Microsoft Flow Patterns and Practices

Contents
Abstract and Learning Objectives: .......................................................... 3
Filtering using arrays and conditions .................................................... 3
Introduction ......................................................................................... 3
Exercise 1: Available ice-cream flavors? ............................................ 3
  Task 1: Login to the Microsoft Flow website and create flow .......... 4
  Task 2: Configure the trigger ......................................................... 4
  Task 3: Compose an array of available flavors .............................. 5
  Task 4: Add a condition to check if the user provide flavor is in your list. 6
  Task 5: Send an email with appropriate response .......................... 8
  Task 6: Save and Test ................................................................... 11
  Task 7: Evaluating results .............................................................. 13
Advanced tip: Case sensitivity ............................................................. 14
Learnings and applications ................................................................. 17
Concurrency Control and Parallel Branches ...................................... 18
Introduction ......................................................................................... 18
Exercise 2 – Understanding Degree of Parallelism .......................... 18
  Task 1: Create from blank and Manually trigger a flow ............... 18
  Task 2: Compose an array of 4 items ............................................. 19
  Task 3: Add and configure an Apply to each loop after the Compose step 20
  Task 4: Add a 5 second delay in the Apply to each block ............. 22
  Task 5: Save and Test ................................................................... 24
  Task 6: Observe time taken to run loop ...................................... 24
  Task 7: Turning on concurrency control ....................................... 25
  Task 8: Test and observe .............................................................. 26
Learnings and applications ................................................................. 26
Exercise 3: Parallel branches ............................................................ 27
  Task 1: Duplicate your flow – Exercise 2 .................................... 27
  Task 2: Turn flow - on | concurrency control - off .................... 28
  Task 3: Add another delay action ............................................... 28
Task 4: Save, Test and Observe

Task 5: Parallelized branching

Task 6: Save, Test and Observe

Learnings and applications

Chaining flows, Request / Response, HTTP + Parse JSON

Problem Statement

Solution overview

Exercise 4.1: Request / Response pattern

Task 1: Configure the request trigger

Task 2: Initialize variable to store unique values from the list provided on the input

Task 3: Store unique values in variable

Task 4: Setup your Response

Task 5: Save and note POST URL

Exercise 4.2 Sending an email with CSV attached

Task 2: HTTP and Parse JSON

Task 3: Getting deals by owner and sending them as CSV attachments

Handling errors

Exercise: Using parallel branches to handle errors

Task 1: Add a parallel branch

Task 2: Handle error in the HTTP action

Task 3: Simulating Failure
Abstract and Learning Objectives:
In this workshop, you will be learning about advanced practices and patterns – that will help you build out robust workflows on the Microsoft Flow Platform.

The workshop is structured to be a text book styled guide – we will start by looking at concepts first and then apply them to a real-world problem.

Filtering using arrays and conditions

Introduction:
In this section we are going to learn how build smart filters by constructing arrays and using them with the condition builder.

1. Condition builder:

2. Compose action:

Exercise 1: Available ice-cream flavors?

Automation Problem: Your company owns ice-cream carts and the ice-cream flavors offered on these carts change week after week. You decide to build a flow that can be used by your customers to check if their favorite ice-cream flavor is available on that day.

Solution overview: We will use the compose action to create a list of available flavors. We will then check if the user provided flavor option belongs to the list of available flavors.

If Yes – We will send the user an email telling them the flavor is available.

If No – We will send the user an email telling them the flavor is unavailable and to try again next week.
Task 1: Login to the Microsoft Flow website and create flow

1. Navigate to https://flow.microsoft.com and sign in with the credentials provided by the workshop instructor
2. (If prompted) Click Get Started.
4. Select Create from blank.

Task 2: Configure the trigger
The first thing you will need to configure is the trigger, i.e. define when this flow should run. There are three types of triggers:

- **Automated / Event driven** - e.g. new item being added to a table, a new email arriving in a user’s inbox, a new tweet being posted that meets certain conditions, etc.
- **Instant** – e.g. when a button is pressed in PowerApps or the Flow Mobile app, for a selected item in SharePoint, for a selected row in Excel, etc.
- **Scheduled** - e.g. hourly, daily, weekly, monthly, etc.

For the purposes of this exercise we are going to select “**Flow button for Mobile – Manually trigger a flow**” trigger

After selecting your trigger:

1. Click on **Add an input**
2. Then select **Text**
3. Configure the trigger inputs as shown in the image below:

Task 3: Compose an array of available flavors

Now we need to create a store of available flavors to compare against the input provided by the user.

1. Select **+New Step** and select **Add an Action**.
2. Search for Compose and select the “Data operations – Compose” action.

3. Configure the Compose action by building an array of available ice-cream flavors.

   Tip: You can come up with your own list of flavors. The important thing is to construct a valid array. It should be in the following format: [“value1”, “value2”] and these values should be in lowercase.

Task 4: Add a condition to check if the user provide flavor is in your list.

1. Select +New Step and select Add a condition.
2. Click in the left edit box that says, “Choose a value” and select **Output** from the dynamic content pane. You may need to press the + icon below the edit box to hide the dynamic content pane.

3. Select **contains** for condition.
4. Click in the right-most edit box that says, “Choose a value” and select **Flavor** from the dynamic content pane. You may need to press the + icon below the edit box to hide the dynamic content pane.

5. Conceptually – we have constructed a condition – that will now check if the **Output (List of available flavors)** contains **user provided flavor value**.

Task 5: Send an email with appropriate response

Now – if the condition evaluates to true it will execute the actions configured in the **If Yes** branch. If the condition evaluates to false – it will execute the actions configured in the **If no** branch.

For this exercise – we will use the “Send email” action to let the user know if the ice cream is available or not.

1. Select **Add an action** in the **If yes** branch
2. Search for **Send an email notification** in the search bar and select **Mail – Send an email notification** action.

![Choose an action](image)

3. Configure the email action as shown below.

**NOTE:** If you’re prompted to create a connection, go ahead and click create.
NOTE: Use an email address in the TO field where you would like to receive an email when we test run this flow. In a “real-world” scenario you can use the User email output from the trigger.

4. Configure the If no branch by following the steps above. We will only change the content of the Send an email notification 2 action as shown below
Task 6: Save and Test

1. If you have executed Tasks 1 through 5 correctly – your flow should look as show below:

![Diagram of flow](image)

2. Go ahead and name your flow **Exercise 1** and the save the flow by selecting the **Save** command in the top right corner.

![Flow with Save and Test buttons](image)

3. Next step is to test this flow in see it in action. In the top right corner – select **Test**.
4. Select the **I’ll perform the trigger action** option.

5. Click **Save & Test**.

6. Select **Continue**
7. Enter your flavor name. I am using **chocolate** for this test run, the Select Run Flow

8. Select **Done**

Task 7: Evaluating results
1. Once you click done – you should see the result of your **test run**
This is the run details view. You can observe that the condition card – show the evaluated result (in this case true – which implies the If yes branch was executed). For chocolate, as our result evaluated to true, you will receive an email mentioning the flavor is available.

Advanced tip: Case sensitivity
Condition evaluation is case sensitive, which means – if we don’t change our flow and keep it as is:

- chocolate – will evaluate to true
- Chocolate – will evaluate to false

You can handle case-sensitivity, by adding an expression that will convert the flavor provide by the user to lower case.

1. Click Edit in the top right corner to edit your flow.
2. Select the Flavor input in the right most input box in the condition card
3. Then select Expression
4. Next select **See more** on the **String functions** row

5. Select **toLowerCase(text)** from the list of available string functions
6. You should see the function bar (fx) populated with the expression. Next select **Dynamic content**

   ![Dynamic content](image)

7. Ensure your cursor is between the two parenthesis and then select **Flavor**

   ![Dynamic content](image)
8. Finally select **OK**. You should see your condition card updated as shown below.

![Condition Card](image)

9. Test your flow with Flavor input **chocolate** and **Chocolate**. Both inputs should now result in the condition evaluating to true.

**Advanced tip: Using the if expression**

With this approach you needed two different Send Email actions. In this case this is okay because there isn’t a lot of email content, but if you have a lot of information in the Send email action it can be difficult to keep these two in sync.

You can solve this by using the **if** expression.

1. Drag the first Send email step out above the condition, and then delete the condition
2. In the subject, delete out the content that says “is available”
3. Add dynamic content and select the Expression tab
4. Enter the expression:
   
   ```
   if(contains(outputs('Compose').triggerBody()['text'], 'is available'), 'is not available')
   ```

5. Repeat this step in the body – feel free to customize the two text strings

Now you have just one action for sending email.

**Learnings and applications**

There are few other ways to through which you can achieve the same scenario in flow. However, these aren’t very scalable.

- We could build an advanced OR condition, along the lines of:
  - If (flavor == “chocolate” || flavor == “vanilla”)
- We could also build this logic into nested conditions.

Using the **array** approach, you can easily maintain your list of allowed or disallowed items and build a performant, robust flow around it. To take it a step further – you can store the list in an external data source like SharePoint or SQL. Fetch it dynamically and do a conditional check. This would eliminate the need to “hardcode” the list in the flow itself.
Concurrency Control and Parallel Branches

Introduction

In this section, we are going to learn how to speed up execution time of flows. The first concept we will look at is concurrency control.

![Concurrency Control settings](image)

Exercise 2 – Understanding Degree of Parallelism

**Note**: This exercise does not capture real world scenarios. It is intended to explain the concepts of Concurrency Control. You are expected to understand the theory and apply the concepts in Exercise 4.

Task 1: Create from blank and Manually trigger a flow
Follow the steps from the previous exercise:

1. Sign in to Flow
2. Select +Create from Blank
3. Select Search hundreds of connectors and triggers
4. Select Flow button for Mobile – Manually trigger a flow

If followed the steps correctly – you should see this on your canvas –
Task 2: Compose an array of 4 items

1. Select **+New Step** and select **Add an Action**.

2. Search for **Compose** and select the “**Data operations – Compose**” action.

3. Configure the **Compose action** by building an array of **4 items**.
Tip: You can come up with your own list of items. The important thing is to construct a valid array. It should be in the following format: [“value1”, “value2”]

Your canvas should now look as shown below:

Task 3: Add and configure an Apply to each loop after the Compose step

1. Select +New Step and then click ... More
2. Select **Add an apply to each** from the flyout menu.

3. Click into the **Select an output from previous steps** field.

4. Select **See more** as instructed in the help text in the **Dynamic Content** flyout and then select **Output**.
Task 4: Add a 5 second delay in the Apply to each block

1. Select **Add an action**

2. Search for **Delay** in the search box and select **Schedule – Delay**
3. Configure the Delay to wait for 5 seconds (An example of real world scenario where this would be useful: Executing a SQL query for each item in the array)
Task 5: Save and Test

1. Name your flow **Exercise 2**, save it and then test your flow by selecting the **I'll perform the action** option.

![Test Flow](image)

Task 6: Observe time taken to run loop

As you can see from the screenshot here – it took **21** seconds for the **Apply to each** loop. About 5 seconds for each item in the array we composed.
Task 7: Turning on concurrency control

By default, all **Apply to each** loops in Flow are executed sequentially. However, you can configure them to run in parallel.

1. Click on **Edit** in the top right corner of the screen.

2. Next, click the ... menu on the **Apply to each** and then select **Settings**.
3. Next turn **Concurrency Control** on and set the **Degree of parallelism** to 2.

4. Select **Done**, and then save your flow.

**Task 8: Test and observe**

1. **Repeat tasks 5,6 and 7.** Each time *incrementing the degree of parallelism*.

If done correctly you should observe the following results:

<table>
<thead>
<tr>
<th>Array Length</th>
<th>Degree of Parallelism</th>
<th>Time taken to run loop</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Off</td>
<td>21 seconds</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>11 seconds</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>6 seconds</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>6 seconds</td>
</tr>
</tbody>
</table>

**Learnings and applications**

Your flow runs can be optimized to run faster when you can parallelize the **Apply to each** loop. This technique can be used whenever you don’t have dependencies on previous or next items in an array. Some real-world scenarios where this approach is applicable:

- Sending individual emails to many recipients
• Updating records in Common Data Service, SharePoint Lists, SQL
• Bulk creating users in Azure Active directory
• Fanning out operations in SharePoint: http://johnliu.net/blog/2018/2/serverless-parallelism-in-microsoft-flow-and-sharepoint

The possibilities are quite frankly endless.

Next, we will look at parallel branches. Parallel branches are useful when you have two or more actions that don’t necessarily depend on one another and as such you can execute them in parallel. The next exercise reuses the flow from Exercise 2, so please ensure you have completed Exercise 2 before you proceed to Exercise 3.

Exercise 3: Parallel branches
Like Exercise 2 – this exercise is geared towards understand the concepts and theory behind parallel branching.

Task 1: Duplicate your flow – Exercise 2
• Go to your My flows list and click on the … contextual menu on the flow named Exercise 2, the select Save As

• On the dialog the follows, change the name to Exercise 3 and select Save
Task 2: Turn flow - on | concurrency control - off

- Turn on your flow and select the *pencil* icon to edit flow named **Exercise 3**

- Ensure that you turn **off** concurrency control for the **Apply to each** action

- Select **Done**

Task 3: Add another delay action

- Select **Add an action** in the **Apply to each** card
Add another 5 second delay – your **Apply to each** card should now look as shown

- **Task 4:** Save, Test and Observe
  - Save and Test your flow
• Observe the amount of time it takes to execute the **Apply to each** loop. (It should take about **42 seconds** to run the loop)

![Diagram](image)

**Task 5: Parallelized branching**

• Select **Edit** in the top right corner to go back to the **flow designer** (editing canvas)

• Hover over the arrow between the **Compose** and **Apply to each** step. You should see a (+) icon pop up

![Diagram](image)

• Select the (+) icon – then move your cursor over **Add a parallel branch** – then select **Add an apply to each**
• Drag the **Delay 2** action and drop it in the **Apply to each 2** card

  ![Image of Delay 2 action]

• Pass the compose **output** to the input field **Select an output from previous steps** in the **Apply to each 2** card. Your flow should now look as shown below

  ![Image of flow with Delay 2 action added]
Task 6: Save, Test and Observe

- Save and Test your flow
- Observe the amount of time it takes to execute the Apply to each loop. (It takes about 22 seconds to run both loops in parallel)

Bonus task 1: Can you run the flow any faster? What is the shortest amount of time you could execute this flow in?

Bonus task 2: Without introducing parallel branches, how quickly can you make the flow run in a single loop? (Your flow would be in the state as described at the end of Task 4 in this exercise)

Learnings and applications

What we observe is that if two unrelated operations – (for demonstration purpose: Delay 1 and Delay 2) – that are not dependent on each other can be run in parallel to optimize execution time. In general,
adding parallels adds a slight overhead to the flow execution – a good rule of thumb is to parallelize only when actions take more than 5 seconds to execute.

You can use parallel branches to achieve some of the following scenarios:

3. Creating or updating records in multiple systems – E.g.: Getting Microsoft form responses and storing them in SharePoint AND streaming them to a Power BI dashboard
4. Getting data from multiple sources and consolidating them in one. E.g.: Reading data from two SharePoint lists, consolidating the data and sending a single email – with data from both sources.

[Important] Remember the learnings from Exercise 2 and Exercise 3 - these will be applicable in the following exercises!

Chaining flows, Request / Response, HTTP + Parse JSON

In this section we will cover a few more advanced concepts –

1. **Chaining flows** – For certain types of scenarios, it can help to break you flow into multiple logical components. In the following exercise we will look at using the Request / Response and the HTTP – Parse JSON pattern.
2. **Select and CSV tables**: We will also look at using the Select operation and creating CSV tables from SharePoint list data

Problem Statement

You are given access to the following SharePoint list, which contains the leads that have been generated by your CRM system. Your job is to **automatically** send a .csv attachment with the leads information to each individual lead owner.

SharePoint Site name: https://mscorpevents.sharepoint.com/sites/Demosite

SharePoint List name: Leads

*(Note: If you’re using your own SharePoint list – please recreate a list that looks like what’s shown in the image)*
Email:

Microsoft PowerApps and Flow <microsoft@powerapps.com>

Hi Ashley M - Please find your leads attached!

Click here to download pictures. To help protect your privacy, Outlook prevented automatic download of some pictures in this message.

Your leads are attached to this email!

If you want to unsubscribe from these emails, please use this form.

Attachment Content:
Solution overview
To solve this problem, we are going to create two flows:

1. The first flow (named Exercise 4.1) will be a simple Request/Response flow that takes in as input a list of strings (with or without duplicates) and returns a list of distinct strings. If the input is – [“Mary”, “John”, “Mary”, “Mark”, “John”] the output will be [“Mary”, “John”, “Mark”]
2. The second flow (named Exercise 4.2) will be a bit complex where we will get the data from SharePoint, clean it, create CSV tables from the data and then attach them to emails and send them over.

Exercise 4.1: Request / Response pattern
Task 1: Configure the request trigger
- Start by creating a flow from blank and add a Request – When an HTTP request is received trigger
- Next Select Use sample payload to generate schema and type in a sample list as shown
*Select Done*, then select Show advanced options and configure method to **POST**

**Task 2:** Initialize variable to store unique values from the list provided on the input
- *Add an action to initialize an Array variable as shown below*
Task 3: Store unique values in variable

- Like the technique we used in Exercise 1 – setup an **Apply to each with a condition** as shown.

**Bonus question:** Should you add concurrency control and parallelize the **Apply to each loop**?

- In the **If Yes** branch – add an action to append **Current item** to the list if it is not already contained in the **Unique** list.
Task 4: Setup your **Response**

- Setup your **Response** to return the **Unique** list

![Response setup](image)

Task 5: Save and note POST URL

- Name your flow Exercise 4.1 and save it.
- As soon as you save your flow the **URL** field should be populated. This is important, and we will use it in the next exercise – **copy** the **URL** in a safe place

![HTTP request](image)

- Overall – Your flow should look as shown:
When a HTTP request is received

Initialize variable

Apply to each

* Select an output from previous steps

Condition

Add an action  Add a condition  More

Response
Exercise 4.2 Sending an email with CSV attached

Task 1: Get all the Owners from the SharePoint List

- Create from scratch and setup a Flow button for mobile – Manually trigger a flow
- Get items from the SharePoint list
  - SharePoint Site Address: https://mscorpevents.sharepoint.com/sites/Demosite
    (NOTE: If the dropdown doesn’t show any options. Select Enter custom value and paste in the site name)
  - List name: Leads
- Next – Initialize an Array of Owners
- Finally – Append all the Owners to the initialized array
- Your flow should now look as shown:
Task 2: HTTP and Parse JSON

- Now, we will send our constructed list (array) to the flow we created in the previous exercise (Exercise 4.1) to be de-duped. You can de-dup the list in this flow as well – however this functionality can be used across multiple flows and as such it is a good idea to generalize it a reusable flow.

- Start by adding an HTTP action and configure the method to POST. This MUST match the method we configured for the trigger in Exercise 4.1.

- Paste in the URI that you copied and noted down as part of Task 5 in Exercise 4.1. Each URL is unique – so please ensure you have the right URI.

- In the Body input – drop the Owners dynamic content.

- Next add the Parse JSON action and configure it as shown below. What this does, is it helps flow understand the schema of the output of your HTTP request in the previous step.

![HTTP action configuration](image)

![Parse JSON action configuration](image)
Your flow should now look as shown below

- Manually trigger a flow
- Get items
- Initialize variable
- Apply to each
- HTTP
  - Method: POST
  - Headers: Enter key, Enter value
  - Body: Owners
- Parse JSON
  - Content: Body
  - Schema:
    ```json
    {
      "type": "array",
      "items": {
        "type": "string"
      }
    }
    ```
  - Use sample payload to generate schema
Task 3: Getting deals by owner and sending them as CSV attachments

- The **body** returned by the **Parse JSON** action should now contain the deduplicated list of owners.

- Next, we are going to perform the following steps **for each** owner in the list:
  1. Get deals assigned to owner
  2. Select columns we need
  3. Create CSV table
  4. Send email notification

- Start with an **Apply to each** right after **Parse JSON** and iterate its output: **body**

  ![Apply to each](image)

- Inside the loop, get deals assigned to each owner by configuring the OData filter query as shown (Owner eq '[Current item]'). *If you’re using your own SharePoint list, replace Owner with appropriate column name*

  ![Get items](image)

- Next, Select the columns you need from the outputs of the **Get Items 2 (!!!)** (Do *not* use outputs from Get items) action and map them to appropriate column names of your choosing.
Next, create the CSV table using the data output from the Select action. Remember to include headers.

The final step is to send the email

1. In the To field – add your own email address
2. Setup Subject, Email Body, and Attachment File Name as shown

3. [Important] The attachment field only takes in as input - binary data, so we will need to covert our CSV table to binary using the binary() expression. Configure as shown
• That’s it – You’re ready to save and test! If you have followed the steps correctly – you should see a few emails in your inbox that have a CSV document attached to it. These emails contain data that is relevant only to the lead owners. In a real-world scenario, you’d be sending emails to actual lead owners.

Pro-tip: You can enable concurrency control, that we learnt about in Exercise 2, on the Apply to each 2 loop – as the actions within the loop are not dependent on each other across the loop.
Your flow should at this point look as shown below
Handling errors
We will build on Exercise 4.1 and Exercise 4.2 in this section. Please ensure you have completed those exercises before diving into this section.

Exercise: Using parallel branches to handle errors
Task 1: Add a parallel branch
- Edit the flow we created in the previous exercise (Exercise 4.2)
- Next, add a parallel branch and a send email action between HTTP and the Parse JSON branch

Task 2: Handle error in the HTTP action
- Select the ... contextual menu on the Send an email notification 2 action
- Then select – Configure run after
  - On the follow configuration page – uncheck: is successful and check: has failed and has timed out
• Select Done
• Configure the Send an email notification action as shown below. Include your own email address in the To and Save your flow

Your flow should now look as shown –

What we have done is that – if the HTTP action succeeds then – the workflow follows the Parse JSON branch, but if the HTTP action times out or fails then it will follow the Send an email notification 2 branch

Task 3: Simulating Failure
• Turn off the flow we created in Exercise 4.1
• Run flow we added the error handling step to (Exercise 4.2)
  You should have received an email informing you of the failure.
Congratulations! You have completed the Advanced Flow patterns and practices lab. Go ahead and start automating!

Additional Resources
http://aka.ms/flow-resources