

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

**Nicolai Schjørmann**

**Business Unit Manager  
Abakion A/S, Denmark**

**Michael Nielsen**

**Senior Power Platform Consultant  
Abakion A/S, Denmark**



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

**Nicolai Schjørmann  
Business Unit Manager  
Abakion A/S, Denmark**



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

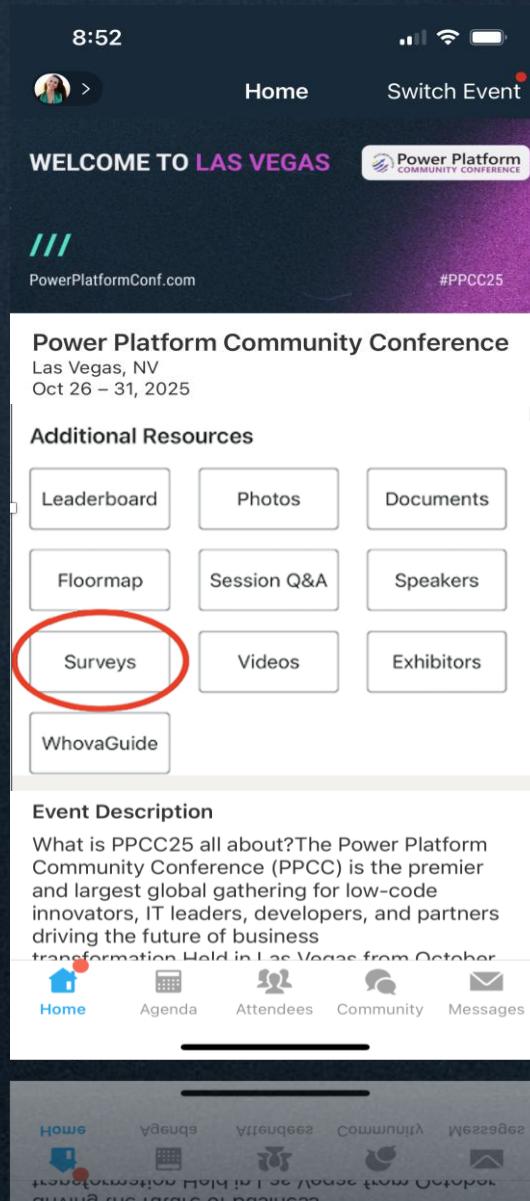
**Michael Nielsen**  
**Senior Power Platform Consultant**  
**Abakion A/S, Denmark**



But first...

Socialtime

# Session Feedback Surveys



**We really want to hear from YOU!**

*In the pursuit of making next year's Power Platform Community Conference even better, we want to hear your feedback about this session.*

**Here's How -**

- *Simply go to the Whova App on your smartphone*
- *Scroll down on the Power Platform Community Conference Homepage to 'Additional Resources' to click "Surveys"*
- *Click Session Feedback.*
- *Scroll down to find this session title.*
- *Complete the session feedback survey.*
- *Finally, click 'Submit'*
- *It's just that easy!*

Let us try to imagine a scenario  
together...

# Power Automate **ERROR HANDLING**





Friday afternoon, the company gets their new system live.





**But then an error occurs...**

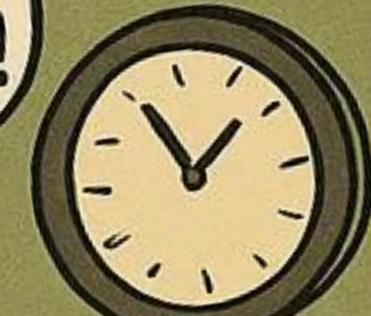
**Our system  
is down!  
Data errors  
everywhere!**







Oh no! I need to  
call the consultant!













We need a more robust system that can identify issues quickly.

## Lesson learned

-  Error management
-  Notifications
-  Proactive monitoring
-  Logging

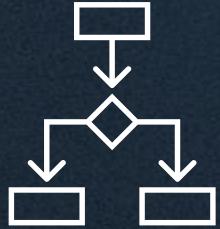
## The Lesson:

Not enough for  
to work—they  
ed to be built for  
the scale, security, and  
resilience that  
enterprises demand.



ROB  
ERRO  
MO

# Scope: We'll explore how to elevate your flows



Design robust error handling to make your flows inform you what went wrong.



implement real-time monitoring that keeps you ahead of failures instead of reacting after the damage is done.



optimize your flows for performance and scalability, so they run smoothly no matter how complex business grows.

# Why error handling matters

## Business disruptions

- Workflow chaos
- Approval delays
- Unseen interruptions

## Data inconsistency

- Asymmetrical sync
- Data incomplete
- Poor data

## User frustration

- Confusion
- Missing notifications
- Manual work

## Hidden failures

- No logs = no clues
- Failures in the shadow
- Quiet breakage

# Common pitfalls



## Run after

Not configure “Run After” conditions



## API status

Assuming APIs and connectors always succeed



## Retry policies

No “Retry policies” configured



## Centralized management

No centralized monitoring or alerting

# Our solution

**Try**

Try scope including your logic

**Catch**

Catch scope to catch and collect potential errors

**Finally**

Finally scope to finish your flow

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

# Interactive demo - Case



# Interactive demo - Purpose

!

Understand the  
importance of error  
handling

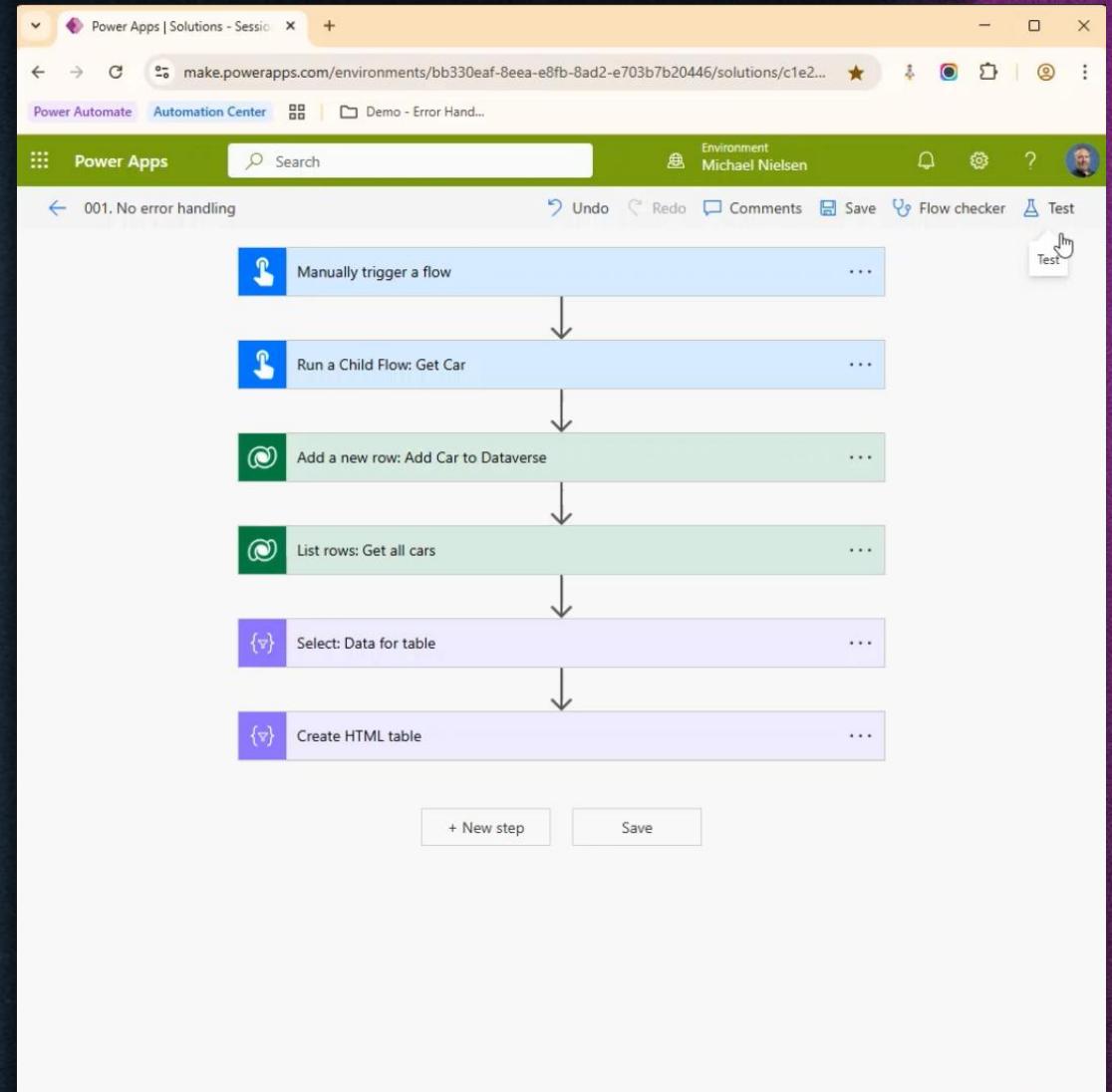
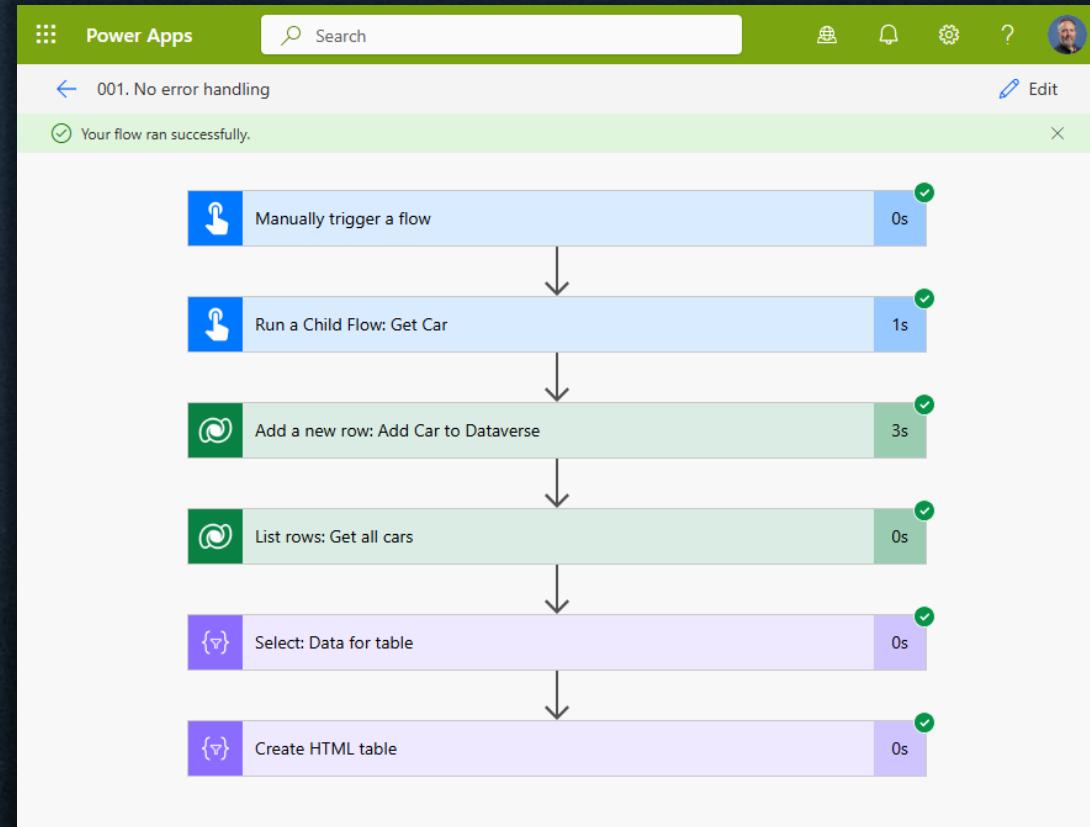


Learn how to implement  
Try-Catch-Finally logic



Implement custom error  
messages and  
notifications

# Flow without error handling



# Who is going to know?



# How will you know if a flow fails

You can see it in the flow run history

28-day run history ⓘ		
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 (one a day)

Microsoft Power Automate

**! 5 of your flow(s) have failed**

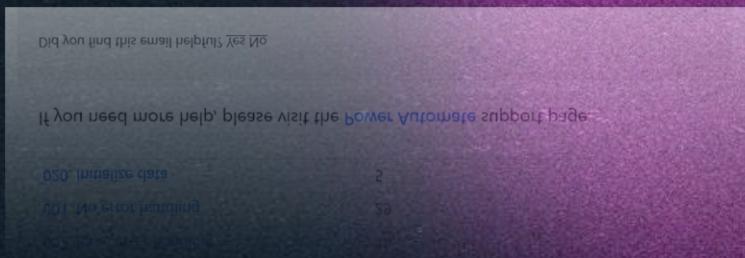
The flow(s) listed failed in the past week and may need your attention.

**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](#)



# How will you know if a flow fails

28-day run history ⓘ			<a href="#">Edit columns</a>	<a href="#">⟳ All runs</a>
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		

(open) May 23:01, 2023	00:00:02	Test succeeded
(open) May 20:11, 2023	10:20:00	Test succeeded

# How will you know if a flow fails

28-day run history <small> ⓘ</small>	
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.

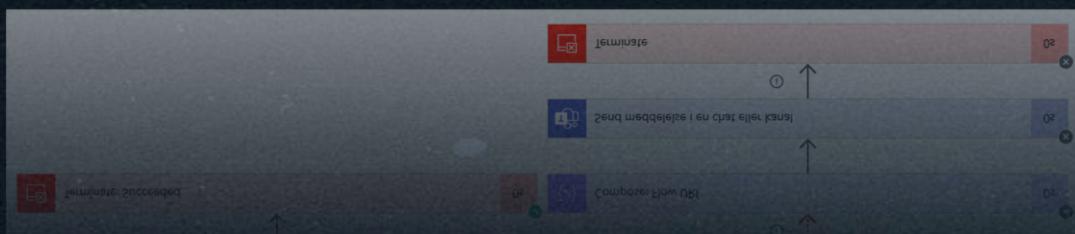
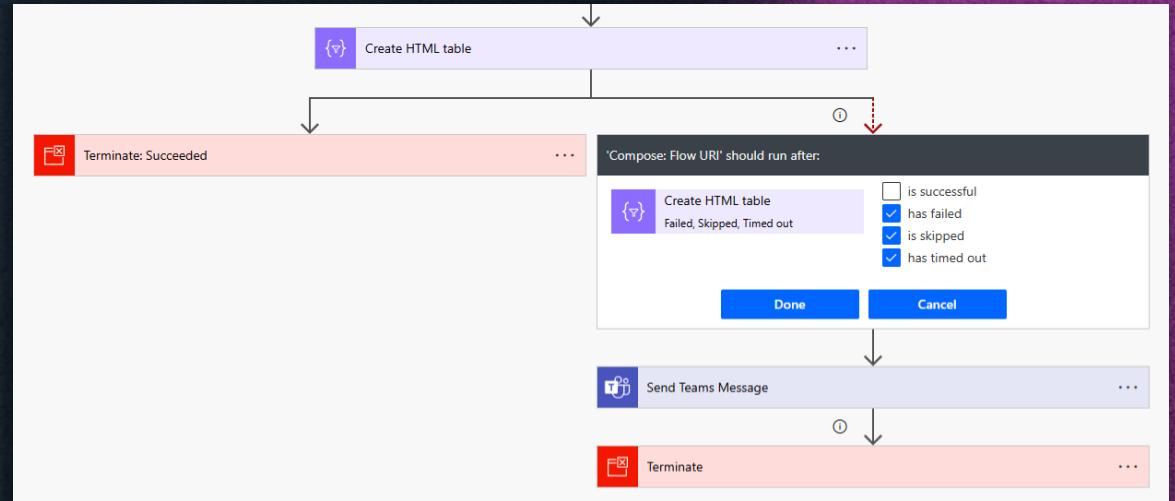
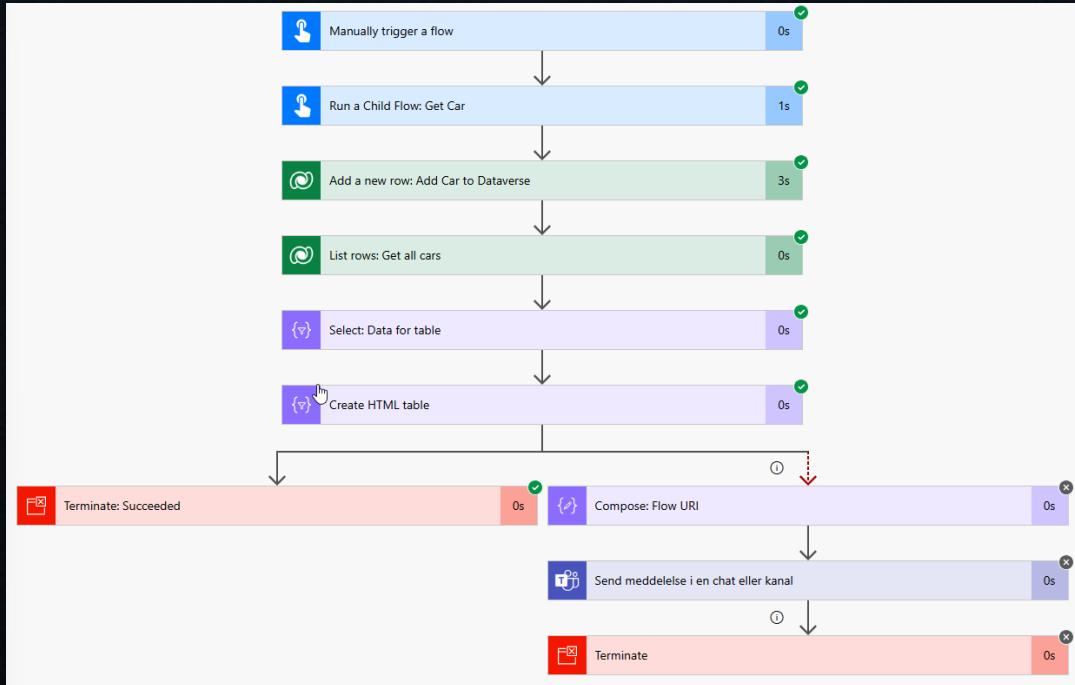
### 5 Notifications:

Flow name	Failure count
<a href="#">050. Get flow runs</a>	2
<a href="#">003. Advanced error handling</a>	49
<a href="#">002. Basic error handling</a>	18
<a href="#">001. No error handling</a>	29
<a href="#">020. Initialize data</a>	5

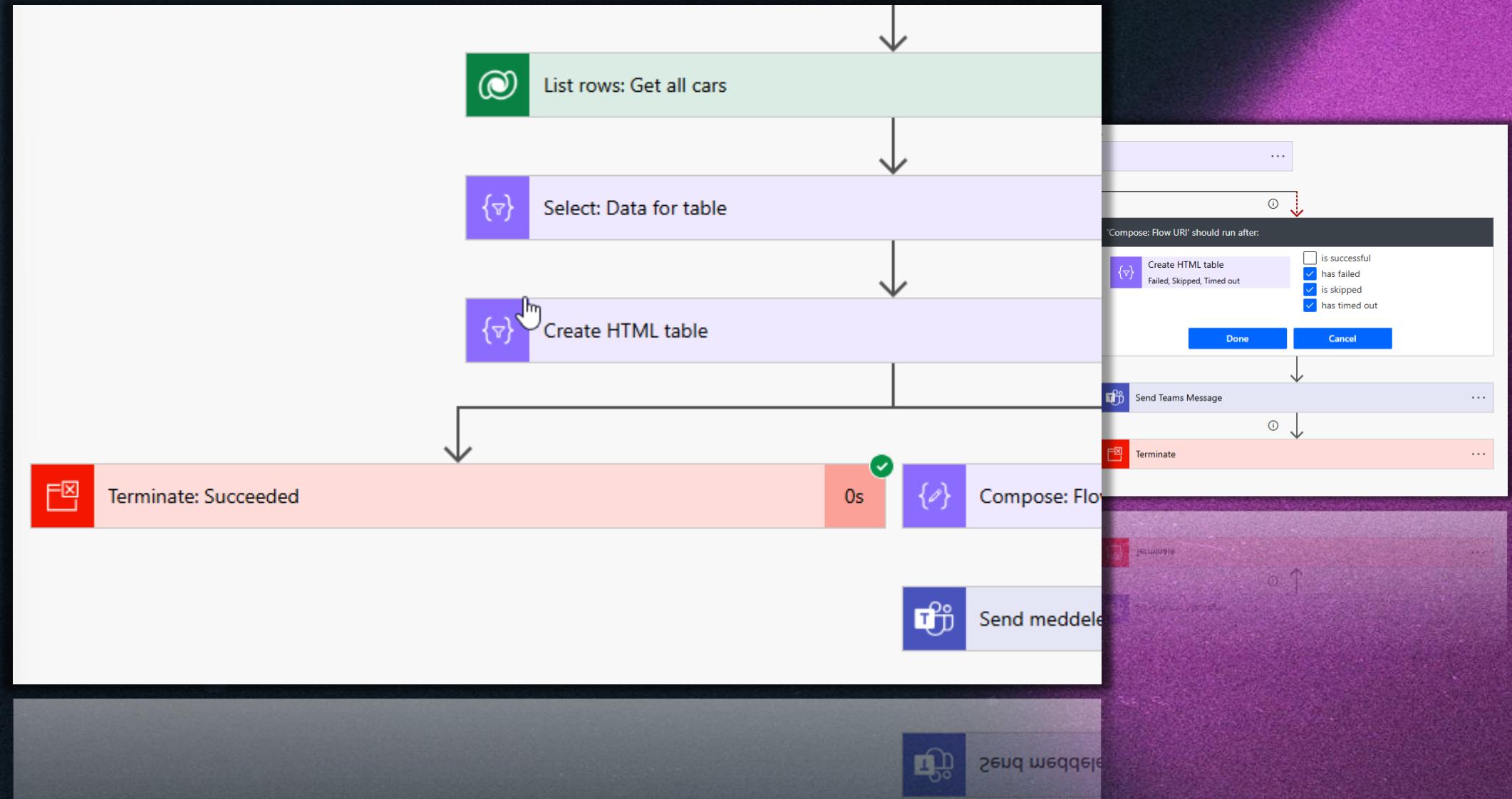
If you need more help, please visit the [Power Automate support page](#).

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

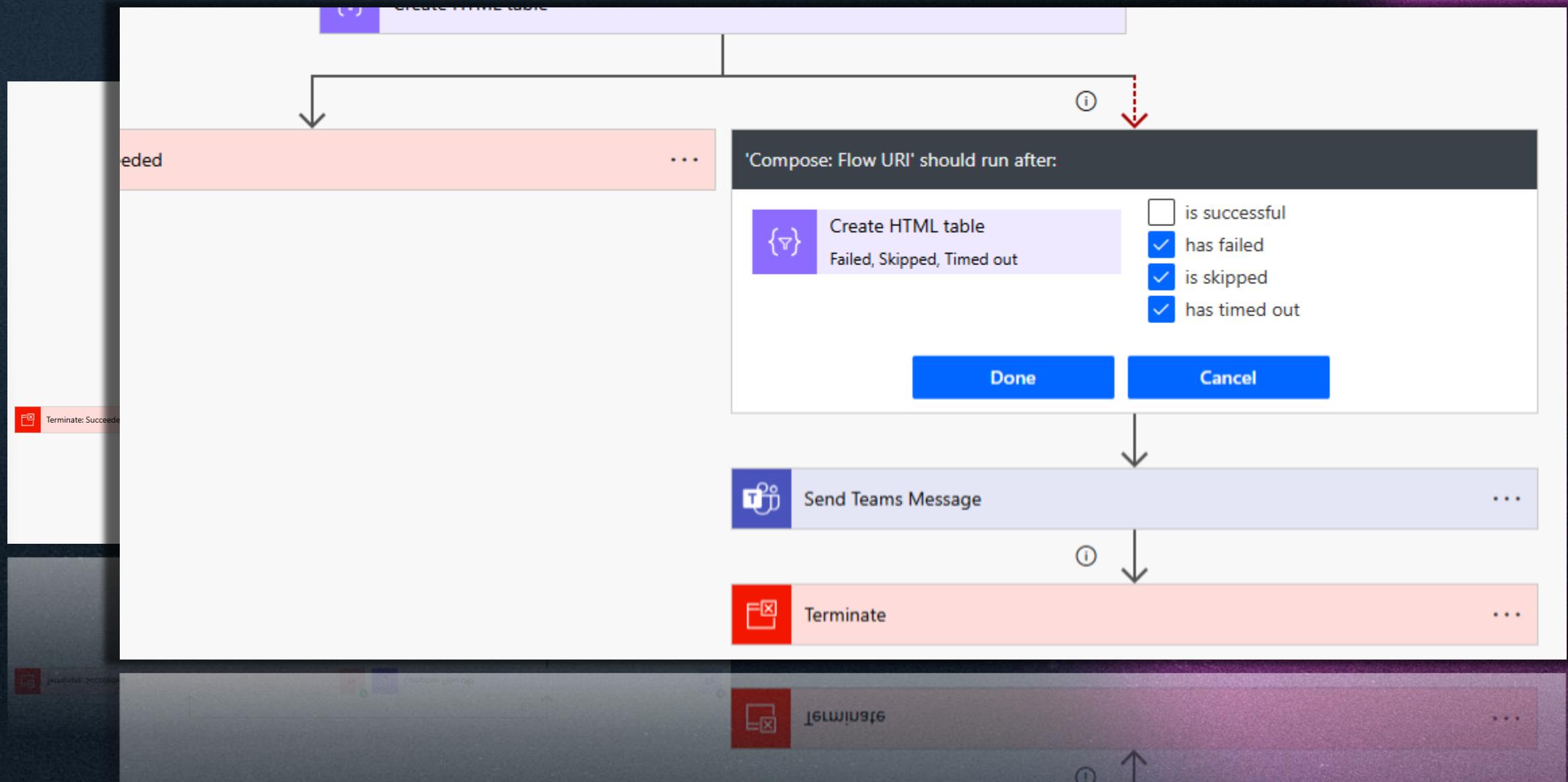
# Basic error handling – Parallel Actions



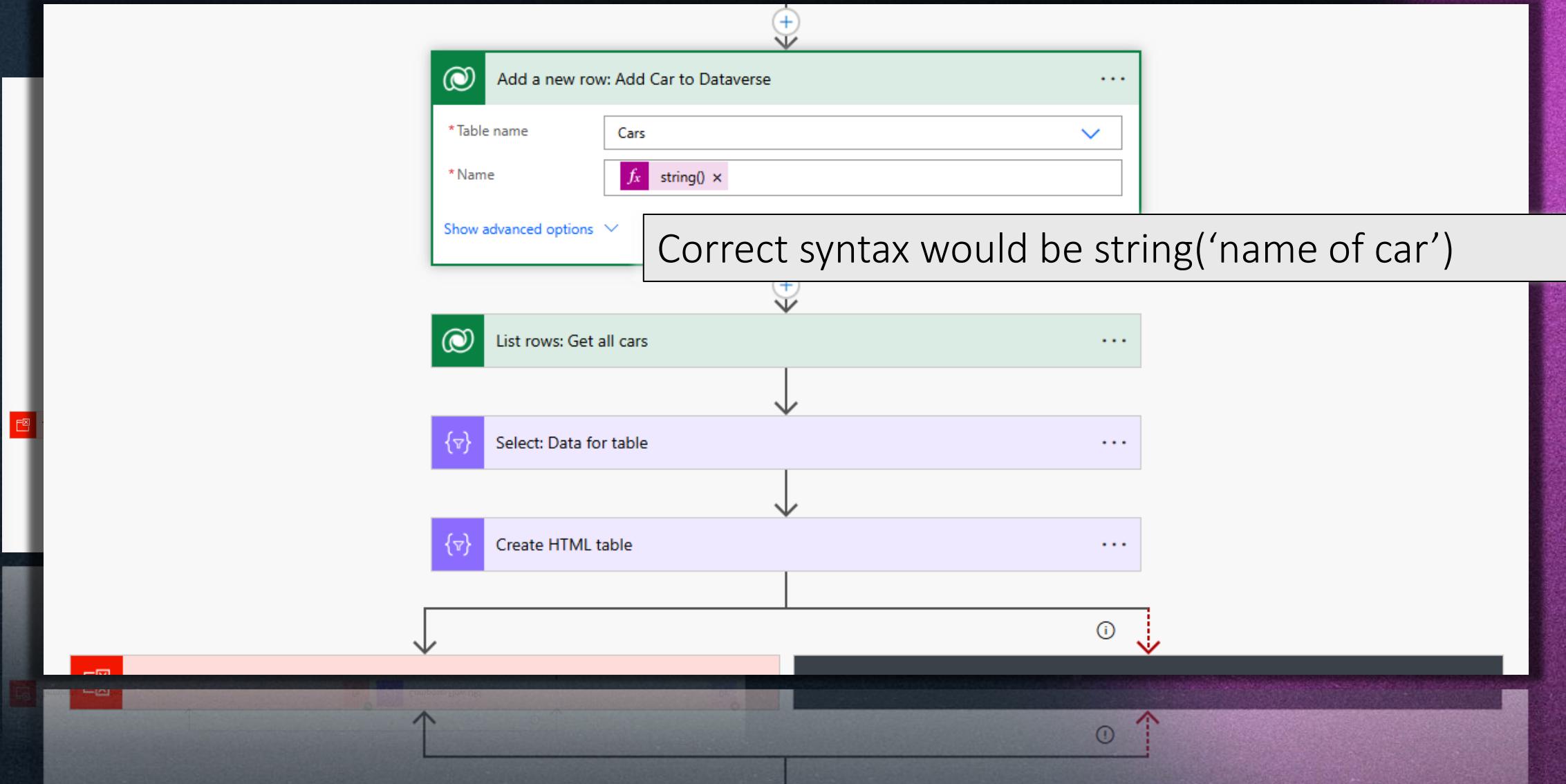
# Left Branch – Success Path

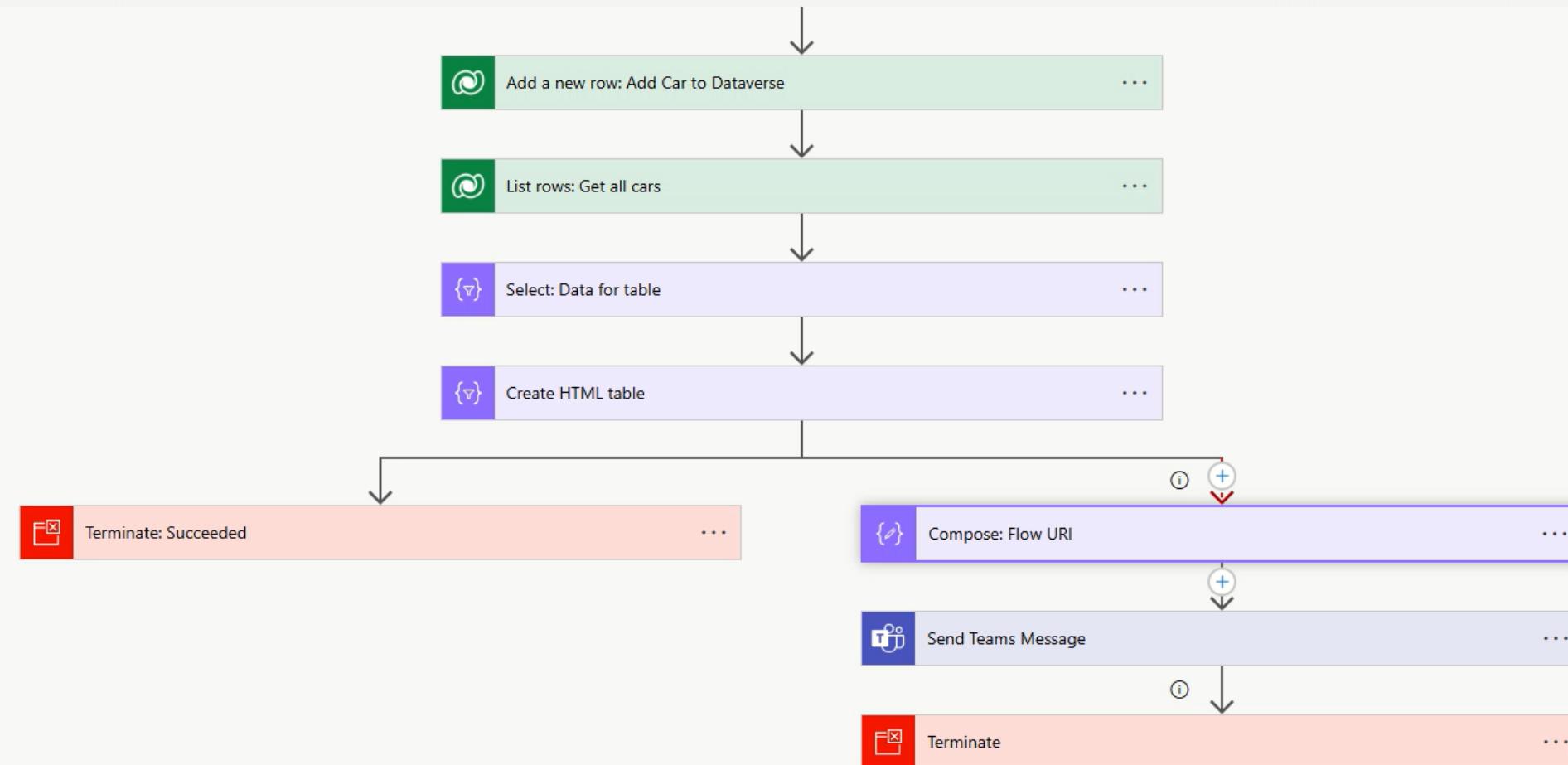


# Right Branch – Failure Path



# Syntax Error Example



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

aberrant

# Messages received after error handling

The screenshot shows a Microsoft Power Automate interface. At the top, there's a navigation bar with 'Workflows' selected. Below it, a message from 'Michael Nielsen via Workflows' at 23:28 says 'An error occurred in your flow:'. A detailed error message follows:

**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 <a href="https://aka.ms/logicexpressions#string for usage details.">https://aka.ms/logicexpressions#string for usage details.</a>

[Link to run](#)

### Run Details

Start time: May 19, 11:28 PM (7 sec ago) Duration: 00:00:03

Error: 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.>

# Flow name

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 <a href="https://aka.ms/logicexpressions#string">https://aka.ms/logicexpressions#string</a> for usage details.'

[Link to run](#)

Run in UI

# 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 <a href="https://aka.ms/logicexpressions#string">https://aka.ms/logicexpressions#string</a> for usage details.'

[Link to run](#)

Run in UI

# 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 <a href="https://aka.ms/logicexpressions#string">https://aka.ms/logicexpressions#string</a> for usage details.'

[Link to run](#)

Run in UI

# Direct link into my failed flow

Michael Nielsen via Workflows 23:28 Oversæt

An error occurred in your flow:

## 002. Basic error handling

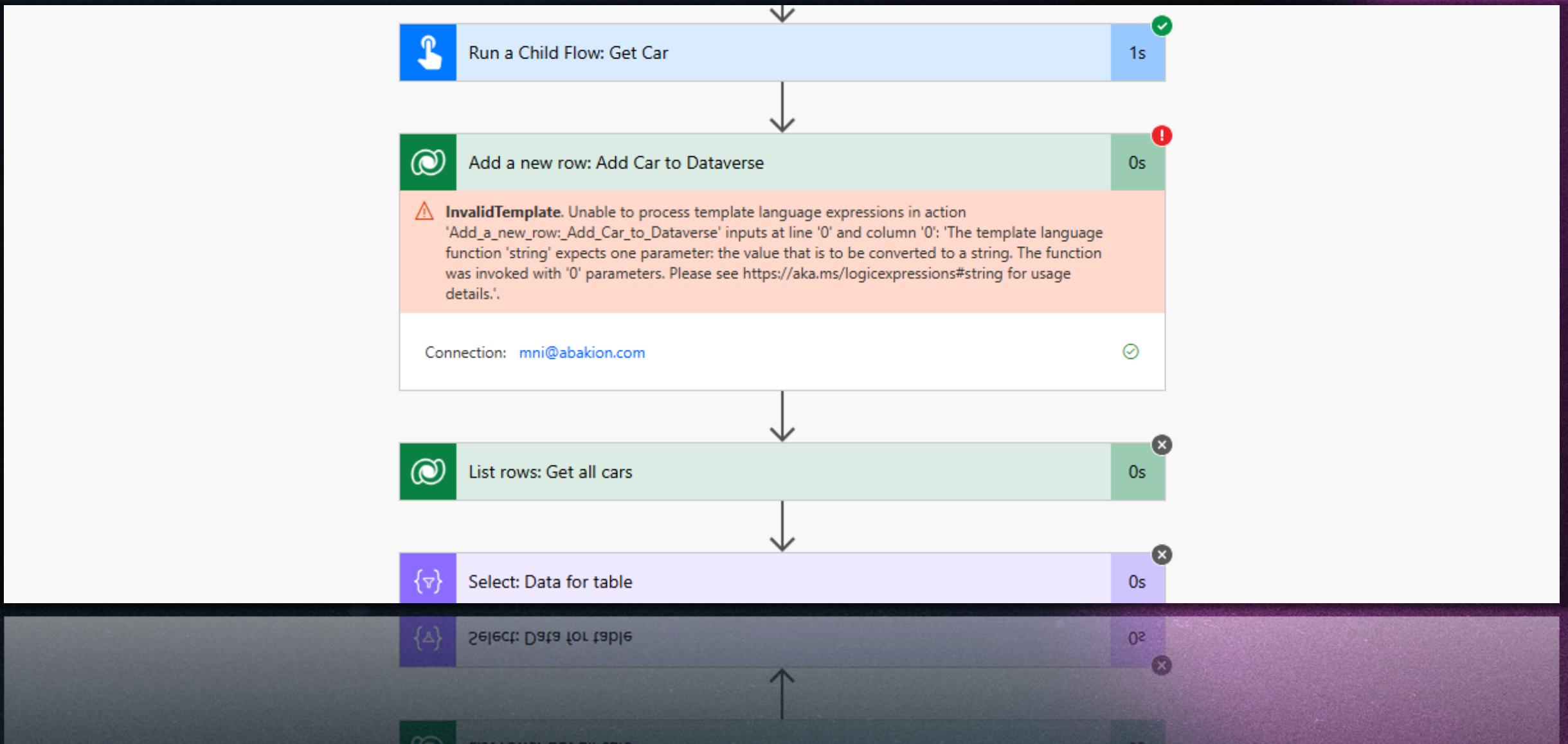
Action	Error
Add_a_new_row:_Add_Car_to_Dataverse	Unable to process template language expressions in action 'Add_a_new_row:_Add_Car_to_Dataverse' inputs at line '0' and column '0': 'The template language function 'string' expects one parameter: the value that is to be converted to a string. The function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

[Link to run](#)

Link to fail

asseleq statement 0, div below in sev  
not found at 0, prints a to batch and to  
is left unequal statement and exceeds  
prints, not found repeatable statement  
with a repeatable, and is around  
with a repeatable, and is around

# Error message within flow



# Create the URL for the actual flow run

The screenshot shows the 'Compose: Flow URI' action configuration screen in the Microsoft Power Automate designer. The action has a duration of 0s and is currently running.

**Inputs:**  
The 'Inputs' section contains a single input URL:  
`https://make.power automate.com/environments/bb330eaf-8eea-e8fb-8ad2`

**Outputs:**  
The 'Outputs' section contains a single output URL:  
`https://make.power automate.com/environments/bb330eaf-8eea-e8fb-8ad2`

A red dashed arrow points from the top center of the slide towards the 'Compose: Flow URI' action card.

Is this content helpful?

# Send the Teams message

Send Teams Message 1s

**INPUTS**

Post as  
Flow bot

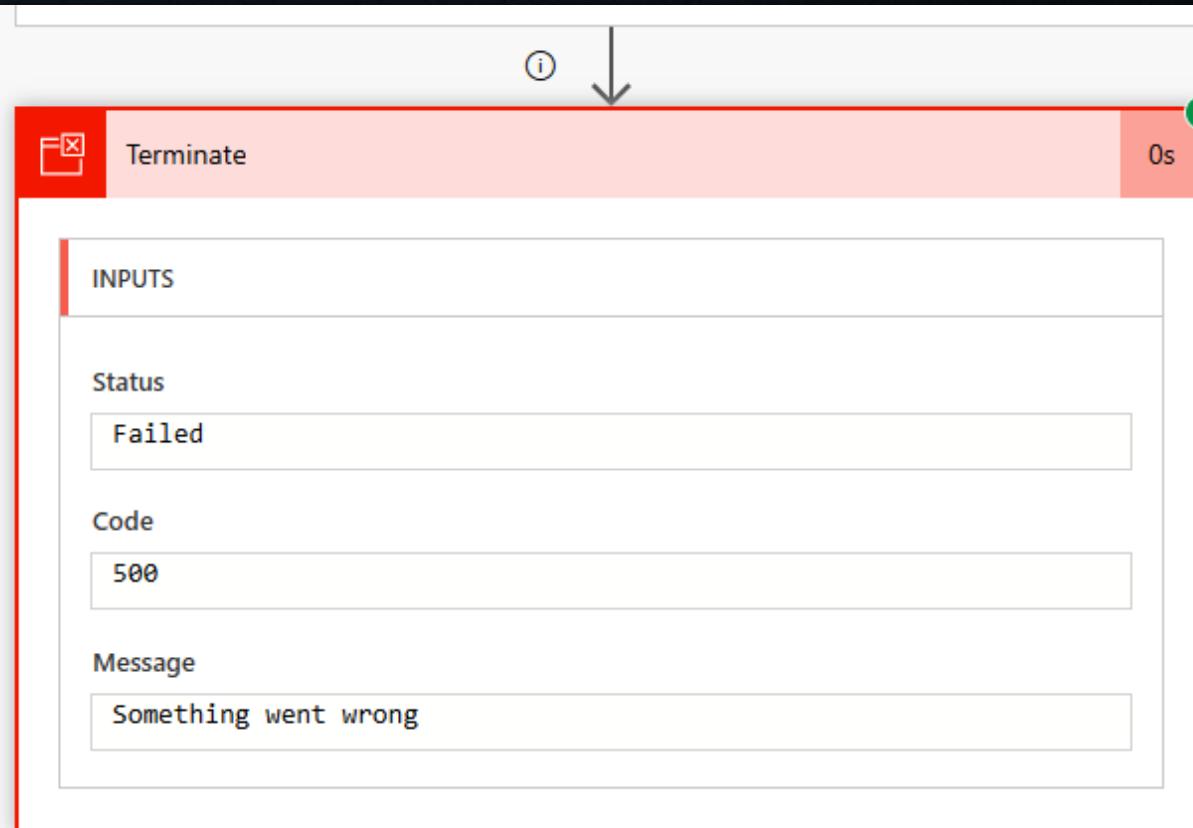
Post in  
Chat with Flow bot

Recipient  
mni@abakion.com;

Message

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

↓

{\Downarrow} Select: Data for table 0s !

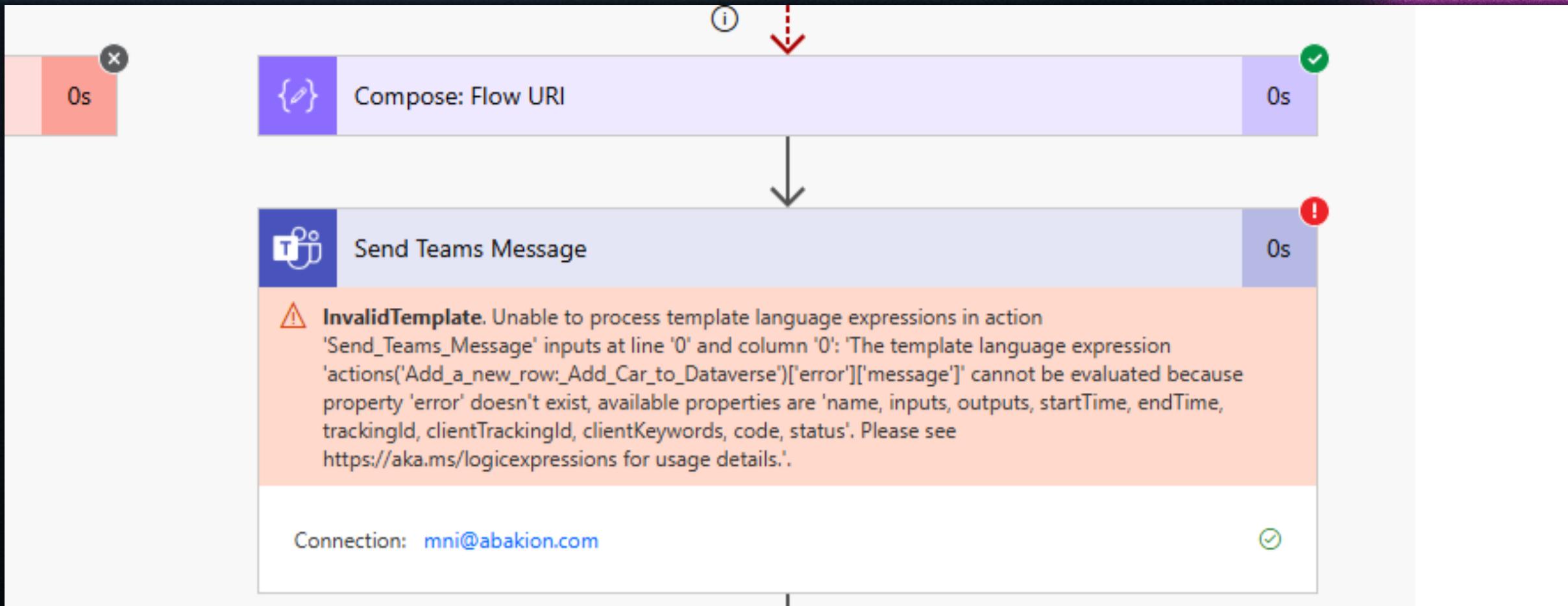
⚠ BadRequest. The 'from' property value in the 'select' action inputs is of type 'String'. The value must be an array.

INPUTS

From

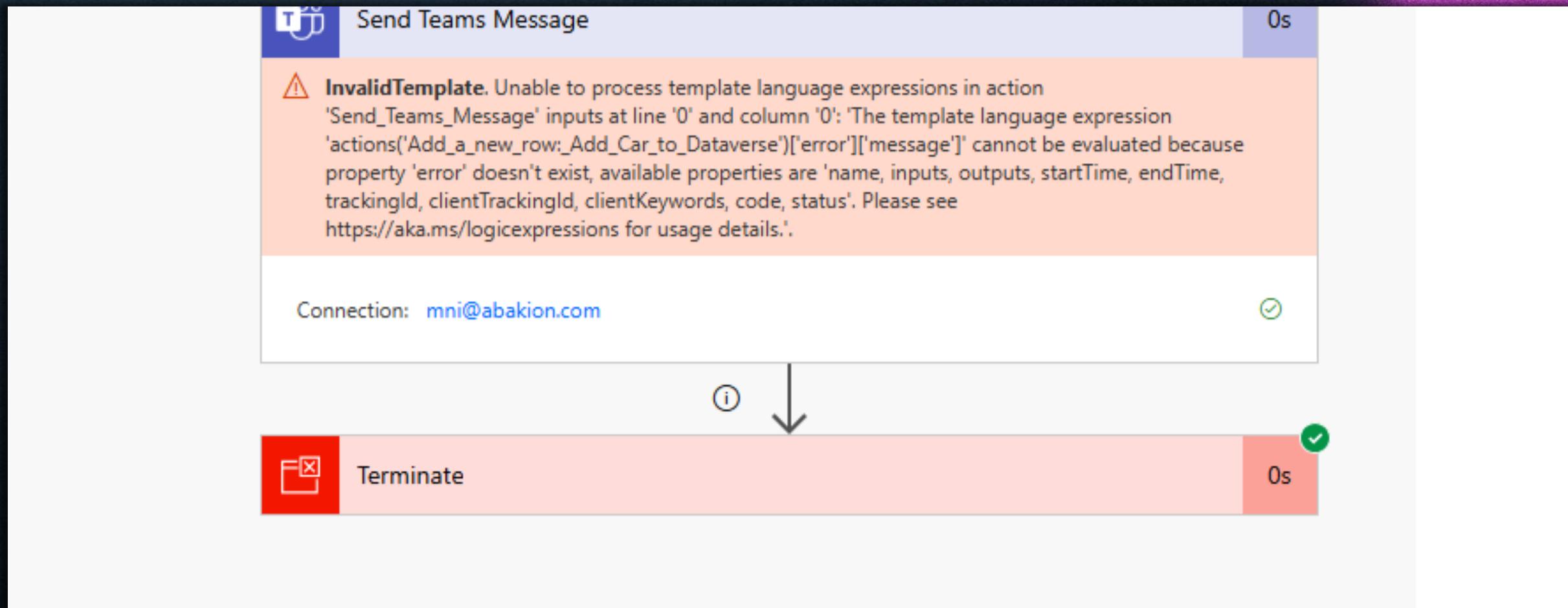
```
This is bad data
[{"@odata.type": "#Microsoft.Dynamics.CRM.mni_session_errorhandling"}]
```

# Basic error handling



Connection: mni@abakion.com

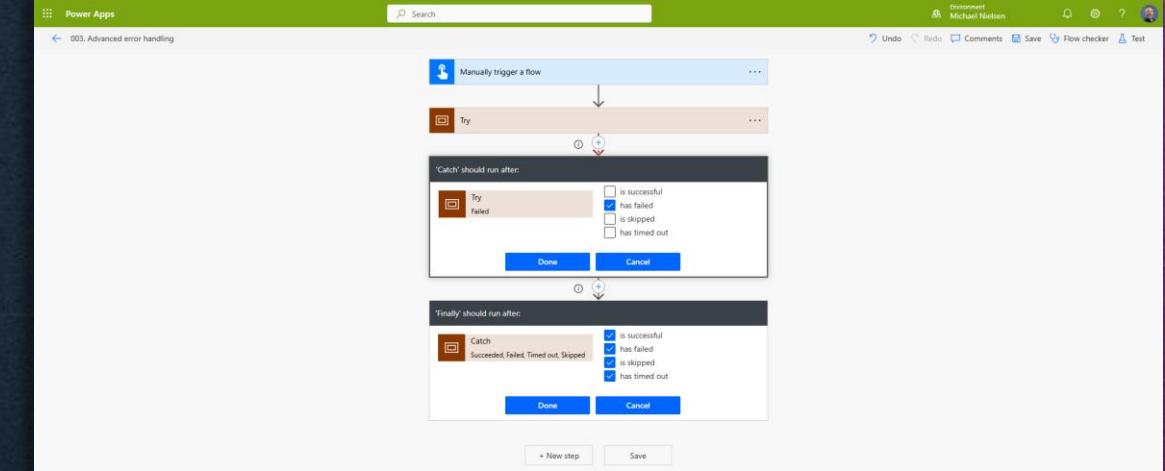
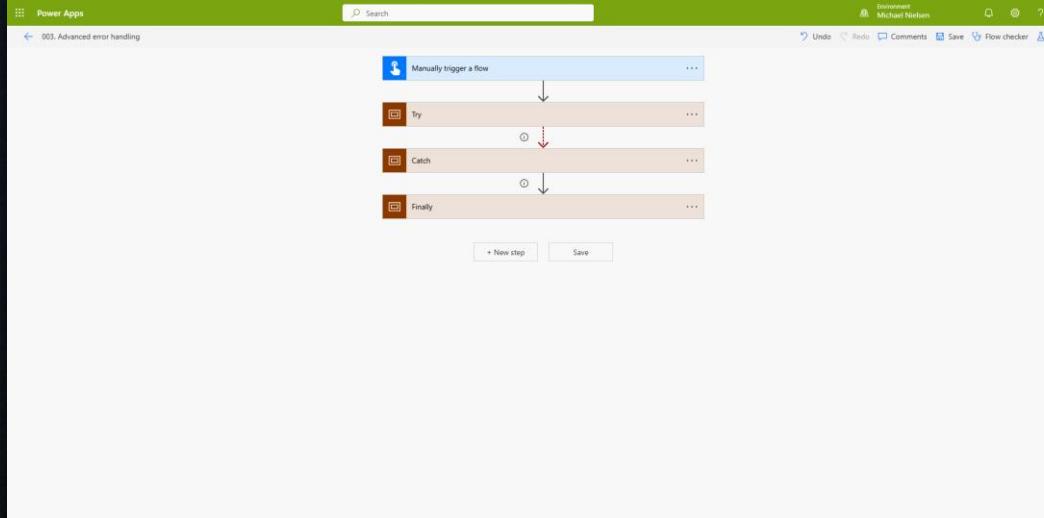
# Basic error handling



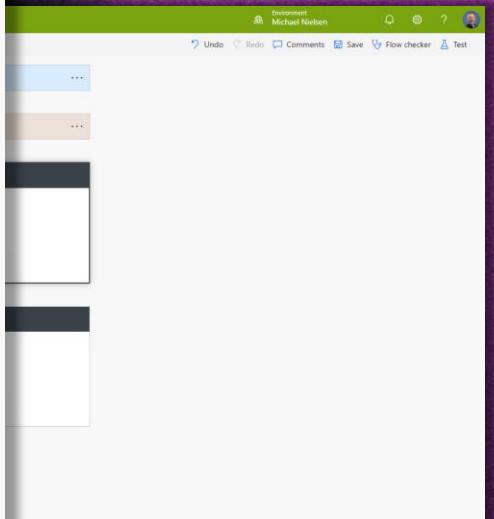
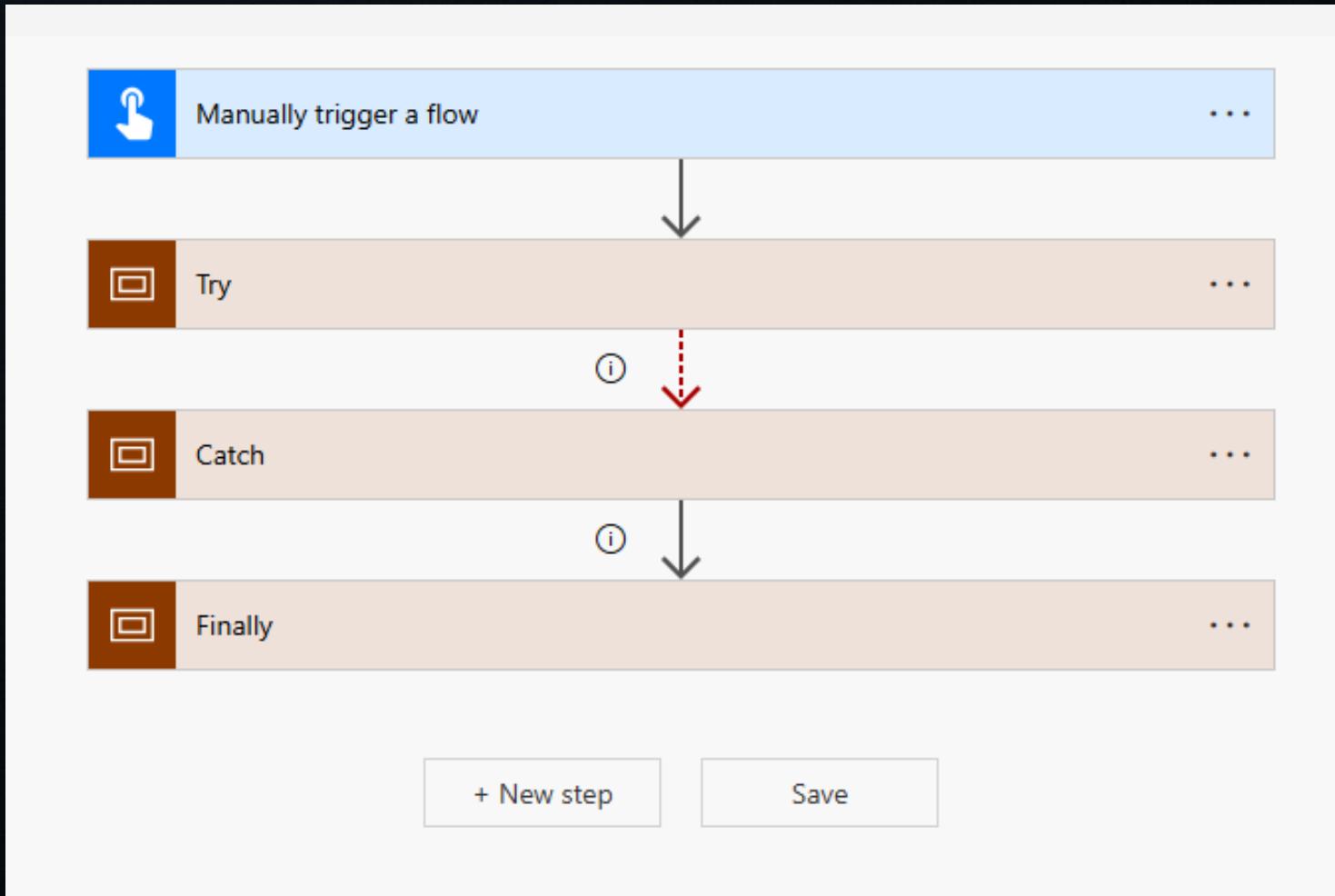
# Implement Try-Catch-Finally

- **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.

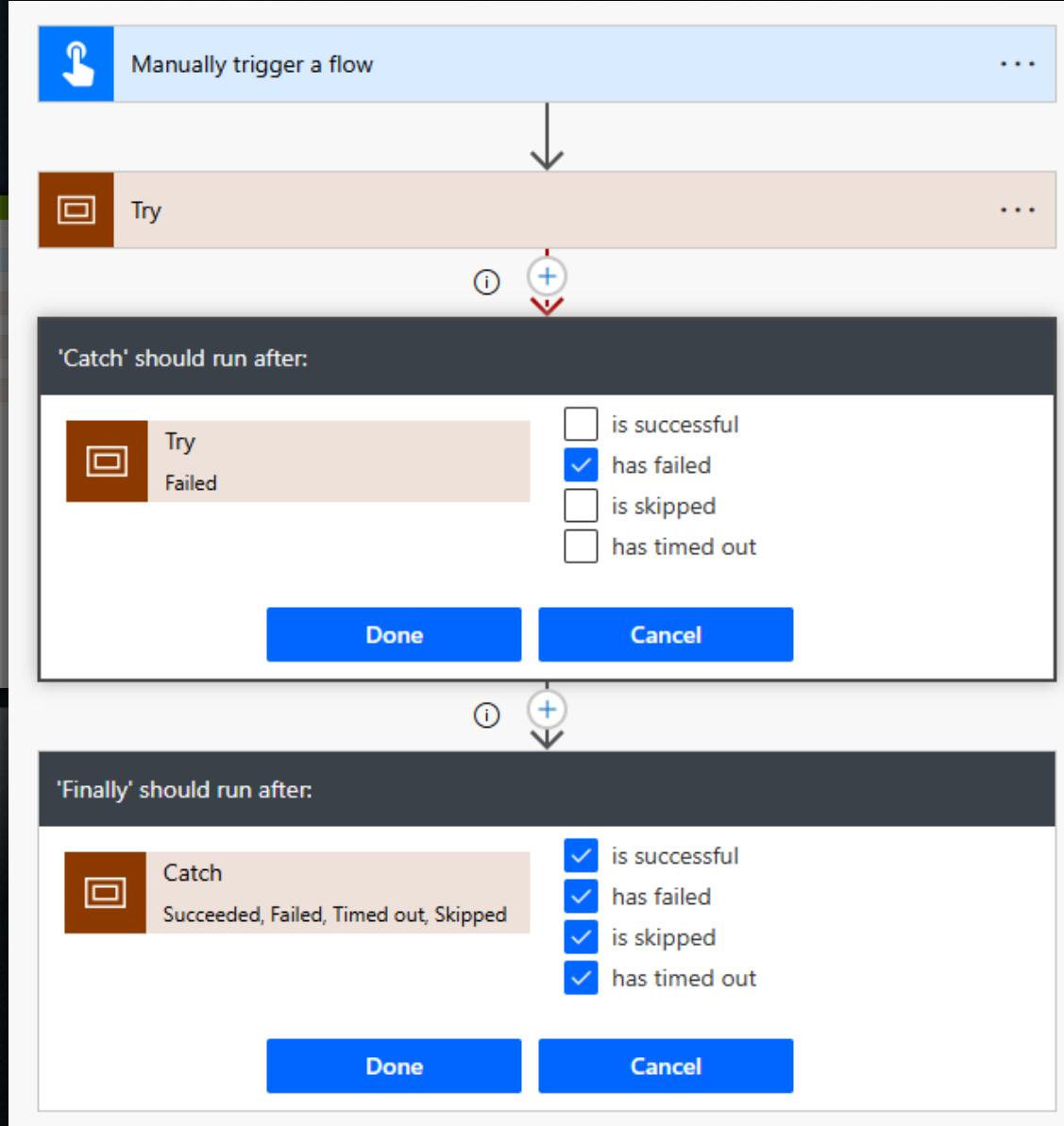
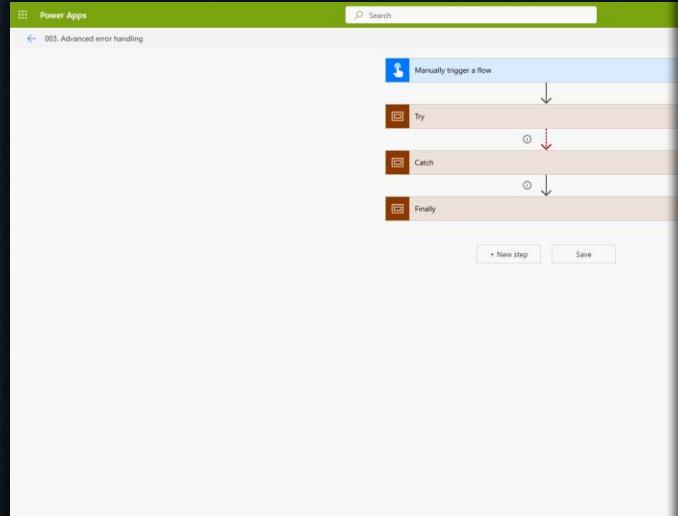
# Implement flow with Try-Catch-Finally logic



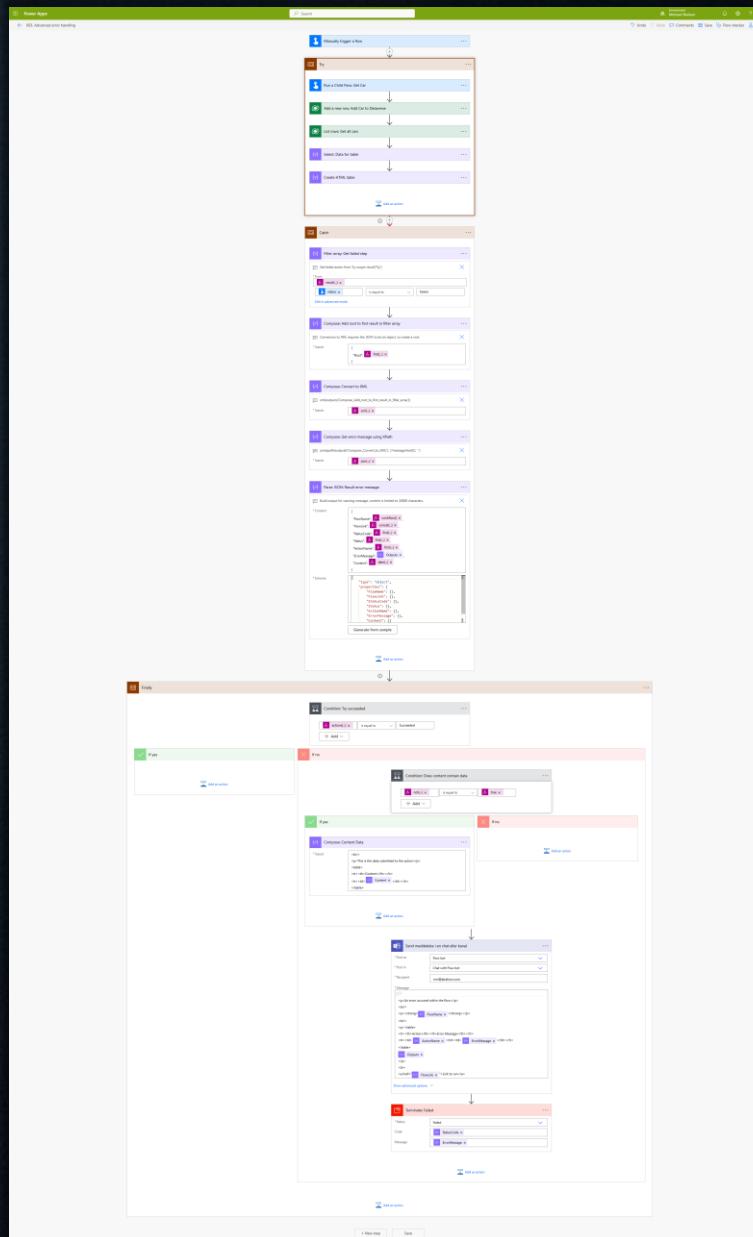
# Implement flow with Try-Catch-Finally logic



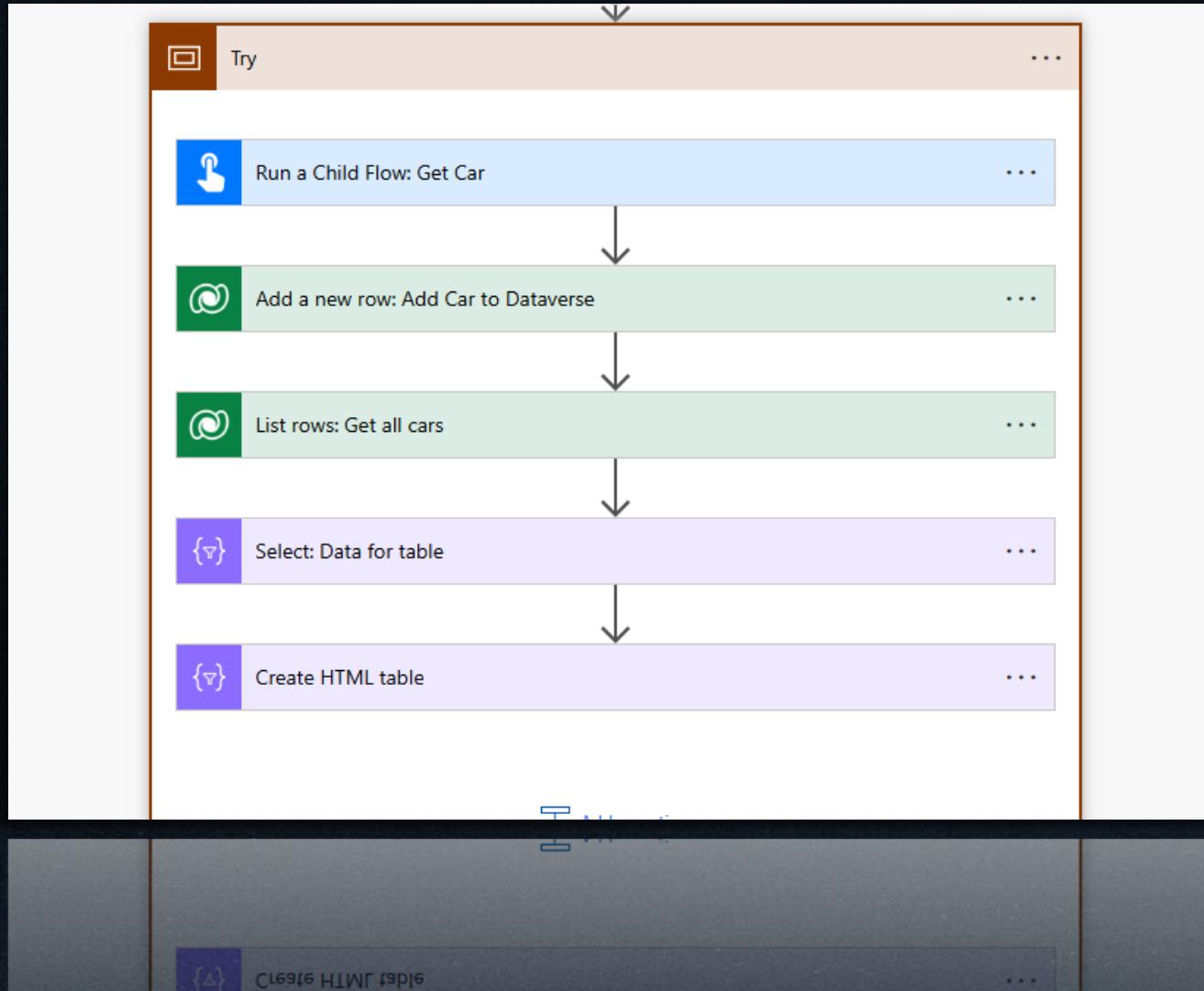
# Implement flow with Try-Catch-Finally logic



# Full flow



# Try Scope



No changes to logic

# Catch Scope - Filter array

The screenshot shows a Microsoft Power Automate flow with the following steps:

- Catch** (Step Type: Catch):
  - Filter array: Get failed step** (Step Type: Filter array):
    - Action: Get failed action from Try scope: result('Try')
    - From: result(...)
    - Condition:
      - status
      - is equal to
      - Failed
  - [Edit in advanced mode](#)

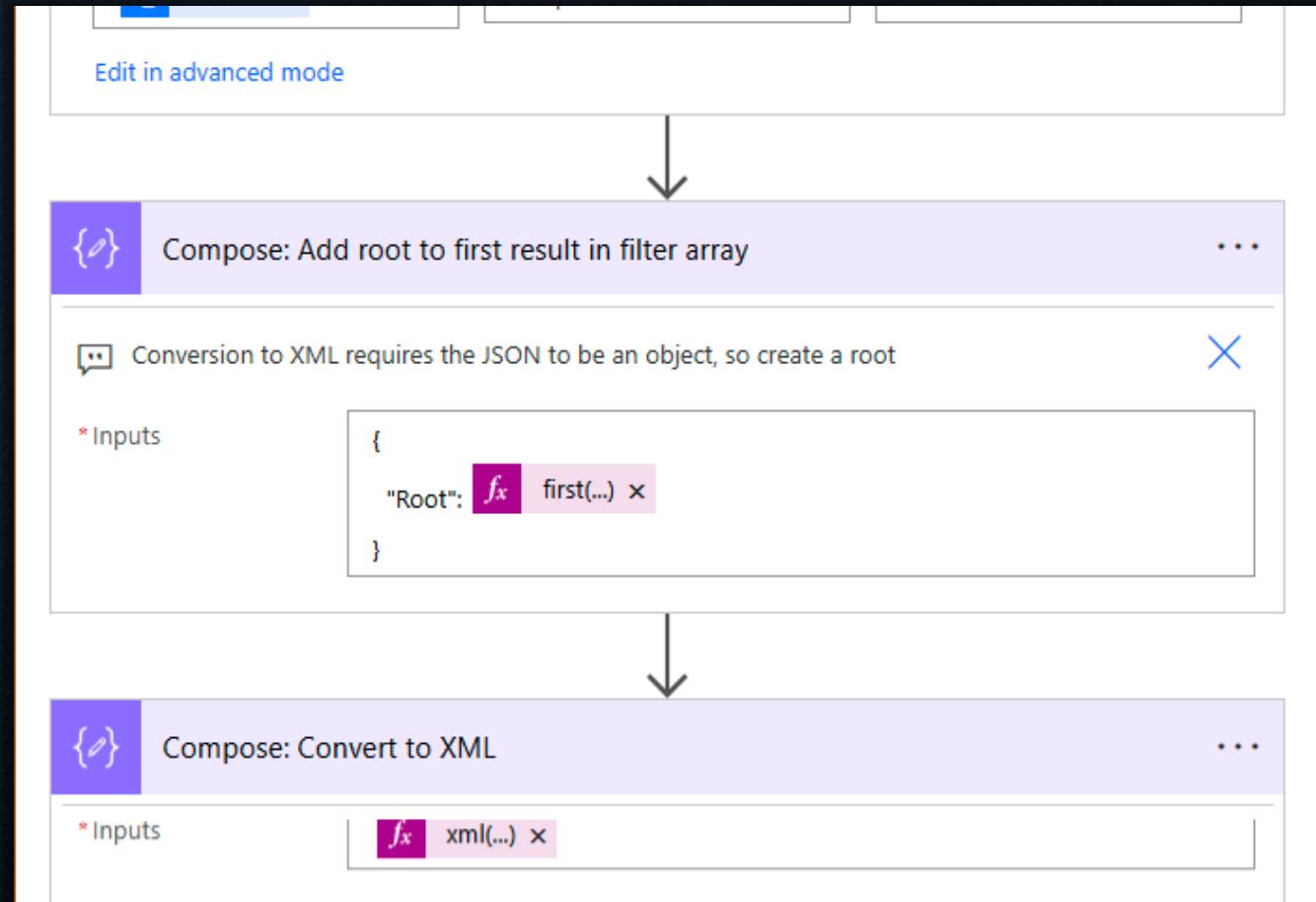
A large downward arrow points from the end of the "Filter array" step to the next step.

- Compose: Add root to first result in filter array** (Step Type: Compose):
  - Action: Conversion to XML requires the JSON to be an object, so create a root
  - Action: Conversion to XML requires the JSON to be an object, so create a root

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

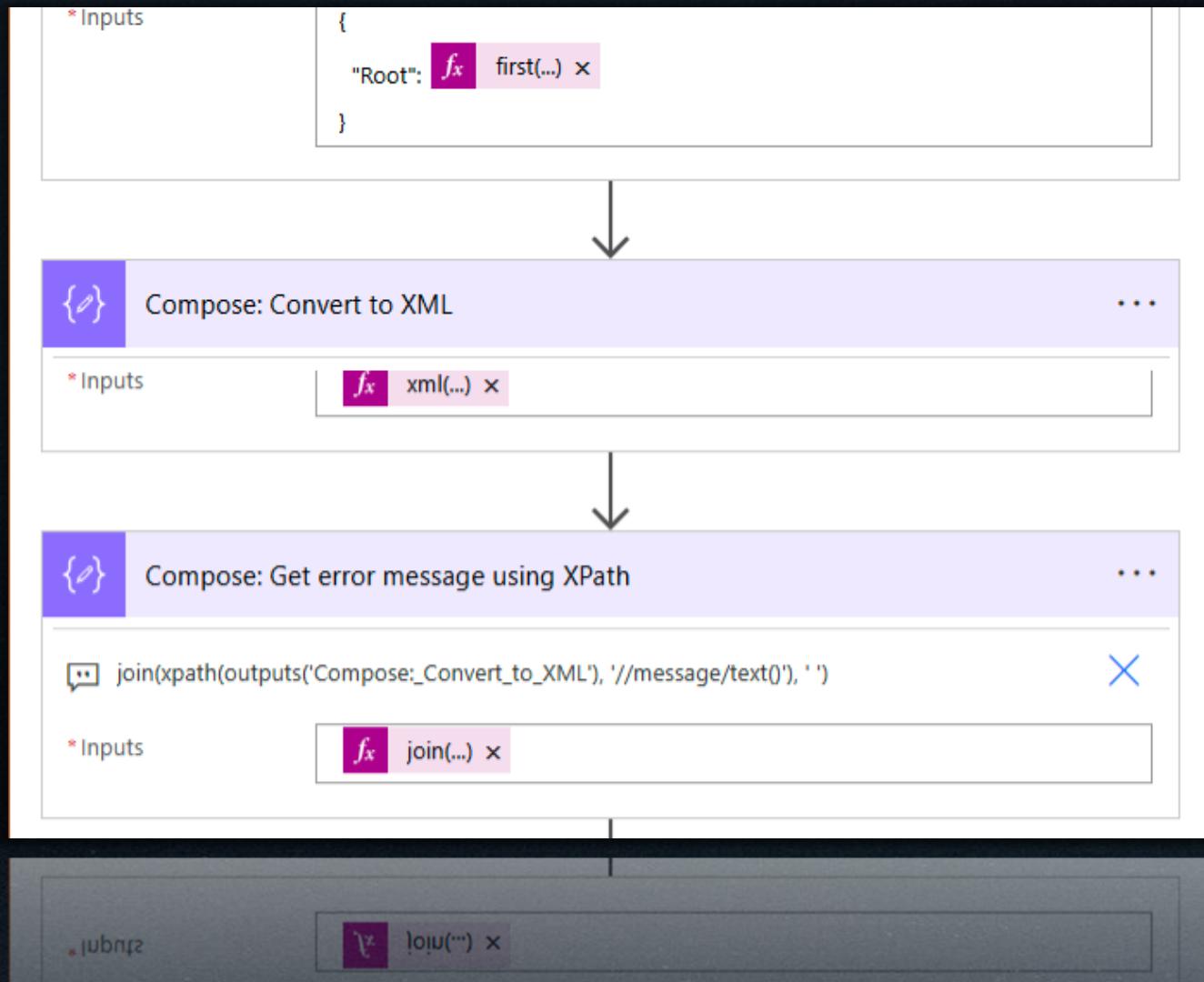
This process filters based on a status of 'Failed'.

# 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.

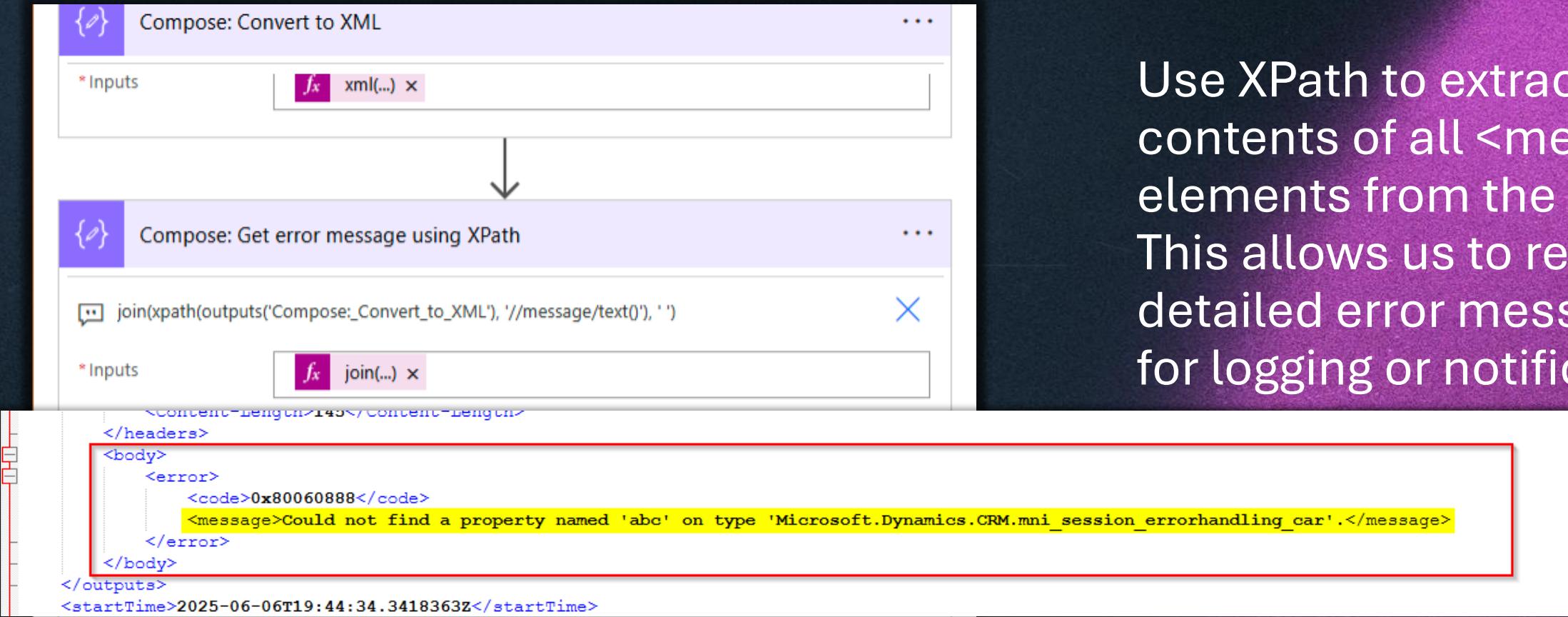
# XPATH Magic

```
<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/commondataservicefo
      <connectionReferenceName>shared_commondataserviceforapps</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=ae4fc7d0a097b5773958bcf3b99a7ca129004741d0b69030609019892c0cf615134d20c556b0b34b9b6ae43ec3f5dodad61788de889ffc592af7aca85fc1c508DDA53
        ; path=/; secure; HttpOnly,ReqClientId=0381256b-63e9-44d6-89e2-715a1f91add7; expires=Thu, 06-Jun-2075 19:44:34 GMT; path=/; secure;
        HttpOnly,ARRAffinity=ae4fc7d0a097b5773958bcf3b99a7ca129004741d0b69030609019892c0cf615134d20c556b0b34b9b6ae43ec3f5dodad61788de889ffc592af7aca85fc1
        99020715; path=/; secure; HttpOnly</Set-Cookie>
      <x-ms-service-request-id>7afafcca-4cee-44af-900c-e16d28dfc53f</x-ms-service-request-id>
      <Strict-Transport-Security>max-age=31536000, includeSubDomains</Strict-Transport-Security>
      <REQ_ID>7afafcca-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-remaining-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>
        15646178130123065243981352242232082219912178912206316770152112738718619361207136206,764820319923219080163166112462061867413114242422105158482112301
      </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>08584523678180444941656090403CU61</clientTrackingId>
  <clientKeywords>testFlow</clientKeywords>
  <code>BadRequest</code>
  <status>Failed</status>
</Root>
```

Message content is limited to 50000 characters.

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 a 'Parse JSON' activity configuration window. The title bar says 'Parse JSON: Result error message'. Below it, a note says 'Build output for warning message, content is limited to 20000 characters.' The main area is divided into two sections: 'Content' and 'Schema'.

**Content:**

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

**Schema:**

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

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 action is titled '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() x,  
    "FlowLink": fx concat(...) x,  
    "StatusCode": fx first(...) x,  
    "Status": fx first(...) x,  
    "ActionName": fx first(...) x,  
    "ErrorMessage": { } Outputs x,  
    "Content": fx take(...) x  
}
```

The 'Schema' section shows a partial schema definition:

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

## 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 a configuration window for a 'Parse JSON' command. The title bar says 'Parse JSON: Result error message'. Below it, a note says 'Build output for warning message, content is limited to 20000 characters.' The main area is divided into two sections: '\* Content' and '\* Schema'. The 'Content' section contains a JSON object with several fields, each with a formula or function applied. The 'Schema' section shows a partial JSON schema definition.

\* Content

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

\* Schema

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

## ActionName

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

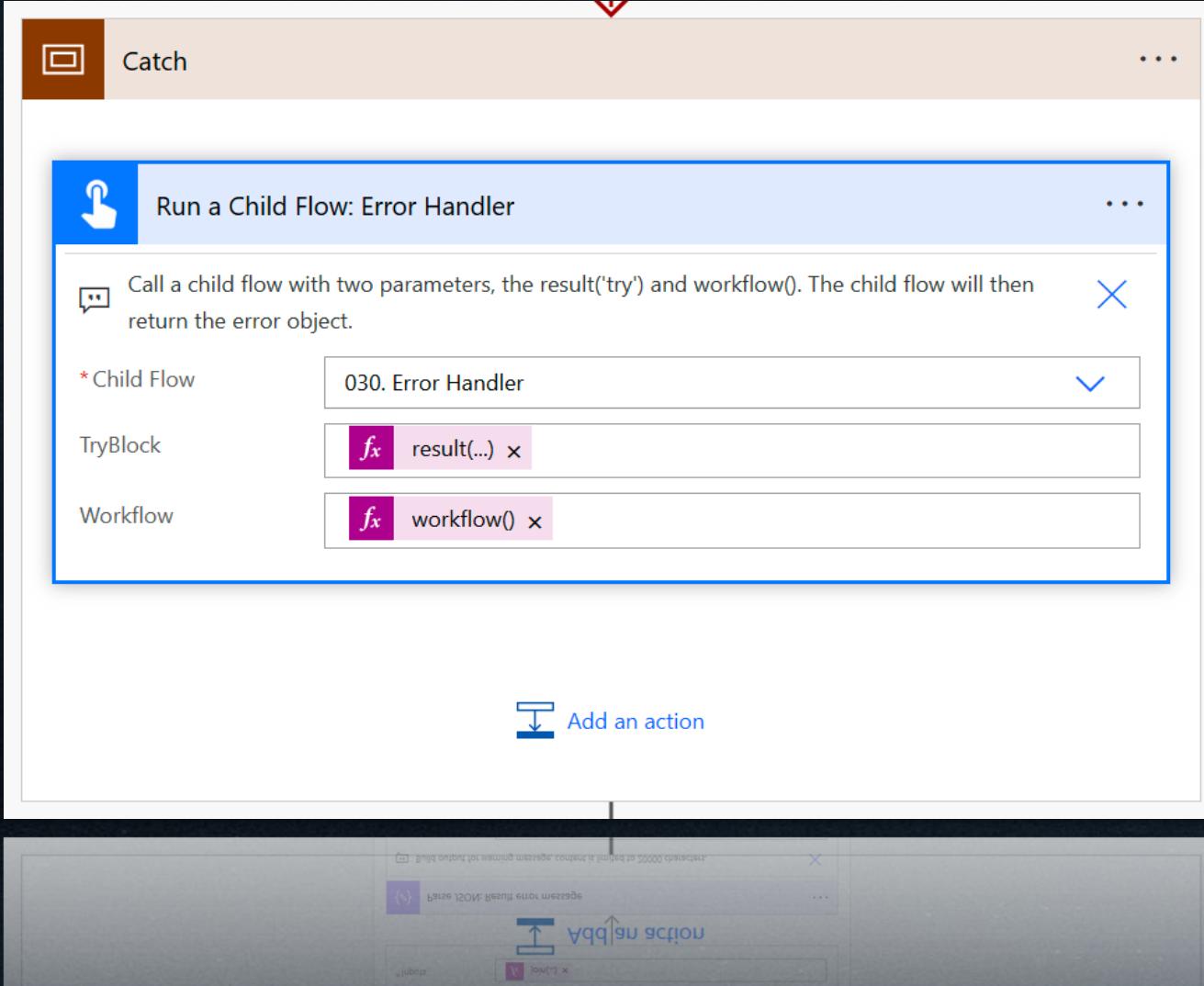
## ErrorMessage

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

## Content

```
take(string(first(body('Filter_array:_Get_failed_step'))), 20000)
```

# CATCH SCOPE – Child flow



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

# FINALLY SCOPE – Was Try successful?

The screenshot shows the Microsoft Power Automate designer interface. At the top, there is a large empty white box with a downward arrow icon and the text "Add an action". Below this is a horizontal timeline bar with a blue "i" icon and a downward arrow pointing to the first condition block.

The first condition block is titled "Condition: Try succeeded". It contains the following logic:

```
fx actions(...) is equal to Succeeded
```

Below this condition is a red "X" icon followed by the text "If no".

Underneath the "If no" section is another condition block titled "Condition: Does content contain data". It contains the following logic:

```
fx not(...) is equal to fx true
```

At the bottom of the screen, there is a dark grey footer bar with the text "notificações da A" and "Condition: Does content contain data".

# FINALLY SCOPE – Try failed, is there content

The screenshot shows the Microsoft Flow designer interface. At the top, there is a header bar with a search field and several icons. Below the header, a large white box represents a scope.

**Scope Header:**

- fx actions(...)**
- is equal to**
- Succeeded**
- + Add**

**Scope Body:**

The scope body contains a condition and two branches:

- If no:** A red box containing a condition block.
  - Condition: Does content contain data**
  - fx not(..)**
  - is equal to**
  - fx true**
  - + Add**
- If yes:** A green box containing a compose action.
  - Compose: Content Data**
  - \* Inputs**:  
`<br/><p>This is the data submitted to the action</p>`
  - Outputs**:  
`<q>The data submitted to the action</q>`
  - + Add an action**

# FINALLY SCOPE – Format content

Condition: Does content contain data

fx not(...) is equal to fx true

+ Add

✓ If yes

{ } Compose: Content Data

\* Inputs

```
<br/>
<p>This is the data submitted to the action</p>
<table>
<tr><th>Content</th></tr>
<tr><td>Content</td></tr>
</table>
```

... Add an action

✗ If no

Add an action

```
graph TD
    A[Condition: Does content contain data] -- "fx not(...)" --> B[is equal to]
    B -- "fx true" --> C[true]
    C -- "+ Add" --> D[If yes]
    D -- "✓" --> E[Compose: Content Data]
    E -- "* Inputs" --> F["<br/><p>This is the data submitted to the action</p><table><tr><th>Content</th></tr><tr><td>Content</td></tr></table>"]
    F -- "... Add an action" --> G[If no]
    G -- "✗" --> H[Add an action]
```

# FINALLY SCOPE – Send Teams message

The screenshot shows the configuration of a 'Send meddelelse i en chat eller kanal' (Send message to a chat or channel) step in Microsoft Power Automate. The step is titled 'Send meddelelse i en chat eller kanal' and has the following settings:

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

The 'Message' field contains the following XML code:

```
</>
<p>An error occurred within the flow:</p>
<br/>
<p><strong>(&gt; FlowName &lt;/strong)</p>
<br/>
<p><table>
<tr><th>Action</th><th>Error Message</th></tr>
<tr><td>(&gt; ActionName &lt;/td><td>(&gt; ErrorMessage &lt;/td></tr>
</table>
(&gt; Outputs &lt;/p>
<br>
<a href="(&gt; FlowLink &lt;/a>">Link to run</a>
```

# FINALLY SCOPE – Terminate as failed

The screenshot shows a Microsoft Flow designer interface. At the top, there is a code block containing the following HTML:

```
<a href="#" FlowLink >Link to run</a>
```

Below this is a "Show advanced options" button.

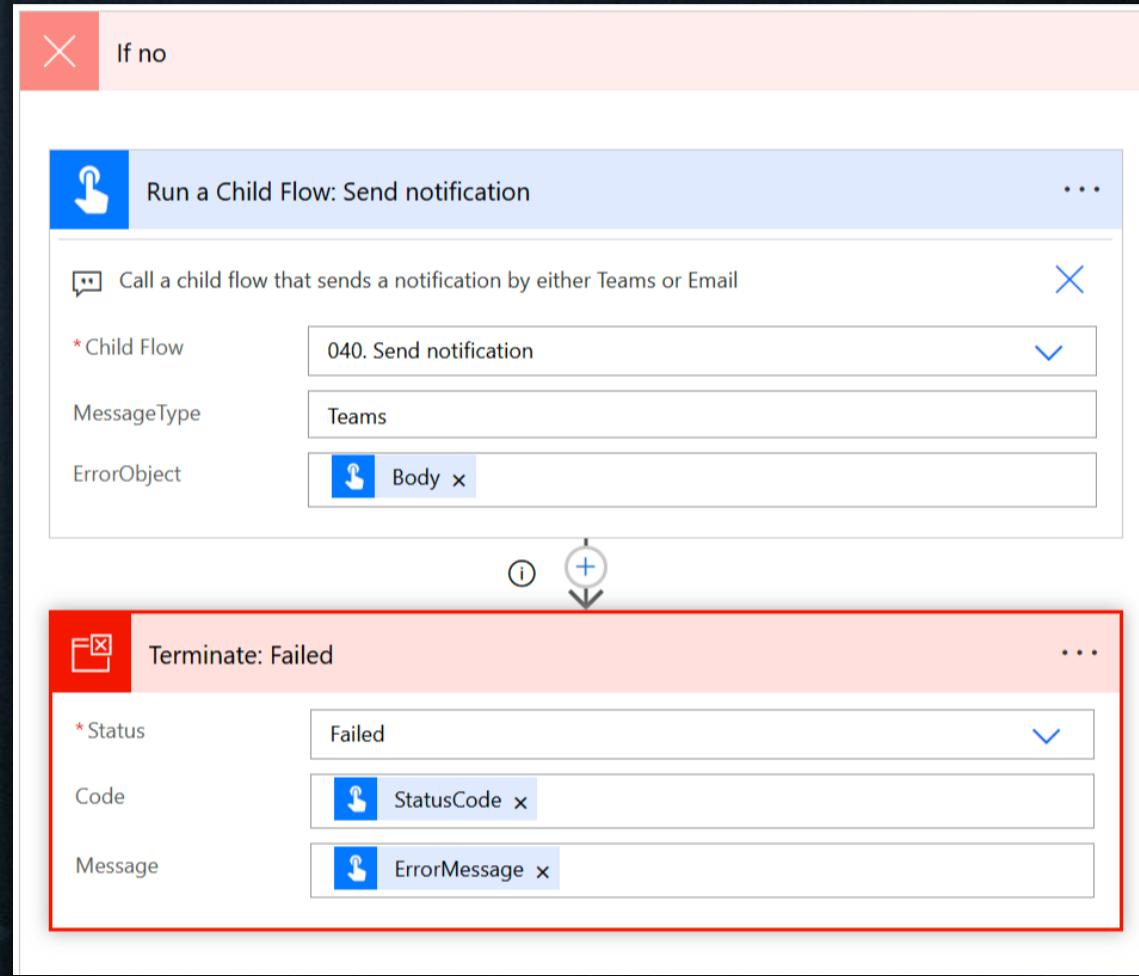
A large downward-pointing arrow indicates a flow from the code block to a "Finally" scope.

The "Finally" scope contains a "Terminate: Failed" action. The configuration for this action is as follows:

- \* Status: Failed
- Code: StatusCode
- Message: ErrorMessage

At the bottom of the scope, there is an "Add an action" button.

# 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

# 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

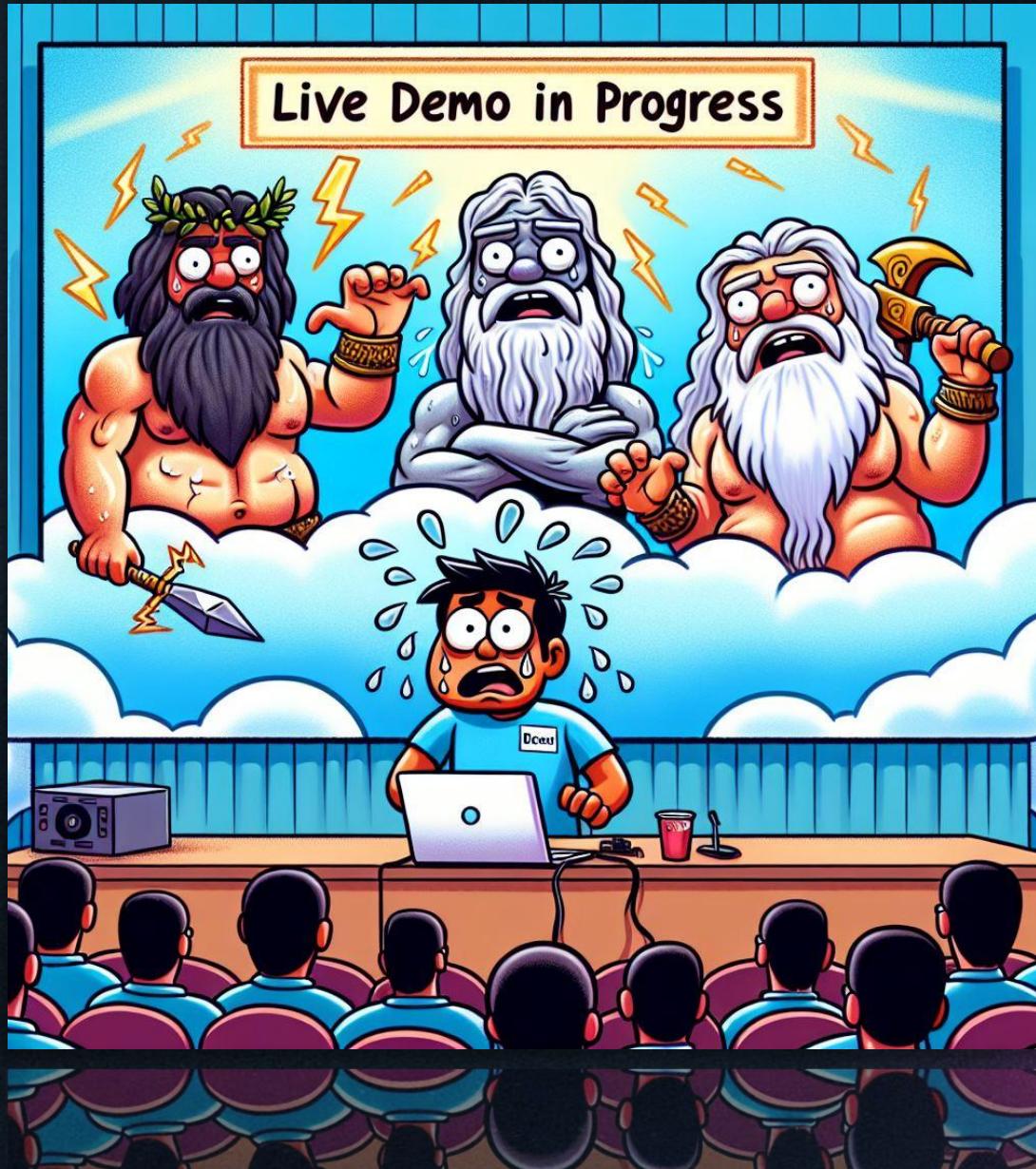
Content
{"name":"Apply_to_each:Direction","inputs":{"foreachItems":[{"Direction":"East"}, {"Direction":"West"}, {"Direction":"Nowhere"}, {"Direction":"East"}, {"Direction":"West"}, {"Direction":"Nowhere"}]}, "inputsMetadata": {"foreachItemCount":6}, "startTime":"2025-06-18T22:51:23.3137558Z", "endTime":"2025-06-18T22:51:26.1082844Z", "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](#)

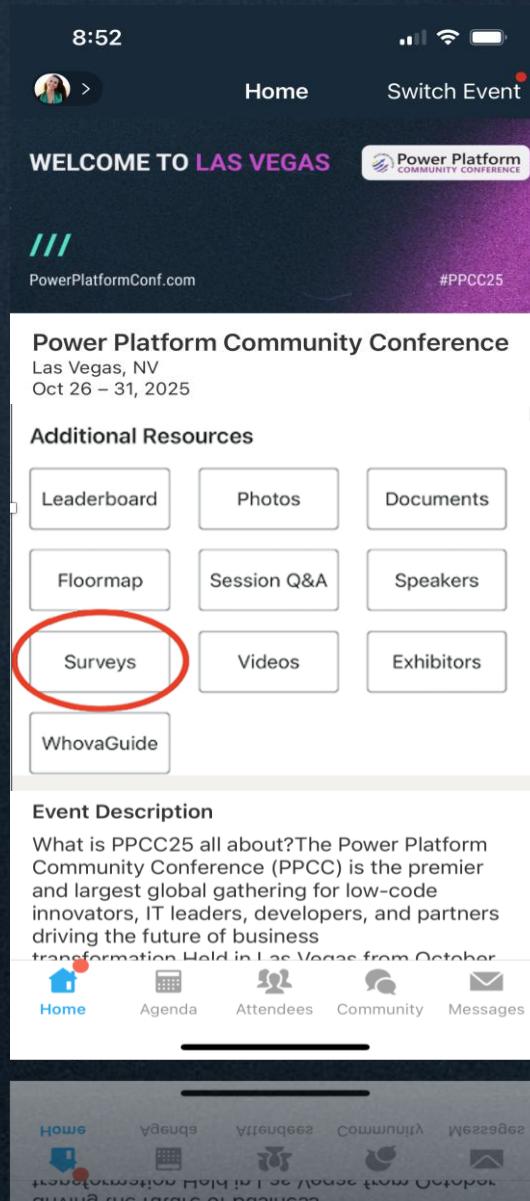
[Link to run](#)

actions succeeded."}}

# Lets look at it live



# Session Feedback Surveys



**We really want to hear from YOU!**

*In the pursuit of making next year's Power Platform Community Conference even better, we want to hear your feedback about this session.*

**Here's How -**

- *Simply go to the Whova App on your smartphone*
- *Scroll down on the Power Platform Community Conference Homepage to 'Additional Resources' to click "Surveys"*
- *Click Session Feedback.*
- *Scroll down to find this session title.*
- *Complete the session feedback survey.*
- *Finally, click 'Submit'*
- *It's just that easy!*

# THANK YOU

**Nicolai Schjørman**

Business Unit Manager,  
Abakion A/S, Denmark

**Michael Nielsen**

Senior Power Platform Consultant  
Abakion A/S, Denmark

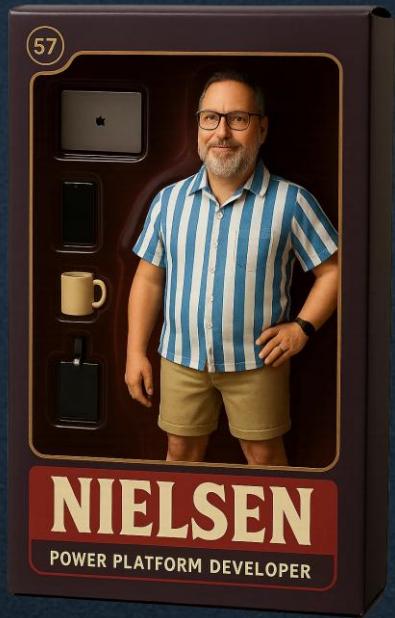
PPCC 2025 DAY 3

# Lets connect



**Nicolai Schjørmann**  
Business Unit Manager  
Abakion A/S, Denmark

PPCC 2025 DAY 3



**Michael Nielsen**  
Senior Power Platform Consultant  
Abakion A/S, Denmark

# Lets connect



PPCC 2025 DAY 3

# Lets connect



PPCC 2025 DAY 3