[**IBM Cloud Docs**](https://console.bluemix.net/docs/)[**Conversation**](https://console.bluemix.net/docs/services/conversation/getting-started.html)

Getting started tutorial

Last Updated: 2017-08-10[Edit in GitHub](https://github.com/IBM-Bluemix-Docs/conversation/blob/master/getting-started.md)

In this short tutorial, we introduce the Conversation tool and go through the process of creating your first conversation.

**Before you begin**

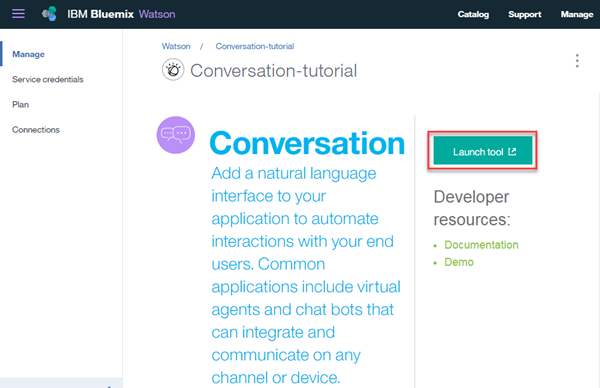
If you already created a service instance, you're all set with these prerequisites. Move on to Step 1.

1. Go to the [Conversation service](https://console.eu-gb.bluemix.net/catalog/services/conversation/)and either sign up for a free Bluemix account or log in.
2. After you log in, type conversation-tutorial in the **Service name** field of the Conversation page and click **Create**.

**Step 1: Launch the tool**

After you create the service instance, you'll land on the dashboard for the instance. Launch the Conversation tool from here.

Click **Manage**, then **Launch tool**.



You might be prompted to log in to the tool separately. If so, provide your IBM Bluemix credentials to log in.

**Step 2: Create a workspace**

Your first step in the Conversation tool is to create a workspace.

A [*workspace*](https://console.bluemix.net/docs/services/conversation/configure-workspace.html) is a container for the artifacts that define the conversation flow.

1. In the Conversation tool, click **Create**.
2. Give your workspace the name Conversation tutorial and click **Create**. Youʼll land on the **Intents** tab of your new workspace.

**Step 3: Create intents**

An [intent](https://console.bluemix.net/docs/services/conversation/intents.html) represents the purpose of a user's input. You can think of intents as the actions your users might want to perform with your application.

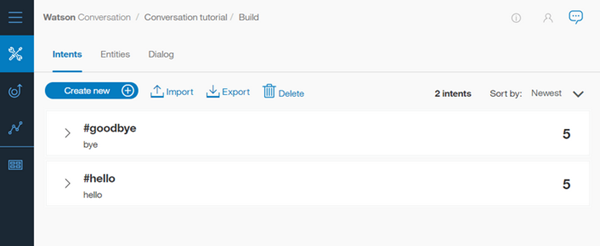
For this example, we're going to keep things simple and define only two intents: one for saying hello, and one for saying goodbye.

1. Make sure you're on the Intents tab. (You should already be there, if you just created the workspace.)
2. Click **Create new**.
3. Name the intent hello.
4. Type hello as a **User example** and press Enter.

*Examples* tell the Conversation service what kinds of user input you want to match to the intent. The more examples you provide, the more accurate the service can be at recognizing user intents.

1. Add four more examples and click **Done** to finish creating the #hello intent:
   * good morning
   * greetings
   * hi
   * howdy
2. Create another intent named #goodbye with these five examples:
   * bye
   * farewell
   * goodbye
   * I'm done
   * see you later

You've created two intents, #hello and #goodbye, and provided example user input to train Watson to recognize these intents in your users' input.



**Step 4: Build a dialog**

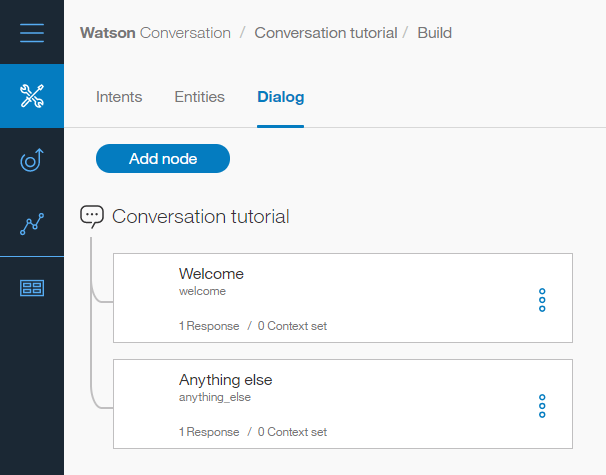
A [dialog](https://console.bluemix.net/docs/services/conversation/dialog-build.html) defines the flow of your conversation in the form of a logic tree. Each node of the tree has a condition that triggers it, based on user input.

We'll create a simple dialog that handles our #hello and #goodbye intents, each with a single node.

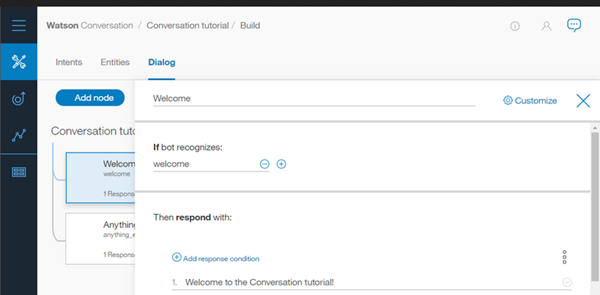
**Adding a start node**

1. In the Conversation tool, click the **Dialog** tab.
2. Click **Create**. You'll see two nodes:

* **Welcome**: Contains a greeting that is displayed to your users when they first engage with the bot.
* **Anything else**: Contains phrases that are used to reply to users when their input is not recognized.



1. Click the **Welcome** node to open it in the edit view.
2. Replace the default response with the text, Welcome to the Conversation tutorial!.



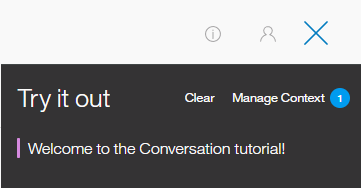
1. Click Close to close the edit view.

You created a dialog node that is triggered by the welcome condition, which is a special condition that indicates that the user has started a new conversation. Your node specifies that when a new conversation starts, the system should respond with the welcome message.

**Testing the start node**

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

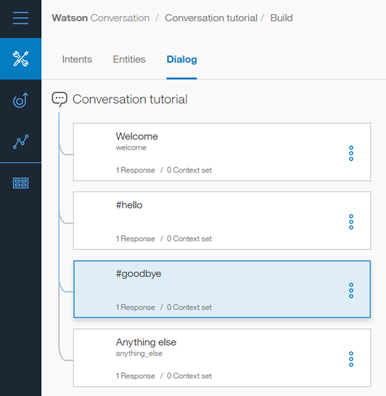
* Click the Ask Watson icon to open the "Try it out" pane. You should see your welcome message.



**Adding nodes to handle intents**

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

1. Click the More icon More options on the **Welcome** node, and then select **Add node below**.
2. Type #hello in the **Enter a condition** field of this node. Then select the **#hello** option.
3. Add the response, Good day to you.
4. Click Close to close the edit view.
5. Click the More icon More options on this node, and then select **Add node below** to create a peer node. In the peer node, specify #goodbye as the condition, and OK. See you later! as the response.



**Testing intent recognition**

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

1. Click the Ask Watson 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:
   * bye
   * howdy
   * see ya
   * good morning
   * sayonara

Watson can recognize your intents even when your input doesn't exactly match the examples 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.

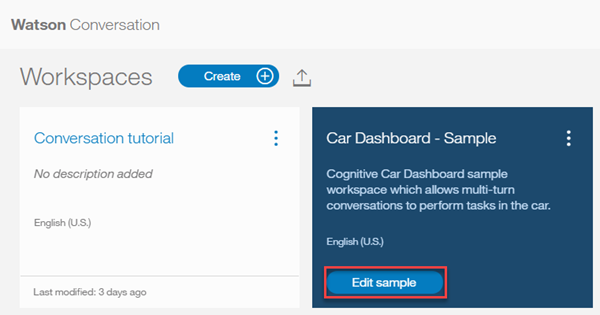
**Result of building a dialog**

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

**Step 5: Review the sample workspace**

Open the sample workspace to see intents similar to the ones you just created plus many more, and see how they are used in a complex dialog.

1. Go back to the Workspaces page. You can click the Back to workspaces button button from the navigation menu.
2. On the **Car Dashboard - Sample** workspace tile, click the **Edit sample** button.



**Advanced Examples**

**Step 6: Add intents and examples**

Add an intent on the Intents tab. An intent is the purpose or goal expressed in user input.

1. On the Intents page of the Conversation tutorial workspace, click **Create new**.
2. Add the following intent name, and then press Enter:

turn\_on

A # is prepended to the intent name you specify. The #turn\_on intent indicates that the user wants to turn on an appliance such as the radio, windshield wipers, or headlights.

1. In the **User example** field, type the following utterance, and then press Enter:

I need lights

1. Add these 5 more examples to help Watson recognize the #turn\_on intent.

Play some tunes

Turn on the radio

turn on

Air on please

Crank up the AC

Turn on the headlights

1. Click **Done** to add the intent.

You now have three intents, #turn\_on, #hello, and #goodbye, all with example utterances. These examples help train Watson to recognize the intents in user input.

**Step 7: Add entities**

An entity definition includes a set of entity *values* that can be used to trigger different responses. Each entity value can have multiple *synonyms*, which define different ways that the same value might be specified in user input.

Create entities that might occur in user input that has the #turn\_on intent to represent what the user wants to turn on.

1. Click the **Entities** tab to open the Entities page.
2. Click **Create new**.
3. Add the following entity name, and then press Enter:

appliance

A @ is prepended to the entity name you specify. The @appliance entity represents an appliance in the car that a user might want to turn on.

1. Click the toggle to turn fuzzy matching **On**. This setting helps the service recognize references to entities in user input even when the entity is specified in a way that does not exactly match the syntax you use here.
2. Add the following value to the **Value** field, but do not press Enter:

radio

The value represents a specific appliance that users might want to turn on.

1. Add other ways to specify the radio appliance entity in the **Synonyms** field. Press tab to give the the field focus, and then enter the following synonyms. Press Enter after each synonym.

music

tunes

1. Click the **Add a new value** icon plus sign to add other types of appliances.
   * Value: headlights. Synonym: lights.
   * Value: air conditioning. Synonyms: air and AC.
2. Click **Done** to add the **@appliance** entity.
3. Repeat Steps 2-8 to create the @genre entity with fuzzy matching on, and these values and synonyms:
   * Value: classical. Synonym: symphonic.
   * Value: rhythm and blues Synonym: r&b.
   * Value: rock. Synonym: rock & roll, rock and roll, and pop.

You defined two entities: @appliance (representing an appliance the bot can turn on) and @genre (representing a genre of music the user can choose to listen to).

When the user's input is received, the Conversation service identifies both the intents and entities. You can now define a dialog that uses intents and entities to choose the correct response.

**Step 8: Create a complex dialog**

In this complex dialog, you will create dialog branches that handle the #turn\_on intent you defined earlier.

**Add a base node for #turn\_on**

Create a dialog branch to respond to the #turn\_on intent. Start by creating the base node:

1. Click the More icon More options on the **#hello** node, and then select **Add node below**.
2. Start typing #turn\_on in the condition field, and then select it from the list. This condition is triggered by any input that matches the #turn\_on intent.
3. Do not enter a response in this node. Click Close to close the node edit view.

**Scenarios**

The dialog needs to determine which appliance the user wants to turn on. To handle this, create multiple responses based on additional conditions.

**Scenario**: The user wants to turn on the music, in which case the bot must ask for the genre.

Add nodes that check these scenario conditions in this order so the dialog evaluates the most specific condition first.

**Address Scenario 1**

Add nodes that address scenario 1, which is that the user wants to turn on the music. In response, the bot must ask for the music genre.

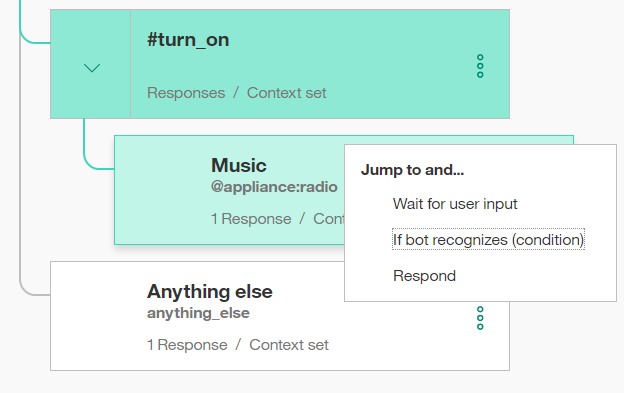
**Add a child node that checks whether the appliance type is music**

1. Click the More icon More options on the **#turn\_on** node, and select **Add child node**.
2. In the condition field, enter @appliance:radio. This condition is true if the value of the @appliance entity is radioor one of its synonyms, as defined on the Entities tab.
3. In the response field, enter What kind of music would you like to hear?
4. Name the node Music.
5. Click Close to close the node edit view.

**Add a jump from the #turn\_on node to the Music node**

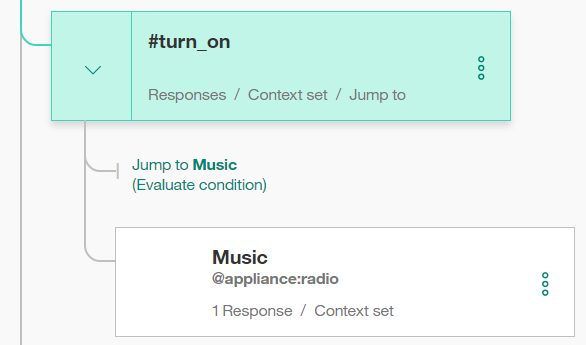
Jump directly from the #turn on node to the Music node without asking for any more user input. To do this, you can use a **Jump to** action.

1. Click the More icon More options on the **#turn\_on** node, and select **Jump to**.
2. Select the **Music** child node, and then select **If bot recognizes (condition)** to indicate that you want to process the condition of the Music node.



Note that you had to create the target node (the node to which you want to jump) before you added the **Jump to** action.

After you create the Jump to relationship, you see a new entry in the tree:



**Add a child node that checks the music genre**

Now add a node to process the type of music that the user requests.

1. Click the More icon More options on the **Music** node, and select **Add child node**. This child node is evaluated only after the user has responded to the question about the type of music they want to hear. Because we need a user input before this node, there is no need to use a **Jump to** action.
2. Add @genre to the condition field. This condition is true whenever a valid value for the @genre entity is detected.
3. Enter OK! Playing @genre. as the response. This response reiterates the genre value that the user provides.

**Add a node that handles unrecognized genre types in user responses**

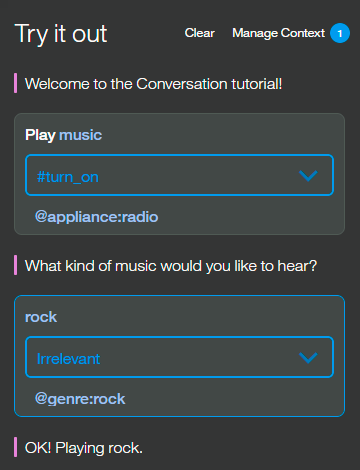
Add a node to respond when the user does not specify a recognized value for @genre.

1. Click the More icon More options on the *@genre* node, and select **Add node below** to create a peer node.
2. Enter true in the condition field. The true condition is a special condition. It specifies that if the dialog flow reaches this node, it should always evaluate as true. (If the user specifies a valid @genre value, this node will never be reached.)
3. Enter I'm sorry, I don't understand. I can play classical, rhythm and blues, or rock music. as the response.

That takes care of all the cases where the user asks to turn on the music.

**Test the dialog for music**

1. Select the Ask Watson icon to open the chat pane.
2. Type Play music. The bot recognizes the #turn\_on intent and the @appliance:music entity, and it responds by asking for a musical genre.
3. Type a valid @genre value (for example, rock). The bot recognizes the @genre entity and responds appropriately.



1. Type Play music again, but this time specify an invalid response for the genre. The bot responds that it does not understand.