AUTOMATION: POWER AUTOMATE

Streamline User Access Requests & Approval with Power Automate

Eliminate Manual Steps and Speed Up User Access Approval for Faster, More Efficient Workflows

Avishek Ghosh (AV_DEVS) · Follow

Published in Microsoft Power BI · 20 min read · Dec 11, 2024

Microsoft

Imagine your development team drowning in a sea of user access requests. New reports are gaining traction, users are eager to dive in, and your team is caught between delivering cutting-edge solutions and managing an endless stream of access permissions. Each manual access grant is a potential minefield — a mistyped user ID or incorrect role could trigger a cascade of security nightmares.

Enter Power Automate — this powerful tool transforms the chaotic, error-prone process of user access management into a streamlined, automated

workflow. No more context-switching between development tasks and permission requests. Instead, Power Automate creates a seamless process that:

- Captures detailed user access requests
- Routes approvals to the right stakeholders
- Automatically provisions access
- Maintains a bulletproof audit trail

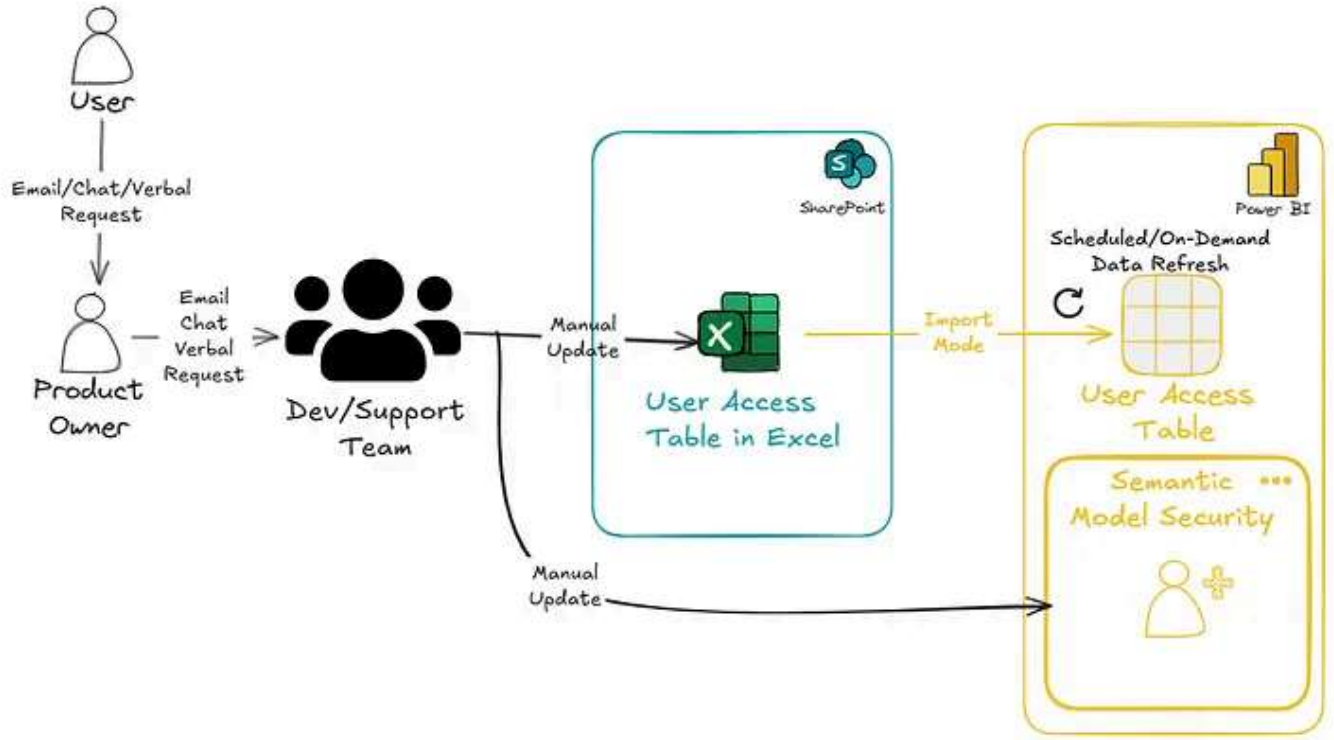
In this article, we'll unveil a step-by-step blueprint for building a Power Automate flow that takes the pain out of user access management. We'll show you how to create a solution that not only saves time but also dramatically reduces the risk of human error.

Imagine freeing your developers from the mundane tasks of access management, allowing them to focus on what they do best — creating innovative solutions. With Power Automate, that's not just a dream — it's your new reality.

Get ready to transform your access management from a headache to a high-performance, automated process.

The Problem Statement

Let's start by looking at the typical "As-Is" process for granting user access to Business Intelligence reports. While the specifics may vary slightly across industries, we'll focus on a straightforward process that's commonly used, especially for reports with dynamic Row-Level Security (RLS) in place.



Schematic by author

Often, when there isn't a well-defined process in place, convenience takes over, and we fall into inefficient routines. Here's a typical example:

Imagine a Power BI report with dynamic Row-Level Security (RLS) applied, using users' User Principal Names (UPNs) to assign roles or provide access to specific departments or regions. These assignments are managed through a user access table stored in SharePoint, which is accessible by the development or support team. The Power BI report imports this table directly from SharePoint.

Whenever a new user needs access, their UPN must first be added to the user access table in SharePoint. Next, the same UPN has to be manually added to the **Security** settings in the Semantic Model, which was created during the report's publication in the Power BI Service workspace.

Once these two manual steps are completed, we have two options to update the user access table in Power BI's data model:

1. **Wait for a Scheduled Refresh**: Allow the next scheduled refresh to automatically include the new user's data.
2. **Perform an On-Demand Refresh**: Trigger a manual refresh in the Power BI Service workspace to update the data model immediately and reflect

the new user's access.

While refreshing the data model is straightforward, the real challenge lies in promptly adding new users and maintaining proper documentation or a paper trail for future reference. This often leads to delays and inconsistencies in the process.

Here are some common issues that can arise when granting user access:

1. **Interruptions and Delays:** Let's face it — granting user access is a tedious task. Breaking away from development work to add users often feels like a hassle, leading to missed requests or significant delays.
2. **Input Errors**: Mistakes can easily happen, such as entering an incorrect UPN, assigning the wrong role, or misclassifying the user's department or region (which defines their access scope in the report). These errors can disrupt the report's functionality or data security.
3. **Lack of Communication**: Even when the access is granted correctly, failing to notify stakeholders can create confusion and unnecessary follow-ups.
4. **Disputes Over Documentation**: Without a proper documentation trail, verbal or informal requests can lead to disputes later. For instance, a

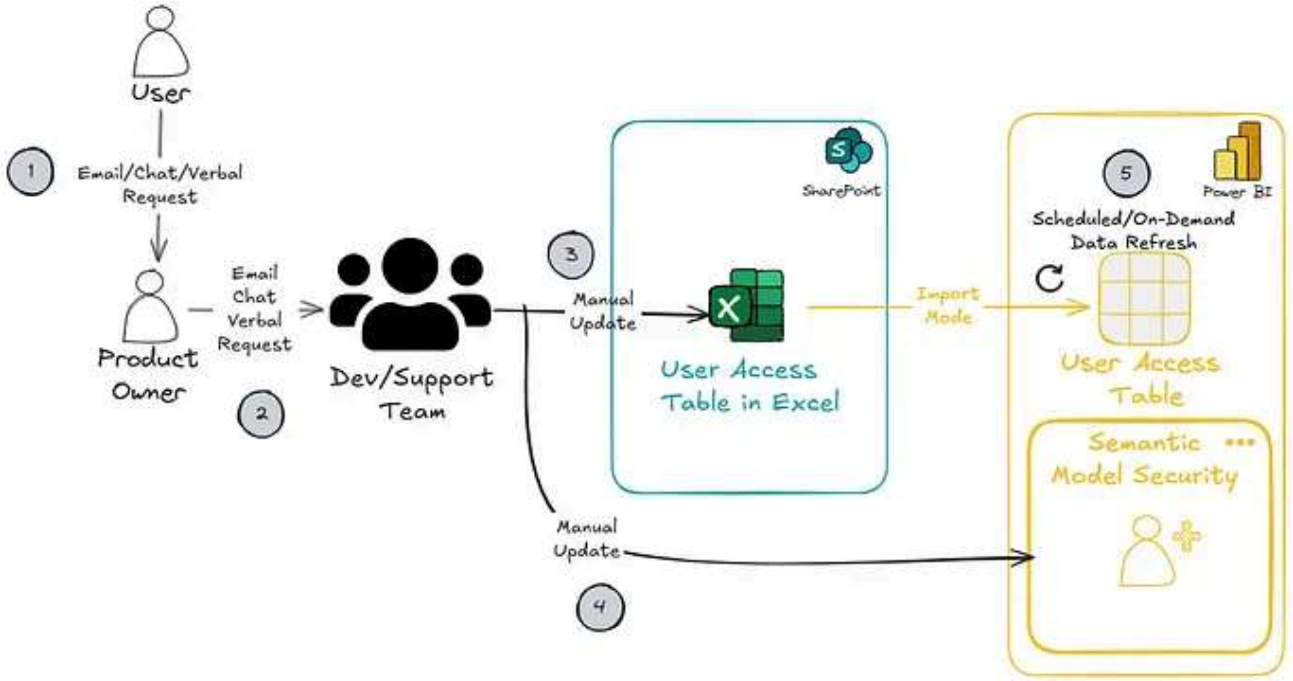
stakeholder might claim a request was made, but without written proof, it becomes difficult to verify or resolve the issue.

5. **Scaling Challenges**: As the number of users grows, managing access manually becomes increasingly unmanageable, leading to bottlenecks and potential lapses in security or efficiency.

Addressing these issues requires streamlining the process, automating repetitive tasks, and ensuring clear communication and documentation practices.

The Solution

Our first objective is to identify the scope of automation in the entire process. By glancing at the As-is process schematic we can deduce that the part that best fits the scope of automation step 1, 2, 3 and 5 from the below diagram.



Schematic by author

In our “To-Be” process, we aim to streamline user access requests by standardizing interactions and providing a simple, user-friendly solution: **Microsoft Forms.**

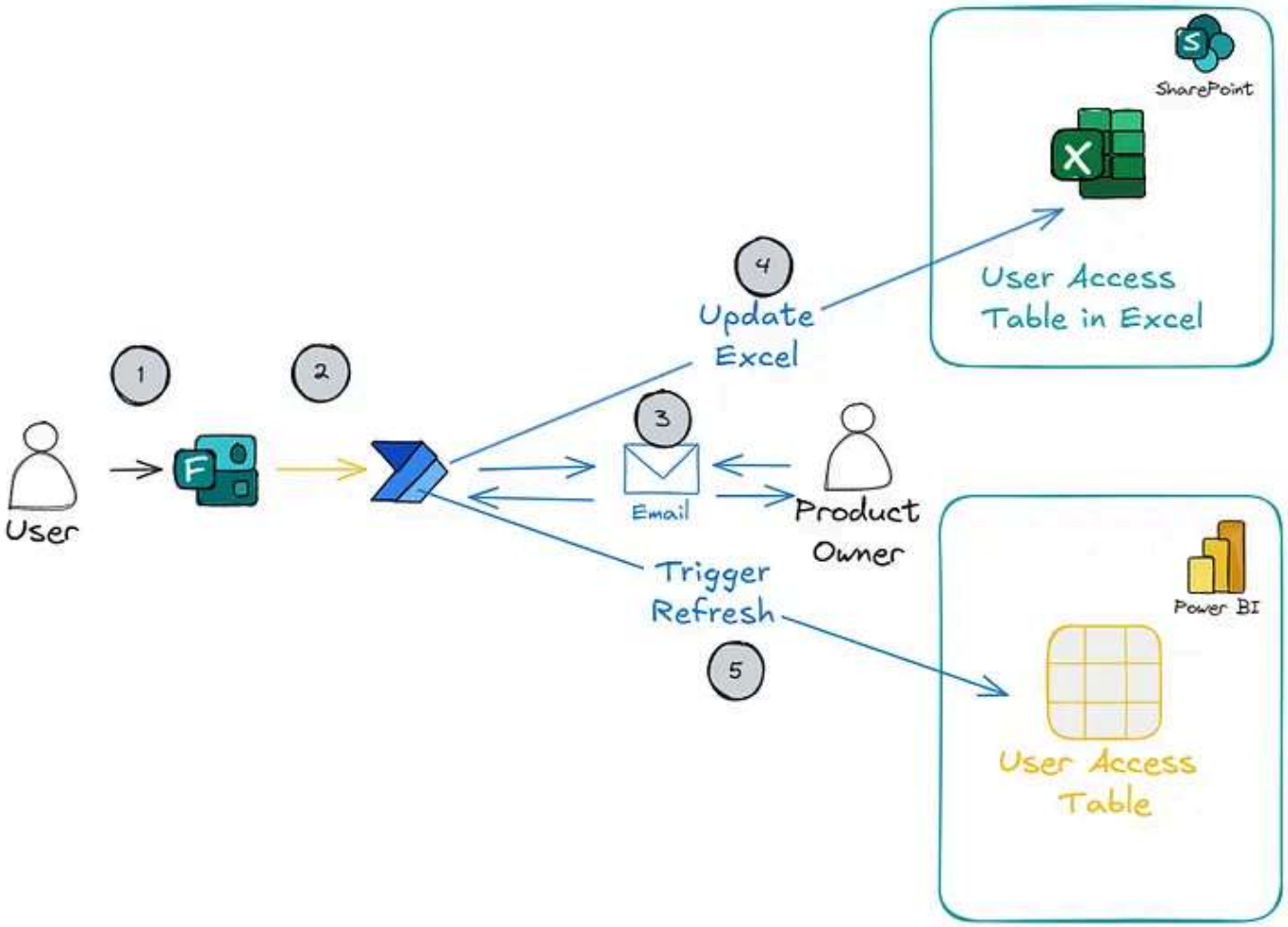
How It Works

We create a Microsoft Form with fields that collect user details in a structured way, ensuring data consistency. Conditional fields allow for dynamic inputs based on previous selections, making the form intuitive and efficient.

Example Form Fields:

- **Name:** *(Free text)*
- **Designation:** *(Free text)*
- **Department:** *(Free text)*
- **Report Visibility:** *(Single-select: Global/Regional/Local)*
- **Regions to Select:** *(Multi-select, visible only if "Regional" is selected)*
- **Country to Select:** *(Multi-select, visible only if "Local" is selected)*
- **Business Rationale:** *(Free text)*
- **Feedback:** *(Optional free text)*

Process Flow



Schematic by author

1. **Accessing the Form:**
   Embed the form link on the report or app page with a clear, user-friendly icon. This allows users without report access to submit requests conveniently and in a standardized format.

2. **Submission and Automation:**
   Once a user submits the form, it triggers a **Power Automate** flow to process the data, including their selected access options and other details.

3. **Approval Workflow:**

- The flow sends an email with the request details to the product owner or designated approver.
- The approver can approve or reject the request, adding comments if necessary.

4. **Updating Access:**

- If approved, the flow updates the **user access table** in SharePoint with the new details.
- If rejected, an automated email informs the requester, including the approver's comments.

5. **Seamless Report Refresh:**
After updating the SharePoint table, the flow triggers an **on-demand refresh**

of the report's semantic model in the Power BI workspace. This ensures the changes are immediately reflected in the report's data model.

Here's a detailed step-by-step guide to setting up the Power Automate flow for this process:

Scenario Overview

Imagine a Power BI report with two levels of user access: **Regional** and **Local**.

- **Regional Access**: Users can view data for specific regions.
- **Local Access**: Users can view data for specific countries.

The access request form is already in place. It first asks users to select the desired report visibility — **Regional** or **Local**. Based on their choice, the form dynamically displays a list of regions or countries for them to select.

Objective

Our goal is to create a Power Automate flow that:

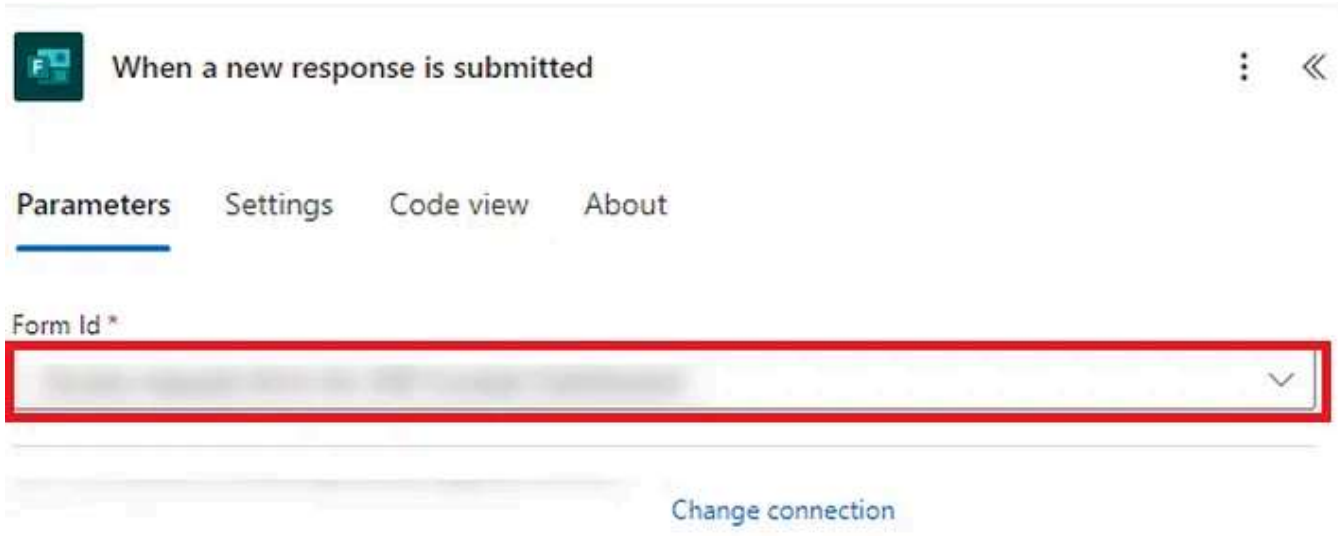
1. Extracts the user's selections from the form.
2. Generates a list (region or country) based on their choice.

3. Attaches this list as a text document in an approval email sent to the approver.
4. Uses the approved list to update the user access table in SharePoint.

This flow ensures a seamless and automated way to manage access requests while maintaining data accuracy and reducing manual effort. Let's dive into the steps to set it up!

Step 1: Set Up the Trigger

1. **Add Trigger Activity**:
   In Power Automate, create a new flow and choose "**When a new response is submitted**" as the trigger.
2. **Select the Form**:
   From the dropdown, select your **User Access Request Form**. This ensures that the flow is activated whenever a user submits the form.



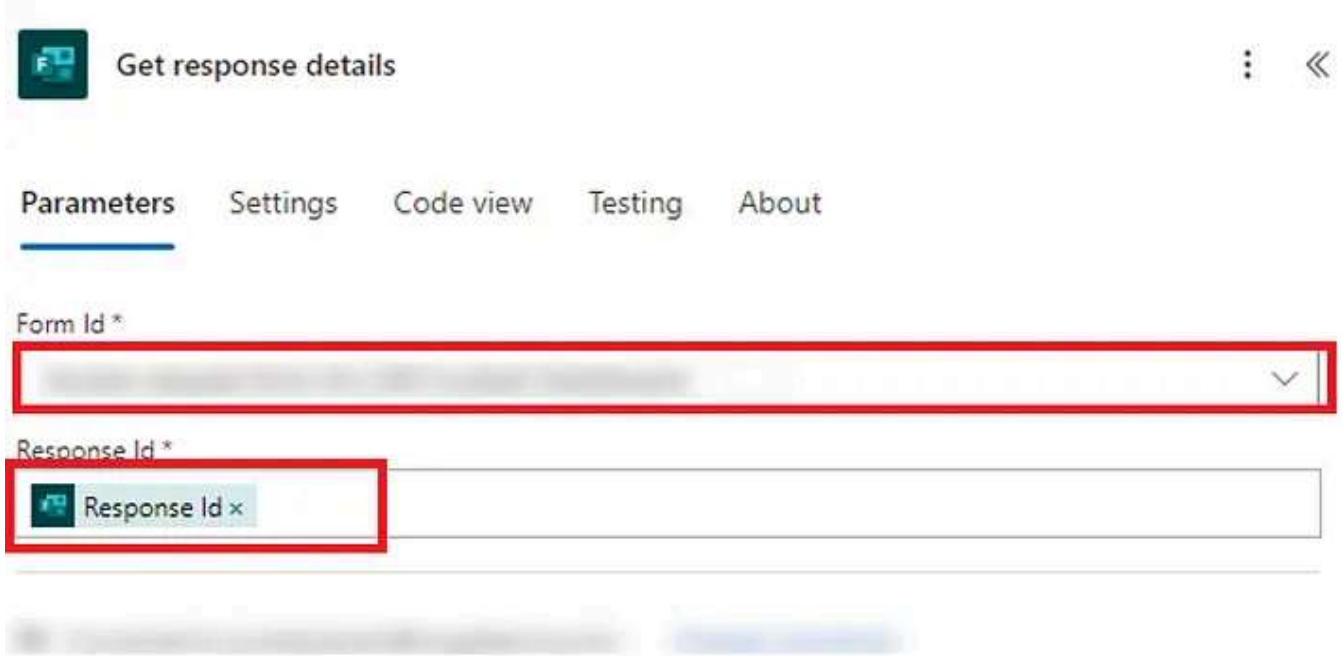
Step 2: Extract Form Details

1. **Add the Action:**
   Click + **New Step** and select the action **"Get response details"** from Microsoft Forms.

2. **Select the Form:**
   In the dropdown menu, choose the same **User Access Request Form** you used in the trigger step.

3. **Map the Response ID:**
   In the **Response ID** field, select **"Response Id"** from the dynamic content

menu. This ensures the flow retrieves the specific details submitted in each form response.



Step 3: Initialize an Array Variable

1. **Add the Action:**
   Click + **New Step** and choose "**Initialize variable**" from the available actions.

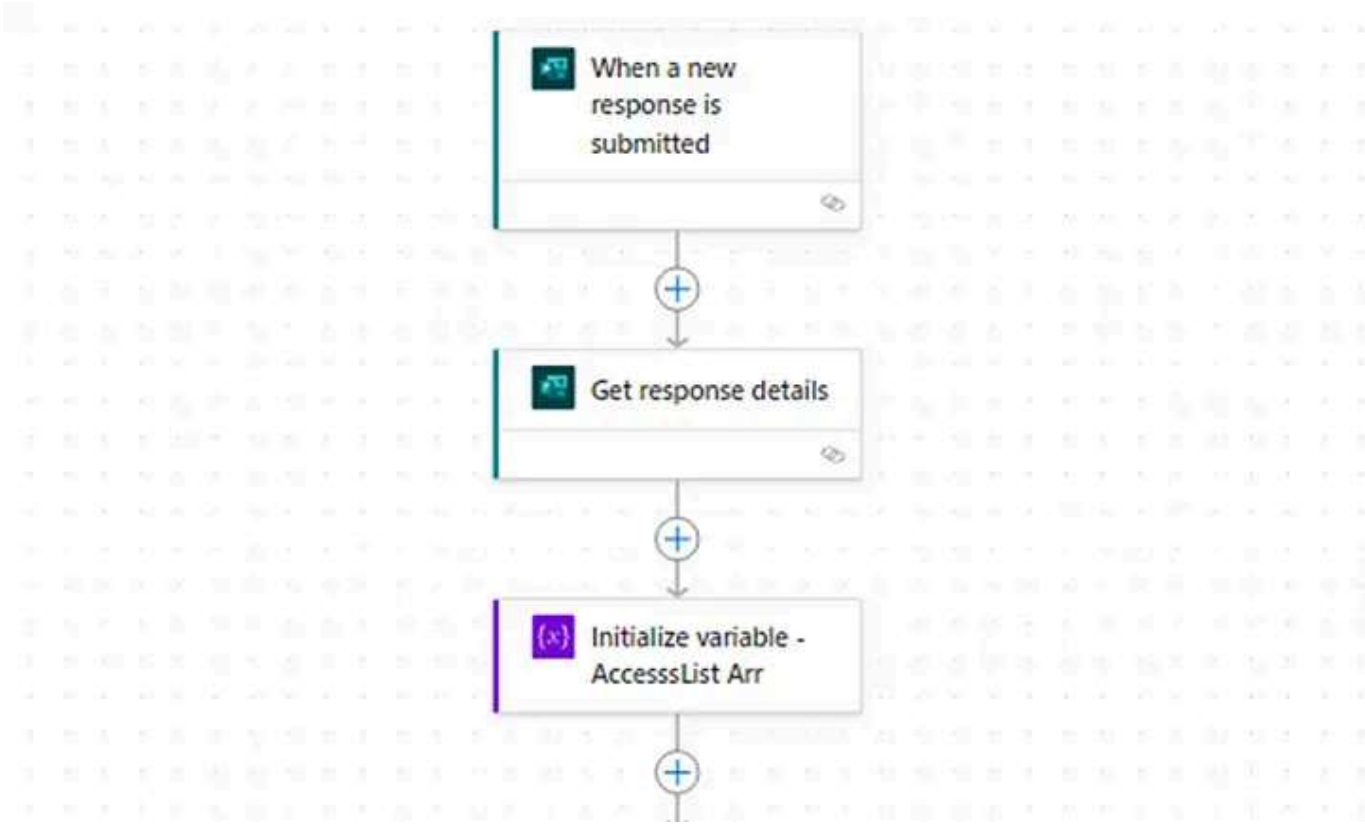
2. **Set the Variable Name:**
   Give the variable a descriptive name, such as "**SelectedAccessList**", to

clearly indicate its purpose.

3. **Select Variable Type:**
   Set the **Type** to **Array**.

This creates an empty variable which we will use later to fill in our region/country list for user access.

At the end of these 3 steps the flow will look like this:



Step 4: Create a Switch Action to Handle Region or Country Selection

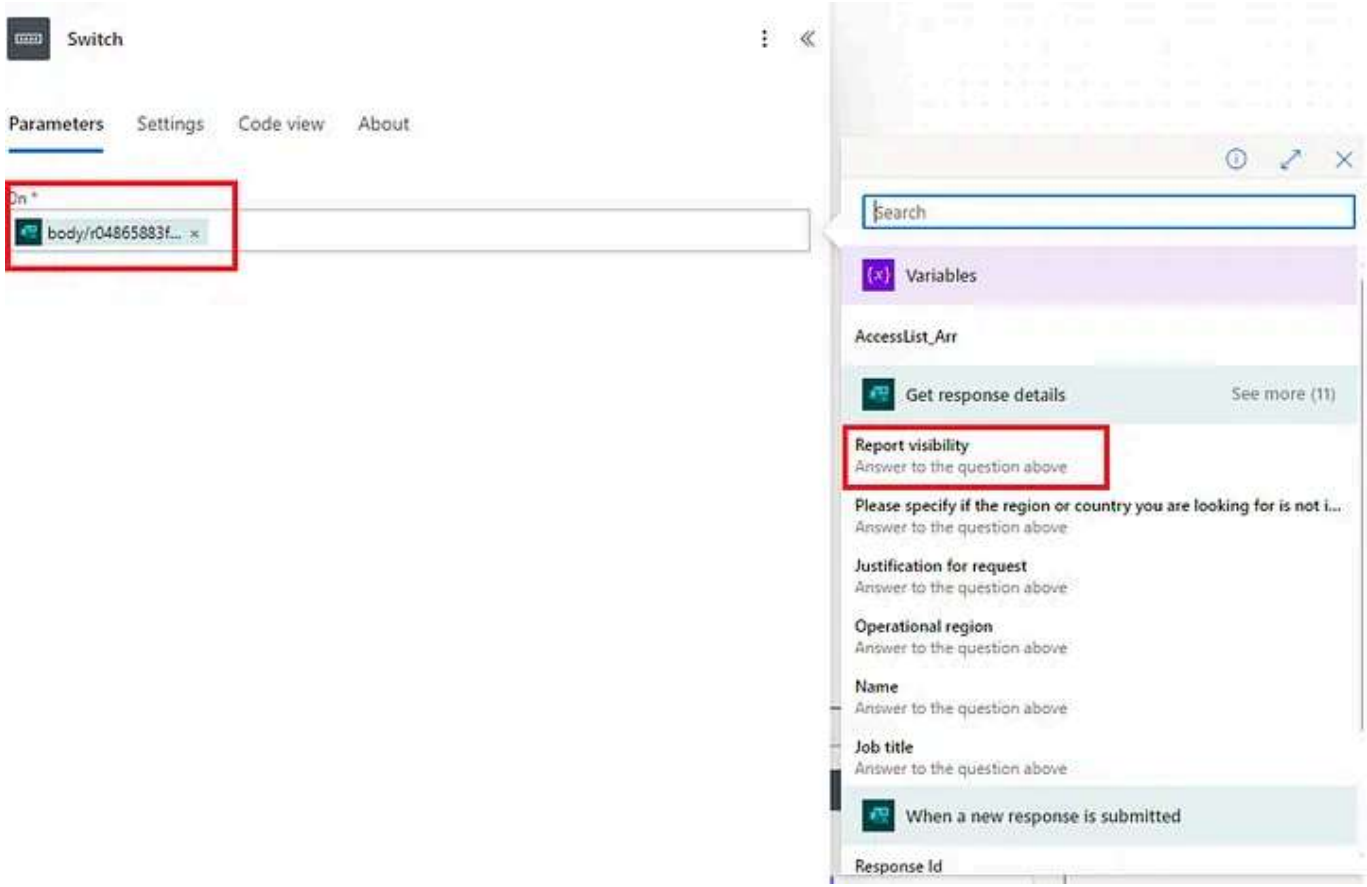
1. **Add the Switch Action:**
   Click + **New Step** and select **"Switch"** from the available actions. This functions similarly to a SWITCH() function in DAX, acting like an IF statement to evaluate a condition.

2. **Set the Condition Field:**
   In the **On** field of the Switch action, select the **"Report Visibility"** field

from the dynamic content menu. This ensures the Switch action evaluates the user's selection from the form.

3. **Define Branches**:
   Create three branches within the Switch action:

- **Regional**: Executes if the user selected "Regional" in the Report Visibility field.
- **Local**: Executes if the user selected "Local" in the Report Visibility field.
- **Default**: A fallback branch to handle unexpected or empty inputs.



Step 5: Configure the First Branch Logic

1. **Select the First Branch:**
   Inside the **Switch** action, click on the first branch.

2. **Set the Condition:**
   In the **Equals** field, type **"Regional"**. This ensures that the logic in this branch executes only when the user has selected "Regional" in the **Report Visibility** field.

It means that this branch will enable if the **Report Visibility** component of the form returns the text “Regional”.

Similarly, we can set-up the alternate branch with “local” paralally.

Step 6: Extract Regional/Country List in JSON

First Branch: Regional

1. **Add the “Parse JSON” Action:**
   Inside the **Regional** branch, click **+ Add an action** and select **“Parse JSON”.**
2. **Set the Content:**
   In the **Content** field, select the dynamic content **“Select Region”** from the

form submission. This captures the regions chosen by the user when they select **Regional**.

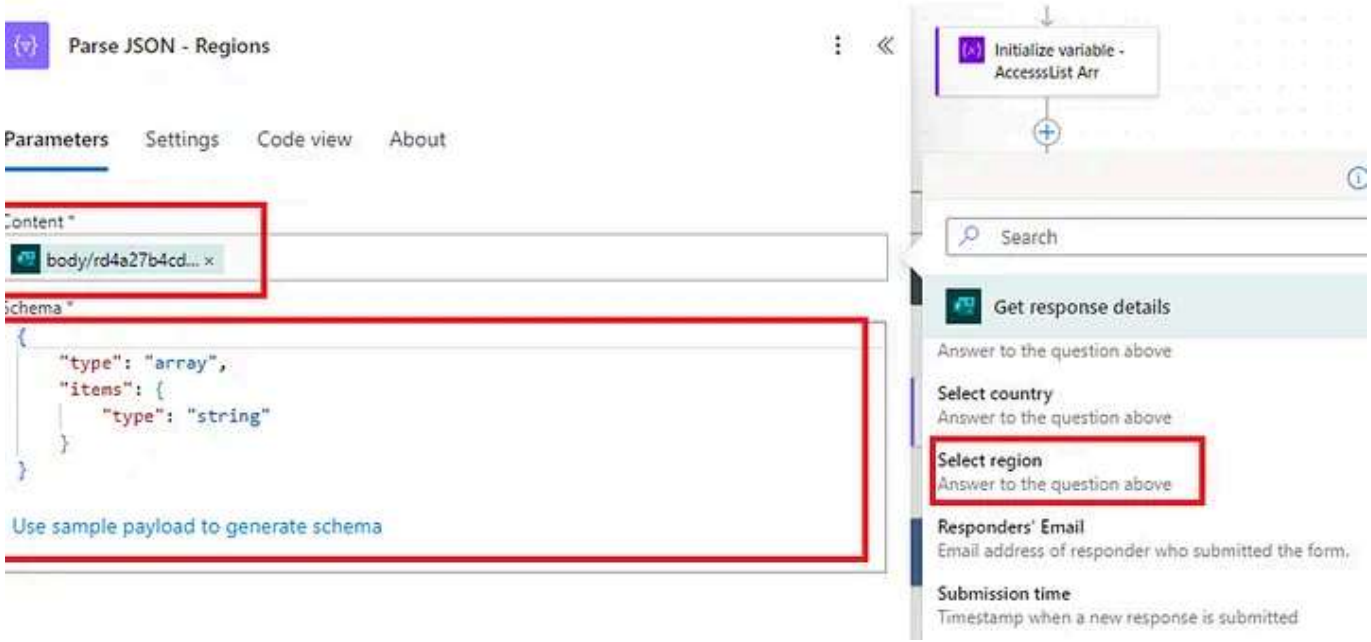
3. **Define the Schema**:
   Use the **Generate from Sample** option to define the JSON schema. Copy a sample output from a form submission where regions are selected (e.g., `["Region1", "Region2"]`) and paste it into the prompt to auto-generate the schema.

Second Branch: Local

1. **Repeat the Steps for Local**:
   Inside the **Local** branch, add a **Parse JSON** action and follow the same steps.

2. **Set the Content**:
   In this branch, select the dynamic content **"Select Country"** from the form submission.

3. **Define the Schema**:
   Use the **Generate from Sample** option with a sample JSON output for countries (e.g., `["Country1", "Country2"]`) to create the schema.



Note: Remember to enter the schema in the exact same way it is displayed.

Step 7: Set Up a "For Each" Loop

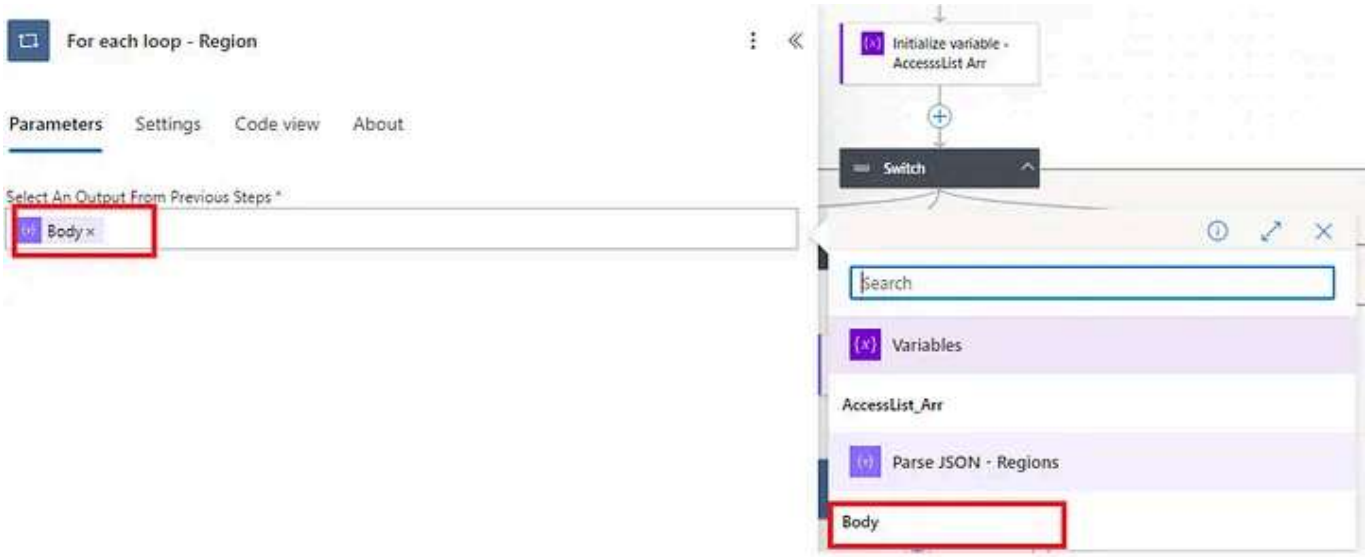
1. **Add the "For Each" Action:**
   Inside both the **Regional** and **Local** branches, click + **Add an action** and select "**For each**" from the available actions.
2. **Set the Input:**

- For the **Regional** branch, set the "**From**" field to the **Body** output from the **Parse JSON** action for regions.

- For the **Local** branch, set the **"From"** field to the **Body** output from the **Parse JSON** action for countries.

3. Define the Loop Logic:
Inside the **For Each** action, you can now specify tasks to be performed on each element (region or country) extracted from the JSON. For example:

- Append the region or country to the array variable initialized earlier.
- Log each region or country for tracking purposes.



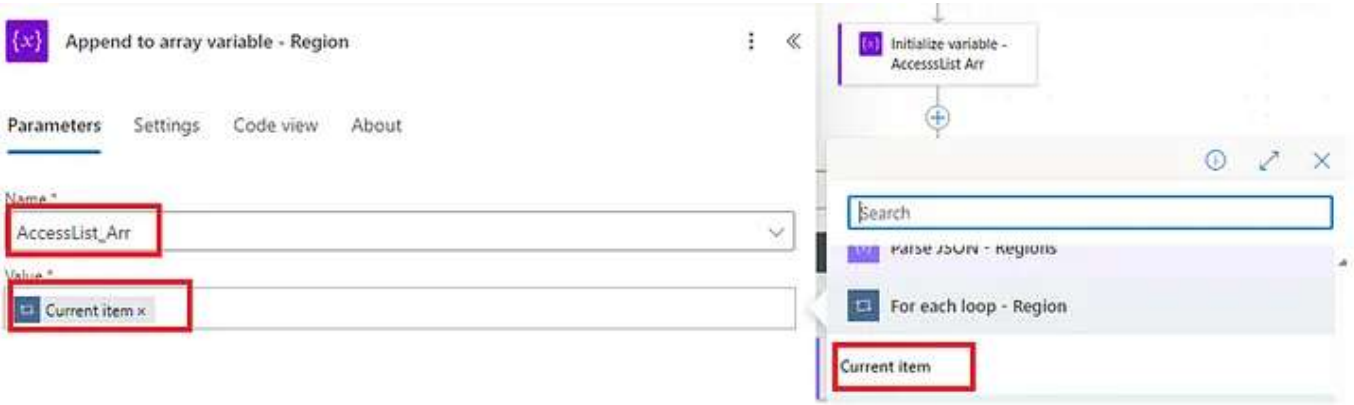
Step 8: Append to Array Variable

1. **Add the "Append to array variable" Action:**
   Inside the **For Each** loop, click **+ Add an action** and select **"Append to array variable"**.

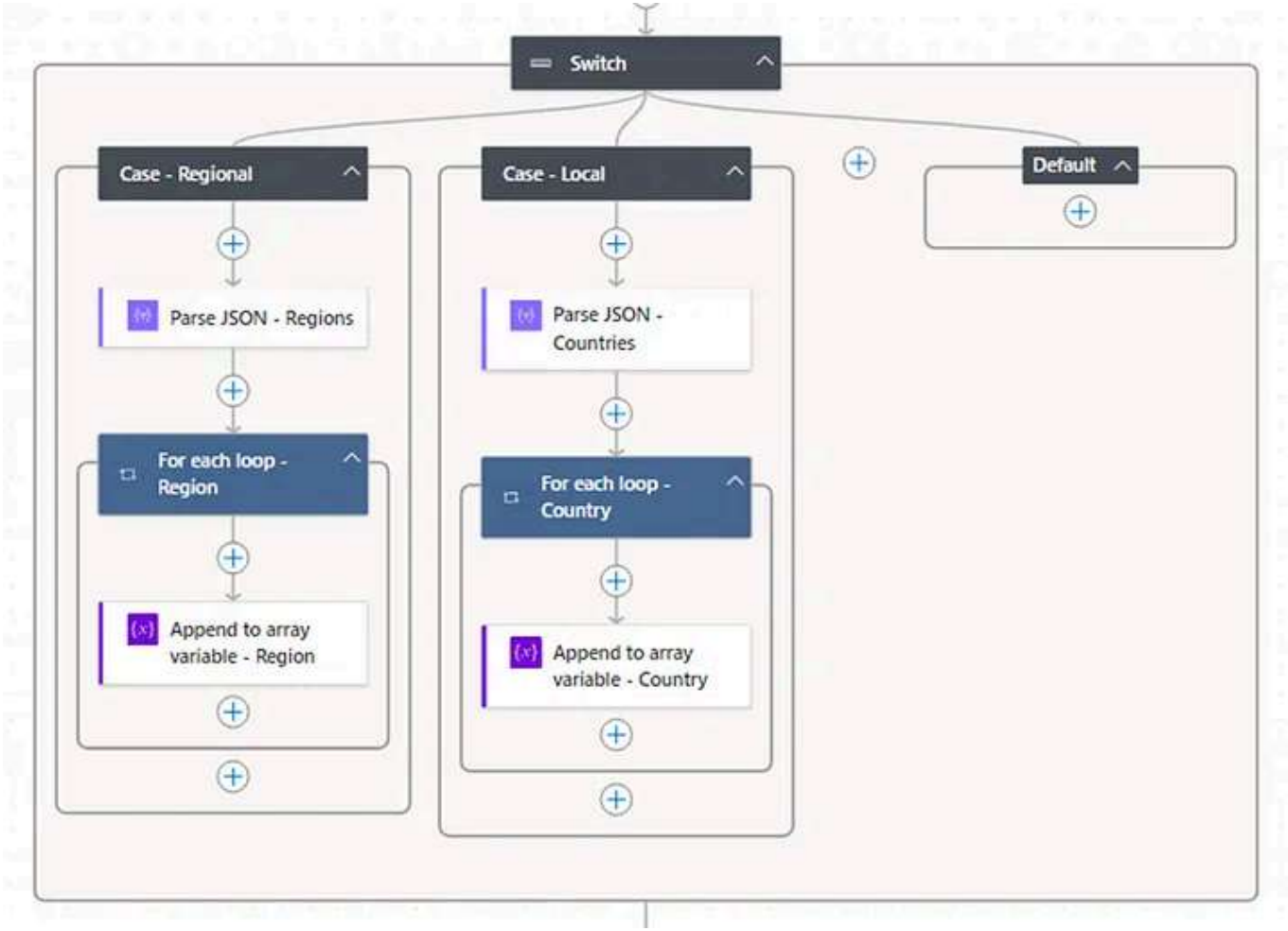
2. **Select the Array Variable:**
   In the **Name** field, select the **array variable** you initialized earlier (e.g., **SelectedAccessList**). This is where the region or country names will be added.

3. **Set the Value to Append:**

- For the **Regional** branch, in the **Value** field, select **"Current item"** from the dynamic content. This represents each region being processed in the loop.
- For the **Local** branch, in the **Value** field, select **"Current item"** as well, representing each country being processed.

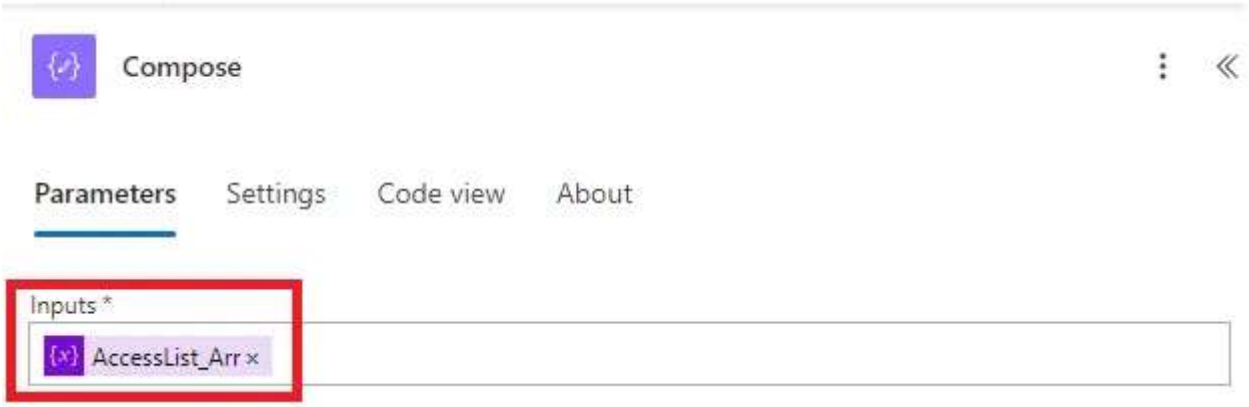


With this step our branching ends and this section of the flow now looks something like this:



Step 9: Check the Output of the Last Step

1. **Add the "Compose" Action:**
   Click + **New Step** and select **"Compose"** from the available actions. The **Compose** action will allow you to check the output of the array variable and ensure it's correctly populated.
2. **Set the Input for Compose:**
   In the **Inputs** field, select the **array variable** (e.g., **SelectedAccessList**) from the dynamic content.



At this stage the contents of the variable will appear in an array format:

["North America", "LATAM", "JAPAC", "EMEAC"]

We need to convert it into a string where each region or country should start in a new line to give it a list like appearance:

North America
LATAM
JAPAC
EMEAC

Step 10: Initialize a String Variable

1. **Add the "Initialize variable" Action:**
   Click + **New Step** and select "**Initialize variable**" from the available actions.
2. **Configure the Variable:**

- In the **Name** field, give the variable a meaningful name, such as **FormattedRegionList**.
- In the **Type** field, select **String** as the variable type. This will allow you to store and manipulate the output as text.

3. Set the Initial Value:
Leave the **Value** field empty for now, as we will populate this variable later using the formatted string from the previous step.

Step 11: Create Another "For Each" Loop

1. **Add the "For Each" Action:**
   Click + **New Step** and select "**For each**". This action will allow us to loop through each item in the array again and append it to the string variable.

2. **Set the Input for the Loop:**
   In the "**From**" field, select the **array variable** (e.g., **SelectedAccessList**) that contains the regions or countries. This is the array you built earlier and now want to iterate through.

3. **Append Each Item to the String:**
   Inside the loop, add a "**Append to string variable**" action.

4. **Configure the Action:**

- In the **Name** field, select the **String variable** you initialized earlier (e.g., **FormattedRegionList**).
- In the **Value** field, use the dynamic content **"Current item"**, which represents each region or country in the loop.
- Add the newline character `\n` after each **"Current item"** to ensure each region or country is placed on a new line. The value should look like this:

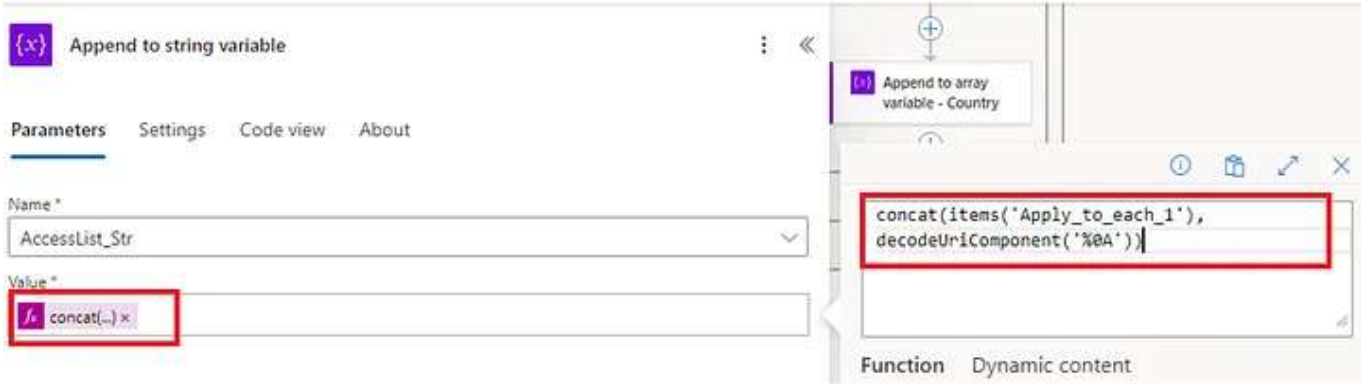
Step 12: Append to String Variable

1. **Add the "Append to string variable" Action:**
   Inside the **For Each** loop, click **+ Add an action** and choose **"Append to**

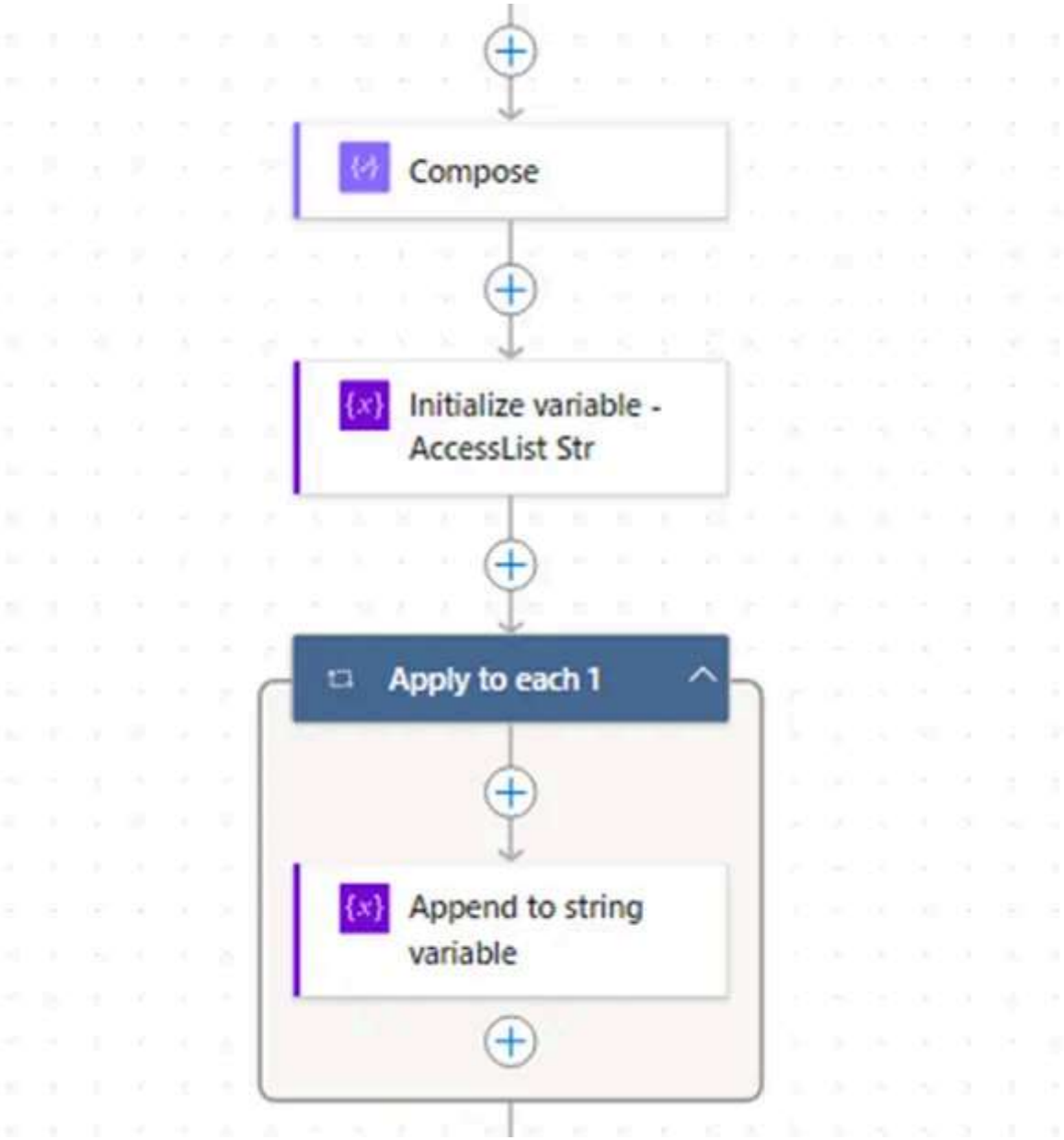
string variable”. This action will allow us to add each item in the list to the string variable we initialized earlier.

2. **Select the String Variable:**
   In the **Name** field, select the **string variable** you initialized earlier (e.g., **FormattedRegionList**).

3. **Add the Expression to Append:**
   In the **Value** field, instead of directly using **Current item**, we will use an expression to ensure each element is added on a new line. Enter the following expression:



At the end of this step this section of the flow will look something like this

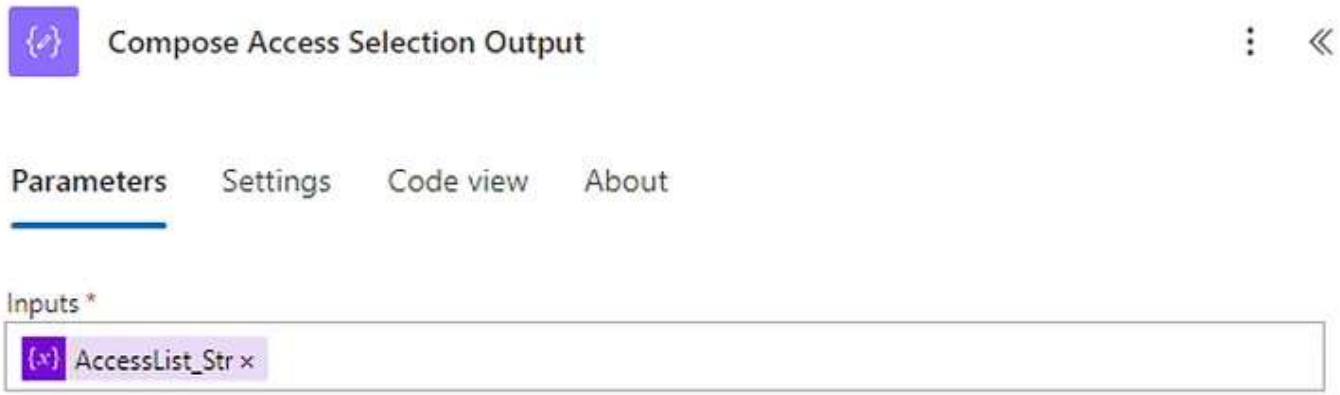


Step 13: Test Extraction Result

1. **Add the "Compose" Action:**
   After the "**Append to string variable**" action, click **+ Add an action** and select "**Compose**". This action helps you view the output of a variable and is especially useful for debugging.

2. **Set the Input for Compose:**
   In the **Inputs** field, select the **string variable** you just updated (e.g., **FormattedRegionList**) from the dynamic content. This will allow you to see the final result of the appended list.

3. **Check the Output:**
   After running the flow, go to the **Run History** and check the output of the **Compose** action. It will display the string variable with each region or country listed on a new line.



We can now see that the contents appear like below:

North America

LATAM

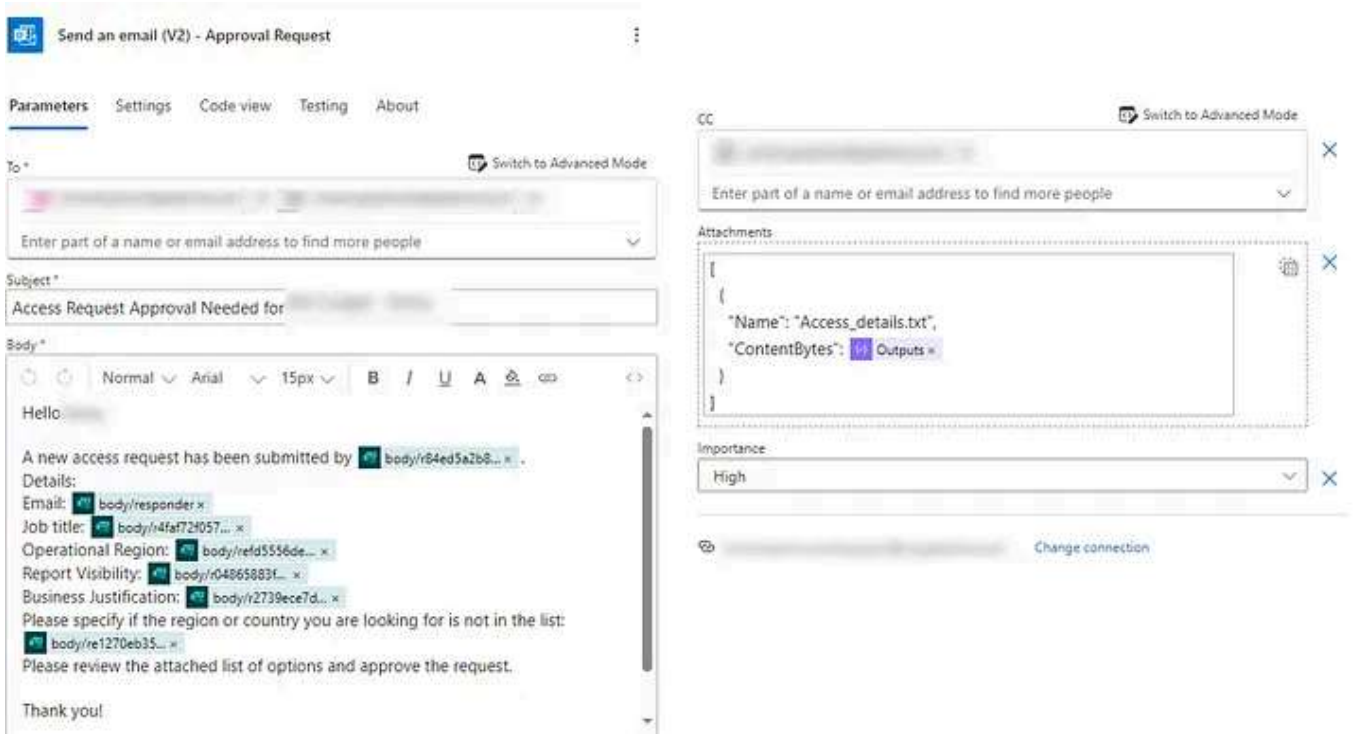
JAPAC

EMEAC

Step 14: Compose an Email with User's Details

1. **Add the "Send an email (V2)" Action:**
   Click + **New Step** and select **"Send an email (V2)"** from the available actions. This step will send an email with the user's details, providing the approver with all the relevant information about the access request before they make their decision.
2. **Configure the Email Fields:**

- **To:** Enter the email address of the **approver** (e.g., product owner or manager).
- **Subject:** Set a subject for the email, such as "User Access Request Details."
- **Body:** In the email body, use a combination of static text and dynamic elements from the form responses to give context.



3. Attach the List of Regions/Countries:

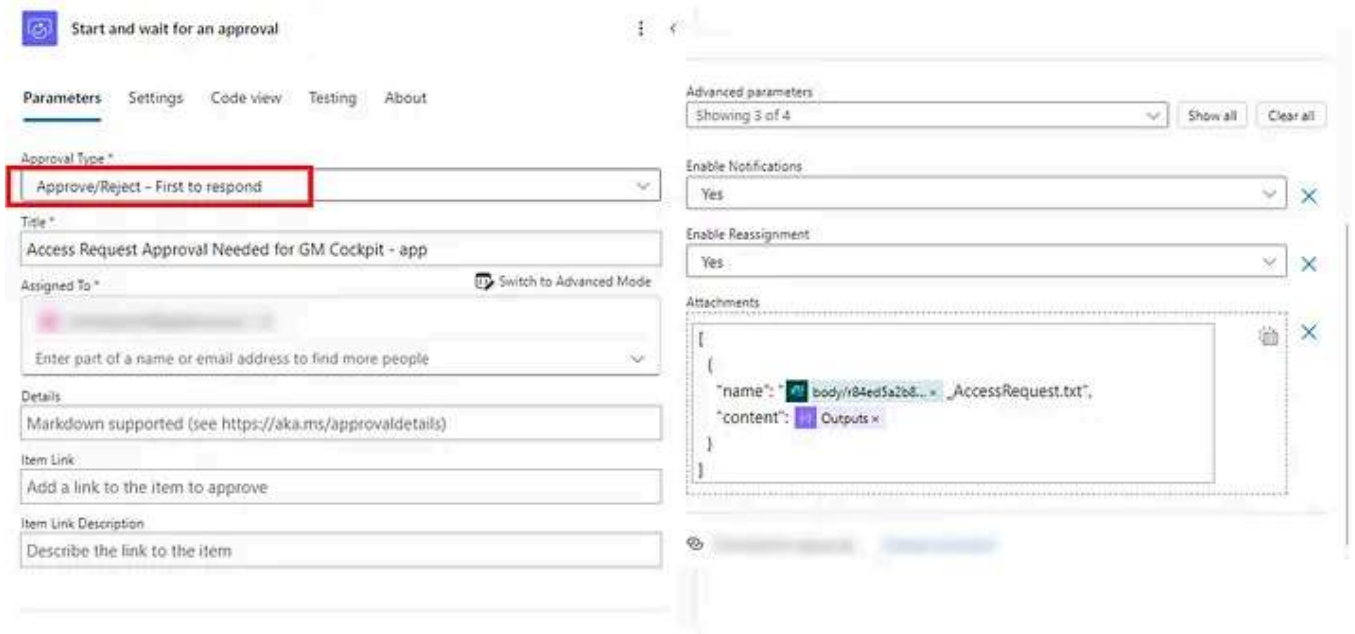
- In the **Attachments** section, add a new attachment.
- Set the **Name** field to something descriptive like “Access List.txt”.
- For the **File Content**, select the **FormattedRegionList** (the string variable you populated in earlier steps). This will attach the list of regions or countries as a text file to the email.

4. Purpose of the Email:
This email is designed to provide the approver with all the relevant details before they approve or deny the access request. The email will arrive **before** the formal approval request email, giving the approver context and helping them make a quicker decision.

Step 15: Approval-Seeking Email

1. **Add the "Start and Wait for an Approval" Action:**
   After composing the initial email with the user's details, the next step is to send the formal approval request. Click **+ New Step**, search for **"Start and wait for an approval"**, and select it.

2. **Set the Approval Type:**
   In the **Approval type** dropdown, select **"Approve/Reject — First to respond"**. This allows the approver to either approve or reject the request, and the process will proceed as soon as one of the approvers responds.



3. Attach the List of Regions/Countries:

Just like the previous step, attach the list of regions or countries as a text file. Use the **FormattedRegionList** variable as the attachment content.

Why Two Emails?

It's important to note that we are sending **two emails** in this process:

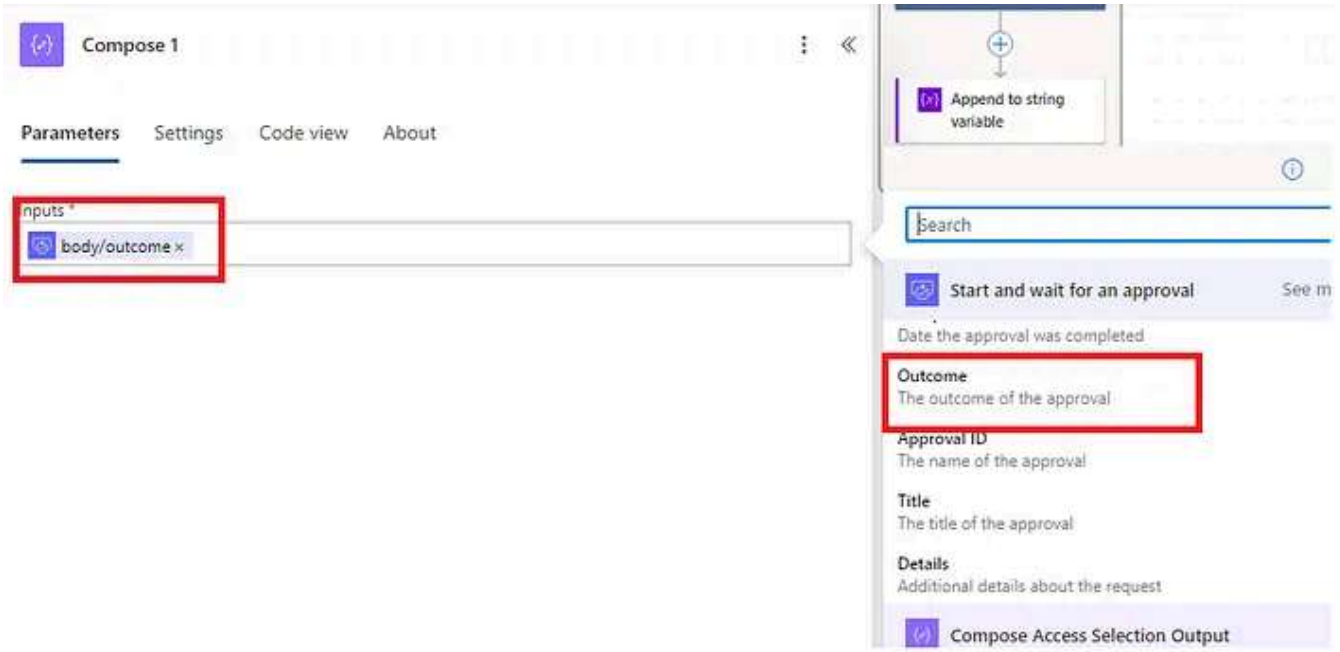
- The **first email** provides the approver with all the details in a clear format, along with the list of regions/countries for access.
- The **second email** is the formal **approval-seeking** email, which is required because the approval email system doesn't allow for a detailed

layout of the user information.

Step 16: Check for Approver's Response

1. **Add a "Compose" Action:**
   To validate the approver's response, add a **Compose** action. This step allows us to inspect the output of the approval process and decide what to do next based on whether the approver selected **Approve** or **Decline**.

2. **Use Dynamic Content for Outcome:**
   In the **Compose** action, use the **"Outcome"** dynamic content from the previous **"Start and wait for an approval"** step. This dynamic content will contain the response from the approver, which will either be **Approve** or **Reject**.



3. Use Compose for Debugging and Validation:

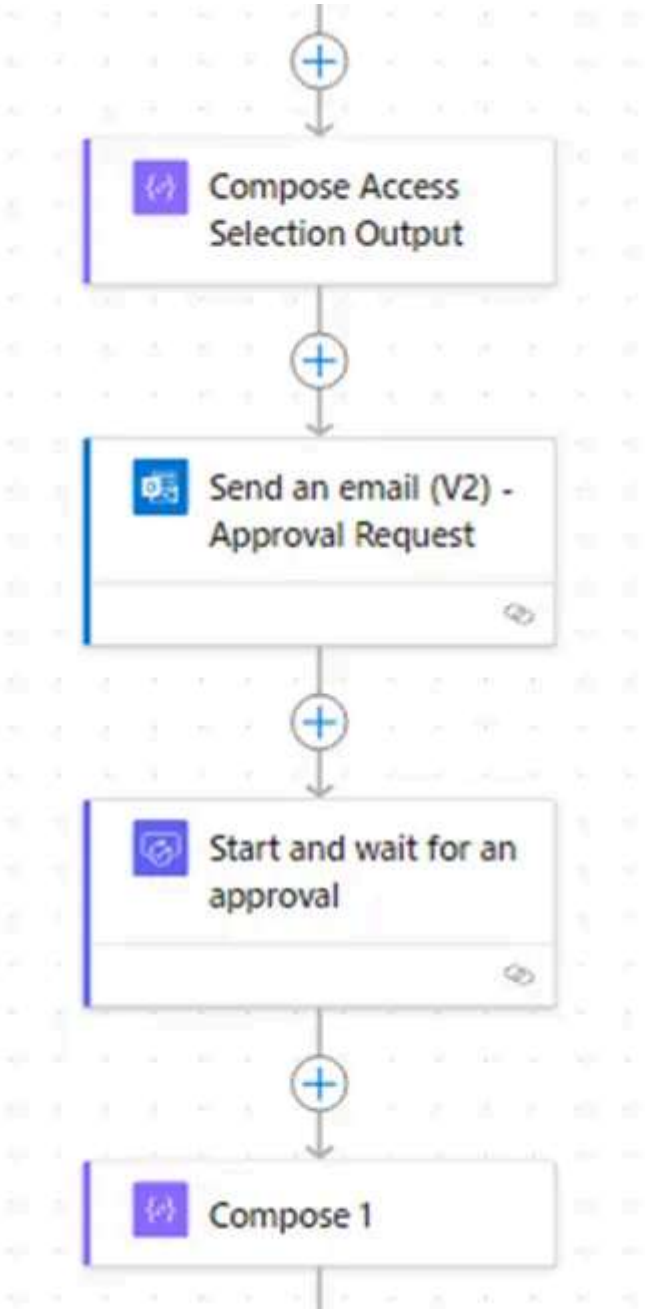
At this stage, the **Compose** action will show the outcome of the approver's response. It's useful for debugging and making sure that the approval process is working as expected.

4. Condition Based on Response:

After checking the response, you can set up a **Condition** step to proceed with different actions depending on whether the outcome is "Approve" or "Reject". For example:

- If **Approve**: Proceed to update the SharePoint user access table and refresh the Power BI model.
- If **Reject**: Send a rejection email to the requester with the approver's comments.

At this stage, this section of the flow looks like this:



Step 17: Create Condition for Approved and Declined

1. **Add a "Condition" Action:**
   To handle the decision-making process after receiving the approver's response, add a **Condition** action. This action will evaluate whether the request was **Approved** or **Declined** by comparing the result with the expected values.

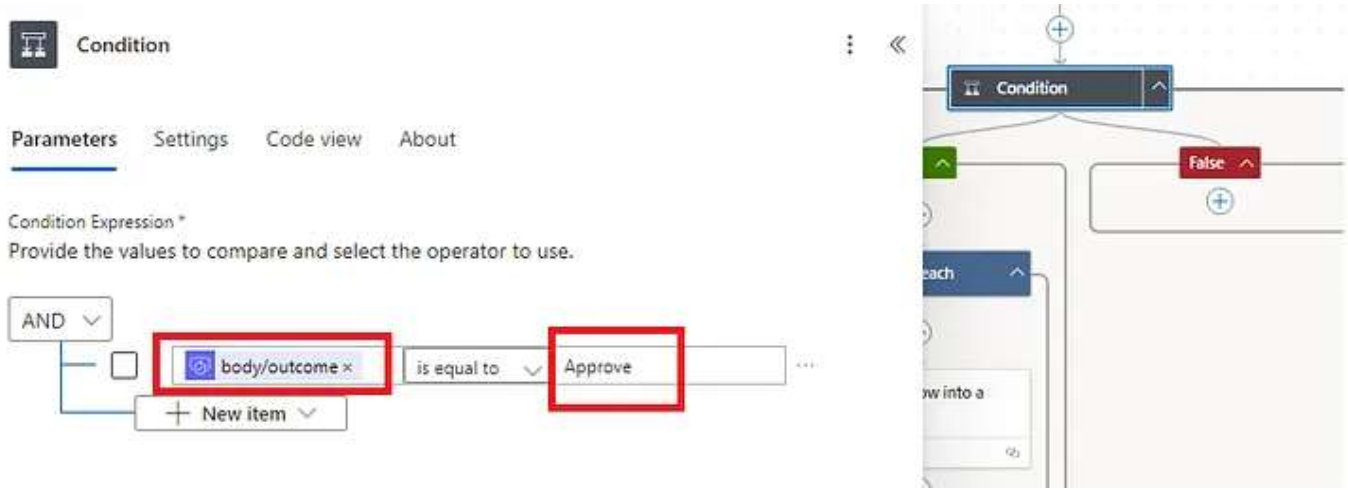
2. **Configure the Condition Expression:**
   In the **Condition** action, use the **"Outcome"** dynamic content from the previous approval step. This will hold the result of the approval process (either **Approve** or **Reject**).

- In the condition expression, select the **"Outcome"** field.
- Use the operator **"is equal to"** and provide the value **"Approve"** (as shown in the image you uploaded).

3. **Condition Branches**:

- **If the Outcome is "Approve"**:
- Proceed to update the **SharePoint user access table** and trigger the **Power BI model refresh** to reflect the new user access.
- You can also notify the requester that their access has been approved.

- **If the Outcome is “Reject”:**
- Notify the requester that their access request has been rejected, including any comments or reasons from the approver.
- No further updates will be made to the user access table or Power BI model.



Step 18: True Branching — Update User Access Table

1. **Add “Apply to Each” Loop:**
   In the **True** branch of the condition (when the request is approved), we will proceed with updating the **user access table** in SharePoint. To do this, create a new **Apply to Each** loop.

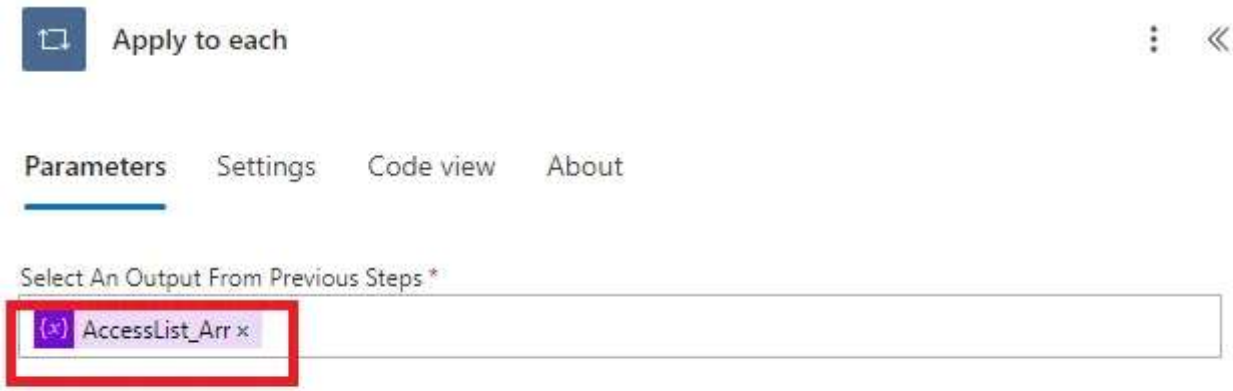
2. **Configure the Payload:**
   For the **Apply to Each** loop, use the **array variable** you initialized earlier. This variable contains the list of regions or countries the user has requested access to (from the form).

- Set the **payload** of the **Apply to Each** loop to the array variable, which will loop through each region or country in the list.

3. Add Action to Update SharePoint:
Inside the **Apply to Each** loop, use an action such as **"Create item"** or **"Update item"** (depending on your requirements) to add or update the user access details in the SharePoint table.

- The action should include the user's details (like name, designation, department, etc.) along with the region or country selected.
- Ensure that the **regions/countries** are stored correctly in SharePoint, along with the user's access permissions.

4. Final Outcome:
Once the loop runs, the SharePoint table will be updated with the user's selected access details (regions or countries). This ensures that the user's access is now correctly reflected in the system.



Step 19: Add the User to the User Access Table

1. **Ensure Data is in Table Format:**
   Before proceeding, verify that the user access table in the Excel file (or SharePoint) is in **table format.** This is crucial because the Power Automate flow needs to reference the **table name** for adding or updating data in the table.

- If your Excel file is not in table format, go ahead and select your range of data in Excel and convert it into a table (using **Insert > Table**). Ensure that the table has a proper name (for example, `UserAccessTable`).

2. Create "Add a Row into a Table" Action:

Once you confirm that the data is in an active table format, go ahead and

create an action in Power Automate called **"Add a row into a table"**. This action will add the details of the user's access request into the table.

3. Configure the Action:
For this action, you'll need to provide:

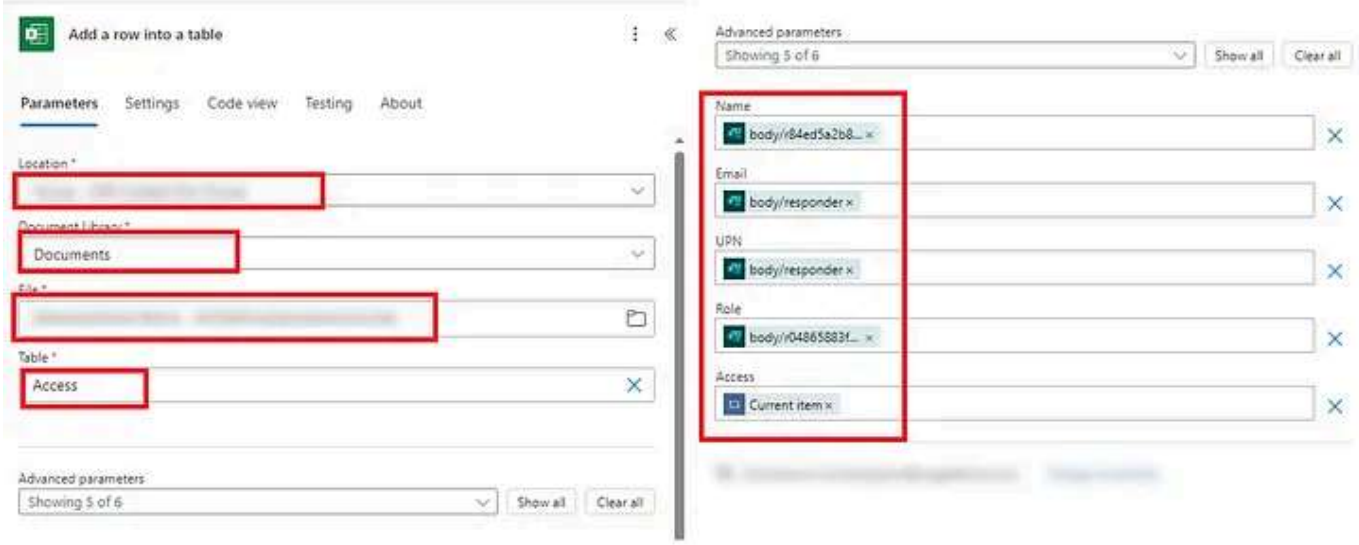
- **Location**: The location of the Excel file (whether it is stored in SharePoint, OneDrive, or any other source).
- **File**: Select the Excel file that contains the user access table.
- **Table**: Select the table within the Excel file that holds the user access information. You should see your active table name in the dropdown (e.g., `UserAccessTable`).

4. Fill in Table Details:
Once the table is selected, Power Automate will allow you to fill in the columns of the table with dynamic content from the form submission and any other required user details. For example:

- **User Name**
- **Designation**
- **Department**

- **Access Level (Region/Country)**
- Any other relevant details.

5. Complete the Row Addition:
Once configured, Power Automate will automatically add a new row to the Excel table with the details filled in, effectively updating the user access table with the user's request.



Step 20: Trigger Power BI Dataset Refresh

After successfully adding the user to the user access table in SharePoint or Excel, you can proceed with setting up a **Power BI dataset refresh** activity.

This refresh ensures that the data model in Power BI reflects the newly added user and any associated changes.

1. **Create Power BI Dataset Refresh Action:**
   Add the action **"Refresh a dataset"** under Power BI in your flow. This action will trigger an on-demand refresh of the dataset associated with the report.
2. **Configure the Action:**

- **Workspace**: Select the Power BI workspace where the dataset resides.
- **Dataset**: Select the specific dataset you want to refresh. This should be the dataset related to your report that contains user access data.

3. Outcome:
After the refresh is triggered, any changes made in the user access table will automatically be incorporated into the dataset, ensuring the report is up to date.

> **Note:** I'll create a separate article that covers the details of setting up a Power BI dataset refresh process, including best practices and how to handle dataset dependencies.

Step 21: Inform the Support Team to Update User Access in Power BI Service Security

Finally, you'll want to notify the support team that the new user has been added to the user access list and that their IDs are now part of the security roles in the Power BI workspace.

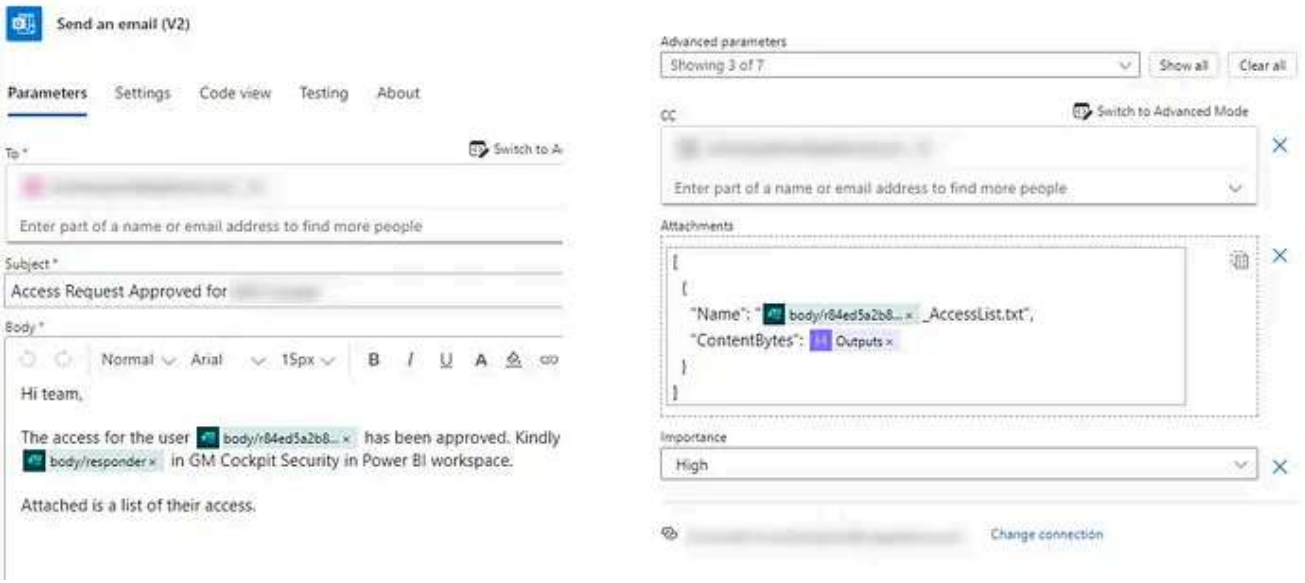
1. **Create "Send an Email" Action:**
   Add the **"Send an email (V2)"** action in your flow to notify the support team.
2. **Configure the Email:**

- **To:** Enter the support team's email address or use a group email address that forwards to the team.
- **Subject:** Set a subject such as: "New User Added to Power BI Report Security".
- **Body:** Include the relevant details about the new user's access, the regions or countries they have access to, and instructions for updating the security roles in the Power BI workspace.

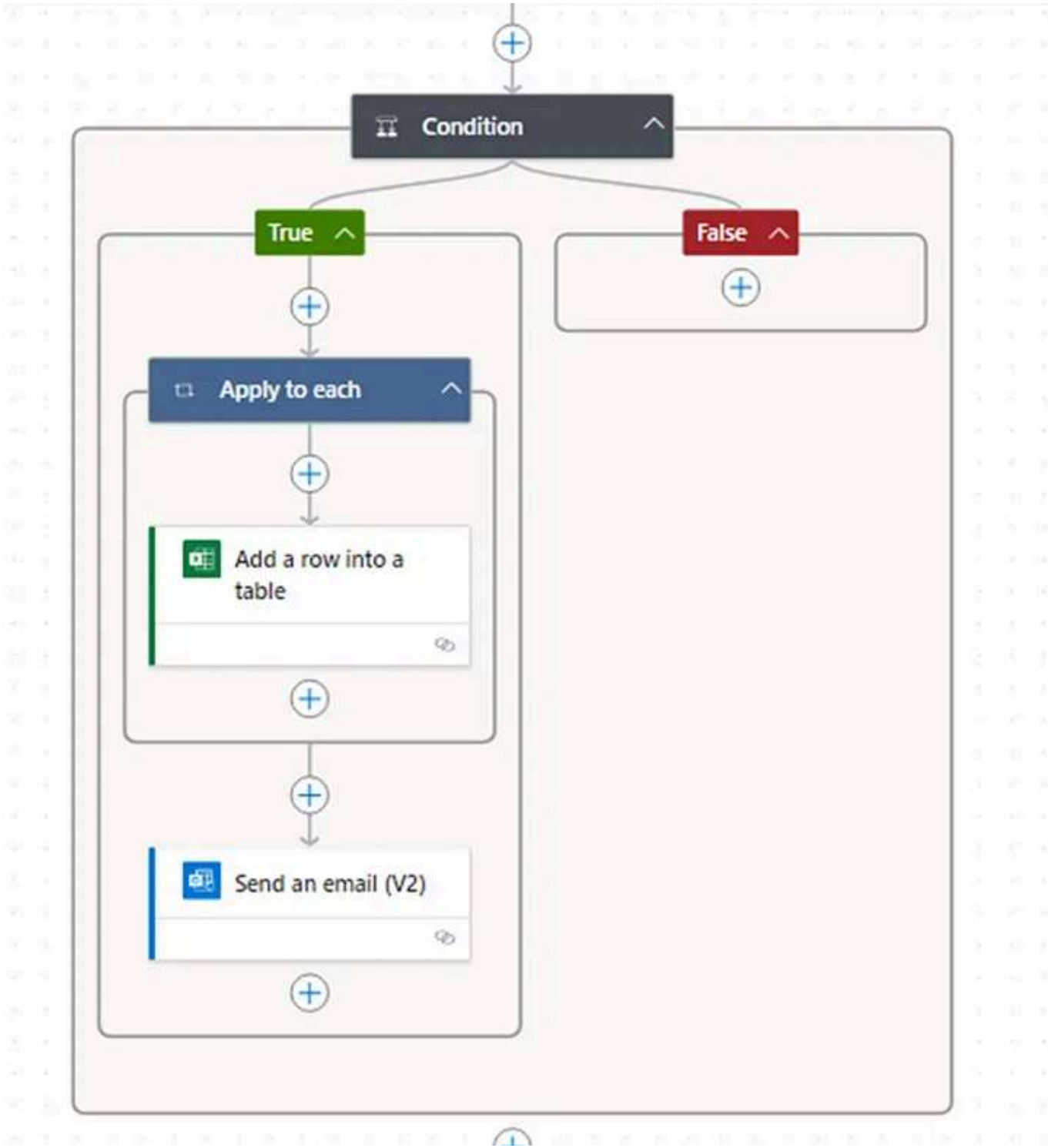
3. Dynamic Content:
You can pull dynamic content from the flow, like the user's name, access

details, and other relevant fields to automatically populate the email with the necessary information.

4. Outcome:
The support team will receive a detailed email informing them about the new user, prompting them to add the user's details into the Power BI security roles. This ensures that the user has the appropriate access in Power BI Service for viewing reports.



The final part of the flow will look like this:


Condition
True
False
Apply to each
Add a row into a table
Send an email (V2)

Note: the false section is not fleshed out. You can add a similar email activity to the requester with the denial details.

Streamlining user access isn't just about simplifying a process — it's about enhancing collaboration, maintaining data integrity, and ensuring timely decision-making. By combining intuitive tools like Microsoft Forms with the automation power of Power Automate, we can shift from a reactive approach to a proactive, user-centric system. This not only reduces friction for both users and administrators but also sets the stage for scalable, efficient report management in an increasingly dynamic business environment.

Thank you for reading!

Please feel free to give 50 claps 👏 if you found this article helpful and leave a comment or share it with others.

👉 Follow me or ✉️ subscribe to get all my Power BI articles!

You can find me on LinkedIn.

Don't forget to subscribe to

👉 Power BI Publication

👉 Power BI Newsletter

and join our Power BI community:

Microsoft Power BI Masterclass | Twitter, Instagram | Linktree

Let’s share our Microsoft Power BI experience. Learn together. Grow together.

linktr.ee

Automation | Powerplatform | Power Bi | Power Bi Tutorials | Power Automate

Published in Microsoft Power BI

8.4K Followers · Last published 16 hours ago

Following

Microsoft Power BI community sharing our Power BI experience, tutorials, use cases, tips and tricks. Learn together. Grow together. Follow our Power BI Masterclass: https://linktr.ee/powerbi.masterclass or me: https://linktr.ee/tomas.kutac

Written by Avishek Ghosh (AV_DEVS)

99 Followers · 35 Following

Follow

Analytics expert and "interestingness hunter-gatherer" passionate about dissecting ideas, connecting dots, and exploring the bigger picture.

No responses yet

What are your thoughts?

Respond

More from Avishek Ghosh (AV_DEVS) and Microsoft Power BI

In Microsoft Power ... by Avishek Ghosh (AV_DEV...

In Microsoft Power BI by Shashanka Shekhar

A Step by Step Guide to Implement Version Controlling in Power BI

From Saving PBIX in SharePoint, to Using PBIP, VS Code, Git and Power BI's...

Oct 8, 2024 113 1

Creating an Yearly Comparative KPI with Bar Charts in Power BI

In today's data-driven world, businesses rely heavily on Key Performance Indicators (KPIs...

Dec 23, 2024 35

In Microsoft Power BI by Tomas Kutac

The Best Power BI Books in 2024

Top Power BI Books to Read in 2024

Dec 3, 2024 185 1

In Microsoft Power ... by Avishek Ghosh (AV_DEV...

From Prediction to Presentation: Sales Forecasting with Meta's...

Unleash Predictive Analytics with Python and Transform Insights into Interactive Power BI...

Dec 17, 2024 11

See all from Avishek Ghosh (AV_DEVS)

See all from Microsoft Power BI

Recommended from Medium

Jon Vöge

Do you ever ask yourself: How can I display Power BI reports to peopl...

The questions is complex, and has multiple different solutions. One is to use Power BI...

Jul 16, 2024 3

DataScience Nexus

Top Power BI Dashboard Examples

Power BI is one of the most popular business intelligence tools used worldwide for data...

Dec 31, 2024 12

Lists

Coding & Development

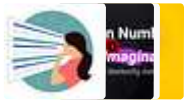
11 stories · 971 saves

The New Chatbots: ChatGPT, Bard, and Beyond

12 stories · 538 saves

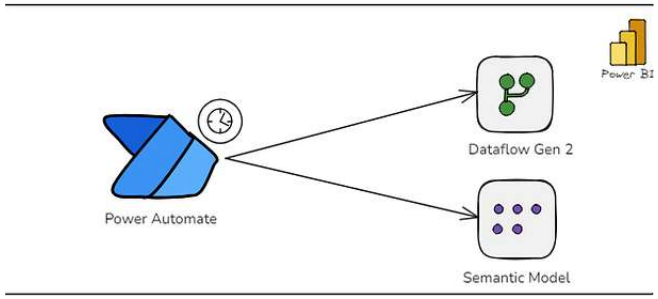
ChatGPT

21 stories · 940 saves



MODERN MARKETING

209 stories · 981 saves

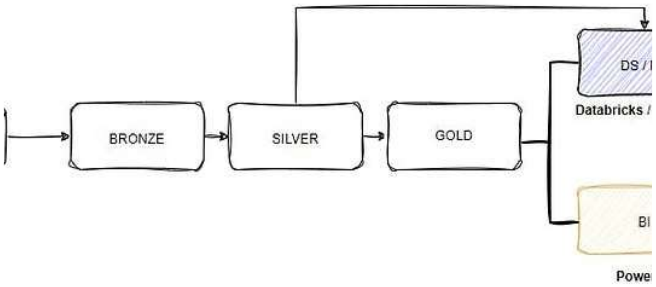


In Microsoft Power ... by Avishek Ghosh (AV_DEV...

How to Schedule Power BI Report Refreshes on Non-Fixed Interval...

Learn how to automate your Power BI report refreshes when your schedule requires...

Jan 6 8



Hanson Olatunde

Fabric Semantic Link and Use Cases

Fabric Semantic Link has finally been announced and is available for everyone to...

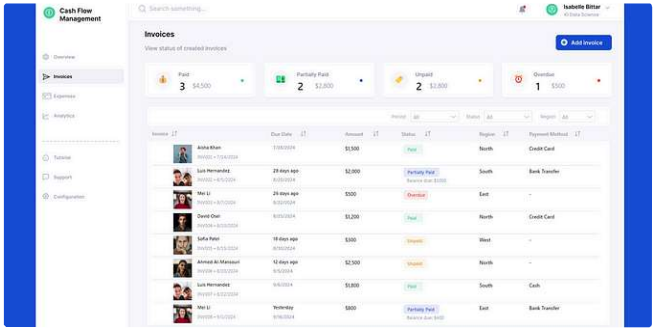
6d ago 2 1

In Python in Plain English by Kiran Maan

50 Python Shortcuts Every Python Beginner Should Know

This mind-blowing cheatsheet will change the way you code.

Jan 6 540 10



In The BI Corner by Isabelle Bittar

Transforming Power BI Tables: 6 Expert Tips for Smarter Data...

Boost User Experience with These Power BI Table Enhancements

Sep 19, 2024 194 1

See more recommendations