

Steps to recreate the Scenario part with Adaptive Cards and Principal Propagation:

Information returned via chatbots should not be restricted to looking only as good as the UI of the platform you deploy the bot on, it would more interactive and personal to your brand to have a way to choose the design in which users get to interact with the bot you created. This is where adaptive cards are a game changer. Adaptive cards allow you to create responses from the bot that are more visually appealing, interactive, and customizable and make a fantastic addition to your flow on Copilot Studio.

To explore how it could be used as shown in the demo video, go back to the Copilot Studio topic – ‘Sales order statuses, and we will swap the text responses at 2 places for adaptive cards. (Or you could just read about how to create an adaptive card and use it elsewhere).

Here are the two places we will add the adaptive cards.

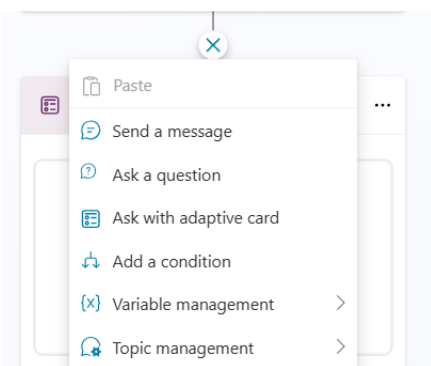
- 1) Getting the sales order number to deep dive into from the customer.
- 2) Displaying information about the order from the SAP system.

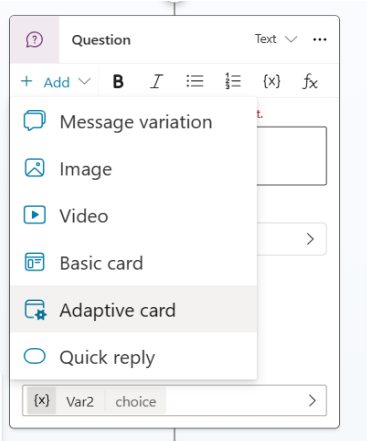
To start off, navigate to the node in which we ask the user for the sales order number they want to deep dive into.

Instead of asking a question, we can replace it with an adaptive card.

You can add an adaptive card like this:

1. Add a question node
2. Click on add adaptive card.
3. Double click on the adaptive card to view/ modify the code.





Here is the adaptive card we will use to get the sales order number from the user. (The code for this is available on GitHub) and you can copy paste It in the “edit JSON” text box shown below.

New condition

Adaptive Card

Enter the sales order number:

Enter Sales Order Number

Submit

Outputs (3)

{x} action string =

{x} action submitId string =

{x} salesOrderNumber string =

{x} salesOrderNumber string

properties

[Open Adaptive Card designer](#)

</> Edit JSON

```
"$schema": "http://adaptivecards.io",
"type": "AdaptiveCard",
"version": "1.2",
"body": [
  {
    "type": "TextBlock",
    "text": "Enter the sales or",
    "weight": "Bolder",
    "size": "Medium"
  },
  {
    "type": "Input.Text",
    "id": "salesOrderNumber",
    "placeholder": "Enter Sales Ord",
    "errorMessage": "Sales Order Num"
  }
]
```

Edit schema

☒ Allow switching to another topic ⓘ

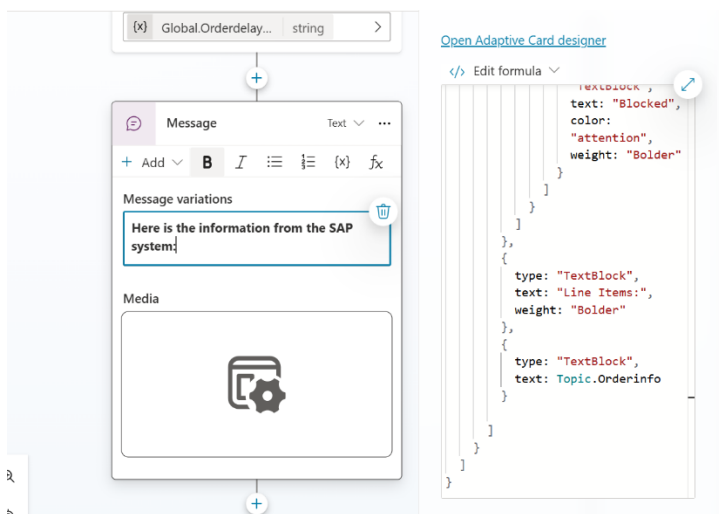
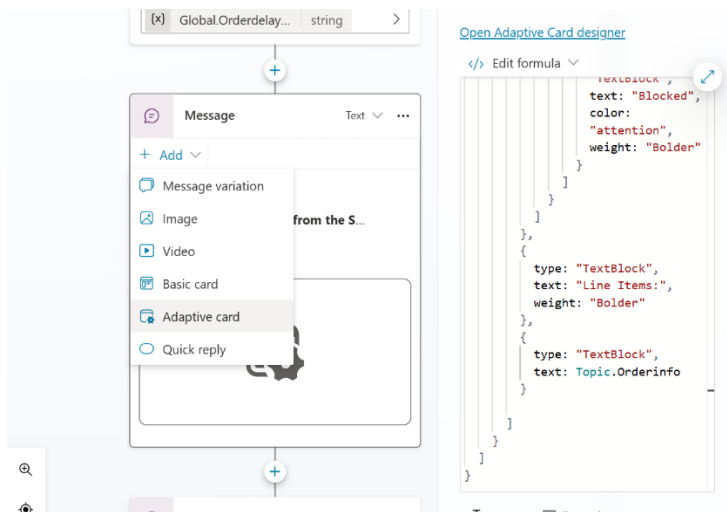
Only selected topics ⓘ

Retry prompt ⓘ


☐ Customize

Now to display the information about the order from the SAP system, we will once again make use of an adaptive card to display the information in a more visually appealing way.

Navigate to the part in the Sales order flow after the Power automate flow “context delayed” is called and add an adaptive card to the message node. Click on add, and add an adaptive card to the message node.



This is how we want the adaptive card to look while in the flow. You can copy the code of this adaptive card from GitHub from the file named “SalesorderviewAdaptivecard.json”.



Sales Order Data

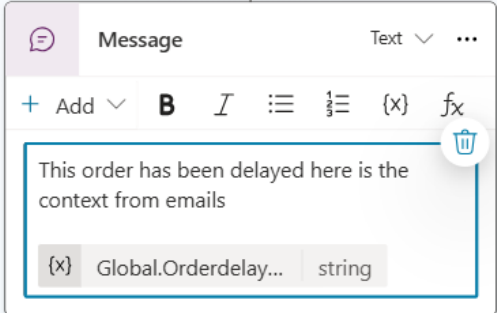
Sales Order N...	Delivery ...	Delivery Block...
0000299756	A (Active)	Blocked

Line Items:

10	MZ-FG-550	M550 Bike	20 PC	\$77,...
Req Delivery date: 20-06-2024				
20	MZ-FG-R200	R200 Bike	20 PC	\$79,...
Req Delivery date: 20-06-2024				

Note: For this adaptive card click on 'Edit JSON' and pick 'Edit Formula' instead as we will be entering dynamic values from the flow which are considered formulas.

Next, create a new message and add the variable with the MS365 context there.



Message

Text

+ Add **B** *I* ☰ ☷ {x} fx

This order has been delayed here is the context from emails

{x} Global.Orderdelay... string

Now referring to the video above, we will create the topic that helps find a replacement item for a material.

This topic will be called “Find similar items in Sales order” and the description will be as shown below.

The screenshot displays the SAP Order Status Copilot interface. The top navigation bar includes 'Overview', 'Knowledge', 'Topics', 'Actions', 'Analytics', and 'Channels'. The 'Topics' tab is active, showing a list of topics. The selected topic is 'Find Similar Items in Sales Order'. Below the topic name, there are icons for 'Copilot', 'Comments', 'Variables', 'Topic checker', 'Details', and 'More'. A 'Save' button is visible on the right. The main content area shows the topic configuration. The 'Trigger' section is expanded, showing a trigger 'Triggered by copilot (preview)' with an 'Edit' link. Below this, there is a section 'Describe what the topic does' with a text box containing the following description: 'Description: This topic enables the copilot to identify and retrieve items from an order that closely match the products in a given sales order. It uses natural language understanding and trigger phrases related to order items and substitution of order items.. When a user mentions relevant keywords, the topic gets triggered, and the copilot can respond with information about similar items.' Below the description, there is a plus icon and an arrow pointing down to an 'Adaptive Card' section.

Next, we will create an adaptive card to ask the user what sales order they want to find the item for. The code for this adaptive card will be on GitHub under the name 'SalesordersubstituteAdaptivecard.json'.

Adaptive Card

Enter the sales order number you need substitutes for:

Enter sales order number

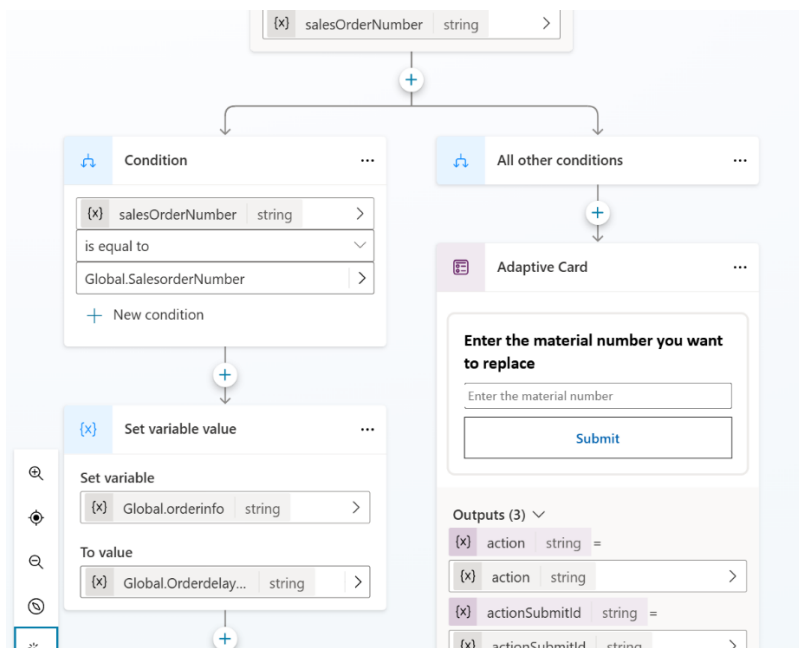
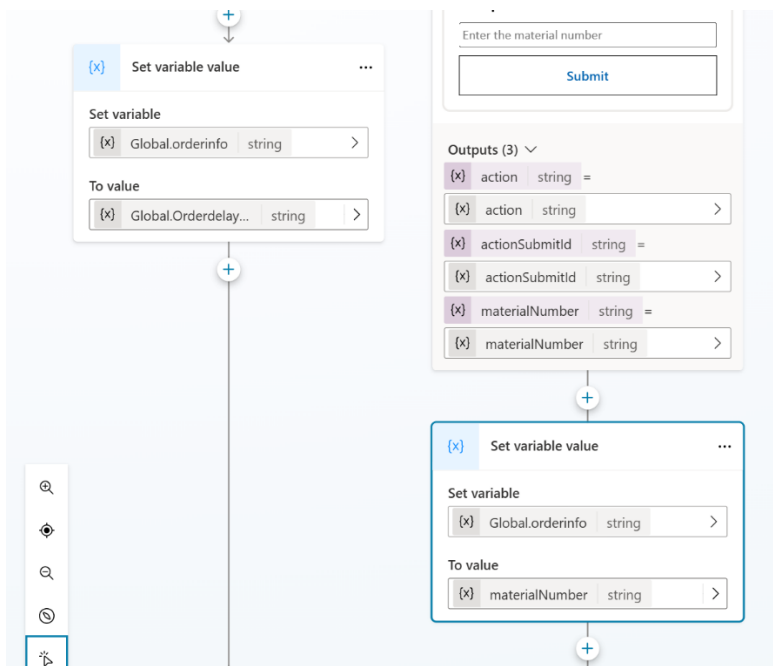
Submit

Outputs (3) ▾

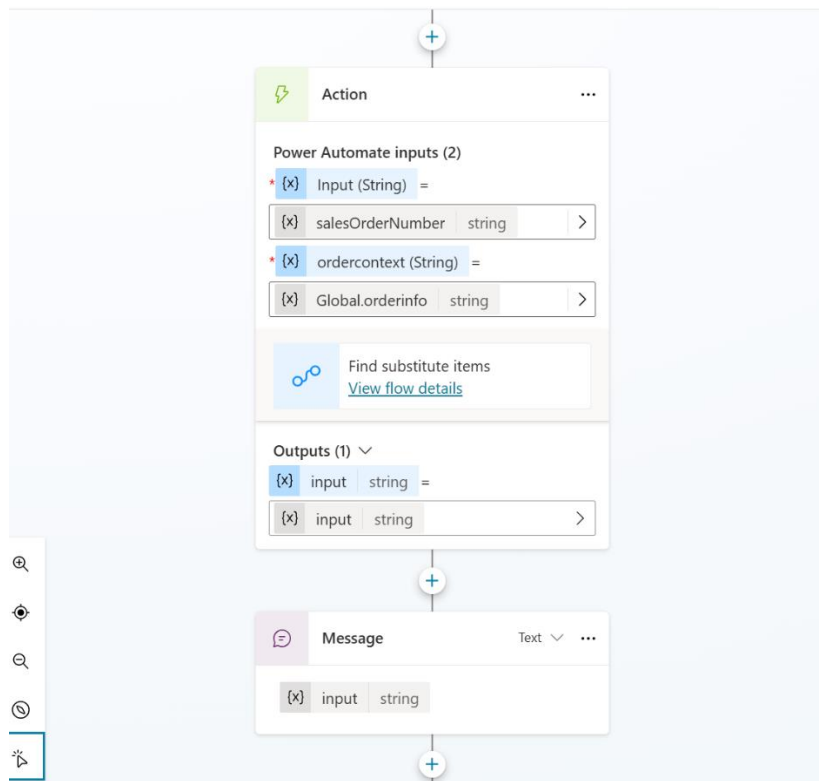
- {x} action string =
- {x} actionSubmitId string =
- {x} salesOrderNumber string =

Next, if the order number matches the one from the previous topic (variable `Global.SalesOrdernumber` from the previous topic 'Sales order status') it means the conversation is a continuation and we will continue to pass information from the MS365 context variable about which material is unavailable, otherwise we will prompt the user to enter the material in the order they want to find a substitute for using a similar adaptive card as above.

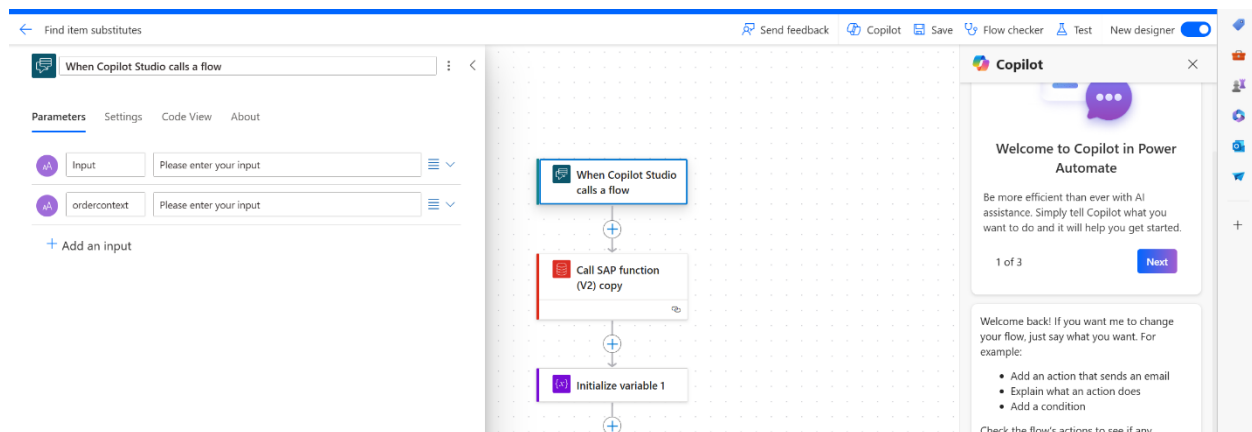
We will create a new order variable called `Global.orderinfo`, and in case the sales order doesn't change, we will populate that variable with the Microsoft MS365 data stored in the variable 'Global.Orderdelaycontext' from the previous topic 'Sales order status', if it is not, we will populate that variable with the material number the user enters as shown below.



Now, we will call a Power Automate flow with two variables as input, the sales order number and information about what the missing material is.



This Power Automate flow is called “Find item substitutes”.



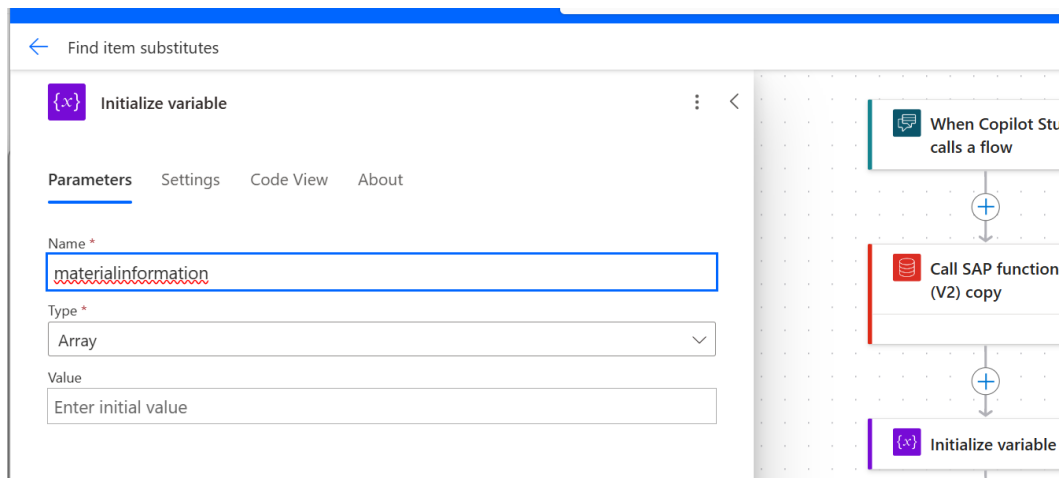
The first step to find the substitute items for an order is to get a list of all materials from the SAP system, for this we will call the BAPI, BAPI_MATERIAL_GETLIST.

The screenshot shows the SAP Studio interface. On the left, the 'Parameters' tab for 'Call SAP function (V2) copy' is active. It displays the 'SAP System' parameter with a JSON object: `{"AppServerHost":"10.15.0.6","Client":"400","SystemNumber":"01","LogonType":"ApplicationServer"}`. The 'RFC Name' is set to 'BAPI_MATERIAL_GETLIST'. Under 'Advanced parameters', it shows 'Showing 8 of 13' with 'Show all' and 'Clear all' buttons. The 'RFC Group Filter' is empty. On the right, a flow diagram shows a sequence: 'When Copilot Studio calls a flow' followed by 'Call SAP function (V2) copy'.

Next, we will initialize an array variable called “matrlist” of type array to store this information.

The screenshot shows the SAP Studio interface. On the left, the 'Parameters' tab for 'Initialize variable 1' is active. The 'Name' is 'matrlist', the 'Type' is 'Array', and the 'Value' is 'MATNRLI...'. On the right, a flow diagram shows a sequence: 'When Copilot Studio calls a flow' followed by 'Call SAP function (V2) copy', and then 'Initialize variable 1'.

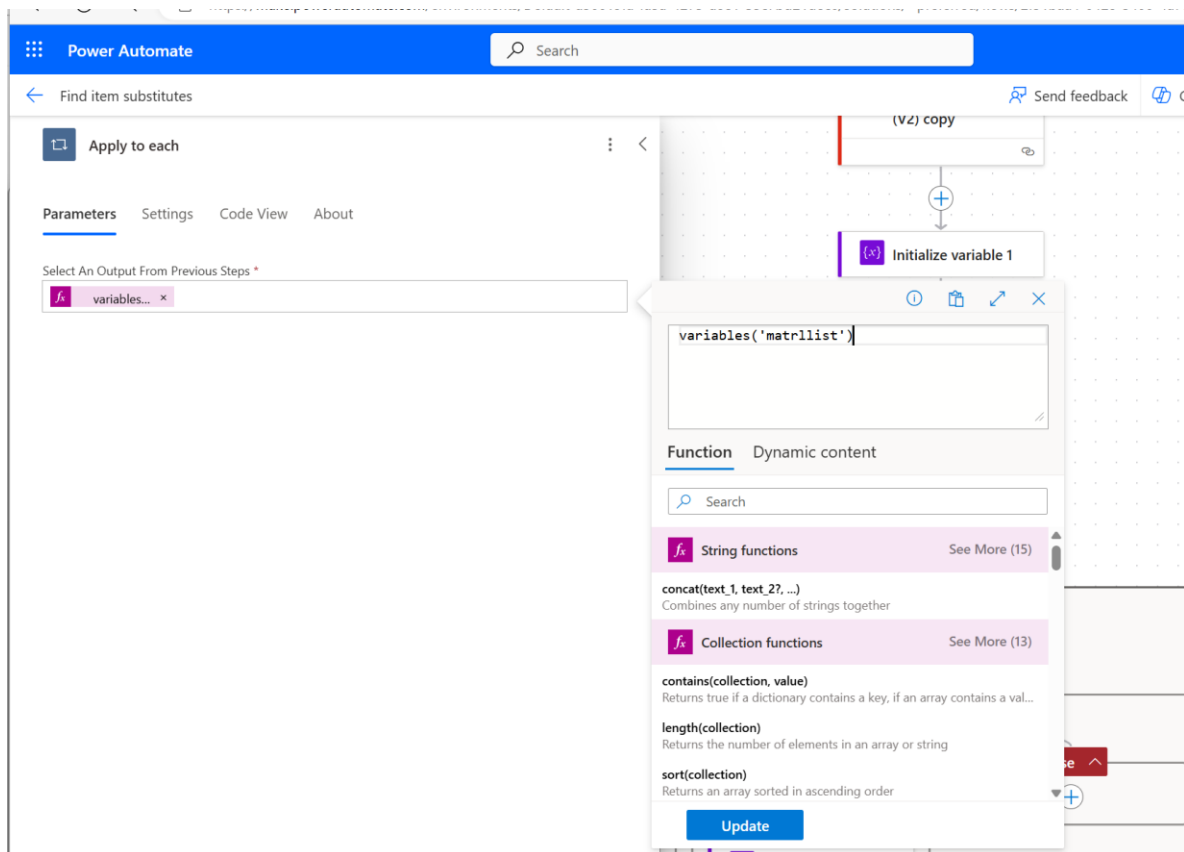
We will initialize another variable called “materialinformation” of type array that will store the material information for each material in the list.



Now, we will create a 'for each loop' to loop over the list materials we received from the BAPI which is the variable "matrllist".

For each material in the loop we will call another BAPI, "BAPI_MATERIAL_GETDETAIL" which will return the data and information on the material.

We will append this information for each material to the array "materialinformation" as shown here:



Find item substitutes

Send feedback

Apply to each 1

Parameters

Settings

Code View

About

Select An Output From Previous Steps *

matrlist

Apply to each 1

Call SAP function (V2) copy 1

Append to array variable 1

Find substitute items

Call SAP function (V2) copy 1

Parameters

Settings

Code View

Testing

About

SAP System *

{ "AppServerHost": "10.15.0.6", "Client": "400", "SystemNumber": "01", "LogonType": "ApplicationServer" }

RFC Name *

BAPI_MATERIAL_GET_DETAIL

Advanced parameters

Showing 3 of 11

Show all

Clear all

RFC Group Filter

*

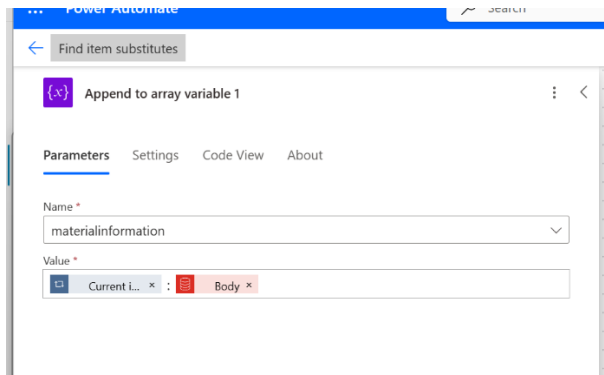
Auto Commit

Yes

RfcInputs/MATERIAL

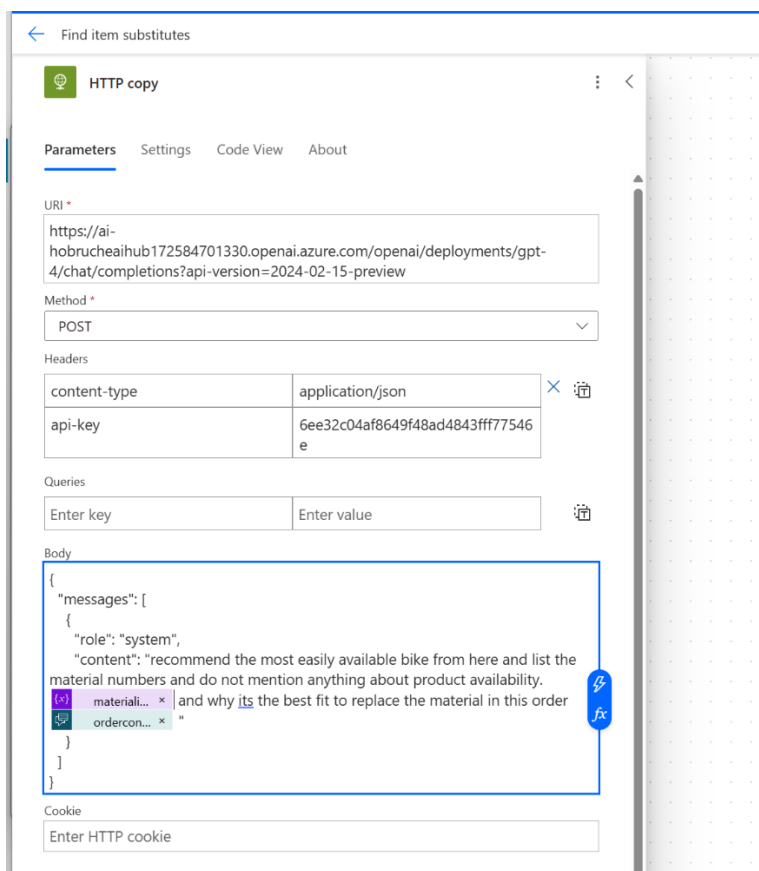
Current i...

Connected to SAP ERP. [Change connection reference](#)

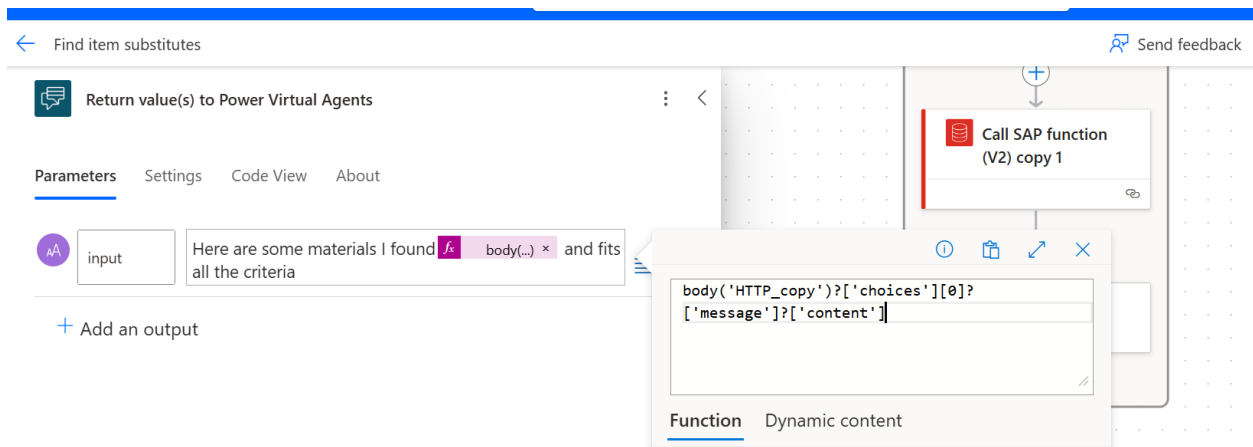


Next, we will call an HTTP function that calls Azure OpenAI and pass to it the order delay context (from the previous topic which was an input to this Power Automate flow) as well as the “materialinformation” variable, therefore giving it the material we need to find a replacement for as well as the list of materials and their details to find the replacement from.

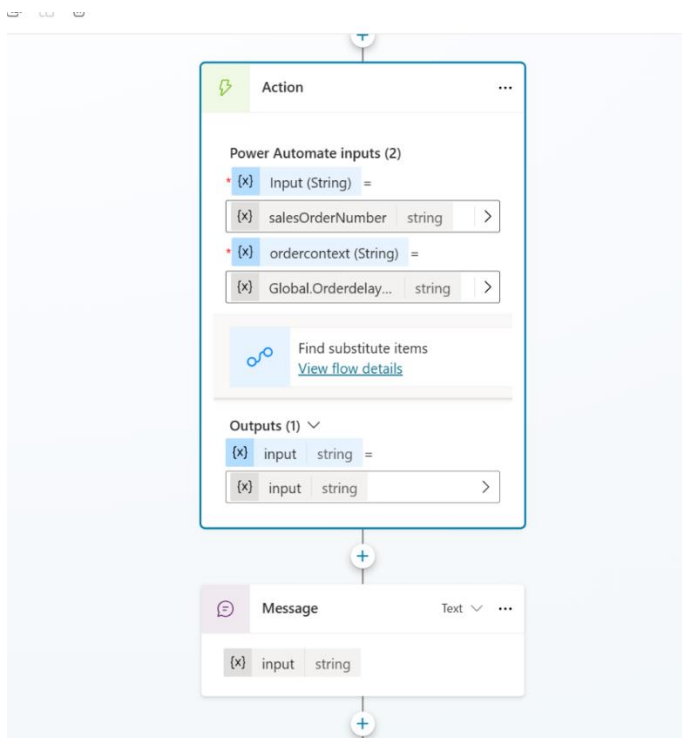
Here is the prompt and the HTTP call:



Finally, we will send the recommended material along with the reasoning for picking the substitute that Azure OpenAI provided, back to Copilot Studio.



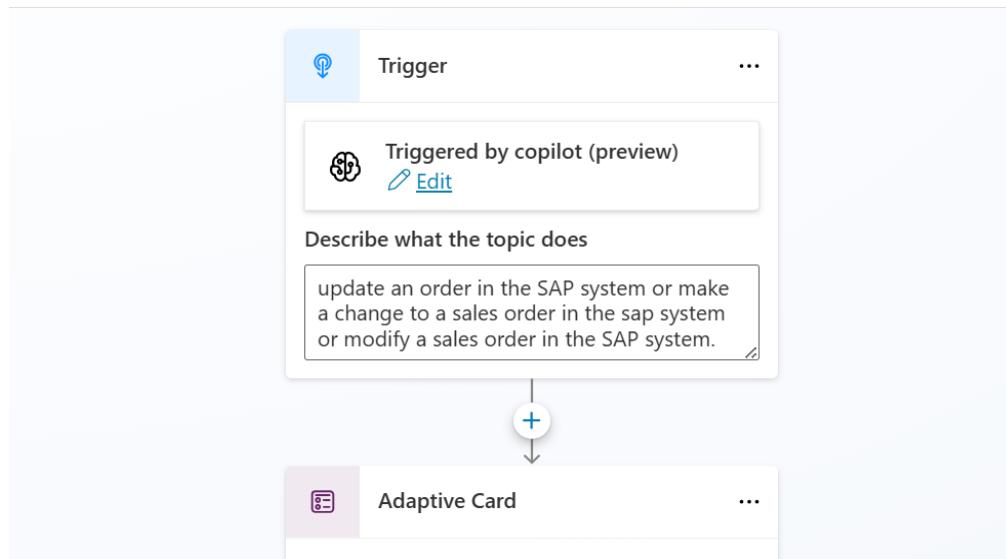
Back in the Copilot Studio we will return the output from the Power Automate flow back to the user as a recommendation.



Once the user knows that the material is in stock, the user wants to update the sales order with the new material:

For this we will create a new Copilot Studio topic called “Modify sales order”.


Here is the description for it:



Next, we will create an adaptive card that asks the user to enter all the required information to change the order as shown below, the code for this adaptive card is on GitHub under the name (SalesorderchangeAdaptivecard.json”).

Adaptive Card

...



Replace Item in sales order

Order Number *

Please provide the following details to change the order for item you want to remove:

Material Number *

Please provide the following details to change the order for item you want to add:

Material Number *

Quantity *

Submit

Material Number *

Quantity *

Submit

Outputs (6) ▾

[x] action string =

[x] action string >

[x] actionSubmitId string =

[x] actionSubmitId string >

[x] addMaterialNumb... string =

[x] addMaterialNumb... string >

[x] orderNumber string =

[x] orderNumber string >

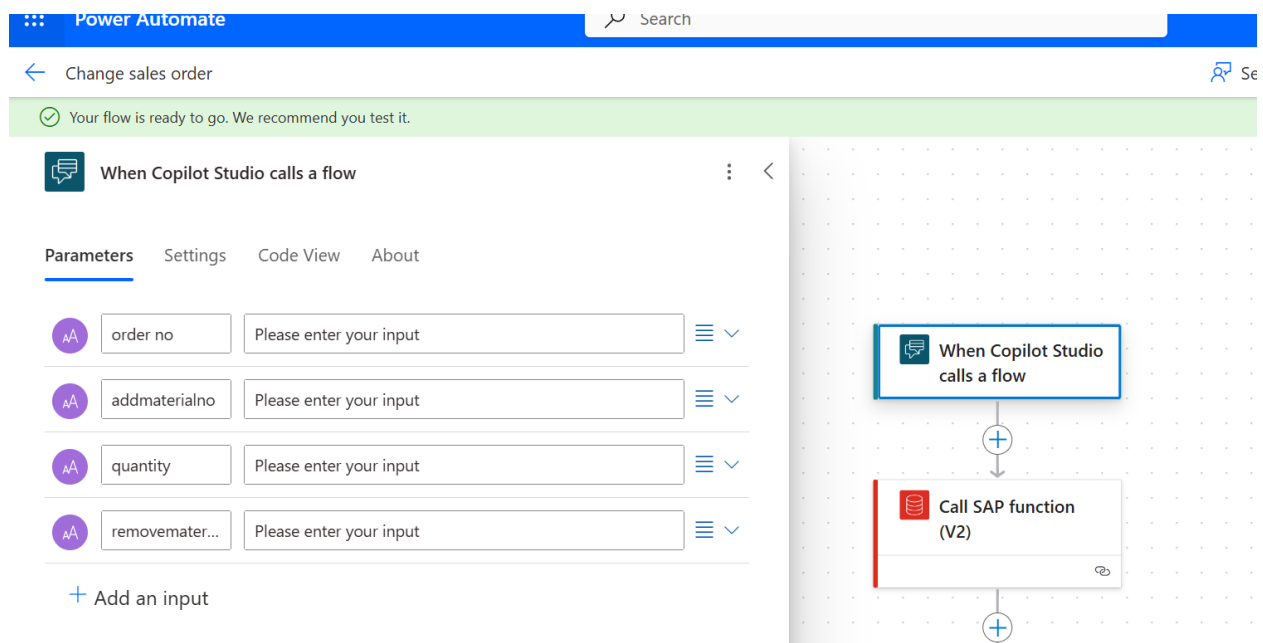
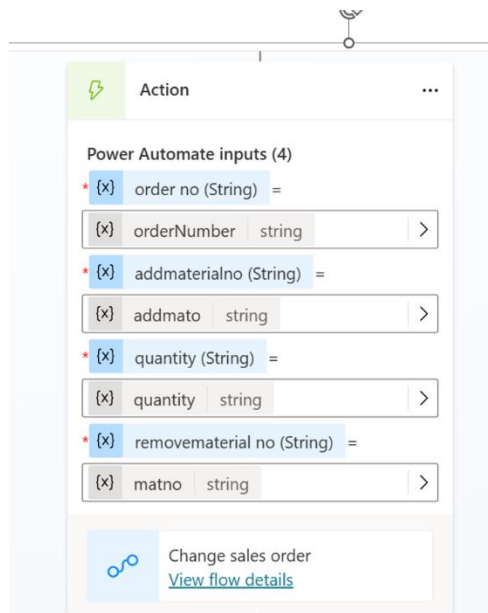
[x] quantity string =

[x] quantity string >

[x] removeMaterialN... string =

[x] removeMaterialN... string >

Now we will call a Power Automate flow to modify the salesorder using a BAPI with the inputs shown:



Next, we call the SAP BAPI : BAPI_SALESORDER_CHANGE. (The inputs / parameters to call this BAPI are complicated and therefore you can view them on GitHub and modify them according to your needs).

Call SAP function (V2)

Parameters

Settings

Code View

Testing

About

SAP System *

{ "AppServerHost": "10.15.0.6", "Client": "400", "SystemNumber": "01", "LogonType": "ApplicationServer" }

RFC Name *

BAPI_SALESORDER_CHANGE

ORDER_HEADER_INX *

{ "SALESDOCUMENT": "order no ×", "DLV_BLOCK": "X", "ITM_NUMBER": "000010", "MATERIAL": "X", "BILL_BLOCK": "X", "UPDATEFLAG": "U" }

RfcInputs/SALESDOCUMENT *

order no ×

Advanced parameters

Showing 7 of 31

Show all

Clear all

When Copilot Studio calls a flow

+

Call SAP function (V2)


+

Return value(s) to Power Virtual Agents


+


To ensure that the items have indeed been changed in the SAP system, we have added the return message from the SAP system to the Copilot as the response.

← Change sales order

 Return value(s) to Power Virtual Agents

Parameters Settings Code View About

 response

The response returned from the SAP system was :
 RETURN ×

+ Add an output

Finally, back in the Copilot Studio, we update the user and let them know whether their sales order has been updated or not.

Outputs (1) ▾
{x} response string =
{x} Var1 string >

+

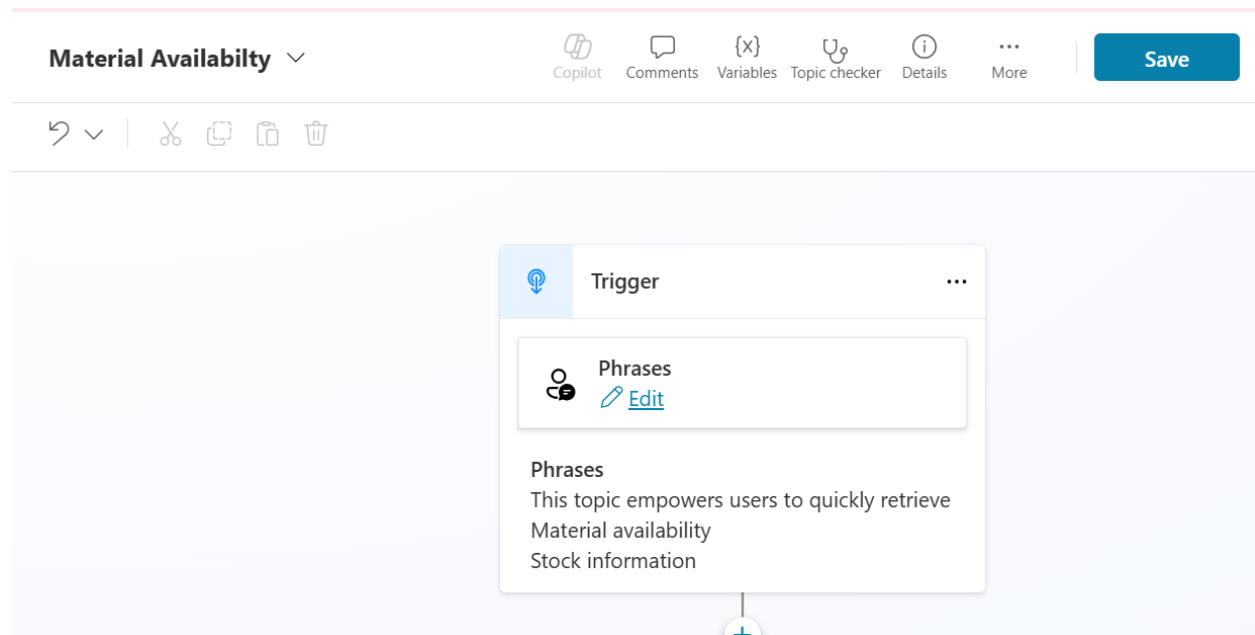
Message Text ▾ ...

+ Add ▾ **B** *I* ≡ ≡ {x} fx

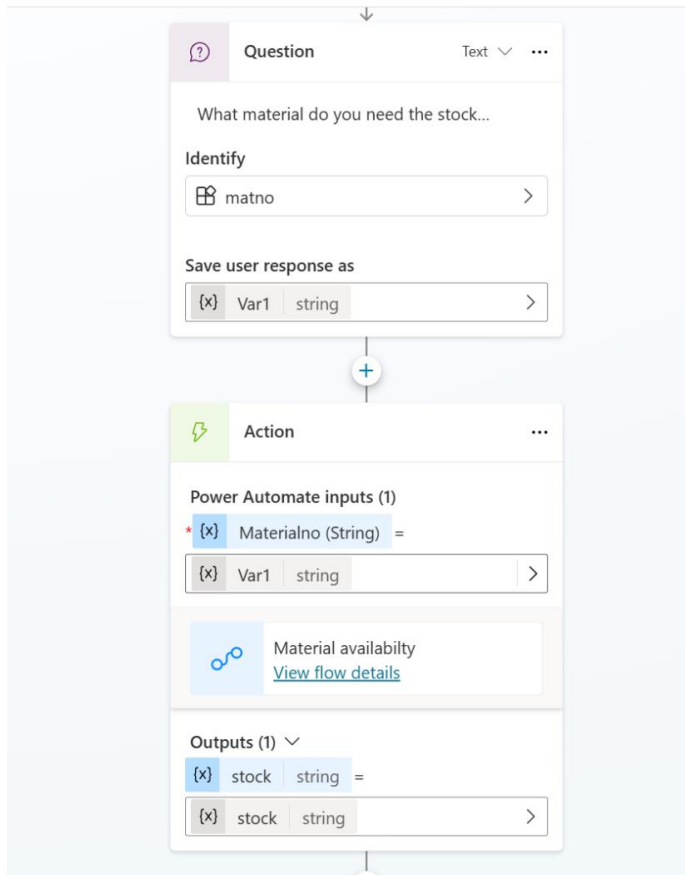
Sales order has been modified with this response:
{x} Var1 string
Cross check with the SAP system.

Once a substitute item has been selected, the user now wants to check the material availability. To check the stock availability of an item, the user must have the right access in the SAP system to access material stock information, therefore, we will implement 'Principal Propagation' which is ensuring that the M365 user has the right access in the SAP system to view this information without them having to use their SAP credentials.

First Create a Copilot studio topic with description as shown below:



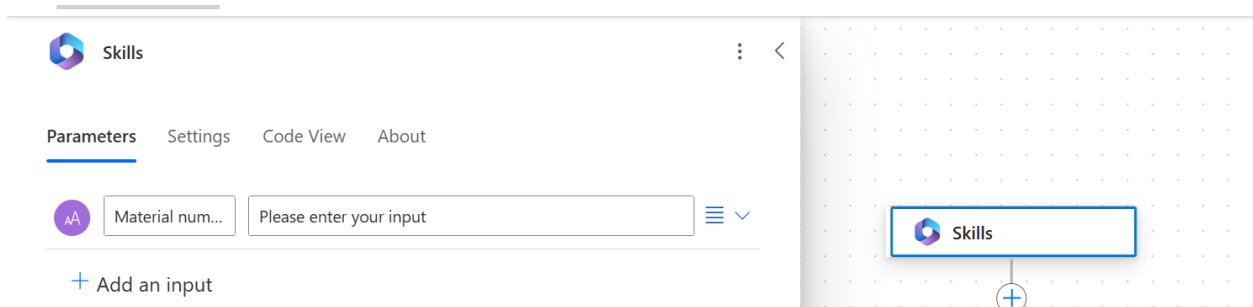
Now we will ask the user to enter the material number they need the information for, for this we will once again make use of entities like we did in the previous steps and as explained [here](#).



Now we will send this material number to a Power Automate flow which is called “Material Availability”

Here are the steps of the Power Automate flow:

First, the input is the material number the user needs availability for from the Copilot Studio.

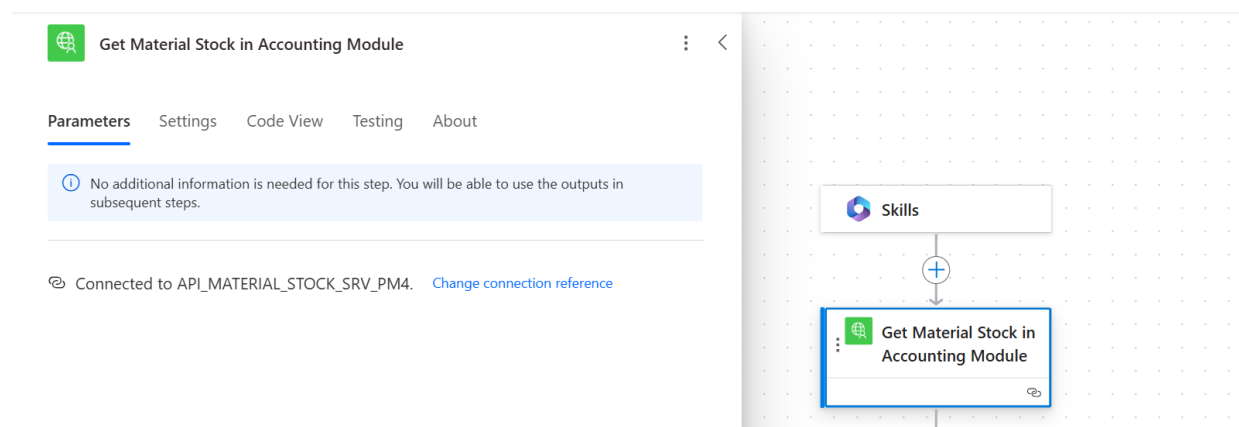


Now we will enter we will add an action and add the custom connector that would look like the image below:

The main addition to the flow here is the custom connector you will create for the 'Material Availability' OData service that has Principal Propagation implemented for authorization. To do that, you would have to make changes to your SAP system, set up an API on Azure using APIM (API Management) with policies that make this possible, and finally call that API by creating a customer connector on Power Platform. This is a process and there are a few resources that discuss Principal Propagation, and the steps associated with it in much more detail you can refer to achieve this:

[Principal propagation in a multi-cloud solution between Microsoft Azure and SAP, Part I: Building th...](#)

[Configure SAP Principal Propagation with AAD and SAP OAuth server](#)



After we get the response from the Custom Connector using OData calls to the SAP system, the response will be a complex JSON that we would need to parse using the action 'Parse JSON'. You need to upload a sample JSON to generate the schema of the response, to do that you can refer to the JSON in the exported flow on Power Automate, uploaded on GitHub under the name "Materialavailability.zip".

Note: The custom connector cannot be exported therefore the Power Automate flow on GitHub has been created without it, you could your custom connector and then make use of the rest of the flow.

Parse JSON

ParametersSettingsCode ViewAbout

Content *

Body x

Schema *

```
{
  "type": "object",
  "properties": {
    "statusCode": {
      "type": "integer"
    },
    "headers": {
      "type": "object",
      "properties": {
        "Cache-Control": {
          "type": "string"
        },
        "Transfer-Encoding": {
          "type": "string"
        },
        "dataserviceversion": {
          "type": "string"
        },
        "sap-metadata-last-modified": {
          "type": "string"
        }
      }
    }
  }
}
```

Use sample payload to generate schema

Skills

+

Get Material Stock in Accounting Module

+

Parse JSON

+

Initialize variable

+

Next, we will initialize a variable that stores stock information.

Material availability

Initialize variable

ParametersSettingsCode ViewAbout

Name *
stock info

Type *
String

Value
Enter initial value

Skills

+

Get Material Stock in Accounting Module

+

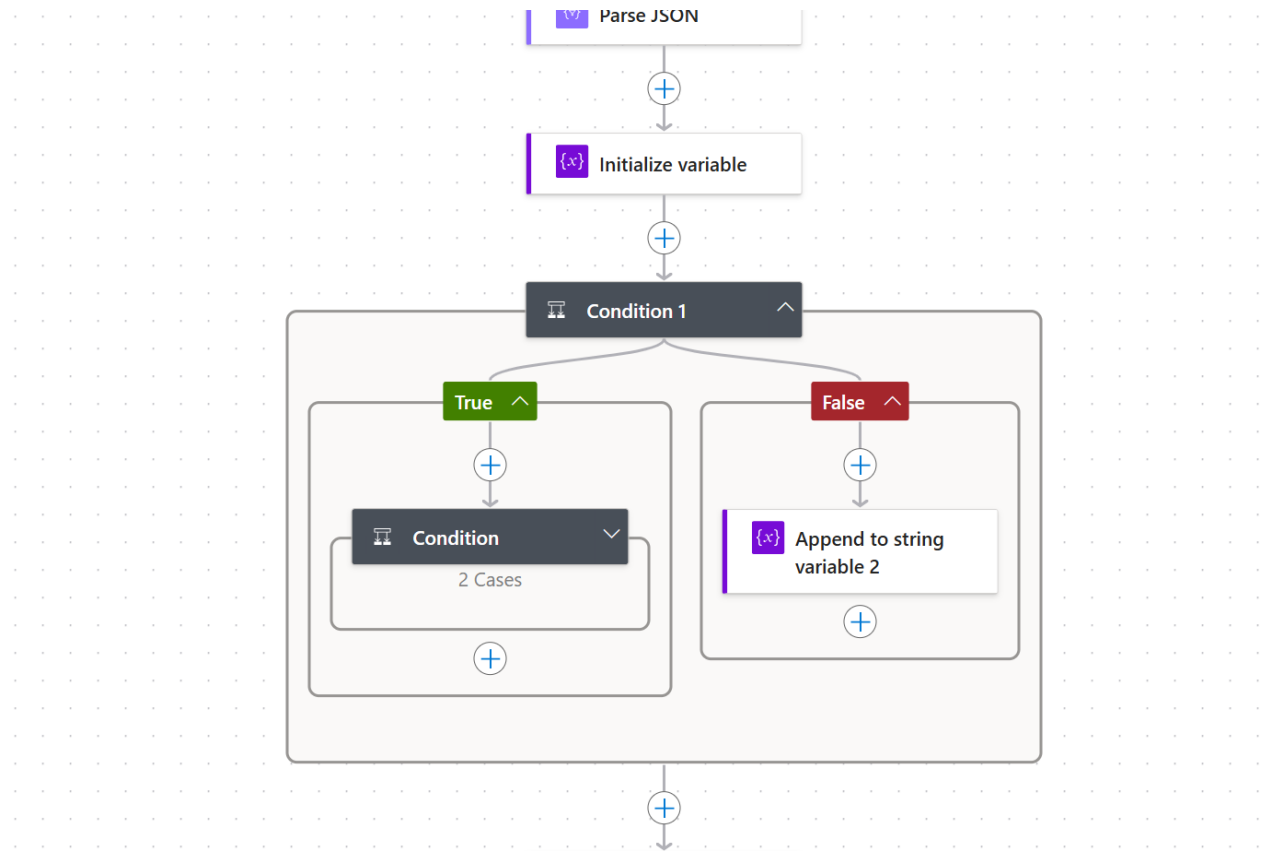
Parse JSON

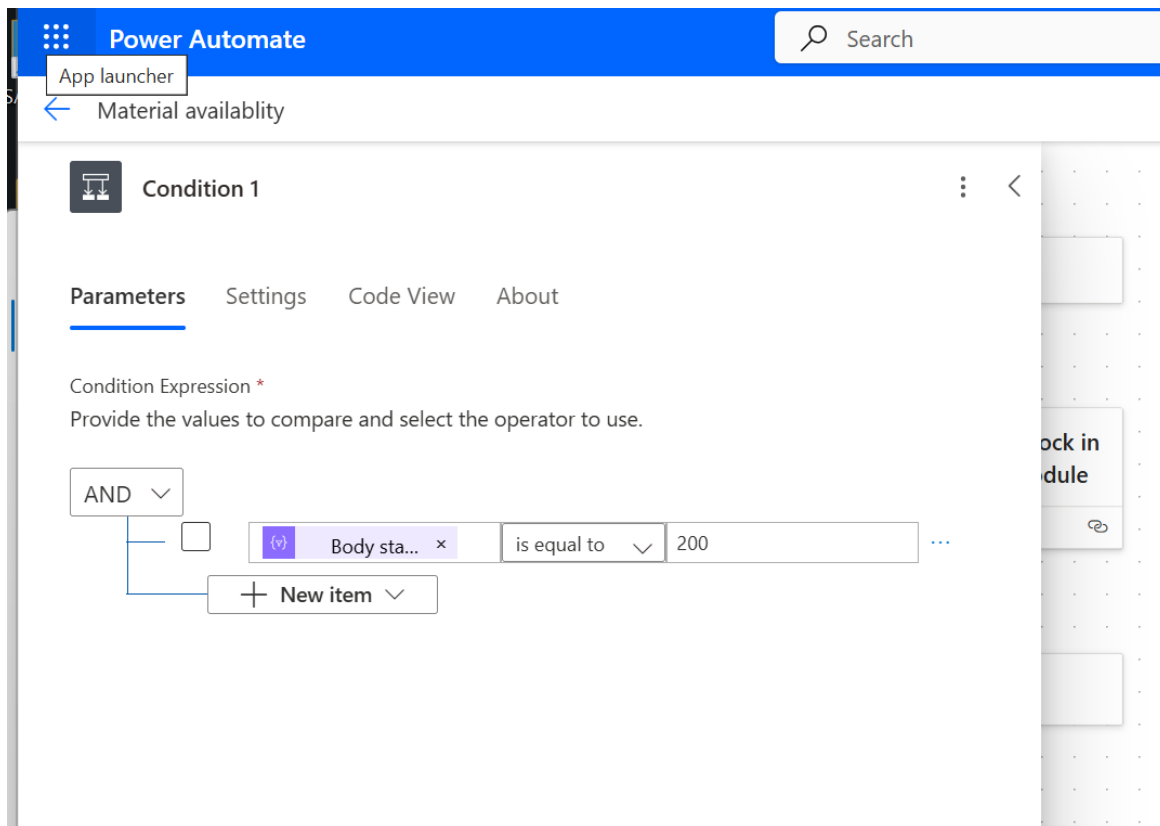
+

Initialize variable

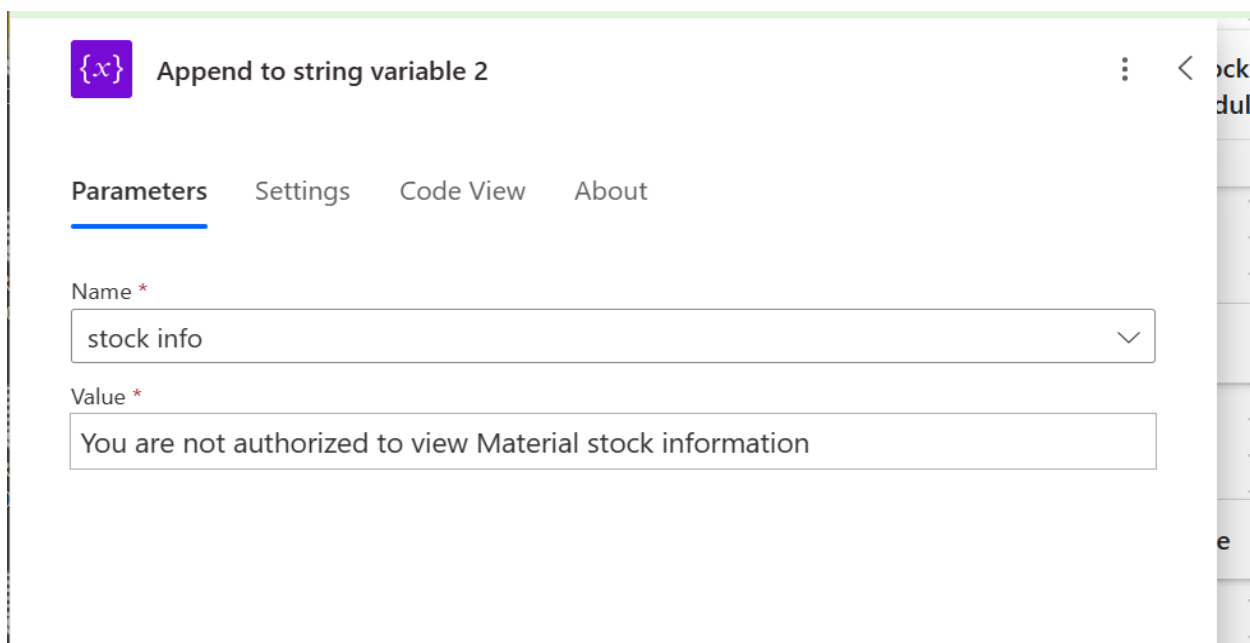
+

Now we need to ensure that the user is authorized, and that the OData API responded with a status code of “200”, for that, we will now add a condition to check that. If the response code is anything other than “200”, for the sake of this demo, we will assume that the user is not authorized to access this information (feel free to add more cases for other status codes).

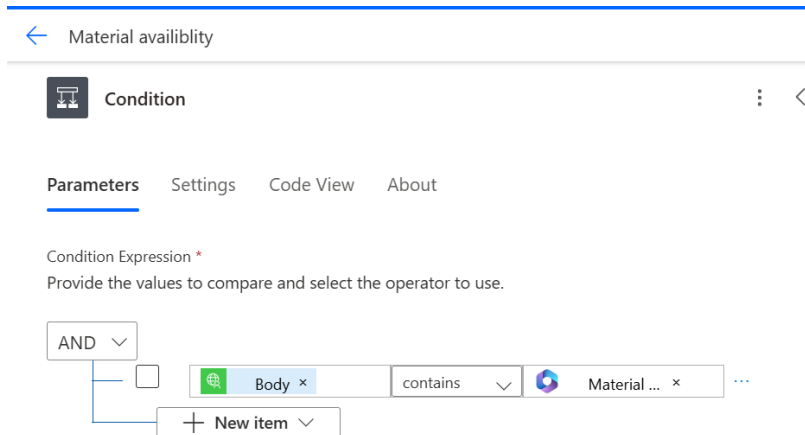




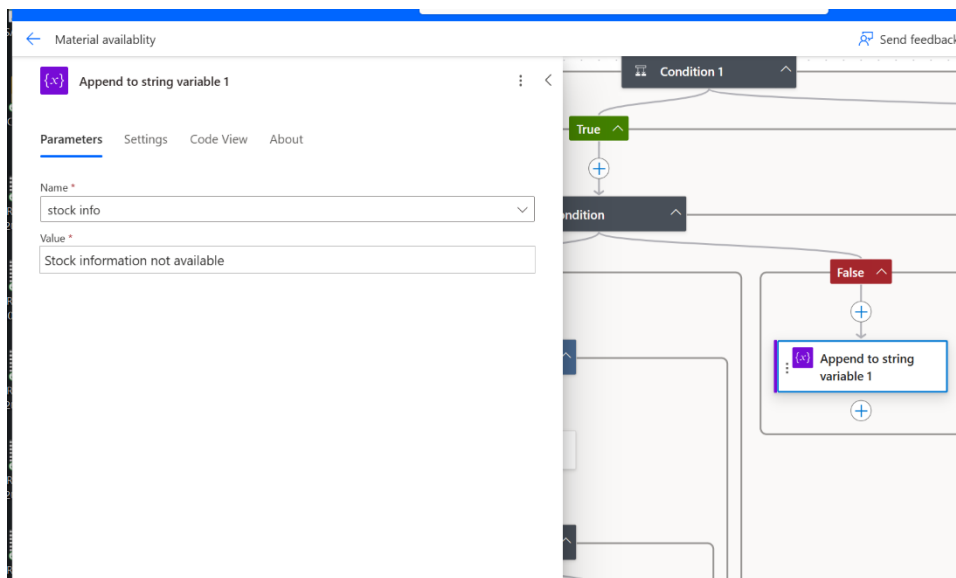
Under the false condition (unauthroized) , we will add this text to stock info:



Under the true condition, we will add another condition to ensure that this material is indeed present in the returned data, we will add a simple condition action to check if the input material is present in the body of the response.

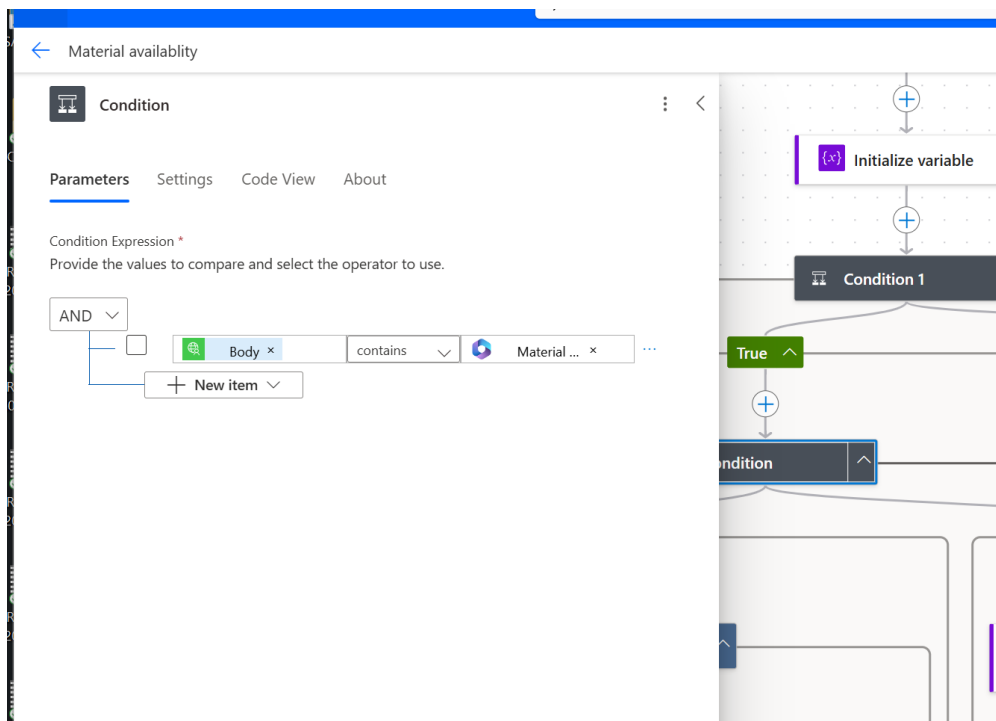
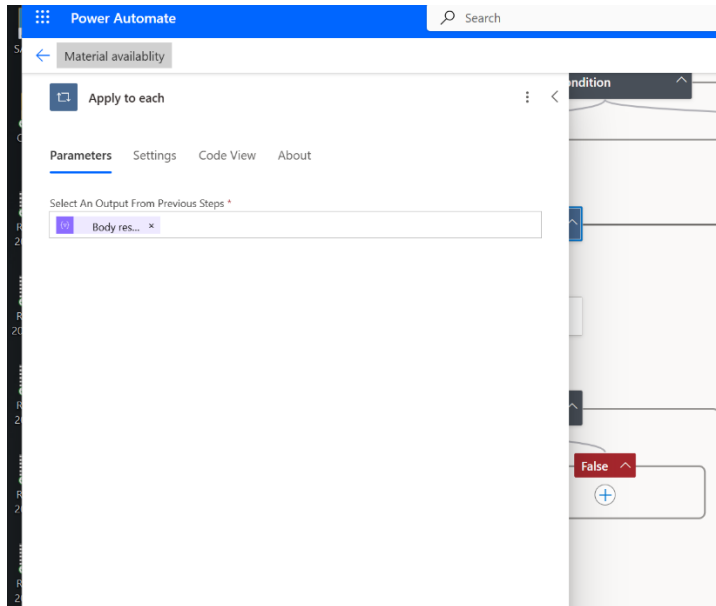


If there is no material present, we will append the below text to the variable “stock info”.

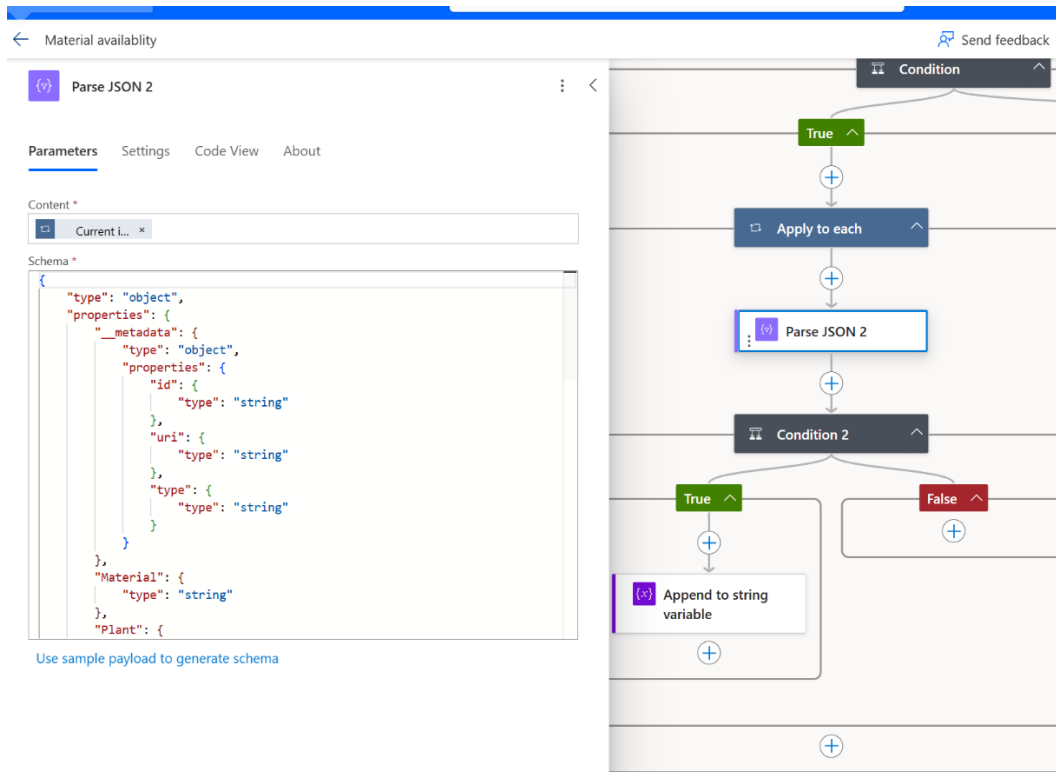


If it is, we will now go through the array of materials we passed, find the one we need the stock information for, and store the stock information.

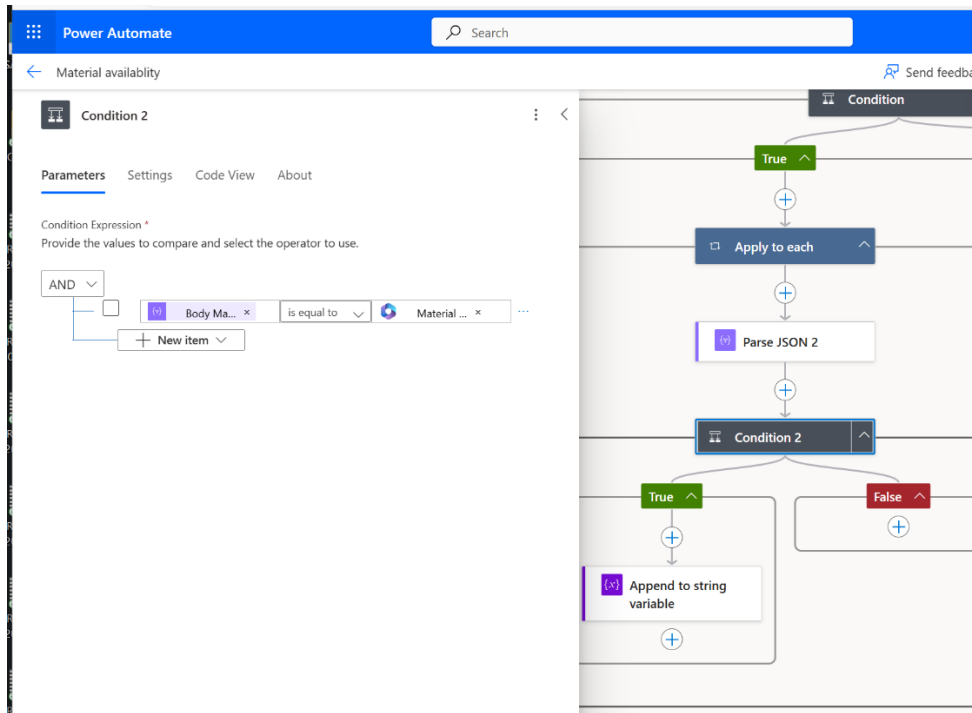
To accomplish that, we add a 'for each loop' and loop over the Parse JSON parameter "Body Results" – which has the array of all materials and their stock information.



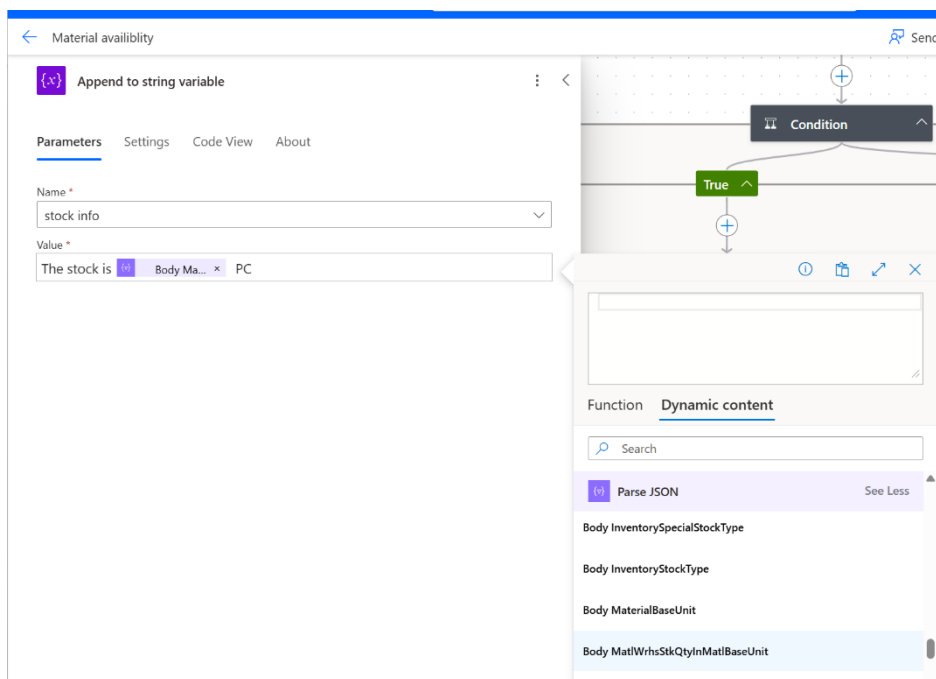
We also must parse each item of the results array as it is also an object (you can find the schema in the Power Automate flow on GitHub)



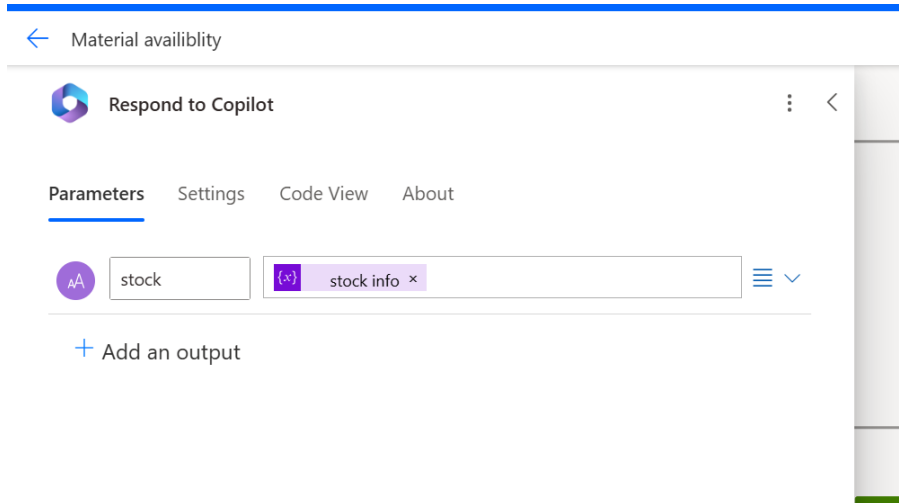
Next, we add another condition that checks if the material number of the current item is equal to the one, we input.



Lastly, if it is, we append the variable “MatlWrhsStkQtyInMaltBaseUnit”(the variable with the stock information) for the current item into the stock info variable.



Now we return the stock info variable back to the Copilot Studio and return the value back to the user.



Finally, Copilot Studio will return the information back to the user.

