Example 3 — Teams bot

A Microsoft Teams channel bot, powered by Ask Sage, built in Power Automate.

> Prerequisites: the Prerequisites section of the Ask Sage Power Automate docs ·
> Auth & secure key storage: the Authentication section of the Ask Sage Power Automate docs.

You can get this bot two ways — both produce the **same** flow (threaded conversation history + bot-loop prevention): **import** the ready-made solution (Quick start), or **build it step by step** (the full guide).

Quick start — import the bot

`solution/` holds an importable Power Platform solution (**AdvancedTeamsBot**) — the full bot this guide describes: it reads channel messages, includes thread history for replies, skips its own messages, and posts Ask Sage's answer back.

1. **Import** `solution/dist/AdvancedTeamsBot_1_0_0_1.zip`.
   Import mechanics (Solutions → Import solution, roles, etc.) are in the Importing a solution section of the Ask Sage Power Automate docs.
2. **Create / confirm a Microsoft Teams connection** when prompted (or under **… More** → **Connections**) — this authorizes the flow to read and post in Teams.
3. **Set your Team and Channel.** The flow ships with **placeholder** group/channel IDs. Open the imported flow and select your real **Team** and **Channel** on the **trigger**, on **List replies of a channel message**, and on **both** *Reply with a message in a channel* actions.
4. **Set your API key.** Replace the `YOUR_ASK_SAGE_API_TOKEN` placeholder in the `x-access-tokens` header of **both** HTTP actions (`Ask Sage HTTP Connection` and `Ask Sage HTTP Connection 1`) with your key — or wire it to Key Vault (see the Authentication section of the Ask Sage Power Automate docs).
5. **Turn the flow On** and post a message in the channel to test.

> Prefer to build it yourself (or want to understand every step)? The full step-by-step is in Overview and below.

Overview

> The step-by-step guide below builds the bot from scratch — the same flow as the importable AdvancedTeamsBot solution.

This guide walks you through creating a Microsoft Power Automate flow that integrates Ask Sage AI into a Microsoft Teams channel. The bot will:

- Respond to new messages in a Teams channel
- Maintain conversation history when users reply in a thread
- Avoid responding to its own messages (preventing infinite loops)

Prerequisites

Before you begin, ensure you have:

- [ ] Access to Microsoft Power Automate (https://make.powerautomate.com)
- [ ] Access to a Microsoft Teams channel where you want the bot to respond
- [ ] An Ask Sage API token (obtain from your Ask Sage account)
- [ ] Appropriate permissions to create flows and connect to Teams

---

Architecture Overview

The flow uses two paths to handle messages:

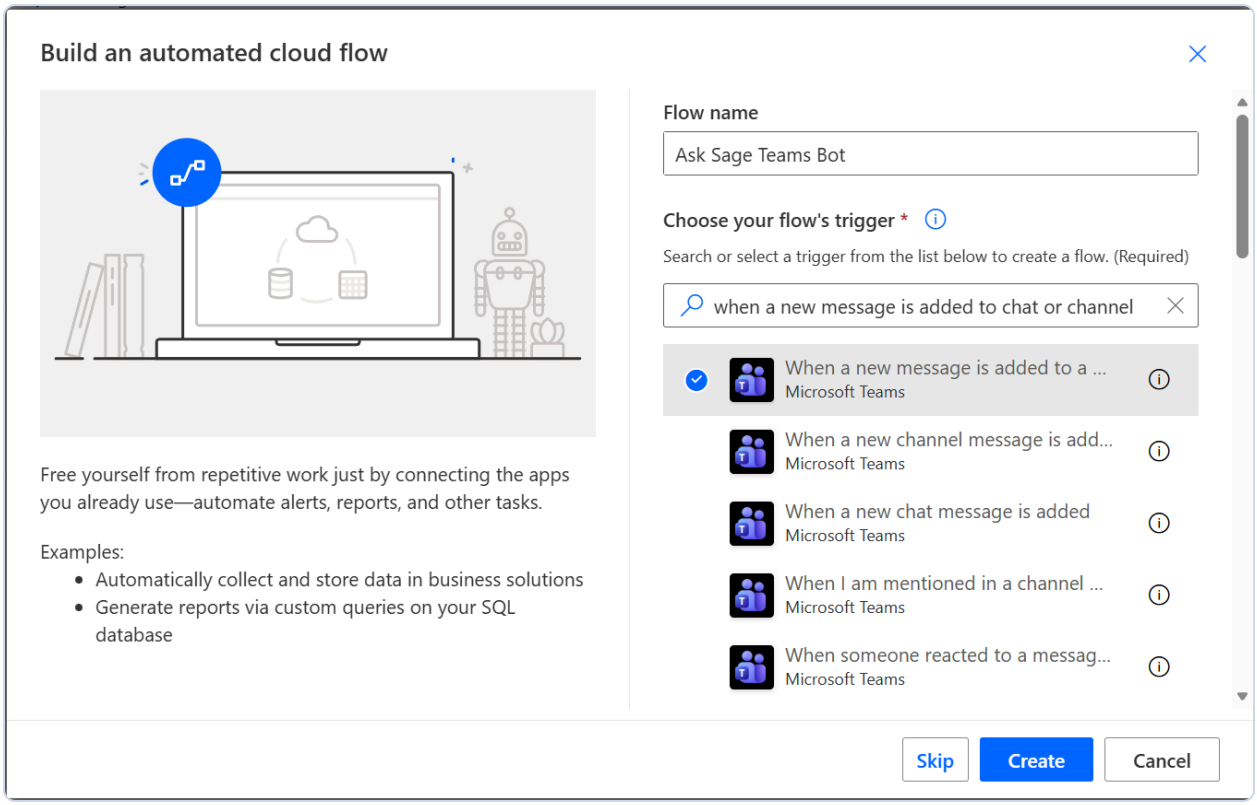
Scenario	Description
New Message (No History)	When a user starts a new conversation, the bot responds with just the current question
Reply Message (With History)	When a user replies in a thread, the bot includes previous conversation context

```
Trigger: New message in Teams channel
    ↓
List replies of channel message
    ↓
```

```
    ├── [Success] → Process with conversation history
    │
    └── [Failed] → Process without history (new thread)
```

Step 1: Create a New Flow

1. Navigate to **https://make.powerautomate.com**
2. Click **+ Create** in the left navigation
3. Select **Automated cloud flow**
4. Name your flow: `Ask Sage Teams Bot`
5. Search for the trigger: **"When a new message is added to a chat or channel"**
6. Select the Microsoft Teams trigger
7. Click **Create**



Step 2: Configure the Trigger

1. In the trigger action, configure the following:
    - **Message Type**: Channel
    - **Team**: Select your team from the dropdown

- **Channel**: Select the channel where the bot should respond

2. Click outside the trigger to save the configuration

> **Note:** If you don't see your channel in the dropdown list, you may need to enter the Channel ID manually.

How to Find Your Channel ID

If your channel doesn't appear in the dropdown:

1. **Get the channel link**:
   - In Teams (desktop or web), click the **three dots (...)** next to your channel name
   - Select **Get link to channel**
   - Copy the link
2. **The link will look like this**:

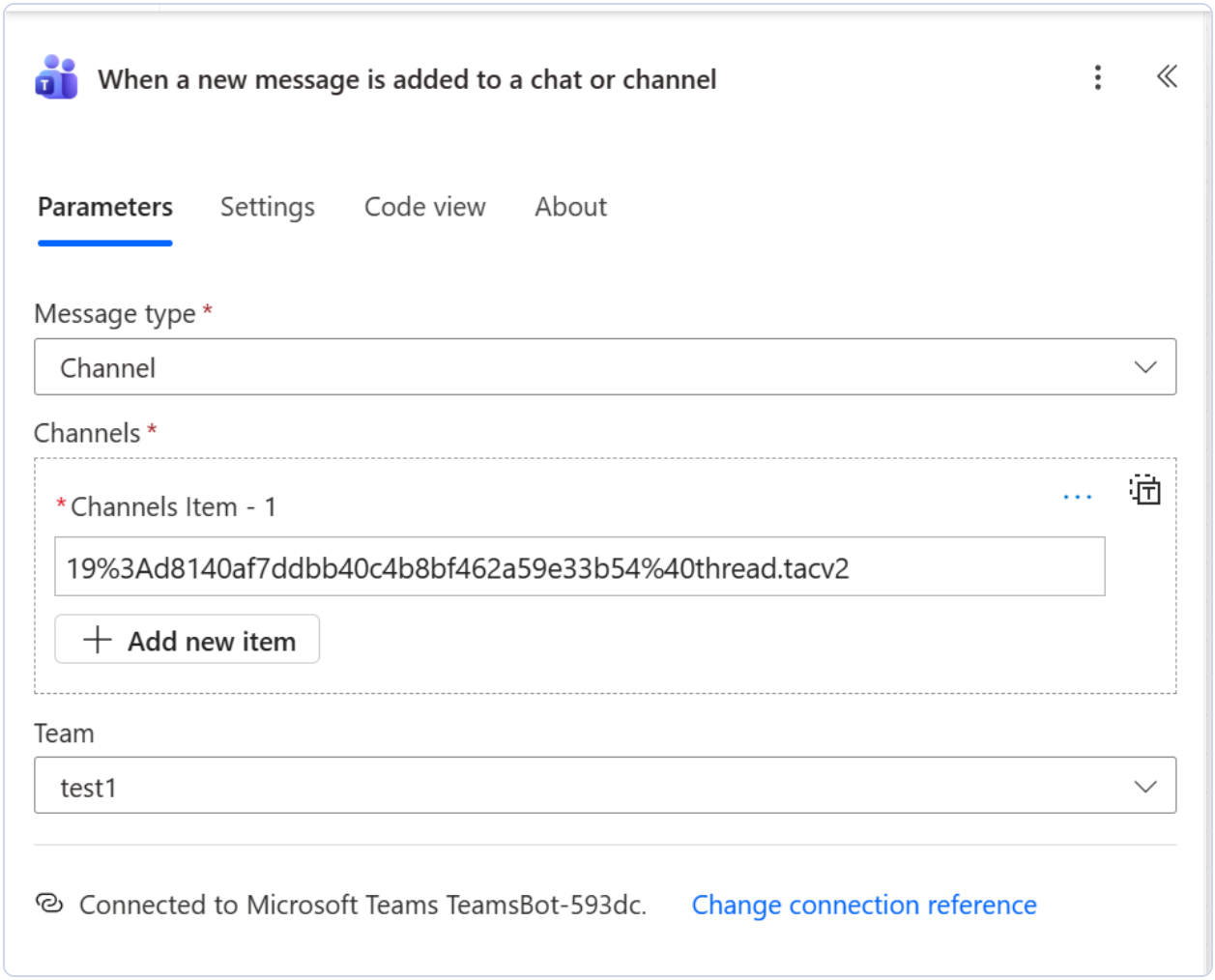
```
https://teams.microsoft.com/l/channel/19%3Aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa%40thread.tacv2/test%20channel%203?groupId=00000000-0000-0000-0000-000000000000&tenantId=11111111-1111-1111-1111-111111111111
```

3. **Extract the Channel ID**:
   - The Channel ID is the URL-encoded string between `/channel/` and the next `/`
   - In the example above, it's: `19%3Aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa%40thread.tacv2`
4. **Decode the URL**:
   - Replace `%3A` with `:`
   - Replace `%40` with `@`
   - Using the example above, the decoded Channel ID is:

```
19:aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa@thread.tacv2
```

5. **Enter the Channel ID** into Power Automate:
   - In the Channel field, click **Enter custom value** if available
   - Paste the **decoded** Channel ID (e.g., `19:aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa@thread.tacv2`)
   - **Important**: Use the decoded version, not the URL-encoded one

> **Quick Tip**: You can use an online URL decoder tool, or simply do the replacements manually (`%3A` → `:` and `%40` → `@`)



Step 3: Add “List Replies of a Channel Message” Action

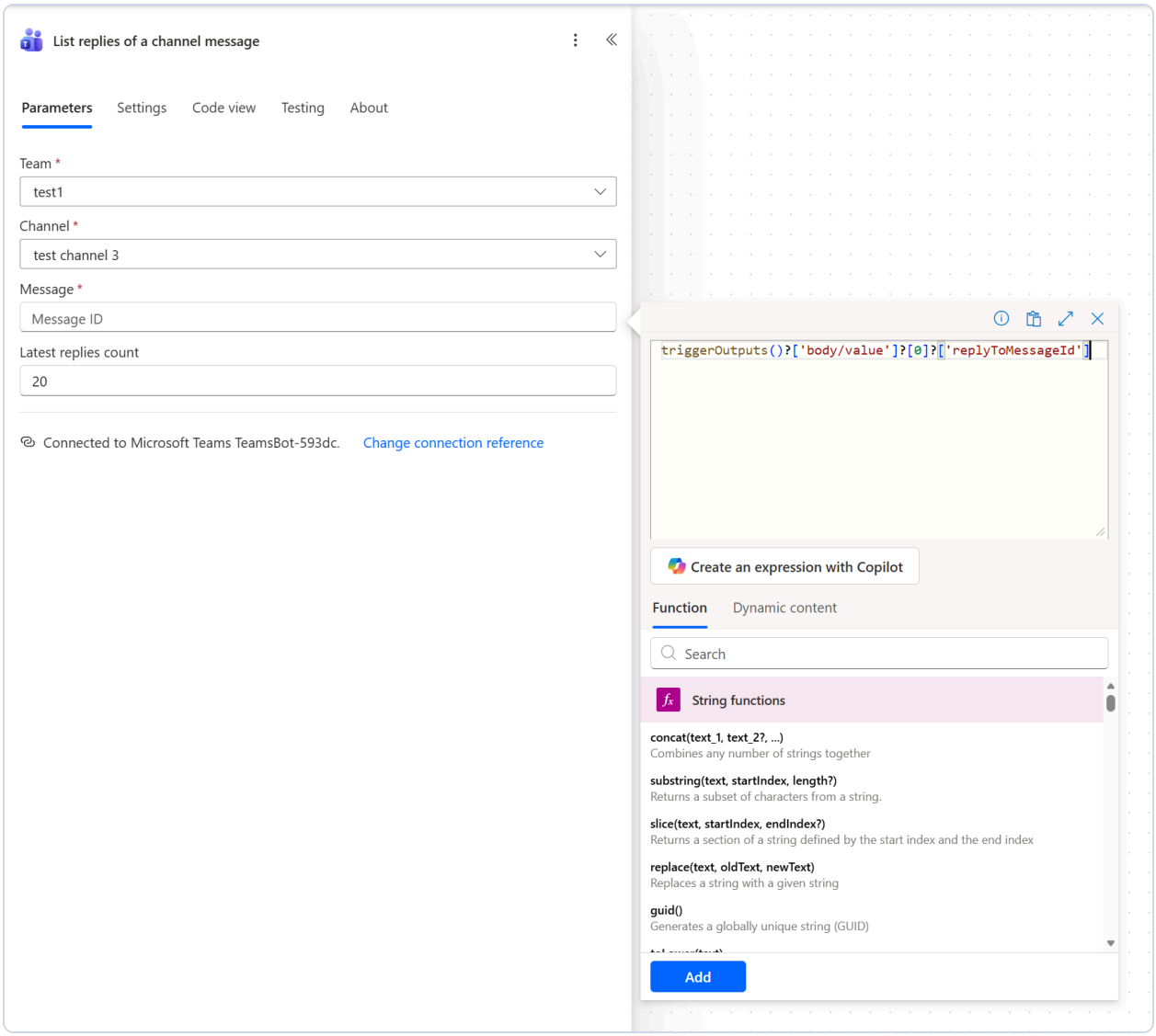
This action attempts to retrieve conversation history. If it fails (meaning it’s a new thread), we’ll handle that separately.

1. Click **+ New step**
2. Search for **“List replies of a channel message”**
3. Select the Microsoft Teams action
4. Configure the following:
   - **Team**: Select your team
   - **Channel**: Select your channel
   - **Message ID**: Click in the field, then select the **Expression** tab and enter:

     ```
     triggerOutputs()?['body/value']?[0]?['replyToMessageId']
     ```

   - **Top**: `20` (this limits the number of replies to retrieve)

5. Click **OK** to save the expression



Step 4: Add “Scope Success” for Messages with History

This scope handles messages that are replies in an existing thread.

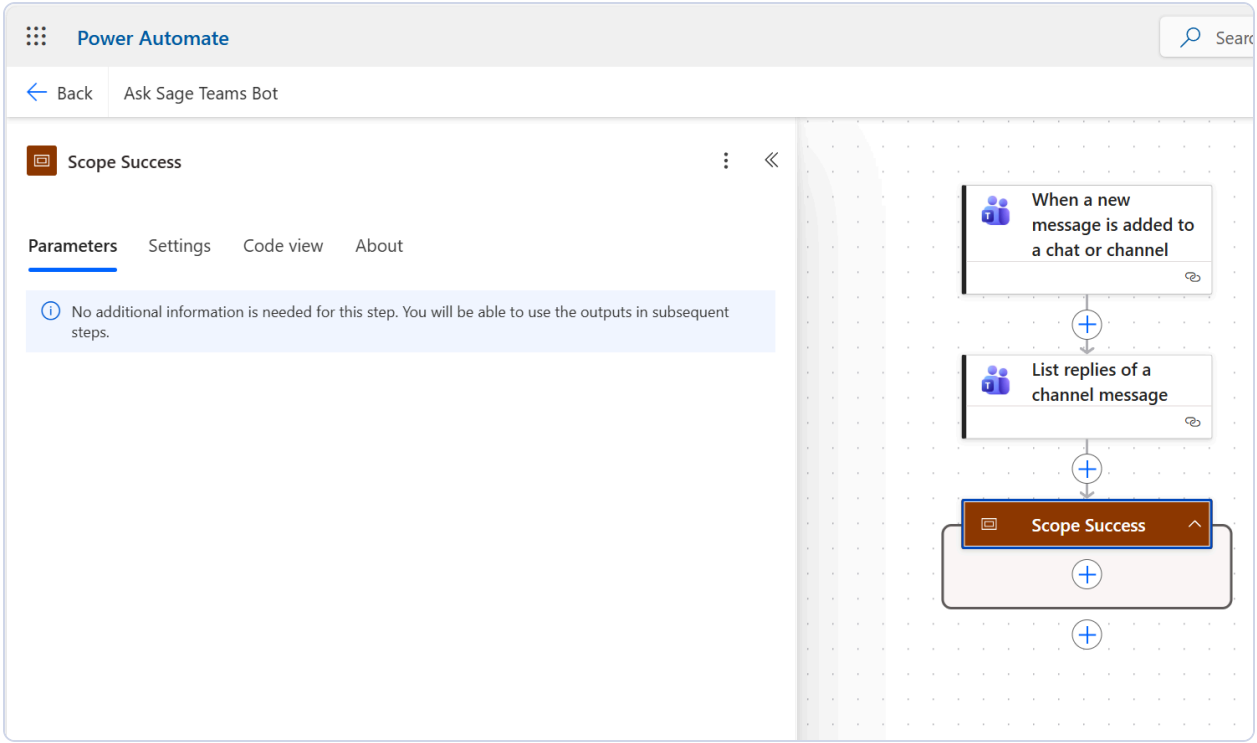
4.1: Add the Scope

1. Click **+ New step**
2. Search for **“Scope”**
3. Add the Scope action
4. Rename it to **“Scope Success”** by clicking on the title

4.1.1: Configure “Run After” for Scope Success

1. Click on the **Scope Success** action to select it
2. At the top of the action card, click on the **Settings** tab (next to Parameters)
3. Under **Run after**, you will see the **“List replies of a channel message”** action listed
4. Click directly on the **“List replies of a channel message”** text/action to expand the checkbox options
5. Ensure that **only** “is successful” is checked:
    - **Check** “is successful” ✓
    - Leave “has failed”, “is skipped”, and “has timed out” unchecked
6. The settings will save automatically

> **Note**: This ensures Scope Success only runs when “List replies” succeeds, creating parallel branches with Scope Failed.



4.2: Add “Select” Action Inside Scope Success

1. Click **Add an action** inside Scope Success
2. Search for **“Select”** (Data Operation)
3. Configure the **From** field:
    - Click in the **From** field
    - Go to **Expression** tab

- Enter:

```
body('List_replies_of_a_channel_message')?['value']
```

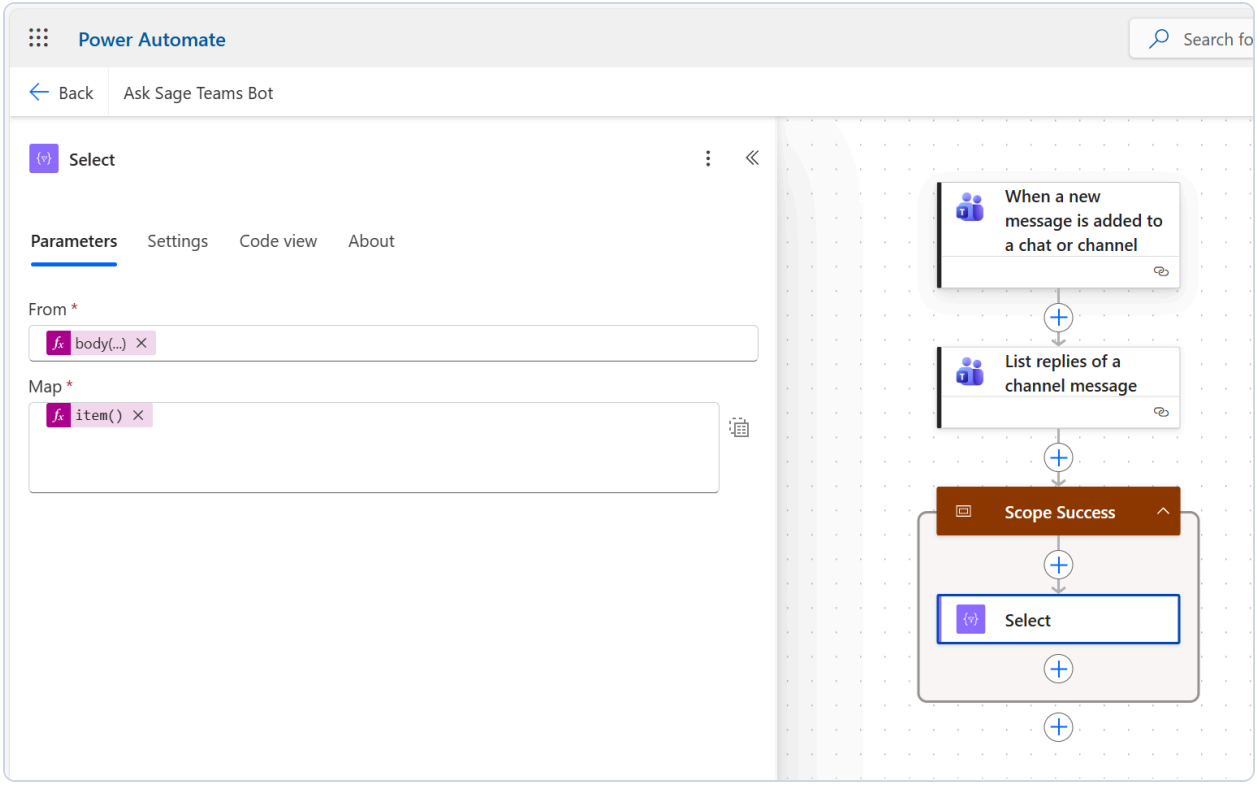
- Click **OK**

4. Configure the **Map** field:
   - Click on the **Map** field
   - **Important**: Click the **T** icon (text mode button) in the top-right corner of the Map field to switch to text mode
   - Once in text mode, click in the field
   - Go to **Expression** tab
   - Enter:

```
item()?['body']?['content']
```

   - Click **OK**

> **Note**: If you don't switch the Map field to text mode first, you won't be able to enter the expression properly. The text mode button looks like a "T" icon.



4.3: Add “Compose 1” Action for History

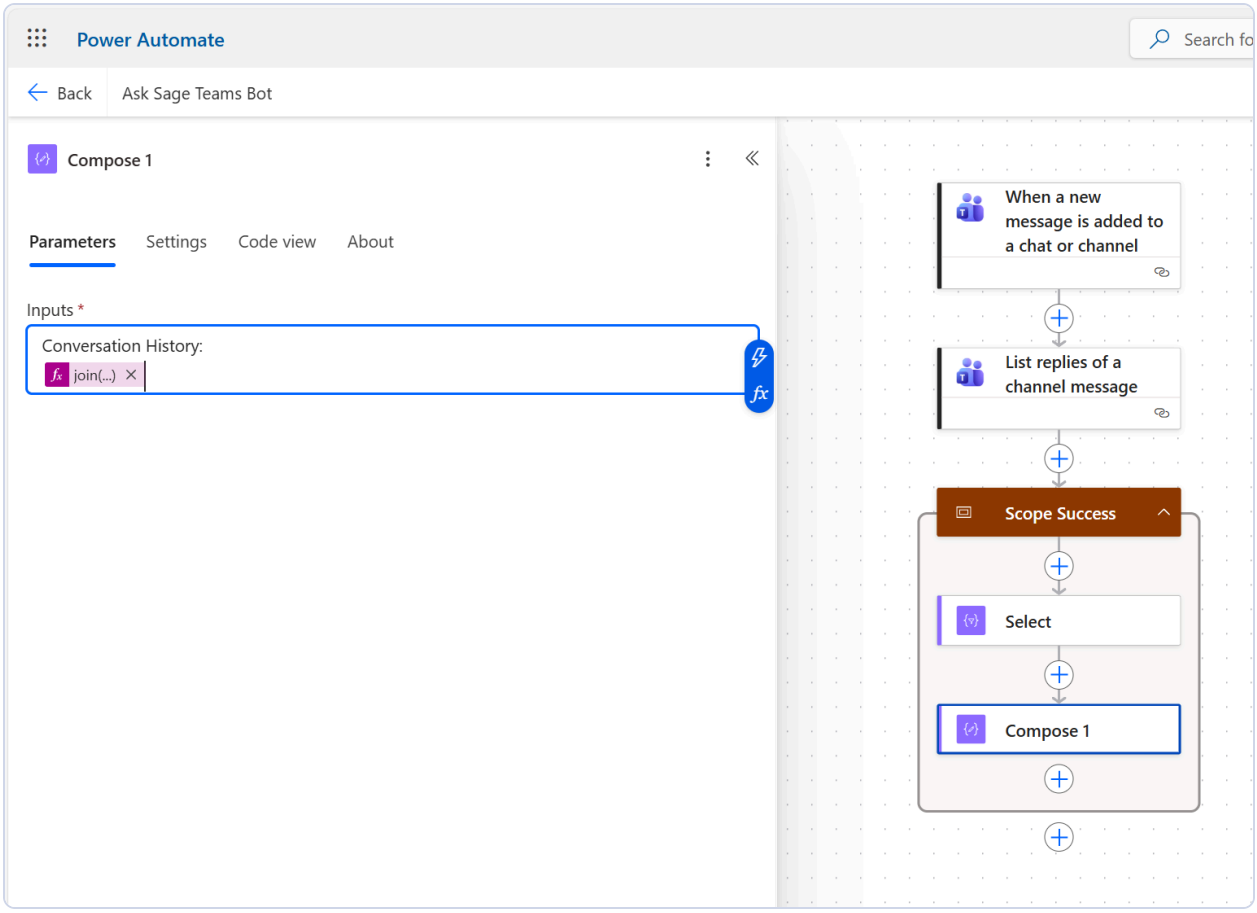
1. Click **+ New step** (still inside Scope Success)
2. Search for **“Compose”** (Data Operation)
3. Rename it to **“Compose 1”**
4. In the **Inputs** field, enter:

```
Conversation History:
@{join(body('Select'), '

')}
```

To add the expression:

- Type `Conversation History:` then press Enter
- Go to **Expression** tab and enter: `join(body('Select'), '\n\n')`
- Click **OK**

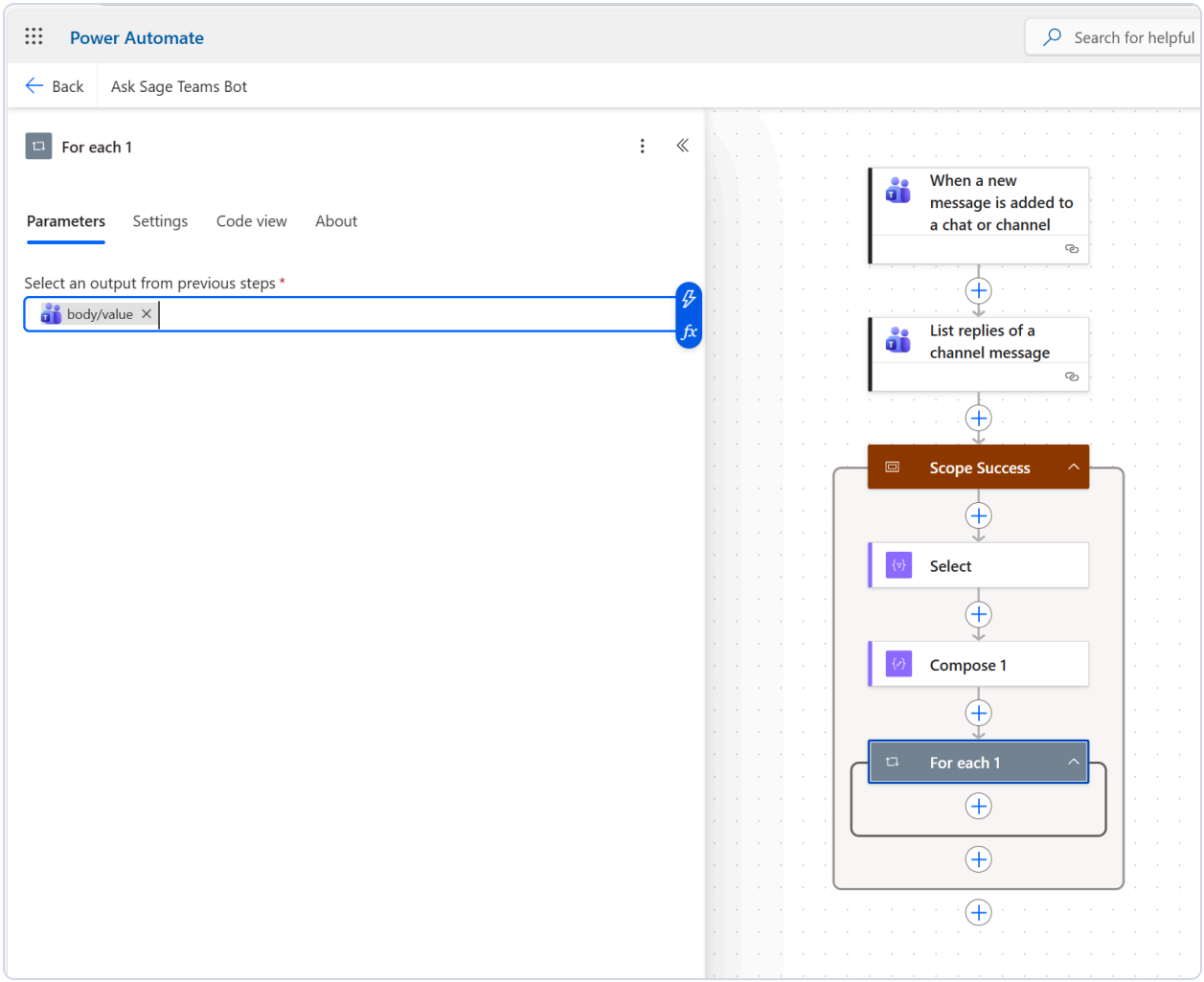


4.4: Add “For Each 1” Loop

1. Click **+ New step** inside Scope Success

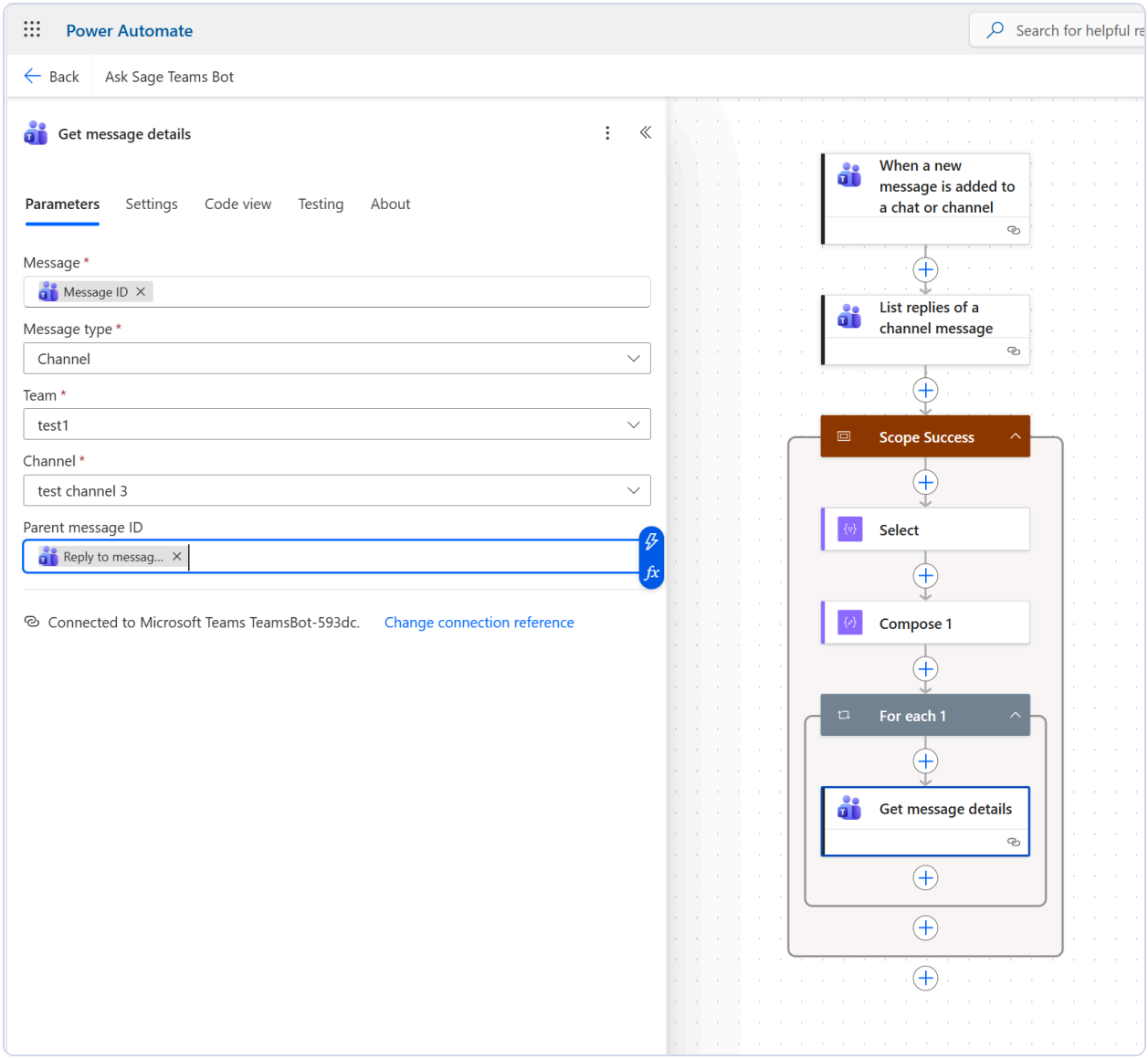
2. Search for **“Apply to each”**
3. Add the action and rename it to **“For each 1”**
4. For **“Select an output from previous steps”**, use expression:

```
triggerOutputs()?['body/value']
```



4.5: Add “Get Message Details” Inside the Loop

1. Inside the For each 1 loop, click **Add an action**
2. Search for **“Get message details”** (Microsoft Teams)
3. Configure:
    - **Message ID**: From Dynamic content, select **messageId** from the trigger
    - **Message Type**: Channel
    - **Team**: Select your team
    - **Channel**: Select your channel
    - **Parent Message ID**: From Dynamic content, select **replyToMessageId** from the trigger



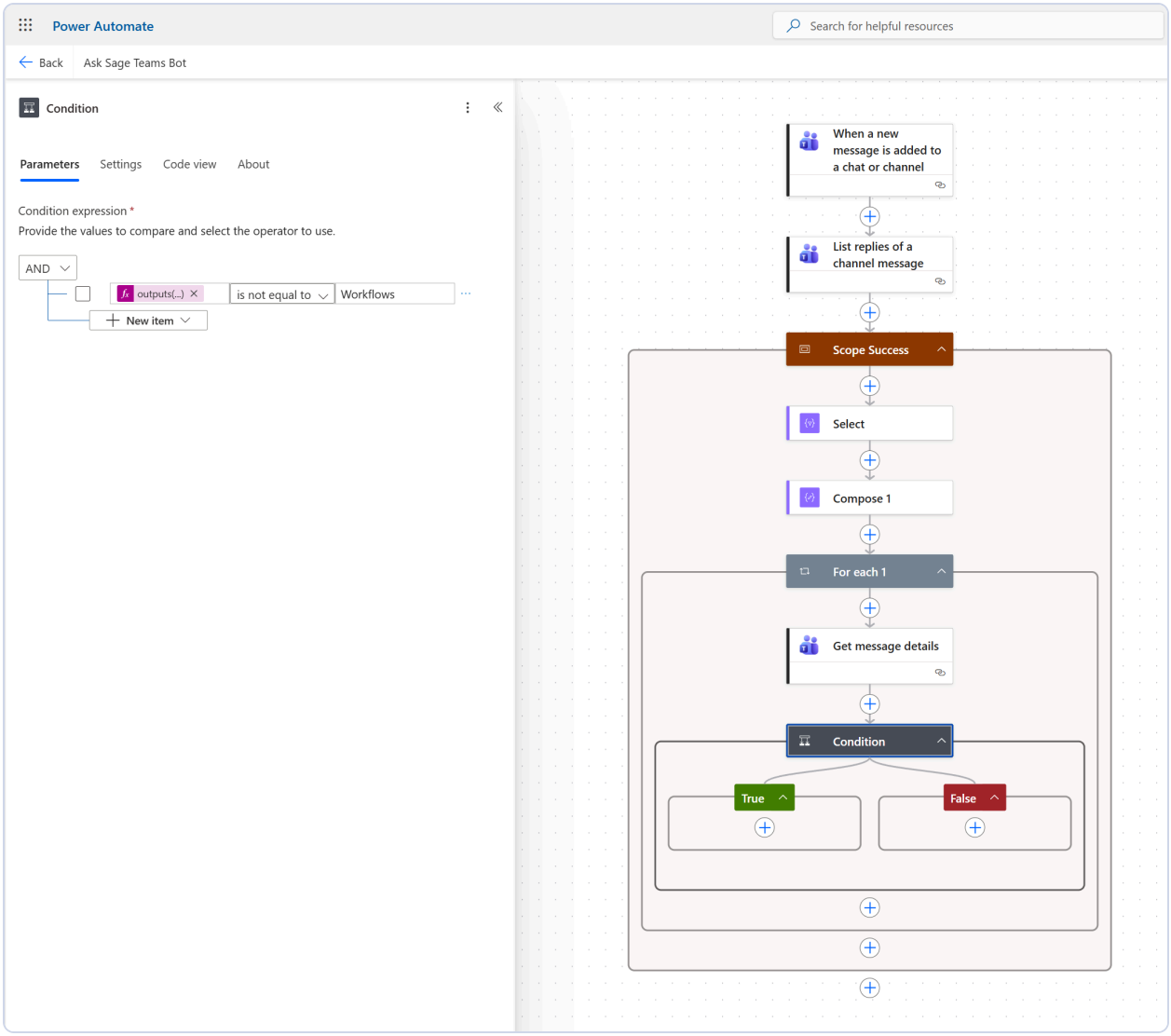
4.6: Add Condition to Filter Bot Messages

1. Click **Add an action** after Get message details (still inside For each 1)
2. Search for **"Condition"**
3. Configure the condition:
   - **Left value**: Go to **Expression** tab and enter:

     ```
     outputs('Get_message_details')?['body/from/application/displayName']
     ```

   - **Operator**: `is not equal to`
   - **Right value**: `Workflows`

> **Note:** This condition ensures the bot doesn't respond to its own messages, preventing infinite loops.



4.7: Add Actions Inside the "True" Branch

Inside the **True** branch of the condition (meaning the message is NOT from the bot):

4.7.1: Add "Compose 2" for Full Message

1. Click **Add an action** inside the True branch
2. Search for and add **"Compose"** action (Data Operation)
3. Power Automate will automatically name it **"Compose 2"** - keep this name as-is
4. In the **Inputs** field, enter:

   ```
   ```

Previous conversation:
@{outputs('Compose_1')}

Current question: @{outputs('Get_message_details')?['body/body/plainTextContent']}

```
![Compose with history and current question](file:///tmp/power-automate-
integration/examples/03-teams-bot/screenshots/10-compose-full-message.png)

#### 4.7.2: Add "Ask Sage HTTP Connection" for API Call

1. Click **Add an action**
2. Search for **"HTTP"**
3. Rename the action to **"Ask Sage HTTP Connection"**
4. Configure:
   - **Method**: `POST`
   - **URI**: `https://api.asksage.ai/server/query`
   - **Headers**: Click **Show advanced options** and add:
     | Key | Value |
     |-----|-------|
     | `x-access-tokens` | `YOUR_ASK_SAGE_API_TOKEN` |
     | `Content-Type` | `application/json` |
   - **Body**:
     ```json
     {
       "message": "@{outputs('Compose_2')}",
       "user_id": "@{outputs('Get_message_details')?['body/from/user/id']}"
     }
     ```

> **Important:** Replace `YOUR_ASK_SAGE_API_TOKEN` with your actual Ask Sage API token.

![HTTP action configured for Ask Sage API](file:///tmp/power-automate-
integration/examples/03-teams-bot/screenshots/11-http-ask-sage.png)

4.7.3: Add "Reply with a Message in a Channel" Action

1. Click **Add an action**
2. Search for **"Reply with a message in a channel"** (Microsoft Teams)
3. Configure:
   - **Poster**: Flow bot
   - **Location**: Channel
   - **Team**: Select your team
   - **Channel**: Select your channel
   - **Parent Message ID**: From Dynamic content, select **replyToMessageId**
   - **Message Body**: Go to **Expression** tab and enter:
     ```
     body('Ask_Sage_HTTP_Connection')?['message']
```
```

```

![Reply action configured](file:///tmp/power-automate-integration/examples/03-teams-bot/screenshots/12-reply-action.png)

---

## Step 5: Add "Scope Failed" for New Messages (No History)

This scope handles messages that start a new thread (no conversation history).

### 5.1: Add the Scope

1. Click **+ New step** (outside of Scope Success, at the same level)
2. Search for **"Scope"**
3. Add the Scope action
4. Rename it to **"Scope Failed"**

### 5.2: Configure "Run After" Settings

This is **critical** - we want Scope Failed to run only when "List replies" fails.

1. Click on the **Scope Failed** action to select it
2. At the top of the action card, click on the **Settings** tab (next to Parameters)
3. Under **Run after**, you will see the **"List replies of a channel message"** action listed
4. Click directly on the **"List replies of a channel message"** text/action to expand the checkbox options
5. **Important**: Configure the checkboxes in this specific order:
   - **First, check** "has failed" ✓
   - **Then, uncheck** "is successful" (remove the checkmark)

   > **Note**: You must check "has failed" BEFORE unchecking "is successful", otherwise Power Automate may not allow you to remove the "is successful" checkmark.

6. Leave "is skipped" and "has timed out" unchecked
7. If you see **"Scope Success"** listed, ensure it is NOT selected or removed
8. The settings will save automatically

> **Why this matters:** This configuration ensures that Scope Failed only runs when the "List replies" action fails, which happens when there's no conversation history (i.e., it's a new thread).

![Configure run after settings for Scope Failed](file:///tmp/power-automate-integration/examples/03-teams-bot/screenshots/13-run-after-settings.png)

### 5.3: Add "For Each 2" Loop Inside Scope Failed

1. Click **Add an action** inside Scope Failed
2. Search for **"Apply to each"**
```

```
3. Add the action and rename it to **"For each 2"**
4. For **"Select an output from previous steps"**, use expression:
```

triggerOutputs()?['body/value']

```
### 5.4: Add "Get Message Details 2" Inside the Loop

1. Inside the For each 2 loop, add **"Get message details"**
2. Configure the same as before:
   - **Message ID**: messageId from trigger
   - **Message Type**: Channel
   - **Team**: Your team
   - **Channel**: Your channel
   - **Parent Message ID**: replyToMessageId from trigger

3. Rename this to **"Get message details 2"**

![Get message details 2 configured](file:///tmp/power-automate-integration/examples/03-teams-bot/screenshots/14-get-message-details-2.png)

### 5.5: Add "Condition 1" to Filter Bot Messages

1. Add a **Condition** after Get message details 2
2. Rename it to **"Condition 1"**
3. Configure:
   - **Left value**: Expression:
     ```
     outputs('Get_message_details_2')?['body/from/application/displayName']
     ```
   - **Operator**: `is not equal to`
   - **Right value**: `Workflows`

![Condition 1 configured](file:///tmp/power-automate-integration/examples/03-teams-bot/screenshots/15-condition-1.png)

### 5.6: Add Actions Inside the "True" Branch

#### 5.6.1: Add "Compose 3" for Current Question Only

1. Add **"Compose"** inside the True branch
2. Power Automate will automatically name it **"Compose 3"** - keep this name as-is
3. In **Inputs**, enter:
```

Current question: @{outputs('Get_message_details_2')?['body/body/plainTextContent']}

````
> **Note:** Unlike Scope Success, this does not include conversation history since this is a new thread.

![Compose 3 with current question only](file:///tmp/power-automate-integration/examples/03-teams-bot/screenshots/16-compose-3.png)

#### 5.6.2: Add "Ask Sage HTTP Connection 1" for API Call

1. Add **"HTTP"** action
2. Rename to **"Ask Sage HTTP Connection 1"**
3. Configure:
   - **Method**: `POST`
   - **URI**: `https://api.asksage.ai/server/query`
   - **Headers**: Click **Show advanced options** and add:
     | Key | Value |
     |-----|-------|
     | `x-access-tokens` | `YOUR_ASK_SAGE_API_TOKEN` |
     | `Content-Type` | `application/json` |
   - **Body**:
     ```json
     {
       "message": "@{outputs('Compose_3')}",
       "user_id": "@{outputs('Get_message_details_2')?['body/from/user/id']}"
     }
     ```

> **Important:** Replace `YOUR_ASK_SAGE_API_TOKEN` with your actual Ask Sage API token.

![Ask Sage HTTP Connection 1 configured](file:///tmp/power-automate-integration/examples/03-teams-bot/screenshots/17-http-ask-sage-1.png)

#### 5.6.3: Add "Reply with a Message in a Channel 1" Action

1. Add **"Reply with a message in a channel"**
2. Configure:
   - **Poster**: Flow bot
   - **Location**: Channel
   - **Team**: Your team
   - **Channel**: Your channel
   - **Parent Message ID**: replyToMessageId from trigger
   - **Message Body**: Expression:
     ```
     body('Ask_Sage_HTTP_Connection_1')?['message']
     ```

![Reply with a message in a channel 1 configured](file:///tmp/power-automate-integration/examples/03-teams-bot/screenshots/18-reply-action-1.png)
````

```
---

## Step 6: Save and Test

### 6.1: Save the Flow

1. Click **Save** in the top right corner
2. Wait for the save to complete
3. If you receive any errors, review the error message and correct the referenced
action names

### 6.2: Turn On the Flow

1. Ensure the flow is turned **On** (toggle in the top right)
2. The flow will now trigger automatically when new messages are posted

### 6.3: Test the Flow

#### Test 1: New Message (No History)

1. Go to your Teams channel
2. Start a **new conversation** (not a reply)
3. Type a question, for example: `What is the capital of France?`
4. Wait for the bot to respond (may take up to 1 minute on first run)
5. Verify the bot responds with the answer

**Expected Flow Path:** Trigger → List replies (fails) → Scope Failed → Response

#### Test 2: Reply Message (With History)

1. In the same thread where the bot responded
2. **Reply** to the conversation
3. Type a follow-up question, for example: `What about Germany?`
4. The bot should respond with context from the previous conversation

**Expected Flow Path:** Trigger → List replies (succeeds) → Scope Success → Response
with history

![Example conversation in Teams showing bot responses](file:///tmp/power-automate-
integration/examples/03-teams-bot/screenshots/19-example-conversation.png)

---

## Troubleshooting

### Issue: Bot responds with empty messages

**Cause**: Invalid or expired API token, or incorrect API response parsing

**Solution**:
```

```
1. Verify your API token is correct and active in Ask Sage
2. Update the `x-access-tokens` header in **both** HTTP actions:
   - Ask Sage HTTP Connection (in Scope Success)
   - Ask Sage HTTP Connection 1 (in Scope Failed)
3. Check the flow run history to see the actual API response
4. Verify the response structure matches the expression
`body('Ask_Sage_HTTP_Connection')?['message']`

### Issue: Flow fails with "ActionFailed" error

**Cause**: Expression references wrong action name or incorrect expression syntax

**Solution**:
1. Go to **Run history** in the flow
2. Identify the failed action
3. Check that all expressions reference the correct action names:
   - Actions in Scope Success: `Get_message_details`, `Ask_Sage_HTTP_Connection`,
`Compose_1`, `Compose_2`
   - Actions in Scope Failed: `Get_message_details_2`, `Ask_Sage_HTTP_Connection_1`,
`Compose_3`
4. Verify all expressions use correct syntax (check for typos in property names)

### Issue: Bot responds to its own messages (infinite loop)

**Cause**: Condition not properly filtering bot messages

**Solution**:
1. Verify both Conditions check for `is not equal to` `Workflows`
2. Ensure the expression references the correct Get_message_details action:
   - In Scope Success: `outputs('Get_message_details')?
['body/from/application/displayName']`
   - In Scope Failed: `outputs('Get_message_details_2')?
['body/from/application/displayName']`
3. If your bot uses a different display name, update the condition value

### Issue: Scope Failed never runs

**Cause**: "Run after" settings not properly configured

**Solution**:
1. Click on **Scope Failed**
2. Select the three dots menu → **"Configure run after"**
3. Ensure "List replies of a channel message" is configured to run on:
   - ✓ has failed
   - ✓ is skipped
   - ✓ has timed out
4. Ensure "is successful" is **unchecked**

### Issue: Flow doesn't trigger at all
```

```
**Cause**: Trigger not properly configured or flow is turned off

**Solution**:
1. Verify the flow is turned **On** (check toggle in top right)
2. Verify the correct Team and Channel are selected in the trigger
3. Ensure you have permissions to the channel
4. Try posting a message with an @mention to ensure Teams recognizes it

### Issue: "List replies" always fails

**Cause**: Expression for Message ID may be incorrect, or the message structure changed

**Solution**:
1. Check the expression in "List replies of a channel message":
```

triggerOutputs()?[‘body/value’]?[0]?[‘replyToMessageId’]

```
2. This is expected behavior for new threads - Scope Failed should handle it
3. If it fails even for replies, check the trigger output structure in run history

---

## Flow Summary

| Component | Purpose |
|-----------|---------|
| **Trigger** | Fires when a new message is posted in the channel |
| **List replies** | Attempts to get conversation history for threaded messages |
| **Scope Success** | Handles replies with conversation history (when List replies succeeds) |
| **Scope Failed** | Handles new threads without history (when List replies fails) |
| **Select & Compose** | Formats conversation history for context |
| **Condition** | Filters out bot messages to prevent infinite loops |
| **Get message details** | Retrieves full message content and sender information |
| **HTTP Action** | Sends message and context to Ask Sage API |
| **Reply Action** | Posts bot response back to Teams thread |

---

## Advanced Configuration

### Limiting Conversation History

The flow retrieves the last **20 replies** by default. To change this:

1. Edit the **"List replies of a channel message"** action
2. Change the **Top** parameter to your desired number (e.g., `10` or `50`)
```

```
**Considerations:**
- More history provides better context but increases API payload size
- Ask Sage API may have token limits for message length
- Performance may degrade with very large histories

### Multiple Channels

To deploy the bot to multiple channels:

1. **Option A: Duplicate the flow**
   - Save the current flow as a template
   - Create new flows for each channel
   - Update the Team/Channel in each flow

2. **Option B: Single flow with dynamic channel selection**
   - Modify the trigger to listen to multiple channels
   - This requires more complex logic and is not covered in this guide

### Customizing Bot Responses

You can modify what gets sent to Ask Sage:

1. Edit the **Compose** actions (Compose and Compose 2)
2. Change the message format, for example:
```

You are a helpful assistant for [Your Company].

Previous conversation:
@{outputs('Compose_1')}

Current question: @{outputs('Get_message_details')?['body/body/plainTextContent']}

Please provide a concise, professional response.

```

### Adding Error Notifications

To receive notifications when the flow fails:

1. Add a **Scope** at the end of your flow
2. Configure "Run after" to run when previous actions fail
3. Inside the scope, add **"Send me an email notification"**
4. Include relevant error information from previous steps

---

## Security Considerations
```

```
### API Token Security

- **Never share your API token** in screenshots or documentation
- **Store tokens securely**: Consider using Azure Key Vault for production environments
- **Rotate tokens regularly**: Update your API tokens periodically
- **Limit token permissions**: Use tokens with minimum required permissions

### Access Control

- **Flow permissions**: Ensure only authorized users can modify the flow
- **Channel access**: The bot will respond to all messages in the configured channel
- **User data**: The flow passes user IDs to Ask Sage - ensure this complies with your
privacy policy

### Data Privacy

- **Conversation logging**: Be aware that conversation history is sent to Ask Sage
- **Compliance**: Ensure your use case complies with your organization's data policies
- **Channel selection**: Avoid deploying to channels with sensitive information without
proper review

### Monitoring

- **Regular monitoring**: Check flow run history regularly for failures or unusual
patterns
- **Rate limiting**: Be aware of Power Automate and Ask Sage API rate limits
- **Cost management**: Monitor Power Automate usage to avoid unexpected costs

---

## Best Practices

### Testing

1. **Test in a development channel first** before deploying to production channels
2. **Test both scenarios**:
   - New conversation threads (Scope Failed path)
   - Replies in existing threads (Scope Success path)
3. **Test edge cases**:
   - Very long messages
   - Messages with special characters
   - Rapid successive messages

### Maintenance

1. **Document your API token location** for easy updates
2. **Keep a backup** of your flow by exporting it regularly
3. **Monitor run history** for patterns of failures
4. **Update the guide** when you make changes to the flow
```

```
### Performance

1. **Limit history retrieval**: Don't retrieve more replies than necessary
2. **Consider timeout settings**: Adjust if API responses are slow
3. **Monitor flow performance**: Check average run duration in analytics

---

## API Reference

### Ask Sage API Endpoint

**URL**: `https://api.asksage.ai/server/query`

**Method**: `POST`

**Headers**:
```

x-access-tokens: YOUR_API_TOKEN
Content-Type: application/json

```
**Request Body**:
```json
{
  "message": "The user's question with optional context",
  "user_id": "unique_user_identifier"
}
```

Response:

```
{
  "message": "The AI-generated response"
}
```

Frequently Asked Questions

Q: Can the bot respond to direct messages?

A: This flow is configured for channel messages only. To support direct messages, you would need to modify the trigger to “When a new message is added to a chat or channel” with Message Type set to “Chat”.
```

Q: How do I stop the bot from responding?

A: Simply turn off the flow using the toggle in the top right corner of the flow editor.

Q: Can I use a different AI provider?

A: Yes, you would need to modify the HTTP actions to call your AI provider's API endpoint and adjust the request/response format accordingly.

Q: Why does the bot take so long to respond?

A: Response time depends on:

- Power Automate processing time
- Ask Sage API response time
- Network latency

First-time runs may be slower. If consistent slowness occurs, contact Ask Sage support.

Q: Can I customize the bot's display name?

A: The bot uses "Flow bot" as configured in the Reply actions. This is a limitation of Power Automate's Teams connector. For custom bot names, consider using Azure Bot Service instead.

Q: How many messages can the bot handle?

A: The flow triggers once per message. Power Automate has usage limits based on your license. Check your organization's Power Automate capacity.

Additional Resources

- **Power Automate Documentation**: https://docs.microsoft.com/en-us/power-automate/
- **Microsoft Teams Connector Reference**: https://docs.microsoft.com/en-us/connectors/teams/
- **Power Automate Expression Reference**: https://docs.microsoft.com/en-us/power-automate/use-expressions-in-conditions
- **Ask Sage API Documentation**: https://docs.asksage.ai/docs/v2/api-documentation/api-endpoints.html
- **Authentication & secure key storage**: the Authentication section of the Ask Sage Power Automate docs

---

Appendix: Expression Reference

Key Expressions Used in This Flow

Get Reply-To Message ID

```
triggerOutputs()?['body/value']?[0]?['replyToMessageId']
```

Retrieves the parent message ID if the current message is a reply.

Get Trigger Array

```
triggerOutputs()?['body/value']
```

Gets the array of messages from the trigger.

Get Message Content from Replies

```
body('List_replies_of_a_channel_message')?['value']
```

Retrieves the array of reply messages.

Map Reply Content

```
item()?['body']?['content']
```

Extracts the content from each reply in the Select action.

Join Conversation History

```
join(body('Select'), '\n\n')
```

Combines all reply contents with double line breaks.

Get Message Plain Text

```
outputs('Get_message_details')?['body/body/plainTextContent']
```

Extracts plain text content from a message.

Get Sender Application Name

```
outputs('Get_message_details')?['body/from/application/displayName']
```

Gets the application name of the message sender (used to filter bot messages).

Get API Response Message

```
body('Ask_Sage_HTTP_Connection')?['message']
```

Extracts the response message from Ask Sage API.

Changelog

Version 1.0 (January 2026)

- Initial release
- Support for threaded conversations with history
- Support for new conversations without history
- Bot message filtering to prevent loops

Support and Feedback

If you encounter issues not covered in this guide:

- **Ask Sage Support**: support@asksage.ai
- **Power Automate Community**: https://powerusers.microsoft.com/

For feedback or suggestions to improve this guide, open an issue or PR in this repo, or email support@asksage.ai.

Guide Version: 1.0 · Part of the Ask Sage × Power Automate examples.
