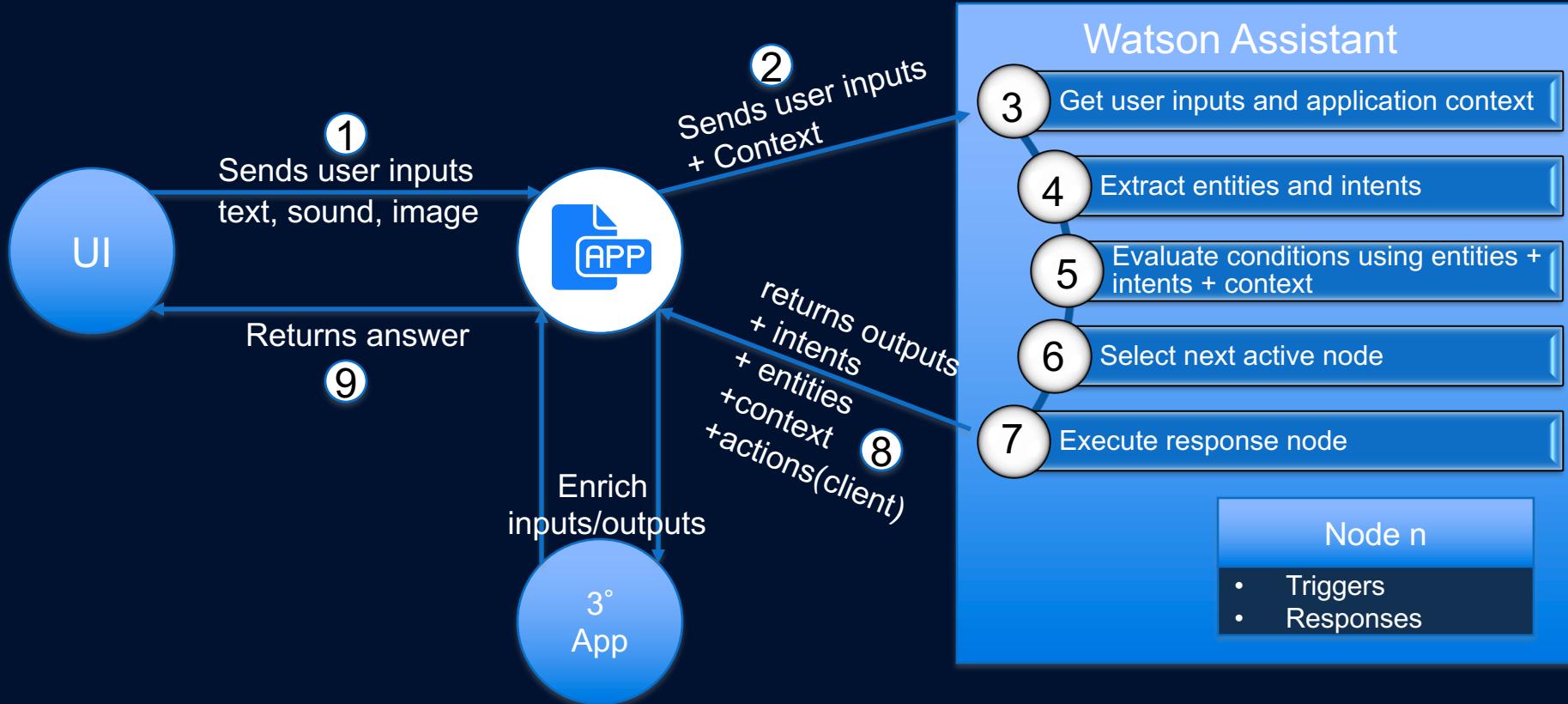
- Allow a dialog designer to organize content based on topics
- Much easier dialog tree navigation and understanding
- Allow performing of bulk setting of node settings at the folder level instead of one by one
- Easier separation of duties for multiple people working on the same bot



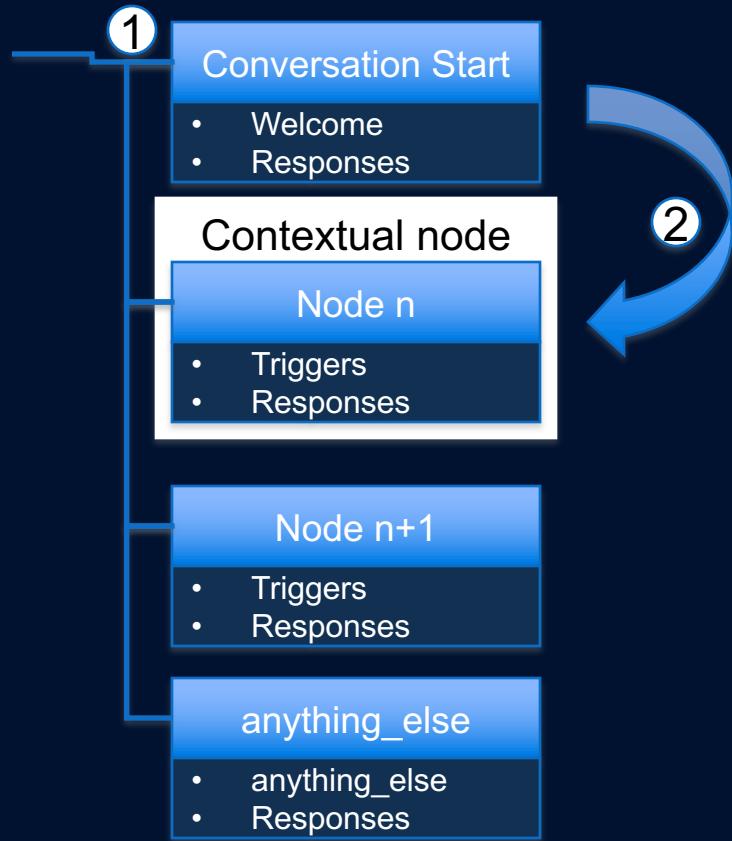
# WA Architecture



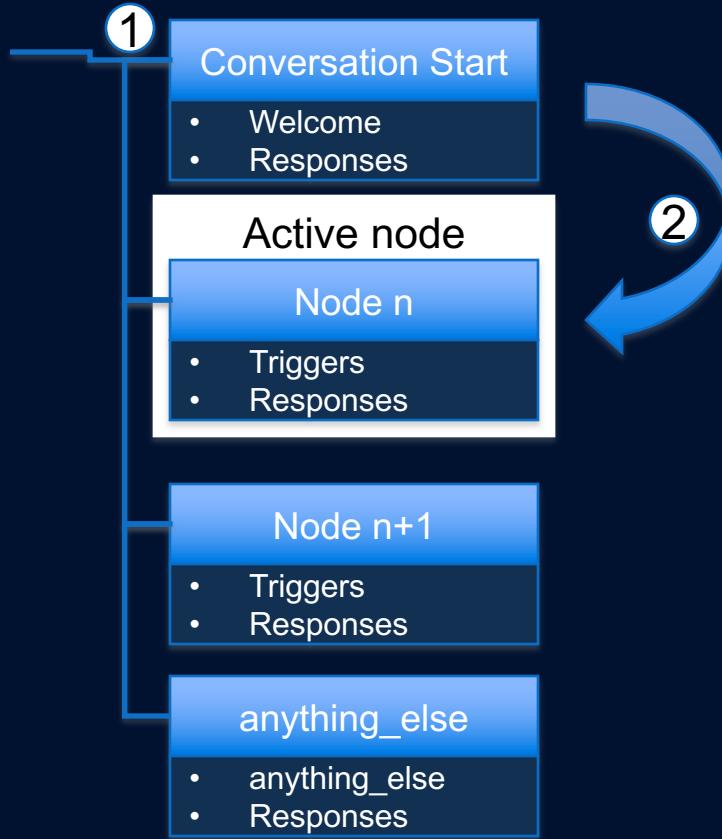
## Conversation turn



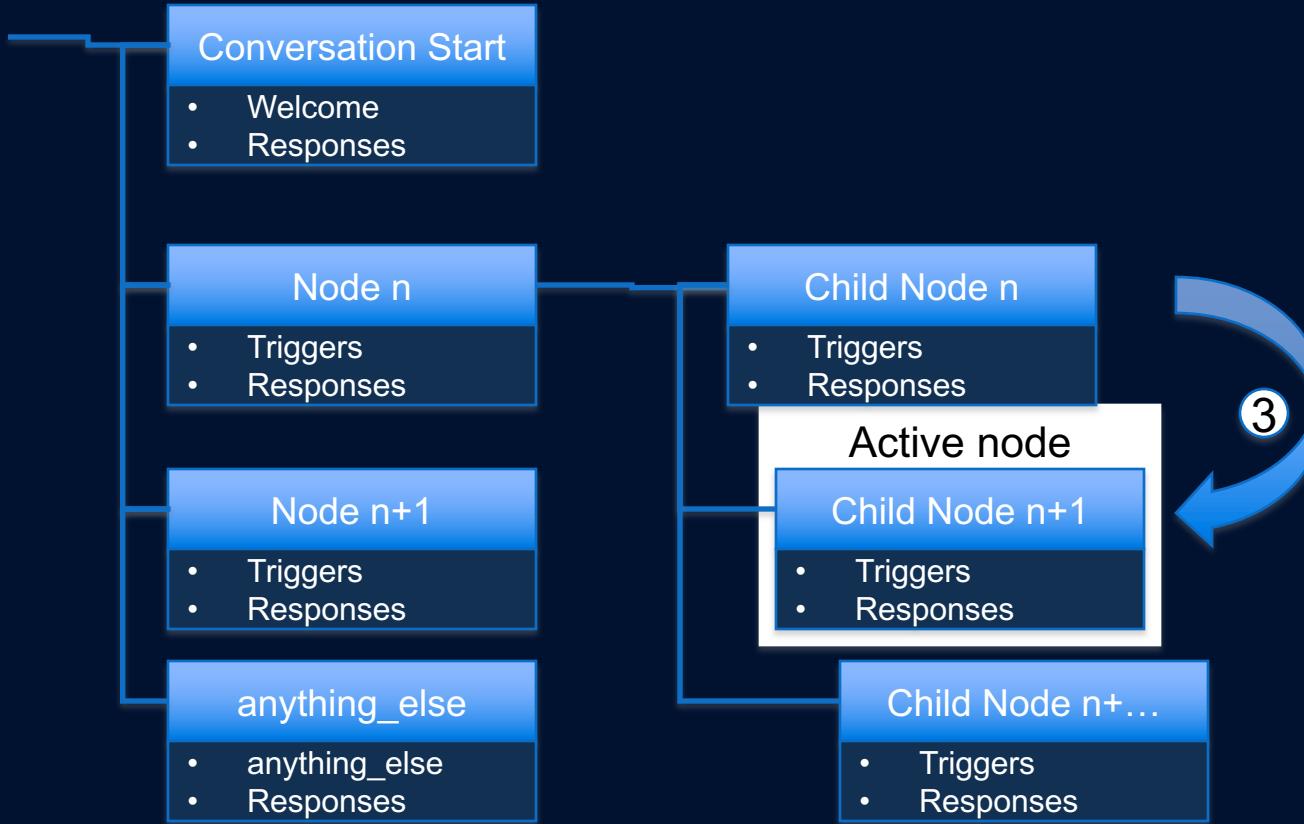
## Dialog flow



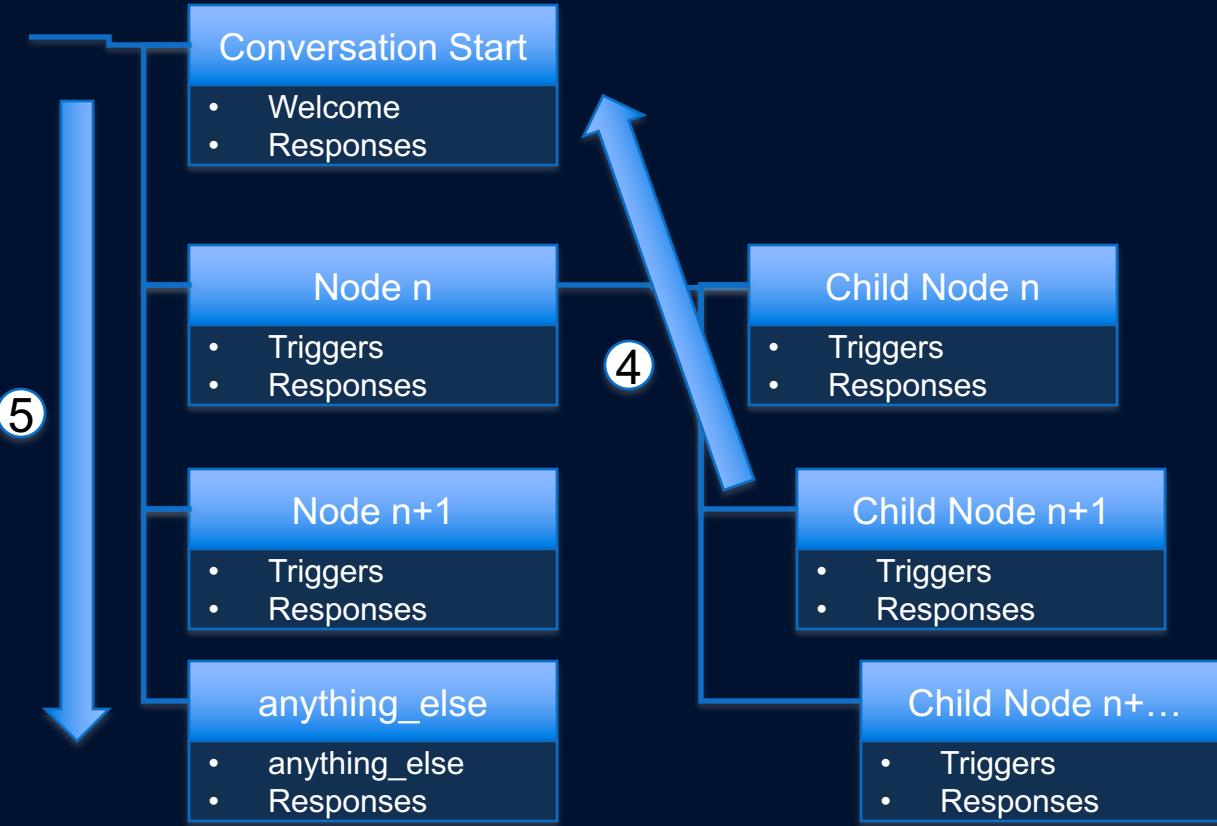
## Dialog flow



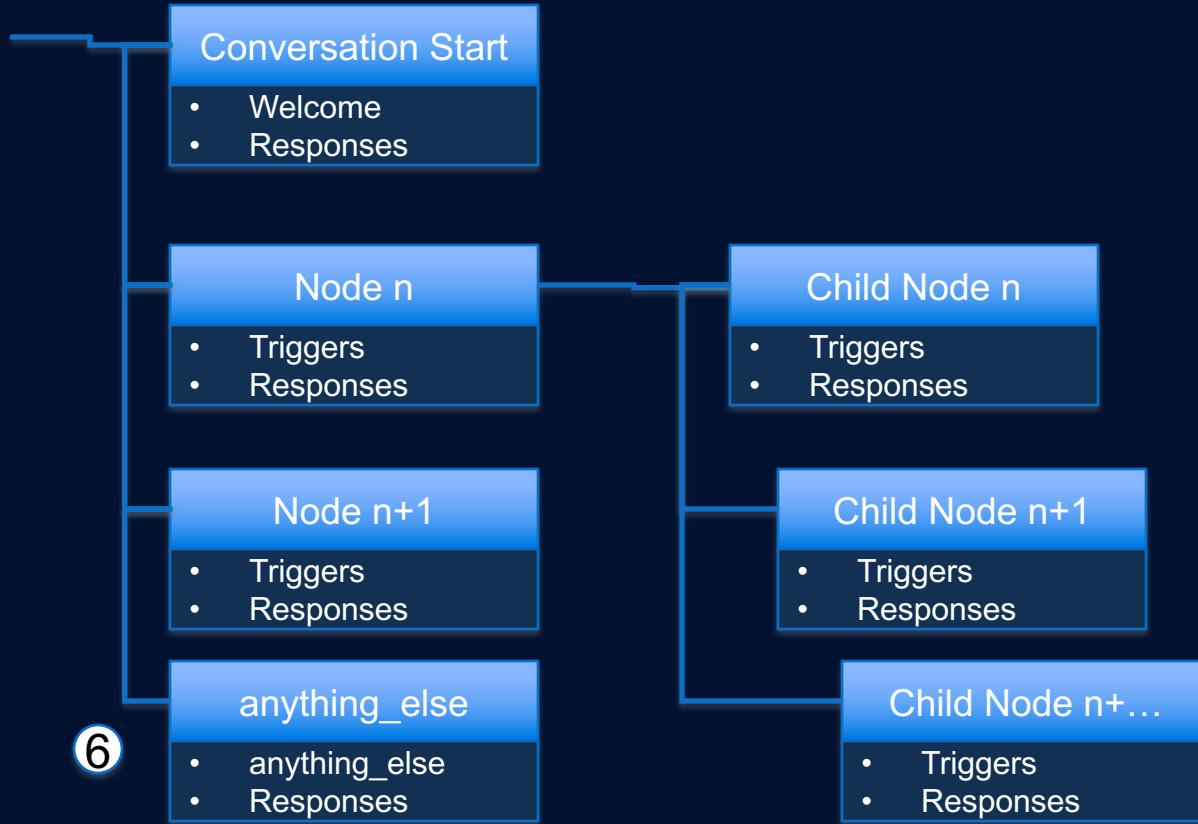
## Dialog flow



## Dialog flow



## Dialog flow



# Digressions

- Allow a user to naturally jump out of a process and seamlessly return back
- Typical use cases
  - Asking side questions while going through a process
  - Chit chat during an unrelated conversation
  - Changing topics without abandoning high value processes

Try it out    Clear    Manage Context 6 X

| Good Afternoon!

| I am Watson, nice to meet you

| The forecast today in Nice is Partly cloudy. Low 15C. ⬇

I would like to order a pizza  
#order\_pizza  
@restaurant:pizza\_restaurant

What size of pizza do you want? ⬇

by the way, where is the pool?  
#hotel\_locations  
@hotel\_amenity:pool

The pool is on the second floor. ⬇

What size of pizza do you want? ⬇

a large vegetarian  
#order\_pizza  
@pizza\_size:large  
@pizza\_type:vegetarian

vegetarian is a good choice. ⬇

I have you for large vegetarian. Is it correct? ⬇

## Defining a Condition / Trigger

- The simplest condition is a single intent : #intent
- The check for a particular value for an entity: @entity:value
- The check for any value for an entity: @entity
- The check for not existing object : !@entity, !#intent
- The check for context variable: \$variable\_name:value
- Value can be evaluated: @price<100 ; @price AND @price<100
- Conditions can be combined : and, && , or , || , ()
- input.text.matches('^[^\\d]\*[\\d]{11}[^\\d]\*\$')

## Special Conditions / Triggers

- **welcome:** This condition is evaluated as true during the first dialog turn (when the conversation starts), only if the initial request from the application does not contain any user input. It is evaluated as false in all subsequent dialog turns.
- **conversation\_start:** Like **welcome**, this condition is evaluated as true during the first dialog turn, but unlike **welcome**, it is true whether or not the initial request from the application contains user input.
- **anything\_else:** You can use this condition at the end of a dialog, to be processed when the user input does not match any other dialog nodes.
- **irrelevant:** This condition will evaluate to true if the user's input is determined to be irrelevant by the Conversation service.
- **true:** This condition is always evaluated to true. You can use it at the end of a list of nodes or responses to catch any responses that did not match any of the previous conditions.
- **false:** This condition is always evaluated to false. You might use this at the top of a branch that is under development, to prevent it from being used, or as the condition for a node that provides a common function and is used only as the target of a **Jump to** action.

## Multi conditions / triggers

If bot recognizes:

\$restaurant  

If bot recognizes:

!\$restaurant  and  (( \$counter && \$counter < 3 ) || !\$counter)  

# Multi conditions / triggers

Hotel Hours Customize X

If bot recognizes:  
#hotel\_hours ⊖ +

Then respond with:

@hotel\_amenity:gym ⊖ +

1. The hotel gym is open from 5am to 2pm. ⊖

Add a variation to this response

@hotel\_amenity:pool ⊖ +

1. The hotel pool is open from 5am to 8pm. ⊖

Add a variation to this response

@hotel\_amenity:(hotel restaurant) ⊖ +

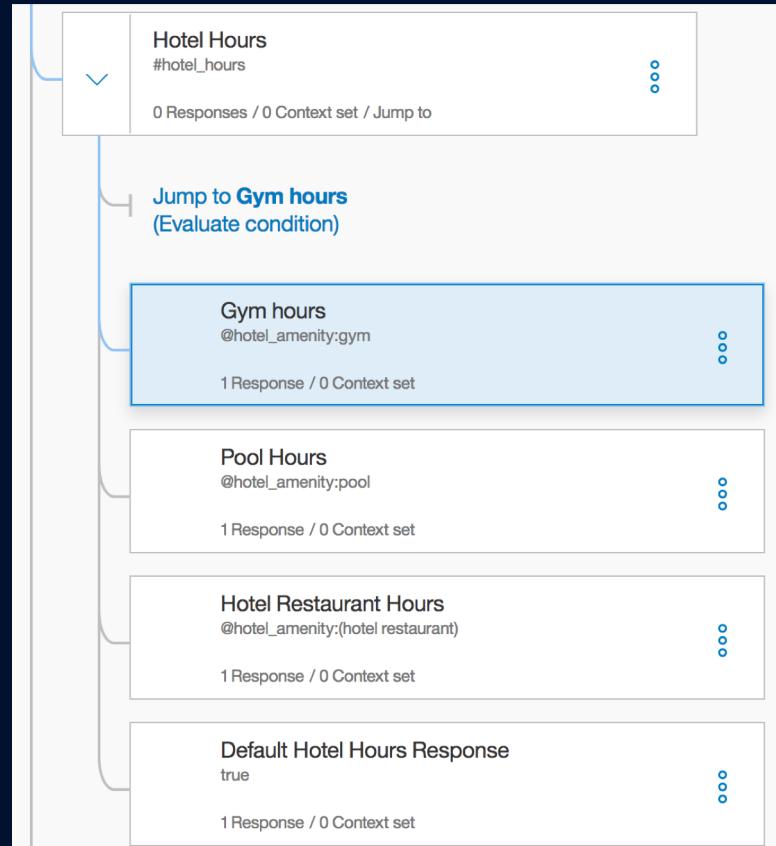
1. The hotel restaurant is open from 6am to 9pm. ⊖

Add a variation to this response

4 true ⊖ +

1. specifically what hours you are looking for, or you may call the front desk for more information. ⊖

OR



# Using Time Method / Context Variable

Responses ⓘ

if now().before('12:00:00') Ⓛ Ⓜ

1. Good Morning!

Add a variation to this response

if now().after('12:00:00') Ⓛ Ⓜ

1. Good Afternoon!

Add a variation to this response

3 Ⓛ Add response condition

1. Hi!

Add a variation to this response

Fulfill with a response ⓘ

[Jump to...](#)

if \$time == "morning" Ⓛ Ⓜ

1. Good morning

Add a variation to this response

if \$time == "evening" Ⓛ Ⓜ

1. Good evening

Add a variation to this response

if anything\_else Ⓛ Ⓜ

1. Hello

Add a variation to this response

# Defining a response

Responses ⓘ

+ Add response condition

1. OK! Turning on the @appliance

Add a variation to this response

+ Add another response

1. Update the dialog context;
2. Provide a response to the user;
3. What to do next!
  - Standby and **Wait for user input** (default)
  - Execute child nodes with **Skip user input** option
  - Change the flow of the dialog by using a **Jump to** action

## Fulfill with a response ⓘ

[Jump to...](#)

+ Add response condition



1. Hello



2. Hi there



3. Howdy!



Add a variation to this response

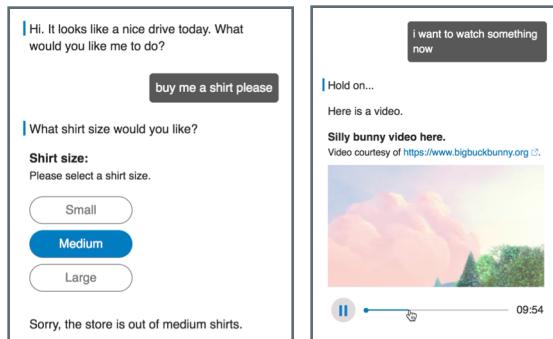
Response variations are **sequential**. [Set to random](#) ⓘ

# Response Types

Enable end user experiences beyond simple text with **response types**

E.g. Buttons, images, videos, pause delays etc.

## End User Experience



## Authoring Experience (Dialog)

The screenshot shows the "Text response" section of the dialog authoring interface. Under "Option list", the "Media" option is selected, highlighted in blue. Other options include "Text response", "Option list", "Pause", "Audio", "Image", "Video", and "Webview". To the right, there's a preview area with the text "you want to check the balance of?". Below the preview, there's a section titled "Then respond with:" containing a "Text response" card for "Which account did you want to check the balance of?", an "Image" card, and a "Webview" card. Each card has fields for "Title (optional)", "Description (optional)", and "URL". At the bottom, there's a "Add response" button.

# Wait for user input

Watson Conversation / ConvWks-171003-LV / Build

Intents Entities Dialog

- #hotel\_locations
- 2 Responses / 1 Context set / Jump to

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

- Pizza Ordering
- #order\_pizza
- 0 Responses / 3 Context set / 3 Slots / Jump to

- Update Profile
- #update\_profile
- 1 Response / 0 Context set

Confirm

true

1 Response / 0 Context set

#feedback

Watson Conversation / ConvWks-171003-LV / Build

Update Profile

If bot recognizes:

#update\_profile

Customize X

Then respond with:

1. Ok. What should I call you?

Add a variation to this response

And finally

Wait for user input

Try it out

Good Morning!

I am Watson, nice to meet you

I woud like to update my profile

#update\_profile

Ok. What should I call you?

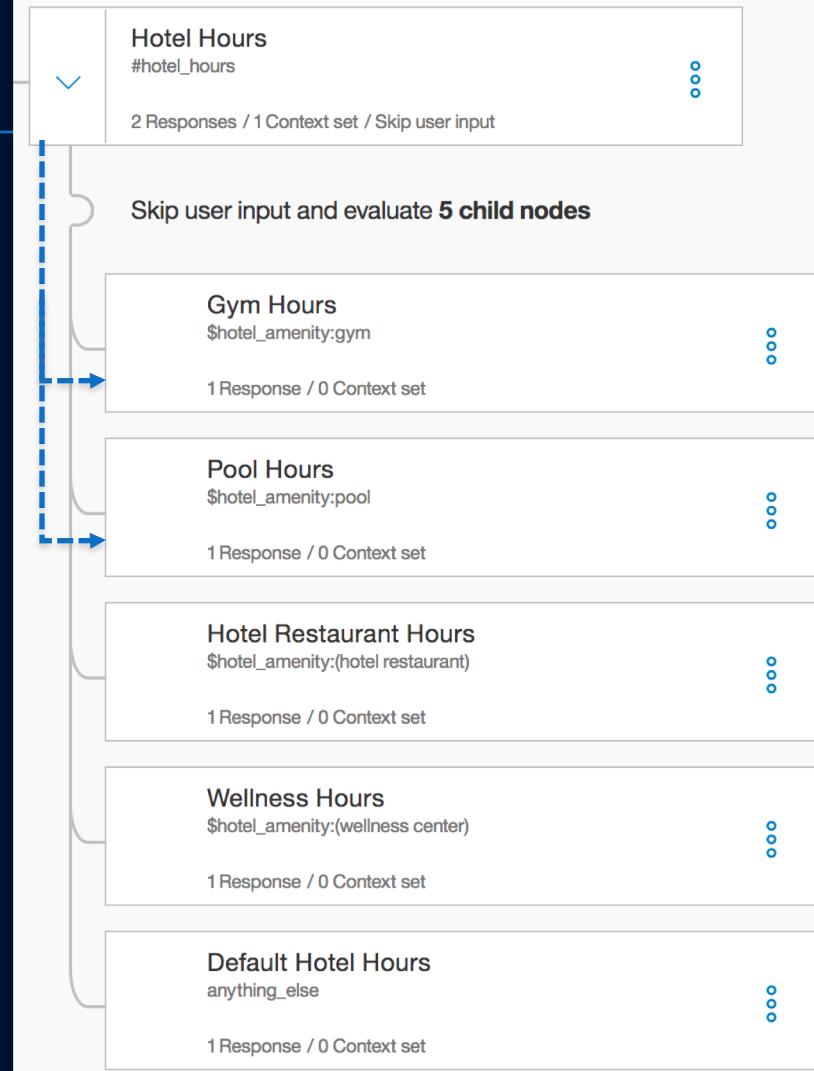
Enter something to test your bot

Use the up key for most recent

```
graph TD; Start(( )) --> Int1[Intent: #update_profile]; Int1 --> Res1[Response: "Ok. What should I call you?"]; Res1 --> Wait[Wait for user input];
```

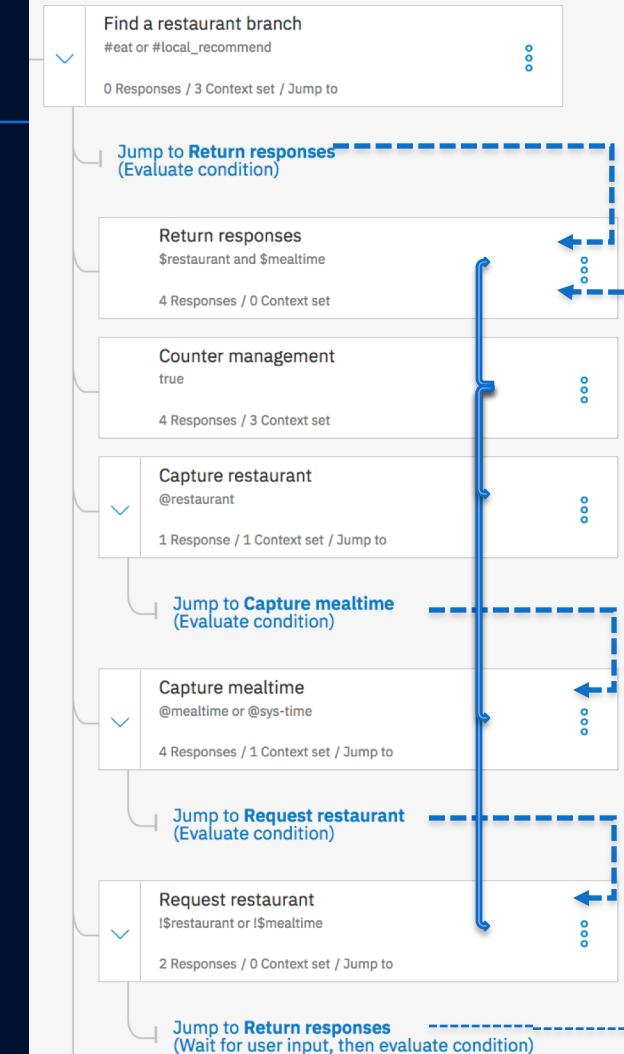
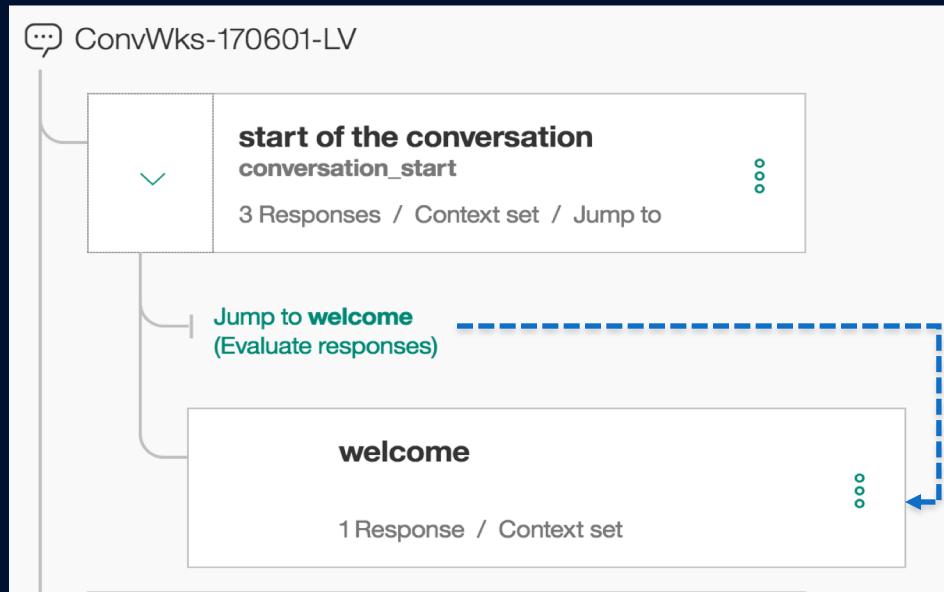
## Skip user input ... acts like a **GoTo** child nodes

- Doesn't wait for user input



## Jump to ... acts like a GoTo

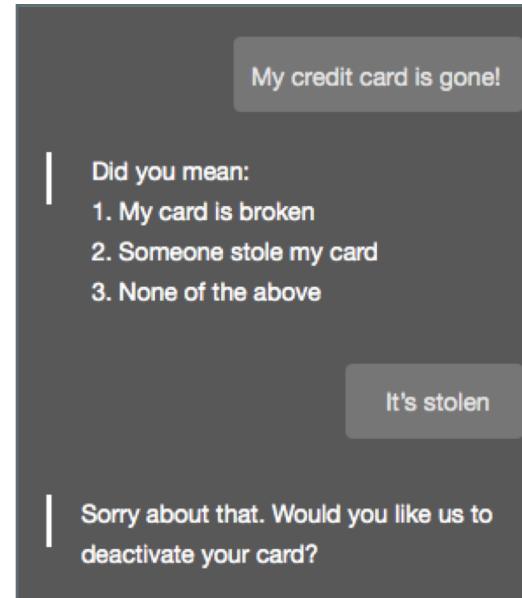
- Condition without waiting for user input
- Response without waiting for user input
- Wait for user input and then condition



## Disambiguation – Premium only

**Disambiguate** a users utterance (Did you mean....)

- End-user picks best option from a list
- Let's your virtual assistant help customers better, which means less transfers over to live agent



## Slots: Ask for everything at once

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.

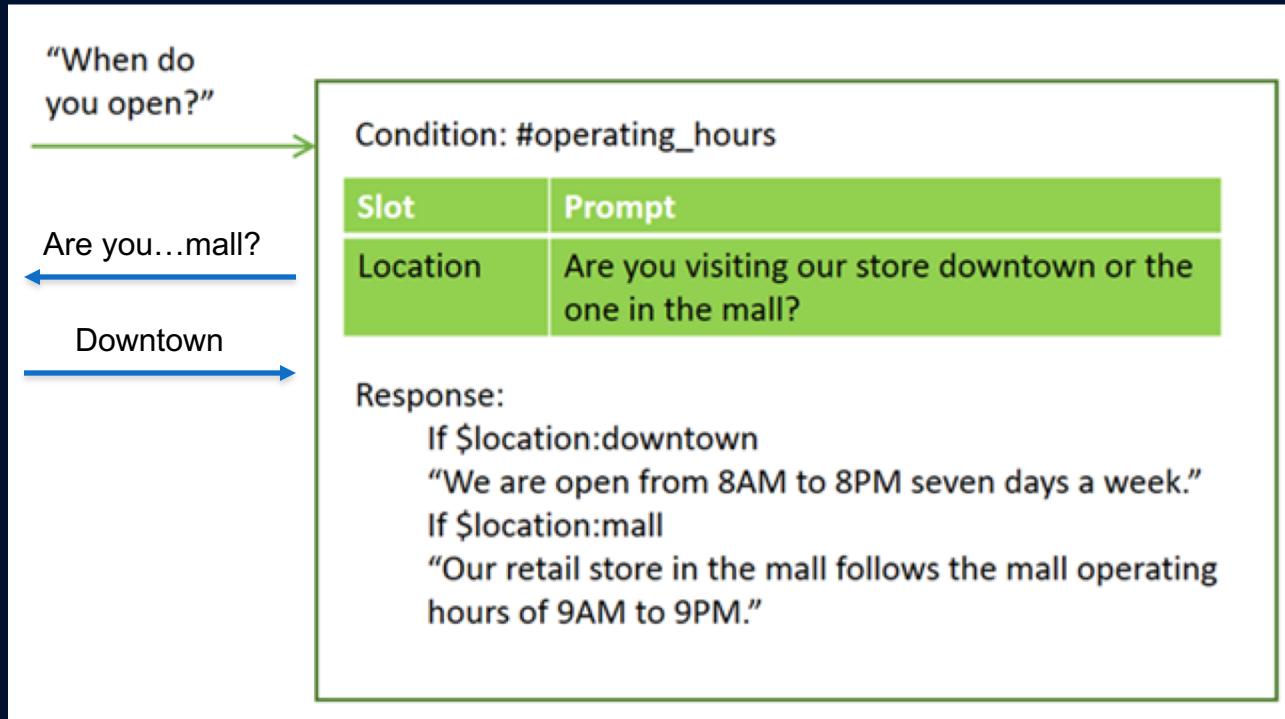
Multiple responses ⓘ

off

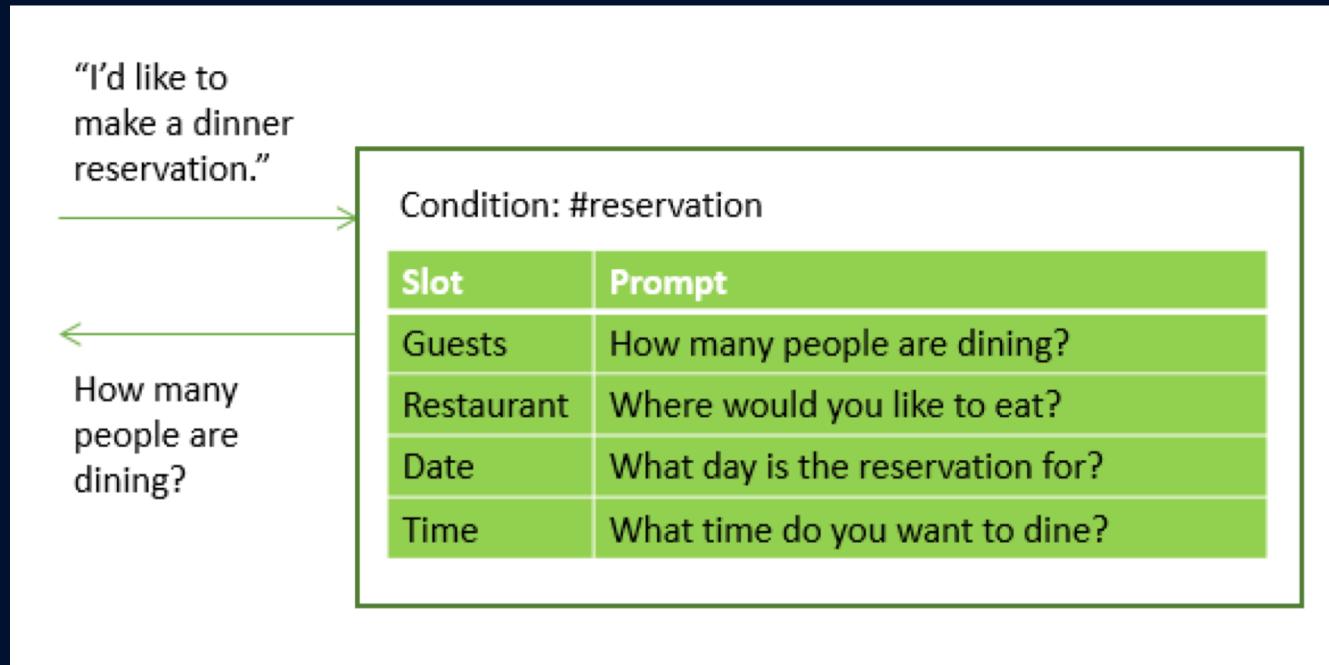
Enable multiple responses so that your bot can provide different responses to the same input, based on other conditions.

[Cancel](#) [Apply](#)

## How slots Work



## Slot: Collect multiple pieces of information



## Slot: Collect multiple pieces of information

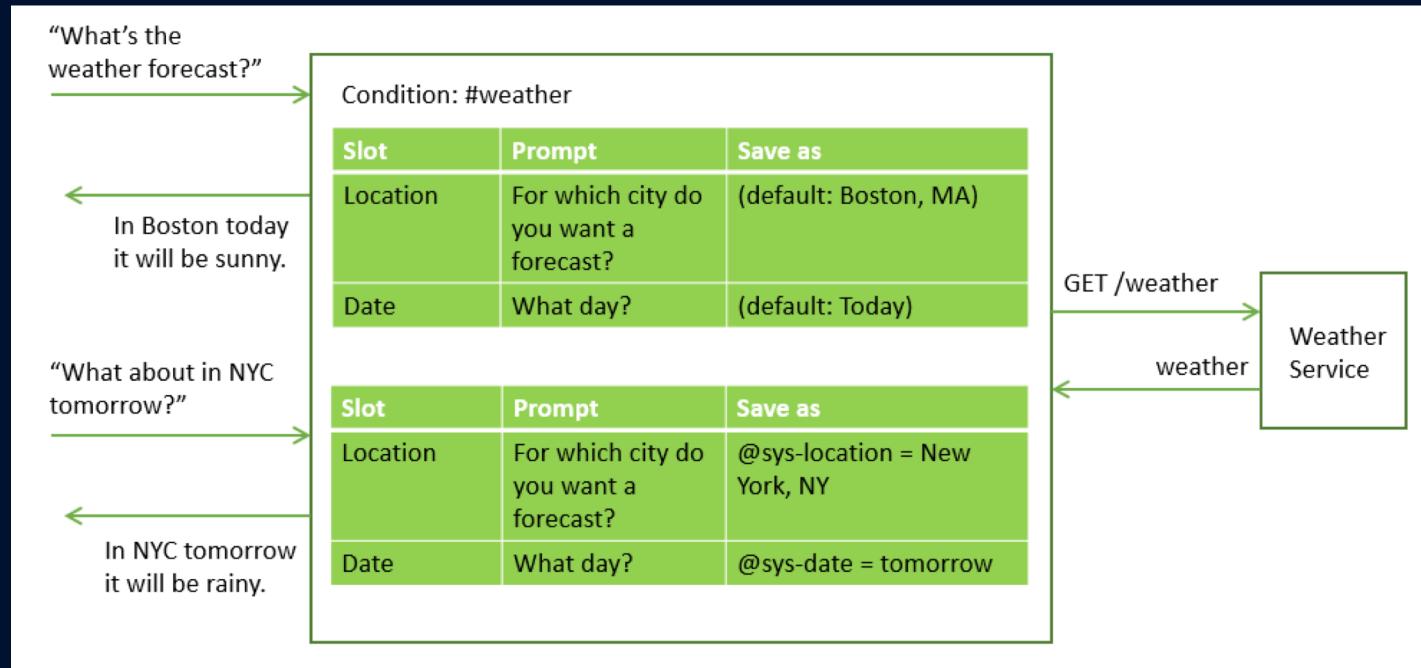
"I'd like to make a reservation.  
There will be **6** of us dining at **7PM**."



Where would you like to eat?

Slot	Prompt	
Guests	How many people are dining?	✓
Restaurant	Where would you like to eat?	
Date	What day is the reservation for?	
Time	What time do you want to dine?	✓

# Update a collected value



## Slot: user inputs and follow up

Information	Check for	Save as	Prompt	Follow-up if found	Follow-up if not found
Size	@size	\$size	"What size pizza would you like?"	"\$size it is."	"What size did you want? We have small, medium, and large."
DeliverBy	@sys-time	\$time	"When do you need the pizza by?"	"For delivery by \$time."	"What time did you want it delivered? We need at least a half hour to prepare it."

## Slot: Optional slot

Information	Check for	Save as	Prompt	Follow-up if found	Follow-up if not found
Size	@size	\$size	"What size pizza would you like?"	"\$size it is."	"What size did you want? We have small, medium, and large."
DeliverBy	@sys-time	\$time	"When do you need the pizza by?"	"For delivery by \$time."	"What time did you want it delivered? We need at least a half hour to prepare it."
Wheat restriction	@dietary	\$dietary	<i>Empty</i>		

## Slot: Multiple Values

Information	Check for	Save as	Prompt	Follow-up if found	Follow-up if not found
Size	@size	\$size	"What size pizza would you like?"	"\$size it is."	"What size did you want? We have small, medium, and large."
DeliverBy	@sys-time	\$time	"When do you need the pizza by?"	"For delivery by \$time."	"What time did you want it delivered? We need at least a half hour to prepare it."
Toppings	@toppings.values	\$toppings	Any toppings on that?	"Great addition."	"What toppings would you like? We offer ..."

## Slot: getting confirmation

Information	Check for	Save as	Prompt	Follow-up if found	Follow-up if not found
Size	@size	\$size	"What size pizza would you like?"	"\$size it is."	"What size did you want? We have small, medium, and large."
DeliverBy	@sys-time	\$time	"When do you need the pizza by?"	"For delivery by \$time."	"What time did you want it delivered? We need at least a half hour to prepare it."
Confirmation	#yes	\$order-confirmed	"I'm going to order you a \$size pizza for delivery at \$time. Should I go ahead?"	"Your pizza is on its way!"	<i>See below</i>

## Slot: Keep Users on track (handlers)

"What's in your  
sauce?"

If: #sauce\_recipe

Then: "It is a secret family recipe passed down  
from generation to generation. And I'll take it to  
my grave."

## Specific slot's Conditions / Triggers

- `slot_in_focus`

You are looking for a response for this slot

- `event.previous_value`

You want to evaluate a change in the context variable of this slot. It associated with `event.current_value`

- `all_slots_filled`

You want to assess that's all slots are filled

## What to do next?

- Found
  - Move on (Default)
  - Clear slot and prompt again
  - Skip to response
- Not found
  - Wait for user input (Default)
  - Prompt again
  - Skip this slot
  - Skip to response
- Handler
  - Prompt again (default)
  - Skip current slot
  - Skip to response

# Advanced Editor: Json editor

```
{  
  "context": {  
    "{any_objects)": " {any_value}"  
  },  
  "actions": [ {  
    "name": "<actionName>",  
    "type": "client | server",  
    "parameters": {  
      "<parameter_name>": "<parameter_value>",  
      "<parameter_name>": "<parameter_value>" },  
    "result_variable": "<result_variable_name>",  
    "credentials": "<reference_to_credentials>"  
  } ],  
  "output": {  
    "text": [ "text one" ],  
    "generic": [ object ],  
    "{action)": " {any_value}",  
  }  
}
```

welcome

If bot recognizes:

true

Then respond with:

Advanced editor menu: Delete a response or open the Json editor for the response field.

⋮

```
1 {  
2   "output": {  
3     "text": {  
4       "values": [  
5         "I am Watson, nice to meet you"  
6       ],  
7       "selection_policy": "sequential"  
8     }  
9   },  
10  "actions": [  
11    {  
12      "name": "/laurent_vincent@fr.ibm.com_Labs/weather",  
13      "type": "server",  
14      "parameters": {
```

# Context

```
{  
  "context": {  
    "{any_objects)": " {any_value}"  
  },  
  "actions": [ {  
    "name": "<actionName>",  
    "type": "client | server",  
    "parameters": {  
      "<parameter_name>": "<parameter_value>",  
      "<parameter_name>": "<parameter_value>" },  
    "result_variable": "<result_variable_name>",  
    "credentials": "<reference_to_credentials>"  
  } ],  
  "output": {  
    "text": [ "text one" ],  
    "generic": [ object ],  
    "{action)": " {any_value}",  
  }  
}
```

Context is what makes your bot more conversational

- Repeats
- Turn Counters
- Capturing information
- Analytics

Within Context, you can by using SpEL and Regex

- Manipulate info
- Increment/decrement values
- Reformat
- Simple Math
- Manage Array

## Call to actions

```
{  
  "context": {  
    "{any_objects)": " {any_value}"  
  },  
  "actions": [ {  
    "name": "<actionName>",  
    "type": "client | server",  
    "parameters": {  
      "<parameter_name>": "<parameter_value>",  
      "<parameter_name>": "<parameter_value>" },  
    "result_variable": "<result_variable_name>",  
    "credentials": "<reference_to_credentials>"  
  }],  
  "output": {  
    "text": [ "text one" ],  
    "generic": [ object ],  
    "{action)": " {any_value}",  
  }  
}
```

Actions enables a smooth integration

- Callout to cloud functions
- Callout to any APIs

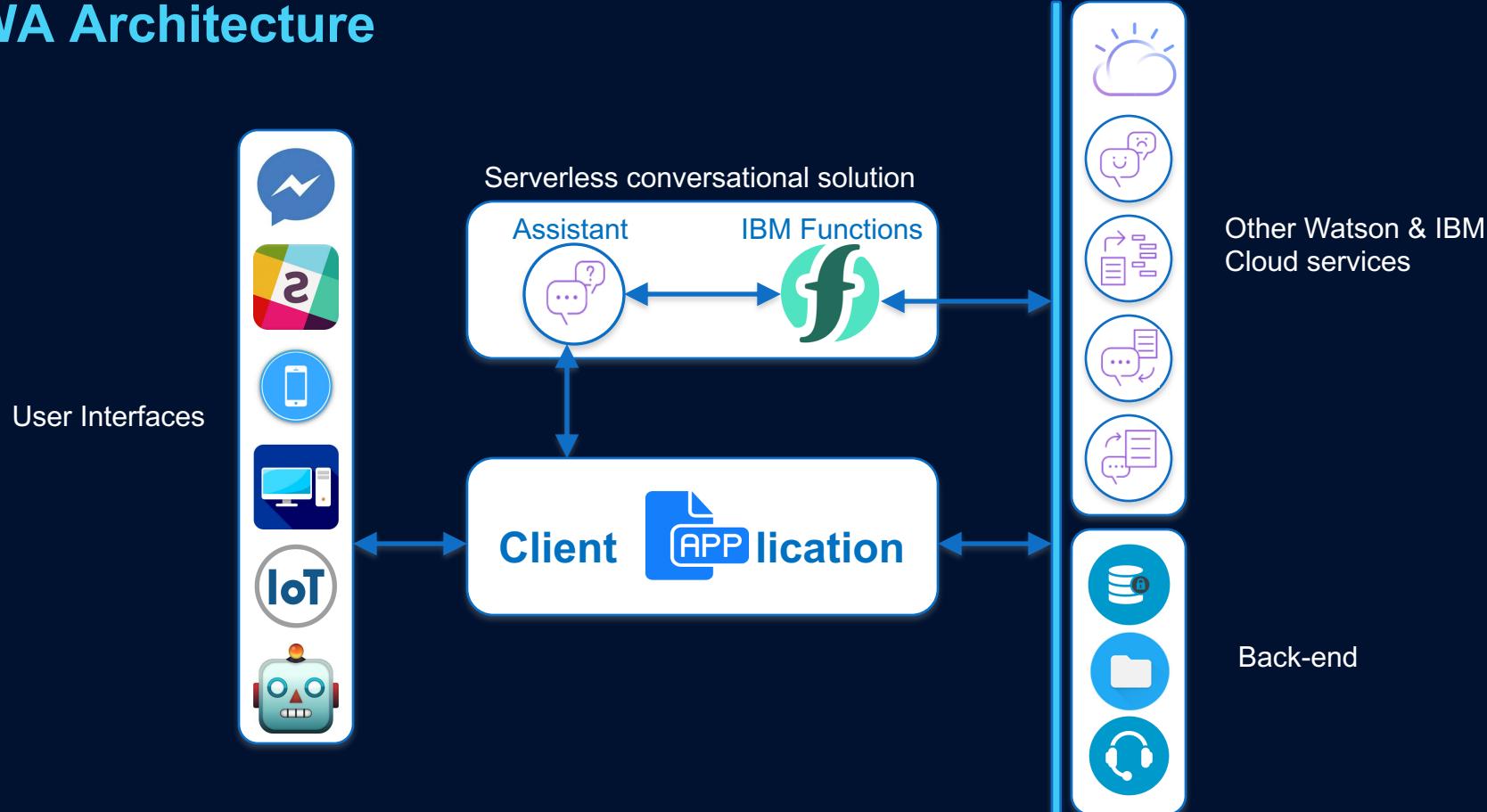
# Context

```
{  
  "context": {  
    "{any_objects)": " {any_value}"  
  },  
  "actions": [ {  
    "name": "<actionName>",  
    "type": "client | server",  
    "parameters": {  
      "<parameter_name>": "<parameter_value>",  
      "<parameter_name>": "<parameter_value>" },  
    "result_variable": "<result_variable_name>",  
    "credentials": "<reference_to_credentials>"  
  } ],  
  "output": {  
    "text": [ "text one" ],  
    "generic": [ object ],  
    "{action)": " {any_value}",  
  }  
}
```

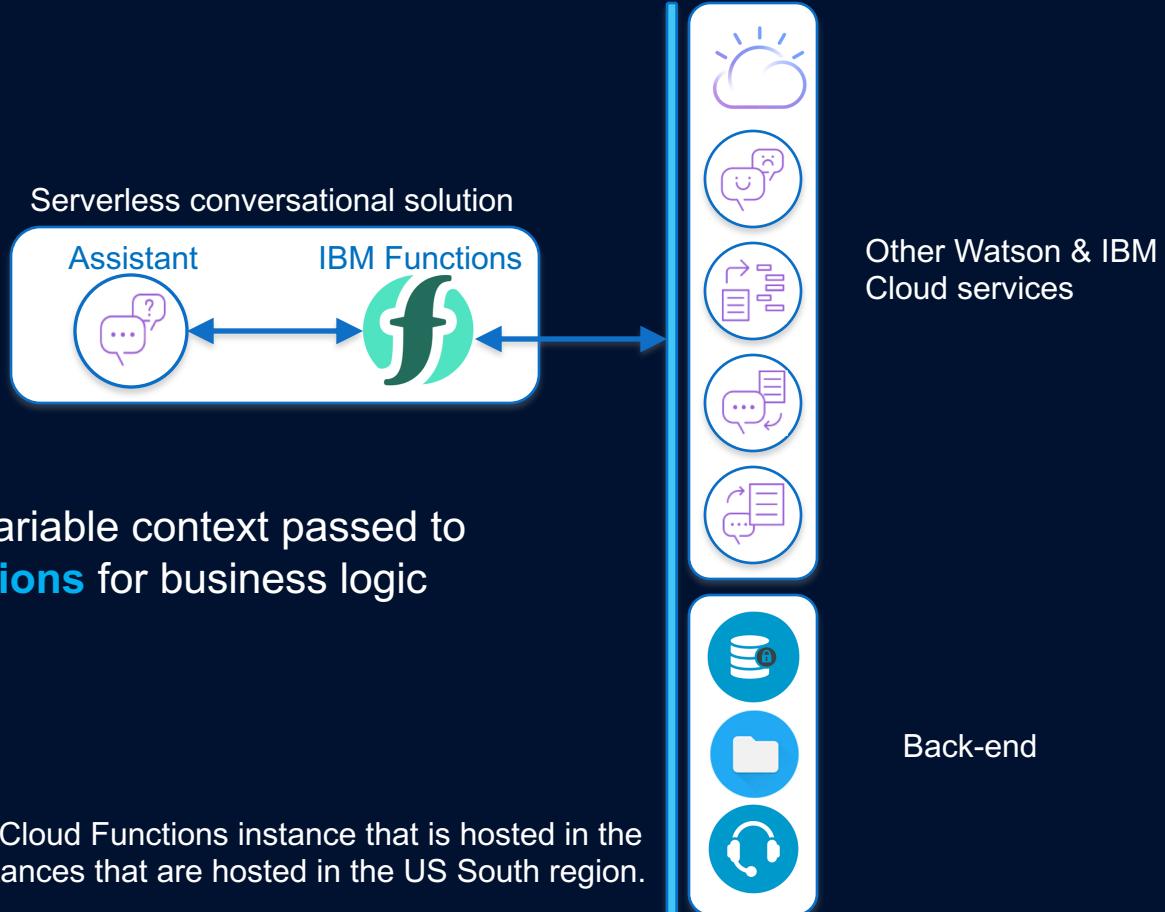
Ouput can be a way to provide details to your application

- To callout any APIs
- To show a form
- To display buttons
- To display any useful information with a custom format

# WA Architecture



# Enrichment



## IBM Cloud Functions



- Overview
  - Built on OpenWisk
  - Fully Managed public cloud offering
  - Out of the box API Gateway support
- Integrations
  - Prebuilt integration with IBM Cloud Services: MessageHub, AppConnect, Couldant, Watson Conversation

# Cloud Functions Features

The screenshot shows the Watson Conversation service interface. On the left, there's a sidebar with icons for navigation. The main area is titled "Watson Conversation / ConvWks-171003-LV / Deploy". It has tabs for "Deploy Options" (which is selected) and "Credentials". Below this, it says "Deploy with Cloud Functions". There are three cards: 1. Slack: Shows the Slack logo, text "Deploy this workspace to Slack using your own app or test it quickly using IBM's app.", and a "Deploy" button. 2. Facebook Messenger: Shows the Facebook Messenger logo, text "Deploy your workspace to Messenger using your own app.", and a "Deploy" button. 3. Web app: Shows a link icon, text "Deploy this workspace to a URL that you can share with anyone.", and the note "Coming soon". A note at the top right says "Depending on your chat volume, additional costs may apply. Learn more".

## Watson Dialog Integration

- Call out to Actions directly from a Watson Conversation dialog
- Via embedded JSON-based declarations of actions, parameters , and results

## Messaging Service Integration

- Rapid setup of Actions that mediate between messaging applications and the Watson Conversation services
- Support deployment of test chatbots to

The screenshot shows a Watson Conversation dialog configuration screen. At the top, it says "welcome". Below that, it asks "If bot recognizes:" and lists "true". Under "Then respond with:", there is a JSON block:

```
1 {  
2   "output": {  
3     "text": {  
4       "values": [  
5         "I am Watson, nice to meet you"  
6       ],  
7       "selection_policy": "sequential"  
8     }  
9   },  
10  "actions": [  
11    {  
12      "name": "/laurent_vincent@fr.ibm.com_Labs/weather",  
13      "type": "server",  
14      "parameters": {}  
15    }  
16  ]  
17}
```

# Advanced Editor: JSON editor

```
{  
  "context": {  
    "{any_objects}": " {any_value}"  
  },  
  "actions": [ {  
    "name": "<actionName>",  
    "type": "client | server",  
    "parameters": {  
      "<parameter_name>": "<parameter_value>",  
      "<parameter_name>": "<parameter_value>" },  
    "result_variable": "<result_variable_name>",  
    "credentials": "<reference_to_credentials>"  
  } ],  
  "output": {  
    "text": [ "text one" ],  
    "generic": [ object ],  
    "{action)": " {any_value}",  
  }  
}
```

welcome

If bot recognizes:

true

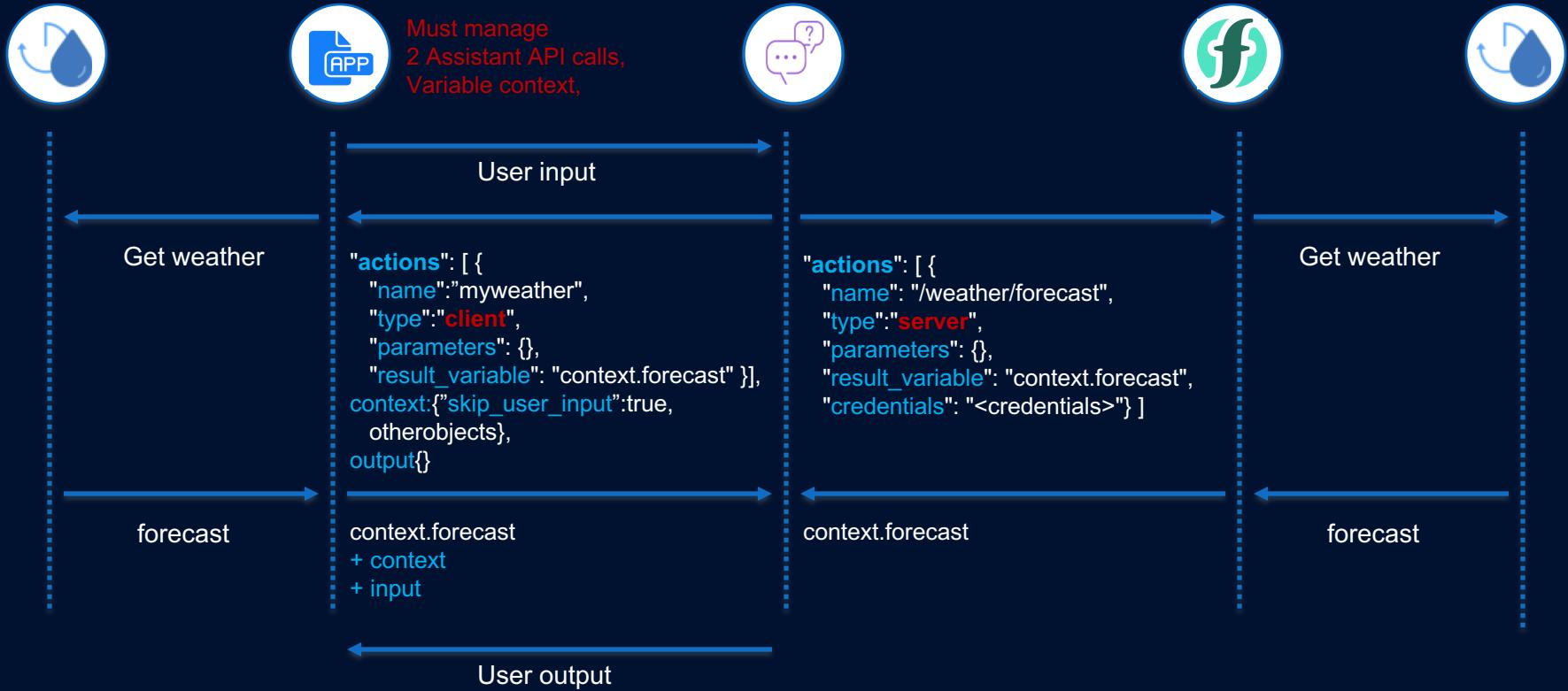
Then respond with:

Advanced editor menu: Delete a response or open the Json editor for the response field.

... 

```
1 {  
2   "output": {  
3     "text": {  
4       "values": [  
5         "I am Watson, nice to meet you"  
6       ],  
7       "selection_policy": "sequential"  
8     }  
9   },  
10  "actions": [  
11    {  
12      "name": "/laurent_vincent@fr.ibm.com_Labs/weather",  
13      "type": "server",  
14      "parameters": {
```

# Making calls from a dialog node



**Let's get started !**