

Conversational AI at the Restaurant

Conversational AI for the Restaurant

Why conversational AI?

Chatbots make software accessible to everyone who understands human language. A customer can avoid the frustration that comes with having to memorize and navigate complex menus and button layouts that are always changing with software updates. Instead, computers can be operated with simple human language that people can understand. Customers can now simply ask a bot to take them where they want to go, or to enable a feature without having to hunt it down. Good bot services encourage users to engage more deeply with software features that might otherwise go unnoticed, because they provide a richer, more natural experience. For example, imagine an image editing suite that can respond to a command like: “Make the background of my photo darker.”

In addition to enhancing the customer experience, chatbots free up agents to respond to the more complex problems that are better solved by a customer service agent. When people are able to delegate a portion of their workload onto a conversational bot, they are now able to participate in higher value decision making for the company and expand their skills, which benefits the company and enriches the agents.

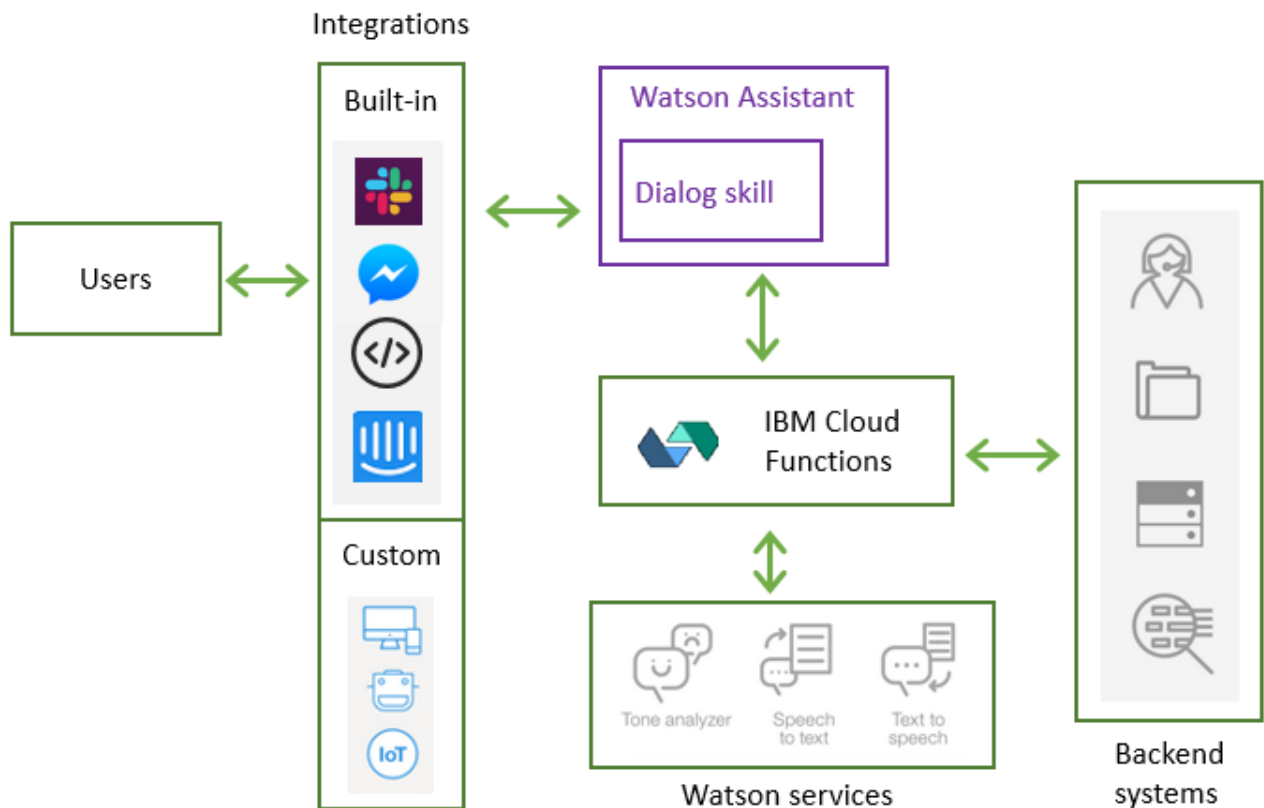
The Royal Bank of Scotland used Watson to build their bot framework, which consists of two bots called “Cora” and “Marge.” Cora handles simpler, “first-time” problem resolution, and Marge assists the agents themselves when they need more information to respond to a customer’s query.

Click the link below and scroll to the bottom of the page. There is an imbedded chatbot with various skills that you can play with. Try it:

<https://www.ibm.com/watson/how-to-build-a-chatbot>

How Does Conversational AI Work?

This diagram shows the overall architecture:



- Users interact with the assistant through one or more of these **integration** points:
 - A chat bot that you publish directly to an existing social media messaging platform, such as Slack or Facebook Messenger.
 - A simple chat bot user interface that is hosted by IBM Cloud.
 - Custom application that you develop, such as a mobile app or a robot with a voice interface.
- The **assistant** receives user input and routes it to the dialog skill.
- The dialog **skill** interprets the user input further, then directs the flow of the conversation and gathers any information that it needs to respond or perform a transaction on the user's behalf.

Implementation

Here's how you will implement your assistant:

- **Create a dialog skill.** Use the intuitive graphical tool to define the training data and dialog for the conversation between your assistant and your customers.

The training data consists of the following artifacts:

- **Intents:** Goals that you anticipate your users will have when they interact with the service. Define one intent for each goal that can be identified in a user's input. For example, you might define an intent named *store_hours* that answers questions about store hours. For each intent, you add sample utterances that reflect the input customers might use to ask for the information they need, such as, *What time do you open?*

Or use prebuilt **content catalogs** provided by IBM to get started with data that addresses common customer goals.

- **Entities:** An entity represents a term or object that provides context for an intent. For example, an entity might be a city name that helps your dialog to distinguish which store the user wants to know store hours for.

As you add training data, a natural language classifier is automatically added to the skill and is trained to understand the types of requests that you have indicated the service should listen for and respond to.

- **Dialog:** Use the dialog tool to build a dialog flow that incorporates your intents and entities. The dialog flow is represented graphically in the tool as a tree. You can add a branch to process each of the intents that you want the service to handle. You can then add branch nodes that handle the many possible permutations of a request based on other factors, such as the entities found in the user input or information that is passed to the service from an external service.
- **Create an assistant.**
- **Add the dialog skill to your assistant.**
- **Integrate your assistant.** Create a channel integration to deploy the configured assistant directly to a social media or messaging channel.

Usecase: Build a Chatbot for Restaurant

While chatbots offer many benefits, the most significant of them are listed below:

- Book tables and orders, even before the customer has reached the restaurant.
- Customers can track the status of their orders.
- Serve any number of customers 24/7
- Personalized marketing
- Customer analytics
- Does the job of many employees and avoids employees training cost
- Integrate chatbot to social media and make use of vast social media user base
- Increases brand engagement
- Businesses can stay in direct contact with the customers, which will enable to stop unnecessary revenue share with services delivery mediums, such as Postmates.

Chatbots help in dealing with the challenges faced by the restaurant industry with analytics, as well. Chatbots can help management gather and organize sales data, then strategize their marketing efforts based on location and customer interests. This helps management to deliver personalized marketing plans, push notifications (about loyalty programs or new items) and personalized dining experiences to the customers. This instills a feeling of community with the customers, which ultimately helps the management retain customers as well as increase their satisfaction.

Let's disrupt the traditional way of ordering food with Watson Assistant.



<https://businessfirstfamily.com/pick-restaurant-online-ordering-software/>

Preface

This guide is an instructional approach to working with the IBM Watson™ Assistant service where you can create virtual agents and bots that combine machine learning, natural language understanding, and integrated dialog tools to provide automated customer engagements. Creating your first conversation using the IBM Watson™ Assistant service entails the following steps:

- Train Watson to understand your users' input with example utterances: Intents and Examples
- Identify the terms that may vary in your users' input: Entities
- Create the responses to your user's questions: Dialog Builder

Complete the following steps:

1. Login into IBM Cloud: <https://console.bluemix.net/catalog/>
2. Click the **Catalog** tab.
3. Search for the **Watson Assistant** service and click that tile under the AI heading.

Instantiate a new Watson Assistant service

Common to the exercises in this document is that you have already obtained an IBM Cloud account and applied the promo code.

1. Login into IBM Cloud: <https://console.bluemix.net/catalog/>
2. Click the **Catalog** tab.
3. Search for the **Watson Assistant** service and click that tile under the AI heading.

Step 1: Open the tool

After you create a Watson Assistant service instance, you land on the **Getting started** page of the service dashboard.

1. Click **Manage**, and then click **Launch tool**. If you're prompted to log in to the tool, provide your IBM Cloud credentials.

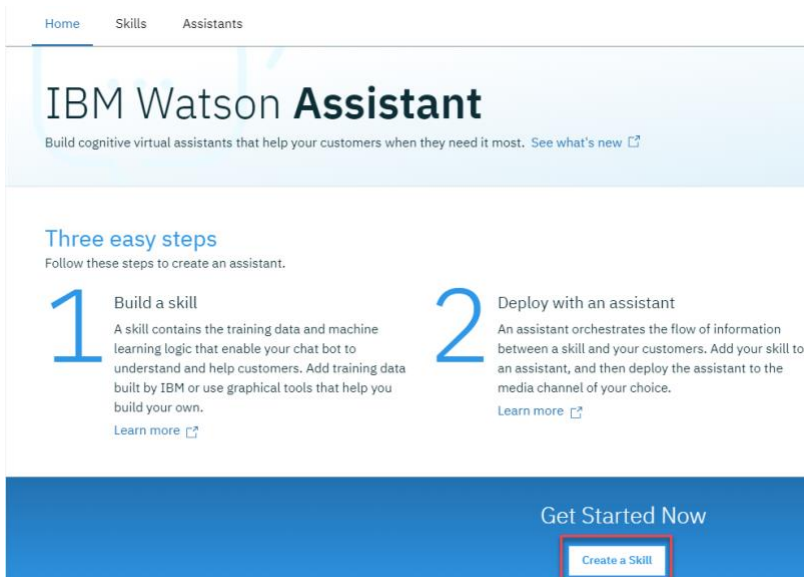
IBM Cloud Dedicated: Select your service instance from the Dashboard to launch the tool.

Step 2: Create a dialog skill

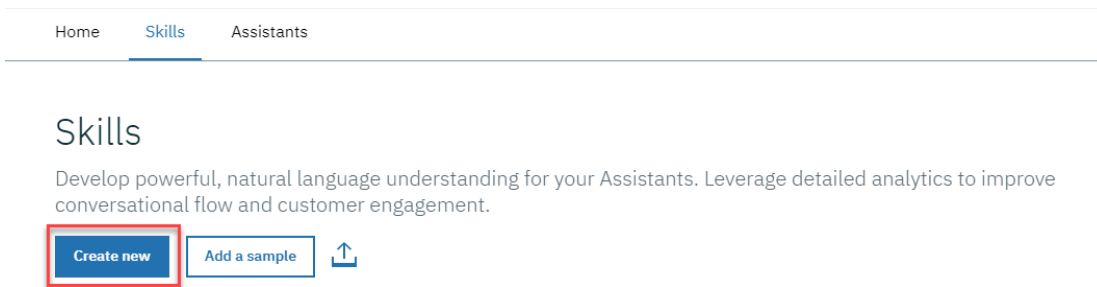
Your first step in the Watson Assistant tool is to create a skill.

A *dialog skill* is a container for the artifacts that define the flow of a conversation that your assistant can have with your customers.

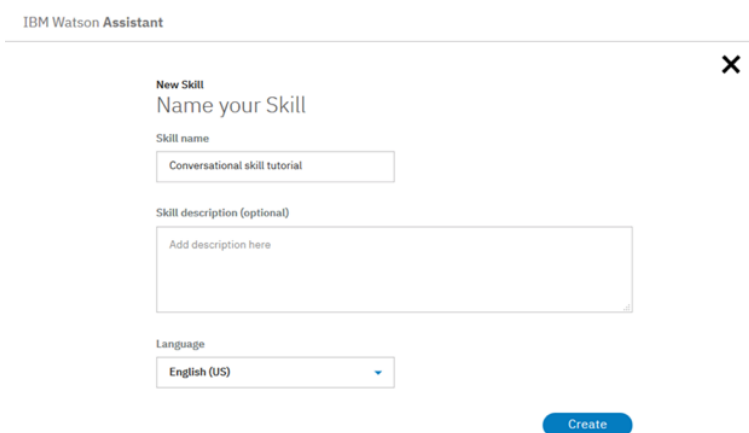
1. From the home page of the Watson Assistant tool, click **Create a Skill**.



2. Click **Create new**.



3. Give your skill the name `Conversational skill tutorial`.
4. **Optional.** If the dialog you plan to build will use a language other than English, then choose the appropriate language from the list.
5. Click **Create**.

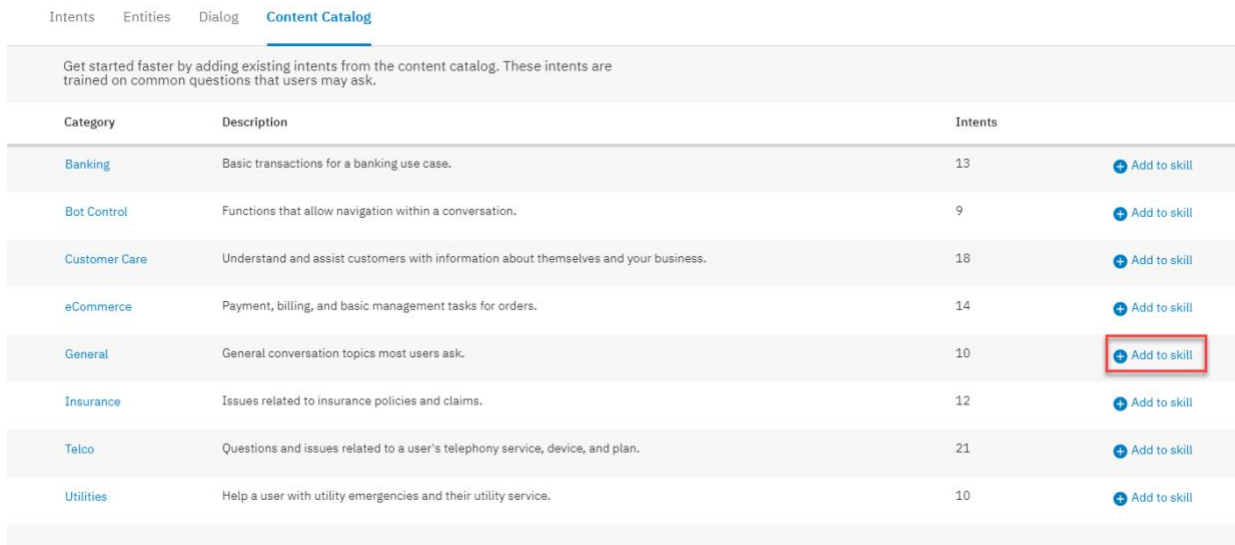


You land on the Intents page of the tool.

Step 3: Add intents from a content catalog

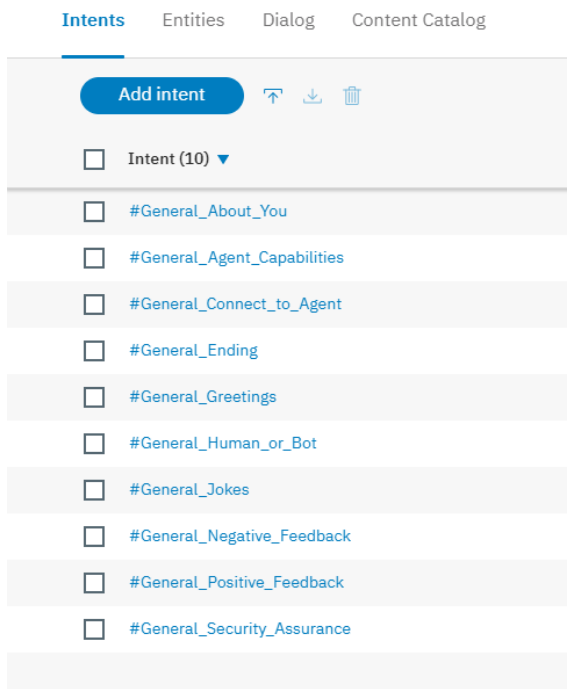
Add training data that was built by IBM to your workspace by adding intents from a content catalog. In particular, you will give your assistant access to the **General** content catalog so your dialog can greet users, and end conversations with them.




1. In the Watson Assistant tool, click the **Content Catalog** tab.
2. Find **General** in the list, and then click **Add to skill**.



Intents	Entities	Dialog	Content Catalog
Get started faster by adding existing intents from the content catalog. These intents are trained on common questions that users may ask.			
Category	Description	Intents	
Banking	Basic transactions for a banking use case.	13	Add to skill
Bot Control	Functions that allow navigation within a conversation.	9	Add to skill
Customer Care	Understand and assist customers with information about themselves and your business.	18	Add to skill
eCommerce	Payment, billing, and basic management tasks for orders.	14	Add to skill
General	General conversation topics most users ask.	10	Add to skill
Insurance	Issues related to insurance policies and claims.	12	Add to skill
Telco	Questions and issues related to a user's telephony service, device, and plan.	21	Add to skill
Utilities	Help a user with utility emergencies and their utility service.	10	Add to skill

3. Open the **Intents** tab to review the intents and associated example utterances that were added to your training data. You can recognize them because each intent name begins with the prefix `#General_`. You will add the `#General_Greetings` and `#General_Ending` intents to your dialog in the next step.



Intents	Entities	Dialog	Content Catalog
Add intent   			
<input type="checkbox"/> Intent (10) ▼			
<input type="checkbox"/> #General_About_You			
<input type="checkbox"/> #General_Agent_Capabilities			
<input type="checkbox"/> #General_Connect_to_Agent			
<input type="checkbox"/> #General_Ending			
<input type="checkbox"/> #General_Greetings			
<input type="checkbox"/> #General_Human_or_Bot			
<input type="checkbox"/> #General_Jokes			
<input type="checkbox"/> #General_Negative_Feedback			
<input type="checkbox"/> #General_Positive_Feedback			
<input type="checkbox"/> #General_Security_Assurance			

You successfully started to build your training data by adding prebuilt content from IBM.

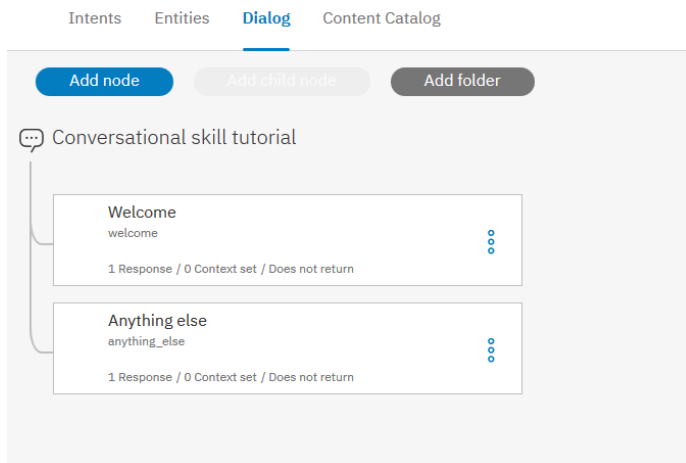
Step 4: Build a dialog

A [dialog](#) defines the flow of your conversation in the form of a logic tree. It matches intents (what users say) to responses (what the bot says back). Each node of the tree has a condition that triggers it, based on user input.

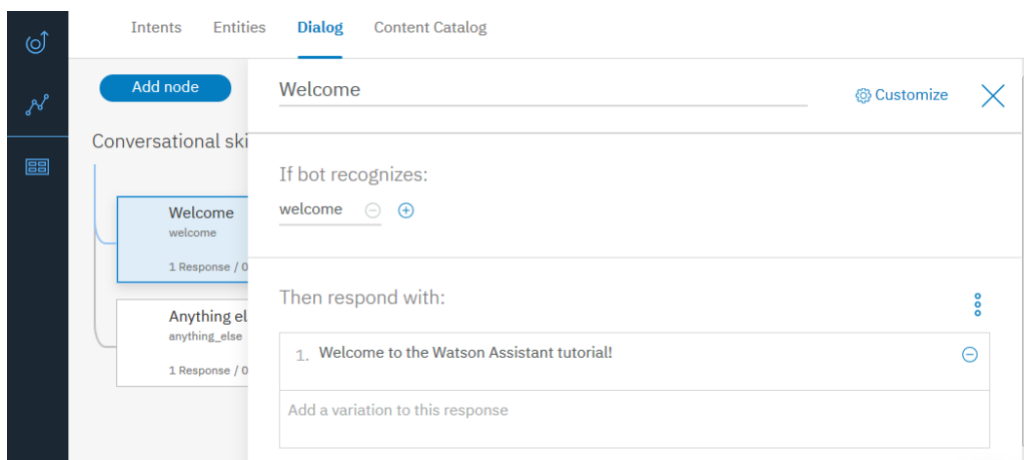
We'll create a simple dialog that handles greeting and ending intents, each with a single node.


1.1.1. Adding a start node

1. In the Watson Assistant tool, click the **Dialog** tab.
2. Click **Create**. You see two nodes:
 - **Welcome**: Contains a greeting that is displayed to your users when they first engage with the assistant.
 - **Anything else**: Contains phrases that are used to reply to users when their input is not recognized.



3. Click the **Welcome** node to open it in the edit view.
4. Replace the default response with the text, Welcome to the Watson Assistant tutorial!.




5. Click  to close the edit view.

You created a dialog node that is triggered by the `welcome` condition (`welcome` is a special condition that functions like an Intent, but does not begin with a #); it is triggered when a new conversation starts. Your node specifies that when a new conversation starts, the system should respond with the welcome message that you add to the response section of this first node.



1.1.2. Testing the start node

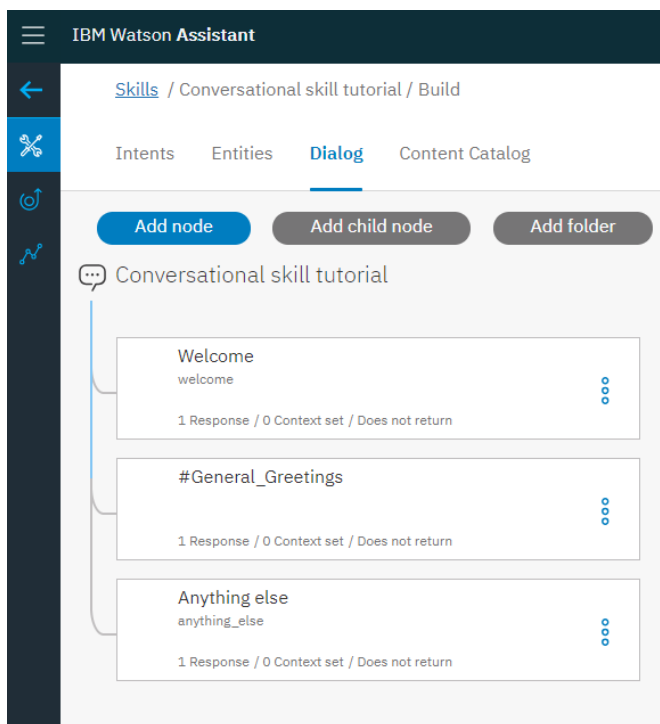
You can test your dialog at any time to verify the dialog. Let's test it now.


- Click the  icon to open the "Try it out" pane. You should see your welcome message.

1.1.3. Adding nodes to handle intents

Now let's add nodes to handle our intents between the `Welcome` node and the `Anything else` node.

- Click the More icon  on the **Welcome** node, and then select **Add node below**.
- Type `#General_Greetings` in the **Enter a condition** field of this node. Then, select the **#General_Greetings** option.
- Add the response, `Good day to you!`
- Click  to close the edit view.



- Click the More icon  on this node, and then select **Add node below** to create a peer node. In the peer node, specify `#General_Ending` as the condition, and `OK. See you later.` as the response.

Intents Entities **Dialog** Content Catalog

Add node Add child node

Conversational skill tutorial

- Welcome
welcome
1 Response / 0 Context set / Does not return
- #General_Greetings
1 Response / 0 Context set / Does not return
- #General_Ending
1 Response / 0 Context set / Does not return
- Anything else
anything_else
1 Response / 0 Context set / Does not return

Name this node...

If bot recognizes:
#General_Ending

Then respond with:


Text

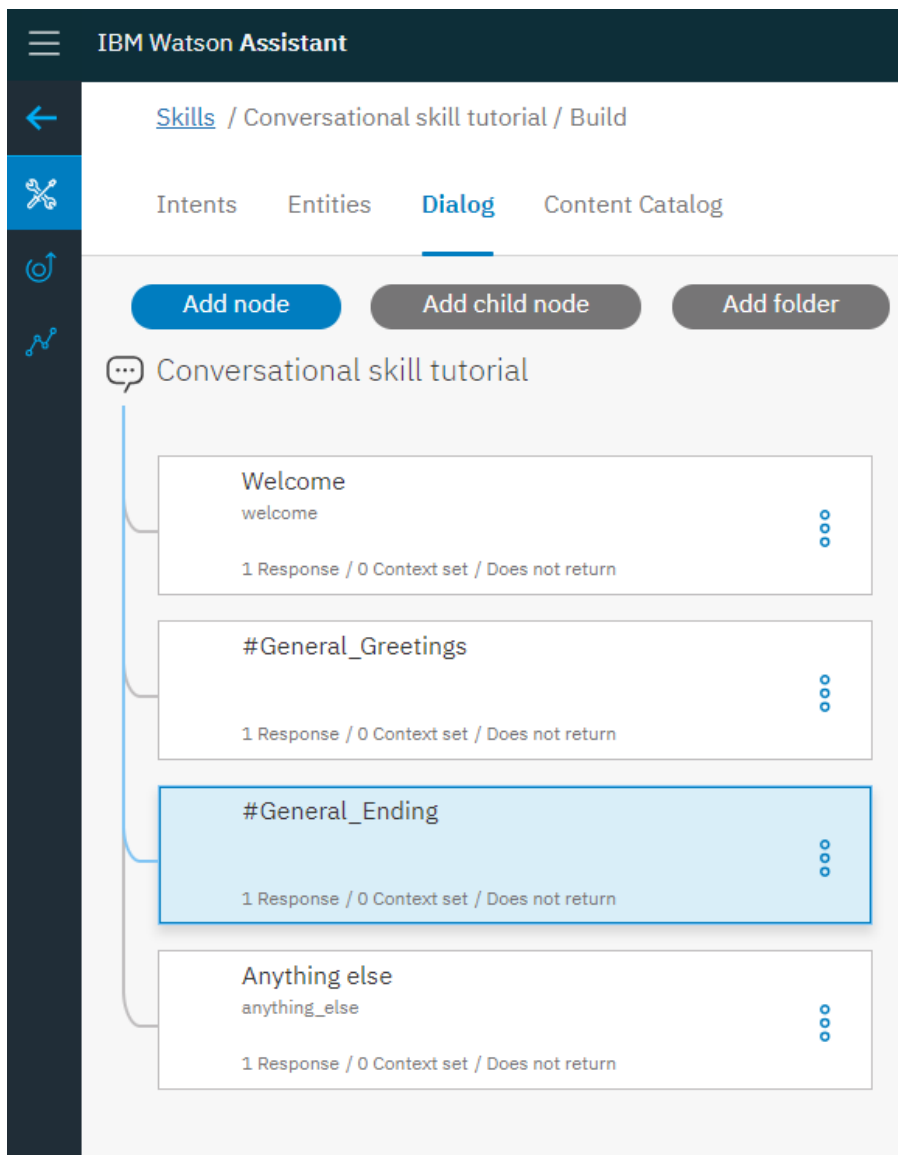
OK. See you later!

Enter response variation

Response variations are set to sequential. Set to random

Add response type


6. Click  to close the edit view.



1.1.4. Testing intent recognition

You built a simple dialog to recognize and respond to both greeting and ending inputs. Let's see how well it works.



1. Click the  icon to open the "Try it out" pane. There's that reassuring welcome message.
2. At the bottom of the pane, type `Hello` and press Enter. The output indicates that the `#hello` intent was recognized, and the appropriate response (`Good day to you.`) appears.
3. Try the following input:
 - o `bye`
 - o `howdy`
 - o `see ya`
 - o `good morning`
 - o `sayonara`

Watson can recognize your intents even when your input doesn't exactly match the examples that you included. The dialog uses intents to identify the purpose of the user's input regardless of the precise wording used, and then responds in the way you specify.

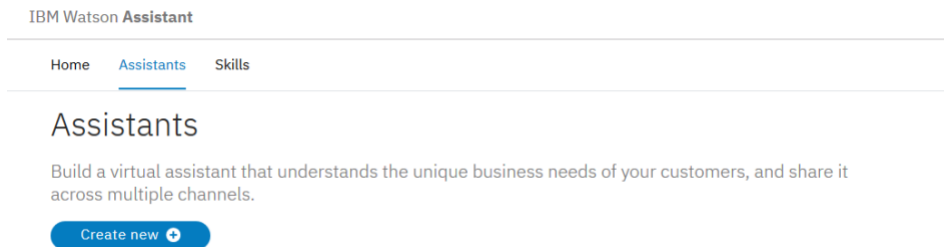
1.1.5. Result of building a dialog

That's it. You created a simple conversation with two intents and a dialog to recognize them.

Step 5: Create an assistant

An [assistant](#) is a cognitive bot to which you add a skill that enables it to interact with your customers in useful ways.

1. Click the **Assistants** tab.
2. Click **Create new**.



3. Name the assistant `Watson Assistant tutorial`.
4. In the **Description** field, enter `This is a sample assistant that I am creating to help me learn.`
5. Click **Create**.

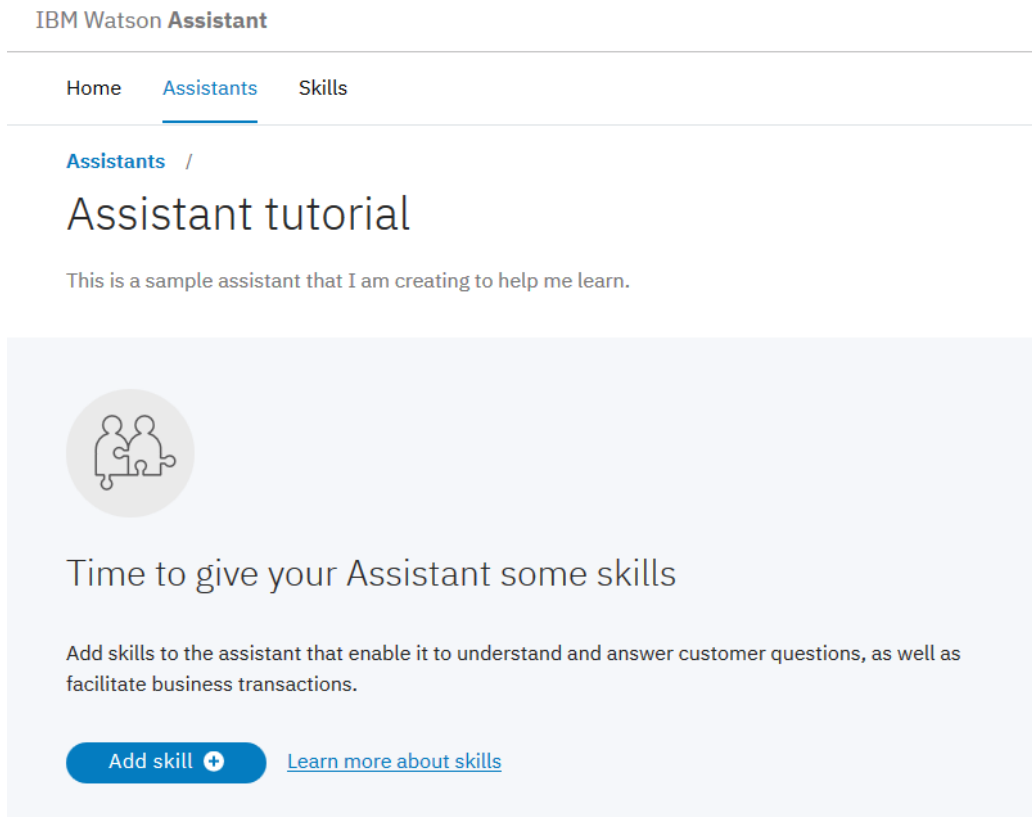
A screenshot of the 'New assistant' form in the IBM Watson Assistant interface. The form is titled 'New assistant' and has a close button (X) in the top right corner. It contains two main input fields: 'Assistant name' and 'Assistant description (optional)'. The 'Assistant name' field has the text 'Assistant tutorial' entered. The 'Assistant description (optional)' field has the text 'This is a simple assistant that I am creating to help me learn.' entered. At the bottom right of the form, there is a blue button labeled 'Create'.

Step 6: Add your skill to your assistant

Add the dialog skill that you build to the assistant you created.

1. From the new assistant page, click **Add skill**.

If you created or were given developer role access to any workspaces that were built with the generally available version of the Watson Assistant service, you will see them listed on the Skills page as conversational skills.



2. Choose to add the skill that you created earlier to the assistant.

Step 7: Integrate the assistant

Now that you have an assistant that can participate in a simple conversational exchange, publish it to a public web page where you can test it out. The service provides a built-in integration that is called a Preview Link. When you create this type of integration, it builds your assistant into a chat widget that is hosted by an IBM-branded web page. You can open the web page and chat with your assistant to test it out.

1. Click the **Assistants** tab, find the `Watson Assistant tutorial` assistant that you created, and open it.
2. From the *Integrations* area, click **Add integration**.
3. Find **Preview Link** and click **Select integration**.
4. Click the URL that is displayed on the page.

The page opens in a new tab.

5. Say `hello` to your assistant, and watch it respond. You can share the URL with others who might want to try out your assistant.

Next steps (optional)

This tutorial is built around a simple example. For a real application, you need to define some more interesting intents, some entities, and a more complex dialog that uses them both. When you have a polished version of the assistant, you can integrate it with channels that your customers use, such as Slack. As traffic increases between the assistant and your customers, you can use the tools that are provided in the **Improve** tab to analyze real conversations and identify areas for improvement.

Check out more [sample apps](#) to get ideas.

Section 2. Tutorial: Building a complex dialog

In this tutorial, you will use the Watson Assistant service to create a dialog for an assistant that helps users with inquiries about a fictitious restaurant called *Truck Stop Gourmand*.

Learning objectives

By the time you finish the tutorial, you will understand how to:

- Plan a dialog
- Define custom intents
- Add dialog nodes that can handle your intents
- Add entities to make your responses more specific
- Add a pattern entity, and use it in the dialog to find patterns in user input
- Set and reference context variables

This tutorial will take approximately 2 to 3 hours to complete. You must first complete the exercises in the prior sections for the following activities build upon the work you did earlier.

Step 1: Plan the dialog

You are building an assistant for a restaurant named *Truck Stop Gourmand* that has one location and a thriving cake-baking business. You want the simple assistant to answer user questions about the restaurant, its menu, and to cancel customer cake orders. Therefore, you need to create intents that handle inquiries related to the following subjects:

- Restaurant information
- Menu details
- Order cancelations

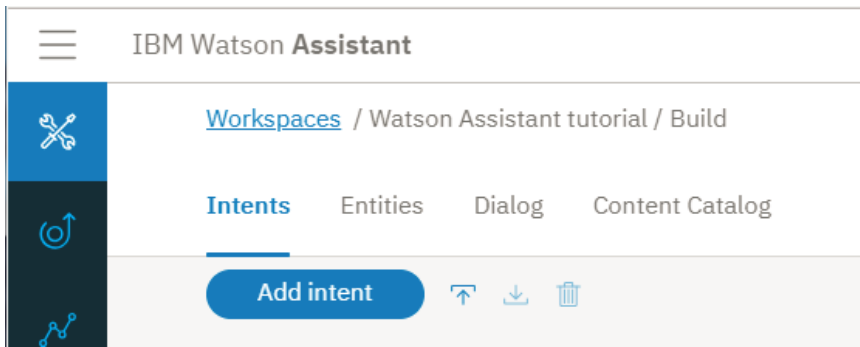
You'll start by creating intents that represent these subjects, and then build a dialog that responds to user questions about them.

Step 2: Answer questions about the restaurant

Add an intent that recognizes when customers ask for details about the restaurant itself. An intent is the purpose or goal expressed in user input. The `#General_About_You` intent that is provided with the *General* content catalog serves a similar function, but its user examples are designed to focus on queries about the assistant as opposed to the business that is using the assistant to help its customers. So, you will add your own intent.


2.1.1. Add the `#about_restaurant` intent

1. Click the **Intents** tab.
2. Click **Add intent**.



3. Enter `about_restaurant` in the *Intent name* field, and then click **Create intent**.
4. Add the following user examples:

```
Tell me about the restaurant
i want to know about you
who are the restaurant owners and what is their philosophy?
What's your story?
Where do you source your produce from?
Who is your head chef and what is the chef's background?
How many locations do you have?
do you cater or host functions on site?
Do you deliver?
Are you open for breakfast?
```

5. Click the **Close**  icon to finish adding the `#about_restaurant` intent.

You added an intent and provided examples of utterances that real users might enter to trigger this intent.

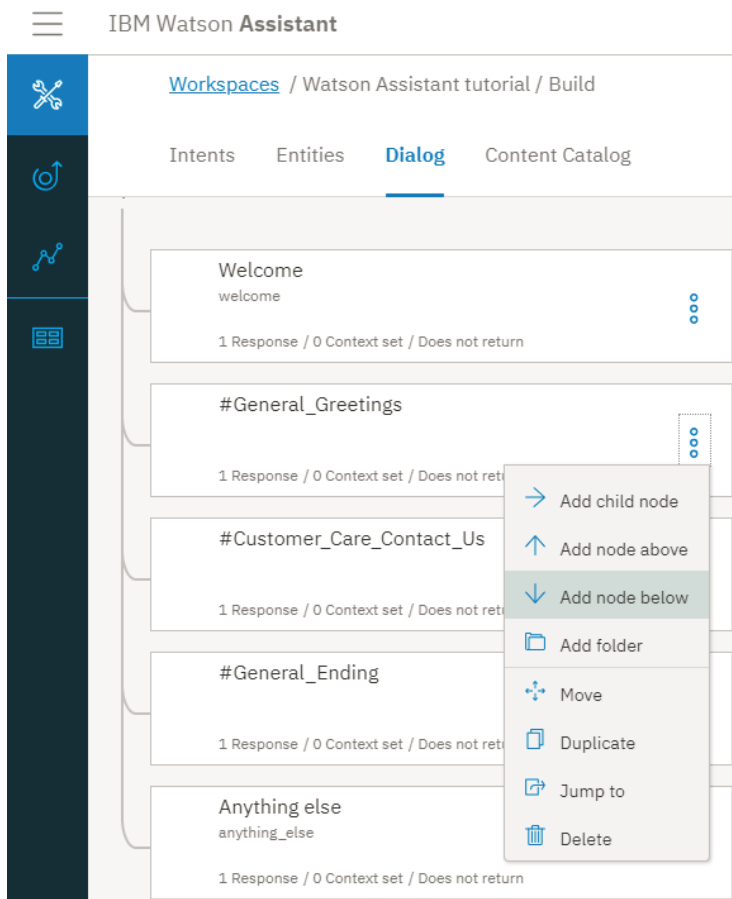
2.1.2. Add a dialog node that is triggered by the `#about_restaurant` intent

Add a dialog node that recognizes when the user input maps to the intent that you created in the previous step, meaning its condition checks whether the service recognized the `#about_restaurant` intent from the user input.

1. Click the **Dialogs** tab.
2. Find the `#General_Greetings` node in the dialog tree.

You will add a node that checks for questions about the restaurant below this initial greeting node to reflect the flow you might expect to encounter in a normal conversation. For example, `Hello.` then `Tell me about yourself.`

3. Click the **More**  icon on the `#General_Greetings` node, and then select **Add node below**.



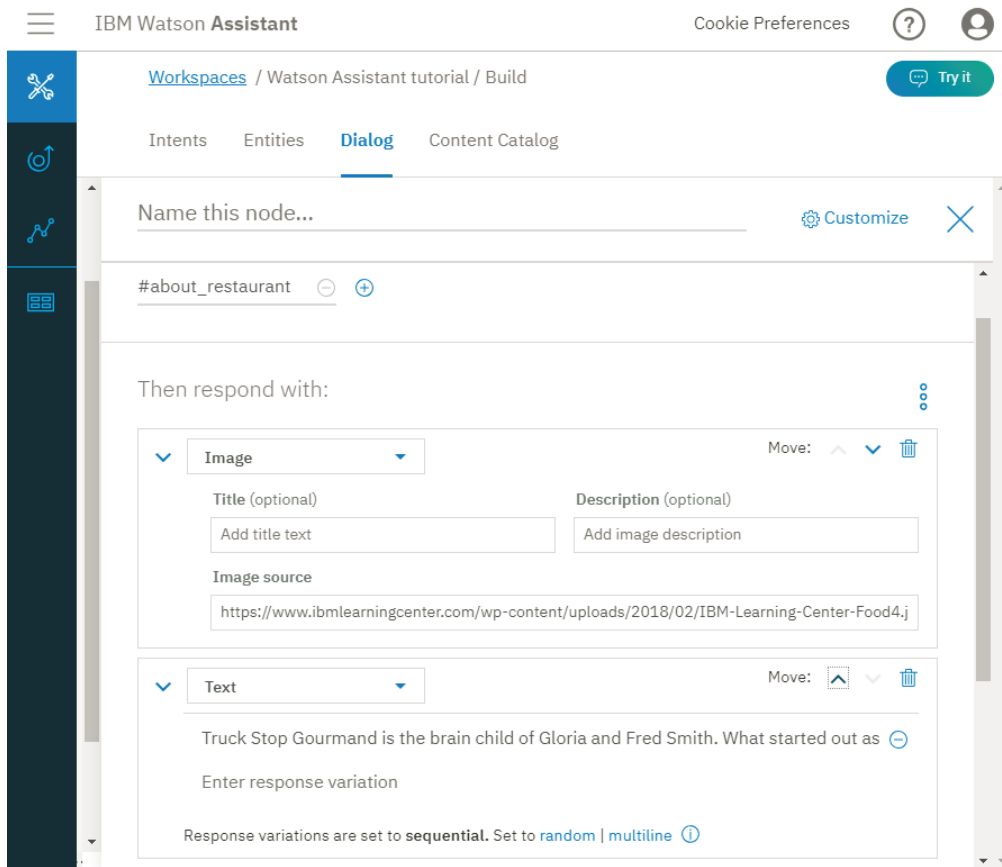
4. Start to type `#about_restaurant` into the **Enter a condition** field of this node. Then select the `#about_restaurant` option.
5. Add the following text as the response:


Truck Stop Gourmand is the brain child of Gloria and Fred Smith. What started out as a food truck in 2004 has expanded into a thriving restaurant. We now have one brick and mortar restaurant in downtown Portland. The bigger kitchen brought with it new chefs, but each one is faithful to the philosophy that made the Smith food truck so popular to begin with: deliver fresh, local produce in inventive and delicious ways. Join us for lunch or dinner seven days a week. Or order a cake from our bakery.

5. Let's add an image to the response also.

Click **Add response type**. Select **Image** from the drop-down list. In the **Image source** field, add `https://www.ibmlearningcenter.com/wp-content/uploads/2018/02/IBM-Learning-Center-Food4.jpg`.


6. Move the image response type up, so it is displayed in the response before the text is displayed. Click the **Move** up arrow to reorder the two response types.



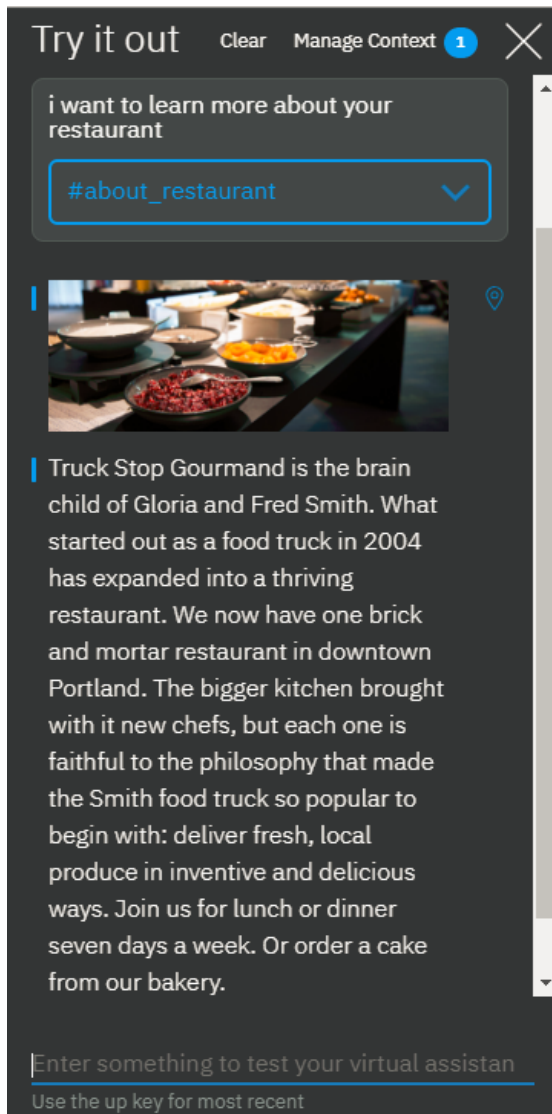
7. Click  to close the edit view.

2.1.3. Test the #about_restaurant dialog node

Test the intent by checking whether user utterances that are similar to, but not exactly the same as, the examples you added to the training data have successfully trained the service to recognize input with an `#about_restaurant` intent.

1. Click the  icon to open the "Try it out" pane.
2. Enter, I want to learn more about your restaurant.

The service indicates that the `#about_restaurant` intent is recognized, and returns a response with the image and text that you specified for the dialog node.



Congratulations! You have added a custom intent, and a dialog node that knows how to handle it. The `#about_restaurant` intent is designed to recognize a variety of general questions about the restaurant. You added a single node to capture such questions. The response is long, but it is a single statement that can potentially answer questions about all of the following topics:

- The restaurant owners
- The restaurant history
- The philosophy
- The number of sites
- The days of operation
- The meals served
- The fact that the restaurant bakes cakes to order

For general, low-hanging fruit types of questions, a single, general answer is suitable.

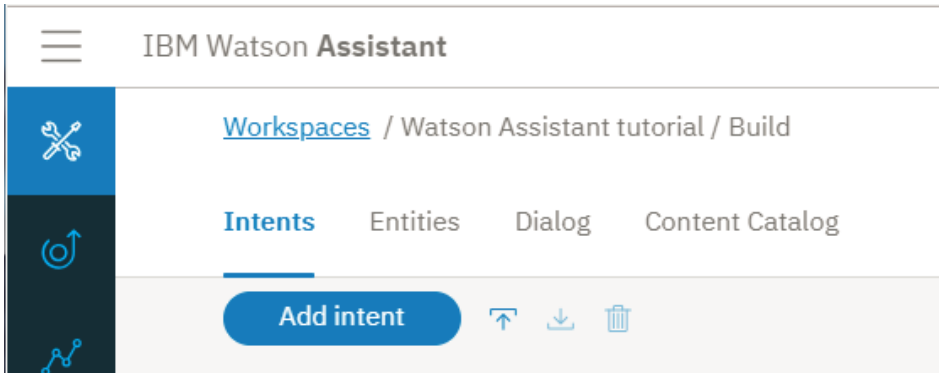
Step 3: Answer questions about the menu

A key question from potential restaurant customers is about the menu. The Truck Stop Gourmand restaurant changes the menu daily. In addition to its standard menu, it has vegetarian and cake shop menus. When a user asks about the menu, the dialog needs to find out which menu to share, and then

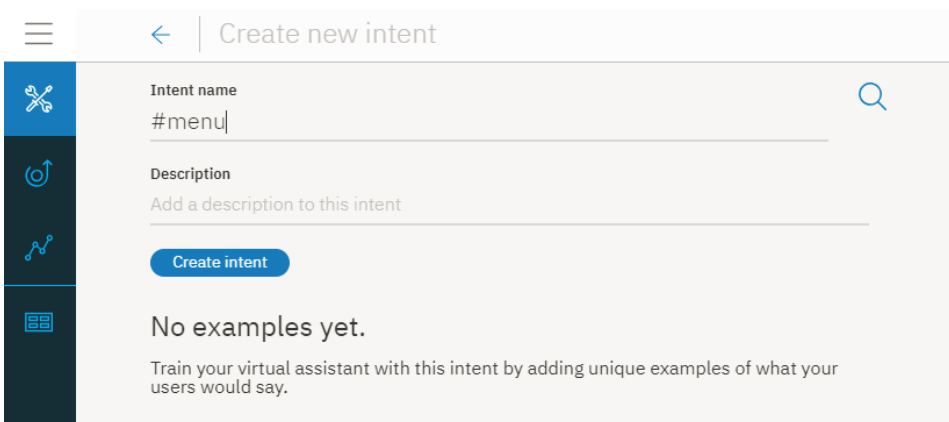
provide a hyperlink to the menu that is kept up to date daily on the restaurant's website. You never want to hard-code information into a dialog node if that information changes regularly.

2.1.4. Add a #menu intent


1. Click the **Intents** tab.
2. Click **Add intent**.



3. Enter `menu` in the *Intent name* field, and then click **Create intent**.



4. Add the following user examples:
5. I want to see a menu
6. What do you have for food?
7. Are there any specials today?
8. where can i find out about your cuisine?
9. What dishes do you have?
10. What are the choices for appetizers?
11. do you serve desserts?
12. What is the price range of your meals?
13. How much does a typical dish cost?
14. tell me the entree choices
15. Do you offer a prix fixe option?

16. Click the **Close**  icon to finish adding the #menu intent.

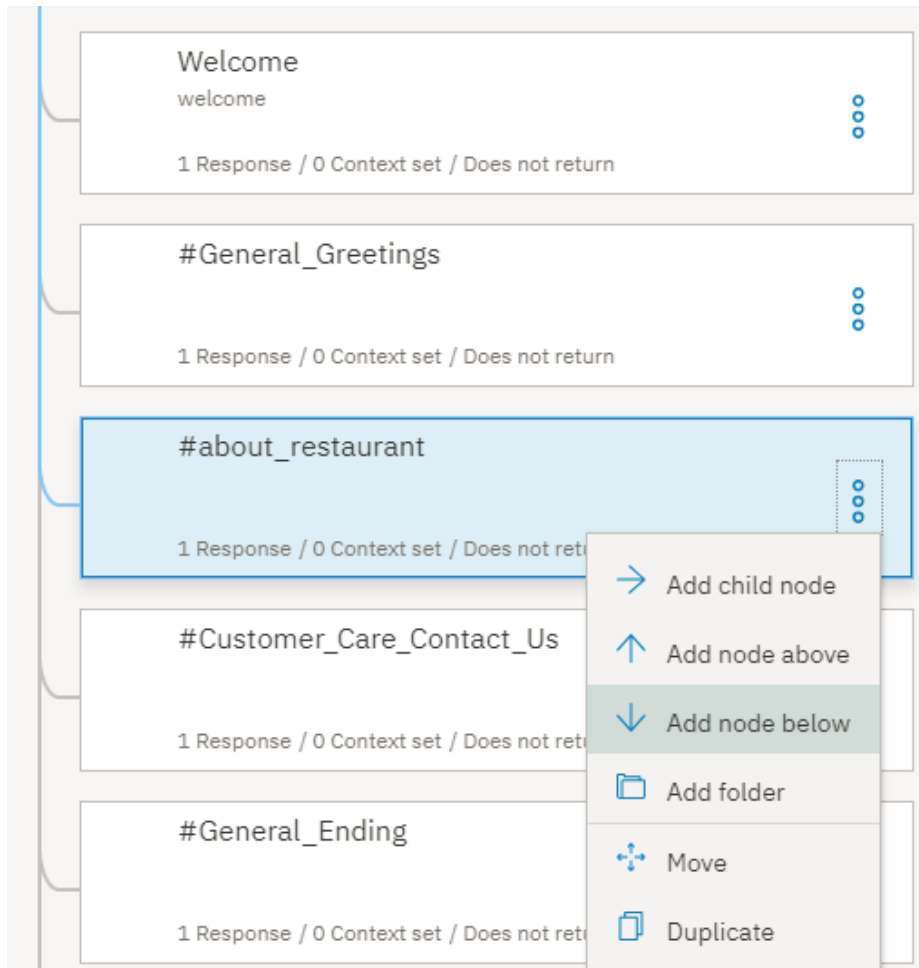
2.1.5. Add a dialog node that is triggered by the #menu intent

Add a dialog node that recognizes when the user input maps to the intent that you created in the previous step, meaning its condition checks whether the service recognized the #menu intent from the user input.

1. Click the **Dialogs** tab.
2. Find the #about_restaurant node in the dialog tree.

You will add a node that checks for questions about the menu below this node.

3. Click the **More**  icon on the #about_restaurant node, and then select **Add node below**.



4. Start to type #menu into the **Enter a condition** field of this node. Then select the #menu option.

If bot recognizes:

#me|  

Intents:

#menu I want to see a menu

#me Create new intent



5. Add the following text as the response:




In keeping with our commitment to giving you only fresh local ingredients, our menu changes daily to accommodate the produce we pick up in the morning. You can find today's menu on our website.


6. Add an *option* response type that provides a list of options for the user to choose from. In this case, the list of options includes the different versions of the menu that are available.

Click **Add response type**. Select **Option** from the drop-down list.


Then respond with:

 Text 

Move:   

In keeping with our commitment to giving you only fresh local ingredients, our menu 

Enter response variation

Response variations are set to **sequential**. Set to [random](#) | [multiline](#) 

 Add response type

Text

Option

Pause

Image

7. In the **Title** field, add *Which menu do you want to see?*

Text

In keeping with our commitment to giving you only fresh local ingredients, our menu

Enter response variation

Response variations are set to **sequential**. Set to [random](#) | [multiline](#)

Option

Title: Which menu do you want to see?

Description (optional): Add description

List label: Value

No options

[+ Add option](#)

[+ Add response type](#)

8. Click **Add option**.
9. In the **Label** field, add `Standard`. The text you add as the label is displayed in the response to the user as a selectable option.
10. In the **Value** field, add `standard menu`. The text you specify as the value is what gets sent to the service as new user input when a user chooses this option from the list, and clicks it.
11. Repeat the previous two steps to add label and value information for the remaining menu types:

Option response type details

Label **Value**

Vegetarian vegetarian menu

Cake shop cake shop menu

Option

Title: Which menu do you want to see?

Description (optional): Add description

List label: Value

1	Standard	standard menu	-
2	Vegetarian	vegetarian menu	-
3	Cake shop	cake shop menu	-

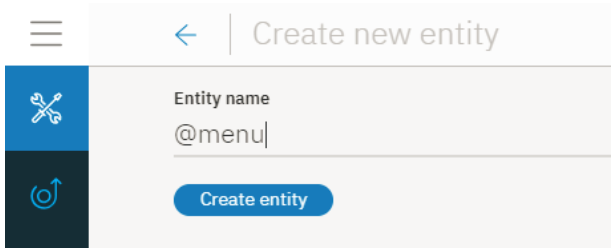
[+ Add option](#)

- 12.
13. Click [X](#) to close the edit view.

2.1.6. Add a @menu entity

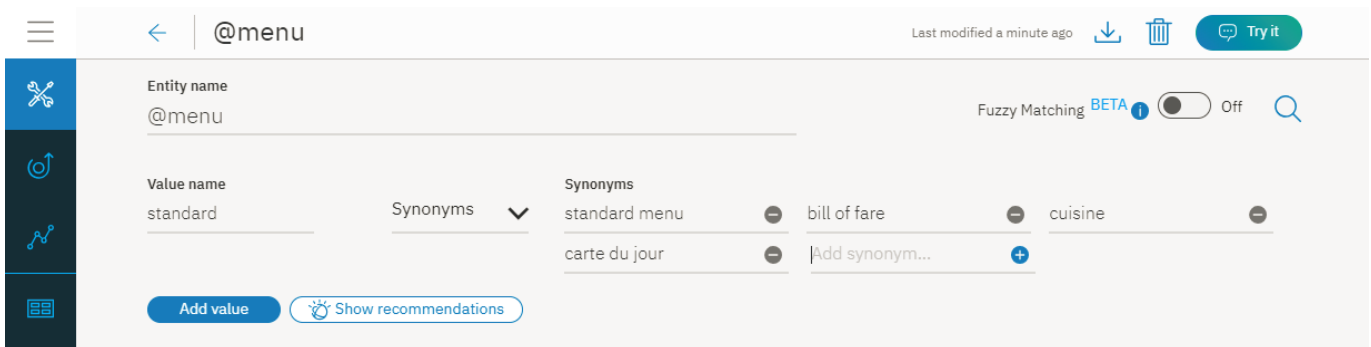
To recognize the different types of menus that customers indicate they want to see, you will add a @menu entity. Entities represent a class of object or a data type that is relevant to a user's purpose. By checking for the presence of specific entities in the user input, you can add more responses, each one tailored to address a distinct user request. In this case, you will add a @menu entity that can distinguish between different menu types.

1. Click the **Entities** tab.
2. Click **Add entity**.
3. Enter `menu` into the entity name field.



The screenshot shows a sidebar with three icons: a wrench, a target, and a document. The main area is titled 'Create new entity'. It contains a text input field labeled 'Entity name' with the text '@menu' entered. Below the input field is a blue button labeled 'Create entity'.


4. Click **Create entity**.
5. Add `standard` to the *Value name* field, and then add `standard menu` to the **Synonyms** field, and press Enter.
6. Add the following additional synonyms:
 - o bill of fare
 - o cuisine
 - o carte du jour



The screenshot shows the configuration page for the '@menu' entity. The 'Entity name' field contains '@menu'. The 'Value name' field contains 'standard'. The 'Synonyms' field contains 'standard menu', 'bill of fare', 'cuisine', and 'carte du jour'. There is an 'Add synonym...' button and a 'Show recommendations' button. The page also shows 'Fuzzy Matching BETA' set to 'Off' and 'Last modified a minute ago'.

7. Click **Add value** to add the `@menu:standard` value.
8. Add `vegetarian` to the *Value name* field, and then add `vegetarian menu` to the **Synonyms** field, and press Enter.
9. Click **Show recommendations**, and then click the checkboxes for *meatless diet*, *meatless*, and *vegan diet*.
10. Click **Add selected**.
11. Click the empty *Add synonym* field, and then add these additional synonyms:
 - o vegan
 - o plants-only

12. Click **Add value** to add the @menu:vegetarian value.
13. Add cake to the *Value name* field, and then add cake menu to the **Synonyms** field, and press Enter.
14. Add the following additional synonyms:
 - cake shop menu
 - dessert menu
 - bakery offerings


15. Click **Add value** to add the @menu:cake value.
16. Click the **Close**  icon to finish adding the @menu entity.

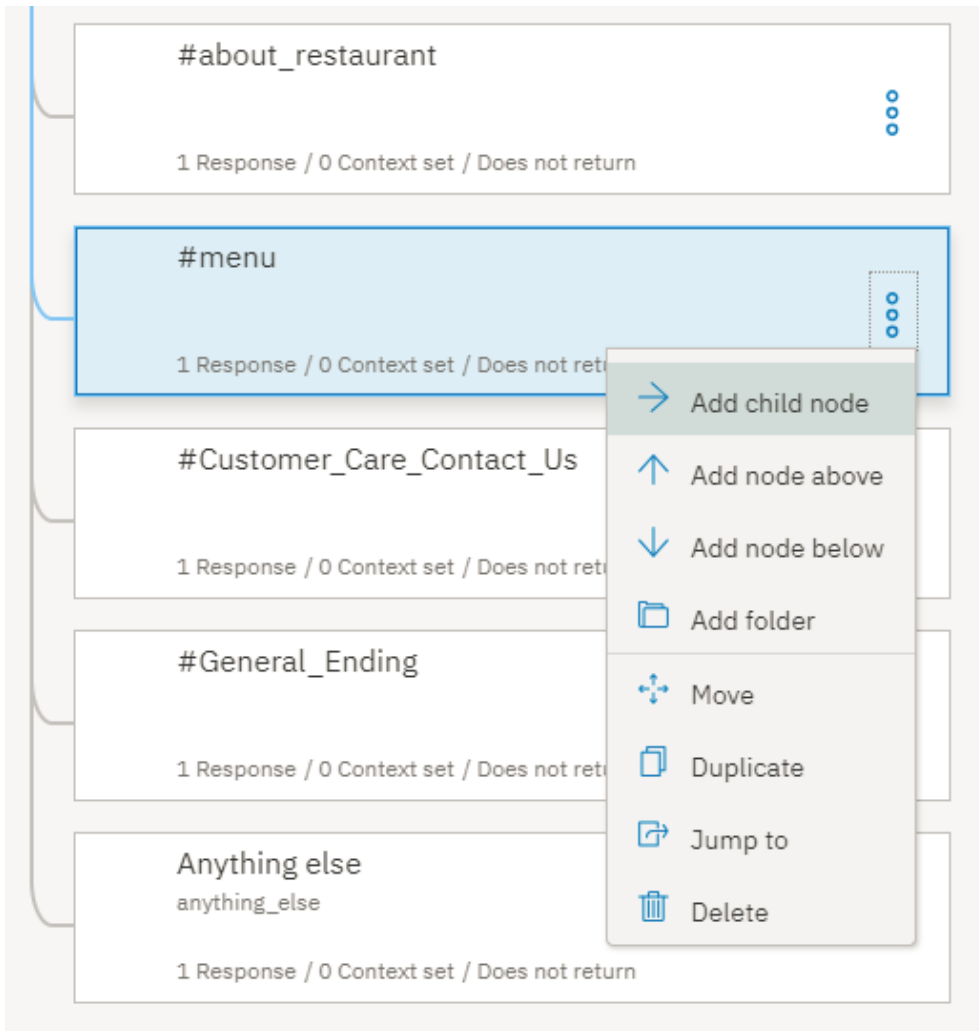
2.1.7. Add child nodes that are triggered by the @menu entity types

In this step, you will add child nodes to the dialog node that checks for the #menu intent. Each child node will show a different response depending on the @menu entity type the user chooses from the options list.

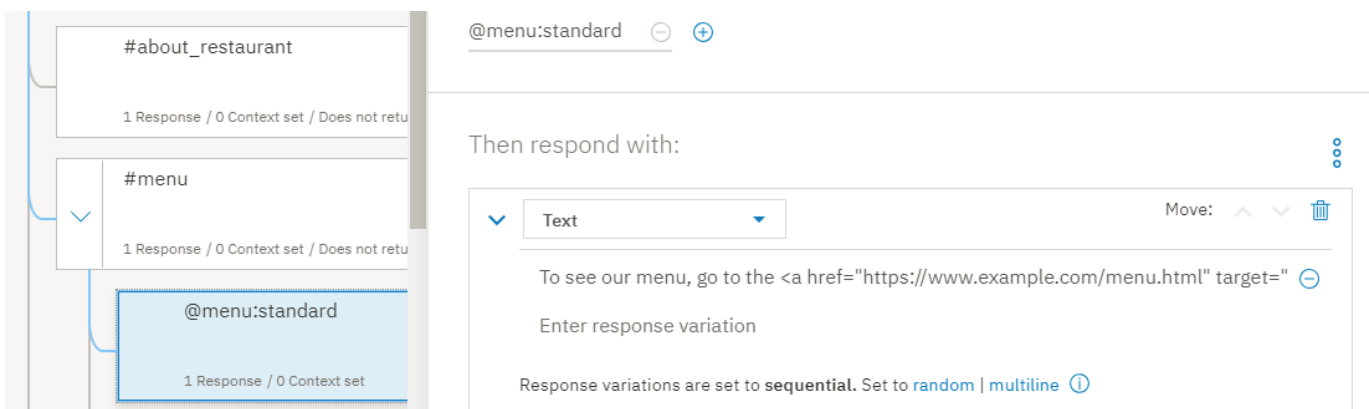
1. Click the **Dialogs** tab.
2. Find the #menu node in the dialog tree.


You will add a child node to handle each menu type option that you added to the #menu node.

3. Click the **More**  icon on the #menu node, and then select **Add child node**.

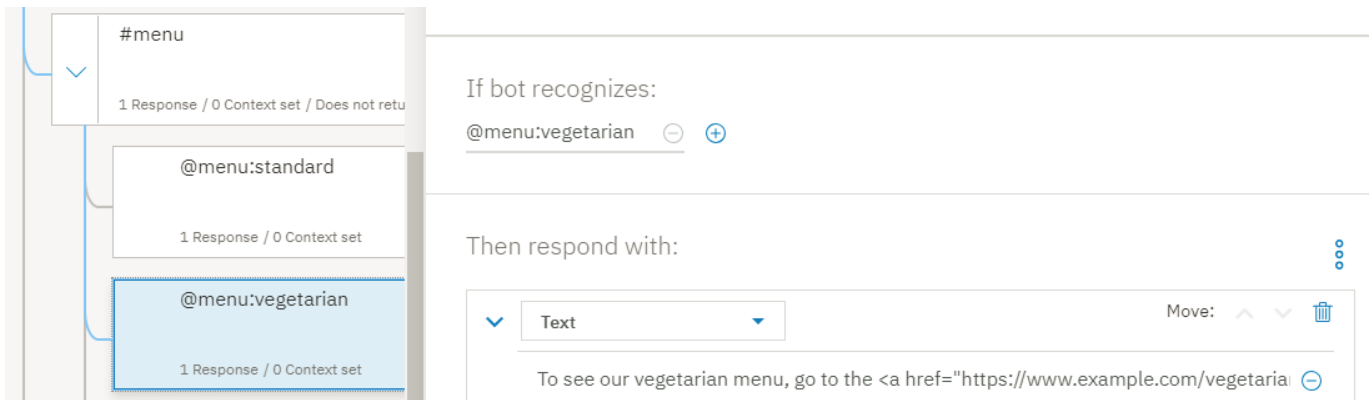



4. Start to type `@menu:standard` into the **Enter a condition** field of this node. Then select the `@menu:standard` option.
5. Add the following message in the response text field, To see our menu, go to the `menu` page on our website.

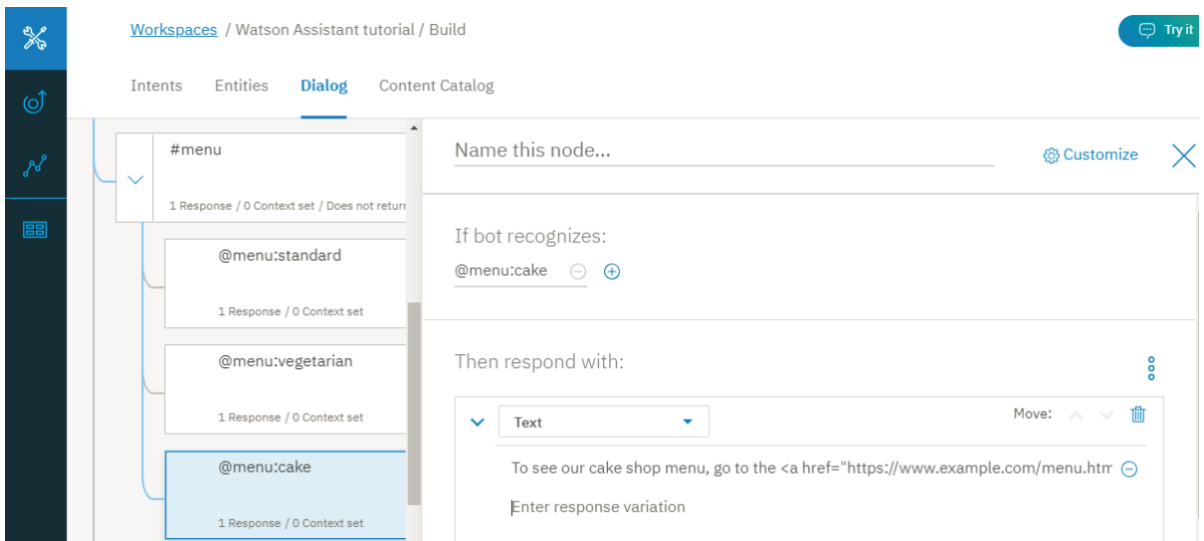



6. Click  to close the edit view.

7. Click the **More** icon on the @menu:standard node, and then select **Add node below**.
8. Start to type @menu:vegetarian into the **Enter a condition** field of this node. Then select the @menu:vegetarian option.
9. Add the following message in the response text field, To see our vegetarian menu, go to the vegetarian menu page on our website.

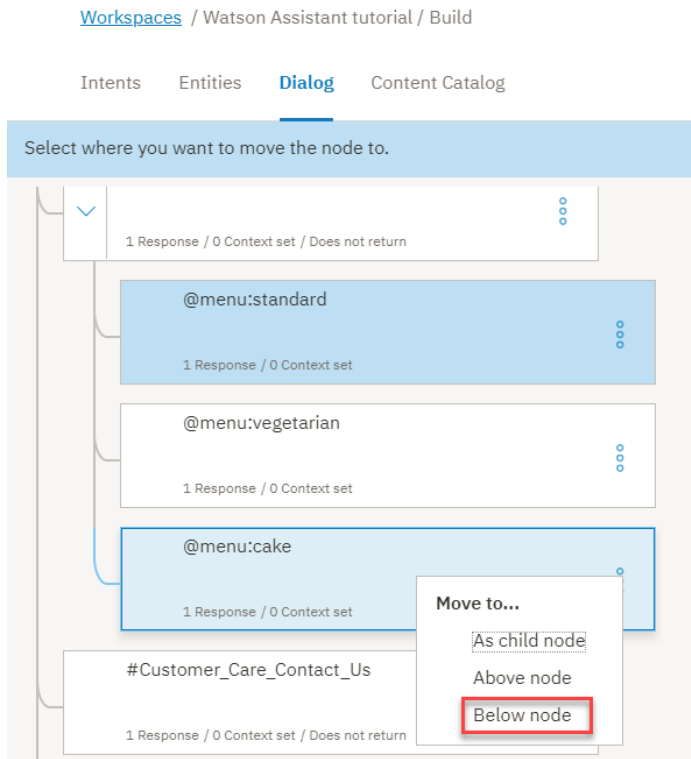


10. Click  to close the edit view.
11. Click the **More** icon on the @menu:vegetarian node, and then select **Add node below**.
12. Start to type @menu:cake into the **Enter a condition** field of this node. Then select the @menu:cake option.
13. Add the following message in the response text field, To see our cake shop menu, go to the cake shop menu page on our website.

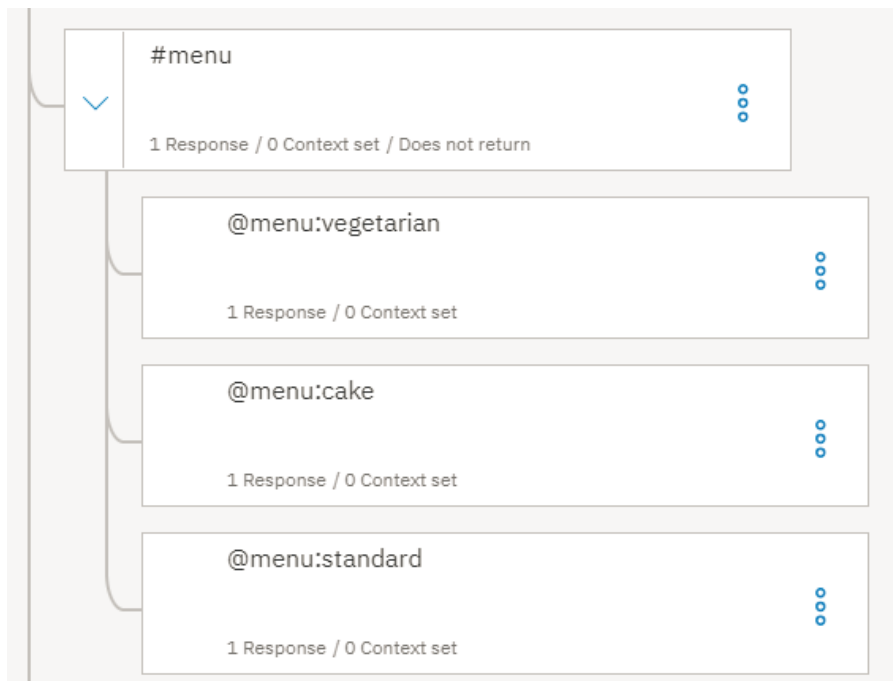


14. Click  to close the edit view.
15. The standard menu is likely to be requested most often, so move it to the bottom of the child nodes list. Placing it last can help prevent it from being triggered accidentally when someone asks for a specialty menu instead the standard menu.

16. Click the **More** icon on the @menu:standard node, and then select **Move**.




17. Select the @menu:cake node, and then choose **Below node**.



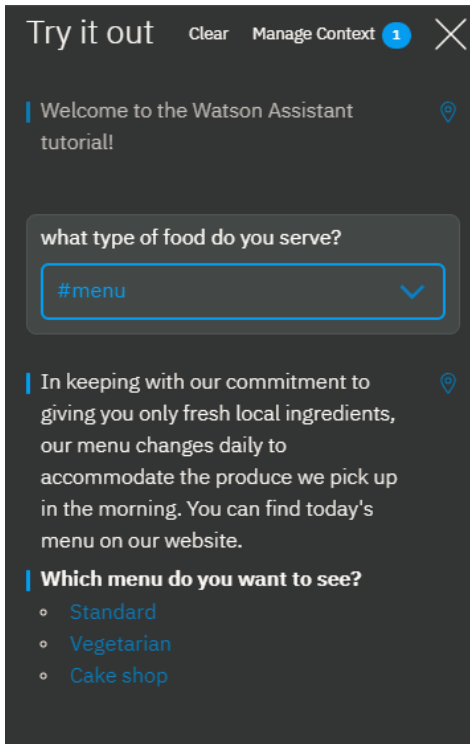
You have added nodes that recognize user requests for menu details. Your response informs the user that there are three types of menus available, and asks them to choose one. When the user chooses a menu type, a response is displayed that provides a hypertext link to a web page with the requested menu details.

2.1.8. Test the menu options dialog nodes

Test the dialog nodes that you added to recognize menu questions.

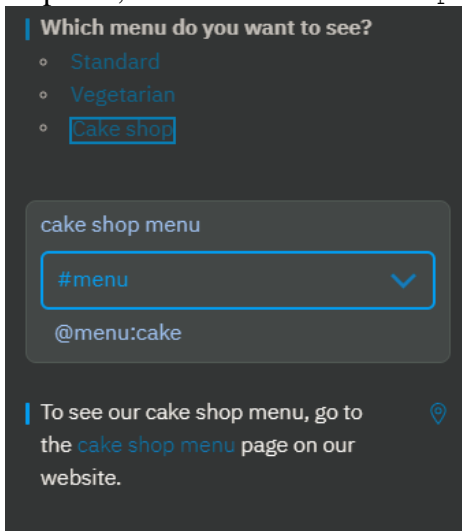
1. Click the  icon to open the "Try it out" pane.
2. Enter, What type of food do you serve?

The service indicates that the #menu intent is recognized, and displays the list of menu options for the user to choose from.



3. Click the `Cake shop` option.

The service recognizes the #menu intent and @menu:cake entity reference, and displays the response, To see our cake shop menu, go to the cake shop page on our website.



4. Click the *cake shop* hyperlink in the response.

A new web browser page opens and displays the example.com website.

5. Close the web browser page.

Well done. You have successfully added an intent and entity that can recognize user requests for menu details, and can direct users to the appropriate menu.

The #menu intent represents a common, key need of potential restaurant customers. Due to its importance and popularity, you added a more complex section to the dialog to address it well.

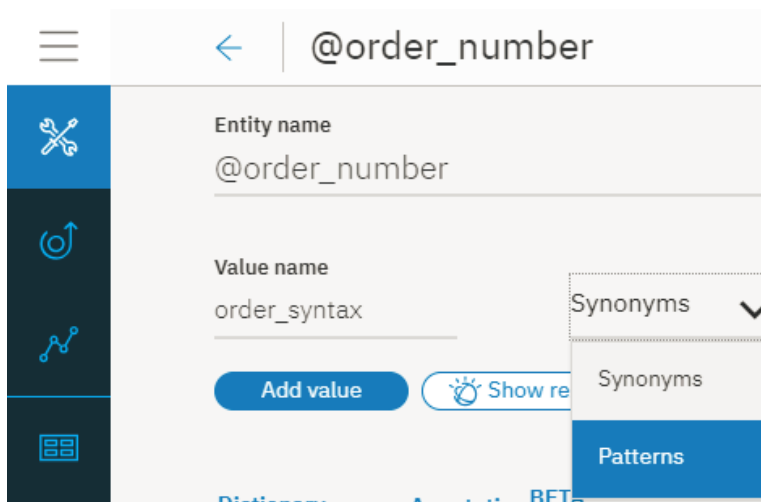
Step 4: Manage cake orders

Customers place orders in person, over the phone, or by using the order form on the website. After the order is placed, users can cancel the order through the virtual assistant. First, define an entity that can recognize order numbers. Then, add an intent that recognizes when users want to cancel a cake order.

2.1.9. Adding an order number pattern entity


You want the assistant to recognize order numbers, so you will create a pattern entity to recognize the unique format that the restaurant uses to identify its orders. The syntax of order numbers used by the restaurant's bakery is 2 upper-case letters followed by 5 numbers. For example, YR34663. Add an entity that can recognize this character pattern.

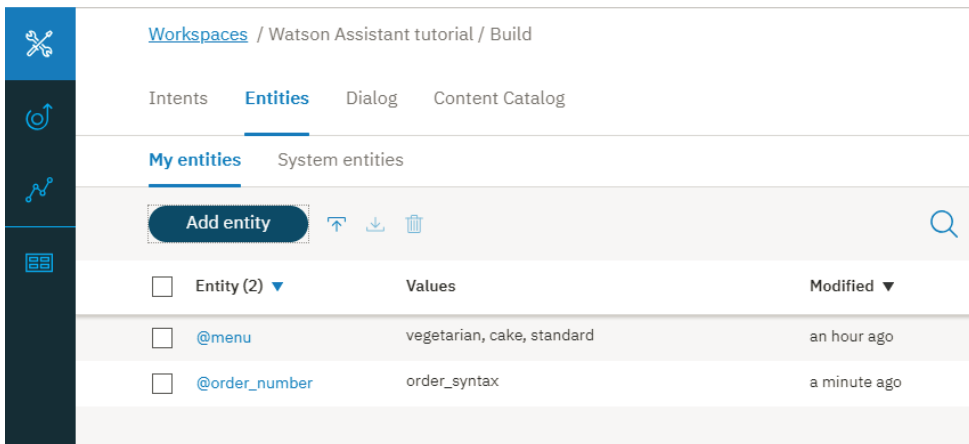
1. Click the **Entities** tab.
2. Click **Add entity**.
3. Enter `order_number` into the entity name field.
4. Click **Create entity**.
5. Add `order_syntax` to the *Value name* field, and then click the down arrow next to **Synonyms** to change the type to **Patterns**.



6. Add the following regular expression to the Pattern field: `[A-Z]{2}\d{5}`

7. Click **Add value**.

8. Click the **Close**  icon to finish adding the `@order_number` entity.

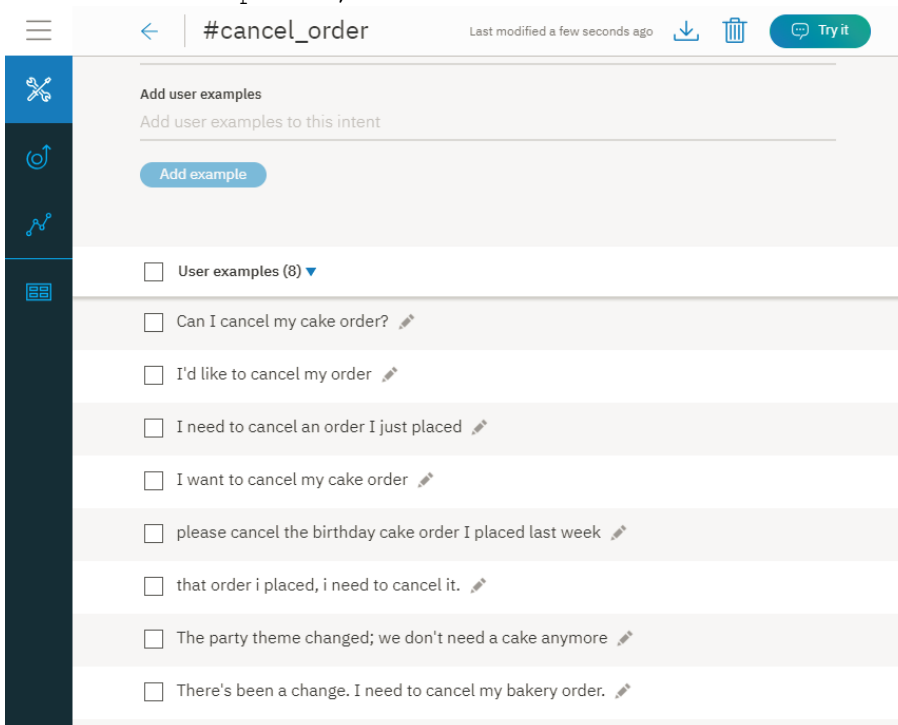


The screenshot shows the 'Entities' tab in the Watson Assistant interface. The breadcrumb path is 'Workspaces / Watson Assistant tutorial / Build'. The 'Entities' tab is selected, showing 'My entities' and 'System entities'. There is an 'Add entity' button and icons for up, down, and delete. A table lists the entities:









Entity (2) ▼	Values	Modified ▼
<input type="checkbox"/> @menu	vegetarian, cake, standard	an hour ago
<input type="checkbox"/> @order_number	order_syntax	a minute ago


2.1.10. Add a cancel order intent

1. Click the **Intents** tab.
2. Click **Add intent**.
3. Enter `cancel_order` in the *Intent name* field, and then click **Create intent**.
4. Add the following user examples:
5. I want to cancel my cake order
6. I need to cancel an order I just placed
7. Can I cancel my cake order?
8. I'd like to cancel my order
9. There's been a change. I need to cancel my bakery order.
10. please cancel the birthday cake order I placed last week
11. The party theme changed; we don't need a cake anymore
12. that order i placed, i need to cancel it.



The screenshot shows the 'Intents' tab in the Watson Assistant interface. The breadcrumb path is 'Workspaces / Watson Assistant tutorial / Build'. The 'Intents' tab is selected, showing the 'cancel_order' intent. The 'Add user examples' section is visible, with an 'Add example' button. A table lists the user examples:

User examples (8) ▼
<input type="checkbox"/> Can I cancel my cake order? 
<input type="checkbox"/> I'd like to cancel my order 
<input type="checkbox"/> I need to cancel an order I just placed 
<input type="checkbox"/> I want to cancel my cake order 
<input type="checkbox"/> please cancel the birthday cake order I placed last week 
<input type="checkbox"/> that order i placed, i need to cancel it. 
<input type="checkbox"/> The party theme changed; we don't need a cake anymore 
<input type="checkbox"/> There's been a change. I need to cancel my bakery order. 

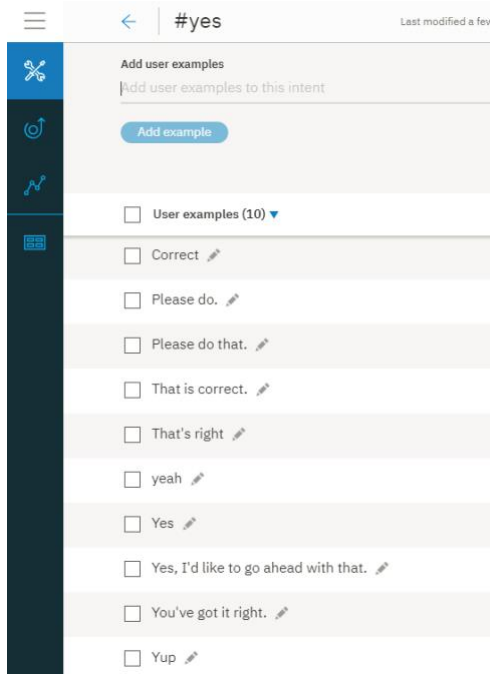
13. Click the **Close**  icon to finish adding the #cancel_order intent.


2.1.11. Add a yes intent

Before you perform an action on the user's behalf, you must get confirmation that you are taking the proper action. Add a #yes intent to the dialog that can recognize when a user agrees with what the service is proposing.

1. Click the **Intents** tab.
2. Click **Add intent**.
3. Enter yes in the *Intent name* field, and then click **Create intent**.
4. Add the following user examples:

Yes
Correct
Please do.
You've got it right.
Please do that.
that is correct.
That's right
yeah
Yup
Yes, I'd like to go ahead with that.




5. Click the **Close**  icon to finish adding the #yes intent.

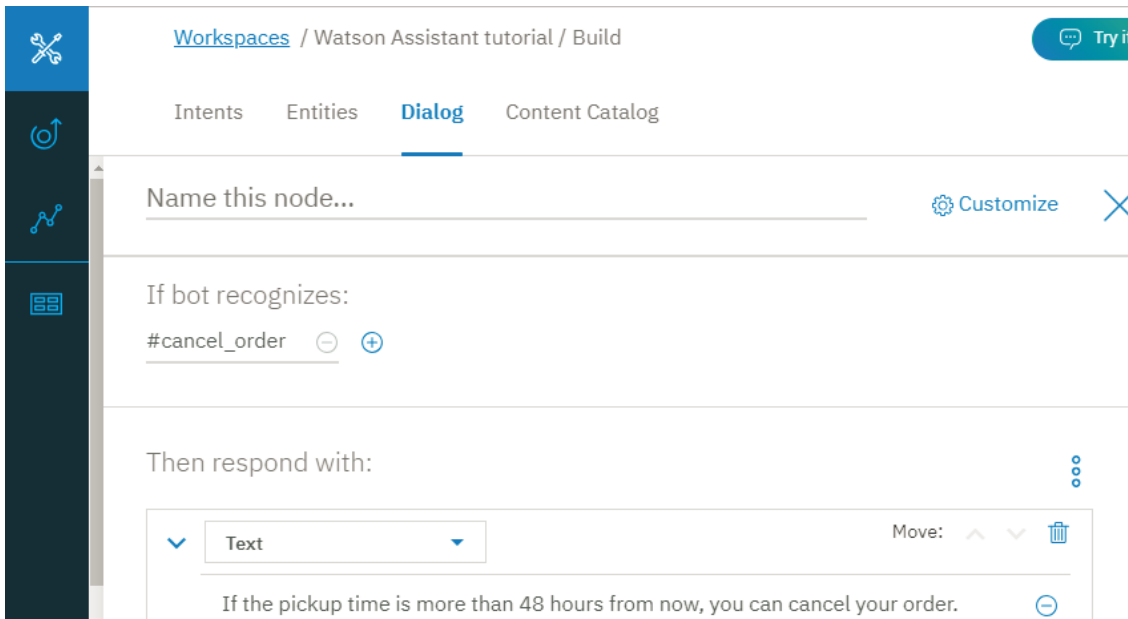
2.1.12. Add dialog nodes that can manage requests to cancel an order

Now, add a dialog node that can handle requests to cancel a cake order.

1. Click the **Dialog** tab.

2. Find the `#menu` node. Click the **More**  icon on the `#menu` node, and then select **Add node below**.
3. Start to type `#cancel_order` into the **Enter a condition** field of this node. Then select the `#cancel_order` option.
4. Add the following message in the response text field:

If the pickup time is more than 48 hours from now, you can cancel your order.



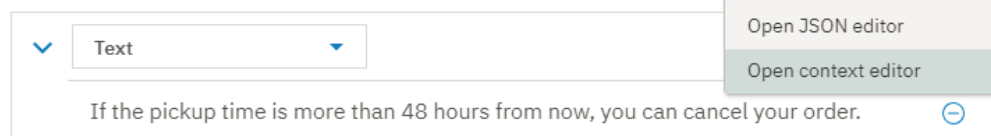
Before you can actually cancel the order, you need to know the order number. The user might specify the order number in the original request. So, to avoid asking for the order number again, check for a number with the order number pattern in the original input. To do so, define a context variable that would save the order number if it is specified.

1. Open the context editor. Click the **More**  icon, and select **Open context editor**.

If bot recognizes:

`#cancel_order`  

Then respond with:



- Enter the following context variable name and value pair:

Order number context variable details

Variable	Value
<code>\$ordernumber</code>	<code><? @order_number.literal ?></code>

The context variable value (`<? @order_number.literal ?>`) is a SpEL expression that captures the number that the user specifies that matches the pattern defined by the `@order_number` pattern entity. It saves it to the `$ordernumber` variable.

If bot recognizes:

#cancel_order

Then set context:

Variable	Value
\$ordernumber	<? @order_number.literal ?>


+ Add variable


And respond with:

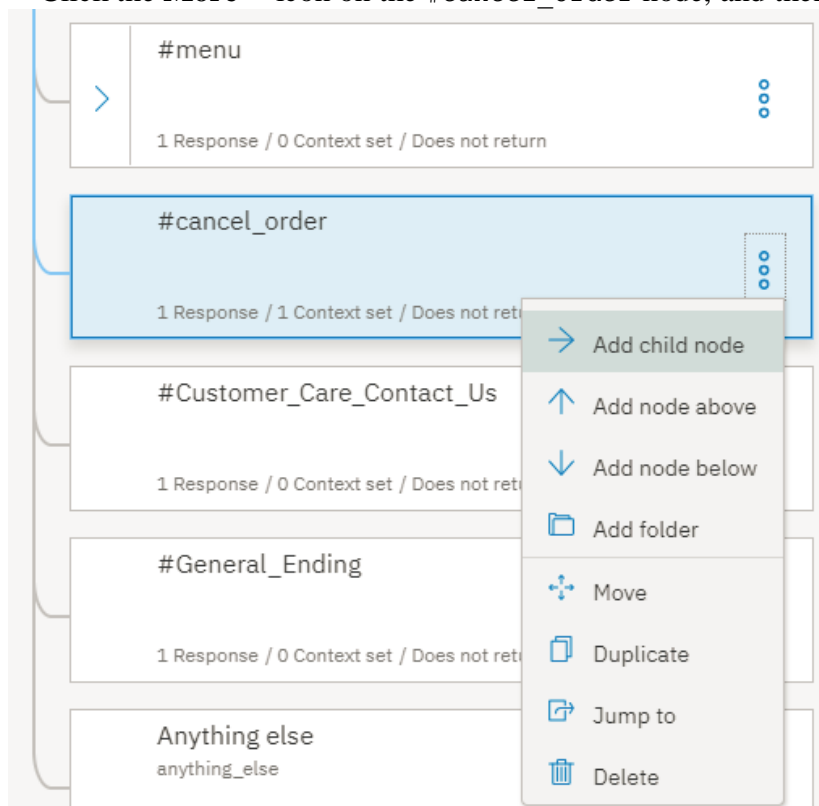
Text

Move: ^ v

If the pickup time is more than 48 hours from now, you can cancel your order.

- Click  to close the edit view.
- Now, add child nodes that either ask for the order number or get confirmation from the user that she wants to cancel an order with the detected order number.

- Click the **More**  icon on the #cancel_order node, and then select **Add child node**.



- Add a label to the node to distinguish it from other child nodes you will be adding. In the name field, add Ask for order number. Type true into the **Enter a condition** field of this node.

- Add the following message in the response text field:

- What is the order number?

- Click to close the edit view.

Now, add another child node that informs the user that you are canceling the order.

- Click the **More** icon on the Ask for order number node, and then select **Add child node**.
- Type `@order_number` into the **Enter a condition** field of this node.
- Open the context editor. Click the **More** icon, and select **Open context editor**.
- Enter the following context variable name and value pair:



Order number context variable details

Variable	Value
\$ordernumber	<? @order_number.literal ?>

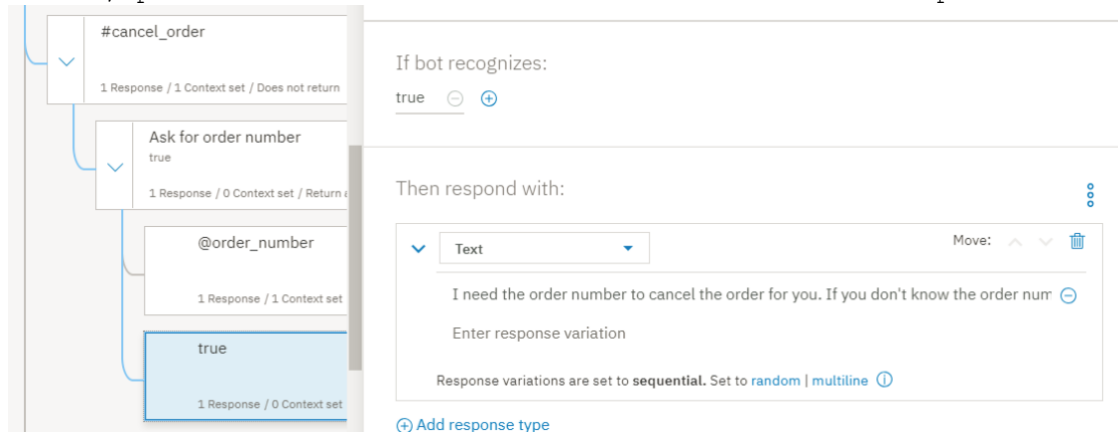
The context variable value (`<? @order_number.literal ?>`) is a SpEL expression that captures the number that the user specifies that matches the pattern defined by the `@order_number` pattern entity. It saves it to the `$ordernumber` variable.

- Add the following message in the response text field:



- Ok. The order `$ordernumber` is canceled. We hope we get the opportunity to bake a cake for you sometime soon.

- Click  to close the edit view.
- Add another node to capture the case where a user provides a number, but it is not a valid order number. Click the **More**  icon on the @order_number node, and then select **Add node below**.
- Type true into the **Enter a condition** field of this node.
- Add the following message in the response text field:

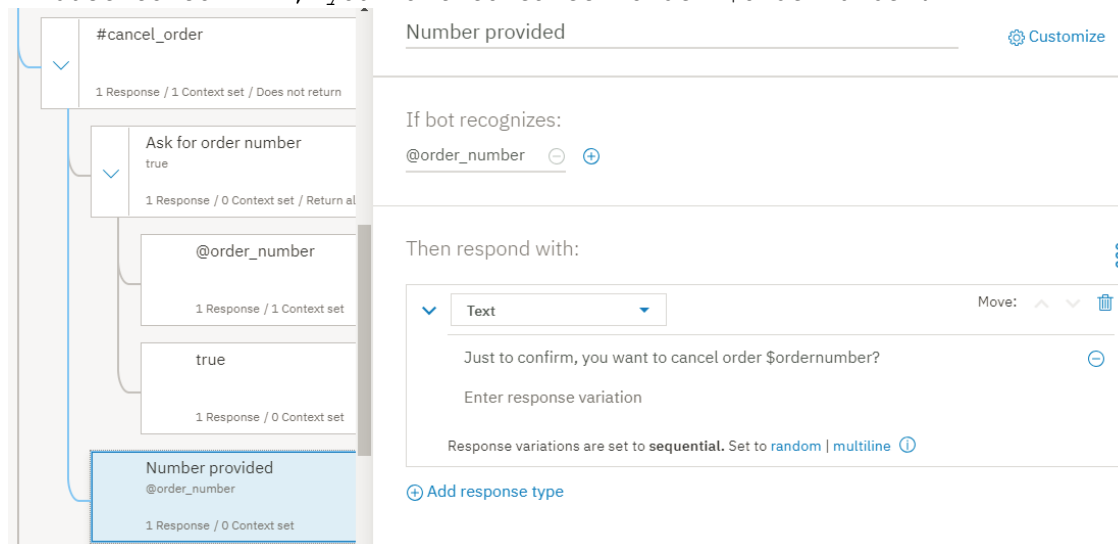
I need the order number to cancel the order for you. If you don't know the order number, please call us at 958-234-3456 to cancel over the phone.




The screenshot shows a flowchart on the left with three nodes: '#cancel_order' (1 Response / 1 Context set / Does not return), 'Ask for order number' (true, 1 Response / 0 Context set / Return al), and '@order_number' (1 Response / 1 Context set). A new node 'true' (1 Response / 0 Context set) is being added below '@order_number'. On the right, the 'Then respond with:' panel is configured with a 'Text' response type. The response text is 'I need the order number to cancel the order for you. If you don't know the order num'. Below the text field, it says 'Response variations are set to sequential. Set to random | multiline'.

- Click  to close the edit view.
- Add a node below the initial order cancelation request node that responds in the case where the user provides the order number in the initial request, so you don't have to ask for it again. Click the **More**  icon on the #cancel_order node, and then select **Add child node**.
- Add a label to the node to distinguish it from other child nodes. In the name field, add Number provided. Type @order_number into the **Enter a condition** field of this node.
- Add the following message in the response text field:


Just to confirm, you want to cancel order \$ordernumber?



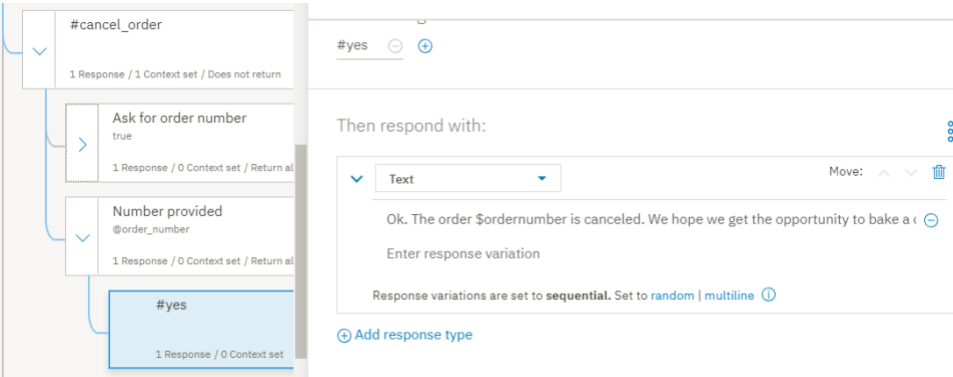
The screenshot shows a flowchart on the left with four nodes: '#cancel_order' (1 Response / 1 Context set / Does not return), 'Ask for order number' (true, 1 Response / 0 Context set / Return al), '@order_number' (1 Response / 1 Context set), and 'Number provided' (@order_number, 1 Response / 0 Context set). On the right, the 'Then respond with:' panel is configured with a 'Text' response type. The response text is 'Just to confirm, you want to cancel order \$ordernumber?'. Below the text field, it says 'Response variations are set to sequential. Set to random | multiline'.


- Click  to close the edit view.


You must add child nodes that check for the user's response to your confirmation question.

- Click the **More**  icon on the `Number provided` node, and then select **Add child node**.
- Type `#yes` into the **Enter a condition** field of this node.
- Add the following message in the response text field:

28. Ok. The order \$ordernumber is canceled. We hope we get the opportunity to bake a cake for you sometime soon.

29. 

30. Click  to close the edit view.

31. Click the **More**  icon on the `#yes` node, and then select **Add node below**.

32. Type `true` into the **Enter a condition** field of this node.

Do not add a response. Instead, you will redirect users to the branch that asks for the order number details that you created earlier.

33. In the *And finally* section, choose **Jump-to**.

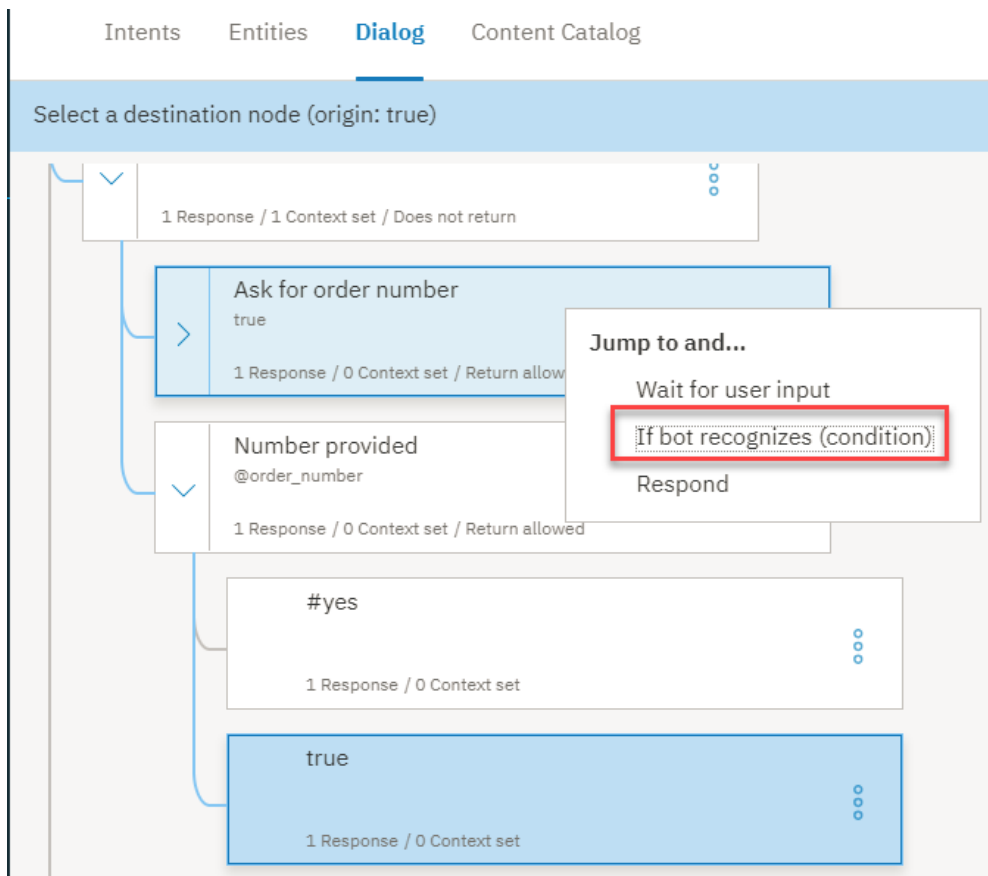
The screenshot displays a workflow editor interface. On the left, a vertical list of nodes is shown, connected by a blue line indicating the flow. The nodes are:


- #cancel_order**: 1 Response / 1 Context set / Does not return
- Ask for order number**: true (1 Response / 0 Context set / Return al)
- Number provided**: @order_number (1 Response / 0 Context set / Return al)
- #yes**: (1 Response / 0 Context set)
- true**: (1 Response / 0 Context set)
- #Customer_Care_Contact_Us**


On the right, a configuration panel is visible with the following sections:

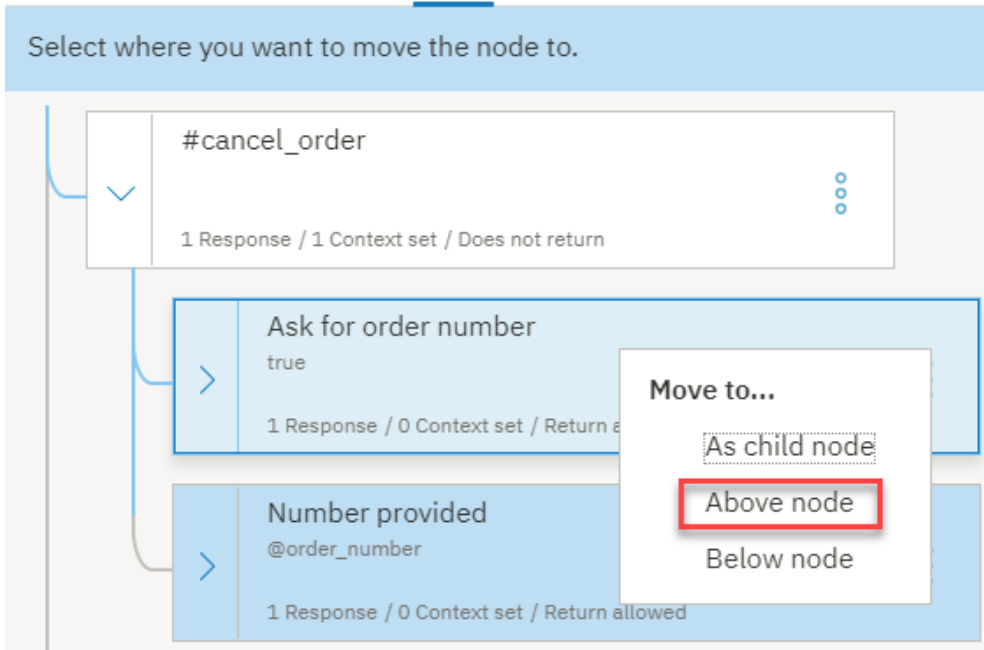
- Name this node...**: A text input field.
- Then respond with:**: A dropdown menu set to **Text**. Below it, a text input field labeled "Enter response text" and a label "Response variations are set to sequ".
- + Add response type**: A button.
- And finally**: A section with a dropdown menu set to **Wait for user input**. Below it, a list of options: **Wait for user input**, **Skip user input** (with an information icon), and **Jump to...**.

34. Select the *Ask for order number* node's condition.

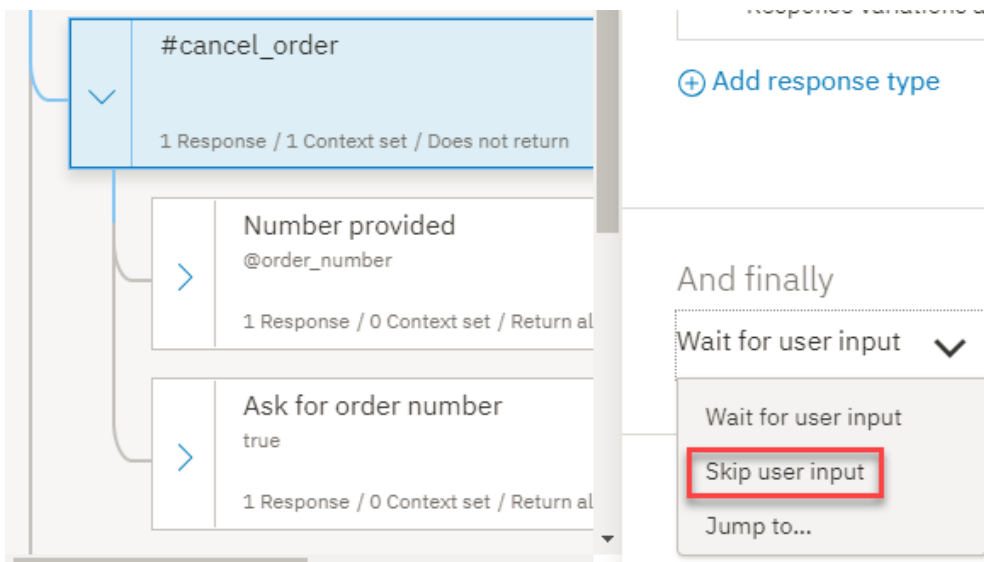


35. Click  to close the edit view.

36. Move the *Number provided* node above the *Ask for order number* node. Click the **More**  icon on the *Number provided* node, and then select **Move**. Select the *Ask for order number* node, and then click **Above node**.




37. Force the conversation to evaluate the child nodes under the #cancel_order node at run time. Click to open the #cancel_order node in the edit view, and then, in the And finally section, select Skip user input.



2.1.13. Test order cancelations

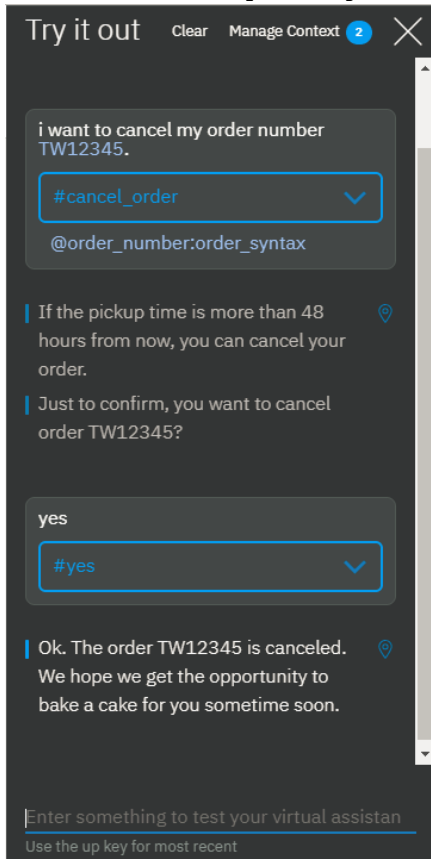
Test whether the service can recognize character patterns that match the pattern used for product order numbers in user input.

1. Click the  icon to open the "Try it out" pane.
2. Enter, i want to cancel my order number TW12345.

The service recognizes both the `#cancel_order` intent and the `@order_number` entity. It responds with, If the pickup time is more than 48 hours from now, you can cancel your order. Just to confirm, you want to cancel order TW12345?

3. Enter, Yes.

The service recognizes the `#yes` intent and responds with, Ok. The order TW12345 is canceled. We hope we get the opportunity to bake a cake for you sometime soon.



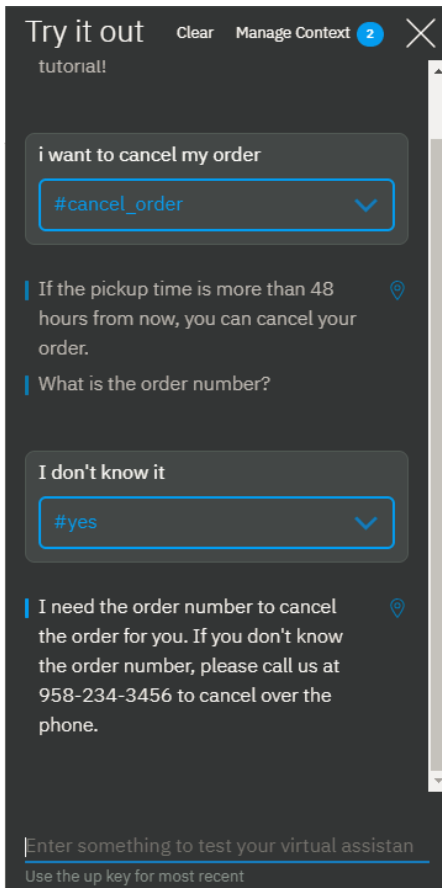
Now, try it when you don't know the order number.

4. Click **Clear** in the "Try it out" pane to start over. Enter, I want to cancel my order.




The service recognizes the `#cancel_order` intent, and responds with, If the pickup time is more than 48 hours from now, you can cancel your order. What is the order number?

5. Enter, I don't know.

The service responds with, I need the order number to cancel the order for you. If you don't know the order number, please call us at 958-234-3456 to cancel over the phone.

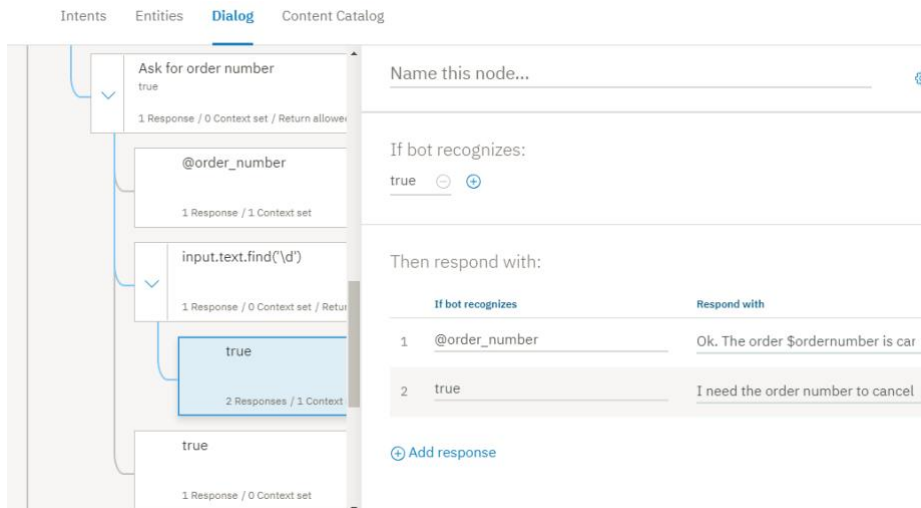


If you do more testing, you might find that the dialog isn't very helpful in scenarios where the user does not remember the order number format. The user might include only the numbers or the letters too, but forget that they are meant to be uppercase. So, it would be a nice touch to give them a hint in such cases, right? If you want to be kind, add another node to the dialog tree that checks for numbers in the user input.


1. Find the `@order-number` node that is a child of the *Ask order number* node.
2. Click the **More**  icon on the `@order-number` node, and then select **Add node below**.
3. In the condition field, add `input.text.find('\d')`, which is a SpEL expression that says if you find one or more numbers in the user input, trigger this response.
4. In the text response field, add the following response:
 - 5.
 - 6.
- The correct format for our order numbers is AAnnnnn. The A's represents 2 upper-case letters, and the n's represents 5 numbers. Do you have an order number in that format?
- Click  to close the edit view.
- Click the **More**  icon on the `input.text.find('\d')` node, and then select **Add child node**.
- Type `true` into the **Enter a condition** field of this node.
- Enable conditional responses by clicking **Customize**, and then switching the *Multiple responses* toggle to **on**.
- Click **Apply**.
- In the newly-added *If bot recognizes* field, type `@order_number`, and in the *Respond with* field, type:

- Ok. The order \$ordernumber is canceled. We hope we get the opportunity to bake a cake for you sometime soon.
- Click **Add response**.
- In the *If bot recognizes* field, type `true`, and in the *Respond with* field, type:

12. I need the order number to cancel the order for you. If you don't know the order number, please call us at 958-234-3456 to cancel over the phone.



13.

14. Click  to close the edit view.

Now, when you test, you can provide a set of number or a mix of numbers and text as input, and the dialog reminds you of the correct order number format. You have successfully tested your dialog, found a weakness in it, and corrected it.

Another way you can address this type of scenario is to add a node with slots. See the [Adding a node with slots to a dialog](#) tutorial to learn more about using slots.

Step 5: Add the personal touch

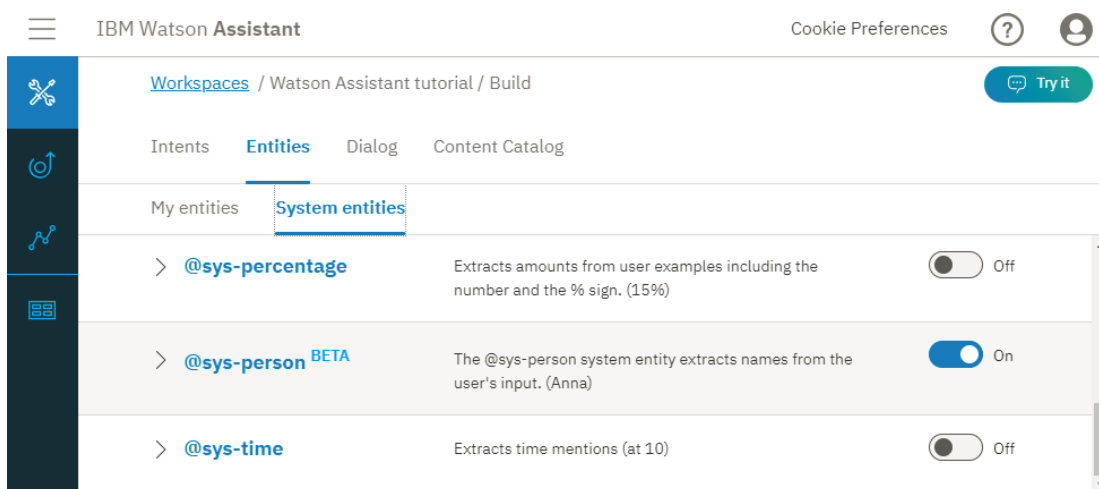
If the user shows interest in the bot itself, you want the virtual assistant to recognize that curiosity and engage with the user in a more personal way. You might remember the `#General_About_You` intent, which is provided with the *General* content catalog, that we considered using earlier, before you added your own custom `#about_restaurant` intent. It is built to recognize just such questions from the user. Add a node that condition on this intent. In your response, you can ask for the user's name and save it to a `$username` variable that you can use elsewhere in the dialog, if available.

First, you need to make sure the service will recognize a name if the user provides one. So, you can enable the `@sys-person` entity, which is designed to recognize common first and last names (in English).

2.1.14. Add a person system entity

The service provides a number of *system entities*, which are common entities that you can use for any application.

1. Click the **Entities** tab, and then click **System entities**.
2. Find the `@sys-person` entity toggle, and then switch it **On**.

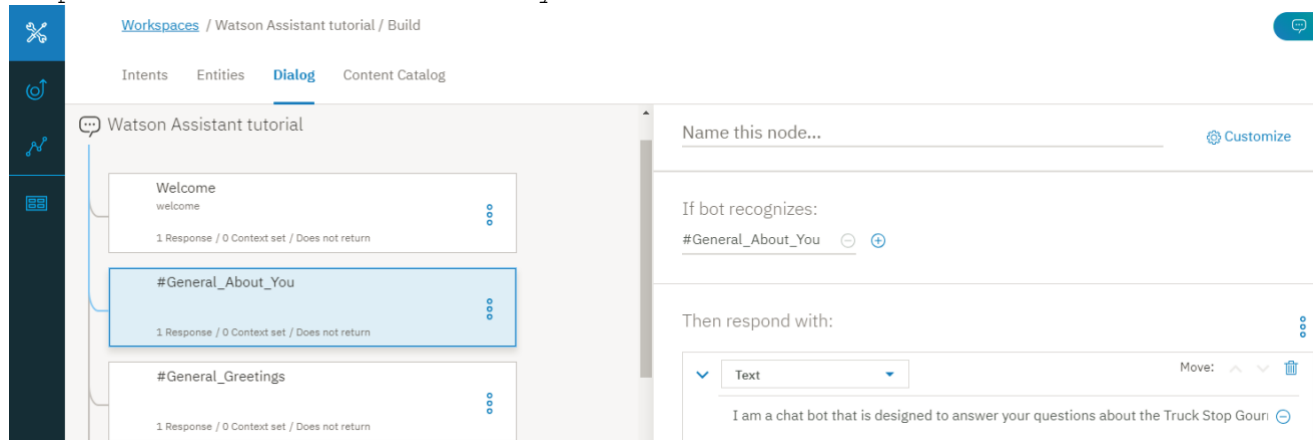



2.1.15. Add a node that handles questions about the bot

Now, add a dialog node that can recognize the user's interest in the bot, and respond.

1. Click the **Dialogs** tab.
2. Find the `Welcome` node in the dialog tree.
3. Click the **More** icon on the `Welcome` node, and then select **Add node below**.
4. Start to type `#General_About_You` into the **Enter a condition** field of this node. Then select the `#General_About_You` option.
5. Add the following message in the response text field:
- 6.
- 7.

- I am a virtual assistant that is designed to answer your questions about the Truck Stop Gourmand restaurant. What's your name?



- Click  to close the edit view.
- Click the **More** icon on the `#General_About_You` node, and then select **Add child node**.
- Start to type `@sys-person` into the **Enter a condition** field of this node. Then select the `@sys-person` option.
- Add the following message in the response text field:

- Hello, <? @sys-person.literal ?>! It's lovely to meet you. How can I help you today.

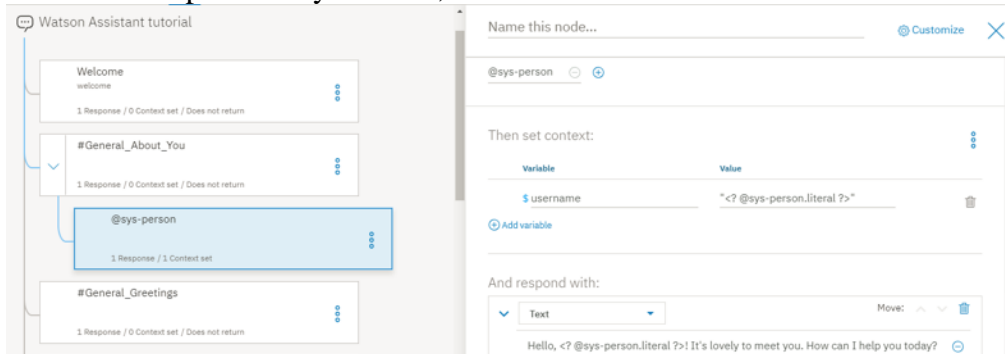
- To capture the name that the user provides, add a context variable to the node. Click the **More** icon, and select **Open context editor**.
- Enter the following context variable name and value pair:

User name context variable details

Variable	Value
\$username	<? @sys-person.literal ?>

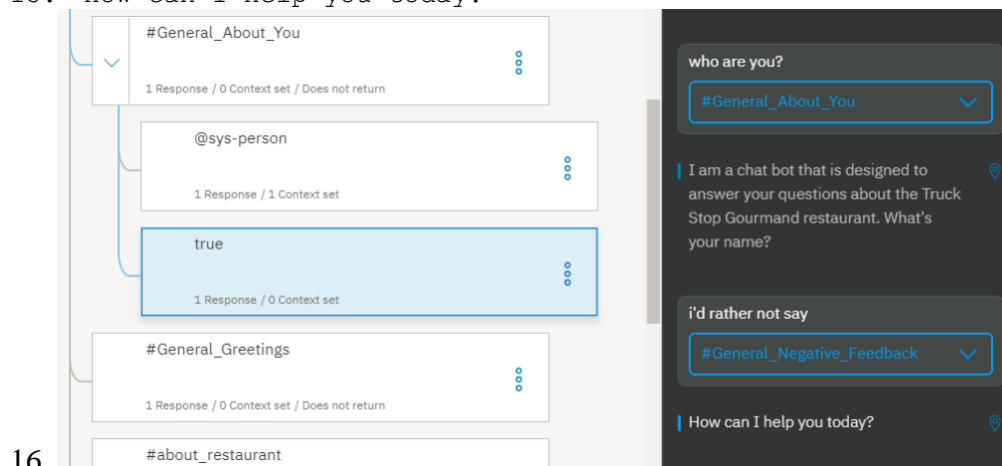
\$username <? @sys-person.literal ?>

The context variable value (<? @sys-person.literal ?>) is a SpEL expression that captures the user name as it is specified by the user, and then saves it to the \$username context variable.



- Click to close the edit view.
- Click the **More** icon on the @sys-person node, and then select **Add node below**. You will add a node to capture user responses that do not include a name. If the user chooses not to share it, you want the bot to keep the conversation going anyhow.
- Type true into the **Enter a condition** field of this node.
- Add the following message in the response text field:

15. How can I help you today?



16. #about_restaurant

- 17. Click to close the edit view.

If, at run time, the user triggers this node and provides a name, then you will know the user's name. If you know it, you should use it! Add conditional responses to the greeting dialog node you added previously to include a conditional response that uses the user name, if it is known.

2.1.16. Add the user name to the greeting

If you know the user's name, you should include it in your greeting message. To do so, add conditional responses, and include a variation of the greeting that includes the user's name.

1. Find the `#General_Greetings` node in the dialog tree, and click to open it in the edit view.
2. Click **Customize**, and then switch the *Multiple responses* toggle to **on**.

Customize "#General_Greetings"

[Customize node](#) [Digressions](#)

Slots ⓘ ☐ off

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 ⓘ ☒ on

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

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

3. Click **Apply**.

Workspaces / Watson Assistant tutorial / Build

[Try it](#)

Intents Entities **Dialog** Content Catalog

Watson Assistant tutorial

Welcome
welcome
1 Response / 0 Context set / Does not return

#General_About_You
1 Response / 0 Context set / Does not return

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

#about_restaurant
1 Response / 0 Context set / Does not return

Name this node...

[Customize](#) [X](#)

If bot recognizes:

#General_Greetings

Then respond with:

If bot recognizes	Respond with
1	Enter an intent, entity or context va

[Add response](#)

4. Click **Add response**.
5. In the *If bot recognizes* field, type `$username`, and in the *Respond with* field, type:
6.
7. Good day to you, `$username`!
- 6.
7. Click the up arrow for response number 2 to move it so it is listed before response number 1 (Good day to you!).

Name this node... Customize ✕

If bot recognizes:

#General_Greetings ⊖ ⊕

Then respond with:

	If bot recognizes	Respond with		
1	\$username	Good day to you, \$username!	⚙️	🗑️
2	Enter an intent, entity or context va	Good day to you!	⚙️	🗑️

⊕ Add response

- Click ✕ to close the edit view.

2.1.17. Test personalization

Test whether the service can recognize and save a user's name, and then refer to the user by it later.

- Click the Try it icon to open the "Try it out" pane.
- Click **Clear** to restart the conversation session.
- Enter, Who are you?

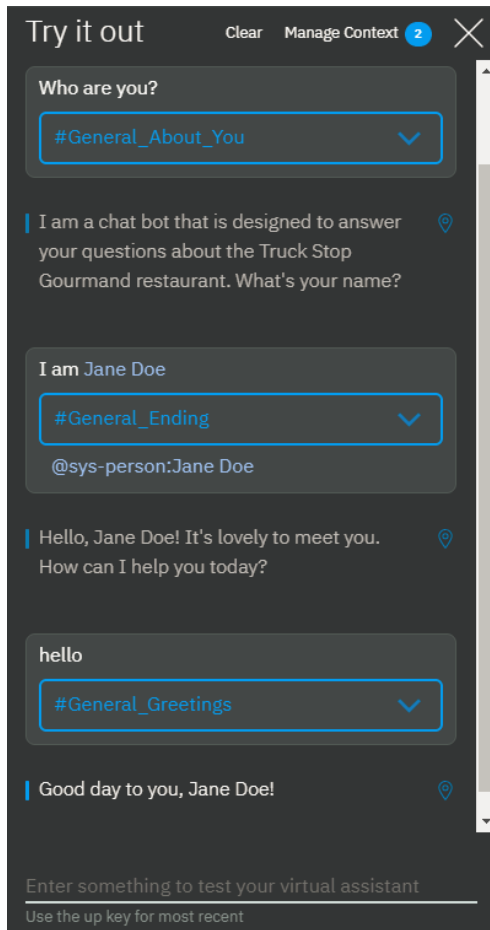
The service recognizes the #General_About_You intent. Its response ends with the question, What's your name?

- Enter, I am Jane Doe.

The service recognizes Jane Doe as an @sys-person entity mention. It comments on your name, and then asks how it can help you.

- Enter, Hello.

The service recognizes the #General_Greetings intent and says, Good day to you, Jane Doe! It uses the conditional response that includes the user's name because the \$username context variable contains a value at the time that the greeting node is triggered.



You can add a conditional response that conditions on and includes the user's name for any other responses where personalization would add value to the conversation.

Step 6: Test the assistant from you web page integration

Now that you have built a more sophisticated version of the assistant, return to the public web page that you deployed as part of the previous tutorial, and then test the new capabilities you added.

1. Open the assistant.
2. From the *Integrations* area, click **Preview Link**.
3. Click the URL that is displayed on the page.

The page opens in a new tab.

4. Repeat a few of the test utterances that you submitted to the "Try it out pane" to see how the assistant behaves in a real integration.

Unlike when you send test utterances to the service from the "Try it out" pane, standard usage charges apply to API calls that result from utterances that are submitted to the chat widget.

Next steps

Now that you have built and tested your dialog skill, you can share it with customers. Deploy your skill by first connecting it to an assistant, and then deploying the assistant. There are several ways you can do this. See [Adding integrations](#) for more details.