



POWER UP!

A composite image featuring four individuals. In the background, a woman with short grey hair and a man with a mustache are looking upwards. In the foreground, two men are looking down at a laptop screen. The word "POWER UP!" is overlaid in large, colorful letters across the center.

WELCOME
TO

VIENNA²⁰
₂₅



JOIN THE
CONVERSATION

....
#EPPC25



Enterprise-Ready Flow Management: Error Handling, Monitoring, and Optimization

Shahid Mahmood Hussain

Lead Technical Advisor, Power Platform
Abakion A/S, Denmark



Enterprise-Ready Flow Management: Error Handling, Monitoring, and Optimization

Michael Nielsen

Senior Developer,
Abakion A/S, Denmark

Let us try to imagine a scenario together...



Friday afternoon, just after
a successful deployment...



Successful deployment

Friday afternoon and
successful deployment has
been completed



But then...



⚠ Critical Issue occur

“A critical business process flow has failed”

What would you do in this situation? 🤔

Chaos ensues!



Everyone's scrambling to find the root cause.

- 📄 The logs are incomplete
- ⚠️ Error handling? Patchy at best.
- 🔍 Monitoring?
- 👁️ Practically nonexistent

The aftermath...



Catch Errors Before They Cause Chaos

Better error handling would've caught this early

The Lesson:

It's not enough for flows to work—they need to be built for the scale, security, and resilience that enterprises demand.

ROBUST FLOWS



ERROR HANDLING



MONITORING



Error management

Logging

The aftermath...

One small failure,
and everything

Notification

Proactive monitoring

In this session, we'll explore how to elevate your flow management game:

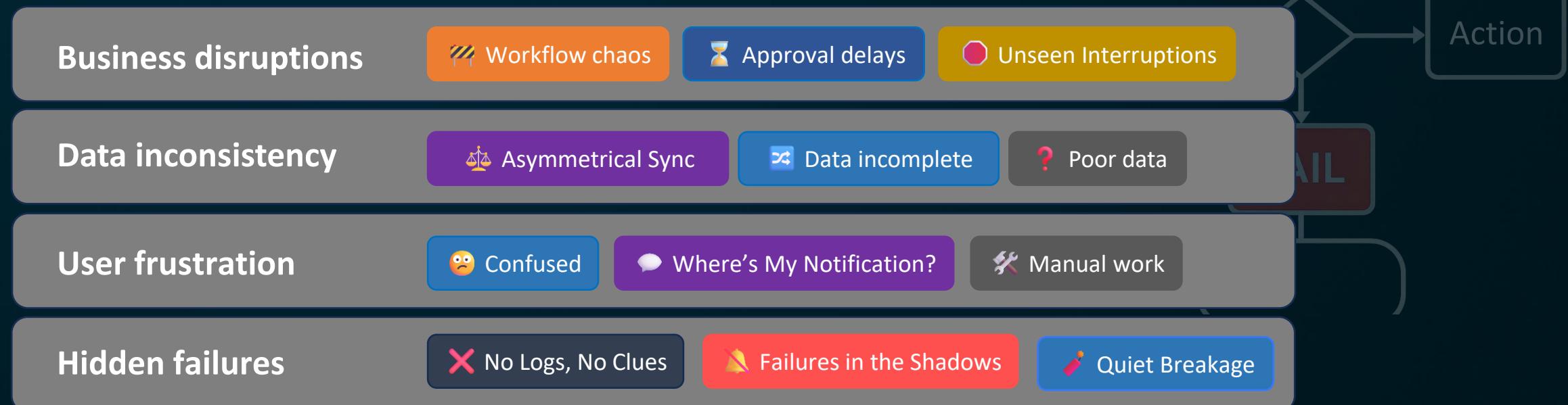
- 🔍 How to design robust error handling so your flows can heal themselves—or at least tell you what went wrong.
- 📊 How to implement real-time monitoring that keeps you ahead of failures instead of reacting after the damage is done.
- ⚙️ And finally, how to optimize your flows for performance and scalability, so they run smoothly no matter how complex your business grows.
Let's dive in and turn our flows into the reliable backbone of the modern enterprise.

Quick Raise of hands



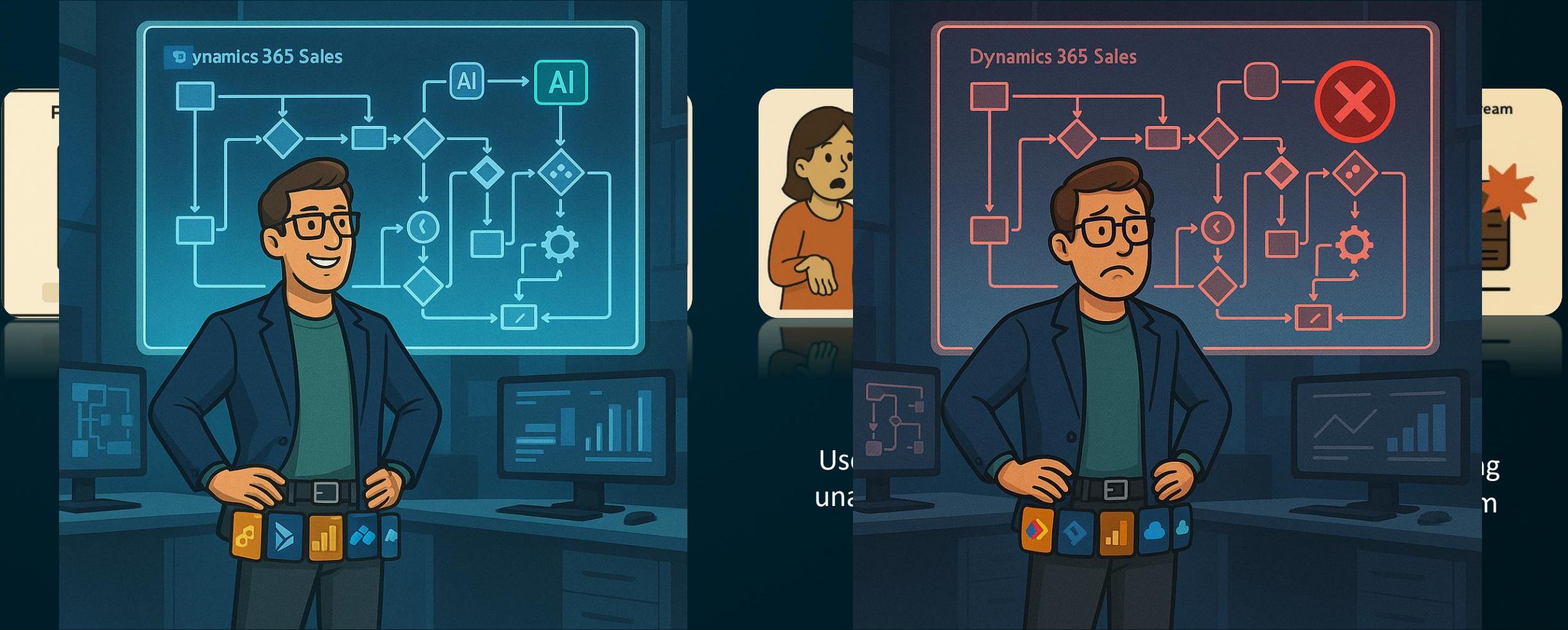
Why Error Handling Matters

- 2016 Power Automate GA
- Power Automate executes business-critical processes (ordre processing, Approval flow, moving data, Integrating with API)
- No process is completely error-free(API failures, data changes, timeouts)
- Unhandled errors lead to:



Let's imagine you've built a great flow...

What Happens Without Error Handling?



Common Pitfalls

Even the best flows can fail if you overlook common pitfalls.

Avoid failure by spotting these common mistakes early:

- No Configure Run After Conditions 
- Assuming APIs or Connectors Always Succeed 
- No Retry Policies Configured 
- No Centralized Monitoring or Alerting 



Solution

Use Try ➔ Catch ➔ Finally patterns using Scope actions and Configure run after settings

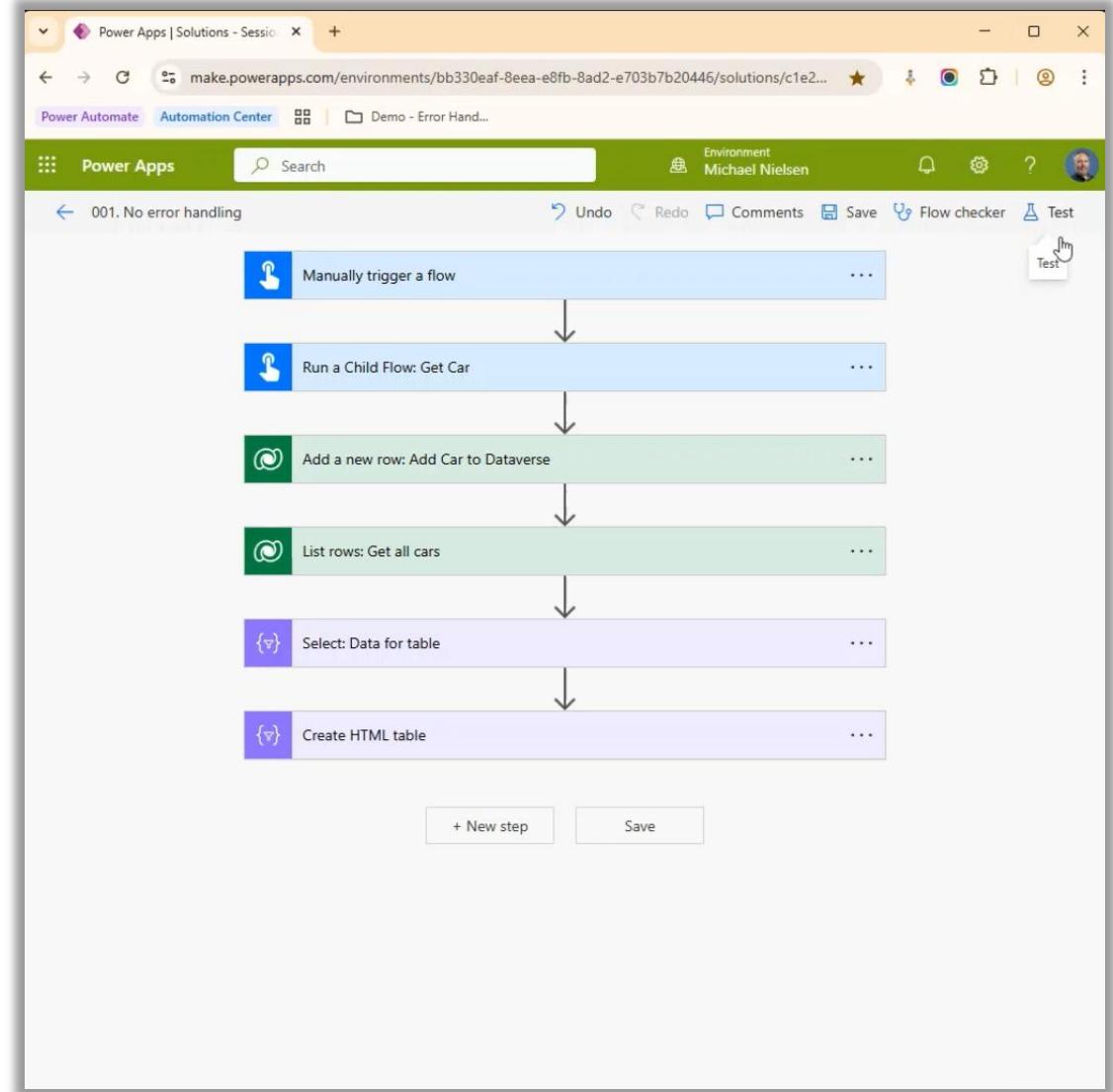
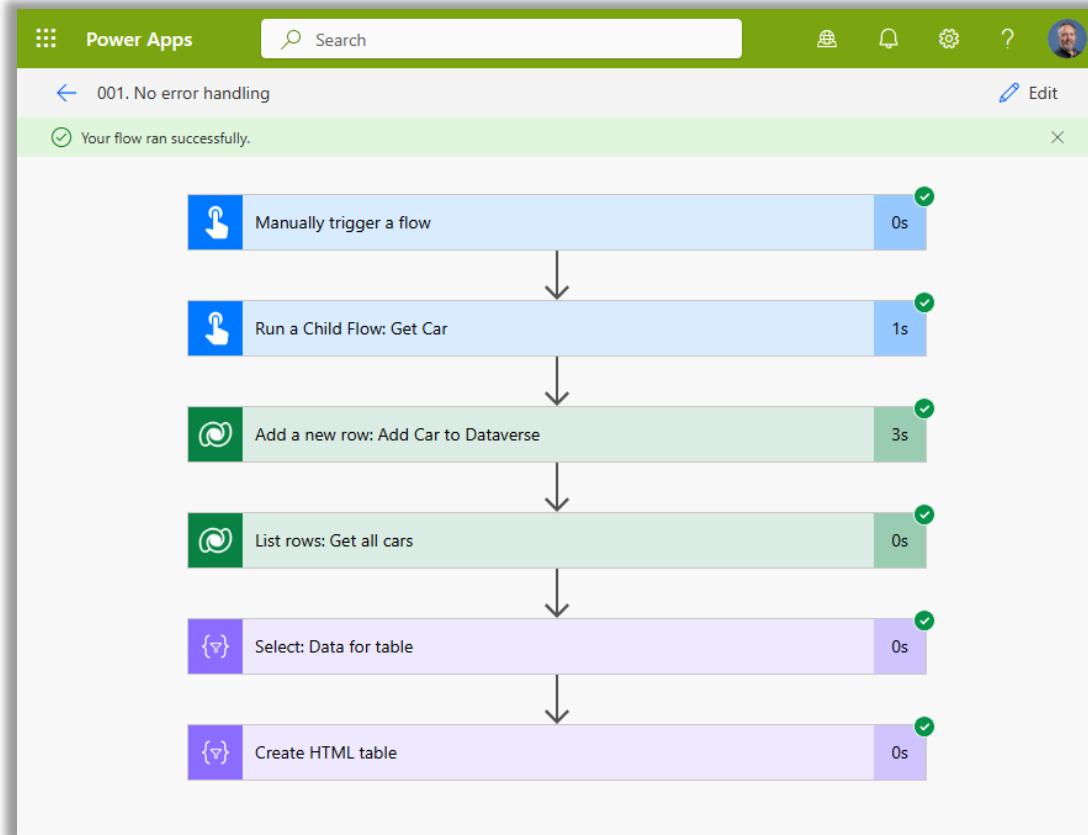
- Set up notifications on failure paths for immediate alerting
- Configure Retry policies with exponential backoff for transient errors
- Use Terminate actions to gracefully end flows with custom status and messages
- Design parallel branches for compensation or rollback mechanisms
- Add structured logging (Dataverse, SharePoint list, Azure Application Insights)
- Leverage variables to capture and pass error details for downstream diagnostics



Interactive Demo time

- Understand the importance of error handling
- Learn how to implement Try-Catch-Finally logic
- Implement custom error messages and notifications

Flow without error handling



Who's gonna know?



How will you know if a flow fails

You can see it in the flow run history:

28-day run history ①		
Start	Duration	Status
May 15, 09:37 AM (3 sec ago)	00:00:01	Failed
May 12, 11:33 AM (2 d ago)	00:00:01	Test failed
May 12, 11:24 AM (2 d ago)	00:00:01	Test failed
May 12, 11:17 AM (2 d ago)	00:00:01	Test failed
May 12, 11:06 AM (2 d ago)	00:05:01	Test succeeded
May 12, 10:53 AM (2 d ago)	00:00:05	Test succeeded

You will receive an email:

Microsoft Power Automate

❗ 5 of your flow(s) have failed

The flow(s) listed failed in the past week and may need your attention.

5 Notifications:

Flow name	Failure count
050. Get flow runs	2
003. Advanced error handling	49
002. Basic error handling	18
001. No error handling	29
020. Initialize data	5

If you need more help, please visit the [Power Automate support page](#).

Did you find this email helpful? [Yes](#) [No](#)

But only once per day.

How will you know if a flow fails

28-day run history (i)			Edit columns	↻ All runs
Start	Duration	Status		
May 15, 09:37 AM (3 sec ago)	00:00:01	Failed		
May 12, 11:33 AM (2 d ago)	00:00:01	Test failed		
May 12, 11:24 AM (2 d ago)	00:00:01	Test failed		
May 12, 11:17 AM (2 d ago)	00:00:01	Test failed		
May 12, 11:06 AM (2 d ago)	00:05:01	Test succeeded		
May 12, 10:53 AM (2 d ago)	00:00:05	Test succeeded		

How will you know?

You can see it in the flow run history.

28-day run history ⓘ	
Start	Duration
May 15, 09:37 AM (3 sec ago)	00:00:01
May 12, 11:33 AM (2 d ago)	00:00:01
May 12, 11:24 AM (2 d ago)	00:00:01
May 12, 11:17 AM (2 d ago)	00:00:01
May 12, 11:06 AM (2 d ago)	00:05:01
May 12, 10:53 AM (2 d ago)	00:00:05

 Microsoft Power Automate

! 5 of your flow(s) have failed

The flow(s) listed failed in the past week and may need your attention.

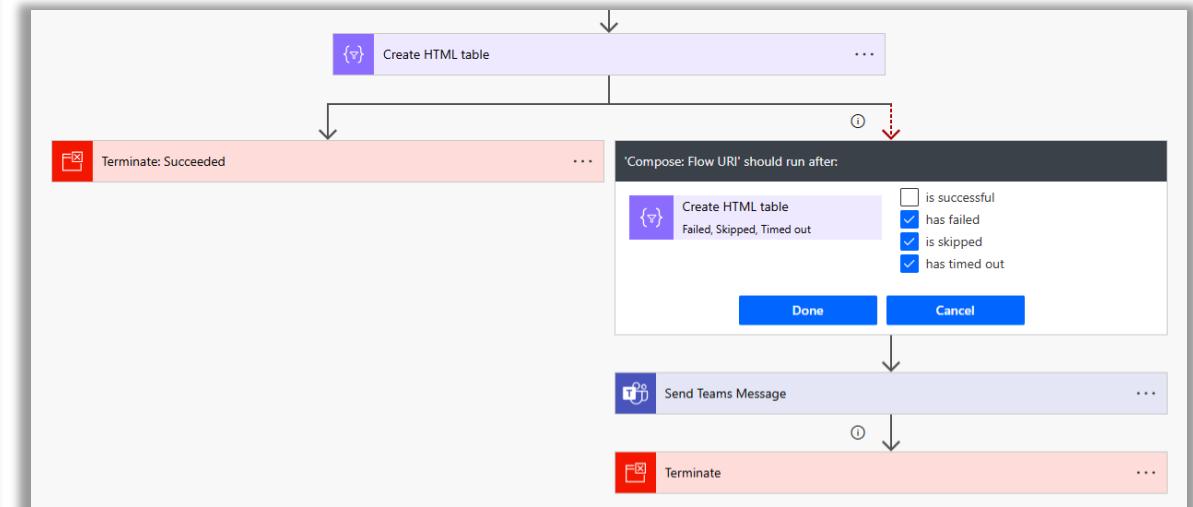
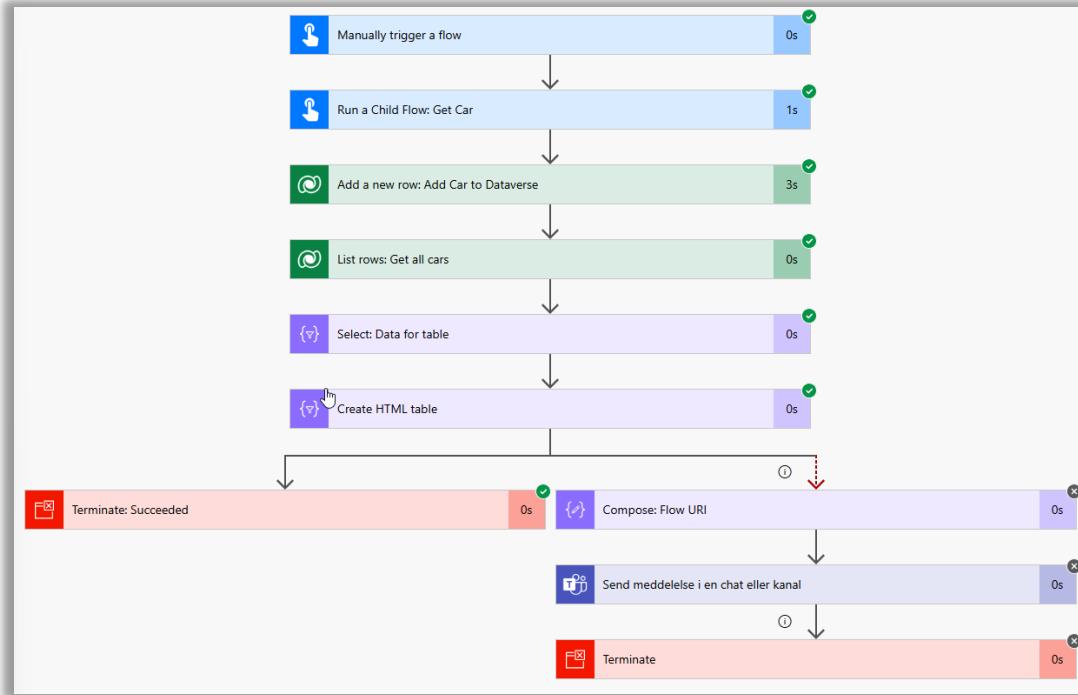
5 Notifications:

Flow name	Failure count
050. Get flow runs	2
003. Advanced error handling	49
002. Basic error handling	18
001. No error handling	29
020. Initialize data	5

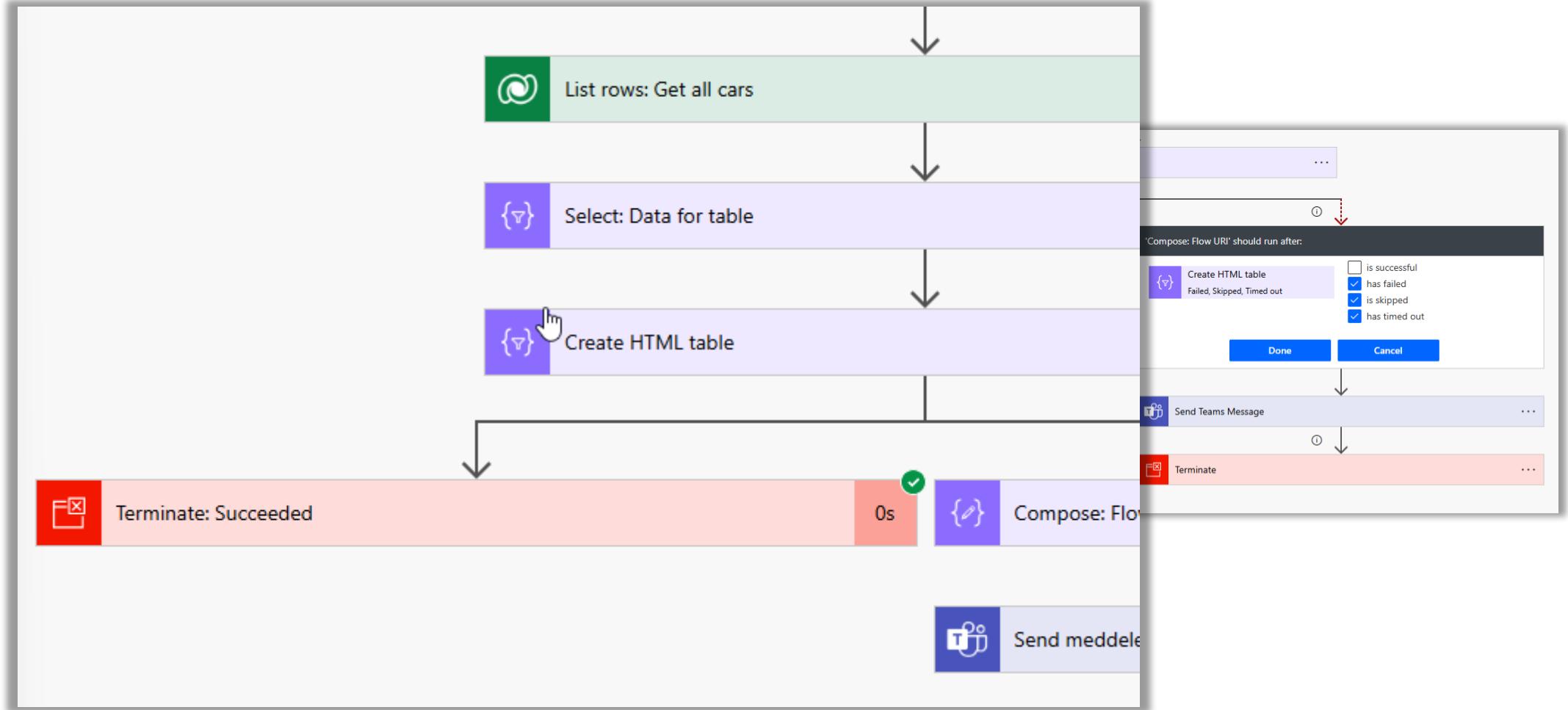
If you need more help, please visit the [Power Automate support page](#).

Did you find this email helpful? [Yes](#) [No](#)

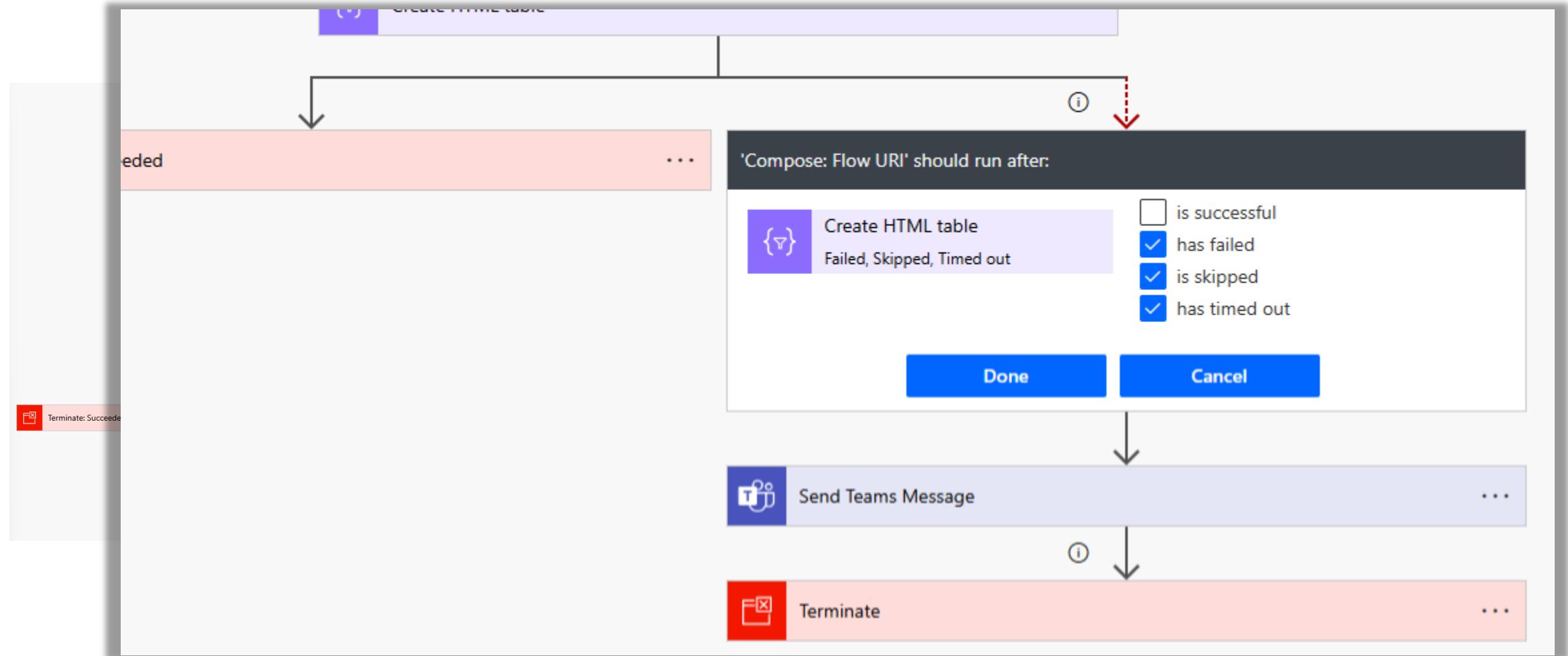
Basic error handling – Parallel Actions



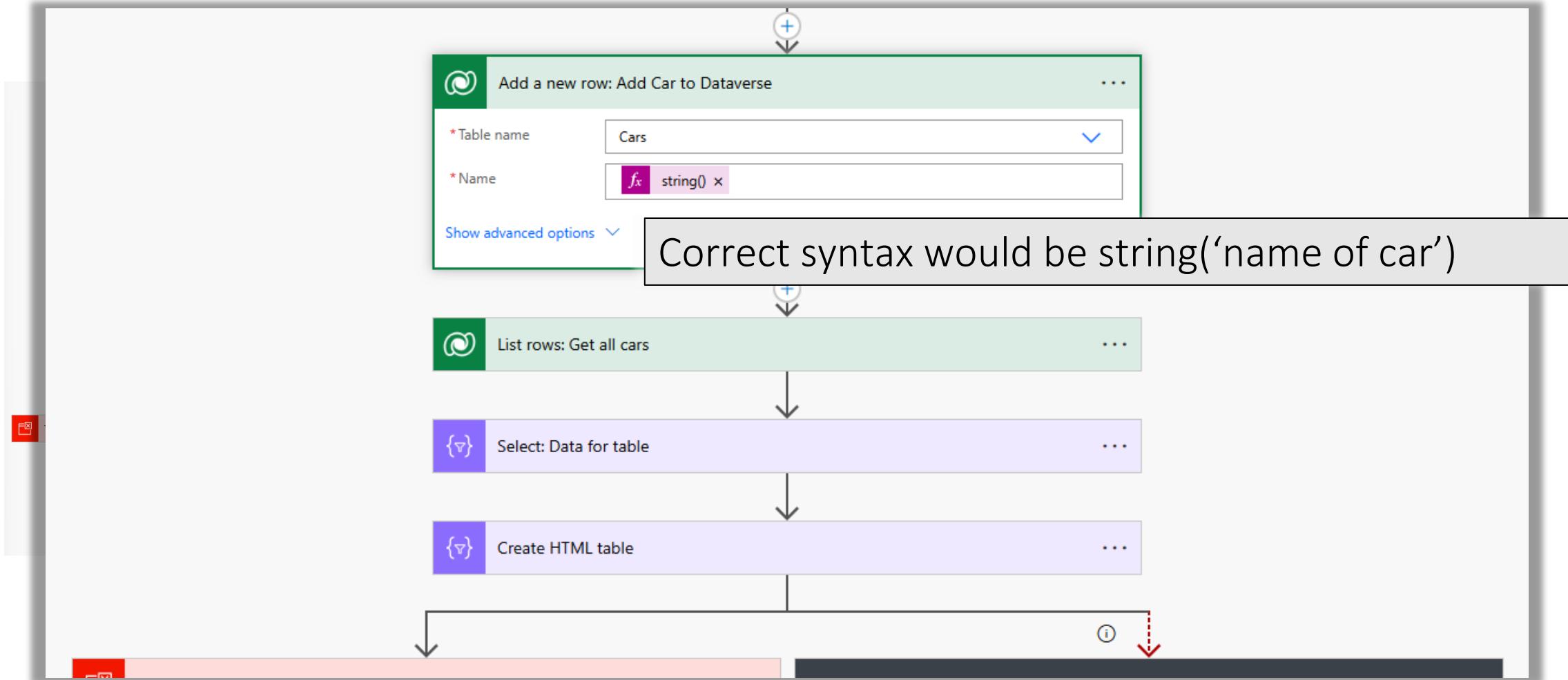
Left Branch – Success Path

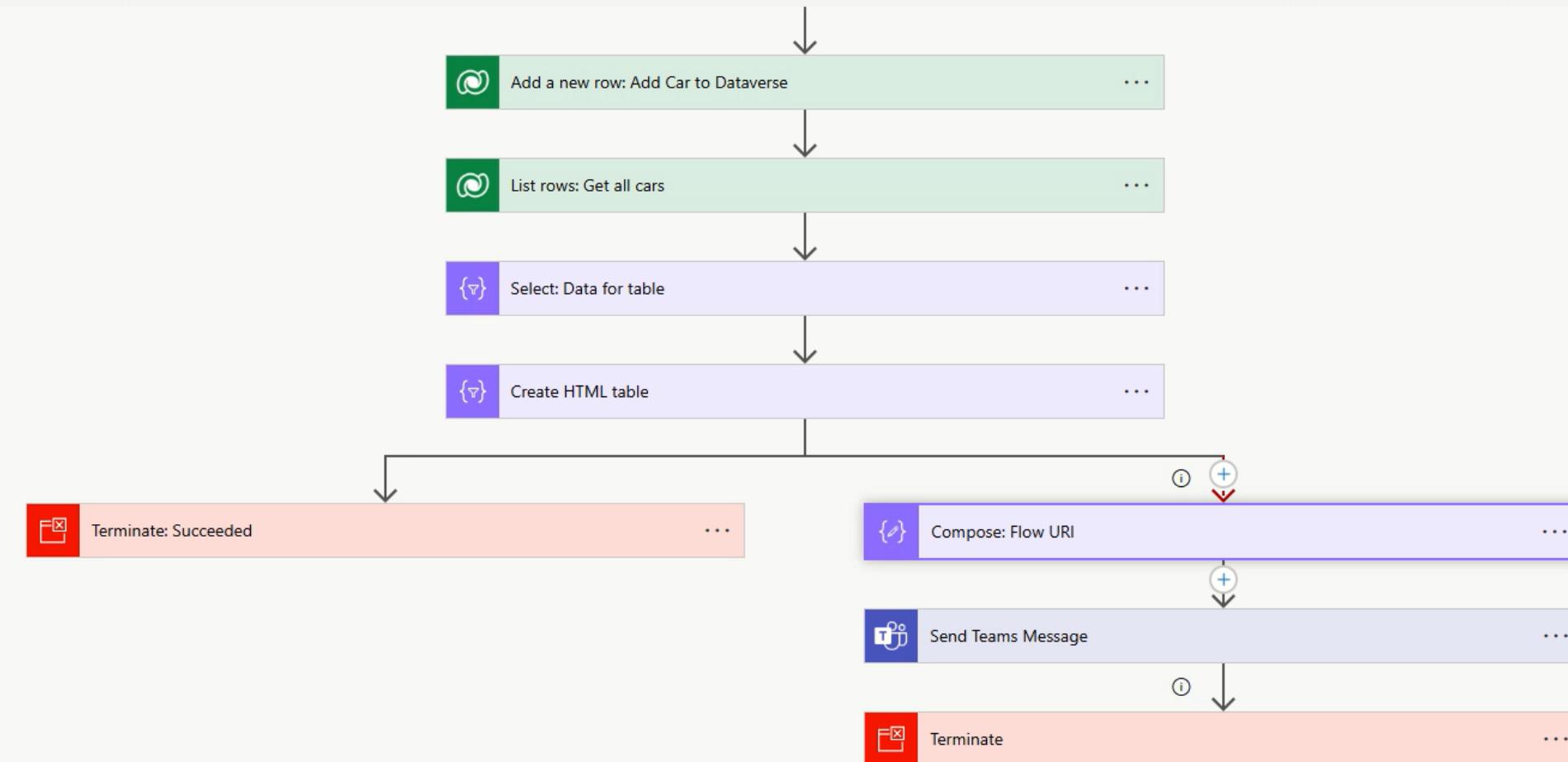


Right Branch – Failure Path



Syntax Error Example



[002. Basic error handling](#)[+ New step](#)[Save](#)

Messages received after error handling

The screenshot shows the Microsoft Power Platform Workflows interface. At the top, there's a navigation bar with 'Workflows' (highlighted in blue), 'Chat', 'Startside', 'Opret', and 'Om'. Below the navigation, a message from 'Michael Nielsen via Workflows 23:28' says 'An error occurred in your flow: 002. Basic error handling'. A table below details the error:

Action	Error
Add_a_new_row;_Add_Car_to_Dataverse	Unable to process template language expressions in action 'Add_a_new_row;_Add_Car_to_Dataverse' inputs at line '0' and column '0': 'The template language function 'string' expects one parameter: the value that is to be converted to a string. The function was invoked with '0' parameters. Please see https://aka.ms/logicexpressions#string for usage details.'

At the bottom left is a 'Link to run' button.

The screenshot shows the 'Run Details' window. It includes a header with 'Run Details' and a close button. Below the header, it lists 'Start time' (May 19, 11:28 PM (7 sec ago)) and 'Duration' (00:00:03). Under the 'Error' section, it shows the 'Action' ('Add_a_new_row;_Add_Car_to_Dataverse') and the detailed error message: 'Unable to process template language expressions in action 'Add_a_new_row;_Add_Car_to_Dataverse' inputs at line '0' and column '0': 'The template language function 'string' expects one parameter: the value that is to be converted to a string. The function was invoked with '0' parameters. Please see https://aka.ms/logicexpressions#string for usage details.'

Teams message received

Michael Nielsen via Workflows 23:28 Oversæt

An error occurred in your flow:

002. Basic error handling

Action	Error
Add_a_new_row:_Add_Car_to_Dataverse	Unable to process template language expressions in action 'Add_a_new_row:_Add_Car_to_DataVerse' inputs at line '0' and column '0': 'The template language function 'string' expects one parameter: the value that is to be converted to a string. The function was invoked with '0' parameters. Please see https://aka.ms/logicexpressions#string for usage details.'

[Link to run](#)

Teams message received

Michael Nielsen via Workflows 23:28 Oversæt

An error occurred in your flow:

002. Basic error handling

Action	Error
Add_a_new_row:_Add_Car_to_Dataverse	Unable to process template language expressions in action 'Add_a_new_row:_Add_Car_to_DataVerse' inputs at line '0' and column '0': 'The template language function 'string' expects one parameter: the value that is to be converted to a string. The function was invoked with '0' parameters. Please see https://aka.ms/logicexpressions#string for usage details.'

[Link to run](#)

Teams message received

Michael Nielsen via Workflows 23:28 Oversæt

An error occurred in your flow:

002. Basic error handling

Action	Error
Add_a_new_row:_Add_Car_to_Dataverse	Unable to process template language expressions in action 'Add_a_new_row:_Add_Car_to_Dataverse' inputs at line '0' and column '0': 'The template language function 'string' expects one parameter: the value that is to be converted to a string. The function was invoked with '0' parameters. Please see https://aka.ms/logicexpressions#string for usage details.'

[Link to run](#)

Teams message received

Michael Nielsen via Workflows 23:28 Oversæt

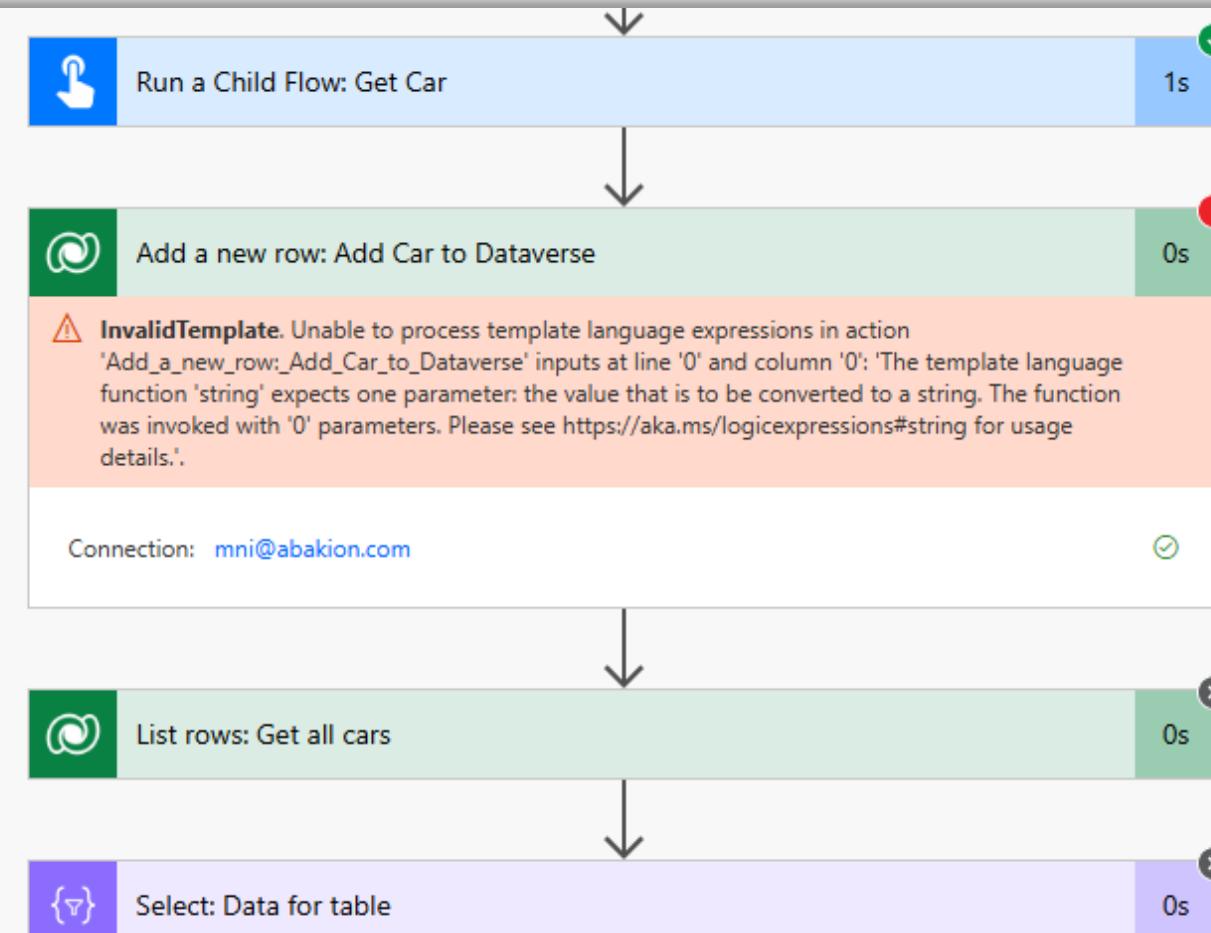
An error occurred in your flow:

002. Basic error handling

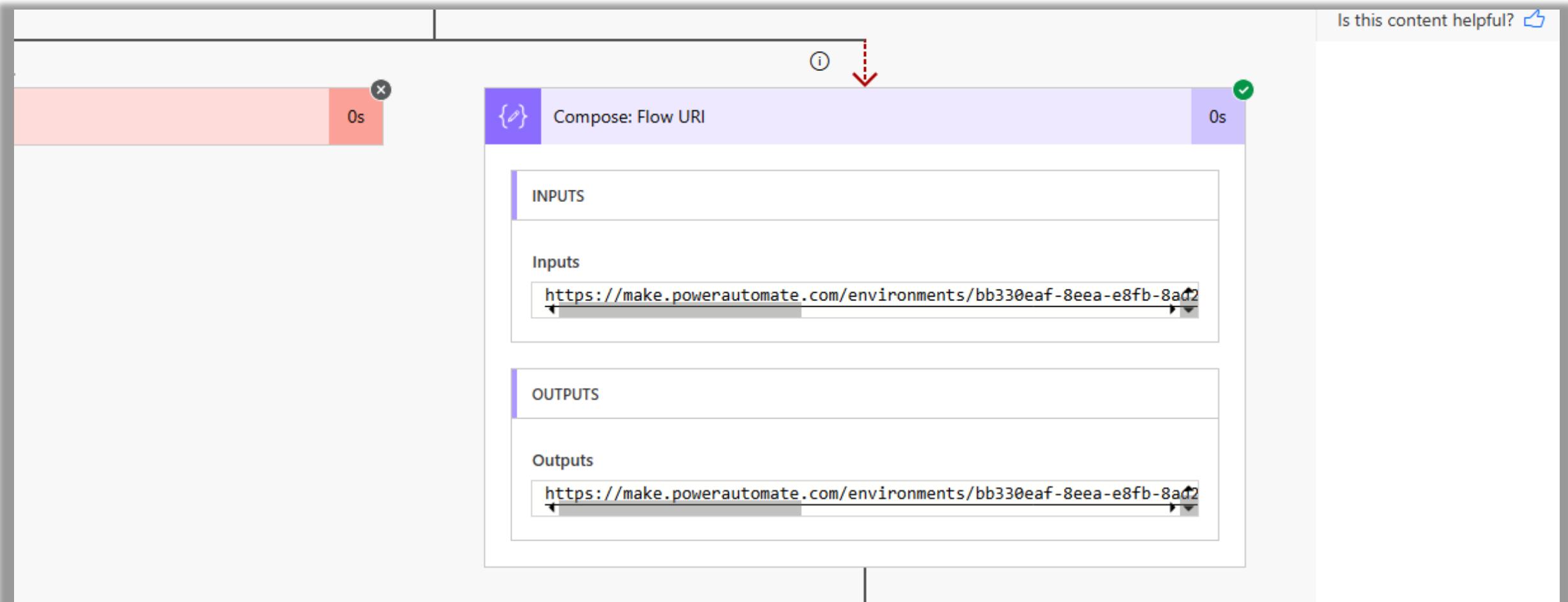
Action	Error
Add_a_new_row:_Add_Car_to_Dataverse	Unable to process template language expressions in action 'Add_a_new_row:_Add_Car_to_Dataverse' inputs at line '0' and column '0': 'The template language function 'string' expects one parameter: the value that is to be converted to a string. The function was invoked with '0' parameters. Please see https://aka.ms/logicexpressions#string for usage details.'

[Link to run](#)

Direct link into my failed flow



Get the URI for the actual flow run



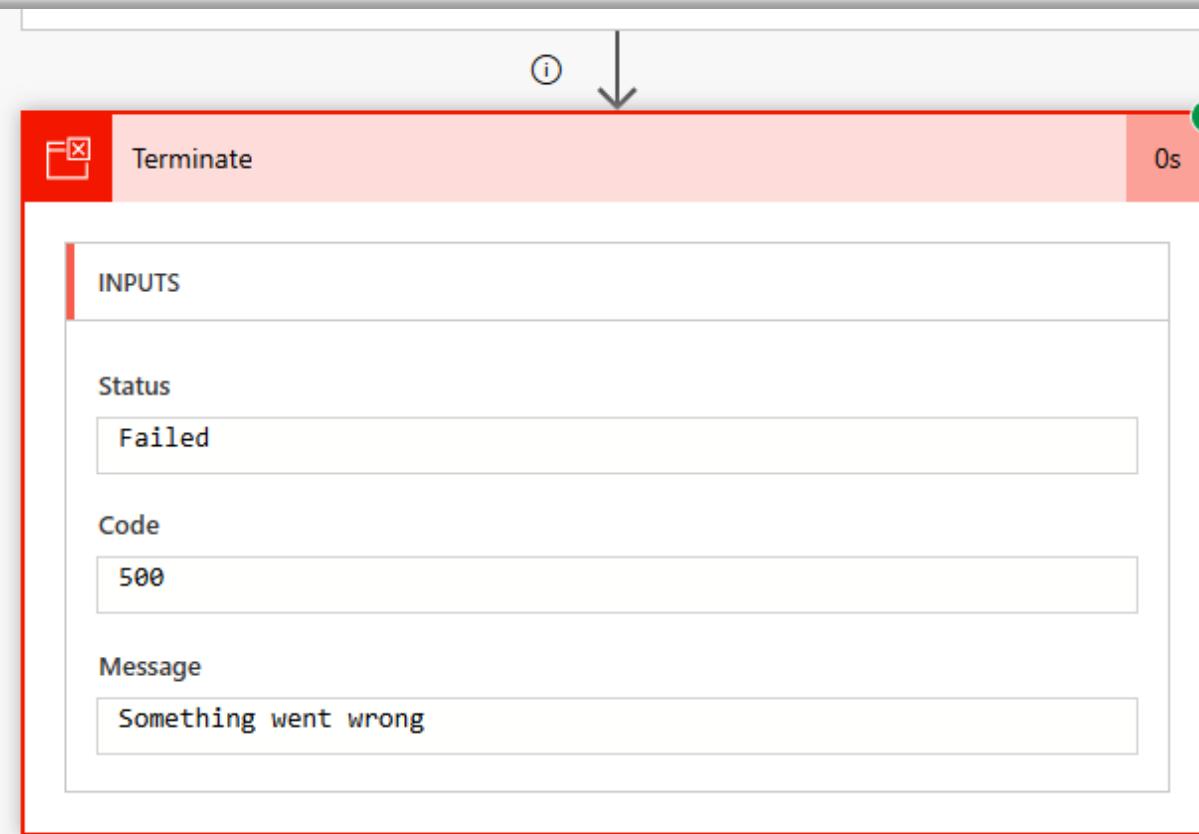
Send the Teams message

The screenshot shows the configuration interface for the 'Send Teams Message' action within Microsoft Power Automate. The action is set to run every 1 second. The configuration fields are as follows:

- Post as:** Flow bot
- Post in:** Chat with Flow bot
- Recipient:** mni@abakion.com;
- Message:**

<table><tr><th>Action</th><th>Error</th></tr><tr><td>Add_a_new_row:_Add_Car_to_Dataverse</td><td>Unable to process</td></tr></table>

Terminate the flow as failed



Unexpected Failure

But what if another action fails than the one you expected?

Run Details X

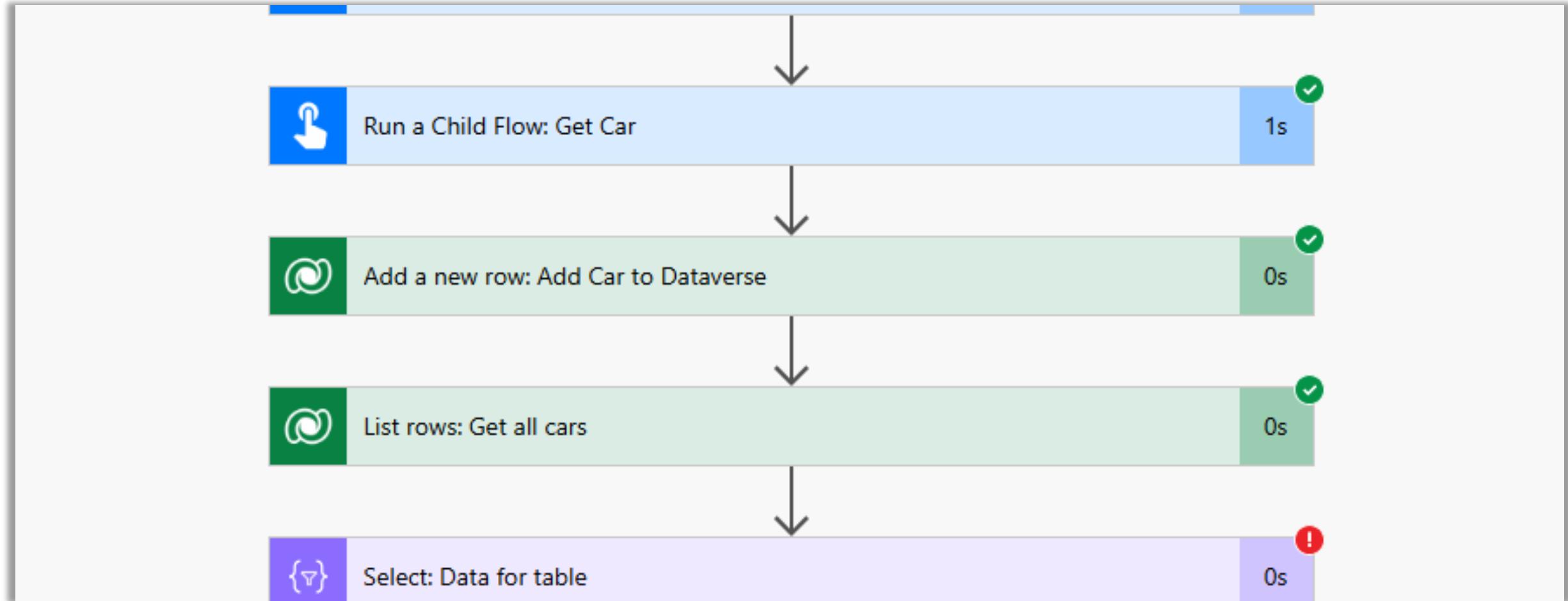
Start time	Duration
May 19, 11:56 PM (3 sec ago)	00:00:02

Error

Action 'Select:_Data_for_table' failed

The 'from' property value in the 'select' action inputs is of type 'String'. The value must be an array.

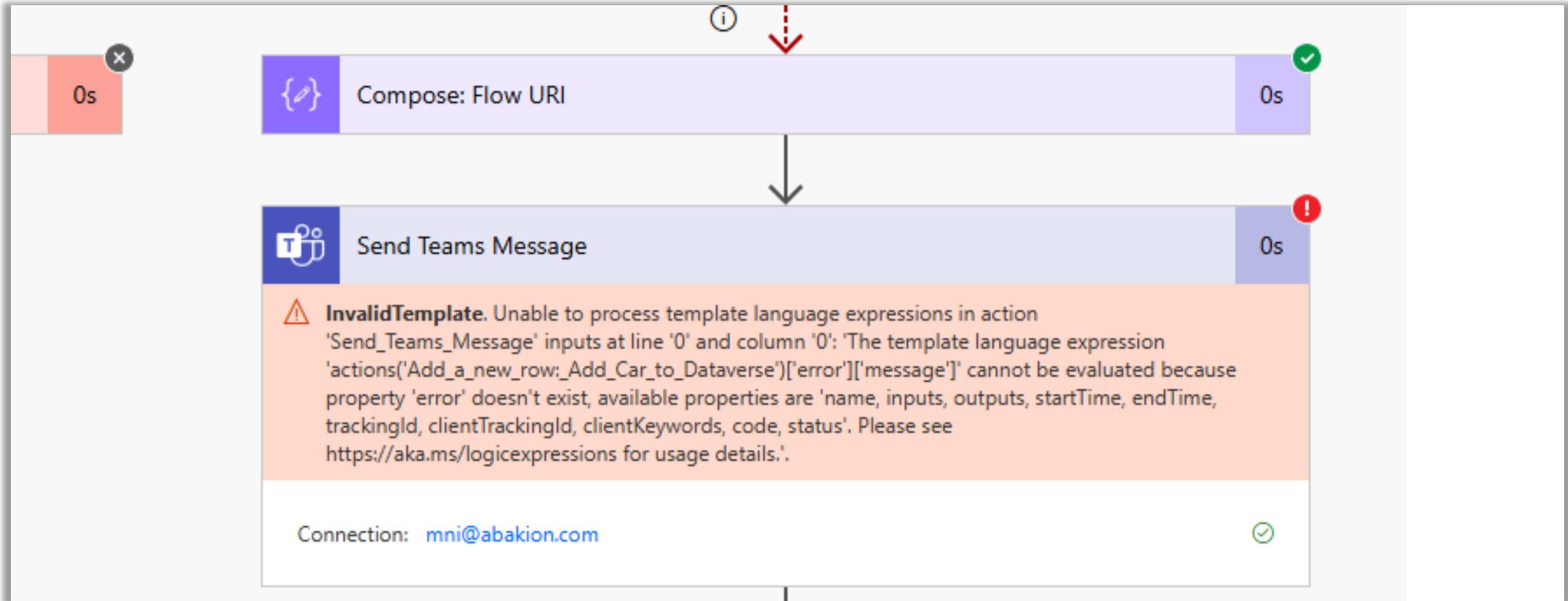
Failure is in another action



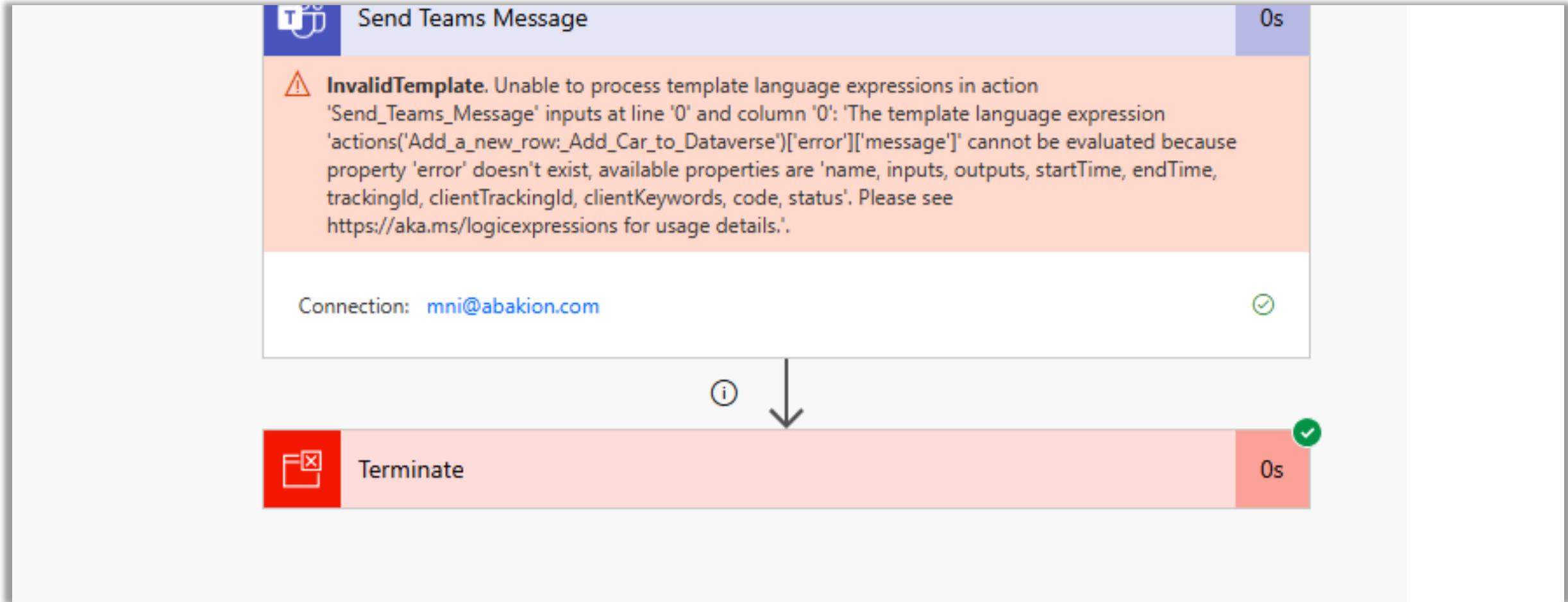
Bad Data was introduced to Select: Data for table

The screenshot shows a Power Automate step titled "Select: Data for table". The status bar indicates "0s" and has a red notification badge with an exclamation mark. A large orange error message box contains the text: "⚠️ BadRequest. The 'from' property value in the 'select' action inputs is of type 'String'. The value must be an array." Below the error message is an "INPUTS" section with a "From" field containing the string "This is bad data". The string is highlighted with a blue selection bar, and a small scroll bar is visible on the right side of the input field.

Basic error handling



Basic error handling



Implement Try-Catch-Finally

- **Try Scope**

Encloses the logic where errors can occur.

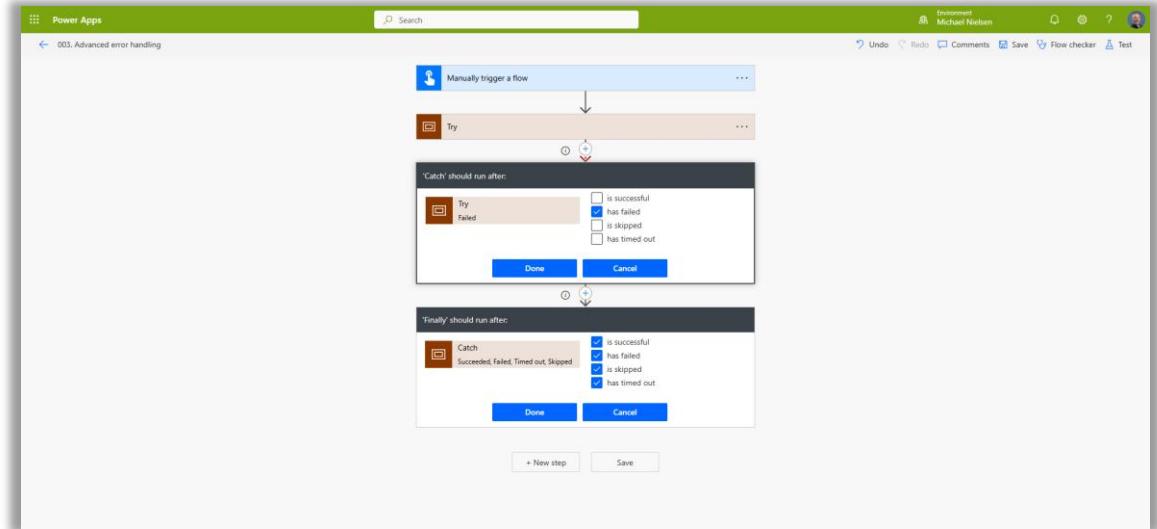
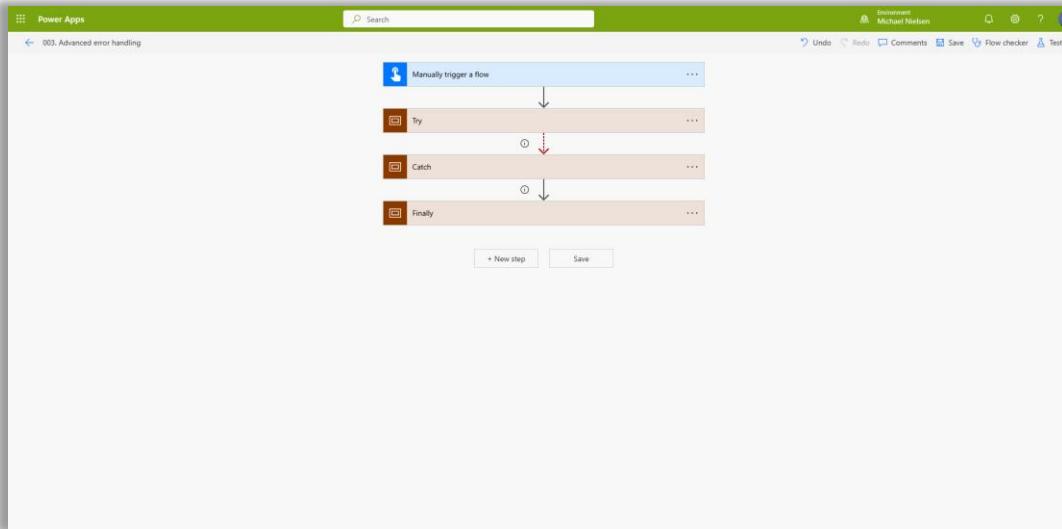
- **Catch Scope**

Catches and retrieves any errors that occur in the Try scope.

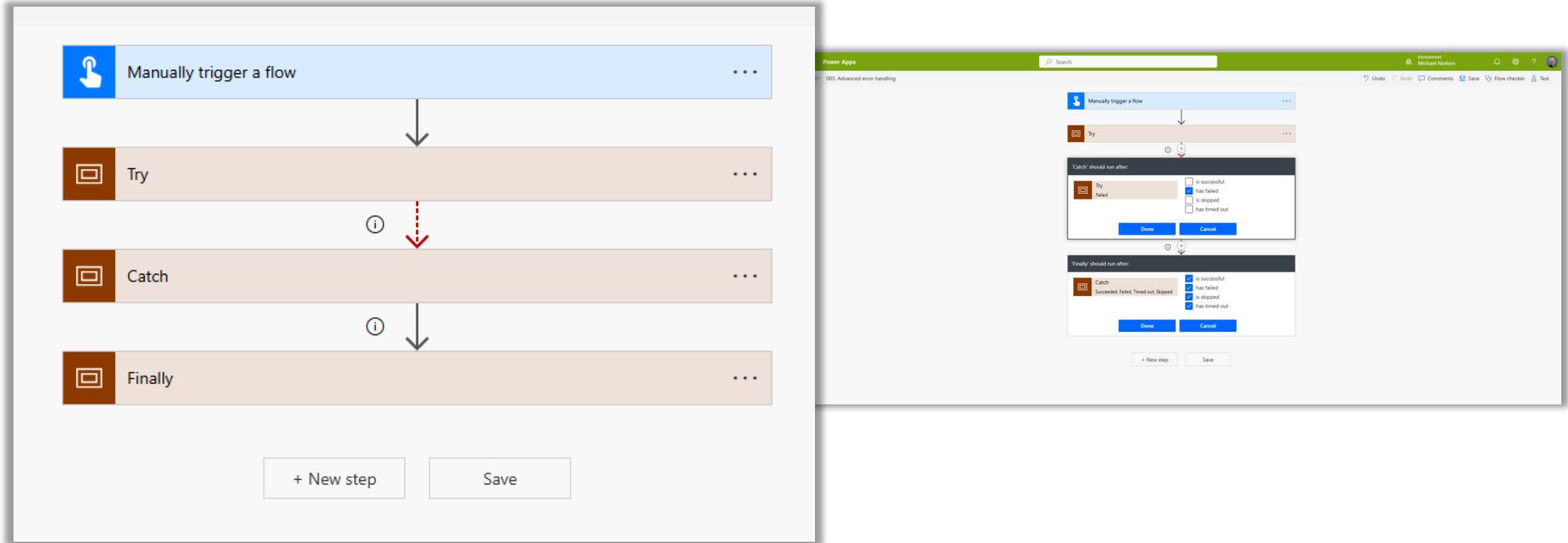
- **Finally Scope**

Runs regardless of whether an error occurred or not, useful for cleanup operations.

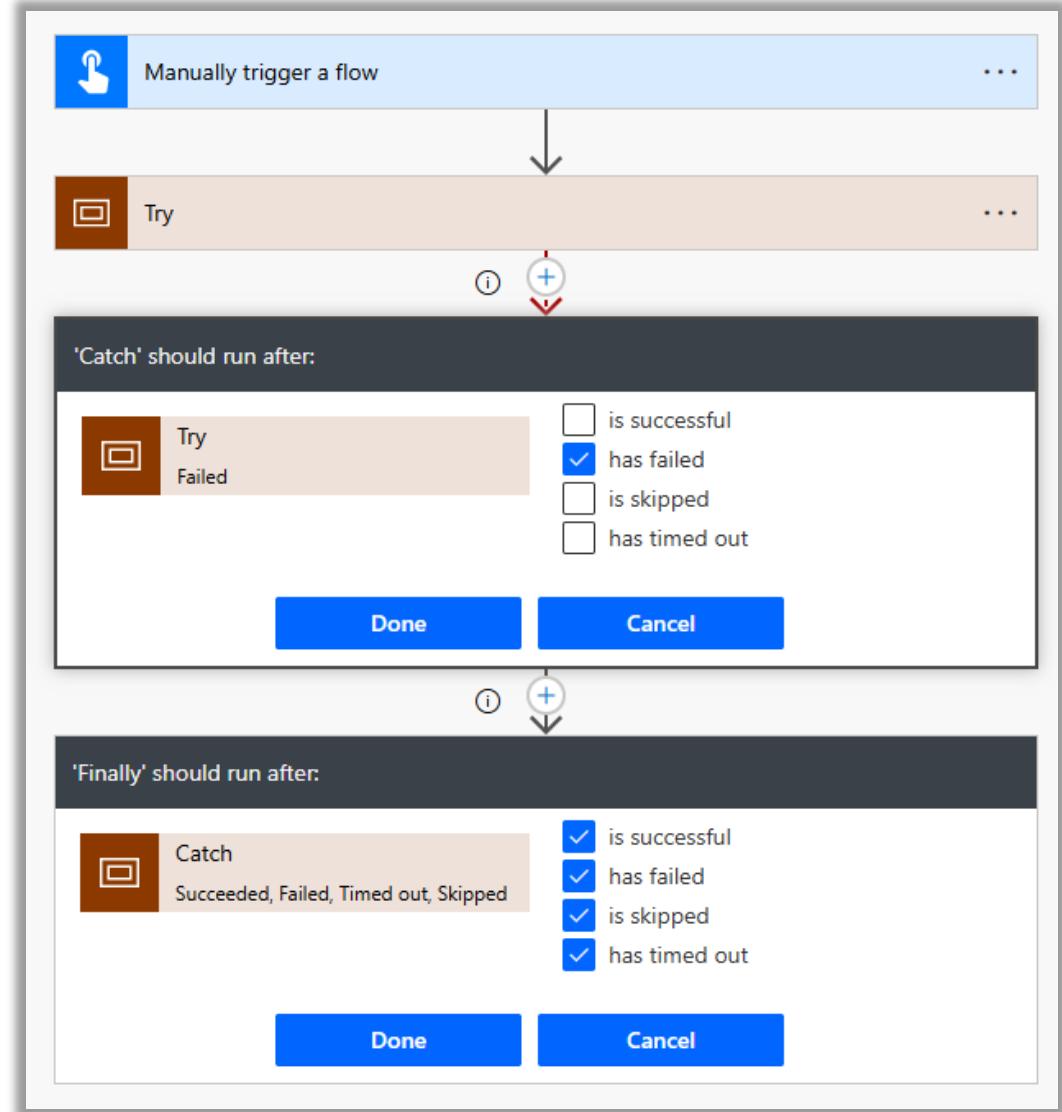
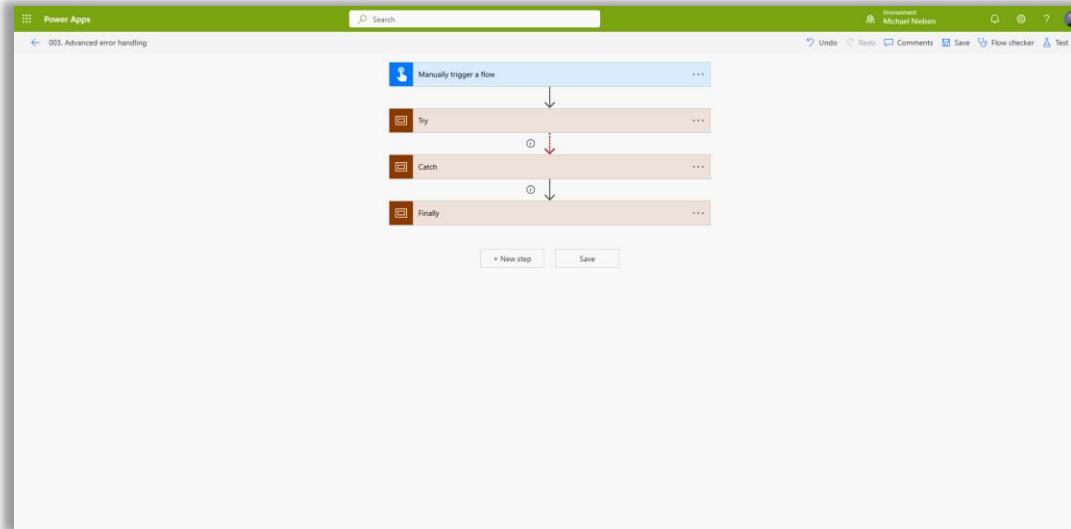
FLOW WITH TRY-CATCH FINALLY



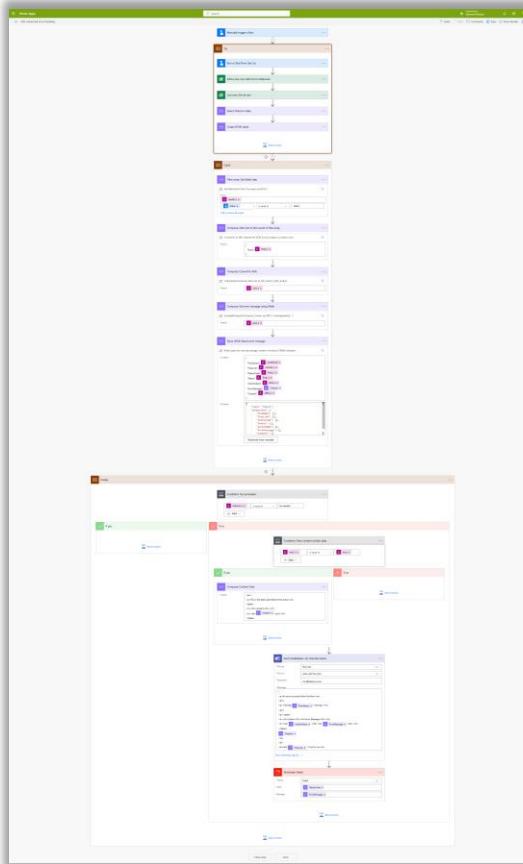
FLOW WITH TRY-CATCH FINALLY



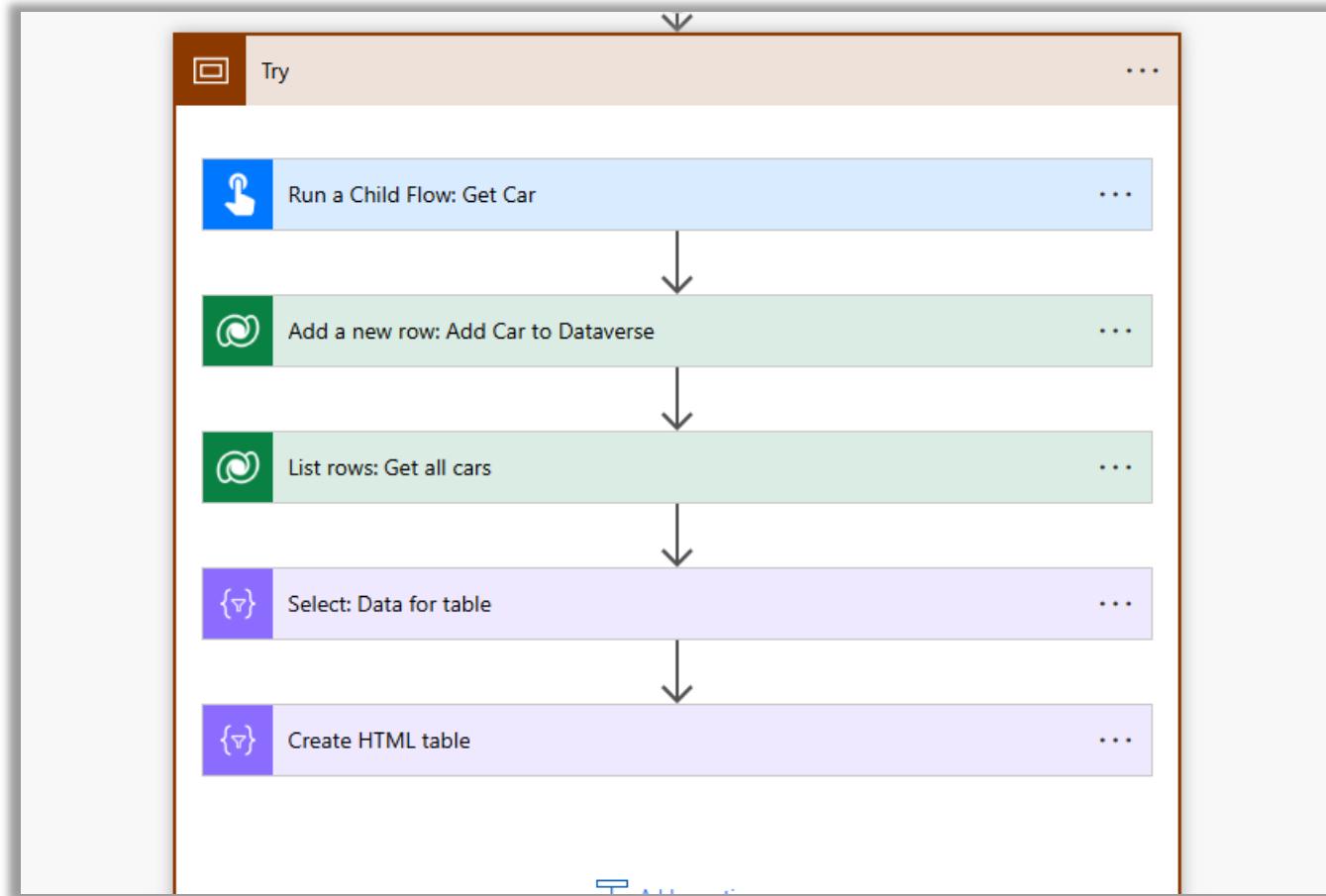
FLOW WITH TRY-CATCH FINALLY



FLOW WITH TRY-CATCH FINALLY

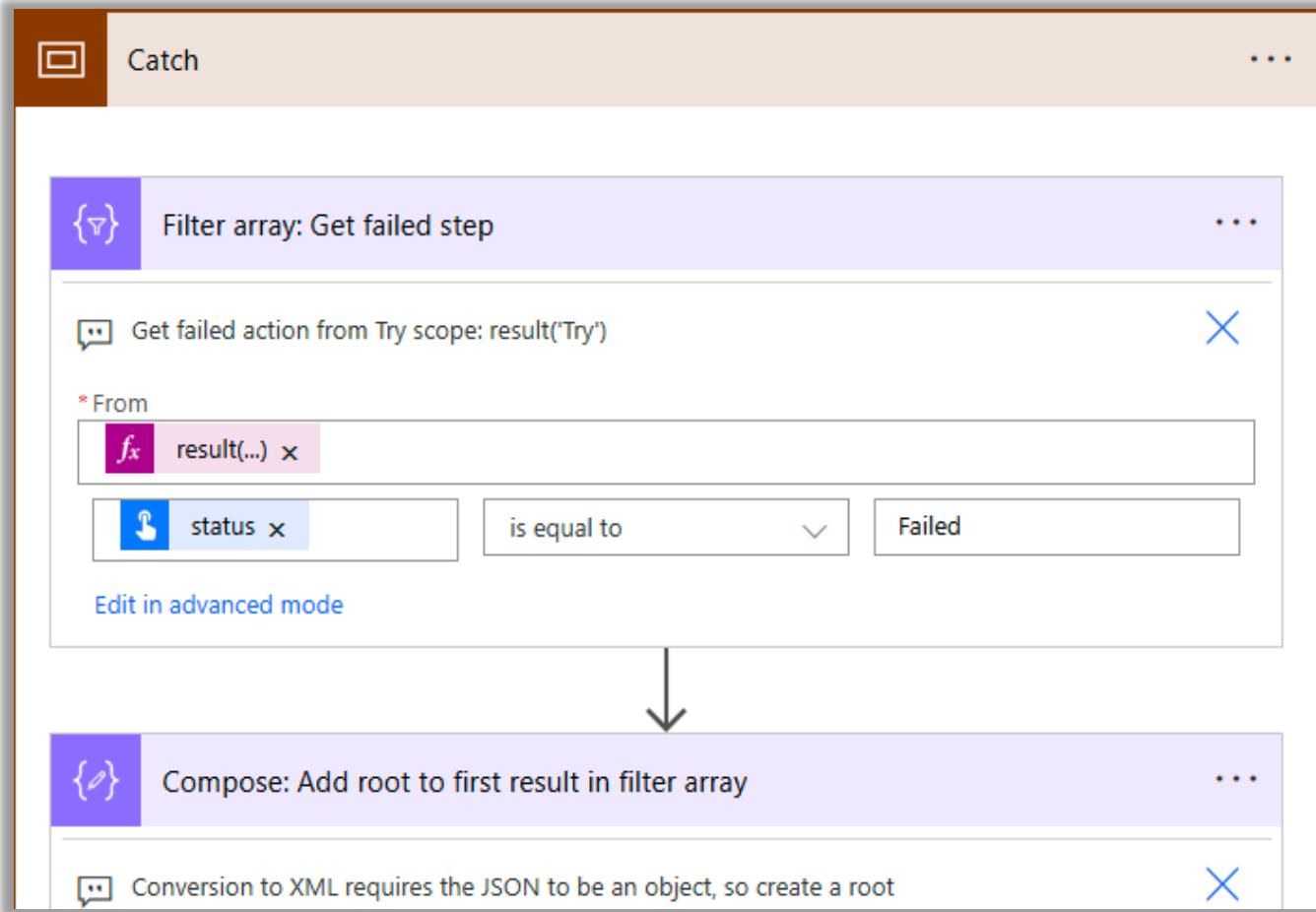


Try Scope



No changes to logic

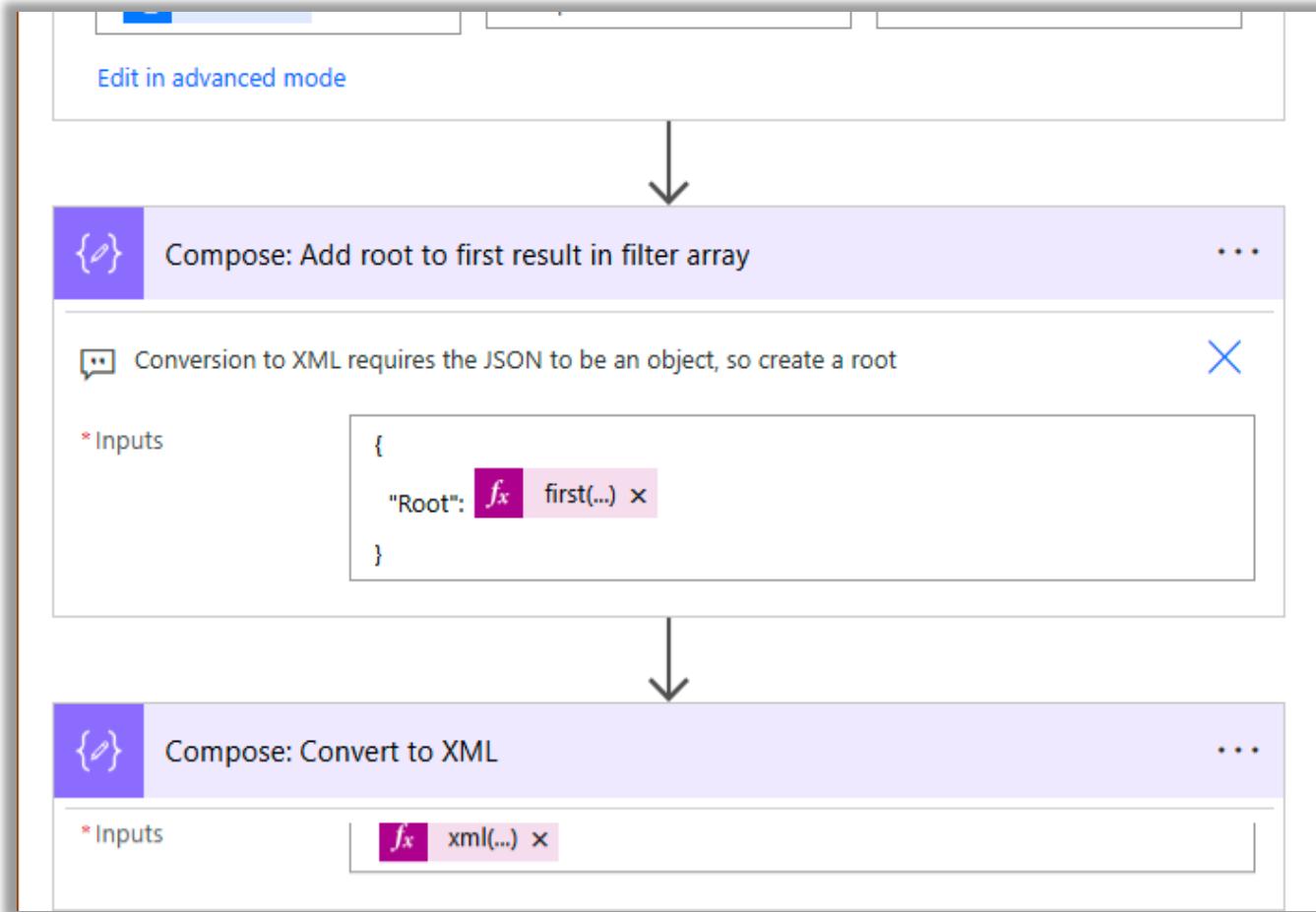
Catch Scope - Filter array



Filter actions within the Try scope container to identify failed steps.

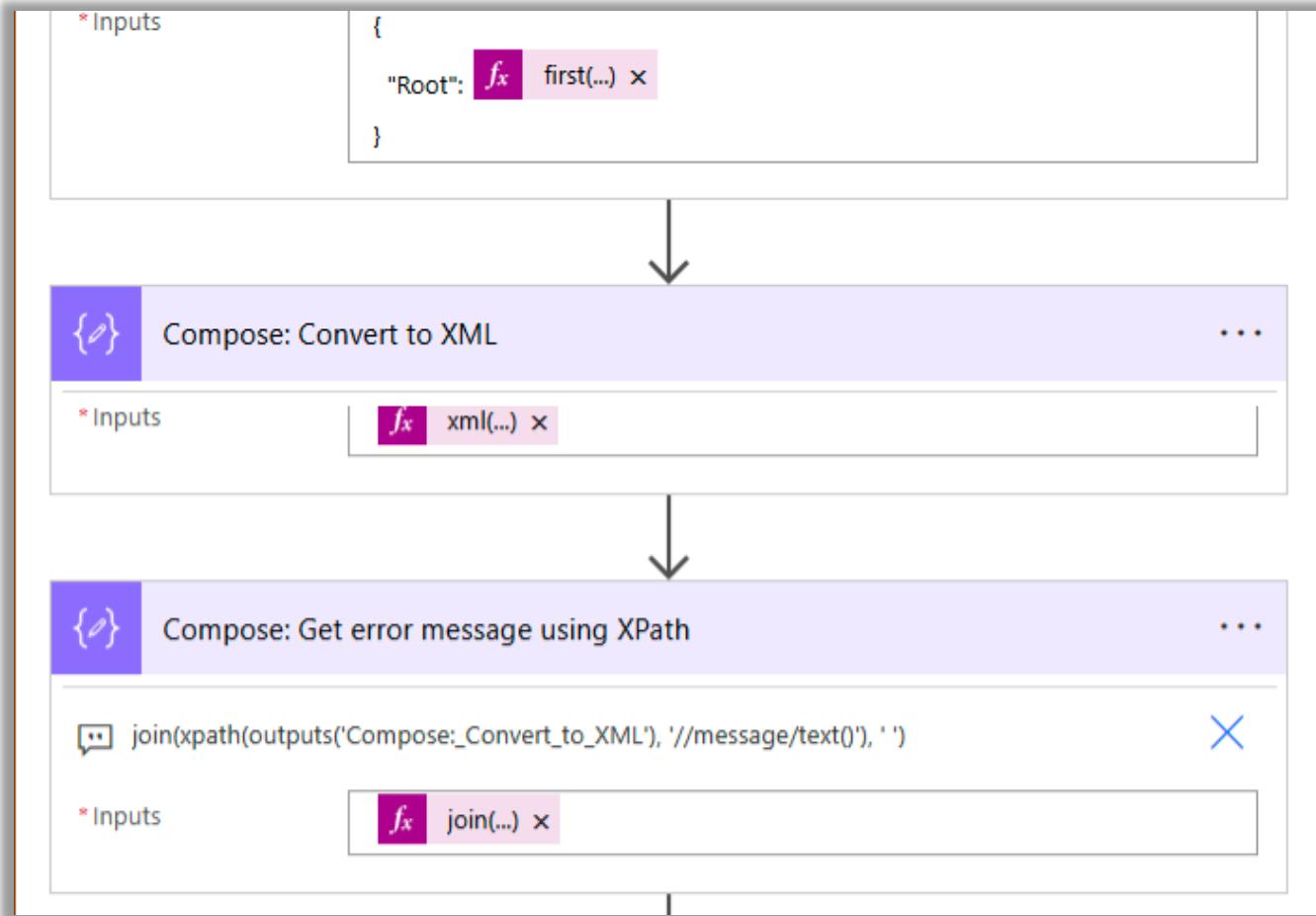
This process filters based on a status of 'Failed'.

Make new JSON object



Next, we need to format the data into an object to ensure proper conversion to XML.

Convert to XML



Convert the structured JSON object to XML using the `xml()` function.

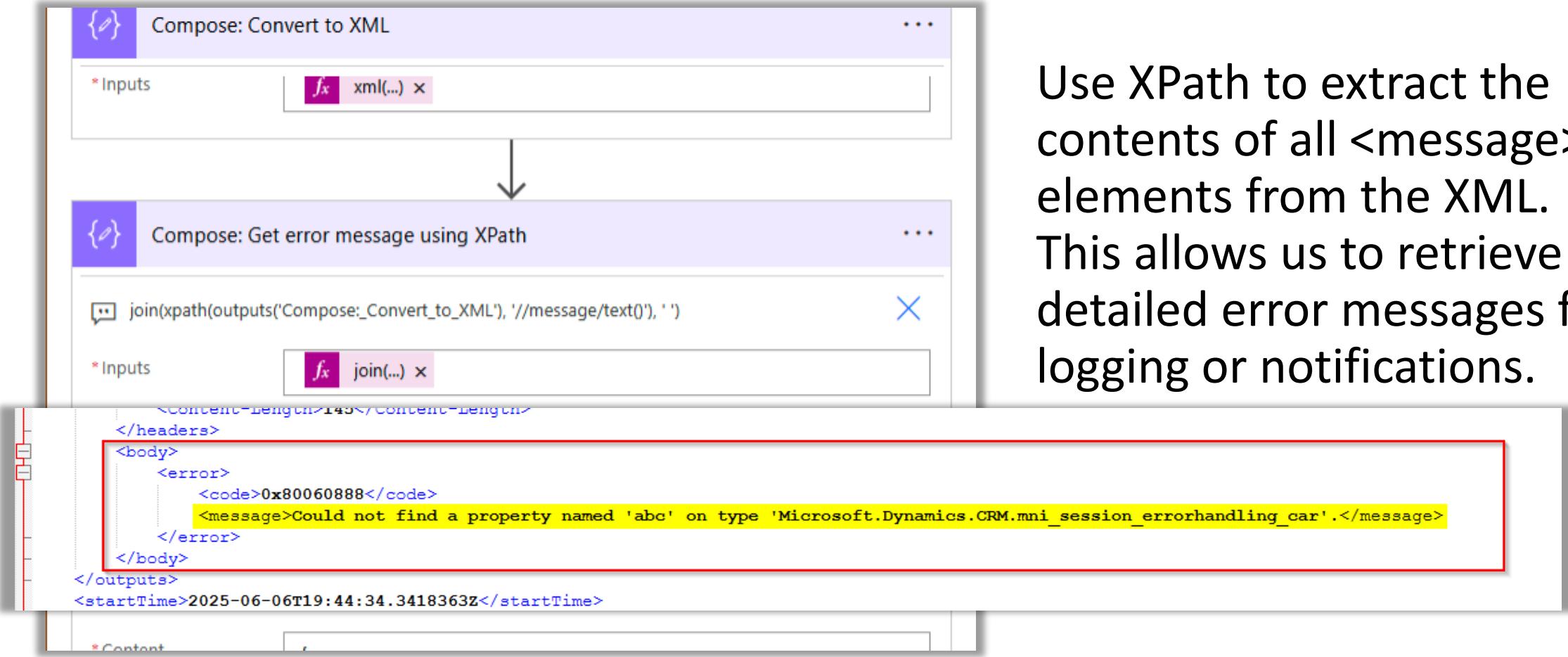
XPATH Magic



```
<Root>
  <name>List_rows:_Get_all_cars</name>
  <inputs>
    <host>
      <apiId>subscriptions/9beb10fe-386f-4597-9e9c-35c0803d67b2/providers/Microsoft.Web/locations/westeurope/runtimes/europe-002/apis/commanddataservicefc
      <connectionReferenceName>shared_commanddataserviceforapps</connectionReferenceName>
      <operationId>ListRecords</operationId>
    </host>
    <parameters>
      <entityName>mni_session_errorhandling_cars</entityName>
      <x0024_select>mni_session_errorhandling_color,mni_session_errorhandling_type,mni_session_errorhandling_licenseplate,createdon</x0024_select>
      <x0024_filter>abo eq 3424</x0024_filter>
    </parameters>
  </inputs>
  <outputs>
    <statusCode>400</statusCode>
    <headers>
      <Cache-Control>no-cache</Cache-Control>
      <Set-Cookie>
        ARRAffinity=aef46c7d0a097b5773958bc2f3b99a7ca129004741d0b69030609019892c0cf615134d20c556b0b34b9b6ae43ec3f5dcad61788de889ffc592af7aca85fc1c508DDA53
        ; path=/; secure; HttpOnly,ReqClientid=0381256b-63e9-44d6-89e2-715a1f91add7; expires=Thu, 06-Jun-2075 19:44:34 GMT; path=/; secure;
        HttpOnly,ARRAffinity=aef46c7d0a097b5773958bc2f3b99a7ca129004741d0b69030609019892c0cf615134d20c556b0b34b9b6ae43ec3f5dcad61788de889ffc592af7aca85fc1
        99020715; path=/; secure; HttpOnly</Set-Cookie>
        <x-ms-service-request-id>7afafccca-4cee-44af-900c-e16d28dfc53f</x-ms-service-request-id>
        <Strict-Transport-Security>max-age=31536000, includeSubDomains</Strict-Transport-Security>
        <REQ_ID>7afafccca-4cee-44af-900c-e16d28dfc53f</REQ_ID>
        <CRM.ServiceId>CRMAppPool</CRM.ServiceId>
        <AuthActivityId>6a19acf6-0ac8-4abc-83d2-97911fd96f8c</AuthActivityId>
        <x-ms-dop-hint></x-ms-dop-hint>
        <x-ms-ratelimit-time-remaining-xrm-requests>1,199.45</x-ms-ratelimit-time-remaining-xrm-requests>
        <x-ms-ratelimit-burst-xrm-requests>7998</x-ms-ratelimit-burst-remaining-xrm-requests>
        <mise-correlation-id>f9cc6531-a63c-45ae-896d-d37a727f67c0</mise-correlation-id>
        <x-Content-Type-Options>nosniff</x-Content-Type-Options>
        <OData-Version>4.0</OData-Version>
        <x-Source>
          1564617813012306524398135224223082219912178912206316770152112738718619361207136206,76482031992321908016316611246206186741311424242105158482112301
        </x-Source>
        <Public>OPTIONS,GET,HEAD,POST</Public>
        <Date>Fri, 06 Jun 2025 19:44:34 GMT</Date>
        <Allow>OPTIONS,GET,HEAD,POST</Allow>
        <Content-Type>application/json; odata.metadata=full</Content-Type>
        <Expires>-1</Expires>
        <Content-Length>145</Content-Length>
    </headers>
    <body>
      <error>
        <code>0x800060888</code>
        <message>Could not find a property named 'abo' on type 'Microsoft.Dynamics.CRM.mni_session_errorhandling_car'.</message>
      </error>
    </body>
  </outputs>
  <startTime>2025-06-06T19:44:34.3418363Z</startTime>
  <endTime>2025-06-06T19:44:34.3786514Z</endTime>
  <trackingId>64d8d08c-fcda-4254-8ede-6eb5685b2f82</trackingId>
  <clientTrackingId>085845236781804449416560904030U61</clientTrackingId>
  <clientKeywords>testFlow</clientKeywords>
  <code>BadRequest</code>
  <status>Failed</status>
</Root>
```

Use XPath to extract all `<message>` elements from the XML and join them into a single string.

XPATH Magic



Use XPath to extract the contents of all `<message>` elements from the XML. This allows us to retrieve detailed error messages for logging or notifications.

Result error object

The screenshot shows the configuration interface for a 'Parse JSON' step. The title bar says 'Parse JSON: Result error message'. A note below it states: 'Build output for warning message, content is limited to 20000 characters.' The main area is titled '* Content' and contains a JSON template:

```
{
    "FlowName": fx workflow(),
    "FlowLink": fx concat(...),
    "StatusCode": fx first(...),
    "Status": fx first(...),
    "ActionName": fx first(...),
    "ErrorMessage": {x} Outputs,
    "Content": fx take(...)
}
```

The bottom section is titled '* Schema' and shows the schema definition:

```
{
    "type": "object",
    "properties": {
```

A custom object containing all necessary details for error message.

CATCH SCOPE – Commands (1)

The screenshot shows the configuration of a 'Parse JSON' action. The title bar says 'Parse JSON: Result error message'. A note below it states 'Build output for warning message, content is limited to 20000 characters.' The 'Content' section contains the following JSON template:

```
{
  "FlowName": fx workflow(),
  "FlowLink": fx concat(...),
  "StatusCode": fx first(...),
  "Status": fx first(...),
  "ActionName": fx first(...),
  "ErrorMessage": { Outputs },
  "Content": fx take(...)
}
```

The 'Schema' section shows the schema definition:

```
{
  "type": "object",
  "properties": {
```

FlowName

workflow().tags.flowDisplayName

FlowLink

concat('https://make.powerautomate.com/environments/', workflow().tags.environmentName, '/flows/', workflow().name, '/runs/', workflow().run.name)

StatusCode

first(xpath(outputs('Compose:_Convert_to_XML'), '//code/text()'))

Status

first(xpath(outputs('Compose:_Convert_to_XML'), '//status/text()'))

CATCH SCOPE – Commands (2)

The screenshot shows the 'Parse JSON' step configuration in the Power Automate designer. The step title is 'Parse JSON: Result error message'. A note says 'Build output for warning message, content is limited to 20000 characters.' The 'Content' section contains the following JSON template:

```
{
    "FlowName": fx workflow(),
    "FlowLink": fx concat(...),
    "StatusCode": fx first(...),
    "Status": fx first(...),
    "ActionName": fx first(...),
    "ErrorMessage": { } Outputs,
    "Content": fx take(...)
}
```

The 'Schema' section shows the following JSON schema definition:

```
{
    "type": "object",
    "properties": {
```

ActionName

```
first(xpath(outputs('Compose:_Convert_to_XML'), '//name/text()'))
```

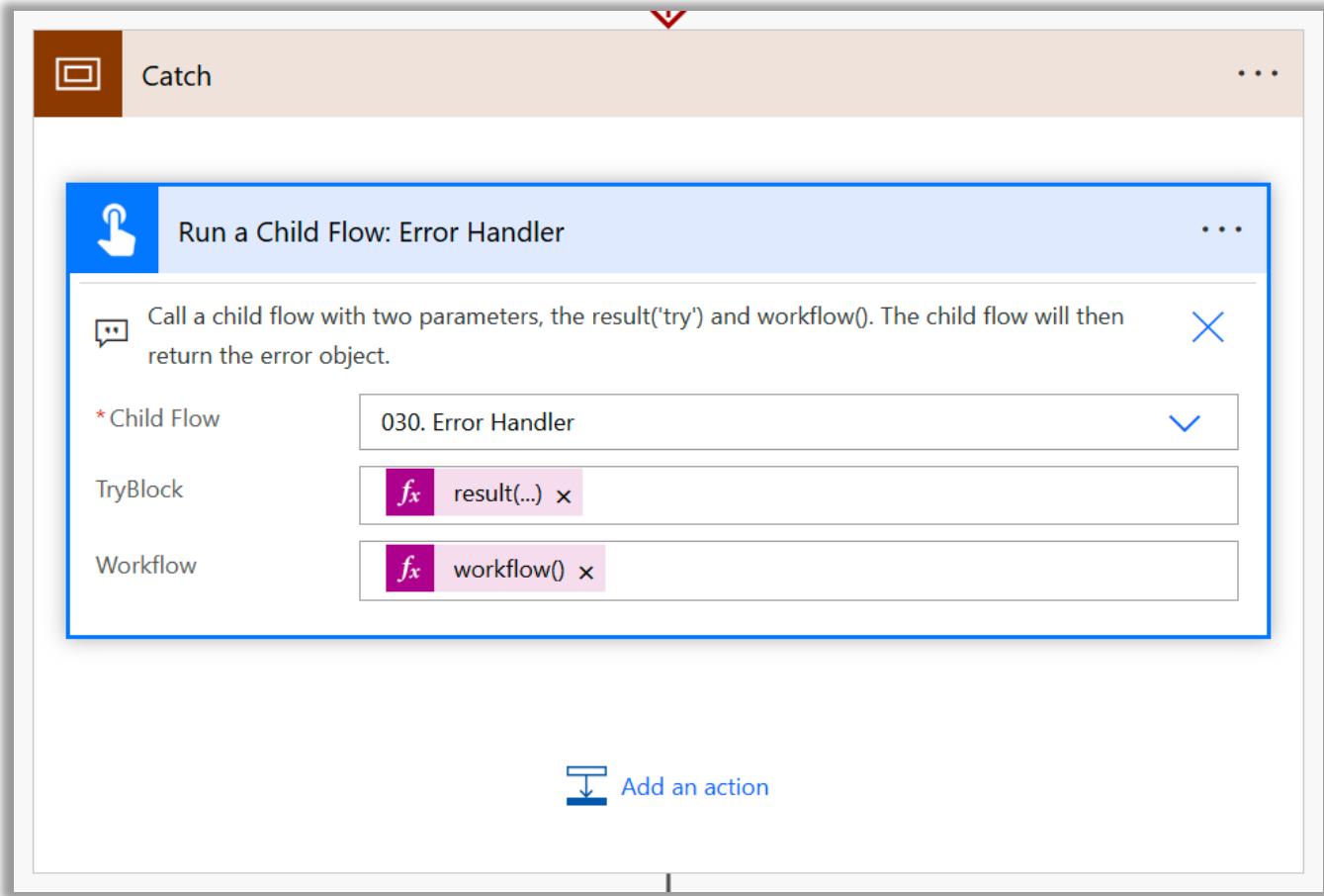
ErrorMessage

```
outputs('Compose:_Get_error_message_using_XPath')
```

Content

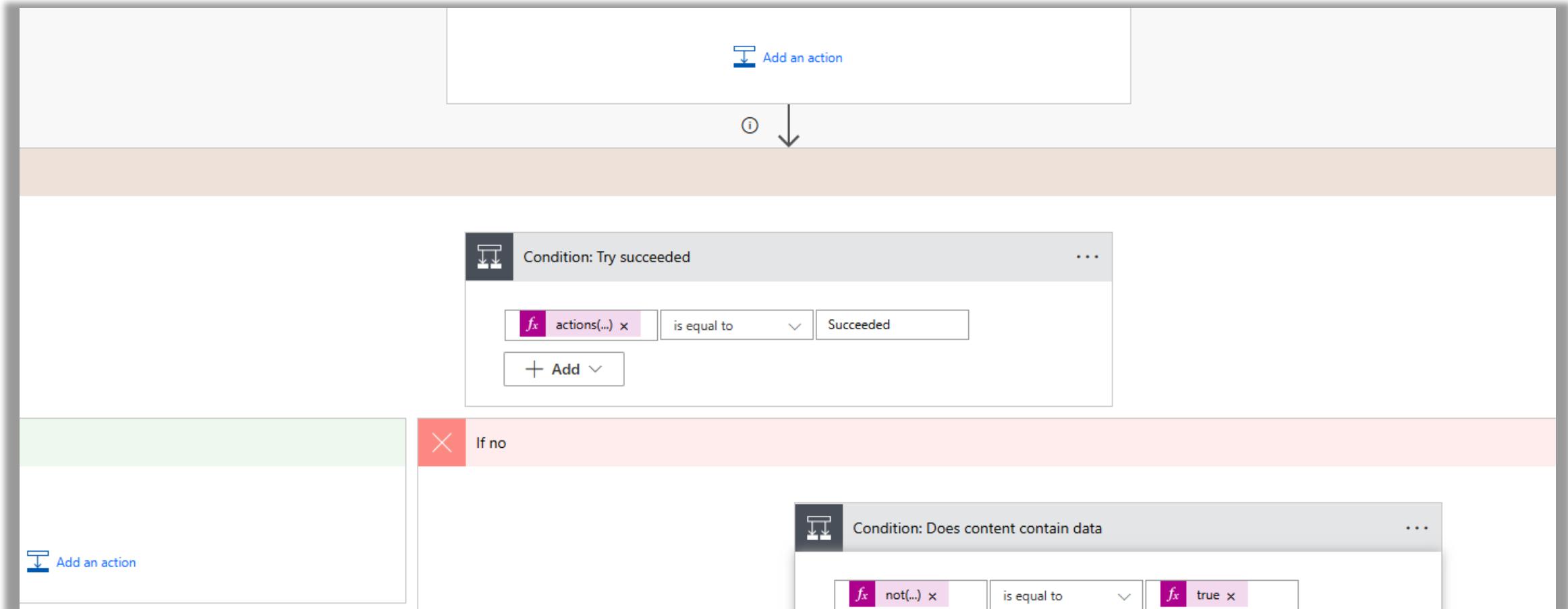
```
take(string(first(body('Filter_array:_Get_failed_step'))), 2000)
```

CATCH SCOPE – Child flow

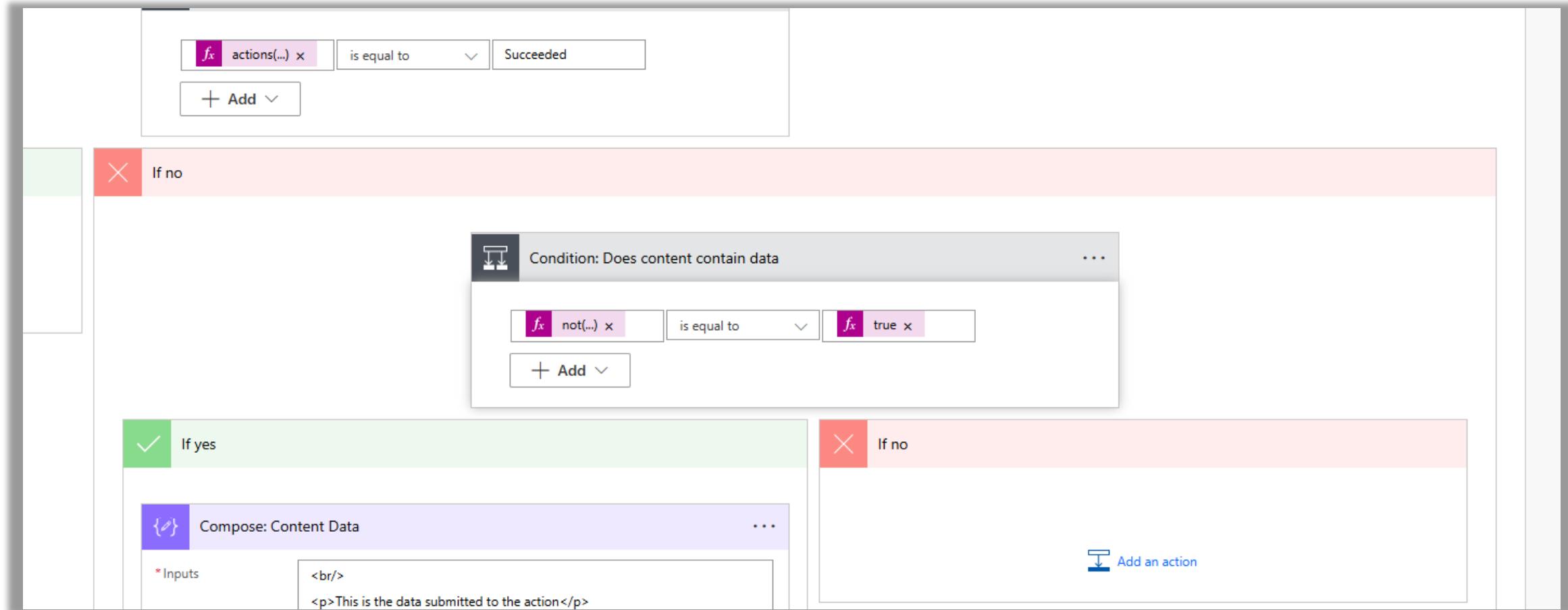


Make the Catch logic into a child flow that can be re-usable across all of your flows.

FINALLY SCOPE – Was Try successful?



FINALLY SCOPE – Try failed, is there content



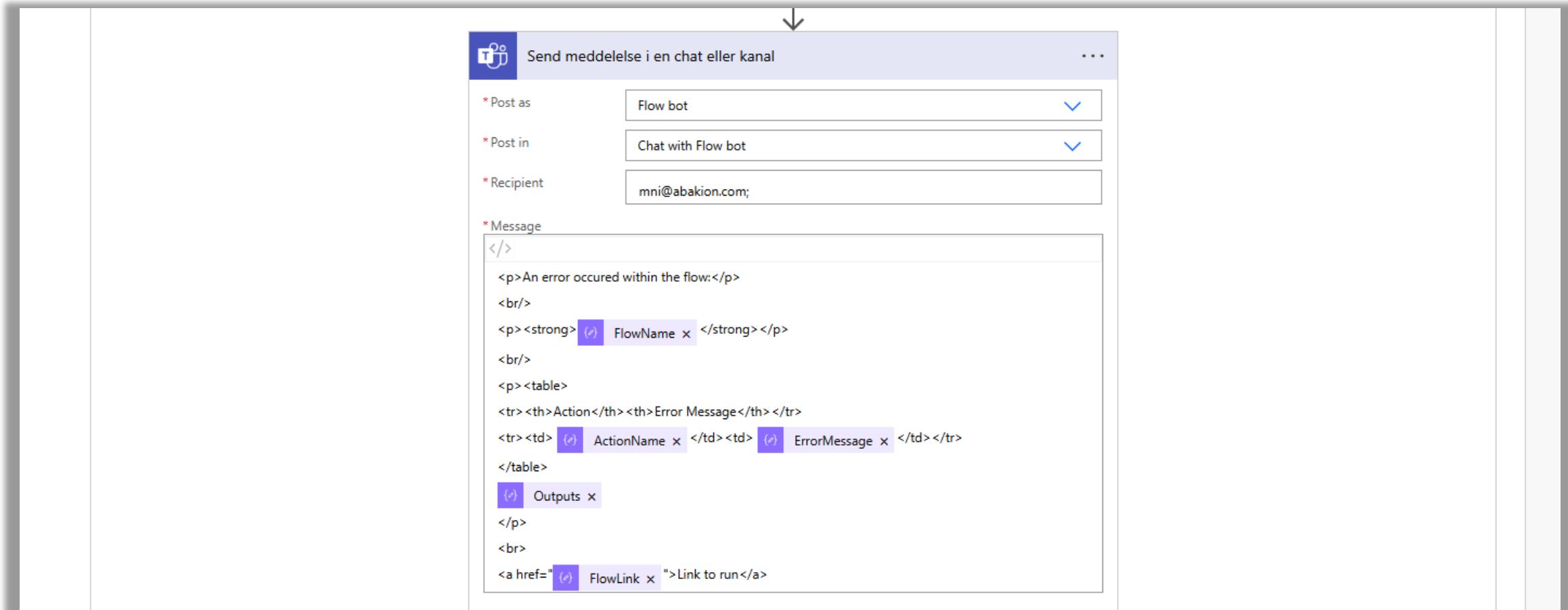
FINALLY SCOPE – Format content

The screenshot shows the 'Format content' scope configuration in Microsoft Power Automate. At the top, a condition block is defined: 'Condition: Does content contain data' with the expression `fx not(...) is equal to true`. Below this, the 'If yes' path is selected, indicated by a green checkmark. This path contains a 'Compose: Content Data' action. The 'Inputs' section of this action shows the following XML code:

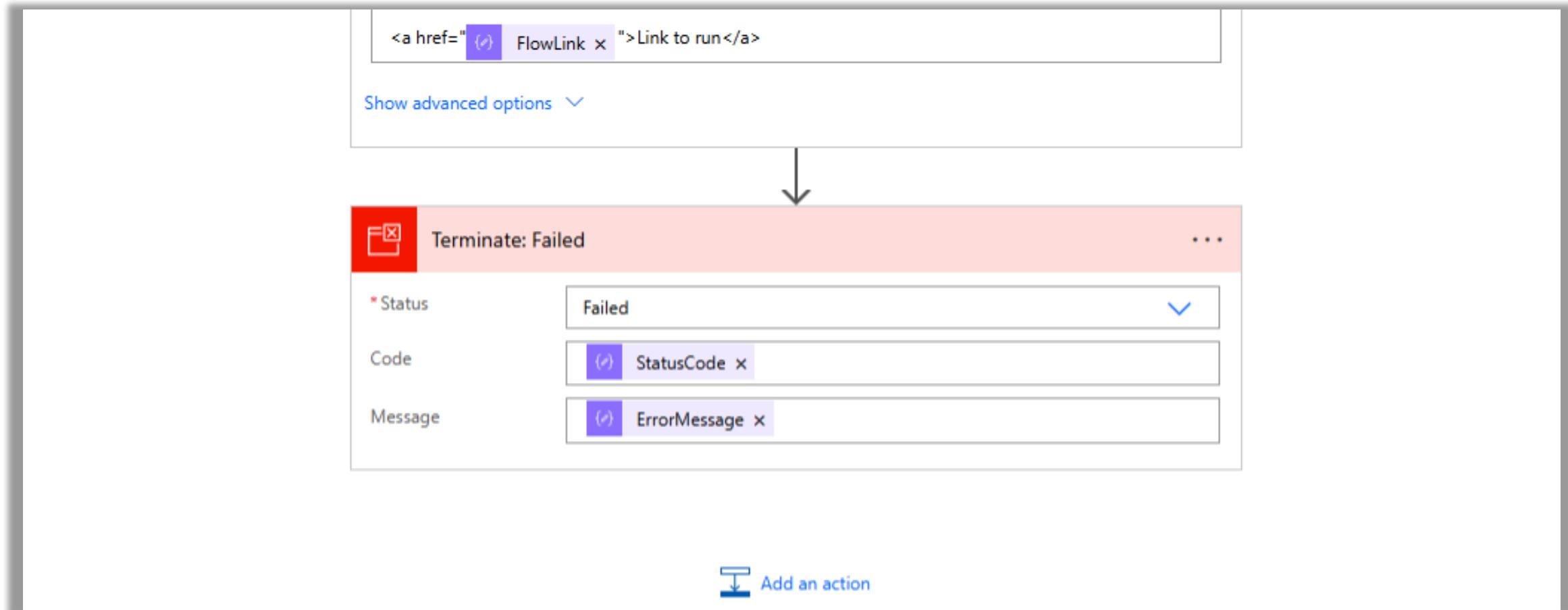
```
<br/>
<p>This is the data submitted to the action</p>
<table>
<tr><th>Content</th></tr>
<tr><td>{&gt; Content &lt;/td></tr>
</table>
```

At the bottom of the 'If yes' section, there is a blue 'Add an action' button. To the right, the 'If no' path is shown with a red X icon and a blue 'Add an action' button.

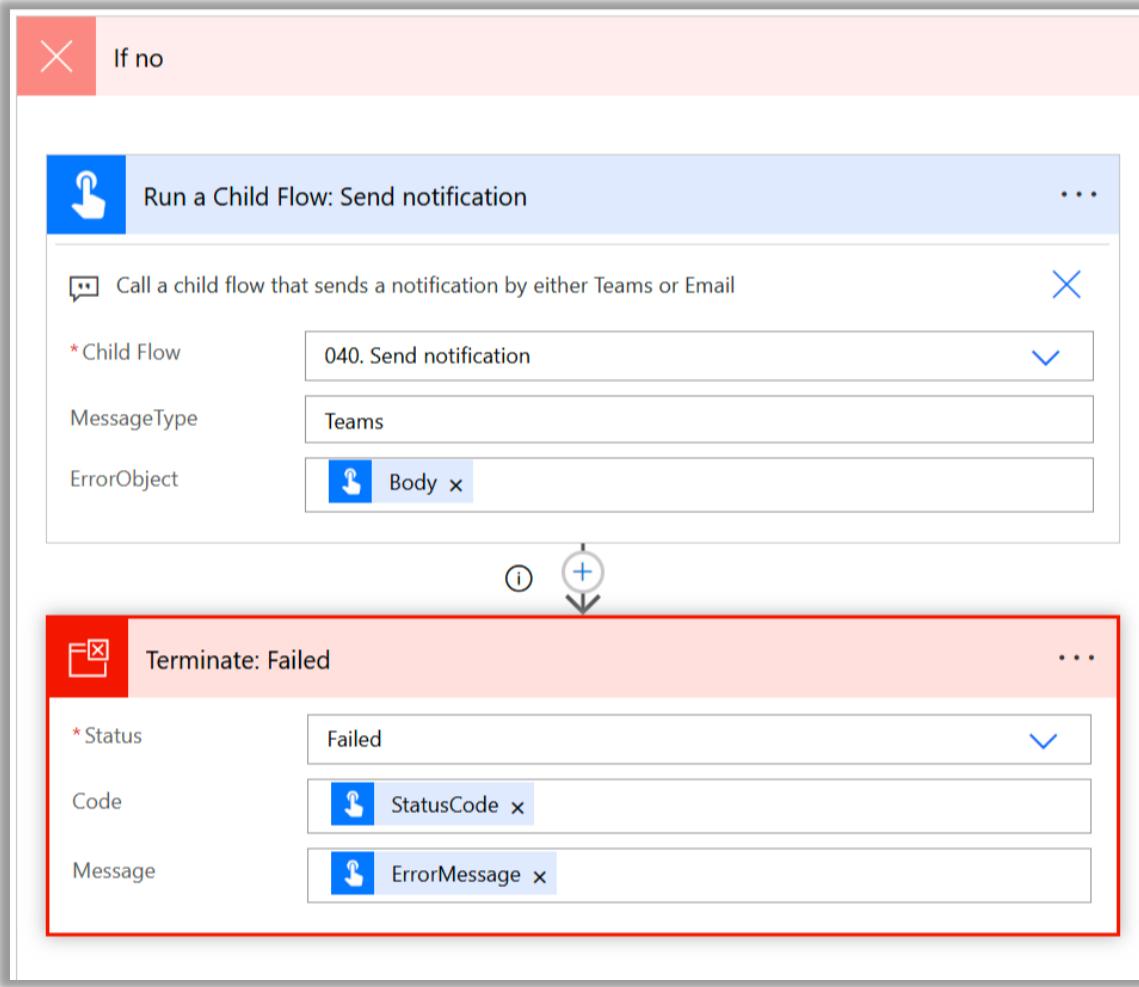
FINALLY SCOPE – Send Teams message



FINALLY SCOPE – Terminate as failed



Child flow for sending notification



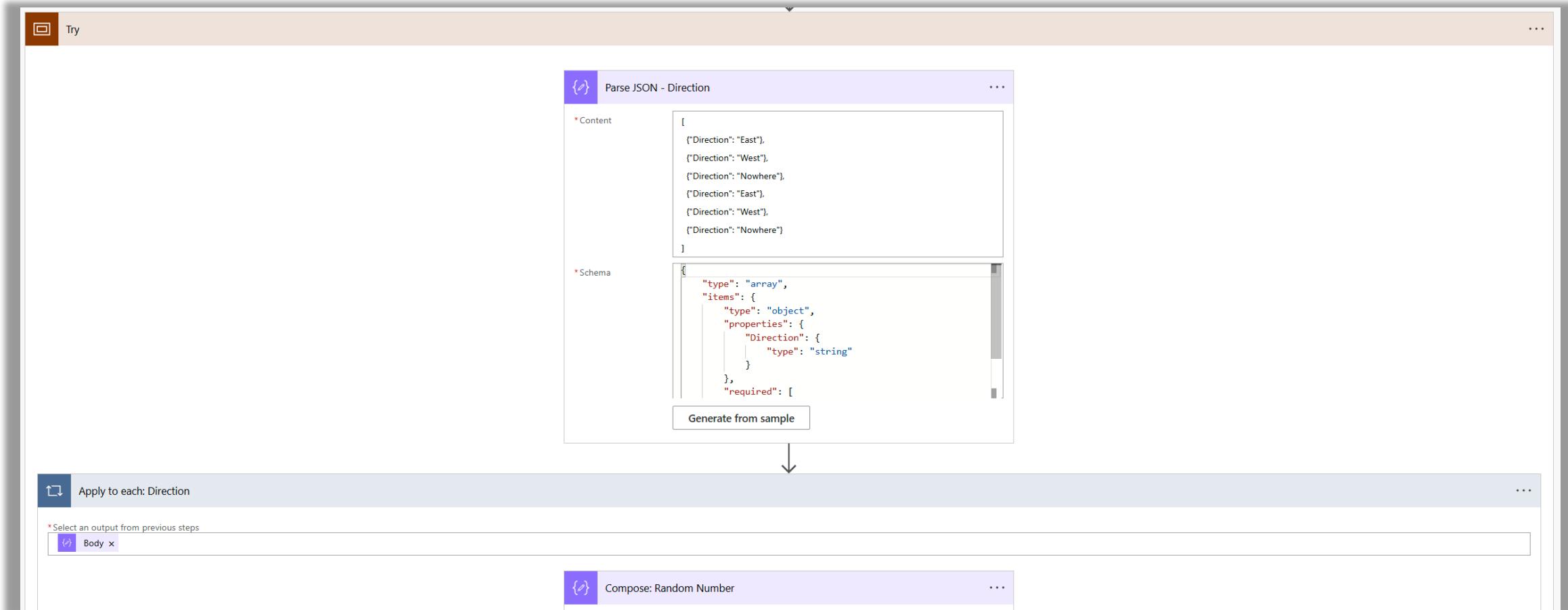
This far

- How to setup Try-Catch-Finally
- How to extract the error message and related details
- Use dynamic content to create custom error messages

What is not covered in this kind of error handling

- Loops and conditions are difficult to handle

Conditions and nested actions



Add a JSON array

The screenshot shows a step in the Microsoft Power Platform canvas editor titled "Parse JSON - Direction". The step has two main sections: "Content" and "Schema".

Content:

```
[{"Direction": "East"}, {"Direction": "West"}, {"Direction": "Nowhere"}, {"Direction": "East"}, {"Direction": "West"}, {"Direction": "Nowhere"}]
```

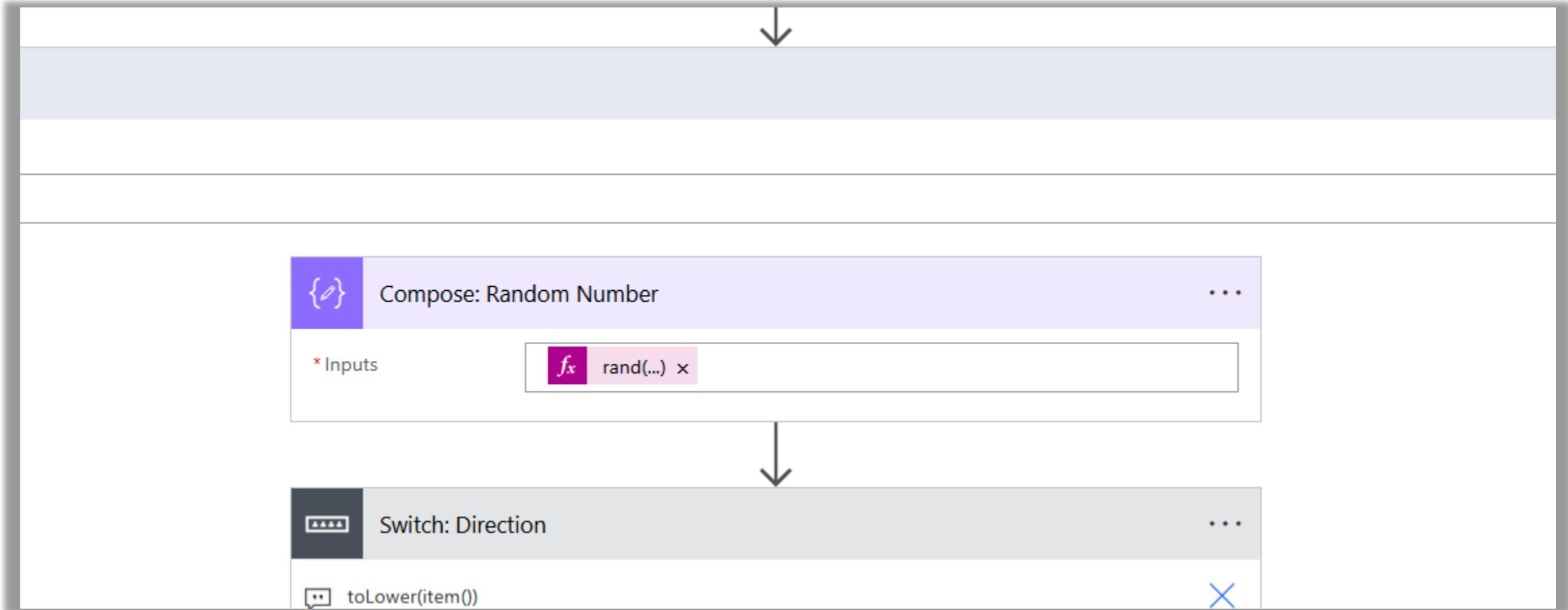
Schema:

```
{
  "type": "array",
  "items": {
    "type": "object",
    "properties": {
      "Direction": {
        "type": "string"
      }
    }
  }
}
```

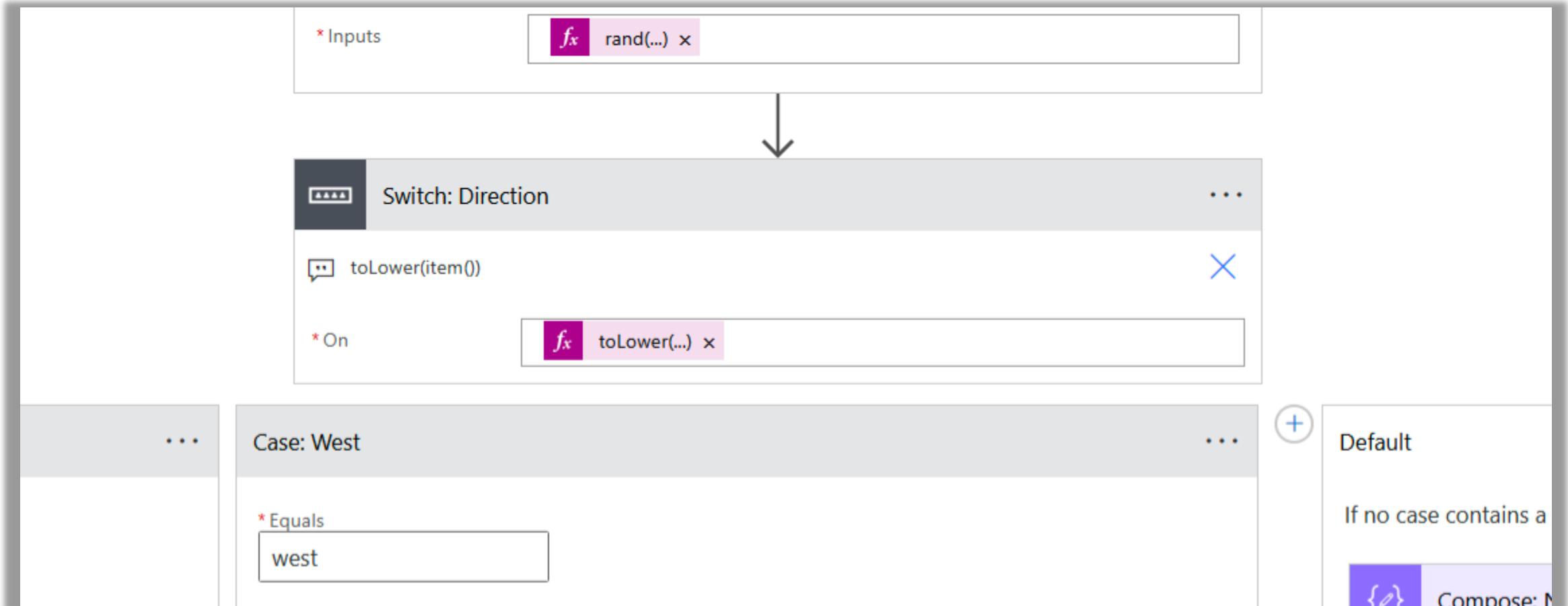
Apply to each element of the JSON array

The screenshot shows a 'Power Automate' interface step titled 'Apply to each: Direction'. A blue icon with a double arrow is on the left. Below it, a red asterisk indicates a required field: 'Select an output from previous steps'. A purple button labeled 'Body' with a brace icon is highlighted.

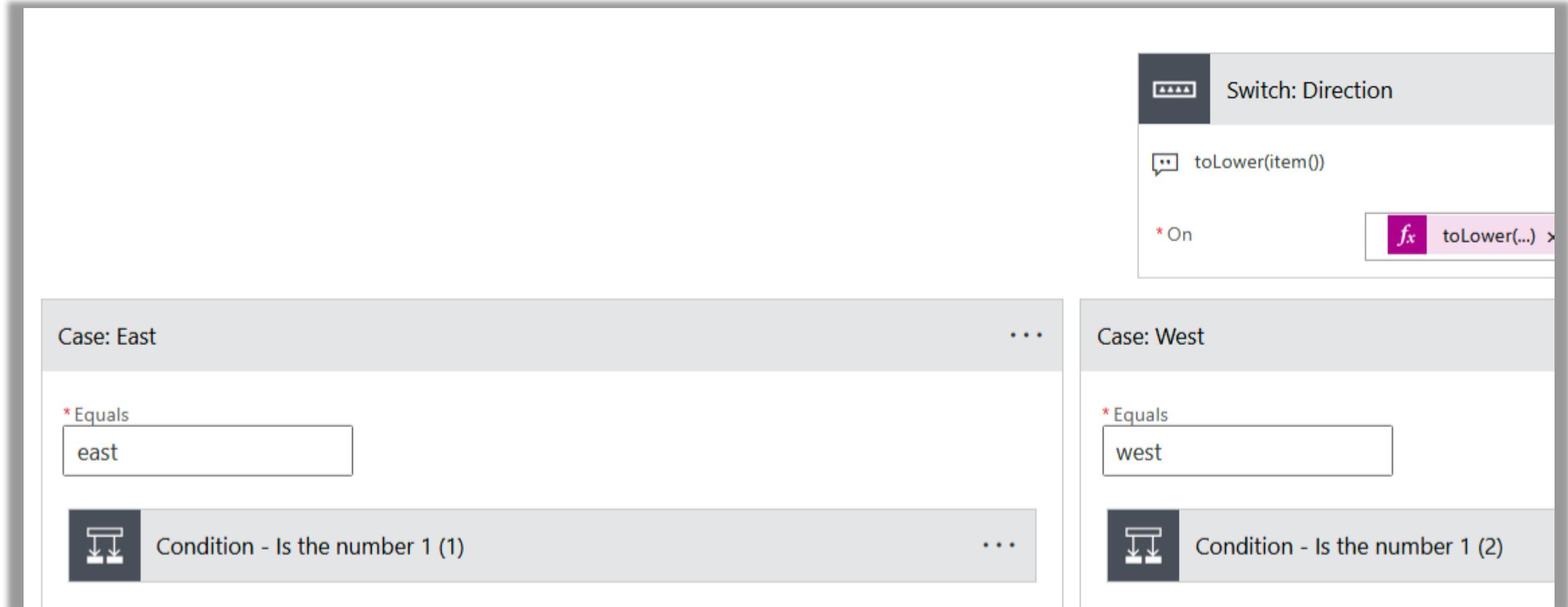
Get a random number



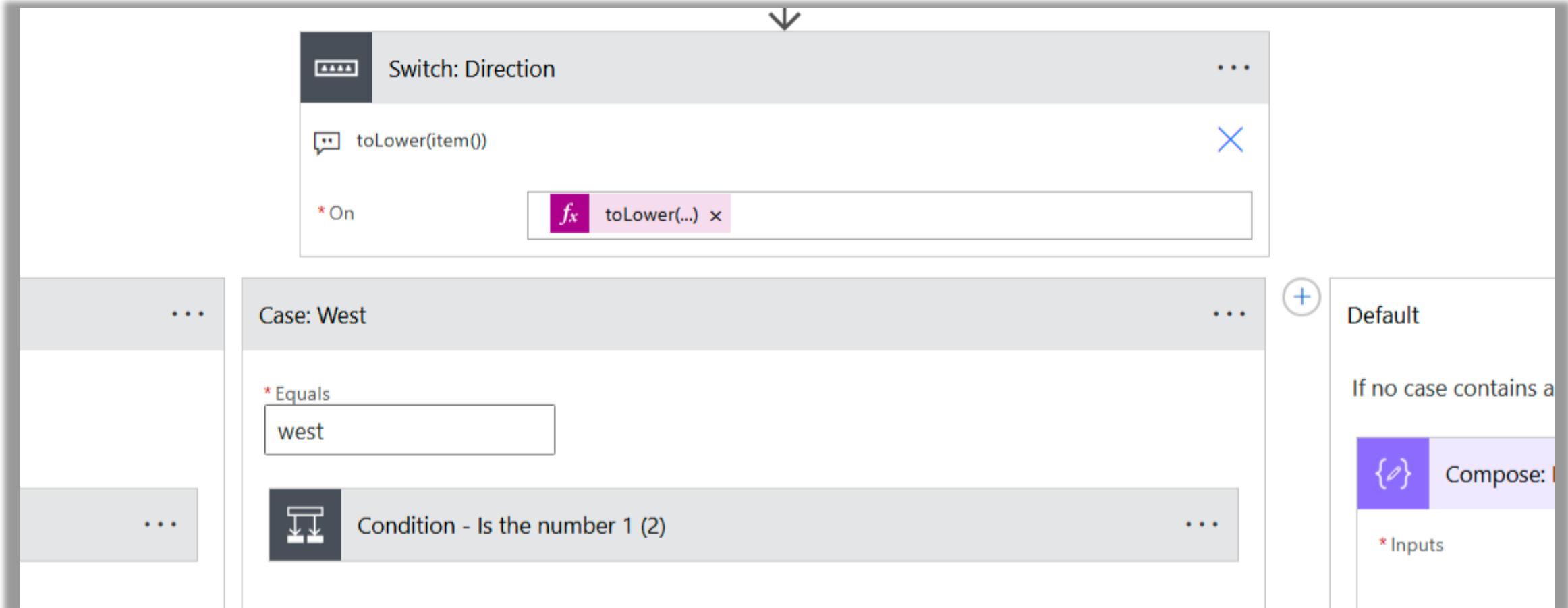
Switch on the current item of the for each loop



First east



Then west



Or nowhere

The screenshot shows a Power Automate flow editor. On the left, there's a vertical list of actions with three ellipsis buttons. In the center, a 'Default' section is expanded, containing the following text: 'If no case contains a matching value'. Below this is a purple 'Compose: Nowhere' action card. The card has a purple header with the action name and a purple '...' button. Underneath is a white area labeled 'Inputs' with a red asterisk. A text input field contains the message 'I am going nowhere / Let's error on that'. At the bottom right of the card is a small 'fx' icon and the text 'string()' followed by a delete 'x' icon. At the very bottom of the card is a blue 'Add an action' button with a plus sign and a downward arrow.

Default

If no case contains a matching value

{ } Compose: Nowhere

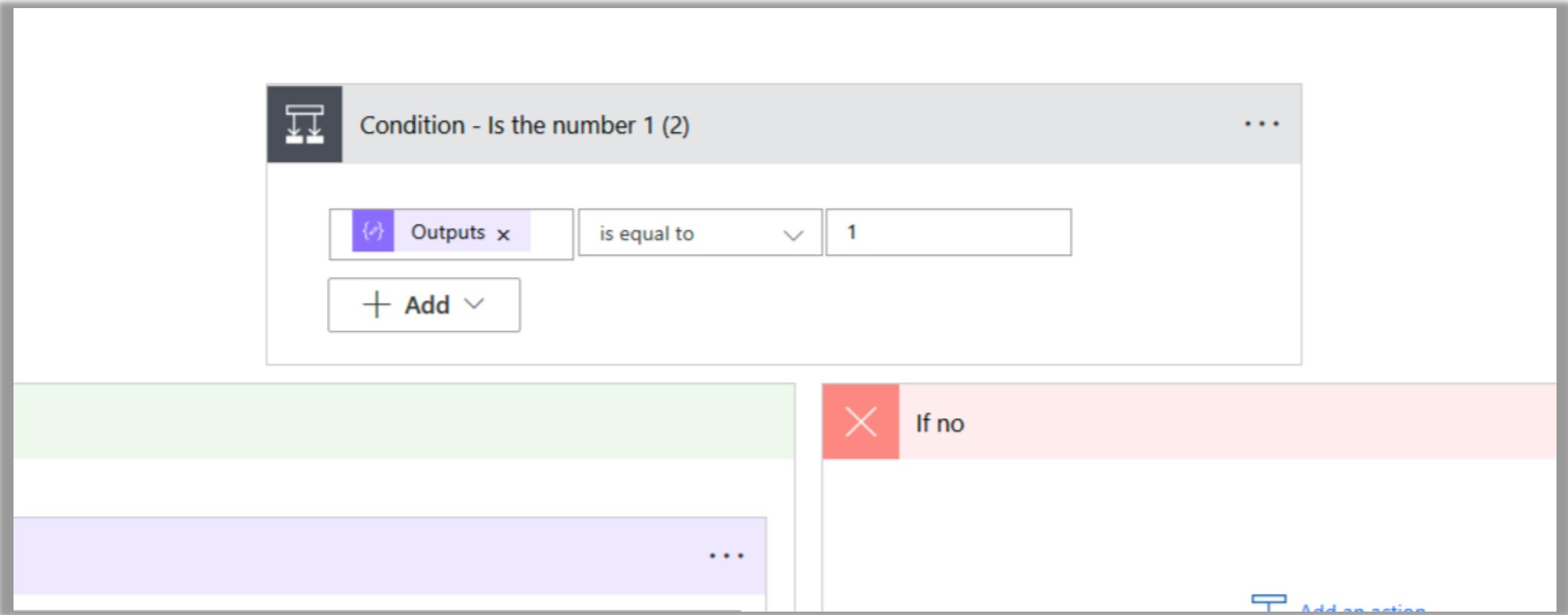
* Inputs

I am going nowhere / Let's error on that

fx string() x

Add an action

Inside the switch is a condition



And inside the condition is a Compose

The screenshot shows a Power Automate flow editor. At the top, there is a condition step with the following configuration:

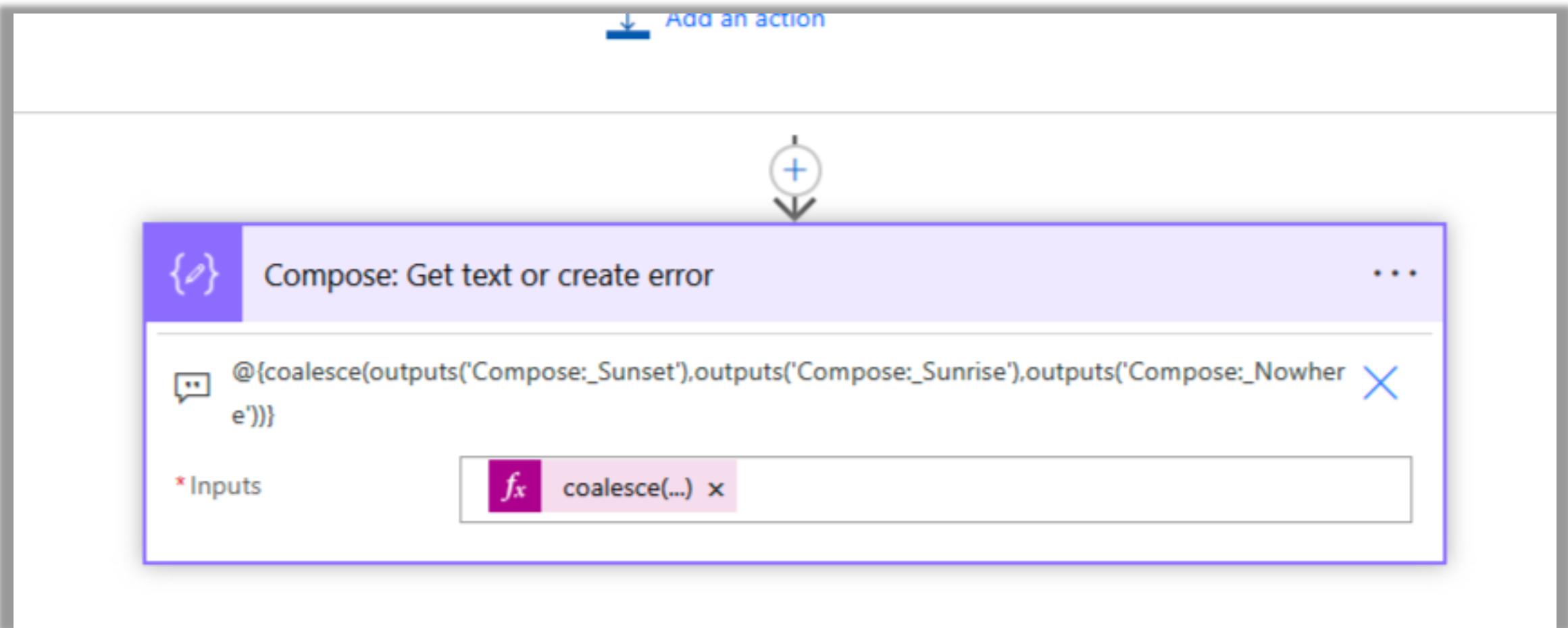
- Condition: Outputs (purple icon) is equal to (dropdown menu)
- Action: + Add (button)

The main body of the flow contains two branches:

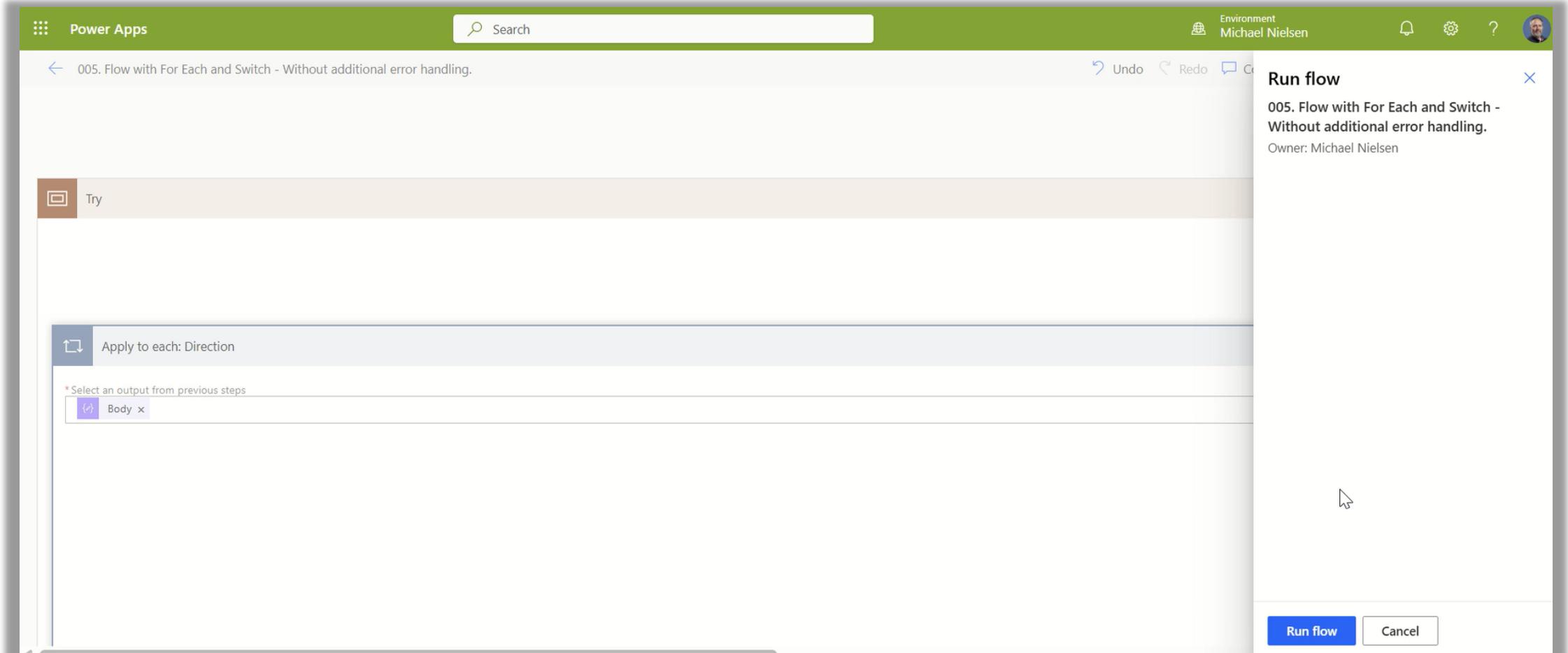
- If yes (Green Branch):** Contains a Compose action step named "Compose: Sunset".
 - Inputs:
 - * Inputs: My direction is toward the sunset
- If no (Red Branch):** Contains an empty action step.

At the bottom of the flow, there is a button labeled "Add an action".

And finally a compose to get the text from either branch



Lets try to run the flow with nested actions



An action failed. No dependent actions succeeded.

An error occurred within the flow:

005. Flow with For Each and Switch - Without additional error handling.

Action	Error Message
Apply_to_each:_Direction	An action failed. No dependent actions succeeded.

This is the data submitted to the action

Content
{"name":"Apply_to_each:_Direction","inputs":{"foreachItems":[{"Direction":"East"}, {"Direction":"West"}, {"Direction":"Nowhere"}, {"Direction":"East"}, {"Direction":"West"}, {"Direction":"Nowhere"}]}, "inputsMetadata": {"foreachItemCount":6}, "startTime":"2025-06-18T22:51:23.3137558Z", "endTime":"2025-06-18T22:51:26.1082844Z", "trackingId": "f3f6430d-da8c-44d5-a160-c4281fe962b9", "clientTrackingId": "08584513198026585602344733126CU216", "clientKeywords": ["testFlow"], "code": "ActionFailed", "status": "Failed", "error": {"code": "ActionFailed", "message": "An action failed. No dependent actions succeeded."}}

[Link to run](#)

Lets look at it live

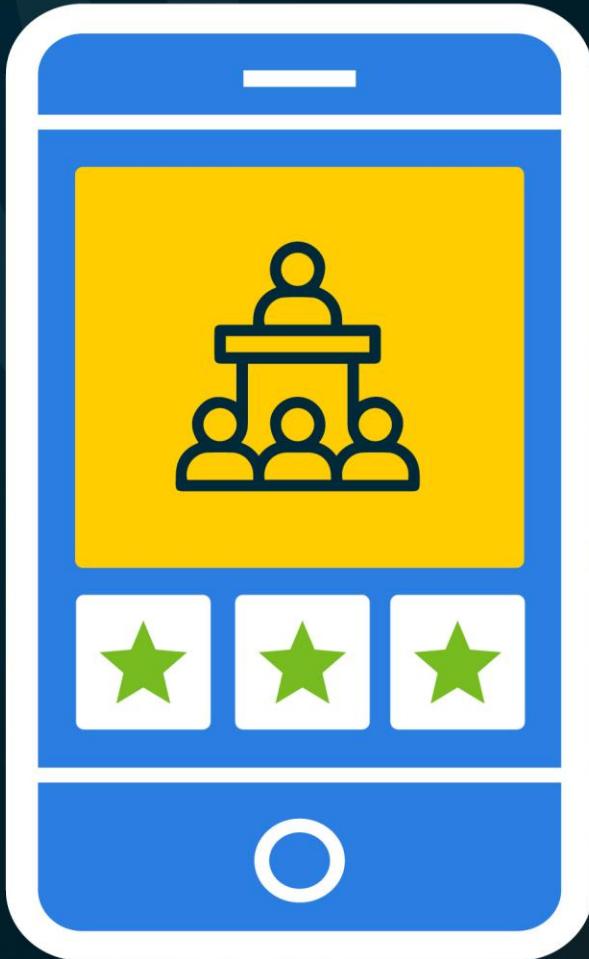


Lets connect





Please rate
this session
on the app



cvent

