Microsoft Copilot Studio

Lab 02: Authoring 101

Hands-on lab step-by-step

January 2025

UDPP Copilot Studio Workshop

Contents

Microsoft Copilot Studio

This lab is subject to the Terms of Use found at the end of this document.

Goals for this lab

This lab covers the following topics: • Learn the fundamentals of the message and question nodes. • Discover the fundamentals of entities and slot filling. • Learn how to use variables in Microsoft Copilot Studio. • Explore the extensibility capability, including Power Fx and extended node properties.	The time to complete this lab is [60] minutes.

Prerequisites

Labs have been designed to be completed with only a Microsoft Copilot Studio trial. You can start most labs without having to complete the previous module but note that some exercises may reference previous labs. To fully experience the features and functionality of the product, it is recommended that you make sure to have completed all pre-requisites below before starting this lab.

For this lab you need:

- A computer with internet access.
- Be able to log into the provided Microsoft tenant (some companies enforce users to only connect to their company tenant) or your own enterprise tenant with a Copilot Studio User License (or trial)
- **Generative AI should be set to "classic" (in Settings, Generative AI)**
- **Complete Lab 01**
- **Access to external website (learn.microsoft.com)**

Entities in Microsoft Copilot Studio

Microsoft Copilot Studio uses natural language understanding (NLU) to interpret what a user is saying and to try to match a user's utterance with an existing topic. For example, if the user says, "*I tried to use my gift card, but it doesn't work*", the agent knows to route the user to the **topic** that's related to gift cards not working, even if that exact phrase isn't listed as a trigger phrase. This concept can also be referred to as **intent recognition**.

NLU can also help the agent identify **entities** in a user's input. An entity represents a key information you're trying to identify and extract from a sentence. This can be a phone number, a zip code, a city, a case ID, a person's name, etc. You can also define your own entities. Your agent can recognize the relevant information from a user input and then save it for later use.

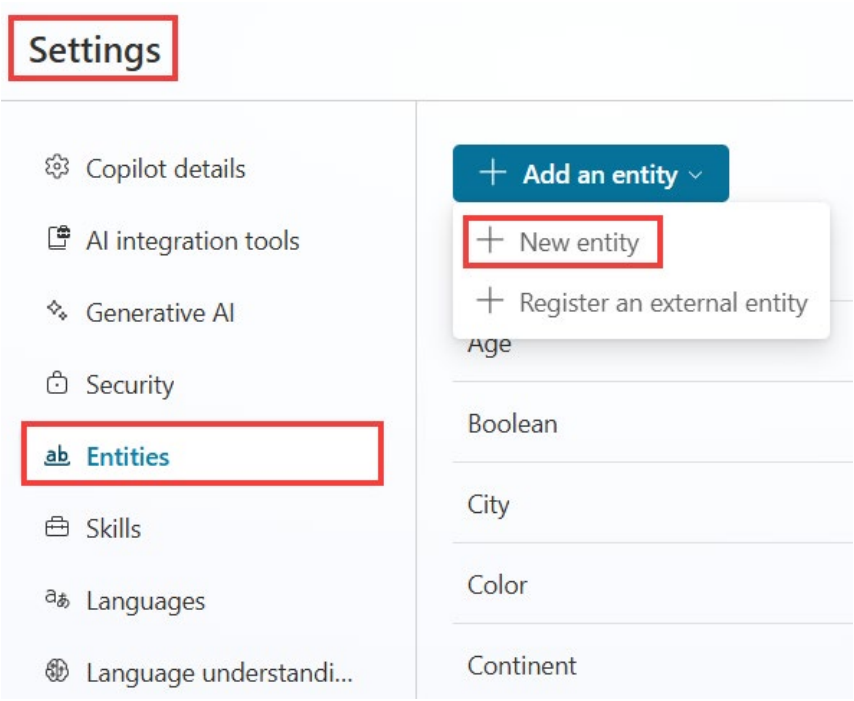
For more information regarding entities, please refer to the documentation: [Use entities and slot filling in copilots - Microsoft Copilot Studio | Microsoft Learn]

The following tasks show you how to create a custom entity and use it within the topic that you already created.

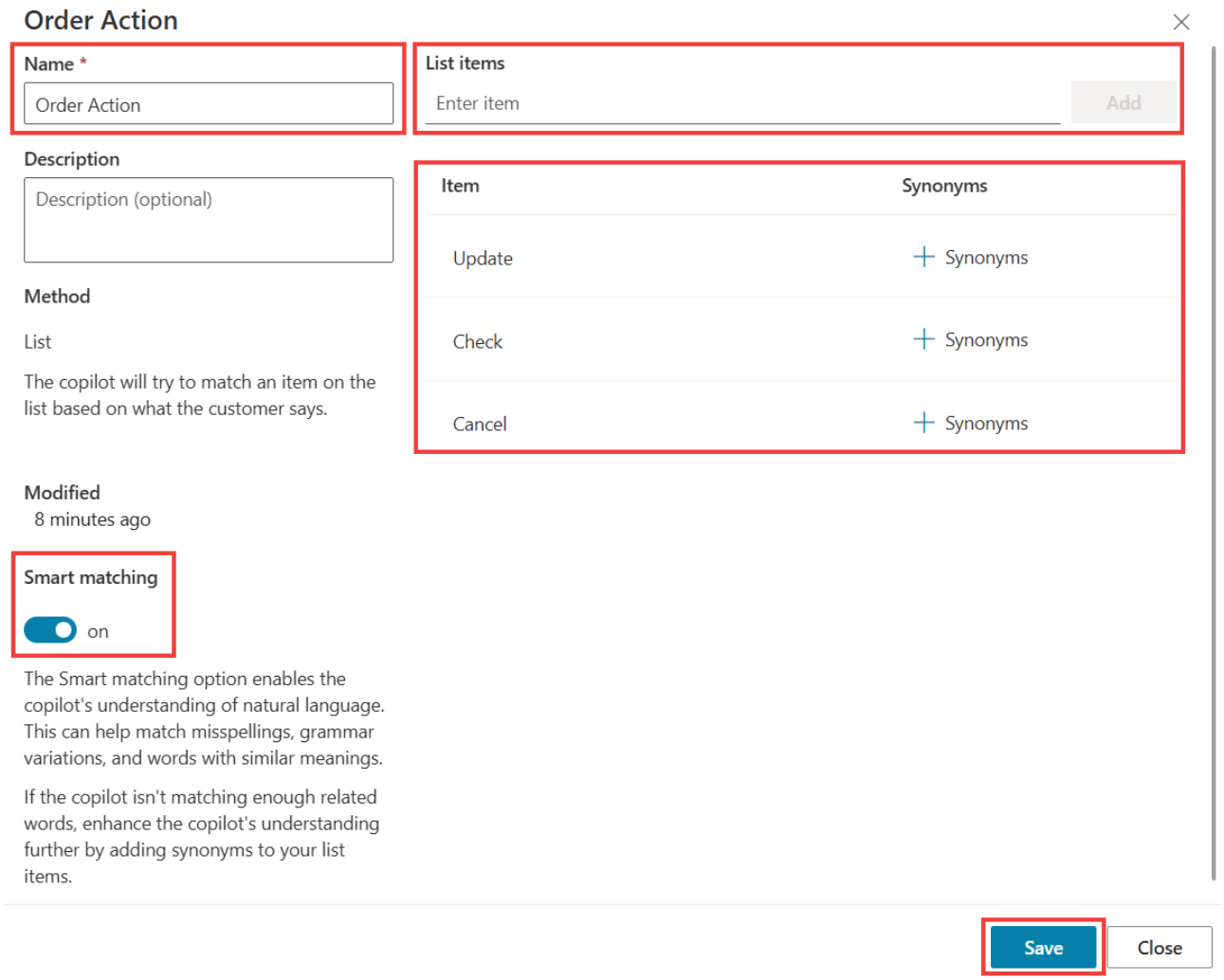
Task 1: Use entities and slot filling

In this task, you'll learn how to use entities and slot filling.

1. With your agent open in Microsoft Copilot Studio, go to **Settings** in top-right corner, and select **Entities**. Then select **Add an entity** and **New entity**:



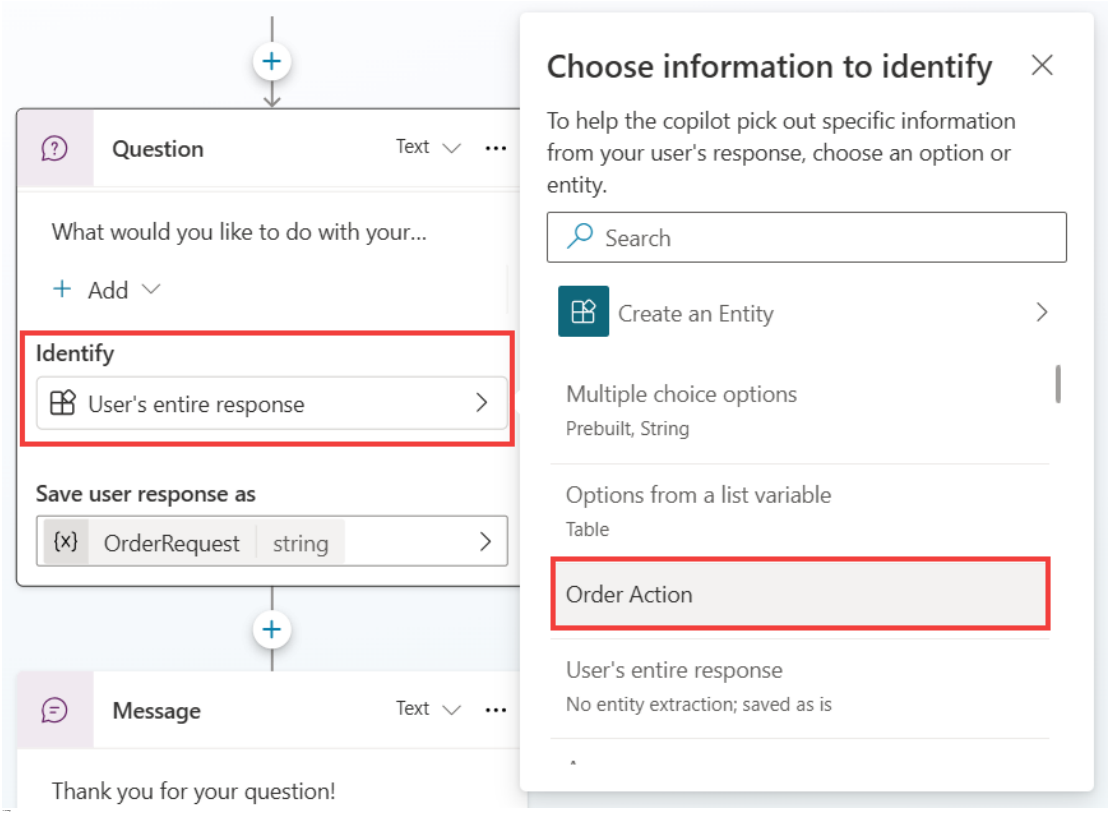
2. Within the **Create an entity** dialog, select **Closed List**.
3. Within the **Name** field of the Order Action pane, enter the name `Order Action`.
4. Add three options within the **List items** called `Update`, `Check`, and `Cancel`. You can also choose to add synonyms by selecting synonyms for each option (*optional for this task*).
5. Make sure **Smart matching** is toggled on, and then select **Save**.



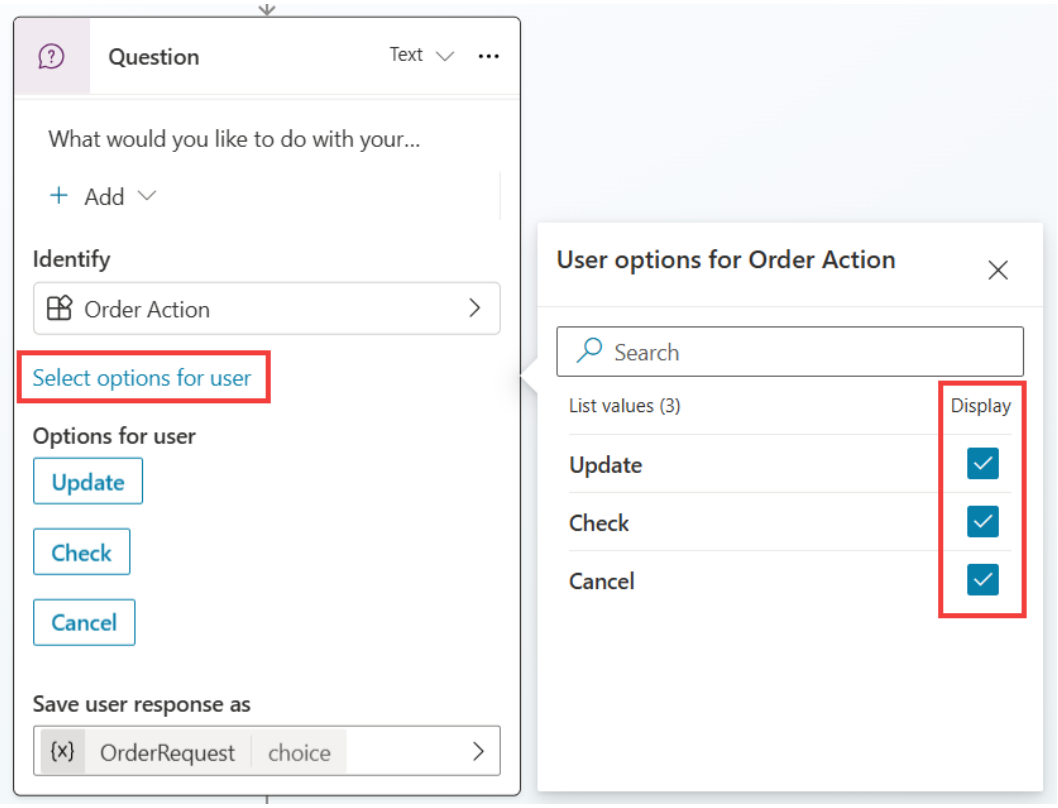
This action creates a new entity called **Order Action** that you can use with the **Question** node in your topic to place **User's entire response** with **Order Action**.

6. Return to the topic that you created in previous lab, called **Check Order Status**, and select the **Question** node that you created to originally identify the user's full response. Select **Identify**, and a slide-out menu will display on the right, where you can select an entity from the list.

   Search for and select the custom entity that you created in the previous step called Order Action.



7. **Select options for user** and then select all the options to display to the user.



8. **Save** your topic

You have successfully set up a custom entity for your **Question** node. The default behavior for question nodes is, if the variable that the question response is stored in has a value already, then the question is skipped and not asked.

Task 2: Testing slot filling

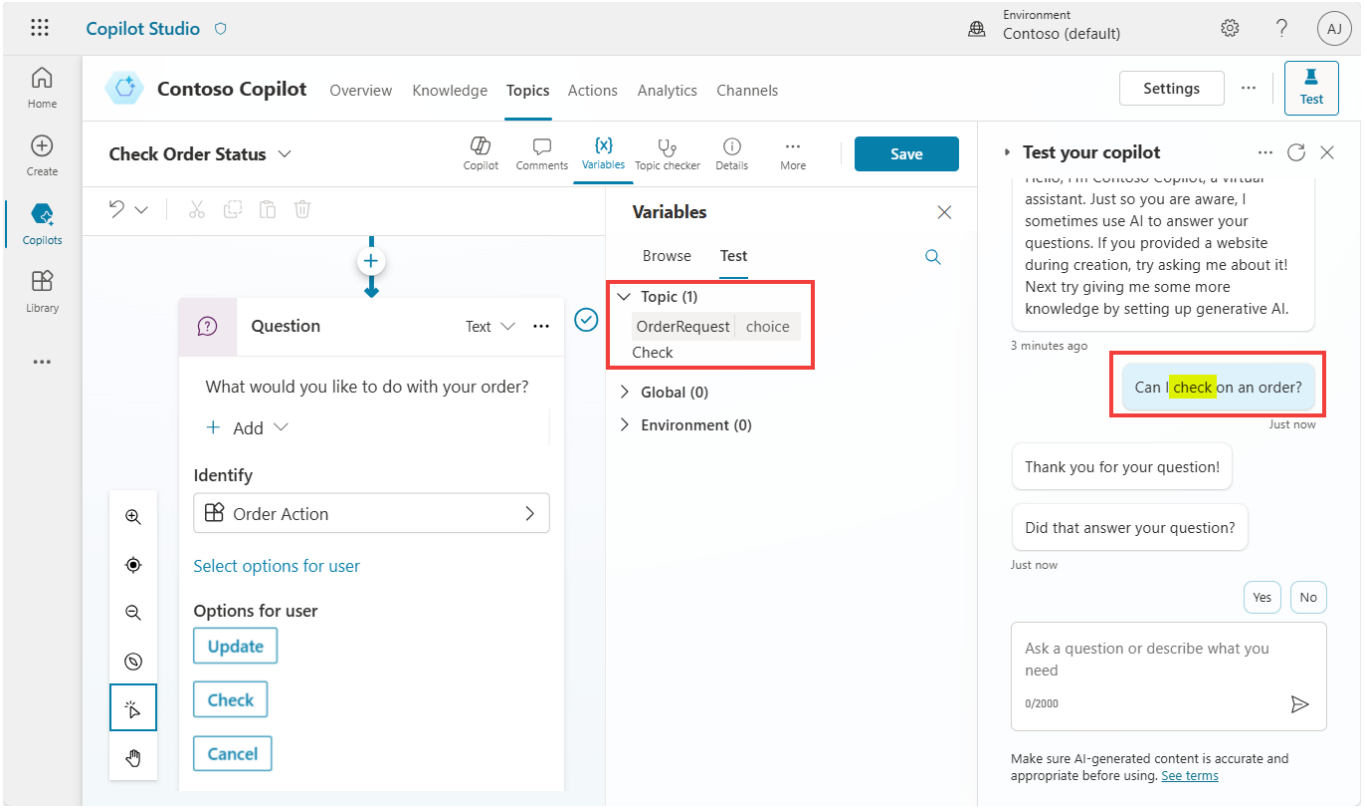
1. Display the **Test** pane
2. Use the **refresh** button to start a new conversation.
3. Try how entities and slot filling work by entering a sentence matching one of your trigger phrases.

```
Can I check on an order?
```

4. Go the to **Check Order Status** topic, and display the **Variables** pane, and in the **Test** tab, expand the **Topic** variables.

You will observe the process working because the user has triggered this topic with the intent to "check" an order, and the entity has been slot filled into the variable from the follow-up question after the trigger phrase. **As a result,**

the question isn't asked and is skipped. This is because you've used entities and slot filling to retrieve the information from utterance the user submitted. This approach avoids you needing to ask the user a question that they've already provided information for.



Variables

You're beginning to enhance the topic that you created in the previous lab. In the previous section, you used entities and slot filling to automatically detect the data from a user's sentence and store specific data in a variable. Now, you'll learn how to use the data that you obtained from the question in a variable and then display it within a message.

Task 1: Learn about variable types

Variables let you save responses from your end-users to help guide the conversation (such as determining whether to provide different instructions for returns based on the purchase price of the item). And you can use them directly in the conversational response from the agent (for example, "*I can help you return your {Topic.ProductName}*").

Different types of variables in Microsoft Copilot Studio include:

- **System** – These variables are normally populated with system data. System variables aren't user-made and are part of the platform.

- **Topic** – These variables are user-made from either topic Inputs, the Set a Variable Value node, the Question node or as the output of other nodes or actions (e.g., cloud flows, HTTP requests, connectors, custom prompts, plugin actions, etc.).

- **Global** – These variables are user-made and are available from any topic, and they're a good way to store data that multiple topics use to help the conversation, regardless of how many topics are triggered within it.

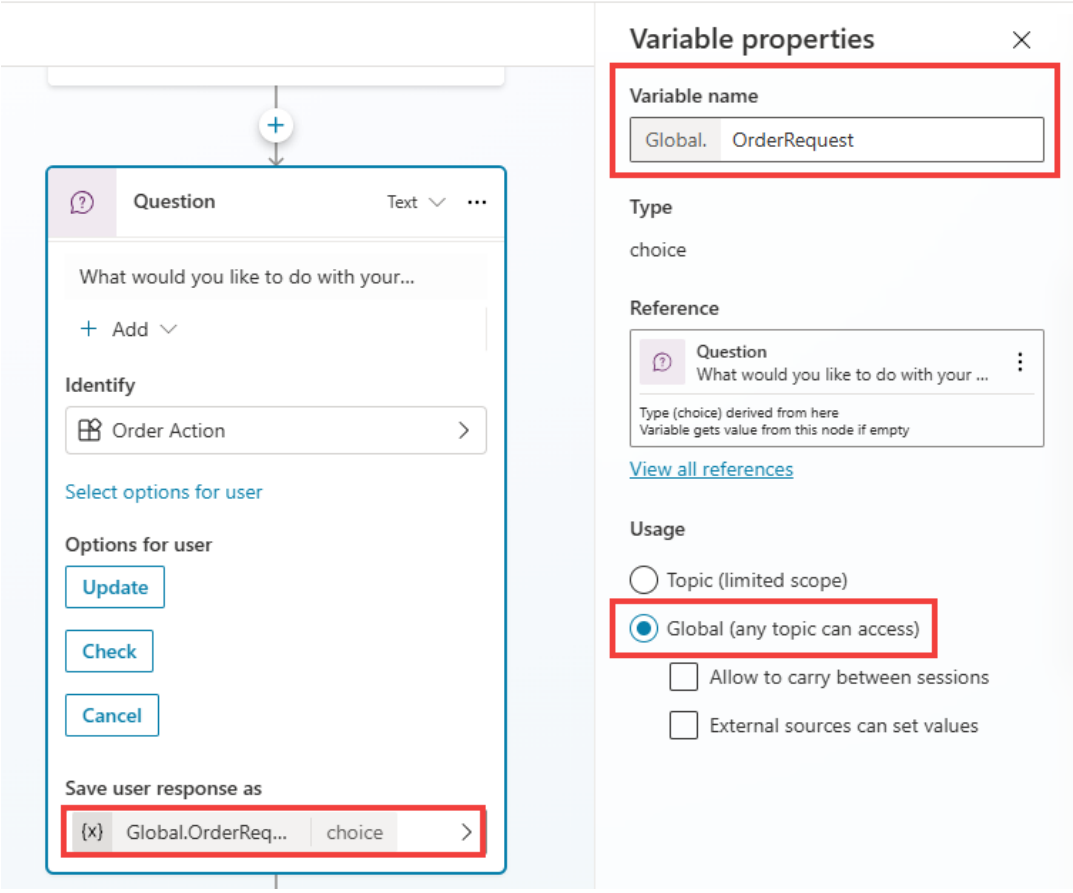
You can use variables in several places, including the Questions, Conditions, and Set Variable Value nodes. The variable can be a custom value that uses Power Fx, a user-entered value, a response from a question, or system variable values.

For more information regarding variables, please refer to the documentation: Work with variables - Microsoft Copilot Studio | Microsoft Learn

Task 2: Use global variables

In this exercise, you learn how to use the data from the previous task in the first exercise, **Check Order Status**. At this point, you should have the **Question** node in your topic linked to an entity.

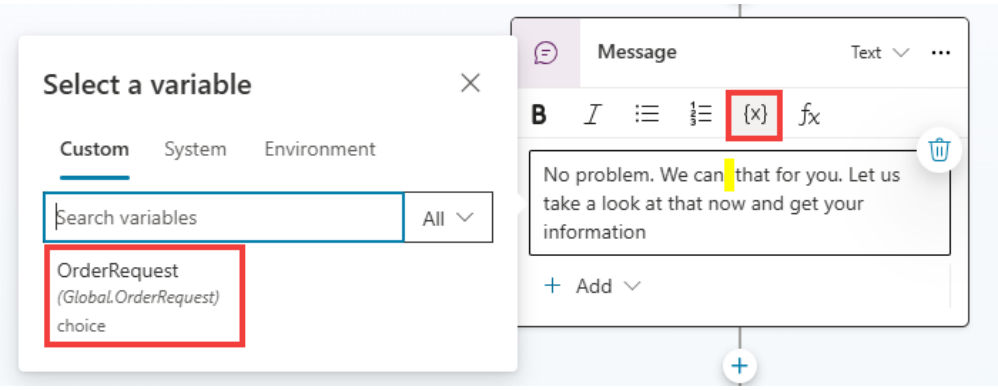
1. In the right panel that opens – *if not already done in the previous lab* – rename the **Variable name** to `OrderRequest` and then select **Global** to change the scope from local (topic) to **Global** so that other topics can access it, as shown in the following screenshot.



2. Now, you use the variable that you completed from the question or trigger phrase in the message as dynamic data. Select the existing `Thank you for your question!` **Message** node and change the text to: `No problem. We can  that for you. Let us take a look at that now and get your information`

3. Place the cursor between the 2 spaces, select the **{x}** variable icon, and then select the **Global.OrderRequest** variable that you recently created. This action is common to insert a variable in place of the literal word, making it dynamic based on data provided from the user or customer's question.

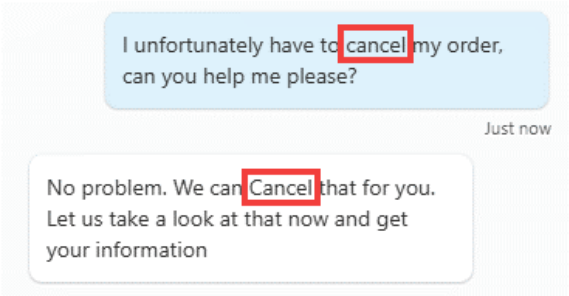
```
No problem. We can {Global.OrderRequest} that for you. Let us take a
look at that now and get your information
```



4. **Save and refresh the test chat**

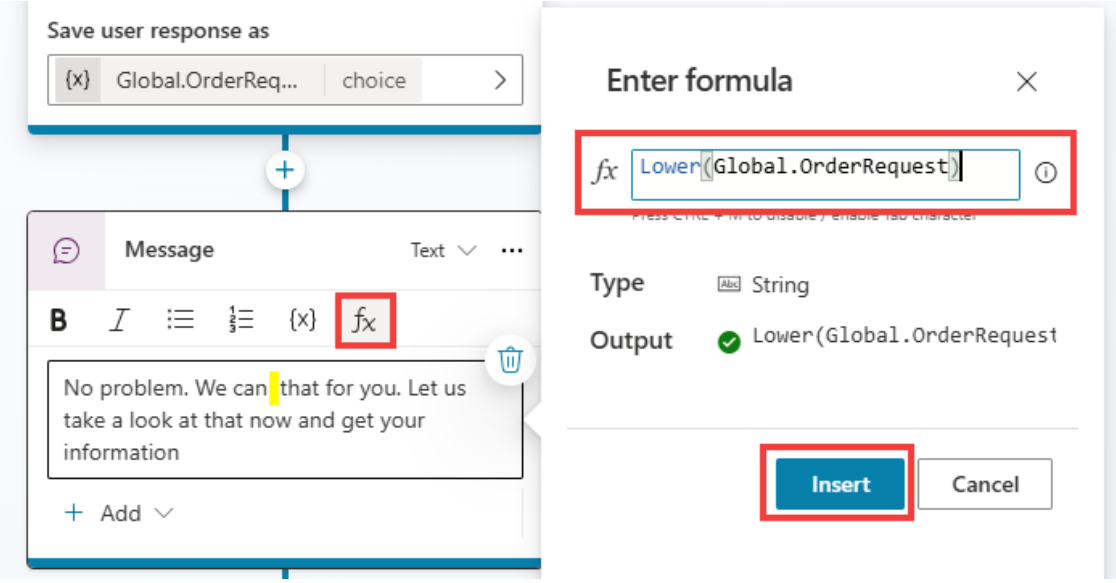
5. Select the **Test** option to test the behavior of the Copilot and the changes that you made by triggering the topic with a trigger phrase. The following screenshot shows this process in action.

```
I unfortunately have to cancel my order, can you help me please?
```



6. Notice how the **OrderRequest** value still has its first letter capitalized. To address this grammatical issue, you can choose to use a formula to lower the word, instead of directly referencing the variable value. Go back to the Message node, remove the initial variable value {Global.OrderRequest}, and instead, select the ***fx*** button, and use the Lower() Power Fx formula. See how the variable values can be referenced within the formula.

```
Lower(Global.OrderRequest)
```



Pro tip: Within the **Variable management** options is the **Clear all variables** option, which clears all variable values. This option is useful if you want to begin or loop back into the same topic but take new values, especially if you've set up question behavior properties where a question could be skipped if it already had a value.

Variables are the best way to store dynamic data or data that you want to perform conditions or checks on to drive conversational behavior in a particular way, as you'll observe in the next task.

Congratulations on completing this task. You've now reviewed variables and the different types in Microsoft Copilot Studio.

Use variables in conditions

Your topic has currently been updated from the second lab with entities and slot filling capabilities, and it's using dynamic data to store variables and reuse those variables in messages to provide a dynamic authoring experience. Now, you'll use the same variable in a condition statement in Microsoft Copilot Studio.

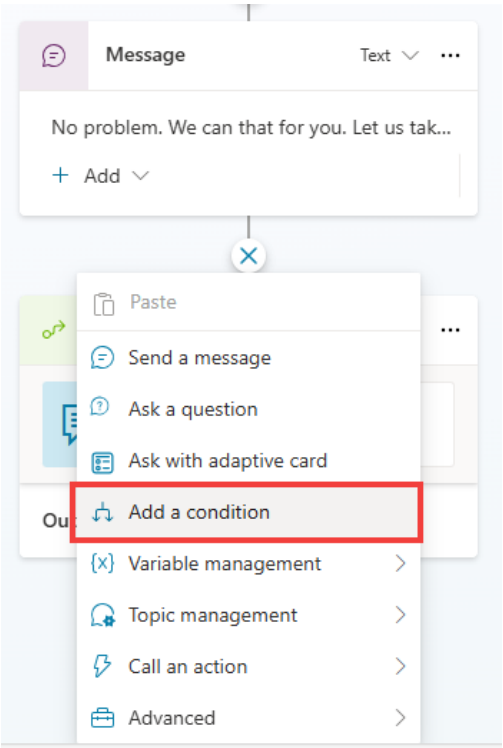
With condition statements in Microsoft Copilot Studio, an agent author can determine behavior under certain conditions that can be true, false, or something else (for example, if it's blank). Condition statements allow and promote flexibility in the authoring canvas, allowing you to provide great customer or user experiences based on their needs while limiting the need to create several similar topics. After you begin to use conditions, you'll create *branches*, which create separate flows that the person who's using the Copilot can be directed to. These branches can have their own conditions, depending on what behavior you want to create.

For more information on conditions, see Authoring using conditions.

Task 1: Create a condition by using variables

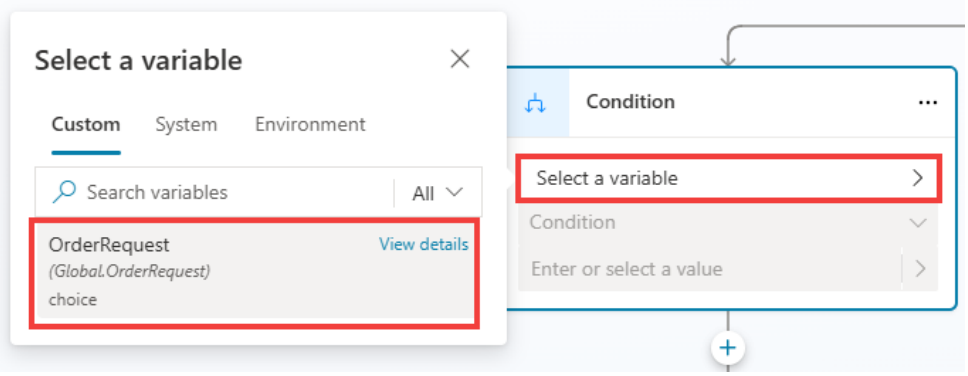
In this task, you'll create a condition based on the three variable options that were used in the custom entity in the first exercise.

1. In your authoring canvas, under the **Message** node that you modified in the previous task, select the plus (**+**) icon to add a new node and then select **Add a condition**.

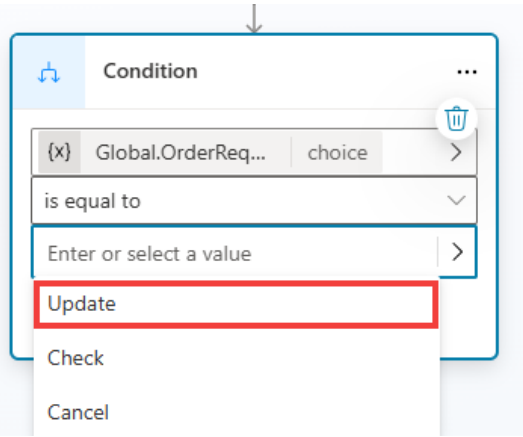


2. Two new nodes will appear, one is your **Condition** and the other is an exception for **All other conditions**.

   In your **Condition** node, select the **Select a variable** option and then select your **OrderRequest** global variable.

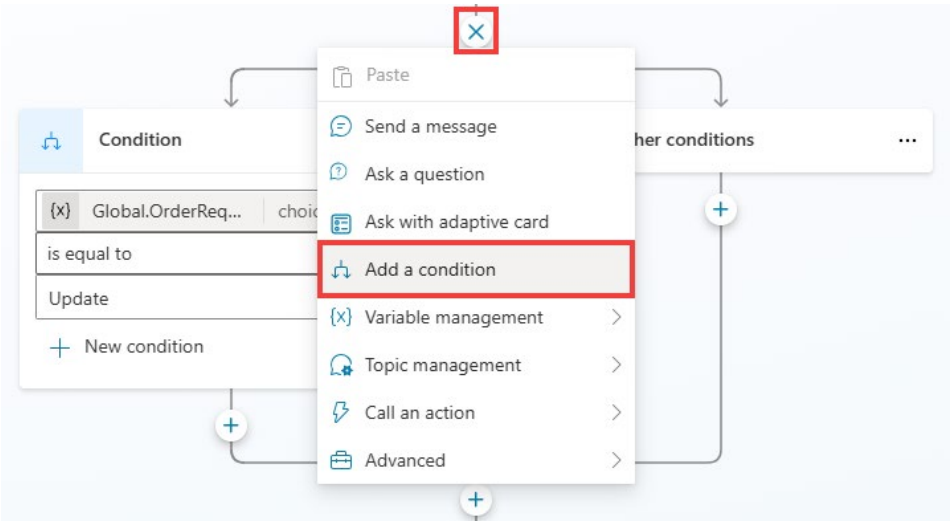


3. Keep the condition operator as **is equal to** and then select the empty box beneath to display the three available options from the selected variable. Select the **Update** option – make sure you click on the *Enter or select a value* part, and not on the chevron icon (>).

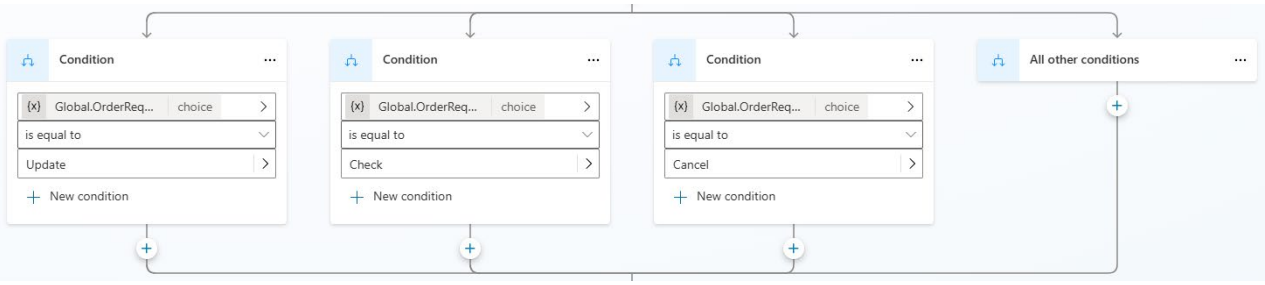


4. A completed condition should now show: if the **OrderRequest** is equal to **Update**.

5. Create two more conditions in this branch for the two other options for the **OrderRequest** variable (check and cancel). Select the plus **(+)** icon to add a node above the condition and then select **Add a condition** to add another conditional branch.

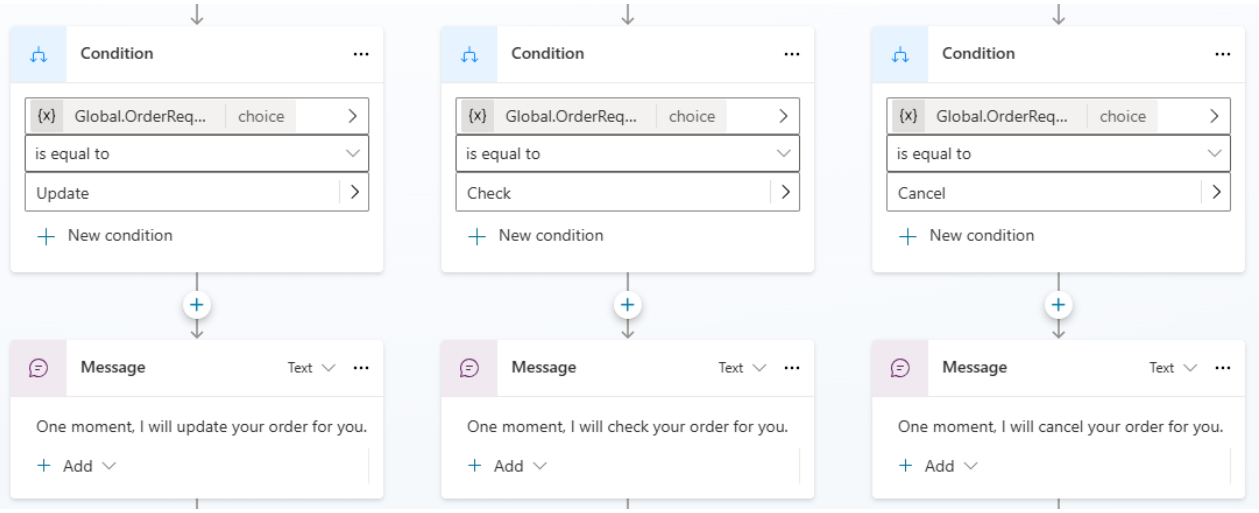


6. Repeat the previous steps by selecting your **Global.OrderRequest** variable and then select the **Check** and **Cancel** options in two other conditions so that you'll have a conditional branch with three options (including **All Other Conditions**), as shown in the following screenshot.



7. Under each condition node, add a **Message** node that will display different text depending on the condition, as shown in the following example.

```
One moment, I will update your order for you.
```



Pro tip: do things faster by selecting a node and **copying** it with the top-left **productivity tools** menu. Once copied, the node gets available to be pasted, using the same **productivity tools** menu or when using the **(+)** icon to add a new node.

8. **Save** your topic and then select the **Test** option to explore the different trigger phrases and conditions that lead the user to view different message outcomes.

Conditions are foundational tools that help you create tailored experiences based on what the user has selected or answered in previous questions. You can nest conditions within other conditions for more complex logic.

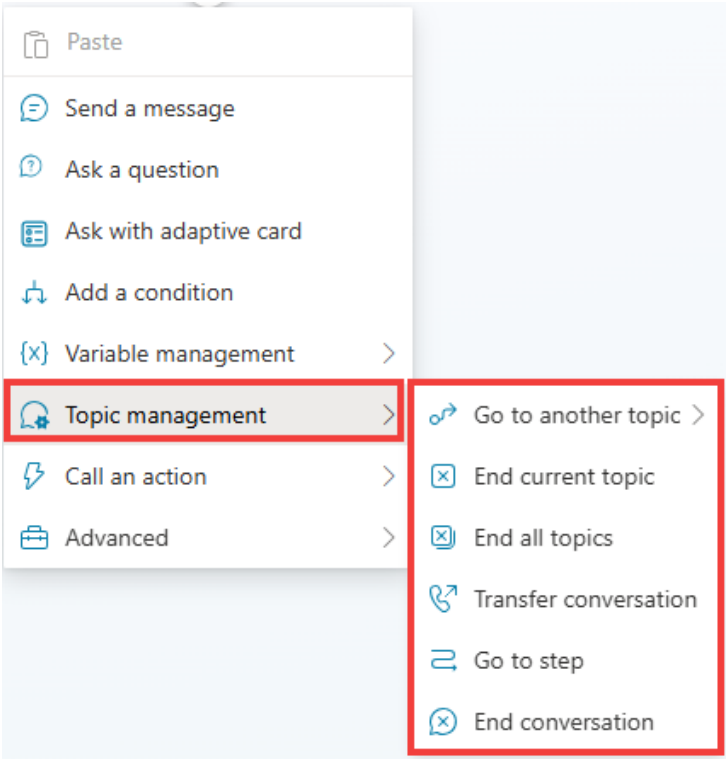
Congratulations, you've now completed the basics of using conditions and using variables as parameters within them.

Use topic nodes

Microsoft Copilot Studio allows authors to redirect to another topic, such as an escalation topic. Typically, topics are set up to trigger based on what users are typing or saying to the Copilot that triggers an intent match, or based on events such as an escalation, conversation start, and other system topics with event-driven triggers. Users can also direct other topics within an authored conversation where it makes sense to do so, or they can perform loops within the current topic to another node.

Task 1: Become familiar with topic type nodes

In the topic that you've been working on during previous exercises for this lab, select the plus **(+)** icon to add a new node and then select **Topic management**. A list of available commands displays, as shown in the following screenshot.



In the **Topic management** menu, you can select from the following options:

- **Go to another topic** – This node has an extended flyout menu where you can go to another topic that you need to select.

Pro tips:

- It's often more manageable to create many **bite-size topics** rather than a few large topics. Taking this approach also helps making triggering more effective, by clearly mapping trigger phrases to the specific topics that address those areas.
- As large topics can be challenging to maintain and update, it's a good idea to break down your agent logic when possible, especially if parts of your bot conversation logic can be shared by multiple topics. This concept is referred to as **reusable topics**.
- **Topics don't need to all have trigger phrases**, as topics can redirect to other topics and pass variable information back and forth.

- **End current topic** – Selecting this option ends the current topic. Typically, you'd use this option where the topic was called from another topic. It would be returned to where it was originally called from. You can also use this option in branching conditions. If you select this option to end an entire topic of a branch, the behavior operates similar to the **End all topics** node.

- **End all topics** – This node ends all active topics. The next message from the user is considered as a new conversation, and triggers the most appropriate topic based on that user message, starting a new active topic.

- **Transfer conversation** – Select this option to transfer to an agent and send a contextual message.

- **Go to step** – Allows an agent author to navigate to another node in the open topic. This option is useful for looping scenarios or if you want to gather more information from the user.

- **End conversation** – Sends an **endOfConverastion** activity. This can be useful for the client chat widget, for examples deployed to your website), to know that the chat session is over.

Now that you're familiar with the basics of the topic management functions, you can practice using the **Go to another topic** node for the next task. The **Go to another topic** node is useful when you want to apply other topics from conditions based on what the user asks for in the dialog.

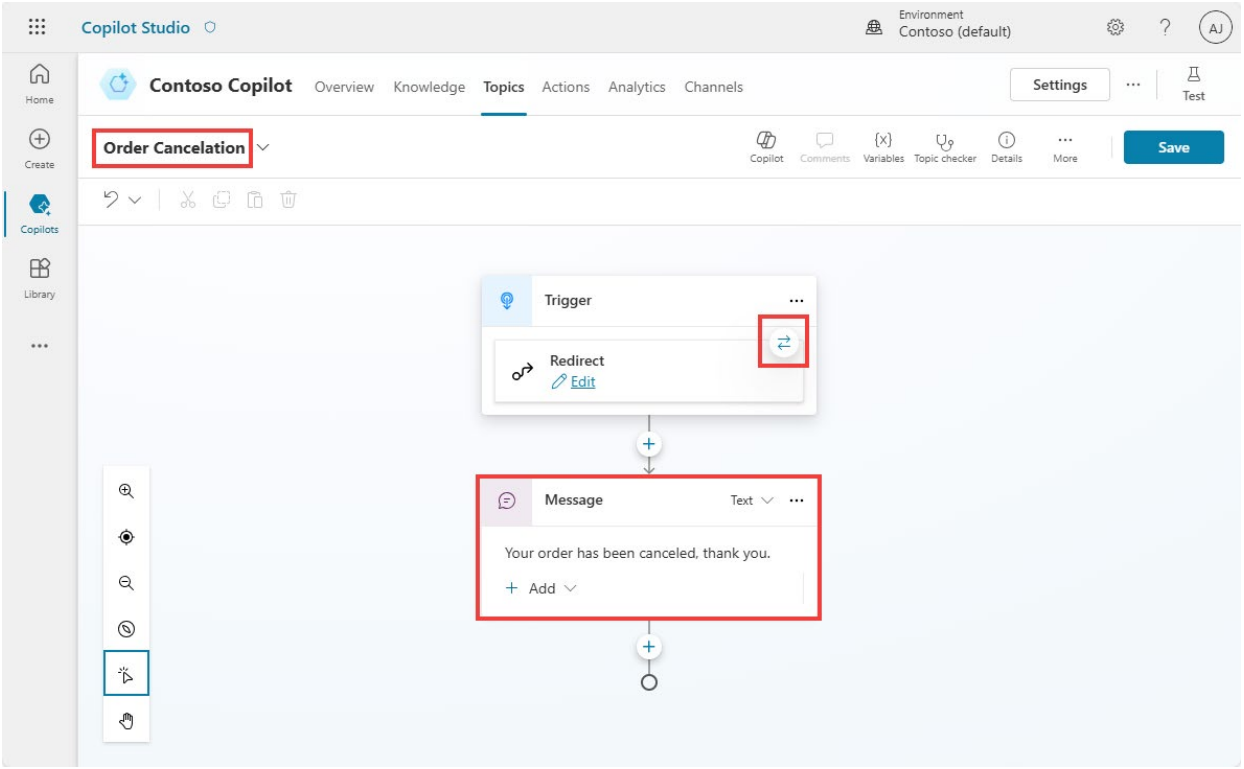
Task 2: Use the Go to another topic node

In this task, you learn how to use the **Go to another topic** node.

1. In your topic, ensure that you completed the last task and that you have a **Condition** node in your authoring canvas.

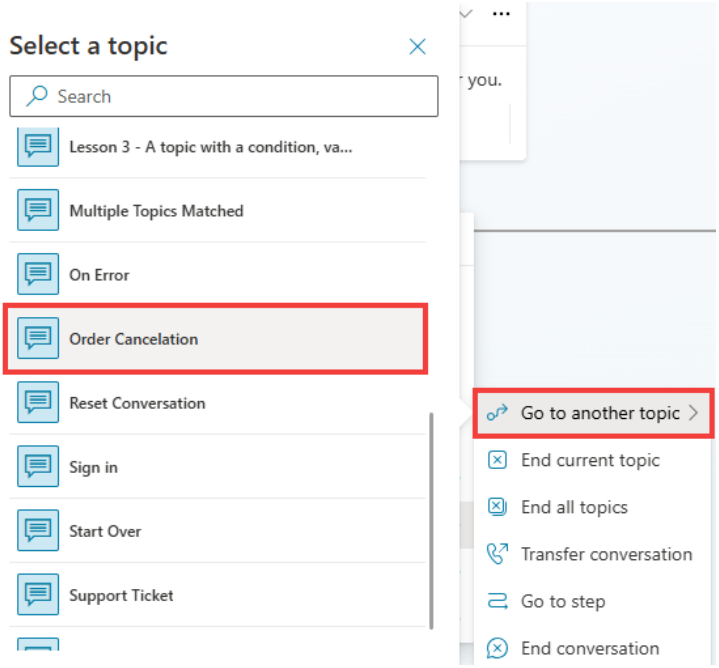
2. Return to the **Topics** page where all your topics are displayed in a list view. Create a new blank topic called `Order Cancellation`. As this topic doesn't need trigger phrases, change the trigger type to **Redirect**. Add a single **Message** node that acknowledges the cancellation and then save this topic.

```
Your order has been canceled, thank you.
```

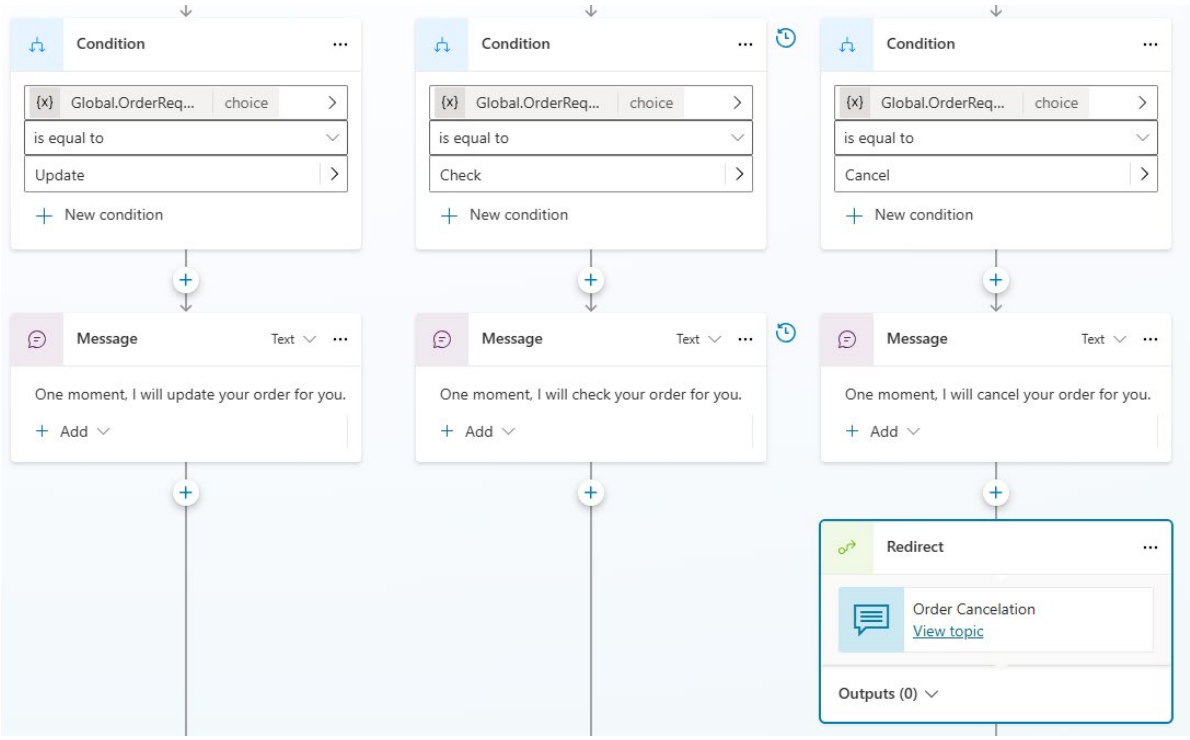


3. **Save** this topic

4. Return to the original topic that you created called **Check Order Status** that has the **Condition** branch within it. Within the branch that had the condition set as **Cancel**, create a new node beneath the **message** node and then select **Topic management**. Then, in the flyout menu, select **Go to another topic** and then find and select your **Order Cancellation** topic in the list (you can search for a topic by name or scroll through the list of topics to find a given topic), as shown in the following screenshot.



5. The three conditions from the previous task should display. The **Cancel** conditional branch should contain a redirect to another topic, which sends an acknowledgment message by redirecting to the new topic that you created. Test the behavior by **Saving** your topic and then selecting the **Test** option.

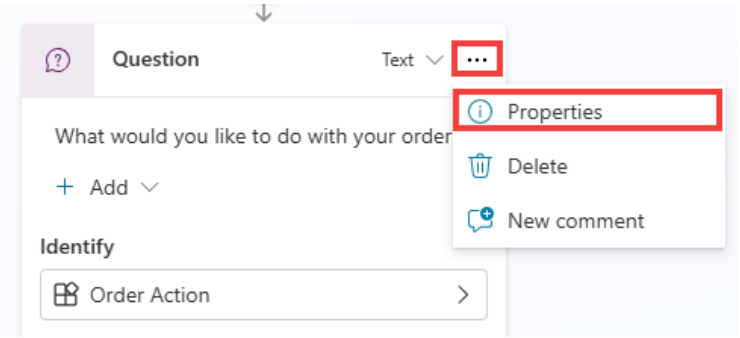


Congratulations, you're now familiar with the available actions under the **Topic management** menu. It would be useful for you to review the other options under **Topic management** before you continue; however, it's not essential for moving on to the next exercise.

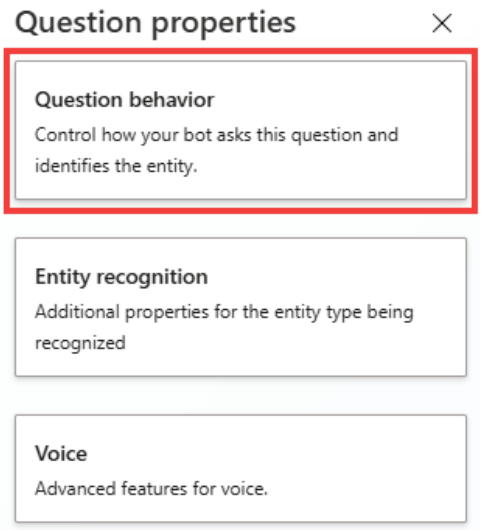
Question node behavior

Previously, this lab covered the basics of the **Question** node and built on this concept by using entities and slot filling. In addition to storing a user's response, the **Question** node has more behavior options that you can set up. One option is the ability to skip the asking of a question if the variable that it's linked to already contains a value. You observed this process in action in a previous exercise where the question was skipped when the agent was asked to check an order. The question was skipped because, by using entities and slot filling, you allowed Microsoft Copilot Studio to retrieve data from the sentence that the user asked and then store the data within the variable. After the **Question** node was reached by Microsoft Copilot Studio, it already contained data, so the question didn't need to be asked again. This approach is more efficient because, when the user or customer is talking to a copilot, they won't need to answer the same question multiple times.

1. Within the **Check Order Status** topic, select the **Question** node. Then, select the ellipsis within the top right corner of the **Question** node to extend the menu, as shown in the following screenshot, and then choose **Properties** from the menu.



2. Select **Question behavior** from the **Question properties** panel that appears.



The **Question** node has several configurable options so that you're able to better identify what the user's response is to the question that you're asking. This component is important when you're developing conversational applications. Because regardless of the type of AI that's behind the scenes managing the natural language responses, a user might provide unexpected or unidentifiable answers. The ability to handle the copilot's behavior in those circumstances help you provide an improved customer experience. This scenario also happens in real life, when you ask a question to another person, and they don't understand the question. For the best experience and conversation, it's important to rephrase or act differently than to repeat the same question that wasn't originally understood.

The following question behavior controls are available for you to choose from in the **Question behavior** property window:

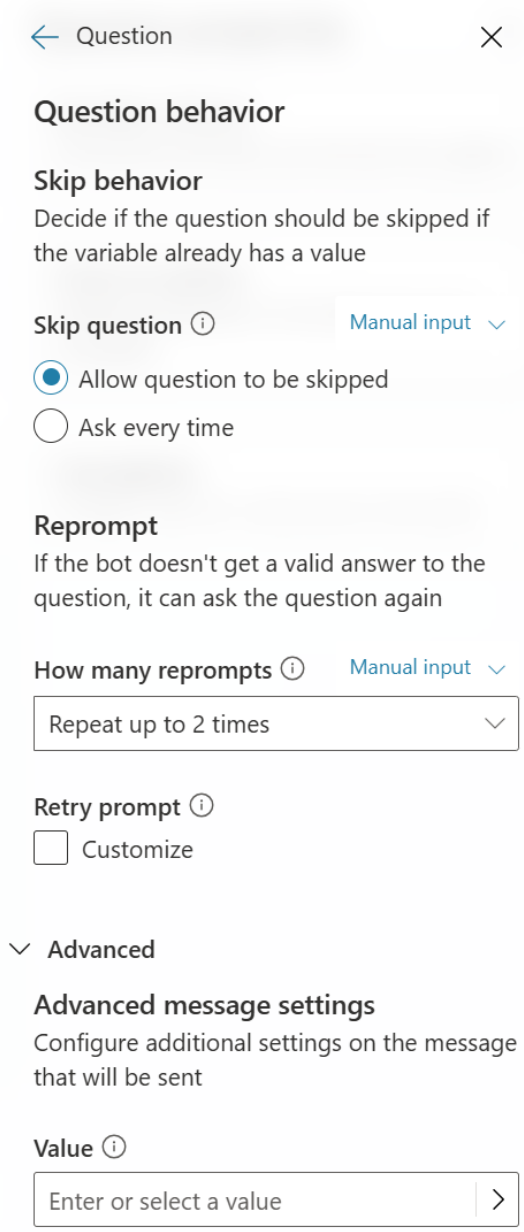
- **Skip behavior/Skip question** – A agent author can skip the question if the variable already has a value. The variable in the question could have a value that was set somewhere else in the topic, in another topic, or through slot filling and by using entities. This behavior allows the agent author to skip the question, or if the variable has a value, to ask the question anyway. Other available options include using Power Fx to create a condition, and if that condition is true, to skip the question.

- **Reprompt/How many reprompts** – You can set up the behavior to repeat the question a specific number of times, and you can select from the dropdown menu to **not repeat**, **repeat once**, or **repeat up to two times**. Similar to the skip question option, you can also use Power Fx to set the condition for this behavior to occur. You can modify the **Retry prompt** option, which only occurs if you have retries selected to repeat the question. By selecting the **Retry prompt** option, you can add a different message to reword the question, and you can add message validations to make the question sound more natural and be more helpful to the customer or user.

- **Additional entity validation/Condition** – This behavior is important if you have specific conditions to validate if an entity is valid to be slot filled and is dependent on the entity type. You can also use the same prompt behavior to change the prompt, if the conditions aren't met, to encourage the user to provide a different input.

- **No valid entity found/Action if no entity found** – If no entity is found, rather than skip the question, you can specify the behavior, such as leave the variable empty, set the variable to something specific or dynamic (by using Power Fx), or call the escalate system topic.

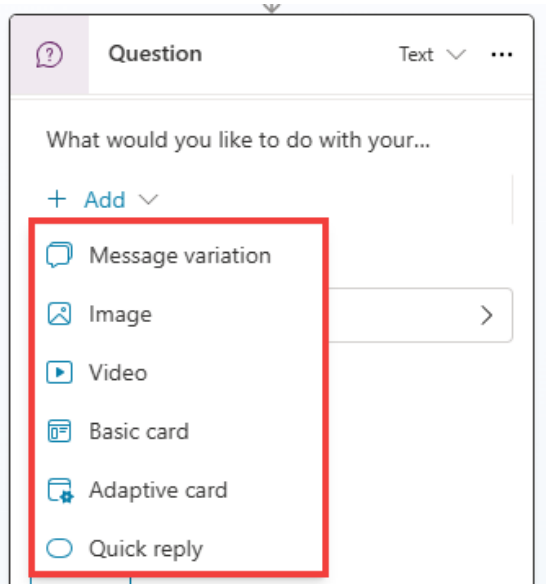
- **Interruptions** – You can indicate whether a customer should be able to switch to a different topic than the current topic that the **Question** node is in. This option is useful if a customer is likely to answer a question with another question and you want to continue the conversation without needing to handle all exceptions within a single question node.



Now that you're aware of the core functionality of the **Question** node and associated behavior, you can explore the rich text responses in the **Message** and **Question** nodes.

Rich text options for message and question nodes

Microsoft Copilot Studio includes several extended capabilities for creating copilots, and it provides positive conversational experiences for customers. One central feature is the rich text authoring capabilities that are available for the **Message** and **Question** nodes.



The types of rich text authoring options that are available include:

- **Image** – You can add an image, which is displayed on the card. Add the URL of the image and a title (optional).
- **Video** – You can add a video URL, which needs to be a publicly available MP4 or a YouTube video URL.
- **Basic card** – This option includes simple cards that provide adaptive cards, such as visuals; however, this option requires standard input such as the title, messages, and the ability to add buttons with basic actions.
- **Adaptive Card** – You can add Adaptive Cards, which are platform-agnostic cards that are designed to be flexible to suit the needs at the time, including requesting action, displaying information so that it's displayed with emphasis on specific information, or more. Microsoft Copilot Studio supports Adaptive Cards v.1.5 at the time when these labs were written.

Pro tips:

- Current version of adaptive cards is 1.6. So be sure that you are working with supported version to mitigate any troubles.
- Version that appears by default in Copilot Studio message node is currently 1.0 so suggestion is to change it to 1.5 as first step to support full functionalities.
- If you switch card content to formula there is different formatting, so be aware of this, when you are designing your cards with "Adaptive card designer".

- **Quick reply** – This option allows users to select from specific options rather than needing to enter the response in text-based scenarios. Quick replies are optional, so a user can still type or speak their own response. You should use these replies to provide common suggestions or to help give the user ideas about the type of information that's being asked.

Message variations are covered in the next task. Now, you set up a few of these options so that you can become familiar with their behavior.

Task 1: Use rich text capabilities in a message node

1. Go the **Topics** tab, select the **System** tab, then enter the **Conversation Start** topic.

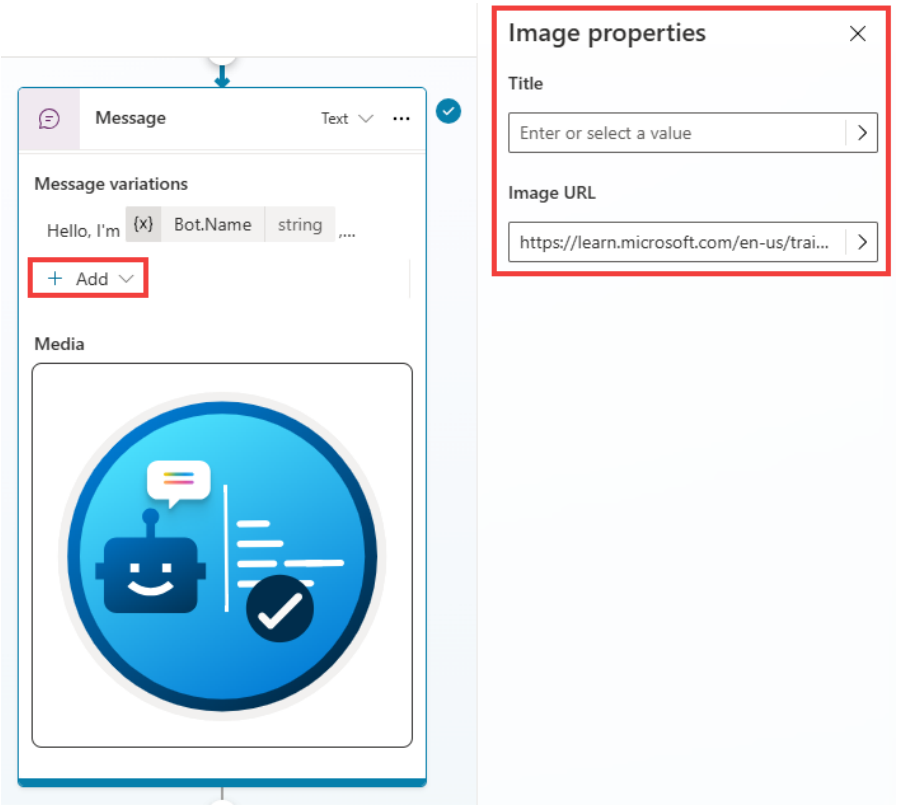
2. In the existing **Message** node, choose **+ Add** again, and then choose **Quick reply**.

```
Help with my order
```

Pro tip: quick replies are a great way to suggest available options for a user to choose from, driving a conversation to successful outcomes by proactively proposing the most common actions.

3. In same node, select **+ Add**, and then choose **Image**. Select the image block that appears, as shown in the following screenshot. The **Title** and **Image** text boxes display. You need to enter the Image URL to a publicly available image. Optionally, you can also add a name for the image by using the **Title** field.

```
https://learn.microsoft.com/en-us/training/achievements/build-
effective-bots.svg
```



4. **Save** the **Conversation Start Topic**

5. Go to the **Check Order Status** topic.

6. Add a new **Question** node after the **trigger**.

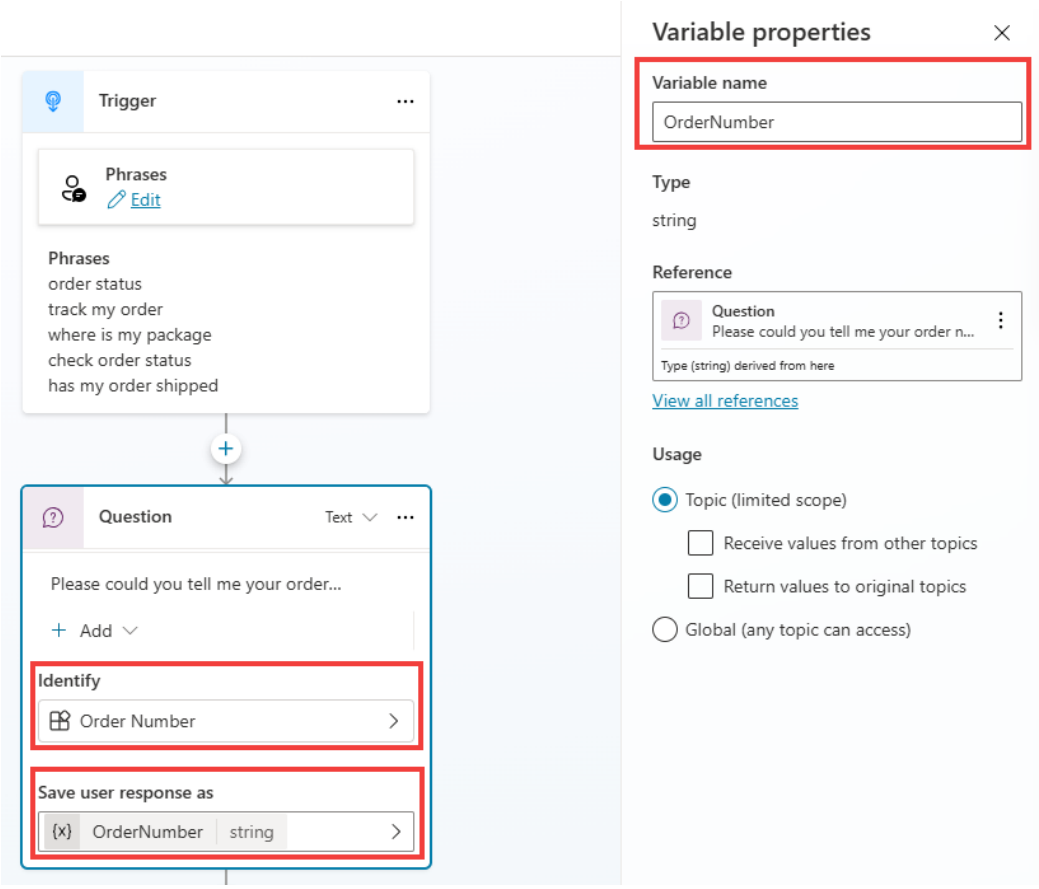
```
Please could you tell me your order number?
```

7. Change the **Identify** property and select **Create an Entity**.

8. Select **Regular expression (Regex)**, name it `Order Number`, and for its pattern use a regular expression that will automatically detect IDs such as ORD-123456

```
ORD-[0-9]{6}
```

9. Select the **Save user response as** variable menu in the new **Question** node, and in the variable properties panel to the right, change the variable name to `OrderNumber`.



Next steps: Take some time to repeat this process with the following different rich response types to become familiar with the different properties before you move on to the next task. For more information, see Send a message.

- Image
- Video
- Basic card
- Adaptive card
- Quick reply

Pro tip: For Adaptive Cards samples and authoring experiences, check these websites:

- https://adaptivecards.io/samples/
- https://adaptivecards.io/designer/
- https://amdesigner.azurewebsites.net/

Message variations

You can use the **Message** node to display a message to the user. Fundamentals of the **Message** node were covered in a previous lab and previous task in this lab. **Message** and **Question** nodes also support message variations.

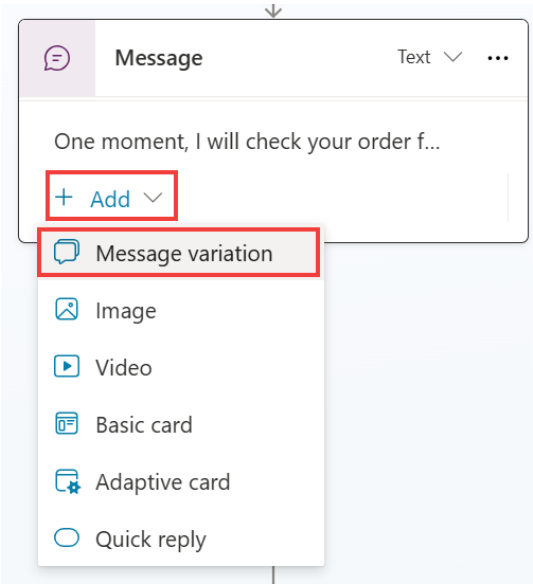
A **Message variation** allows you to add up to 15 messages in a single node so that, when the agent is triggered, it randomly selects a message variation. Message variations allow an agent author to add different styles of sentence in the same **Message** node, where at runtime, Microsoft Copilot Studio will randomly select one message variation when the topic is triggered. This feature provides authors with the ability to create natural sounding interaction and provides customers with a more natural sounding experience when they're interacting with copilots over time.

Tip: *Because message variations are selected randomly when a topic is triggered, you can add multiple versions of the same message if you want to provide an experience that leans on a certain style while providing smaller degrees of probability that still offers differentiation on some occasions.*

In the following exercise, you'll add a **Message variation** node on an existing **Message** node.

1. Open the **Check Order Status** topic that you've been working with in this lab. Select the **Message** node that you want to add a **Message variation** to, such as the one in the check order **Condition** branch shown in the preceding image.

2. Within the **Message** node, select **Add > Message variation**.



3. Add at least one message variation of your choice, or you can use the following example to observe how message variations stack in the message node. For example, under the **Update** condition path:

```
Absolutely. One moment, I'll get the order for you.
```

Congratulations, you've now completed the basics of using the **Message** node. Now, you can practice testing your message variations by selecting the **Test** option and then trigger the condition multiple times to observe it working.

The **Message** node has other properties, which is an advanced feature that's not covered in these labs.

Speech Authoring

Within Microsoft Copilot Studio, Copilot makers can use Speech Synthesis Markup Language (SSML) tags in **Message** and **Question** nodes so that they can extend the behavior when they're using Microsoft Copilot Studio for speech-enabled copilots. You can use Microsoft Copilot Studio for text authoring and speech authoring. By default, on voice-enabled channels, the message text that's entered in the message node will be used for text display and voice. You can override this behavior by providing different behavior for text and speech. For example, you'd override the behavior when you want to provide more emphasis on certain areas of a sentence or on an image message because you want to provide an alternative description that can be read aloud.

When using SSML, you can set up how the text will be converted to synthesized speech to ensure that it sounds like natural speech. You can use SSML tags like **Audio**, **Break**, **Emphasis**, and **Prosody**, to change the behavior of how your sentence is spoken.

- **Audio** - Add prerecorded audio.
- **Break** - Insert pauses or breaks between words.
- **Emphasis** - Add levels of stress to words and phrases.
- **Prosody** - Specify changes to pitch, contour, range, rate, and volume.

For more information, see Speech Synthesis Markup Language.

Code view and Power Fx

Now that you're more familiar with the authoring fundamentals in Microsoft Copilot Studio, you can explore some extended capabilities that you can use to set up and further customize the Copilot experience. The following sections cover two capabilities: **code view** for pro developers and **Power Fx** (for Microsoft Power Platform makers and professional developers).

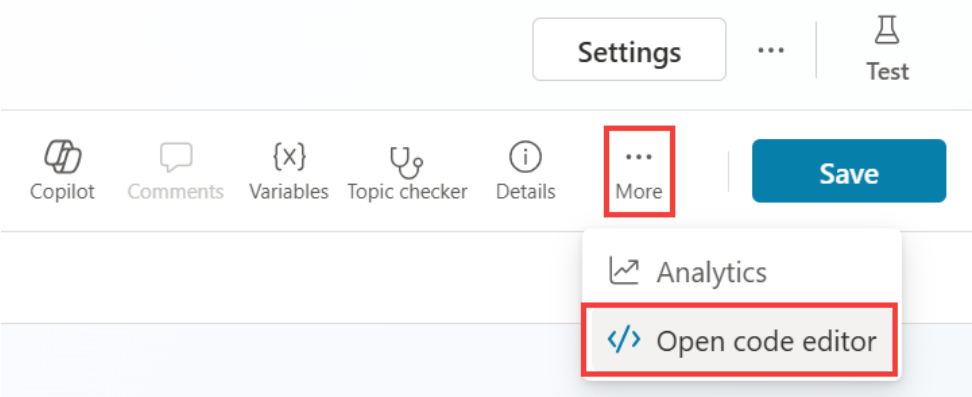
Microsoft Copilot Studio has the capability to view the code behind a topic. This capability is incredibly useful for advanced makers and pro-code developers, where they can view and edit the syntax directly within the web browser, and when saved, the syntax is immediately visible in the graphical authoring canvas. As a result, the process of copying and editing multiple actions becomes faster and easier. Some specific actions are only available in the code editor view.

Task 1: Access the code editor

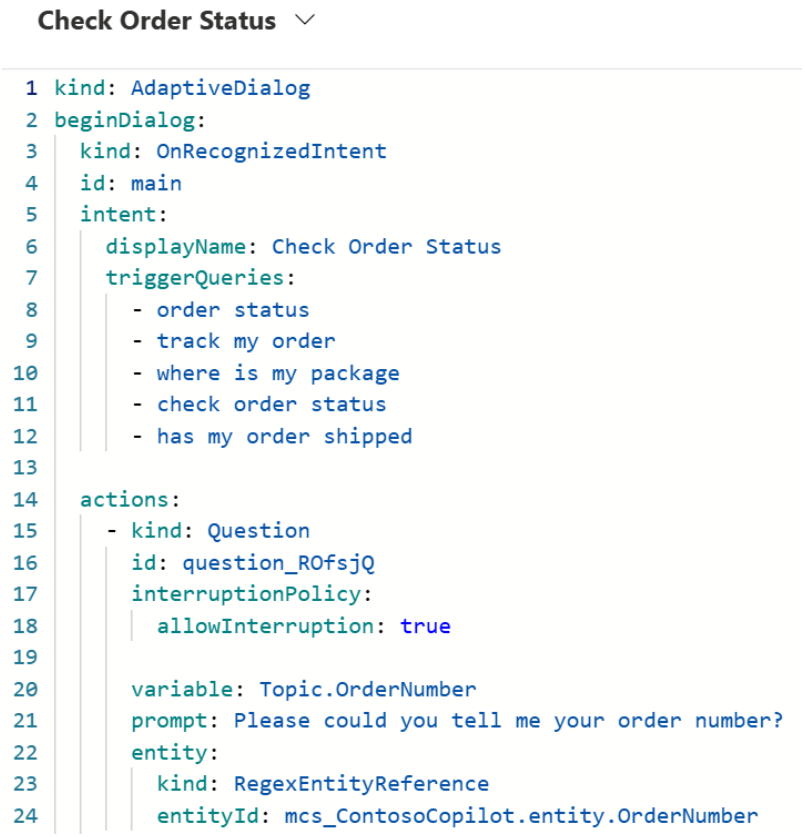
Follow these steps to access the code editor.

1. Open the topic that you've been working with in this lab titled **Check Order Status**.

2. In the upper right of the topic, next to the **Save** icon, select the extended (**...**) menu and then select **Open code editor**, as shown in the following screenshot.



3. The code editor should open, where you can view your dialog in the code view (YAML code).



4. Select **Close code editor** in the upper right after exploring this feature.

Use Power Fx across Microsoft Copilot Studio

Power Fx is available in Microsoft Copilot Studio. With Power Fx, you can add functions, similar to how makers currently do in canvas apps from Microsoft Power Apps or Dataverse within the Microsoft Copilot Studio authoring canvas. You can use Power Fx in **Message** and **Question** nodes, when you're using the **Set a variable value** node, and in other areas such as **Conditions**, **Actions**, **Question behavior** configuration and **Adaptive Cards**. This feature gives you greater control over the data that's displayed to customers and users within the conversational interface. Additionally, it allows you to perform common operations in the runtime of Microsoft Copilot Studio.

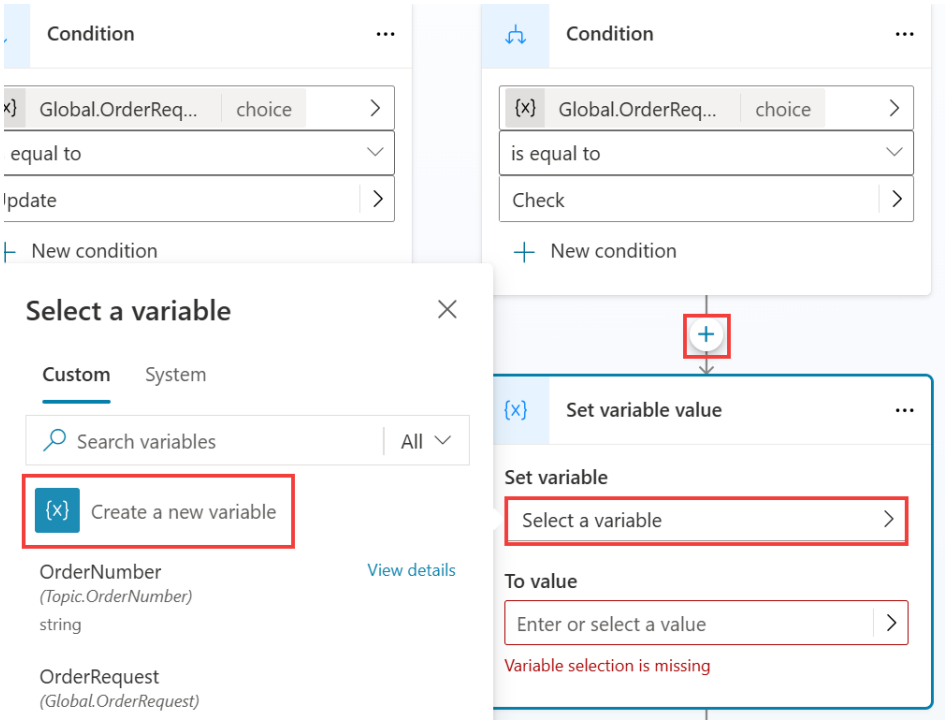
The following task goes through a basic scenario of using Power Fx within a variable and then displaying the value to the user.

Task 2: Use Power Fx to modify how the date is displayed

Follow these steps to use Power Fx to modify how the date is displayed.

1. Open the **Check Order Status** topic that you've been working with during these labs.

2. Under the **Check** condition branch, add **Set a variable value** node, and choose **Create a new variable**.

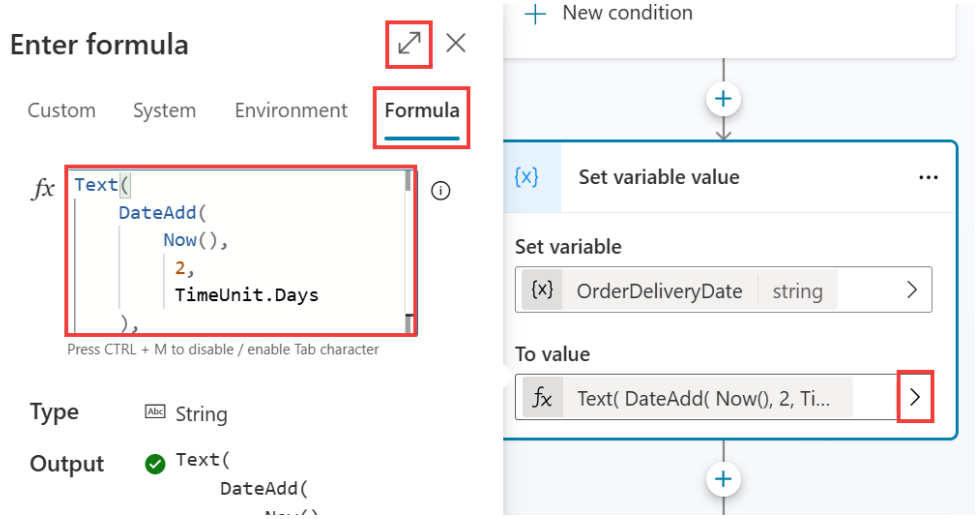


3. Selecting the new variable opens the **Variable properties** window to the right. Change the name of the variable to `OrderDeliveryDate`.

   On the **Set variable value** node, under **To Value**, select the flyout menu and then select **Formula** to open the **Enter formula** panel. Optionally, you can select the **expand** icon at the top-right of the **Enter formula** window to enlarge the area for entering a formula, and paste the below one.

4. In the formula bar, enter the following function and then select **Insert**, as follows:

```
Text(
    DateAdd(
        Now(),
         2,
         TimeUnit.Days
    ),
    DateTimeFormat.LongDate
)
```



This function takes today's date and time, which technically has a specific date and time format (e.g., 5/31/2024 8:00 AM), adds 2 days to it, and then formats it in a long date format (e.g., Friday, May 31, 2024). This approach is important if you want to display simple date formats that are user-friendly or if you want to store the date as a string, in text format.

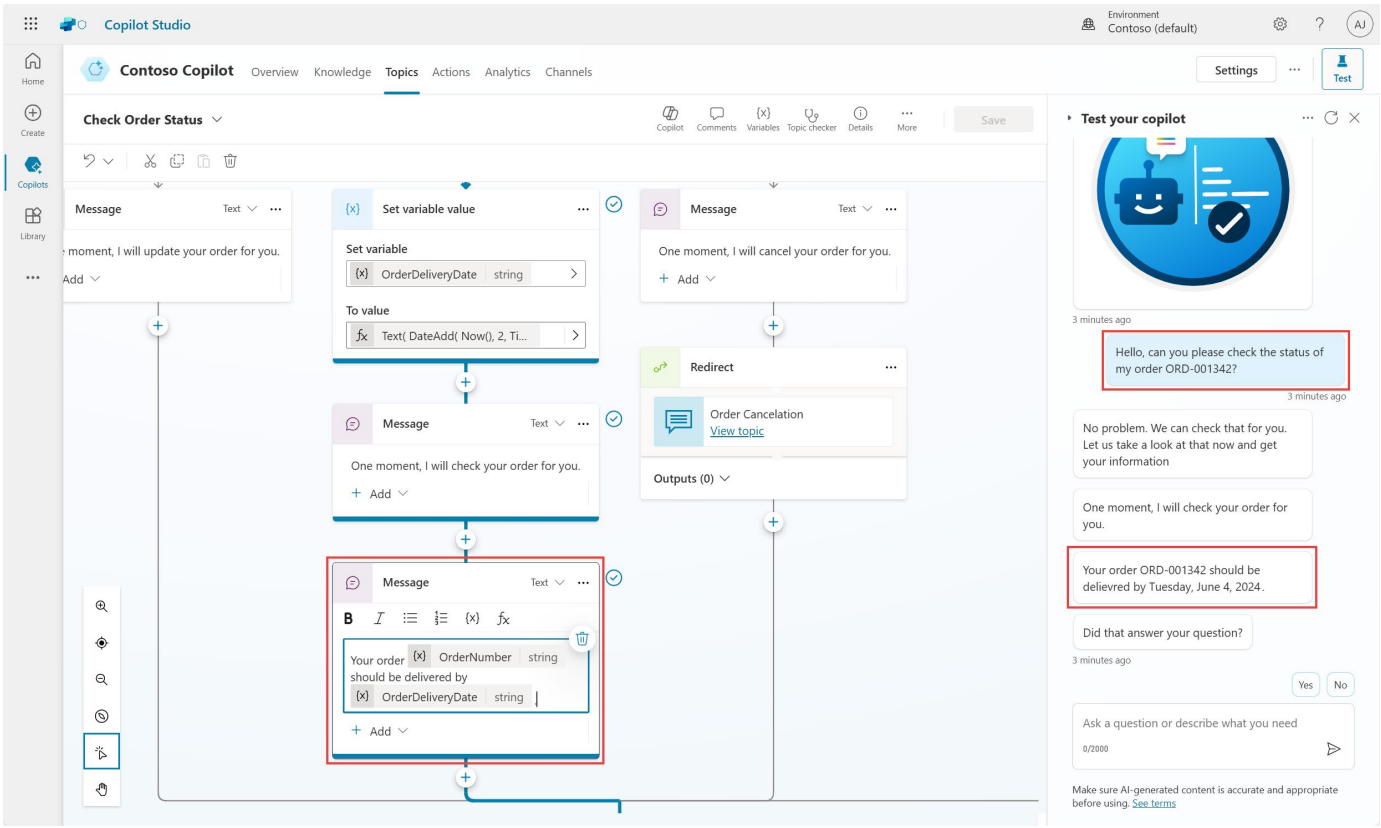
5. For demo purposes, you can update the last **Message** in the **Check** path, to use the news variables you have configured:

```
Your order {Topic.OrderNumber} should be delivered by
{Topic.OrderDeliveryDate}.
```

6. **Save and refresh the test pane**

7. Test your Copilot by opening the testing pane, trigger the topic, and then follow the topic prompts to reach your **Message** node, as shown in the following screenshot.

```
Hello, can you please check the status of my order ORD-001342?
```



Congratulations, you've successfully worked through all labs on the central authoring features in Microsoft Copilot Studio unified authoring.

Terms of Use

By using this document, in whole or in part, you agree to the following terms:

Notice

Information and views expressed in this document, including (without limitation) URL and other Internet Web site references, may change without notice. Examples depicted herein, if any, are provided for illustration only and are fictitious. No real association or connection is intended or should be inferred. This document does not provide you with any legal rights to any intellectual property in any Microsoft product.

Use Limitations

Copying or reproduction, in whole or in part, of this document to any other server or location for further reproduction or redistribution is expressly prohibited. Microsoft provides you with this document for purposes of obtaining your suggestions, comments, input, ideas, or know-how, in any form, ("Feedback") and to provide you with a learning experience. You may use this document only to evaluate its content and provide feedback to Microsoft. You may not use this document for any other purpose. You may not modify, copy, distribute, transmit, display, perform, reproduce, publish, license, create derivative works from, transfer, or sell this document or any portion thereof. You may copy and use this document for your internal, reference purposes only.

Feedback

If you give Microsoft any Feedback about this document or the subject matter herein (including, without limitation, any technology, features, functionality, and/or concepts), you give to Microsoft, without charge, the right to use, share, and freely commercialize Feedback in any way and for any purpose. You also give third parties, without charge, the right to use, or interface with, any Microsoft products or services that include the Feedback. You represent and warrant that you own or otherwise control all rights to such Feedback and that no such Feedback is subject to any third-party rights.

DISCLAIMERS

CERTAIN SOFTWARE, TECHNOLOGY, PRODUCTS, FEATURES, AND FUNCTIONALITY (COLLECTIVELY "CONCEPTS"), INCLUDING POTENTIAL NEW CONCEPTS, REFERENCED IN THIS DOCUMENT ARE IN A SIMULATED ENVIRONMENT WITHOUT COMPLEX SET-UP OR INSTALLATION AND ARE INTENDED FOR FEEDBACK AND TRAINING PURPOSES ONLY. THE CONCEPTS REPRESENTED IN THIS DOCUMENT MAY NOT REPRESENT FULL FEATURE CONCEPTS AND MAY NOT WORK THE WAY A FINAL VERSION MAY WORK. MICROSOFT ALSO MAY NOT RELEASE A FINAL VERSION OF SUCH CONCEPTS. YOUR EXPERIENCE WITH USING SUCH CONCEPTS IN A PHYSICAL ENVIRONMENT MAY ALSO BE DIFFERENT.

THIS DOCUMENT, AND THE CONCEPTS AND TRAINING PROVIDED HEREIN, IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, WHETHER EXPRESS, IMPLIED, OR STATUTORY, INCLUDING (WITHOUT LIMITATION) THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, AND NONINFRINGEMENT. MICROSOFT DOES NOT MAKE ANY ASSURANCES OR REPRESENTATIONS WITH REGARD TO THE ACCURACY OF THE RESULTS, THE OUTPUT THAT DERIVES FROM USE OF THIS DOCUMENT OR THE CONCEPTS, OR THE SUITABILITY OF THE CONCEPTS OR INFORMATION CONTAINED IN THIS DOCUMENT FOR ANY PURPOSE.

MICROSOFT COPILOT STUDIO (1) IS NOT INTENDED OR MADE AVAILABLE AS A MEDICAL DEVICE FOR THE DIAGNOSIS OF DISEASE OR OTHER CONDITIONS, OR IN THE CURE, MITIGATION, TREATMENT OR PREVENTION OF DISEASE, OR OTHERWISE TO BE USED AS A COMPONENT OF ANY CLINICAL OFFERING OR PRODUCT, AND NO LICENSE OR RIGHT IS GRANTED TO USE MICROSOFT COPILOT STUDIO FOR SUCH PURPOSES, (2) IS NOT DESIGNED OR

INTENDED TO BE A SUBSTITUTE FOR PROFESSIONAL MEDICAL ADVICE, DIAGNOSIS, TREATMENT, OR JUDGMENT AND SHOULD NOT BE USED AS A SUBSTITUTE FOR, OR TO REPLACE, PROFESSIONAL MEDICAL ADVICE, DIAGNOSIS, TREATMENT, OR JUDGMENT, AND (3) SHOULD NOT BE USED FOR EMERGENCIES AND DOES NOT SUPPORT EMERGENCY CALLS. ANY CHATBOT YOU CREATE USING MICROSOFT COPILOT STUDIO IS YOUR OWN PRODUCT OR SERVICE, SEPARATE AND APART FROM MICROSOFT COPILOT STUDIO. YOU ARE SOLELY RESPONSIBLE FOR THE DESIGN, DEVELOPMENT, AND IMPLEMENTATION OF YOUR CHATBOT (INCLUDING INCORPORATION OF IT INTO ANY PRODUCT OR SERVICE INTENDED FOR MEDICAL OR CLINICAL USE) AND FOR EXPLICITLY PROVIDING END USERS WITH APPROPRIATE WARNINGS AND DISCLAIMERS PERTAINING TO USE OF YOUR CHATBOT. YOU ARE SOLELY RESPONSIBLE FOR ANY PERSONAL INJURY OR DEATH THAT MAY OCCUR AS A RESULT OF YOUR CHATBOT OR YOUR USE OF MICROSOFT COPILOT STUDIO IN CONNECTION WITH YOUR CHATBOT, INCLUDING (WITHOUT LIMITATION) ANY SUCH INJURIES TO END USERS.