**LAB-**12 **Measure success: Track conversation outcomes and user feedback on AI responses**

**You can't improve what you can't measure: design your agent to track successful and unsuccessful outcomes while collecting user feedback on AI-generated responses.**

# Lab Details

|  |  |  |  |
| --- | --- | --- | --- |
| Level | Persona | Purpose | Estimated time to complete |
| 300 | Advanced Maker | After completing this lab, attendees will be able to design an agent that tracks conversation outcomes and collects user feedback on AI-generated responses. They will gain meaningful analytics to identify which knowledge sources drive the highest satisfaction (CSAT) and understand patterns leading to abandoned or escalated conversations. | 60 minutes |

## Prerequisites

You need to have access to Microsoft Copilot Studio using <https://copilotstudio.microsoft.com/>.

You can either customize the agent from **LAB-10 Create a knowledge agent for your public website** or create a new agent with at least one knowledge source.

## Summary of targets

Deliver a better user experience by designing an agent that not only answers questions but also tracks conversation success and collects user feedback on AI-generated responses. Gain meaningful analytics to understand which knowledge sources and topics drive the highest satisfaction (CSAT) and uncover trends behind abandoned or escalated conversations.

|  |  |
| --- | --- |
| A 100% abandoned rate won’t tell you how to improve—track outcomes to turn data into action. | |
| Agent with no tracking of conversation outcomes  with the end of conversation topic | Agent configured to track conversation outcomes  with the end of conversation topic |

Learn how to structure conversations for clear outcome tracking, simplify the end-of-conversation experience, and implement feedback collection to continuously improve your agent. Finally, explore how these insights can refine your content strategy and enhance the effectiveness of your responses.

| Use case/topic | Tagline | Page |
| --- | --- | --- |
| The end of conversation topic  [understand the purpose of the end of conversation topic and when and how to redirect to it] | Every conversation should have a conclusion – Design for clear outcomes | 3 |
| Create a smooth and intuitive end-of-conversation experience. [customize the default end of conversation topic] | End conversations without friction – create a smooth, unobtrusive way to gather feedback without disrupting the flow. | 12 |
| Add thumbs-up/thumbs-down feedback to AI responses [Use a discreet, lightweight UX to quickly capture reactions on each AI-generated response.] | Quick, intuitive reactions – replace disruptive confirmations ("Did that answer your question?") with instant, unobtrusive feedback. | 18 |
| Collect detailed user feedback after negative reactions [capture end-user feedback on AI answers] | Capture specific user suggestions when responses miss the mark – turn insights into continually improved answers. | 24 |
| Summary of learnings | Mastery is not a destination but a journey—a joyful path where every step brings growth, discovery, and endless possibilities. | 33 |
| Glossary | Speak the language, bridge the world—unlock hearts, opportunities, and the true essence of every land. | 35 |

## Documentation and Additional Training Links

* [Measuring agent engagement](https://learn.microsoft.com/en-us/microsoft-copilot-studio/guidance/measuring-engagement)
* [Measuring agent outcomes](https://learn.microsoft.com/en-us/microsoft-copilot-studio/guidance/measuring-outcomes)
* [Deflection overview](https://learn.microsoft.com/en-us/microsoft-copilot-studio/guidance/deflection-overview)
* [Key concepts – Analytics](https://learn.microsoft.com/en-us/microsoft-copilot-studio/analytics-overview)

# Use Case #1: The end of conversation topic

*Every conversation should have a conclusion – Design for clear outcomes.*

|  |  |  |
| --- | --- | --- |
| Use case | Value added | Estimated effort |
| The end of conversation topic | Effectively manage user interactions by understanding when and how to seamlessly redirect users to the end-of-conversation topic, enabling accurate tracking of conversation outcomes. | 15 minutes |

## Summary of tasks

In this section, you’ll learn how the end of conversation topic works in Copilot Studio and how to use it effectively in your conversation design. By structuring conversations for clear outcomes, you’ll enable meaningful analytics and actionable insights to improve your agent’s performance.

**Scenario**: You’ve built an agent with the knowledge to answer user questions—but is it actually delivering? If your analytics dashboard isn’t showing meaningful data, it’s time to track successful conversation outcomes and CSAT scores. You can’t improve what you can’t measure.

## Step-by-step instructions

1. Navigate to the Copilot Studio **agent** you have created for this lab (e.g., LAB-10, or a new one).

<https://aka.ms/MCSStart>

1. Go to the **Topics** tab, display **All**, and select **End of conversation**.

A screenshot of a computer screen

AI-generated content may be incorrect.

**Explore** what the topic is doing.

The **end of conversation** topic is meant to be triggered **when the agent has presumably fulfilled the user’s request**. This can happen either after providing a direct answer, such as retrieving information from knowledge sources, or after completing a more complex multi-turn interaction where the user and agent exchange multiple messages to complete a task.

By default, when the conversation reaches this stage, the agent asks, "*Did this answer your question?*" At this point, the **resolution** is considered implicit, meaning that if the user leaves without responding, it is assumed that their query was resolved. However, if the user confirms that their question was answered, the resolution becomes explicit, and they are then prompted to provide a Customer Satisfaction Score (**CSAT**) to rate their experience. After providing feedback, they are asked if they have another question—if they do, the conversation can continue.

If the user indicates that their question **was not answered**, then the outcome remains set to **abandoned**, and the agent offers them the chance to try again. If they accept, they are prompted to rephrase or ask another question. If they decline, the conversation is **escalated**, either by handing off to a human agent (if available) or by offering an alternative resolution, such as submitting a support ticket.

💡 **PRO TIPS:**

* A conversation between a user and an agent can involve multiple questions or requests. This is why individual queries and their outcomes are tracked separately as **sessions**.
* A **session** in Copilot Studio represents a distinct interaction focused on a specific user request. Within a single conversation, multiple sessions may be generated—some successfully **resolved**, while others may remain **unresolved**, **escalated**, or **abandoned**.
* Each session has an **engagement** status and **outcome**. Unengaged sessions occur when the user does not meaningfully interact with the agent and are assigned an outcome of None. Engaged sessions track whether the request was resolved, escalated, or abandoned, providing critical insights into agent performance.

1. **Explore** other topics in your agent.   
   By default, are they redirecting to the End of conversation topic?

For newly created agents, that shouldn’t be the case. The End of conversation topic must be **redirected to** explicitly from the places where you feel **the user request has been fulfilled**.

**Where do you think that should happen for your agent?**

1. Assuming the **Generative** mode for orchestration is **Disabled** on your agent (you can see that option either in the Overview tab, or in Settings, under Generative AI).   
   *Note: you may disable that option during this lab and re-enable it later.*

A black text on a white background

AI-generated content may be incorrect.

Let’s add a redirect to the **End of conversation** topic from the **Conversational boosting** topic. This is the topic that gets triggered when no matching topic (also known as intent in the field of natural language understanding) is identified. This is the topic that gets triggered when your agent uses knowledge sources to summarize answers back to the user.

**After sending out an AI answer to the user, you would want to ask for confirmation.**

Go to the **Topics** tab, display **All**, and select **Conversational boosting**.

**Explore what the topic does**

A screenshot of a computer

AI-generated content may be incorrect.

The conversational boosting topic triggers when a user query doesn’t match an existing topic and searches the configured knowledge sources to provide a grounded answer based on your content. If an answer is found, it is automatically sent to the user by the **Create generative answers** node, and the topic ends (i.e., the conversation stops). If no answer is found, the conversation goes to the **All other conditions** path, and actually continues running to the next triggered topic for the **On unknown intent** trigger, which is the **Fallback** topic (that will typically say “*I'm sorry, I'm not sure how to help with that. Can you try rephrasing?*”

Let’s change this behavior: **Delete** the **End current topic** node, and instead, add a new node: **Topic management** > **Go to another topic** > and select **End of Conversation**.

A screenshot of a computer

AI-generated content may be incorrect.

**Save** the topic.

1. Now, **test** your agent in the test pane by asking a question that will trigger the Conversational boosting topic, and see how it behaves.

For example, you may ask:

|  |
| --- |
| What are the key metrics offered by the analytics dashboard? |

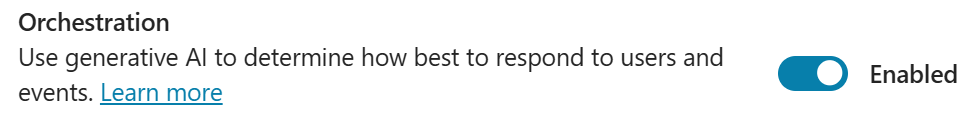
**Answer** the different questions until you can ask a new question.  
In the happy path, notice you must answer 3 questions before you can ask a new question.

A screenshot of a chat

AI-generated content may be incorrect.

1. With the **classic** orchestration (i.e., not using generative AI), we experienced that each user query that wasn’t matching a topic was triggering the Conversational boosting topic, and that the End of conversation topic was then triggered after an answer was provided to the user.

Let’s now try what the experience is by toggling the **Generative** mode on. You can enable it either in the Overview tab, or in Settings, under Generative AI.



**Refresh** the test pane, and **ask the same question**.

|  |
| --- |
| What are the key metrics offered by the analytics dashboard? |

**Notice how the experience is different**. The Activity map is displayed and shows you the agent’s reasoning based on the user query. Here, because you had knowledge sources configured and no topic was matching the user query, it chose to do a **knowledge search**.

What is great with the **generative** mode is that the agent can understand more complex queries and is able to chain multiple topics, actions, and knowledge steps together, in order to fulfill a complex request in one go.

A screenshot of a diagram

AI-generated content may be incorrect.

In the test pane, **notice** that you are no longer prompted with “*Did this answer your question?*”.

Why isn’t **End of conversation** triggered? That is because the **Conversational boosting** topic wasn’t traversed with **generative** mode.

So, how can you **redirect** to **End of conversation** after the **generative** mode provides an answer? For that, you’re going to create a **new** topic, with a special trigger.

1. Go to the **Topics** tab, select **+ Add a topic**, and choose **From** **blank**.

Don’t leave it with the default “Untitled” label. Select **Untitled** and change the text to **Plan complete**.

Then, change the trigger by **hovering** over the “**Triggered by agent**” box until the icon to **swap** the trigger for another type appears. Then scroll down and choose **Plan complete**.

Screens screenshot of a computer

AI-generated content may be incorrect.

What this topic and trigger does is letting you **customize** what happens after the generative mode is done answering a user query – a great place for **End of conversation**, don’t you think?

Add a **new** node and select **Topic management** > **Go to another topic** > **End of Conversation**.

A screenshot of a chat

AI-generated content may be incorrect.

**Save** the topic.

1. Now, **refresh** the test pane, and test your agent again.

|  |
| --- |
| What are the key metrics offered by the analytics dashboard? |

Notice that after the answer is provided by the agent, **the user is prompted for confirmation**.

Screens screenshot of a phone

AI-generated content may be incorrect.

1. **Refresh** the test pane and **send** a simple, everyday message, like “hello”.

|  |
| --- |
| Hi! |

**Notice** how the **End of conversation** topic isn’t triggered.   
**Why is that?**

Open the **Greeting** topic by clicking the **edit (✏️)** icon.

**Notice** how the **End all topics** node prevents this behavior.

Any subsequent user messages remain within the same conversation session, as the agent assumes the user's request hasn't yet been resolved.

A screenshot of a chat box

AI-generated content may be incorrect.

1. For the rest of the lab, you may **disable** **the generative mode**.

A black text on a white background

AI-generated content may be incorrect.

💡 **PRO TIPS:**

* Each topic in Copilot Studio has a code version that sometimes reveals more details than the no-code interface. The code is stored in YAML, a human-readable format that organizes the topic logic. You can access it from any topic or action by selecting “… More” and choosing "Open code editor."
* If you open the End of conversation topic in the code editor, you’ll notice hidden properties in the YAML:
  + conversationOutcome: ResolvedImplied – Used when the agent assumes the query is resolved, but the user doesn’t explicitly confirm.
  + conversationOutcome: ResolvedConfirmed – Used when the user explicitly confirms their issue was resolved.
* These properties only work in the End of conversation topic, making it essential to redirect conversations to it to properly track session outcomes. Customizing this topic allows you to tailor the user experience while ensuring accurate analytics tracking.
* The notion of implicit and explicit resolution isn’t reflected in the standard analytics dashboard, but is stored in the conversation transcripts in Dataverse, and may be surfaced when creating custom reports. See: [Custom analytics strategy](https://learn.microsoft.com/en-us/microsoft-copilot-studio/guidance/custom-analytics-strategy) and [Copilot Studio Kit Conversation KPIs](https://aka.ms/CopilotStudioKit).
* Conversations in the test pane are not logged in analytics nor do they consume capacity.

## Test your understanding

Now that you’ve explored how the end of conversation topic works, take a moment to reflect on what you’ve learned.

**Key takeaways:**

* **Conversation resolution tracking** – Redirecting to the end of conversation topic allows you to track successful, abandoned, and escalated interactions.
* **Session-based analytics** – Conversations can contain multiple sessions, each with a distinct outcome (resolved, escalated, abandoned).
* **Customizing conversation endings** – The end of conversation topic can be tailored to enhance user experience, ensuring smooth and meaningful conversation conclusions.
* **Managing conversation flow with "End all topics"** – Using the End all topics node keeps the conversation active, signaling that the user's request might still be unresolved, and prevents unintended triggers of the end-of-conversation topic.

**Lessons learned & troubleshooting tips:**

* If your analytics dashboard is showing too many abandoned sessions, check if conversations properly redirect to the end of conversation topic.
* Analytics dashboards don’t show sessions from your own tests in the test pane. Only the interactions that happened over your deployed channels will show – including the demo website.
* When using conversational boosting, ensure the topic transitions correctly to the end of conversation topic to capture user feedback.
* If enabling generative mode, be aware that it may bypass standard conversation flows—consider using the Plan complete trigger to regain control over conversation endings.
* Use YAML properties (conversationOutcome: ResolvedImplied or ResolvedConfirmed) to track conversation success at precise points in your End of conversation flow.
* Implicit and explicit resolution details are not visible in the standard analytics dashboard but are stored in conversation transcripts in Dataverse. You can surface this data when creating custom reports.

**Challenge: apply this to your own use case**

* How can you integrate clear conversation endings into your existing agent topics?
* Where should you collect user feedback to improve response quality?
* What patterns can you identify in abandoned vs. resolved conversations?

Take it further: Experiment with different ways to end conversations smoothly while ensuring user feedback is captured effectively. Test how generative mode impacts conversation tracking and whether it bypasses standard flows. Explore the Plan complete trigger to regain control over conversation endings. Finally, analyze session outcomes in your analytics dashboard to uncover insights that help refine your agent’s design and improve resolution rates.

|  |
| --- |
|  |
|  |
|  |
|  |

# Use Case #2: Create a smooth and intuitive end-of-conversation experience

*End conversations without friction – create a smooth, unobtrusive way to gather feedback without disrupting the flow.*

|  |  |  |
| --- | --- | --- |
| Use case | Value added | Estimated effort |
| Create a smooth and intuitive end-of-conversation experience | Customize the default end-of-conversation topic to create a more seamless, conversational-friendly experience for users. | 15 minutes |

## Summary of tasks

In this section, you'll learn how the default End of conversation topic can unintentionally interrupt the user’s conversational flow, forcing unnecessary feedback prompts or confirmations. You'll see how to modify this default behavior to create a smoother and more intuitive experience.

**Scenario**: Visitors on your website frequently have multiple related questions about products and solutions. The default End of conversation prompt can disrupt their experience by forcing them into providing feedback or acknowledgments prematurely. By customizing the End of conversation topic, you’ll enable a more fluid interaction, allowing users to naturally continue conversations without friction or interruption. This results in improved user satisfaction, more accurate tracking of outcomes, and enhanced overall engagement.

## Step-by-step instructions

1. **Let’s start with a test**.   
   **Refresh** the test pane and **ask two questions consecutively**.

|  |
| --- |
| What is Copilot Studio? |

|  |
| --- |
| What knowledge sources does it support? |

Notice how the default End of conversation topic interrupts the interaction, preventing the second question from being answered until the user responds to the prompt, **"Did that answer your question?"**

A screenshot of a chat

AI-generated content may be incorrect.

1. Go to the **Topics** tab, display **All**, and select **End of conversation**.

On the **question** “Did that answer your question?”, select the **ellipsis (…)** an open **properties**.

Go to **Question behavior**.

Set **How many reprompts** to **Don’t repeat**.  
Now, if the user provides a response that isn't recognized as confirmation or negation—such as a follow-up question—the agent won’t repeat the question, ensuring a smoother conversation flow.

Return to the **Question Properties**, then select **Entity recognition**:

For **Action if no entity found**, choose **Set variable to empty (no value)**.

If the agent doesn’t recognize a confirmation or negation, the variable that represents the expected answer to that question remains blank, allowing the conversation to proceed without interruption.

1. Below the "Did that answer your question?" **question**, notice the **condition** only tests if SurveyResponse is true. Therefore, a follow-up question without a clear confirmation (where SurveyResponse remains blank) will incorrectly fall under "**All other conditions**", triggering the message: "Sorry I wasn't able to help better. Would you like to try again?"

**Let’s add another condition path**, by clicking on the **(**➕**) action** above the various conditions:

A screenshot of a computer

AI-generated content may be incorrect.

**In Select a variable, choose** SurveyResponse. For the test, leave it to **is equal to**, and set **false** for the value.

Let’s now **move everything** that is under **All other conditions** under the new **false** branch.

You can do so by clicking on the “Sorry I wasn't able to help better. Would you like to try again?” **question** and selecting the **cut (✂️)** action in the toolbar.



Under the **false condition branch**, select the **(➕)** action and select **Paste**.

**Do the same steps for the entire sub Condition branch, and paste those under the question you just pasted.**

The **All other conditions branch** should now be empty**.**

1. Assuming you will get follow-up questions and want to redirect back to your knowledge sources, add a **redirect** to the **Conversational boosting** topic under the **All other conditions** path. The three branches should look like below.

A screenshot of a computer

AI-generated content may be incorrect.

1. **Save** your topic.
2. **Let’s do a new test**.   
   **Refresh** the test pane and **ask again two questions consecutively**.

|  |
| --- |
| What is Copilot Studio? |

|  |
| --- |
| What knowledge sources does it support? |

**Notice how follow-up questions are no longer blocking the conversation flow.**

💡 **PRO TIPS:**

* Triggering existing topics instead of Conversational boosting: if you want the user’s follow-up questions to trigger existing topics (with configured trigger phrases or descriptions) rather than always defaulting to Conversational boosting, verify the Interruptions setting, in the "Did that answer your question?" question properties. By default, they allow interruptions, meaning the agent can seamlessly switch to a recognized topic based on the user's next input.

A screenshot of a chat

AI-generated content may be incorrect.

1. **Play around** with the new configuration and do multiple tests, some by responding positively or negatively to the “Did that answer your question”, or by asking anything else.
2. You can **further simplify** what happens after the user answers **Yes**.

After the **CSAT** question, add a **message** node asking

|  |
| --- |
| Thank you for your feedback!  Feel free to ask me something else. |

Then add a node **Topic management** > **End conversation**.

**Delete everything further below that path.**

A screenshot of a chat

AI-generated content may be incorrect.

1. You can **further simplify** what happens after the user answers **No**.

Under the SurveyResponse **false** condition path, add a new **message** node

|  |
| --- |
| Sorry I wasn't able to help better.    You may try reaching out to our [Microsoft Copilot Studio community](https://aka.ms/CopilotStudioCommunity) or submitting a [support request](https://learn.microsoft.com/en-us/power-platform/admin/get-help-support).    Would you like to try again? Feel free to ask a new question. |

Then add a node **Topic management** > **End conversation**.

**Delete everything further below that path.**

**Save** topic

## Test your understanding

Now that you've explored creating a smooth and intuitive end-of-conversation experience, reflect on the key points you've learned.

**Key takeaways:**

* Seamless Conversation Flow: default End of Conversation questions can disrupt user interactions. Customizing these ensures users aren’t forced into unnecessary confirmations.
* Flexible User Responses: adjusting question behavior and entity recognition settings allows conversations to continue fluidly even if a user's response doesn't explicitly confirm or deny satisfaction.
* Enhanced User Experience: a more conversational-friendly approach to ending topics helps users naturally transition between multiple related questions.

**Lessons learned & troubleshooting tips:**

* If users complain about interrupted experiences, revisit your End of Conversation questions to ensure they don’t unnecessarily block conversation flow.
* Regularly test conversation paths by responding in various ways—positively, negatively, or neutrally—to verify the End of Conversation logic.
* Utilize "Interruptions" settings to allow new topics to trigger seamlessly from follow-up user inputs.

**Challenge: apply this to your own use case**

* Where in your current agent can users experience unnecessary conversational friction?
* How can you apply these customization strategies to ensure smoother conversation endings?
* In what ways can user feedback be naturally integrated without interrupting conversational flow?

Take it further: experiment with additional customizations that allow smooth transitions and continuous user engagement. Assess analytics regularly to refine your agent’s conversational experience, optimizing both user satisfaction and response quality.

|  |
| --- |
|  |
|  |
|  |
|  |

# Use Case #3: Add thumbs-up/thumbs-down feedback to AI responses

*Simple reactions, smoother conversations.*

|  |  |  |
| --- | --- | --- |
| Use case | Value added | Estimated effort |
| Add thumbs-up/thumbs-down feedback to AI responses | Quick, intuitive reactions—replace disruptive confirmations ("Did this answer your question?") with instant, unobtrusive feedback. | 15 minutes |

## Summary of tasks

In this section, you'll learn how to enhance user interactions by embedding an intuitive thumbs-up/thumbs-down Adaptive Card after each AI-generated response. This approach streamlines feedback collection, replacing disruptive prompts ("Did that answer your question?") with simple, unobtrusive reactions. You’ll configure the End-of-conversation topic to capture conversation outcomes and include a lightweight Customer Satisfaction (CSAT) prompt after positive feedback, while immediately allowing users to continue asking new questions. Additionally, you’ll ensure the thumbs-up/thumbs-down UX appears only after AI-generated answers, preserving the standard feedback prompt for manually authored topics.

**Scenario**: Website visitors exploring your products and solutions need quick answers without disruption. By implementing intuitive thumbs-up/thumbs-down feedback after each AI-generated response, you simplify user interactions—eliminating intrusive confirmation prompts and capturing valuable sentiment effortlessly. Positive reactions can seamlessly prompt a lightweight satisfaction survey, while visitors retain the freedom to instantly ask follow-up questions. This streamlined approach increases user engagement, improves satisfaction, and provides actionable insights to continuously enhance your AI agent’s effectiveness.

## Step-by-step instructions

1. Go to the **Topics** tab, display **All**, and select **End of conversation**.

Open the topic **Details**: in the **Input** tab, add a new variable AIAnswer, of type **Boolean**.

We’re doing this in order to identify when we’re entering the End of Conversation topic in the context of an AI-generated response.

💡 **PRO TIPS:**

* Topic inputs are great when you redirect to them, as it allows you to pass values between one topic to another – when using generative mode, they can even be automatically filled by the AI from the context.

1. Add a **condition node** right after **the topic trigger**.

In **Select a variable**, choose AIAnswer, and set is equal to **true**.

1. Select the **“Did that answer your question?”** question node, and **Copy** it.

**Paste** it under the **true** path.

1. Select the **“Did that answer your question?”** question node again, but this time **Cut** it.

**Paste** it under the **All other conditions** path.

A screenshot of a computer

AI-generated content may be incorrect.

1. Now **clear** the “Did that answer your question?” text from the question node the AIAnswer **true** path **without deleting the node itself**.

Select **+** **Add** and choose **Adaptive card**

*For this lab, we’ll provide you with a template, but you can explore many other great templates. For best performance avoid using embedded data URLs for images – instead, use hosted URLs.*

**Go** to [thumbs-feedback-card.json](https://github.com/microsoft/CopilotStudioSamples/blob/main/AdaptiveCardSamples/thumbs-feedback-card.json) and copy the code.

**Paste** the JSON in the Adaptive Cards properties input box.

**Save** your topic.

1. Go to the **Conversational boosting** topic.

💡 **PRO TIPS:**

* Navigate from one topic to the other by using the chevron icon next to the topic name.

Notice that the **Redirect** **to End of Conversation** node now has a **+ Add** input option.   
**Select** AIAnswer in the list and set the value to **true**.

**Save** your topic.

1. Apply the same to the **Plan Complete** topic.
2. **Let’s test**. **Refresh** the test pane and **ask a question**.

*Note: if you still receive the “Did that answer your question?” message, go to the End of conversation topic, go to the* ***… More*** *menu and go to the* ***code editor view****. Search for variable:* ***init:Topic.SurveyResponse*** *and remove the* ***init:*** *part from it before* ***saving****.*

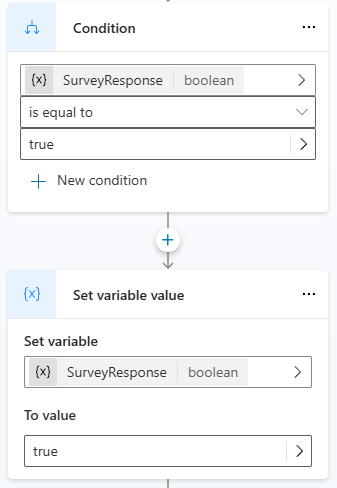
A screenshot of a phone

AI-generated content may be incorrect.

**Observe the response**: Notice that even though we have a nice adaptive card with thumbs up and thumbs down buttons, the agent still suggests **Yes** and **No** as quick replies. This is because the SurveyResponse variable is of **Boolean** type, and Copilot Studio always surfaces Yes and No as quick replies for these.

1. Go to the End of Conversation topic.

**Under** the SurveyResponse is equal to **true** branch, add a **Set a variable value** node, and select the SurveyResponse one and set it to **true**.



Do the same under the SurveyResponse is equal to **false** branch, add a **Set a variable value** node, and select the SurveyResponse one and set it to **false**.

**We’re doing this so that Copilot Studio can track those standard variables as we’re going to change the variable type for the standard initial questions that were using Boolean.**

1. Go back to the **thumbs-up thumbs-down question**, and change the **Identify** value, instead of Boolean, choose **Create an Entity**.

Choose **Closed list**

Name: **Helpful**

Description: **Determines if the provided answer or actions were useful for the user to answer their query.**

For items, add:

**Yes**, with a synonym of **Yes, this was useful**.

**No**, with a synonym of **No, this didn't help**.

Turn **Smart matching** on

**Save** the entity.

In **Save user response as**, select **Create a new variable**.

It will automatically be named as Var1.

Select Var1 to change its name to IsHelpful.

1. On the “**Did that answer your question?**” question node, also change Identity and select the **Is Helpful** entity.

To continue to offer **Yes**, **No** suggested answers, **select options for user** and pick both for **display**.

**Notice** the user interface doesn’t let you update the **Save user response as** to the IsHelpful variable. Select **Create a new variable**, it will again automatically be named as Var1.

Switch to the code editor view, and **search** for Topic.Var1, **update** the value to Topic.IsHelpful and **close** the code editor.

1. **Update** the two condition nodes, so that they use the IsHelpful as the condition variable, and select the **Yes**, and **No** value in place of **true** and **false**.

**Save** the topic.

1. **Let’s see if this works!  
   Refresh** the test pane and **ask your agent a question**.

A screenshot of a chat

AI-generated content may be incorrect.

## Test your understanding

Reflect on what you've learned about adding thumbs-up/thumbs-down feedback to AI responses:

**Key takeaways:**

* Simple feedback improves user experience – Using thumbs-up/thumbs-down reactions creates a smooth, unobtrusive way to collect feedback on AI responses without interrupting the conversation flow.
* Adaptive Cards enhance interactions – Rich yet discreet Adaptive Cards effectively capture user sentiment and provide actionable insights.
* Conditional feedback display – You can selectively apply the thumbs-up/thumbs-down UX exclusively to AI-generated responses, keeping traditional confirmation prompts for standard topics.
* Conversation outcome tracking – Integrating feedback into the End-of-conversation topic allows you to track resolution success accurately and continuously improve your agent.

**Lessons learned & troubleshooting tips:**

* If standard Yes/No quick replies appear: Check your variable type. Replace Boolean variables with a custom closed list entity to avoid unwanted quick replies.
* Missing feedback from analytics: Ensure your feedback responses update standard tracking variables to maintain analytics accuracy.
* Conditional UX display: Always verify the conditional paths in your End-of-conversation topic clearly separate AI-generated responses from manually-authored topic responses.

**Challenge: apply this to your own use case**

* Where could streamlined feedback improve interactions on your own website?
* What insights could you gain by differentiating feedback between AI-generated and authored content?
* How might integrating lightweight CSAT questions following positive feedback enhance customer satisfaction?

Take it further: Experiment with other Adaptive Card templates to enhance user experience. Review your analytics to measure user engagement before and after implementing feedback cards.

|  |
| --- |
|  |
|  |
|  |
|  |

# Use Case #4: Collect detailed user feedback after negative reactions

*Capture specific user suggestions when responses miss the mark – turn insights into continually improved answers.*

|  |  |  |
| --- | --- | --- |
| Use case | Value added | Estimated effort |
| Collect detailed user feedback after negative reactions | Easily capture detailed end-user feedback after negative reactions—gaining actionable insights to continuously refine and enhance your agent’s responses. | 15 minutes |

## Summary of tasks

In this section, you’ll learn how to collect detailed user feedback after a negative reaction to an AI-generated response. Instead of forcing immediate survey participation, users will have the choice to retry their question or provide structured feedback via an Adaptive Card. This card will present the original question and AI response and ask users what was wrong with the answer. While this feedback isn't logged in built-in analytics, it will be captured in conversation transcripts and can be integrated with external systems like Azure Application Insights, Dataverse, SharePoint, or Excel for human review and further analysis.

**Scenario**: For businesses leveraging AI assistants, ensuring accurate and relevant responses is key to maintaining customer trust and engagement. However, when answers fall short, users may abandon the conversation, leaving teams without insight into what went wrong.

By offering a structured feedback option after a negative reaction, organizations can capture valuable user input without disrupting the experience. Instead of forcing users into a survey, they have the choice to retry their question or provide feedback on why the response was unhelpful.

This feedback becomes a critical resource for continuous improvement. Teams can identify recurring issues, refine the assistant’s responses, and update knowledge source content to address gaps. Over time, this process enhances answer accuracy, increases user satisfaction, and ensures the AI assistant evolves to meet customer needs more effectively.

## Step-by-step instructions

1. For our feedback collection process on wrong answers, we’re going to create an Adaptive Card form summarizing the initial user question, the provided AI-generated answer, and an input text box so that the user can provide comments.

For this to work, we’re going to need the user answer and generated answer to be available across topics, as **global variables**.

Go to the **Topics** tab, display **All**, and select **Conversational boosting**.

Immediately after the **Trigger**, add a new **Set a variable** value node.

Create a new **variable**, and name it UserQuery.

In the variable properties, set the **Usage to Global (any topic can access)**.

If used in **generative** mode, you can make it available to the broader orchestration context by setting the **Dynamic chaining visibility** to **Use in dynamic chaining**.

💡 **PRO TIPS:**

* Custom variables are by default scoped to the topic you are in.
* Variables can be set to have a global scope. Global variables are useful when you need values to be available across all topics and actions. Global variables can also be set from external sources, which is very useful for example when you want to lead your agent on your website by passing custom context to it (user language, current web page, custom IDs, etc.). Learn more about variables here: [Variables overview](https://learn.microsoft.com/en-us/microsoft-copilot-studio/authoring-variables-about?tabs=webApp).

**Set** the value of the variable to Activity.Text **system** **variable**.

💡 **PRO TIPS:**

* The Activity.Text variable is used to get access to the last message the user has sent.

Screens screenshot of a computer

AI-generated content may be incorrect.

1. **Go** the **Create generative answers** node, select the **ellipses** **(…)**, and choose **Properties**.

**Scroll down** and expand the **Advanced** tab.

Set **Save LLM response** to **Complete (recommended)**.

💡 **PRO TIPS:**

* Saving the LLM response as “Complete (recommended)” allows to have access to a more detailed answer variable object, containing both the source citations – i.e. the source content that was used to summarize an answer to the user – as well as variations of the answer (with and without Markdown formatting).

Select the **Answer** variable, and make it **Global** too (and available to **Dynamic** **chaining**)

A screenshot of a computer screen

AI-generated content may be incorrect.

1. **Save** the topic.
2. **Do a test**: **refresh** the test pane, and **ask a question**

In the test pane, select on **the generated answer text**.

**Notice** how this will display the **Conversational boosting** topic (if not already opened).

💡 **PRO TIPS:**

* In classic mode, from the test pane, you can click on any user or agent message to navigate to the source topic it was sent from.

From the **command bar**, **open** the **Variables** pane.

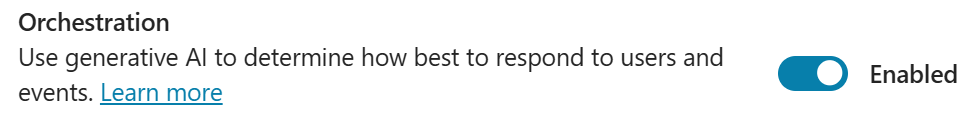
A screenshot of a computer

AI-generated content may be incorrect.

Go the **Test** tab, and expand the **Global** menu**.**

**Notice** that you see the runtime values of your test: you can see both the Global.UserQuery value along the full Global.Answer record.

1. **Optional:** Let’s make this work for generative mode as well, by toggling the **Generative** mode on. You can enable it either in the Overview tab, or in Settings, under Generative AI.



The answer record is going to be slightly different than in the **Conversational boosting** topic. To access it, you actually need to create a new topic.

Go to the **Topics** tab, select **+ Add a topic**, and choose **From blank**.

Change the name from **Untitled** to **AI response generated**.

Change the **trigger** to **AI response generated**.

A screenshot of a computer

AI-generated content may be incorrect.

💡 **PRO TIPS:**

* Understanding the AI response generated trigger – The AI response is generated when the agent, in generative mode, synthesizes information from all outputs from topics, knowledge sources, and actions to provide a summarized answer to the user query.
* Gaining more control over AI responses – By default, the AI-generated answer is sent automatically. However, if more control is needed, makers can intercept the response before it’s sent. To prevent automatic delivery, they can use a Set a variable value action in this topic and set the ContinueResponse system variable to false.

Add a **Set variable value** node, for Set variable, choose Global.Answer.Text.MarkdownContent, and for **To value**, Response.FormattedText

**Save** topic.

**You may do a test and look at the AI response generated topic runtime global variable values to see they are properly recorded as well in a generative mode.**

To continue with the lab scenario, disable **generative** orchestration.

A black text on a white background

AI-generated content may be incorrect.

1. Now let’s look at how we’re going to **capture** **user feedback** **on a given answer.** For this we’re going to create a new AI Feedback topic.

Go to the **Topics** tab, select **+ Add a topic**, and choose **From blank**.

Change the name from **Untitled** to **AI Feedback**.

In **Phrases**, add “**I want to provide feedback**” as a trigger phrase

To make sure this topic doesn’t get triggered by mistake, go to the **Trigger** ellipses **(…)** and to **Properties**.

Switch **Condition** to **Formula**. Select the chevron icon and go to the **Formula** tab to **paste** the below **Power Fx** expression. What this does is that the topic is only triggered/entered if the user query is exactly “I want to provide feedback”.

|  |
| --- |
| System.Activity.Text = "I want to provide feedback" |

A screenshot of a computer

AI-generated content may be incorrect.

1. Next, we’re going to add an Adaptive Cards form. Forms allow you to capture multiple pieces of data—such as text, dates, and numbers—in a single, structured experience, making it easier for users to provide detailed input efficiently.

💡 **PRO TIPS:**

* When to use "Ask with Adaptive Card" – This node is best suited for forms or Adaptive Cards that require user input. If your Adaptive Card doesn’t contain input fields, consider attaching it to a message or question node instead.
* Limitations of "Ask with Adaptive Card" – Unlike standard question nodes, this option has limited configuration flexibility. Users must submit the card before continuing the conversation—it cannot be skipped or bypassed.

Add an **Ask with adaptive card** node.

Because we’re going to make the adaptive card, dynamic – i.e. we want to show a repeat of the original user question and of the provided AI-generated answer, **switch the format** of the Adaptive Card from **Edit JSON** to **Formula**. Notice how this automatically transforms the code from a **JSON object** to a **Power Fx record**.

*For this lab, we’ll provide you with a template, but you can explore many other great templates. Check out the Documentation and Additional Training Links section to find more resources!*

**Go** to [CopilotStudioSamples/AdaptiveCardSamples/ai-feedback-adaptive-card.fx](https://github.com/microsoft/CopilotStudioSamples/blob/main/AdaptiveCardSamples/ai-feedback-adaptive-card.fx) and copy the Power Fx code.

**Paste** the code.

After the **Ask with adaptive card** node, add a **Message** node and send:

|  |
| --- |
| Thank you! Feel free to ask me anything else. |

Then add a **Topic management** node, and select **End the conversation**.

**Save** the topic.

A screenshot of a chat

AI-generated content may be incorrect.

💡 **PRO TIPS:**

* With this approach, feedback isn’t surfaced in the standard analytics dashboards, but is logged in the conversation transcripts in Dataverse and are hence available for a [custom analytics strategy](https://learn.microsoft.com/en-us/microsoft-copilot-studio/guidance/custom-analytics-strategy), or through the conversation KPIs feature of the [Copilot Studio Kit](https://aka.ms/CopilotStudioKit).
* You may consider adding an action after the feedback Adaptive Card form is submitted to send the captured data to any desired system of record (Dataverse, SharePoint Lists, Excel file, etc.) for review.

1. **Navigate** to the **End of conversation** topic.

We’re going to offer users the ability to provide feedback if they select the **thumb down** icon.

**Under** the **Set variable value** SurveyResponse to false node (under the IsHelpful is equal to false condition branch), **add a new condition**: we only want to offer users the option to provide feedback when we’re in the context of an AI-generated answer.

**Set the condition** to AIAnswer is equal to **true**.

**Cut** the “Sorry I wasn’t to help you better […]” message node, and **paste it twice**, under both condition branches.

**For the one under** the AIAnswer is equal to **true**, select **+ Add** to add a **Quick reply**, and use the below sentence (it must exactly match the one we have configured for the AI Feedback topic!)

|  |
| --- |
| I want to provide feedback |

A screenshot of a computer screen

AI-generated content may be incorrect.

**Save** the topic.

1. Let’s test! **Ask a question**, then **select** the **thumb down** button.

Instead of asking a follow up question, **select** the “**I want to provide feedback**” option.

**Notice** how the user is now prompted to fill out an adaptive card summarizing their question along with the AI-generated answer and is asked to provide feedback.

A screenshot of a phone

AI-generated content may be incorrect.

## Test your understanding

Now that you’ve implemented a feedback collection process for negative reactions in your Copilot Studio AI Assistant, reflect on the key points you’ve learned.

**Key takeaways:**

* Actionable Insights – Collecting structured feedback allows you to understand where and why responses are failing, enabling continuous improvements.
* User-Centric Approach – Providing users with a seamless way to retry their question or give feedback reduces frustration and encourages engagement.
* Analytics & Integration – While feedback is not logged in built-in analytics, capturing it in conversation transcripts enables integration with external systems like Dataverse, SharePoint, or Azure Application Insights for further analysis.

**Lessons learned & troubleshooting tips:**

* Regularly review conversation transcripts to identify patterns in negative reactions and refine response strategies accordingly.
* Ensure that feedback requests do not disrupt the natural flow of conversation—offering an easy way to skip or retry a query keeps users engaged.

**Challenge: apply this to your own use case**

* Where in your current AI assistant could feedback collection be improved?
* How can you adjust the feedback prompt to encourage higher response rates without frustrating users?
* What mechanisms can you implement to analyze and act on feedback effectively?

Take it further: Consider automation options for processing feedback data, such as flagging recurring issues or triggering alerts for human review, to enhance your AI assistant's continuous learning and adaptation.

|  |
| --- |
|  |
|  |
|  |
|  |

# Summary of learnings

Une image contenant jaune, conception

Le contenu généré par l’IA peut être incorrect.*Mastery is not a destination but a journey—a joyful path where every step brings growth, discovery, and endless possibilities.*

Congratulations on completing this lab! You’ve explored essential strategies for measuring the success of agent by tracking conversation outcomes and collecting user feedback on AI-generated responses. These insights will help you refine your assistant to enhance user satisfaction and optimize engagement.

* **Tracking conversation success –** Implementing structured conversation outcomes using the End of Conversation topic to monitor resolved, abandoned, and escalated interactions.
* **Enhancing user feedback collection –** Enabling thumbs-up/thumbs-down reactions to AI-generated responses for quick sentiment analysis without disrupting user flow.
* **Collecting detailed insights –** Capturing structured feedback from users when AI responses miss the mark, turning qualitative input into actionable improvements.
* **Refining conversation design –** Ensuring smooth and intuitive end-of-conversation experiences, preventing unnecessary disruptions while still capturing valuable engagement data.
* **Leveraging analytics –** Understanding which knowledge sources and topics drive the highest customer satisfaction (CSAT) and recognizing patterns behind abandoned or escalated conversations.
* **Optimizing generative AI behavior –** Controlling how generative AI integrates with structured tracking to maintain visibility into user interactions and resolution rates.

**Conclusions and recommendations**

To maximize the effectiveness of your Copilot Studio AI Assistant:

* Ensure every conversation has a clear outcome by redirecting interactions to the End of Conversation topic when appropriate.
* Regularly review conversation analytics to identify areas where users abandon interactions or express dissatisfaction.
* Use thumbs-up/thumbs-down reactions to collect real-time sentiment data without interrupting natural conversation flows.
* Implement detailed feedback collection for negative interactions, allowing users to provide constructive suggestions for improving responses.
* Align feedback mechanisms with your organization’s data strategy by storing responses in Dataverse, SharePoint, or Azure Application Insights for deeper analysis.
* Continuously refine your assistant based on user feedback, adjusting knowledge sources, conversational flow, and AI responses to improve accuracy and relevance.

By integrating these best practices, you will transform your AI Assistant into a data-driven, continuously improving tool that delivers real value to users and enhances the overall conversational experience.

**We want your feedback!**

[**Start now**](https://aka.ms/MCSLabsFeedback)

**Recommended Next Steps**

To continue building your expertise, consider diving into these advanced labs:

* **TBD**

# Glossary

*Speak the language, bridge the world—unlock hearts, opportunities, and the true essence of every land.*

**End of Conversation Topic:**A designated topic in Copilot Studio that helps track and categorize conversation outcomes, such as resolved, abandoned, or escalated interactions.

**CSAT (Customer Satisfaction Score):**A key metric used to measure user satisfaction based on their interactions with the AI assistant.

**Conversation Analytics:**Insights derived from conversation data that help track user engagement, resolution rates, and abandonment patterns.

**Generative mode:**A setting that allows the AI assistant to generate responses dynamically by synthesizing information from various knowledge sources.

**Conversational Boosting:** A fallback mechanism that directs users to knowledge sources when no predefined topic matches their query.

**Adaptive Cards:** Rich, interactive UI elements used in chatbots to capture structured feedback or display formatted information.

**Thumbs-up/Thumbs-down Feedback:** A lightweight user feedback mechanism that allows quick sentiment capture on AI-generated responses.

**Escalated Conversation:**A conversation that could not be resolved by the AI assistant and requires human intervention.