



SAP Integration Developer Workshop

Hands-on Lab Step-by-Step Guide

Dec 2022

Contents

Lab Overview and Pre-requisites.....	3
Learning Objectives.....	3
Pre-requisites	3
Exercise 0: Copy sample invoices and templates to your desktop.....	13
Exercise 1: Open Power Automate and select your environment.....	14
Exercise 2: Open the SAP Integration Solution and Create an Environment Variable.....	15
Exercise 3: Build a flow that accesses the Purchase Order Table in SAP	20
Exercise 4: Configure the flow to modify the PO data.....	36
Exercise 5: Create a Power App to Display PO data.....	44
Exercise 6: Shape the data returned from SAP	57
Exercise 7: Procure to Pay Example.....	84
Exercise 8: Create and Add AI Builder Model.....	94
Exercise 9: Update purchase order cloud flow to extract more information.....	113
Exercise 10: Update Power App.....	123
Exercise 11: Finish the Procure to Pay journey	128

Lab Overview and Pre-requisites

Learning Objectives

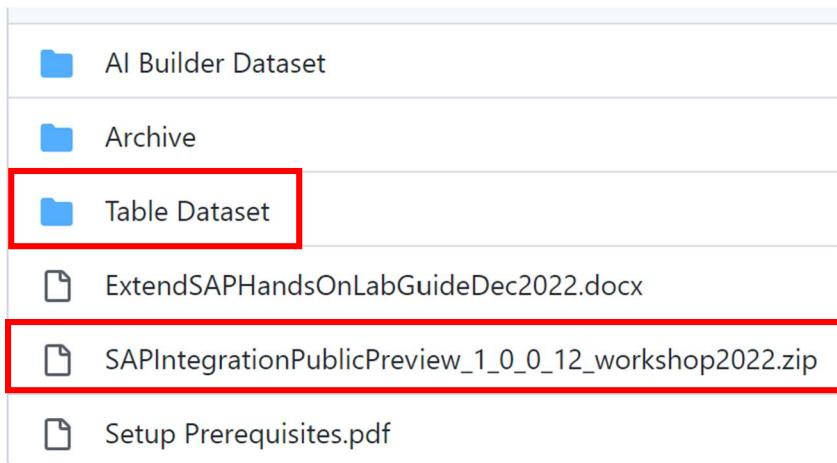
This lab is designed for the Integration Developer that will link SAP BAPI's and other services to Power Automate Flows that return and process SAP data. These Power Automate flows will be used by Application or Citizen Developers that will consume the SAP data coming out of the flow or potentially returning processed data back to SAP. This lab will help the Integration Developer understand the configuration and utilization of the SAP ERP API through hands on labs. In the labs, you will create a Canvas app and Power Automate flow to retrieve PO data from SAP. You will also learn how to shape the PO data with the Power Automate flows so your developer community can seamlessly interact with the SAP data. You will then use a pre-built solution to demonstrate the Procure to Pay process. You will create a Purchase Order app in Power Apps, configure AI Builder to intelligently process vendor invoices, and configure Power Automate to perform 3-way matching in SAP.

Pre-requisites

You will need credentials to an installed and configured SAP Data Gateway with access to an SAP server and access to a Power Apps and Power Automate environment before beginning these labs. Please see your instructor for these credentials.

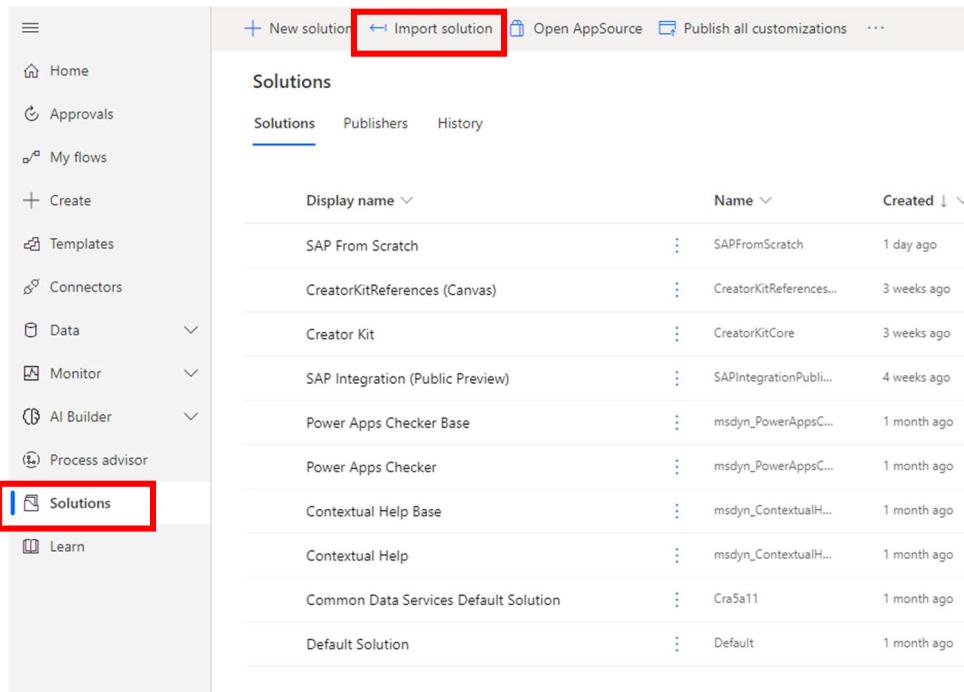
Download the following items from this [GitHub Repository](#):

- SAPIntegrationPublicPreview_1_0_0_12_workshop2022.zip
- (3) CSV files in the Table Dataset: gilman_dropdownses.csv, gilman_menugroups.csv, gilman_menuitems.csv



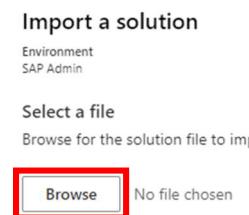
Part 1: Install Solutions

1. Browse to make.powerautomate.com and select the dedicated environment for your user. Please refer to the credentials provided to you by the instructor.
2. Select the **Solutions Tab** and click on **Import Solution**.



The screenshot shows the Power Automate interface. On the left, there's a sidebar with various options like Home, Approvals, My flows, Create, Templates, Connectors, Data, Monitor, AI Builder, and Process advisor. Below these, a red box highlights the 'Solutions' option, which is also underlined. At the top of the main content area, there are several buttons: '+ New solution', 'Import solution' (which has a red box around it), 'Open AppSource', 'Publish all customizations', and three dots. The main area is titled 'Solutions' and shows a table of existing solutions. The columns are 'Display name', 'Name', and 'Created'. The listed solutions include SAP From Scratch, CreatorKitReferences (Canvas), Creator Kit, SAP Integration (Public Preview), Power Apps Checker Base, Power Apps Checker, Contextual Help Base, Contextual Help, Common Data Services Default Solution, and Default Solution. Each solution entry has a three-dot menu icon to its left.

3. Click on **Browse** and select the Solution **SAPIntegrationPublicPreview_1_0_0_12_workshop2022.zip**. (You should have downloaded this file to your desktop. File is located in this [GitHub Repository](#)) After you have selected the file, click on **Next**.



The screenshot shows the 'Import a solution' dialog. It has two sections: 'Environment' (set to SAP Admin) and 'Select a file' (with a placeholder 'Browse for the solution file to import'). Below this is a 'Browse' button, which is highlighted with a red box. To the right of the button, it says 'No file chosen'.

4. Then hit **Next**, and **Next** to start the import.

Import a solution

Environment
Dev - labadmin1

Select a file

Browse for the solution file to import.

Browse

SAPIntegrationPublicPreview_1_0_0_12_workshop2022.zip

Import a solution

Environment
Dev - labadmin1

Details

Name
SAPIntegrationPublicPreview

Type
Unmanaged

Publisher
SAP

Version
1.0.0.12

Patch
No

[Advanced settings ▾](#)

Next

Cancel

Next

Cancel

5. On the next step of the import, we need to create a Connection for each Connection Reference in our solution. You will see a list of Connection References that are in your solution.

Import a solution

X

Environment
Dev - labadmin1

Refresh

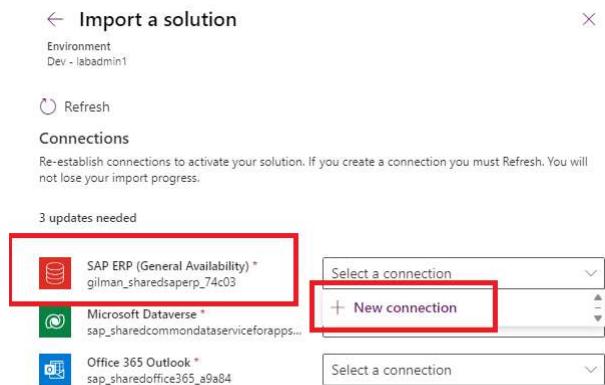
Connections

Re-establish connections to activate your solution. If you create a connection you must Refresh. You will not lose your import progress.

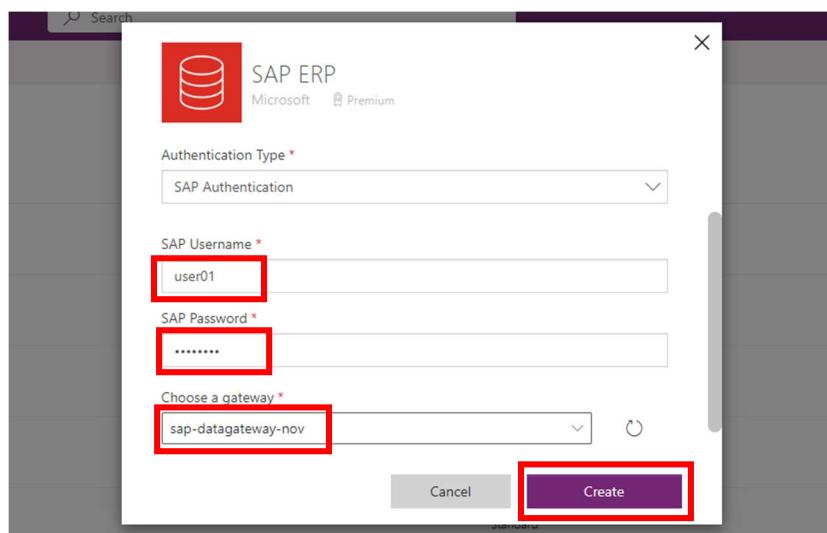
3 updates needed

- | | | |
|--|--|---------------------|
| | SAP ERP (General Availability) *
gilman_sharesaperp_74c03 | Select a connection |
| | Microsoft Dataverse *
sap_sharedcommondataserviceforapps... | Select a connection |
| | Office 365 Outlook *
sap_sharedoffice365_a9a84 | Select a connection |

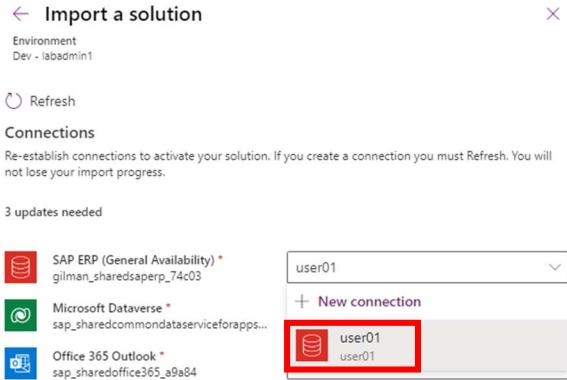
6. For the SAP ERP Connector, click on the drop down and then select + New Connection.



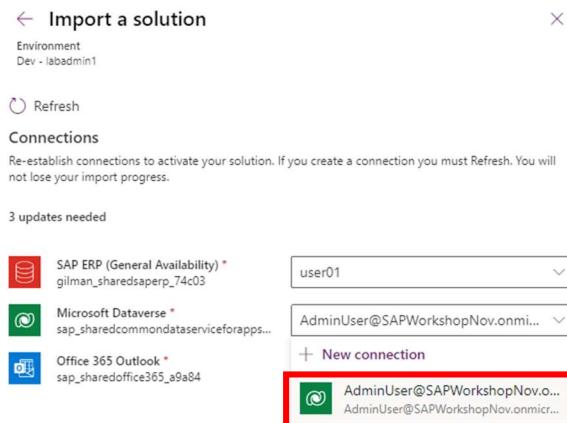
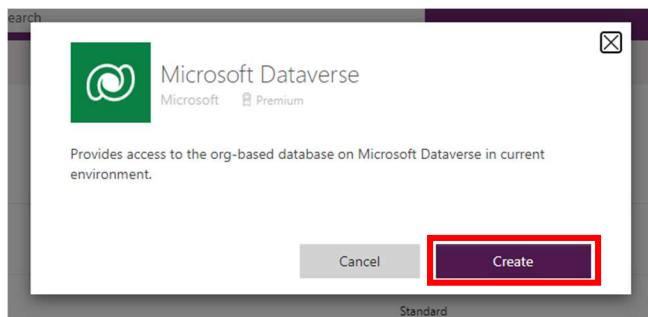
This will open a new tab to configure the connection. Use the SAP login and gateway information provided to you by the instructor and then click on **Create**.



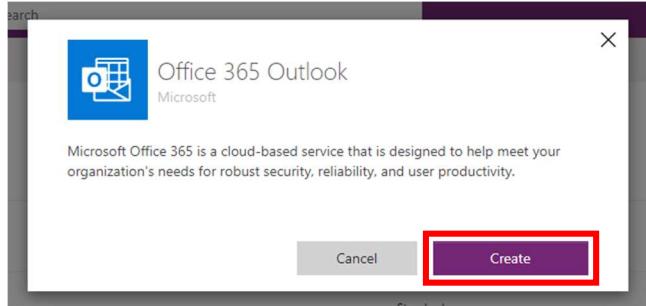
7. Once the Connection is created, go back to Import Solution tab, hit refresh and select the newly created Connection.



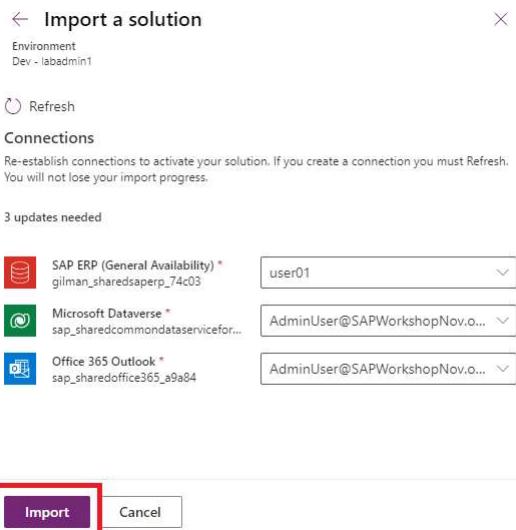
8. Repeat the last 2 steps to configure the Dataverse Connection. You might need to sign in again with your Microsoft Login Account. Use the O365 account login information provided by the instructor.



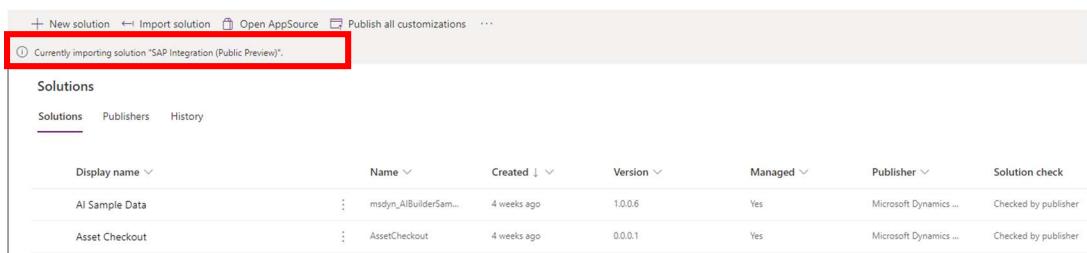
9. Repeat the last 2 steps to configure the Office 365 Outlook Connection. You might need to sign in again with your Microsoft Login Account. Use the O365 account login information provided by the instructor.



- Once we have all Connections selected, select **Import** button to Import Solution.



- This notification shows that Solution is importing. This will take several minutes to complete the import.



- When Solution has been successfully imported, then select **Publish all customizations**.

The screenshot shows the SAP Power Platform canvas interface. On the left is a navigation sidebar with options like Home, Learn, Apps, Create, Dataverse, Tables, and Choices. The main area is titled 'Solutions' with tabs for Solutions, Publishers, and History. A green notification bar at the top says 'Solution "SAP Integration (Public Preview)" imported successfully.' Below it, a red box highlights the 'Publish all customizations' button in the top navigation bar. The 'Solutions' table lists one item: 'SAP Integration (Public Preview)' with a creation timestamp of '6 minutes ago' and version '1.0.0.12'.

13. You should see notification that all your changes have been published successfully as below.

The screenshot shows the SAP Power Platform canvas interface. The navigation sidebar is identical to the previous screenshot. The main area shows a green notification bar with two messages: 'Solution "SAP Integration (Public Preview)" imported successfully.' and 'Published all customizations successfully.' A red box highlights the second message. The 'Solutions' table lists one item: 'SAP Integration (Public Preview)' with a creation timestamp of '8 minutes ago' and version '1.0.0.12'.

Part 2: Import Data

Option 1: Import Data via Excel

We have chosen to have the data tables blank for the solution. You can create your own datasets that are customized for your customer and then include them in the solution for an easier solution import.

1. Open the SAP Integration (Public Preview). You can either click on the solution name, click on **Edit** in the top ribbon after you have selected the solution or click on the **ellipses (...)** and select **Edit**.

The screenshot shows the SAP Integration Solutions list. The left sidebar includes links like Home, Approvals, My flows, Create, Templates, Connectors, Data, Monitor, AI Builder, Process advisor, and Solutions. The Solutions section is active. The main area lists solutions with columns for Display name, Name, and Created. One solution, "SAP Integration (Public Preview)", is highlighted with a red box and has a blue checkmark icon next to its name. Its row is also highlighted with a red box.

Display name	Name	Created
SAP From Scratch	SAPFromScratch	1 day ago
CreatorKitReferences (Canvas)	CreatorKitReferences...	3 weeks ago
Creator Kit	CreatorKitCore	3 weeks ago
SAP Integration (Public Preview)	SAPIntegrationPubli...	4 weeks ago
Power Apps Checker Base	msdyn_PowerAppsC...	1 month ago
Power Apps Checker	msdyn_PowerAppsC...	1 month ago
Contextual Help Base	msdyn_ContextualH...	1 month ago
Contextual Help	msdyn_ContextualH...	1 month ago
Common Data Services Default Solution	Cra5a11	1 month ago
Default Solution	Default	1 month ago

- Click on the **Edit** button in the top navigation bar.
- Click on the Drop Down to expand all the Tables.

The screenshot shows the SAP Integration Objects page for the "SAP Integration (Public Preview)" solution. The left sidebar includes Back to solutions, Overview, Objects (which is selected), and History. The main area shows a tree view of objects under "Objects". The "Tables (4)" node is expanded, showing four items: DropDownListValues, MenuGroup, MenuItem, and UserSearchCache. A red box highlights the "Tables (4)" node.

- All (40)
 - AI models (1)
 - Apps (3)
 - Chatbots (0)
 - Cloud flows (22)
 - Connection references (6)
 - Environment variables (4)
 - Tables (4)**
 - > DropDownListValues
 - > MenuGroup
 - > MenuItem
 - > UserSearchCache

3. Import Data to DropDownListValues Table. Select the **DropDownValues** table on the left-hand side. Then click on **Import** in the upper ribbon and then select **Import data from Excel**.

The screenshot shows the SAP Integration Public Preview interface. On the left, there's a sidebar with 'Objects' and a search bar. Under 'Tables (4)', the 'DropDownValues' table is selected, highlighted with a red box. In the top right, there's a ribbon with various icons. A red box highlights the 'Import data from Excel' icon in the 'Import' section.

4. Click on **Upload** and select the file **gilman_dropdownses.csv** file. (You should have downloaded this file to your desktop. The file is located in the **Table Dataset Folder** in this [GitHub Repository](#))

The screenshot shows the 'Import data' screen. It says 'Click the Upload button to upload csv or Excel files to import. Use the Map columns button to map source file columns to table columns.' Below is a table with one row for 'DropDownValues'. The 'File' column shows 'File not uploaded'. The 'Upload' button is highlighted with a red box. The 'Mapping status' column shows 'Not mapped'.

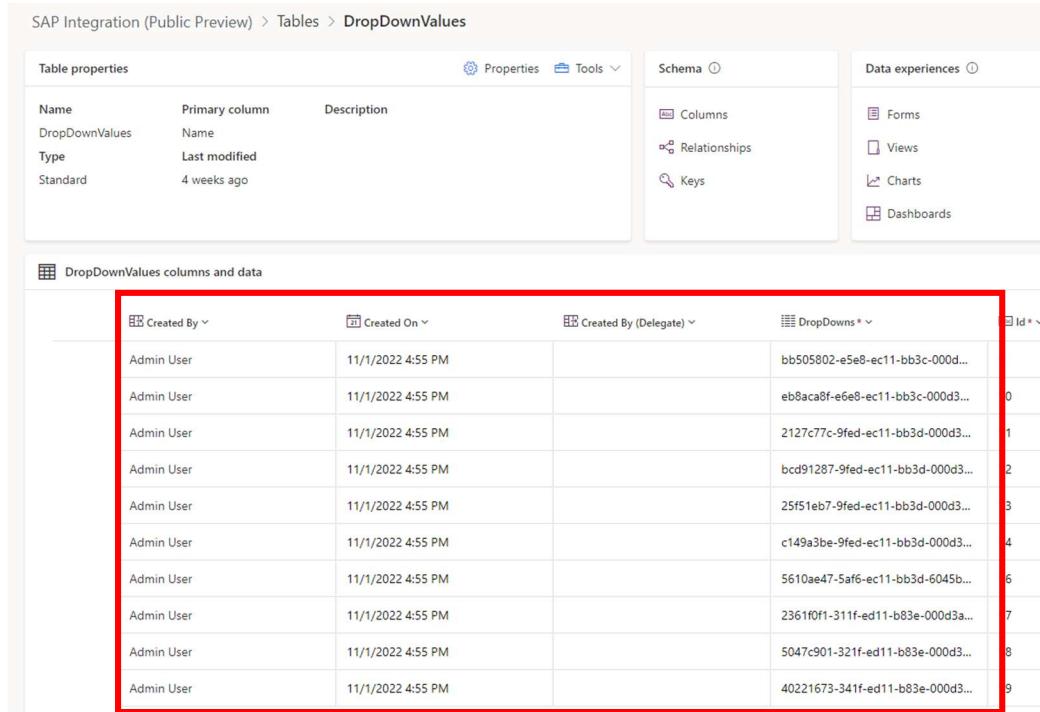
Name	File	Mapping status
DropDownValues	File not uploaded	Upload Not mapped

5. Wait for **Mapping Status** to show mapping was successful. Then click on **Import** button on top right corner.

The screenshot shows the 'Import data' screen again. It says 'Click the Upload button to upload csv or Excel files to import. Use the Map columns button to map source file columns to table columns.' Below is a table with one row for 'DropDownValues'. The 'File' column shows 'gilman_dropdownses.csv'. The 'Upload' button is highlighted with a red box. The 'Mapping status' column shows 'Mapping was successful' with a green info icon. At the bottom, a green bar says 'Import completed successfully.'

Name	File	Mapping status
DropDownValues	gilman_dropdownses.csv	Upload Mapping was successful

Refresh and ensure you are showing data in **DropDownValues** table as shown below.



The screenshot shows the SAP Integration (Public Preview) interface with the path "Tables > DropDownValues". The top navigation bar includes "Properties", "Tools", "Schema", "Data experiences", and links for "Columns", "Relationships", "Keys", "Forms", "Views", "Charts", and "Dashboards". The main area displays the "DropDownValues columns and data" table. The table has columns: "Created By", "Created On", "Created By (Delegate)", "DropDowns", and "Id". The data grid contains 10 rows, each representing a record in the table. The "DropDowns" column contains unique identifiers for each row, and the "Id" column contains values from 0 to 9. The entire data grid is highlighted with a red border.

Created By	Created On	Created By (Delegate)	DropDowns	Id
Admin User	11/1/2022 4:55 PM		bb505802-e5e8-ec11-bb3c-000d...	0
Admin User	11/1/2022 4:55 PM		eb8aca8f-e6e8-ec11-bb3c-000d3...	1
Admin User	11/1/2022 4:55 PM		2127c77c-9fed-ec11-bb3d-000d3...	2
Admin User	11/1/2022 4:55 PM		bcd91287-9fed-ec11-bb3d-000d3...	3
Admin User	11/1/2022 4:55 PM		25f51eb7-9fed-ec11-bb3d-000d3...	4
Admin User	11/1/2022 4:55 PM		c149a3be-9fed-ec11-bb3d-000d3...	5
Admin User	11/1/2022 4:55 PM		5610ae47-5af6-ec11-bb3d-6045b...	6
Admin User	11/1/2022 4:55 PM		2361f0f1-311f-ed11-b83e-000d3a...	7
Admin User	11/1/2022 4:55 PM		5047c901-321f-ed11-b83e-000d3...	8
Admin User	11/1/2022 4:55 PM		40221673-341f-ed11-b83e-000d3...	9

6. Repeat the last 3 steps for the remaining tables using the data files indicated below.

Table Name	Data File to Import
MenuGroup	gilman_menugroup.csv
MenuItem	gilman_menuitems.csv

Option 2: Import Data using PowerShell Script

If you are comfortable using PowerShell Script, you can download the script from the same [GitHub repository](#). Download from the [PowerShell Script for Importing Table Dataset](#) the following files:

- Configdata.zip
- importConfigDataSAPSolution.ps1

Execute the `importConfigDataSAPSolution.ps1` to import Data with `configdata.zip` file using following syntax:

```
\importConfigDataSAPSolution.ps1 -url -configDataPath
```

Example:

```
.\importConfigDataSAPSolution.ps1 -url https://orgXXXXXXX.crm.dynamics.com/ -  
configDataPath C:\Downloads\configdata.zip
```

The PowerShell Script should create a new authentication profile to the environment specified and import the configuration data.

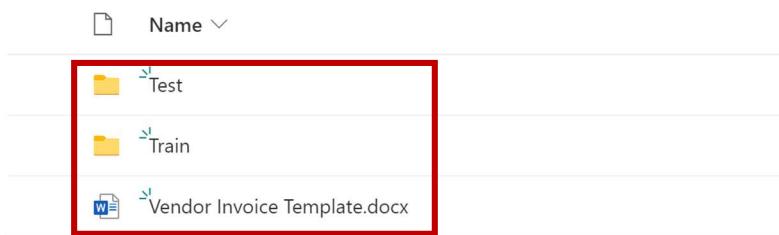
If you have any problems running PowerShell Script, check out the complete Deployment Guide here.

[PowerPlatformSAPIIntegration/deployment at main ·
jongilman88/PowerPlatformSAPIIntegration \(github.com\)](#)

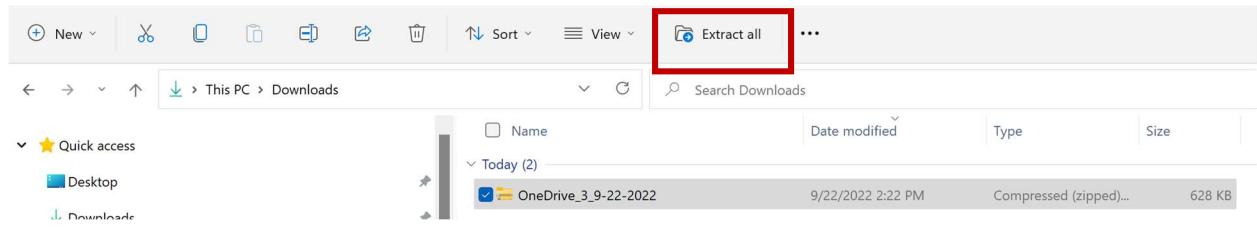
Exercise 0: Copy sample invoices and templates to your desktop.

1. Browse to [GitHub Respository for the Workshop](#). Download the **Test** and **Train** folders and the **Vendor Invoice Template** from the **AI Builder Dataset** folder.

Documents > General > GitHub Repository > **AI Builder Dataset** ↗

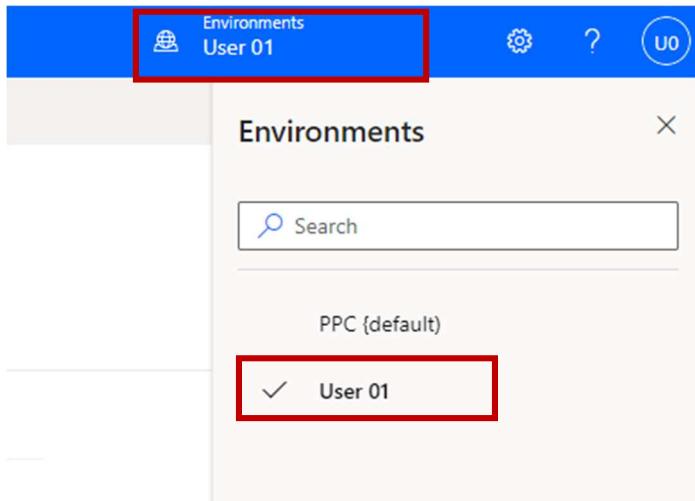


2. Create a new folder on your desktop and then extract the zip file from the downloads so you have these documents available during the lab.



Exercise 1: Open Power Automate and select your environment.

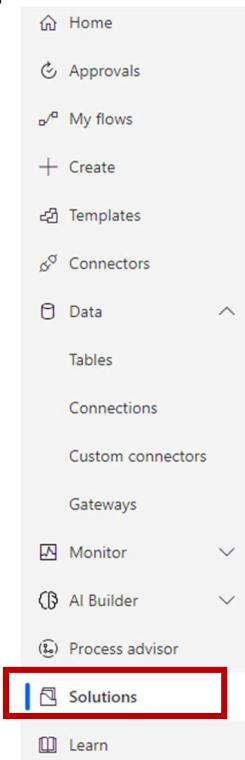
1. Open your Power Platform Maker Portal. Using an In-Private or Incognito browser, browse to <https://make.powerautomate.com> using your new credentials supplied by the instructor.
2. Select the correct environment. In the upper right-hand corner, click on the **Environment** and select the correct environment for this lab based on the information supplied by the instructor.



Exercise 2: Open the SAP Integration Solution and Create an Environment Variable.

It is best practice to use a solution as a container for all your apps and flows. For this workshop, we will be working in a solution that has already been created for you. We will add some new cloud flows, canvas apps and Environment Variables but use the existing Connector Reference.

1. Open a Solution. In right left-hand panel, click on Solutions.



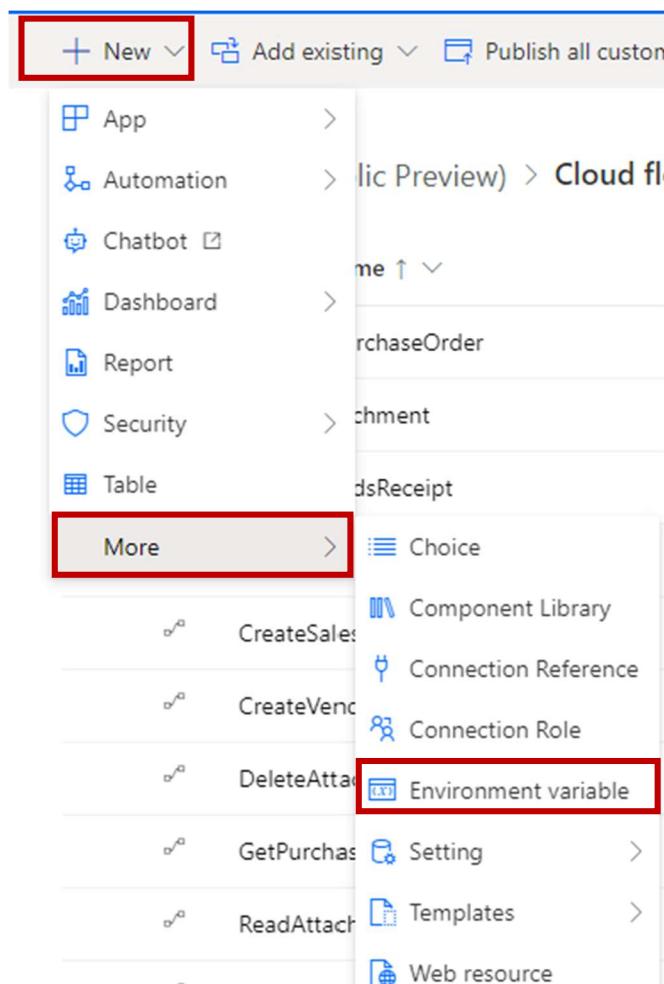
2. Select the SAP Integration (Public Preview) solution.

The image shows the 'Solutions' list page. The top navigation bar includes New solution, Edit, Delete, Export solution, Solution checker, Show dependencies, See history, Clone, Apply Upgrade, and more. The main table lists solutions with columns: Display name, Name, Created, Version, and Managed. The 'SAP Integration (Public Preview)' solution is selected and highlighted with a red box.

Display name	Name	Created	Version	Managed
Power Platform Conference	PowerPlatformConfe...	26 minutes ago	1.0.0.0	No
SAP Integration (Public Preview)	SAPIntegrationPubli...	1 hour ago	1.0.0.2	No
Power Apps Checker Base	msdyn_PowerAppsC...	5 days ago	1.2.0.198	Yes
Power Apps Checker	msdyn_PowerAppsC...	5 days ago	1.2.0.198	Yes
Contextual Help Base	msdyn_ContextualH...	5 days ago	1.0.0.22	Yes

3. Create a new Environment variable. SAP has several different connection parameters, and it is cumbersome for makers to enter these for each Power Automate SAP action. Makers are now able to reuse environment variables within solution-aware flows that contain all the SAP system information. Once the environment variable is defined, makers can use the environment variable in their SAP actions without having to re-enter all the system information again.

Click on **+ New, More** and then **Environment variable**.



4. Configure the Environment variable. Enter the following to configure the Environment variable then click on **Save**.

Display Name: **sap_application**

Description: **SAP System Variables**

Data Type: **Text**

NOTE: Even though we are entering a JSON blob, we must choose Text for the variable data type.

Default Value: Copy and paste the JSON blob below.

```
{"AppServerHost": "sap.clearsoftware.com", "LogonType": "ApplicationServer",  
"SystemNumber": "00", "Client": "100"}
```

New environment variable X

Environment variables can have different values when re-used, enter information about this variable so that future users can understand its purpose. [Learn more](#)

Display name *

 sap_application

Name * (i)

 sap_ sap_application

Description

 SAP System Variables

Data Type *

 Text

Default Value (i)

 {"AppServerHost" : "sap.clearsoftware.com", "...}

Current Value

Override the default value by setting the current value for your environment.

+ New value

Save

Cancel

Additional SAP Properties you may be using for environment variables for your customers:

Property	Description
AppServerHost	The hostname of the SAP Application Server.
AppServerService	The service name or port number of the specific SAP Application Server to connect to (Optional for connection type (Logon) A - Application Server).
Client	The SAP client ID to connect to the SAP system. The SAP backends' client (or 'Mandant') into which to log in. It is a number ranging from 000 to 999.
LogonGroup	The Logon Group for the SAP System, from which the Message Server shall select an Application Server (Only available if connection type (Logon) is B - Message Server (Group)).
LogonType	The type of logon to the SAP System, either Application Server Logon (Type A) or Group Logon (Type B aka Message Server). Allowed values: ApplicationServer, Group
MessageServerHost	The hostname of the SAP System's Message Server (central instance) aka R3 System Name (Mandatory if connection type (Logon) is B - Message Server (Group)).
MessageServerService	The Service Name (as defined in etc/services) or the Port Number under which the Message Server is listening for load-balancing requests (Mandatory if connection type (Logon) is B - Message Server (Group) and System ID is not present).
SncCertificate	X.509 certificate in Base64 encoded form, without the begin or end certificate tags.
SncMyName	The installed SNC solution usually knows its own SNC name. Only for solutions supporting 'multiple identities', you may need to specify the identity to be used for this destination/server (optional).
SncLibraryPath	Name or path of the SNC library to be used. With the On-Premises Data Gateway, the path can be an absolute or relative to the NCo library.
SncPartnerName	The backends' SNC name (Required when Logon Type is Application Server).
SncQop	Quality of Service to be used for SNC communication of this destination/server. Allowed values: Authentication, Integrity, Privacy, Default, Maximum
SncSso	The SNC SSO specifies whether to use SNC identity or credentials provided on RFC level.

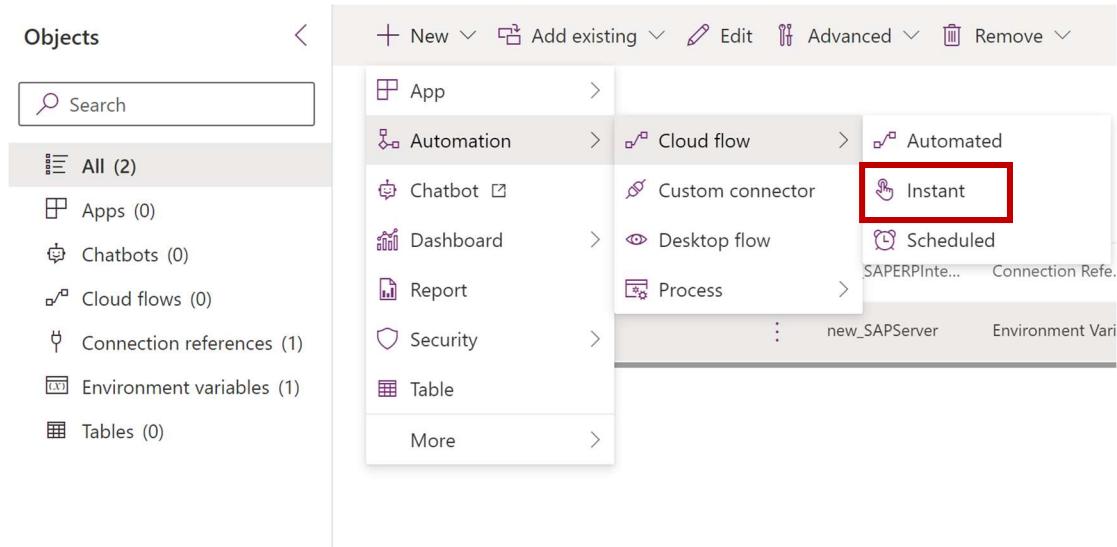
SystemId	The SAP system's three-letter system ID (Mandatory if connection type (Logon) is B - Message Server (Group) and Message Server Service is not present).
SystemNumber	The SAP System's System Number. It is a number ranging from 00 to 99 (Mandatory if connection type (Logon) is A - Application Server).
UseSnc	When selected, the connections will be secured with SNC. Allowed values: Yes

Exercise 3: Build a flow that accesses the Purchase Order Table in SAP.

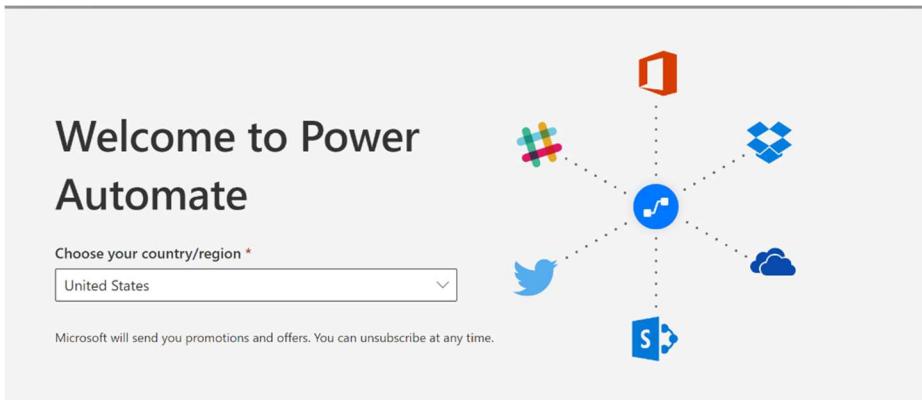
There are two primary ways to access SAP data:

- Using an SAP Business API - also known as a BAPI. This returns a set of fields in JSON format that you can manipulate in your flow. BAPI's can also update SAP by calling an Update BAPI and passing the expected JSON formatted fields and values. Microsoft provides examples of hundreds of common BAPI calls as part of the sample applications we provide with the SAP connector. To help you understand how those were constructed in case you need to update or modify them we'll build a simple example of using a BAPI.
- Performing an SAP ReadTable action. Reading tables - especially with hundreds or thousands of rows - is much faster and more efficient by using the ReadTable action. Unlike a BAPI, however, ReadTable returns a set of fixed length records which you have to map into variables before you can manipulate the individual fields. Microsoft provides samples of several common ReadTable actions against various SAP Tables. To help you understand how those were constructed in case you need to update or modify them we'll build a simple request.

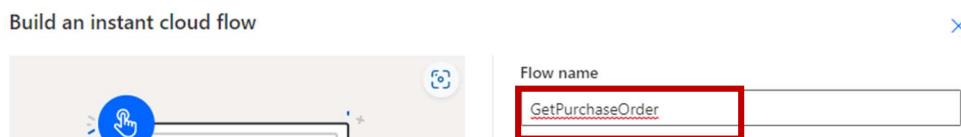
1. Create a new Cloud Flow. In the SAP Integration (Public Preview) solution, click on **+** **New**, then **Automation**, then **Cloud flow** and finally **Instant**.



2. Select Region for Power Automate. You may be prompted to select your region for Power Automate. Please select **United States** and then **Get Started**.



3. Name the flow. Type **GetPurchaseOrder** for the Flow name.



4. Select the trigger and create the flow. Select the **PowerApps** trigger. Then click on **Create**.

Build an instant cloud flow

Flow name
GetPurchaseOrder

Choose how to trigger this flow *

- Manually trigger a flow
Flow button for mobile
- PowerApps PowerApps
- When Power Virtual Agents calls a fl...
Power Virtual Agents
- When a flow step is run from a busin...
Microsoft Dataverse
- For a selected message (V2)
Microsoft Teams
- From the compose box (V2)
Microsoft Teams
- When someone responds to an adap...

Triggered manually from any device, easy-to-share instant flows automate tasks so you don't have to repeat yourself.

Examples:

- Get an automatic mobile alert whenever a VIP client emails you
- Save all your email attachments to a folder automatically

Skip Create Cancel

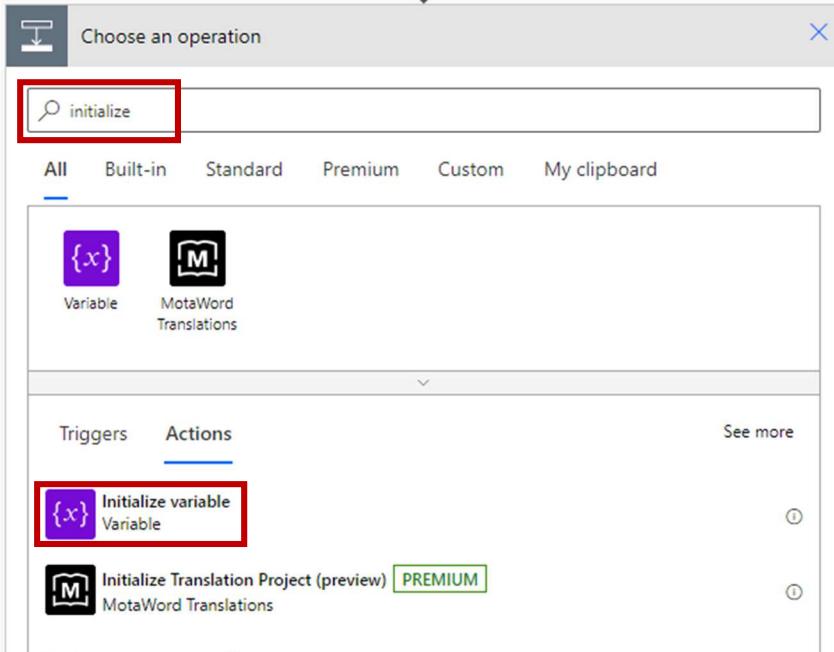
5. Add a new step to create a Variable to hold the Purchase Order being passed in from Power Apps. Click on **+ New Step**.

PowerApps

+ New step

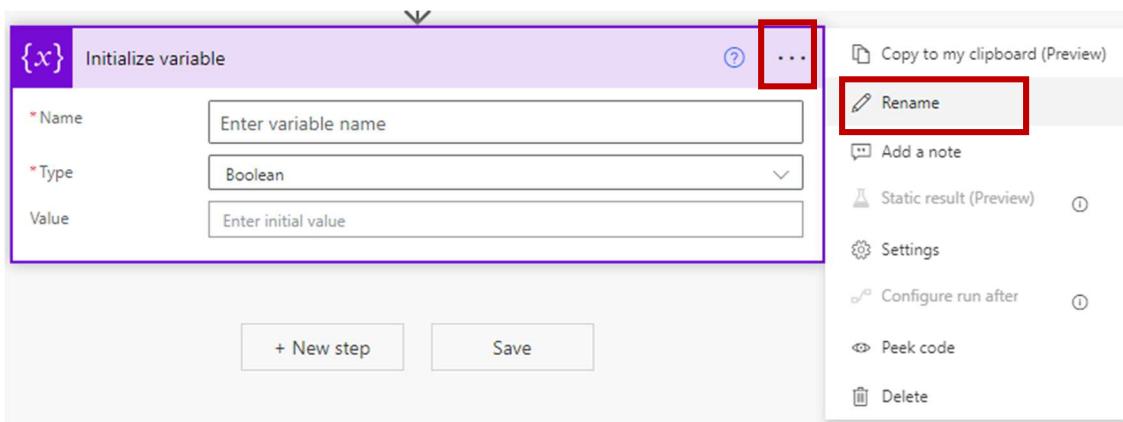
Save

6. Search for **Initialize** and select the **Initialize Variable** action.



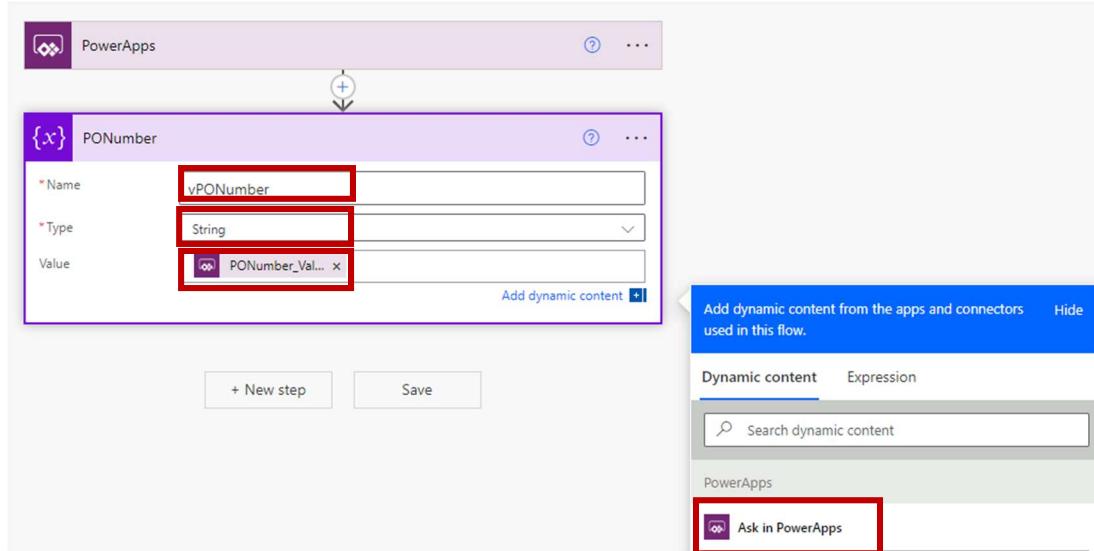
7. Rename the action. Click on the ellipses (...) and then **Rename**. Rename to PONumber.

NOTE: The name of the variable is used in the Power App, so naming to something logical will help when using the variable in Power Apps.

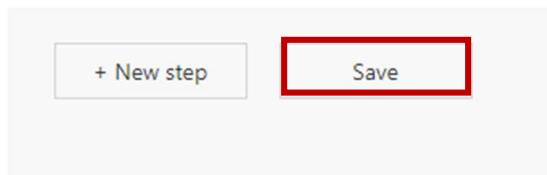


8. Configure the action. Fill out the Initialize Variable with the following:

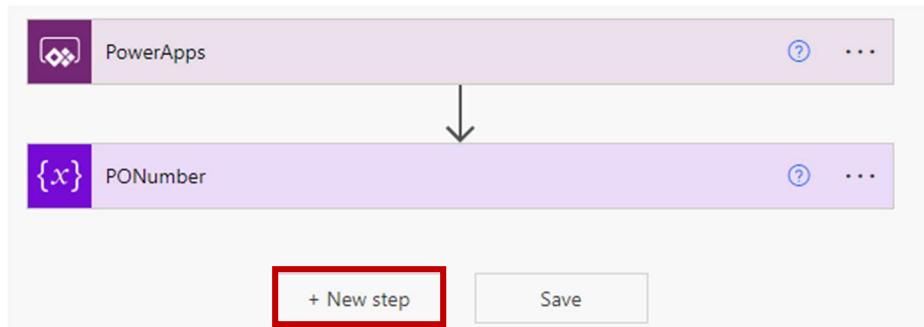
- ❑ Name of Variable – vPONumber
 - ❑ Best Practice – use a small v in front of name to indicate it is a variable.
- ❑ Type – String
- ❑ Value – Using Dynamic Content and select Ask in Power Apps
 - ❑ NOTE: The value changes to the Action Name.



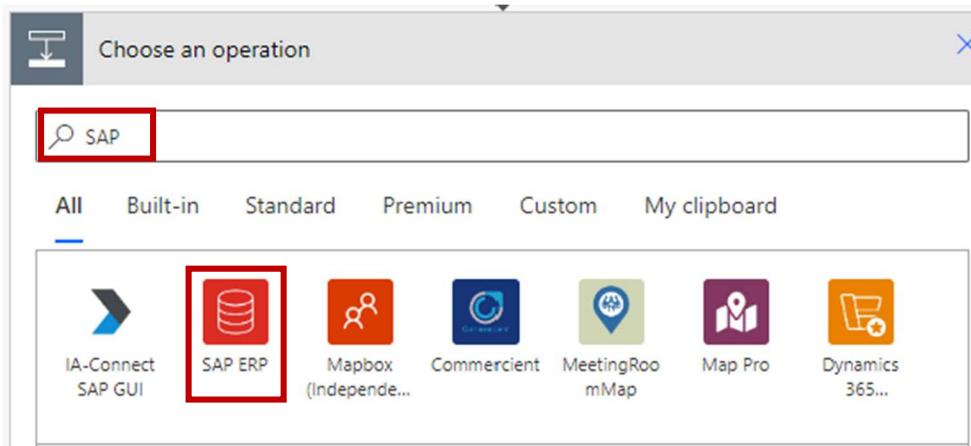
9. Save the flow. Click on Save.



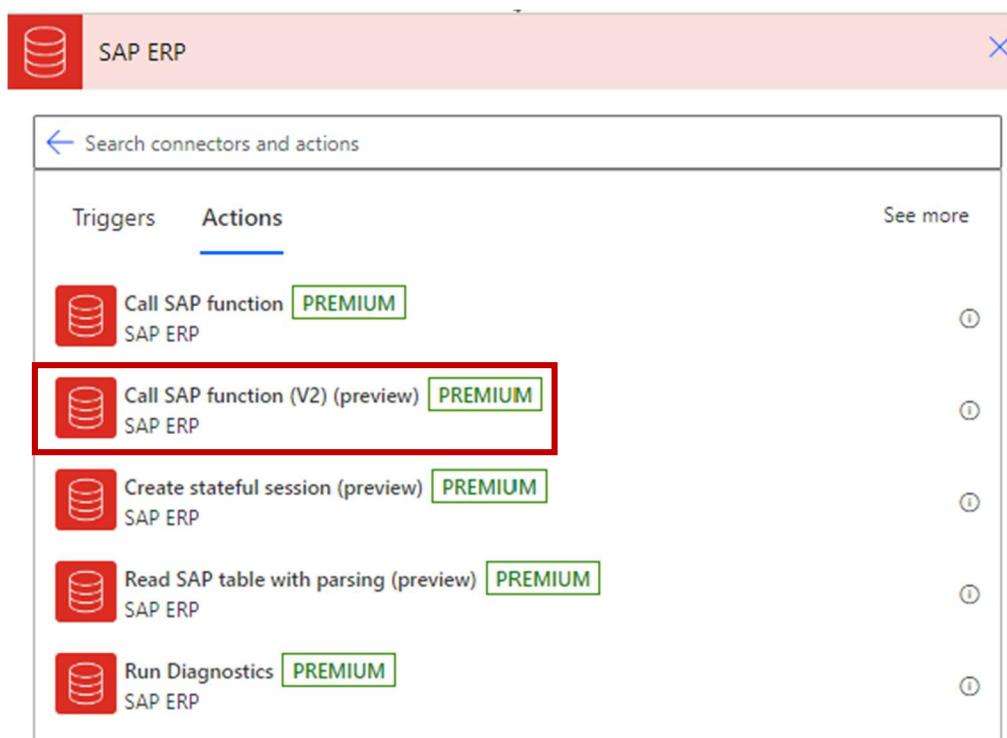
10. Add a new step to call the SAP Function. Click on + New Step.



11. Select SAP ERP Connection. Type in SAP in search bar and select SAP ERP.



12. Select the Action. Click on the Call SAP function (V2) (preview)



13. Configure the action.

- SAP System – Select the Environment variable we created – `sap_application` from Dynamic Content.

- RFC Name – Type in **BAPI_PO_GETDETAIL1**
- PURCHASEORDER – Select the Variable we created - **vPONumber** – from Dynamic Content.

The screenshot shows the 'Call SAP function (V2) (Preview)' step configuration and the 'Dynamic content' pane side-by-side.

Call SAP function (V2) (Preview) Step Configuration:

- * SAP system: sap_application
- * RFC name: BAPI_PO_GETDETAIL1
- * PURCHASEORDER: {x} vPONumber
- ACCOUNT_ASSIGNMENT: Account Assignment Data
- DELIVERY_ADDRESS: Delivery address
- HEADER_TEXT: Header Txt
- INVOICEPLAN: Invoicing Plan
- ITEM_TEXT: Item Text
- SERIALNUMBERS: Serial Numbers
- SERVICES: External Service Data
- VERSION: Version Management

Dynamic content pane:

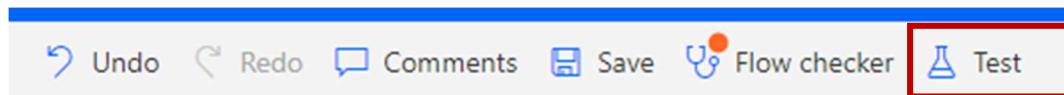
- Add dynamic content from the apps and connectors Hide
- Dynamic content Expression
- Search dynamic content
- Environment Variables
 - SAP Application Server (sap_SAPIAddress)
 - SAP Development (sap_SAPDevelopment)
 - SAP SSO with Load Balancing (sap_SAPSSOwithLoadBal...)
 - sap_application (sap_sap_application)**
- Variables
 - {x} vPONumber**
- PowerApps

14. Save the flow. Click on **Save**.

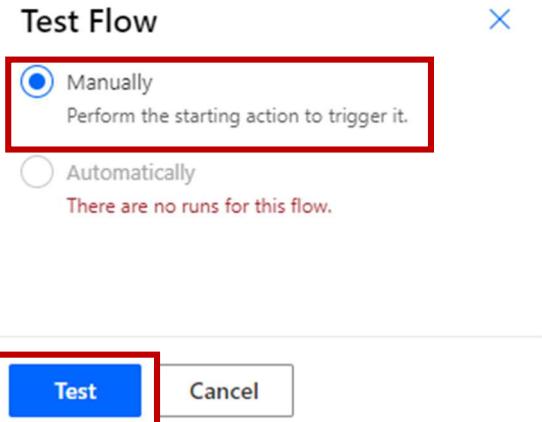
NOTE: You may have to scroll down through all the parameters to find the save button.



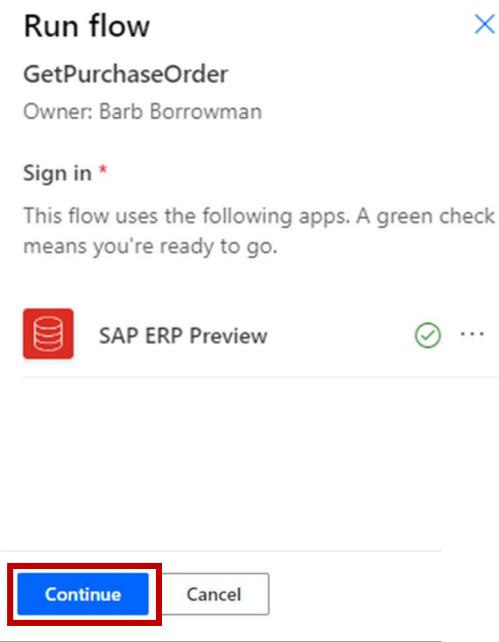
15. Test our SAP Function. We can test the SAP Function inside the flow. Click on **Test** in the Upper Ribbon.



16. Select Manually to test the flow and then **Test**.



17. First, the flow will confirm we have the right privileges to connect to the SAP System.
Click **Continue**.



18. Next, we will enter a PO Number that we want to return data. Enter **4500000566**

Run flow X

GetPurchaseOrder
Owner: Barb Borrowman

PONumber_Value *

4500000566

This flow uses SAP ERP Preview.
[Review connections and actions](#)

Run flow Cancel

19. View the flow run. Click on **Done** and it will bring you to the Flow Runs Page.

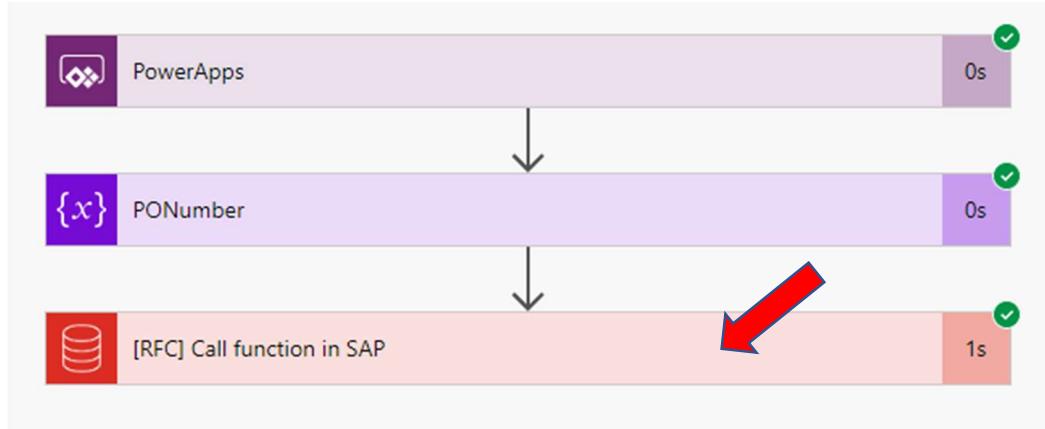
Run flow X



Your flow run successfully started. To monitor it,
go to the [Flow Runs Page](#).

Done

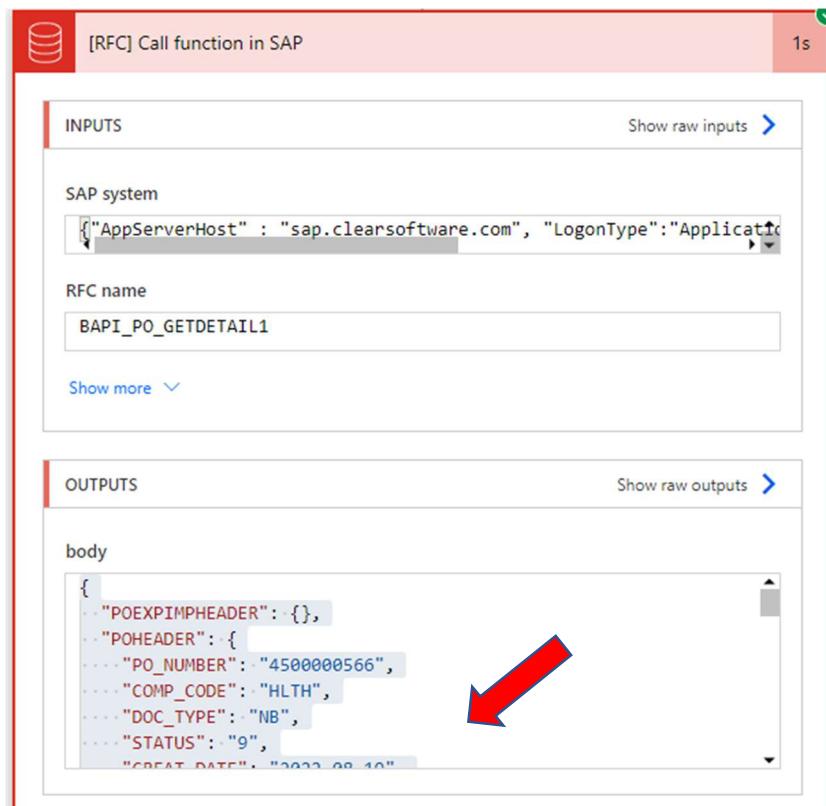
20. Expand the SAP Call Function. Click on the Call Function in SAP to expand the flow.



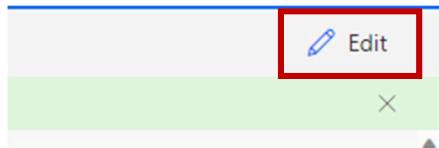
21. Copy Output Section. You can see all the data that was returned from the flow in the Outputs section. We want to copy this data so we can insert into a Parse JSON action to layout the data in the editor so we can view it. We will use this multiple times to allow the sample to generate the schema rather than typing it in.

To select all the text in the Output section, click in the body and then Ctrl-A.

To copy all the text in the Output section, click Ctrl-C



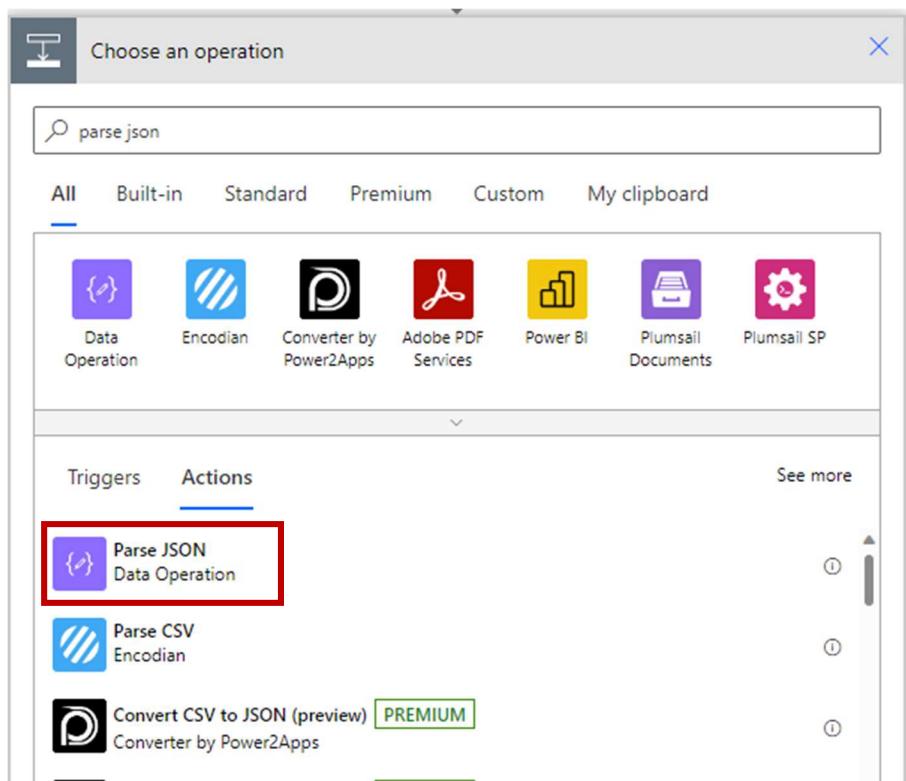
22. Edit the flow. Click on **Edit** in the upper right-hand corner to go back to editing the flow.



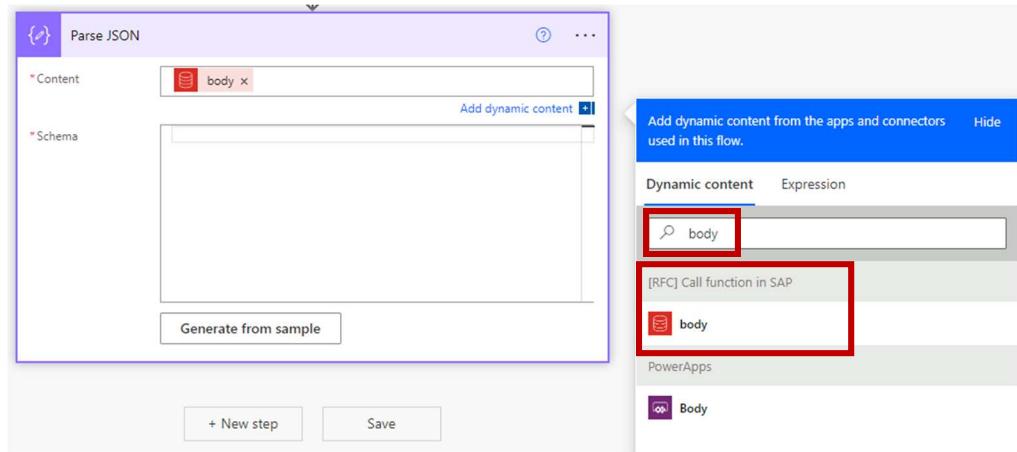
23. Add a new step. Click on **+ New Step**.



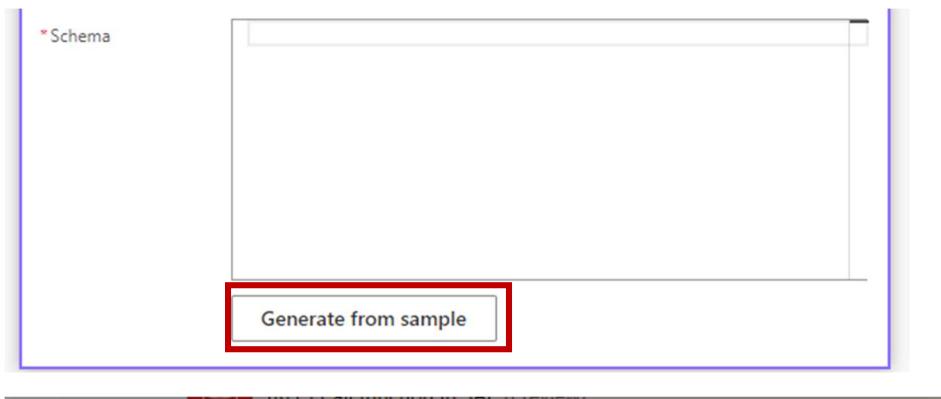
24. Search for **Parse JSON** in the Operation and select **Parse JSON** as the action.



25. Configure the Action. The content will be pulled from our previous SAP action, so from the **Dynamic Content**, search for body and select **Body** from the [RFC] Call function for SAP.



26. The Schema will be what we copied from the Output of the flow. Click **Generate from Sample** and then paste (Ctrl-V) into the JSON Payload and click **Done**.



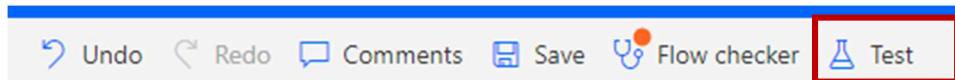
27. Save the flow. Click on **Save**.



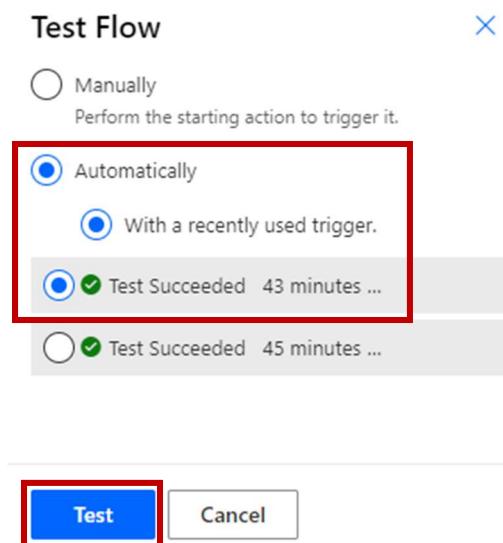
It may take a few minutes to save and you can look for the completed action at the top of the screen,



28. Test the flow again. We will now test the flow again. Click on **Test** in the upper right-hand corner.



29. Select **Automatically** and then **With a recently used trigger**. Select the **most recent run** and then **Test**. If you need to test manually, use PO # 4500000566 for the test.



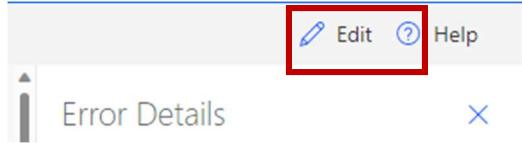
30. Test Failed. We are moved over to the Run History and can see that our Parse JSON call has failed. The sample we inserted had different types than our schema was expecting. We can use the error output to update our schema.

Expand the Parse JSON action and look at the Errors section in Outputs. You can see there are (3) values that have an error – CL_VAL_LOC, IVVAL_LOC and IVVAL_FOR. All these were expecting an integer but received back a number.

The screenshot shows the configuration of a Parse JSON action in SAP Integration Studio. The action is titled "Parse JSON" and has an input of type "JSON". The "Content" field contains a JSON object with fields like POEXPIMPHEADER, POHEADER, PO_NUMBER, COMP_CODE, DOC_TYPE, STATUS, and CREATE_DATE. The "Schema" field shows the corresponding JSON schema definition. Below the action, the "Outputs" section is expanded, showing an "Errors" list. This list contains three entries, each detailing an error found during schema validation. One entry is highlighted with a red box and a red arrow pointing to it, indicating the specific errors related to the question.

```
[{"message": "Invalid type. Expected Integer but got Number.", "lineNumber": 0, "linePosition": 0, "path": "POHISTORY[1].CL_VAL_LOC", "value": 4003.4, "schemaId": "#/properties/POHISTORY/items/properties/CL_VAL_LOC"}, {"message": "Invalid type. Expected Integer but got Number.", "lineNumber": 0, "linePosition": 0, "path": "POHISTORY[1].IVVAL_LOC", "value": 4003.4, "schemaId": "#/properties/POHISTORY/items/properties/IVVAL_LOC"}, {"message": "Invalid type. Expected Integer but got Number.", "lineNumber": 0, "linePosition": 0, "path": "POHISTORY[1].IVVAL_FOR", "value": 4003.4, "schemaId": "#/properties/POHISTORY/items/properties/IVVAL_FOR"}]
```

31. Update the Schema. We can update the schema to change from Integer to Number. Click on **Edit** in the upper right-hand corner to edit the flow again.



32. Click inside the Schema of the Parse JSON action.

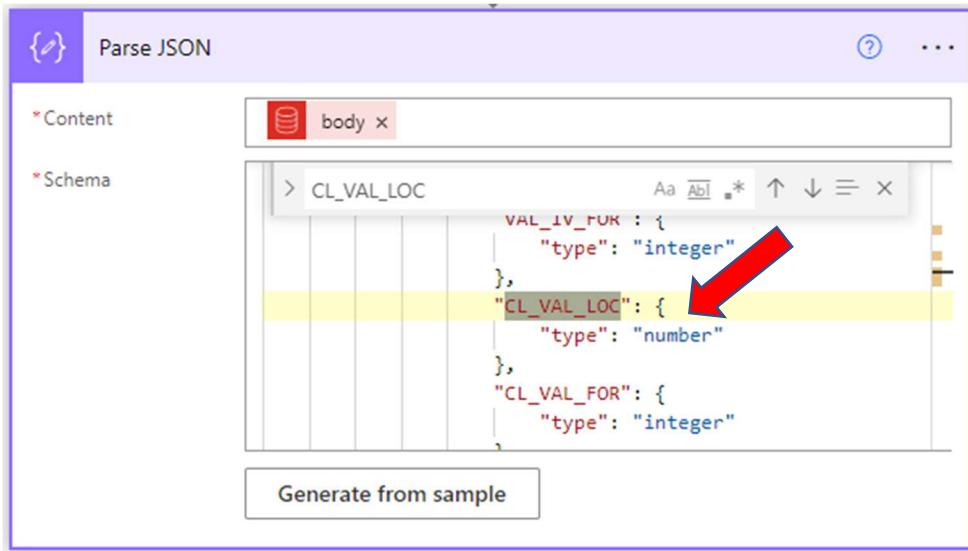
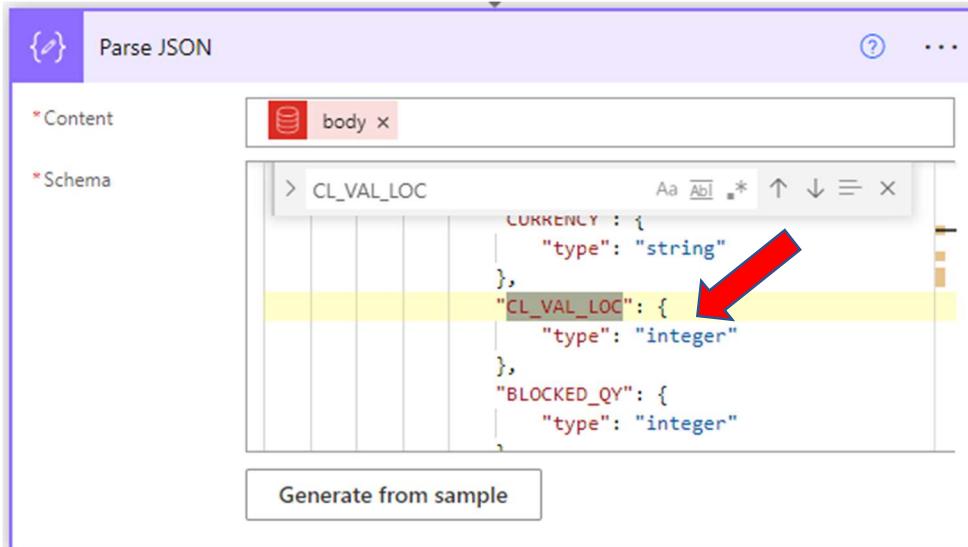


33. Find and replace the values that have incorrect data types. You have the Find and Replace functionality inside the schema. Hit **Ctrl-F** to find the values below and replace with the correct data type. Each value needs to be replaced in 2 spots. Use the down arrow in the search bar to find the next instance.

CL_VAL_LOC – Replace “integer” with “number”

IVVAL_LOC – Replace “integer” with “number”

IVVAL_FOR – Replace “integer” with “number”



34. Save the flow. Click on **Save**.
35. Re-Test the flow. Click on **Test** to retest the flow using the same steps above and select the recent run that was successful.
36. Review the Output of the flow. The flow should run successfully and take you to the Flow Run History. **Expand** the Parse JSON action by clicking in the box.



37. Examine the Body of the Outputs. You can scroll through the body of the outputs section and see all the data that was returned from SAP. In many cases, the field name does not give a good description of the data. For example, if you scroll down to POItem, you see the Short_Text field is actually the name of the item. We also see there is a lot of data returned that the user may not need. When we are delivering this data to Power Apps, we want to deliver data that is more user friendly and only the data they require. We will do that in the next exercise.

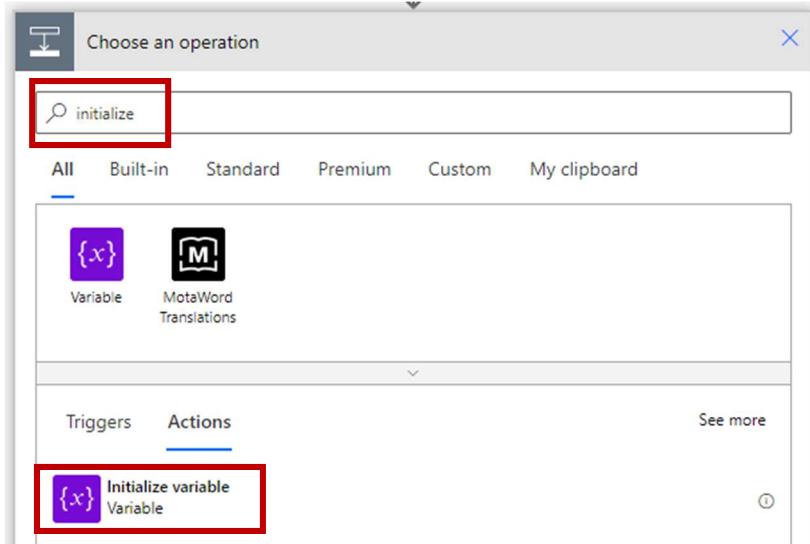
The screenshot shows the 'Outputs' section of the SAP Integration Developer Workshop. The 'Body' tab is selected, displaying a JSON-like structure of data. A red arrow points to the 'SHORT_TEXT' field, which contains the value 'HP Printer'. Other fields visible include 'PO_ITEM', 'DELETE_IND', 'MATERIAL', and 'MATERIAL_EXTERNAL'.

```
[{"POITEM": [{"PO_ITEM": "00001", "DELETE_IND": "", "SHORT_TEXT": "HP Printer", "MATERIAL": "OFFICE SUPPLIES", "MATERIAL_EXTERNAL": ""}]}]
```

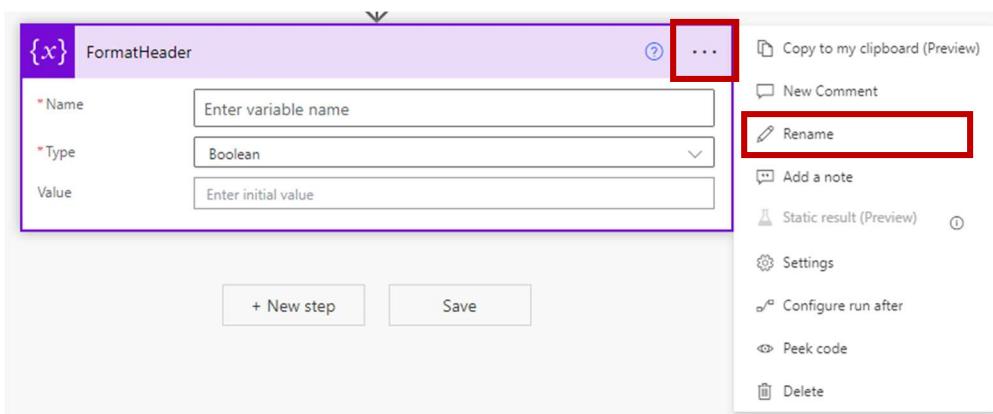
Exercise 4: Configure the flow to modify the PO data.

We will now create some variables to hold our data and then modify the data so it is ready to be consumed in a Power App.

1. Click on **Edit** in the upper right-hand corner.
2. Create a variable to hold the PO Header information. Click on **+ New Step** and then search for **Initialize** and select **Initialize variable**.



3. Rename the Action. Click on the ellipse (...) and select **Rename**. Rename the action to **FormatHeader**.



4. Configure the variable. Type in the following information for the variable:

Name: **vFormatHeader** (Best Practice - small v in front indicates variable)

Type: **Object**

Value: Enter the data below using dynamic content for the highlighted items. Type the highlighted text in search to get you to the correct area of Dynamic Content. Start and end the object with curly brackets. The name has double quotes, put a colon between the name and the dynamic content and a comma after each name/value pair except the last one.

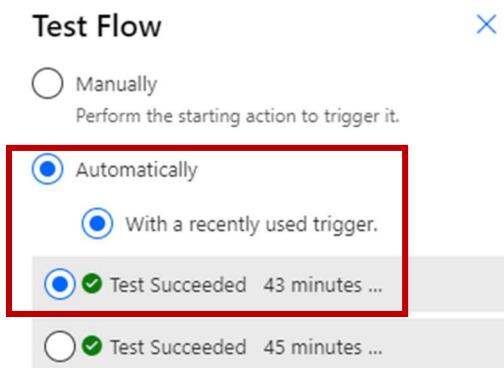
```
{  
    "OrderType": "POHEADER DOC_TYPE",  
    "PurchasingOrganization": "POHEADER PURCH_ORG",  
    "Vendor": "POHEADER VENDOR",  
    "CreatedBy": "POHEADER CREATED_BY",  
    "CreatedDate": "POHEADER_CREAT_DATE"  
}
```

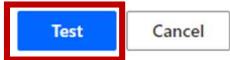
Make sure you pull the data from the SAP call which has the red object in front.

Value

```
{  
    "OrderType": "POHEADER DO... x",  
    "PurchasingOrganization": "POHEADER PU... x",  
    "Vendor": "POHEADER VE... x",  
    "CreatedBy": "POHEADER CR... x",  
    "CreatedDate": "POHEADER CR... x"  
}
```

5. Save the Flow.
6. Test the flow using the same method as before – Automatically with the most recent successful trigger.





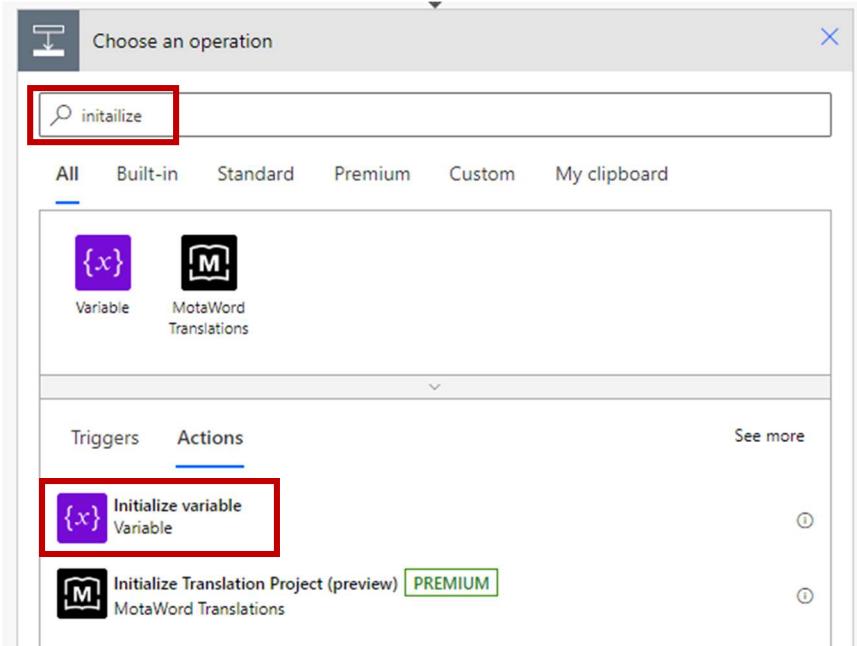
7. Review the data in the Value. Expand the FormatHeader action confirm the data coming back is valid.

```
{  
  "OrderType": "NB",  
  "PurchasingOrganization": "HLTH",  
  "Vendor": "000000088",  
  "CreatedBy": "JON",  
  "CreatedDate": "2022-08-19"  
}
```

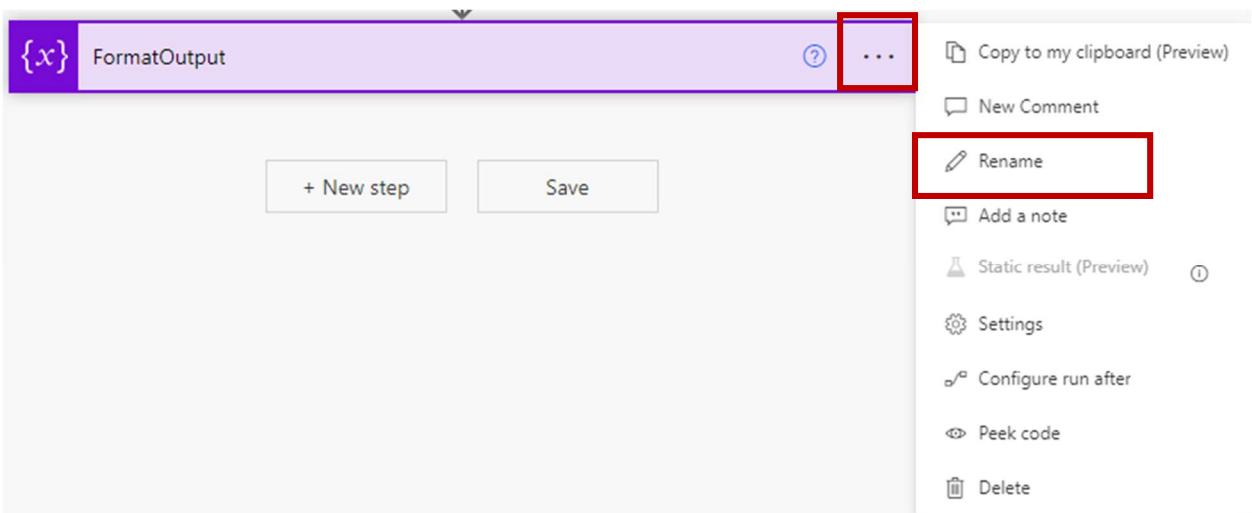
8. Click on Edit in the upper right-hand corner.



9. Next, we will create a variable for the PO Header information that we can use to verify the data is being returned properly before we send it to Power Apps. Click on + New Step and search for Initialize and select Initialize Variable action.



10. Rename the Action. Click on the ellipse (...) and select Rename. Rename the action to FormatOutput.



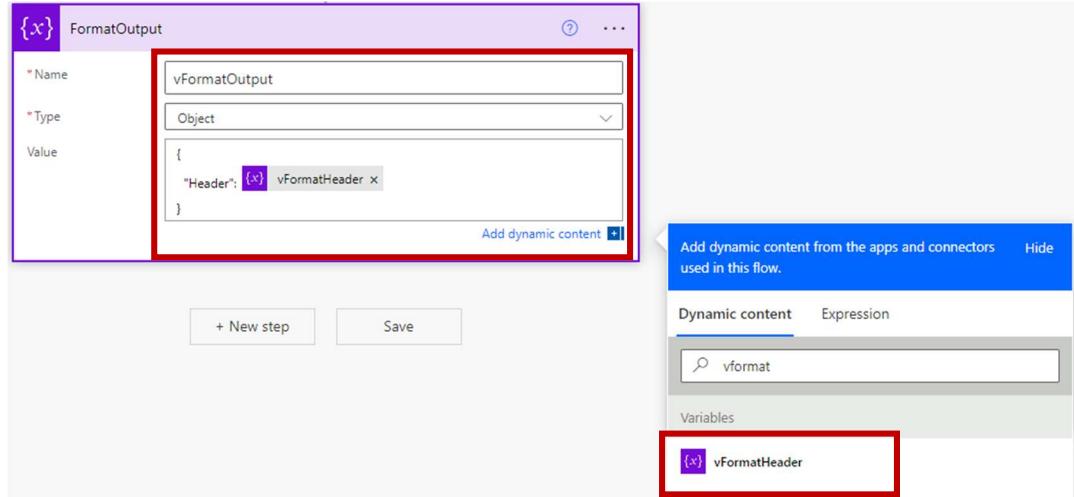
11. Configure Action. Type in the following information for the action.

Name: vFormatOutput

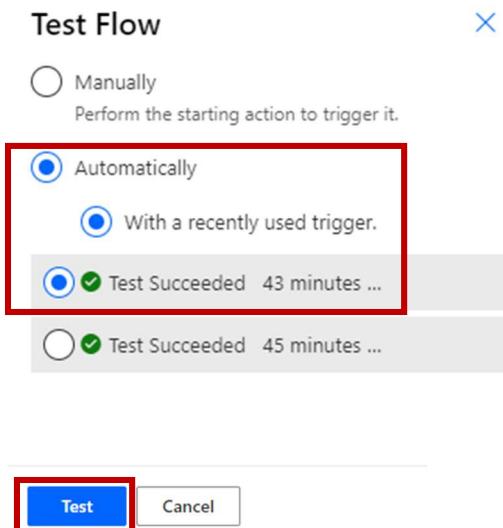
Type: Object

Value: Type in the information below with the highlighted text being selected from Dynamic Content.

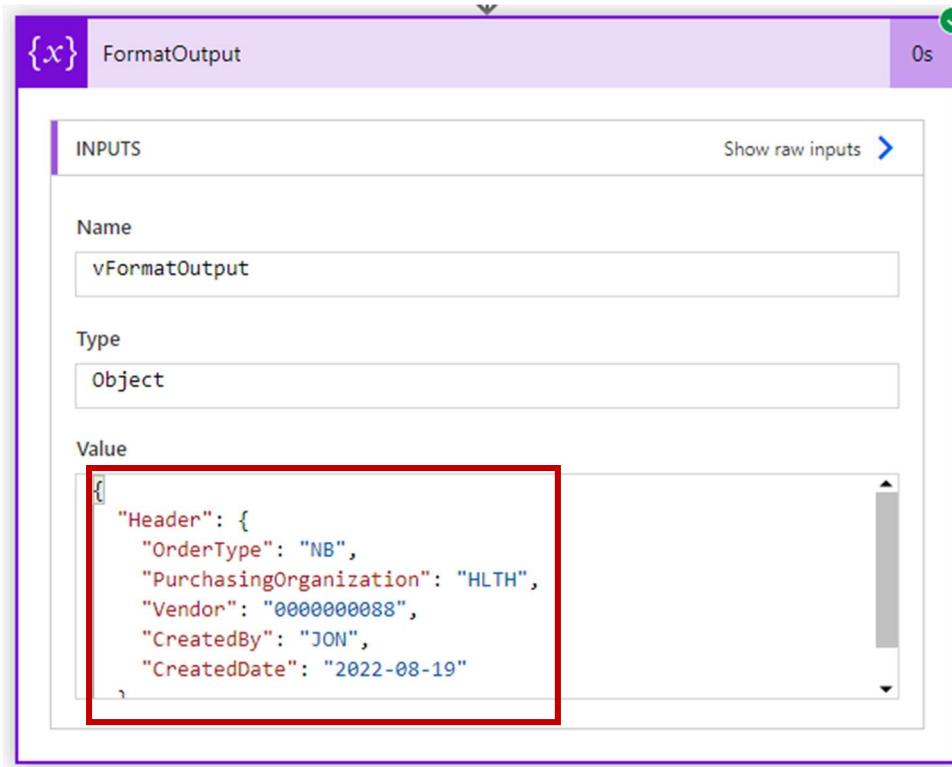
```
{
  "Header": vFormatHeader
}
```



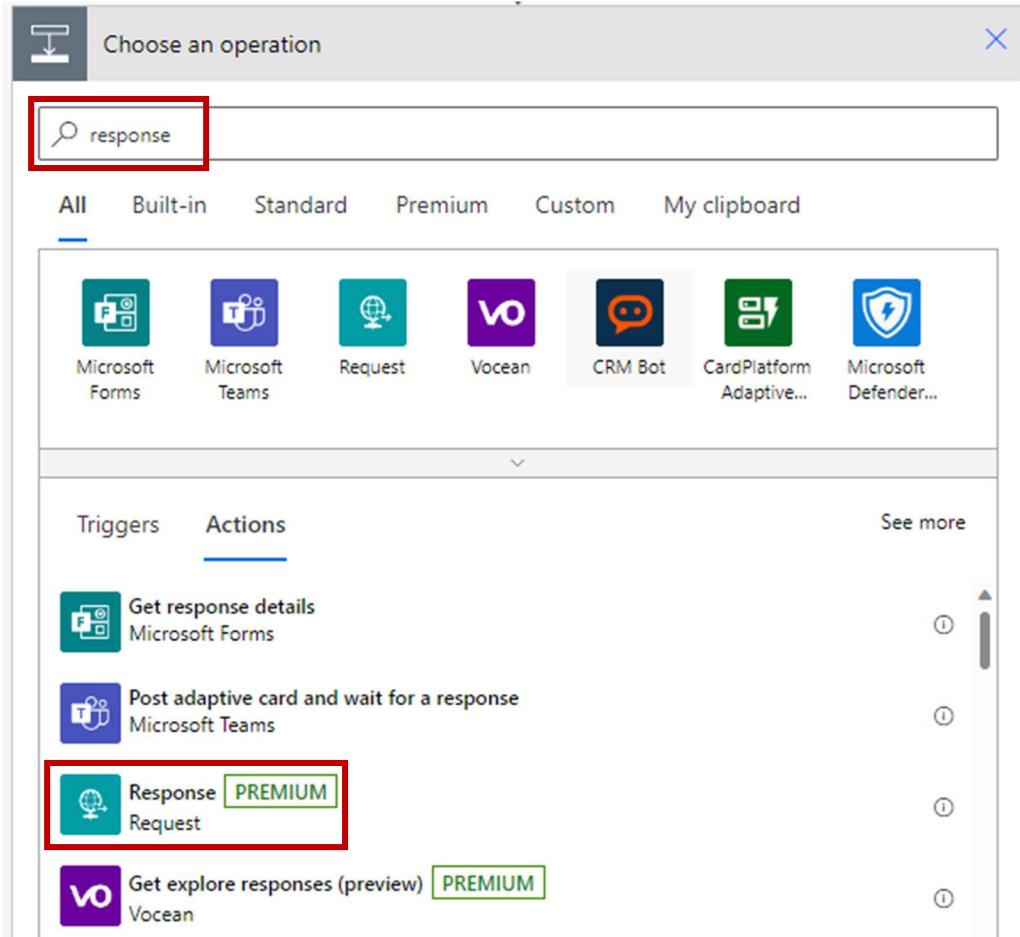
12. Save and then test the flow using the same method as before – Automatically with the most recent successful trigger.



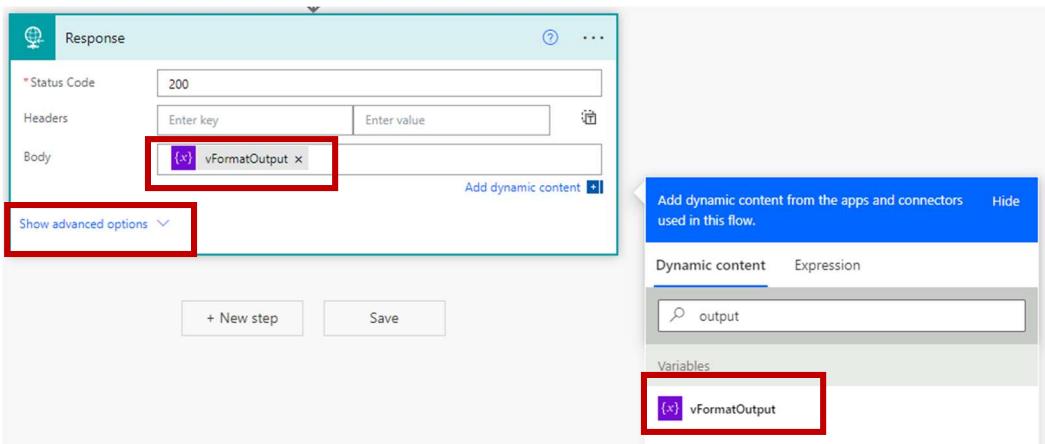
13. Expand the FormatOutput action and type **Ctrl-A** to select all the data and then **Ctrl-C** to copy the data from the **Value** field.



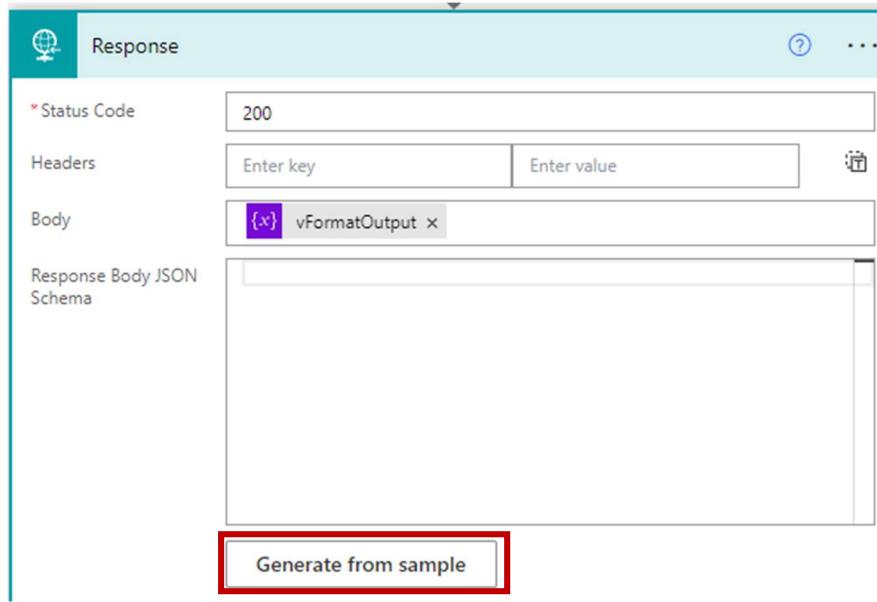
14. Add a new action to build out the data format to be returned to Power Apps . Click on **Edit** and **+ New Step**. Search for **Response** and select **Response (Request)**.



15. Configure the Response. For the **Body**, search for output in Dynamic Content and select **vFormatOutput** and then click on **Show advanced options**.



16. After clicking on Show advanced options, click on **Generate from Sample** and paste what we just copied - the data from the value of the FormatOutput - into the Response Body JSON Schema. Click on **Done**.



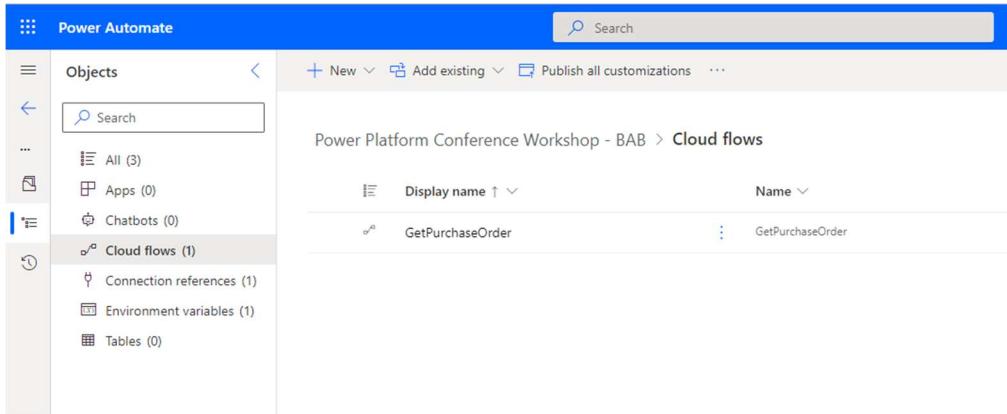
17. Click on **Save**.

We have completed a flow that will return a set of fields from the Header PO Data that we can call from Power Apps.

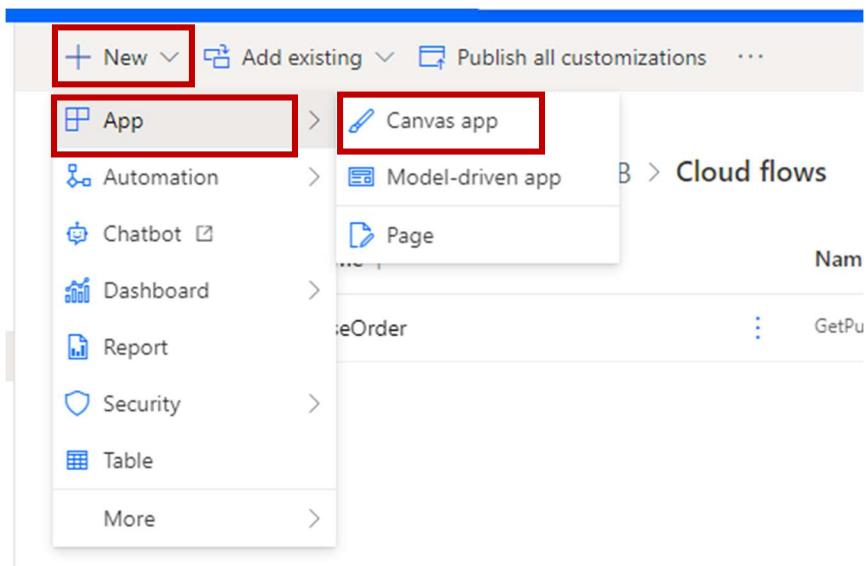
Exercise 5: Create a Power App to Display PO data

We are now ready to create our Power App that can consume the PO Data from SAP. We want to create the Power App inside our existing solution.

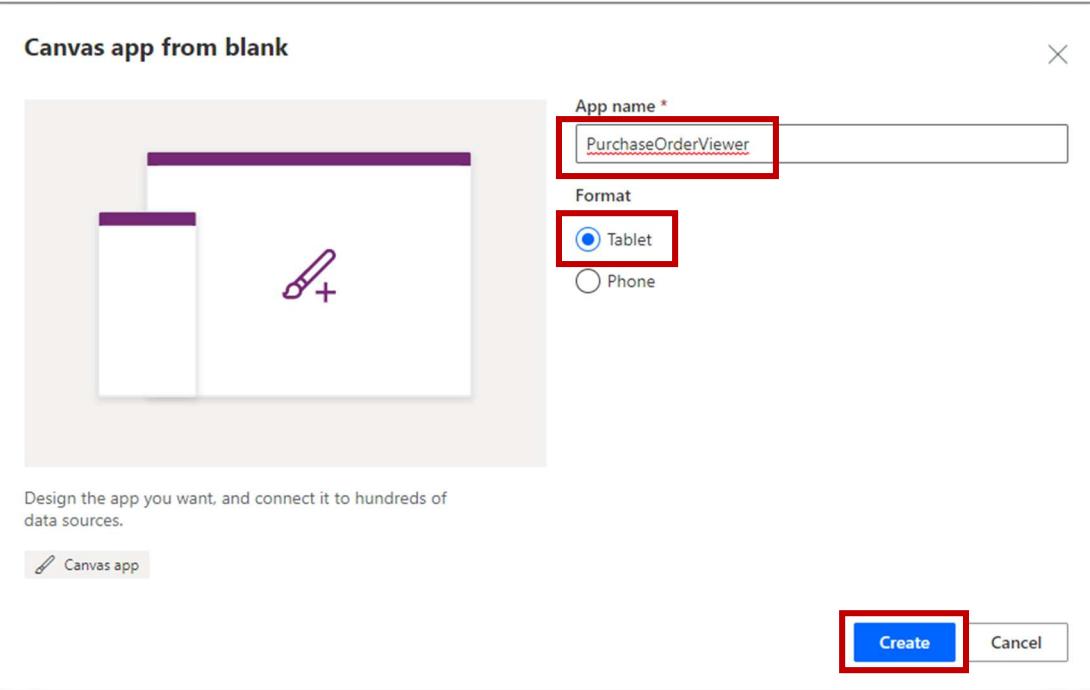
1. Click on back arrow to close your flow and stay in your current solution. Or if you are just picking up the labs again, browse to <https://make.power automate.com> and login with your workshop credentials, choose the correct environment and then click on Solutions. Edit the SAP Integration (Public Preview) solution.



2. Create a Canvas App. Click on + New, App, then Canvas App.



3. Type in **PurchaseOrderViewer** for the App Name, keep **Tablet** for the format and then click on **Create**.



4. Click on **Skip** if you get the Welcome to Power Apps Studio message.

Welcome to Power Apps Studio

Here are a few ways to start building an app from a blank canvas.

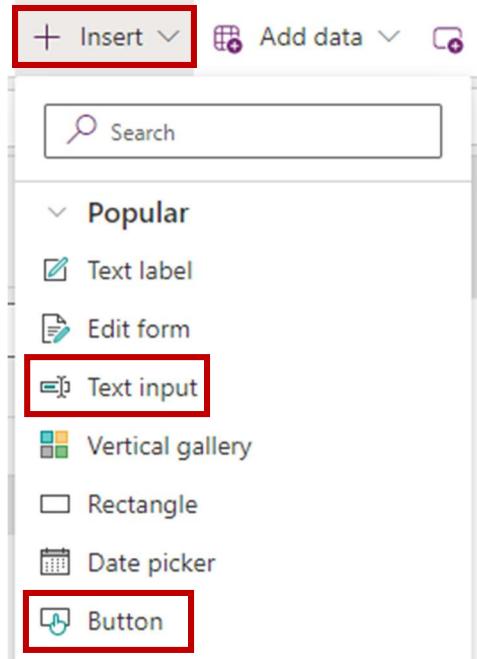
[Create a form >](#)

[Create a gallery >](#)

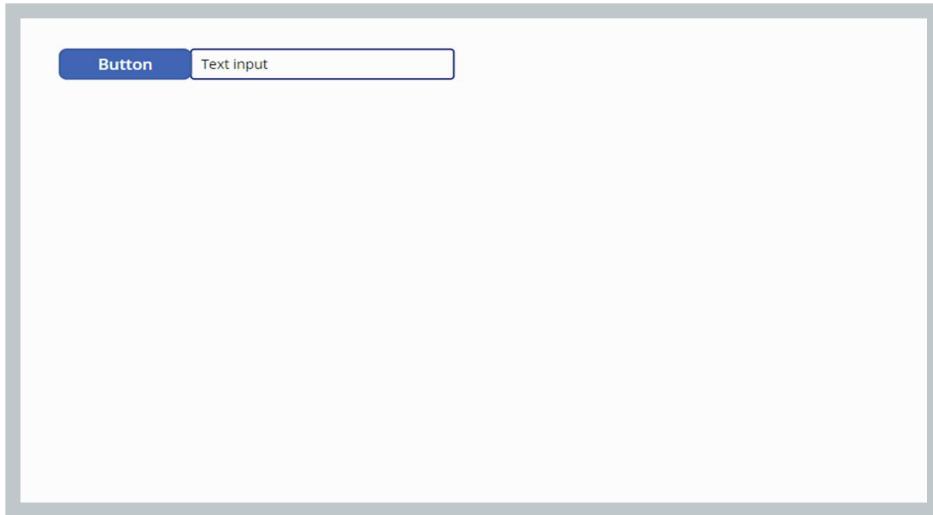
Don't show me this again

Skip

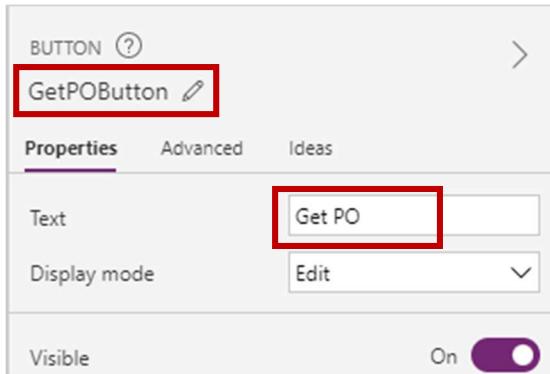
5. We need to create a Button and Text Input for initiating the flow based on a PO entered by the user. In the upper ribbon, click on **Insert** and then **Button**. Click on **Insert** again and select the **Text Input**.



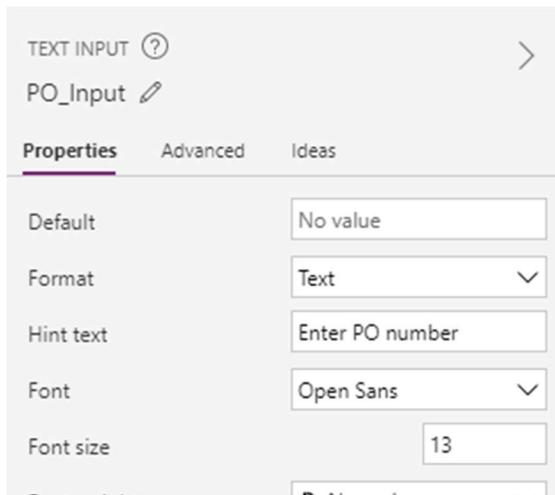
6. Drag and Drop the button and the Text Input so they are lined up horizontally in the upper left-hand corner of the screen to match the screen shot below.



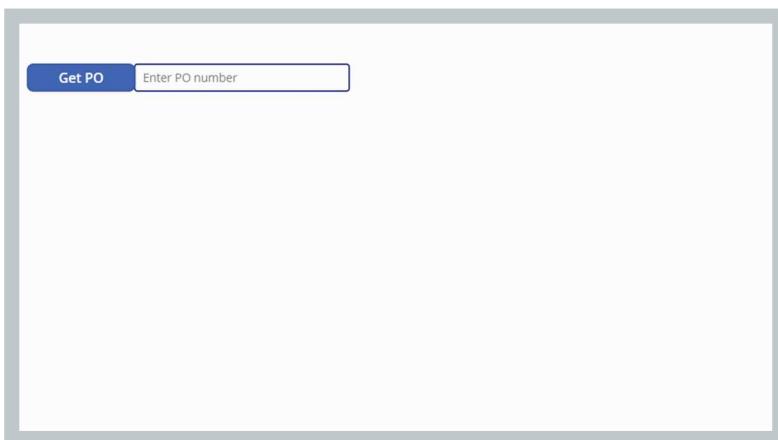
7. Change properties of the Button. Click on the Button on the canvas to select it. On the right-hand side, mouse over the name Button1 and then click on the pencil icon to rename to GetPOButton. In the Text Box, type Get PO.



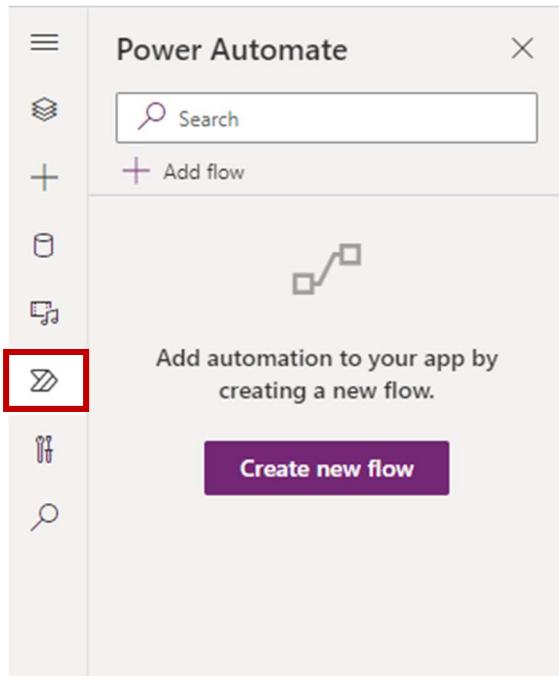
8. Change properties of the Text Input. Click on the Text Input box on the canvas to select it. On the right-hand side, click on the pencil icon to rename to PO_Input. Delete the value in the Default. Type Enter PO number in the Hint text.



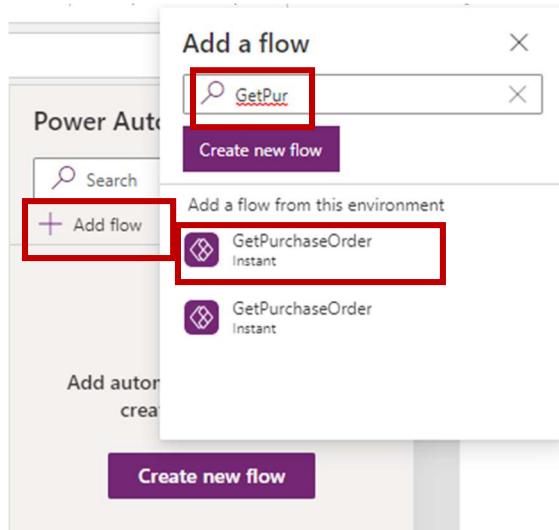
Your canvas should now look like the screen shot below.



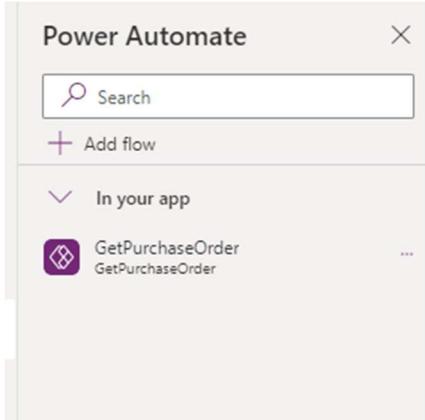
9. Connect the Flow we created in the last exercises to this Power App. On the left-hand tree view, click on the Power Automate icon.



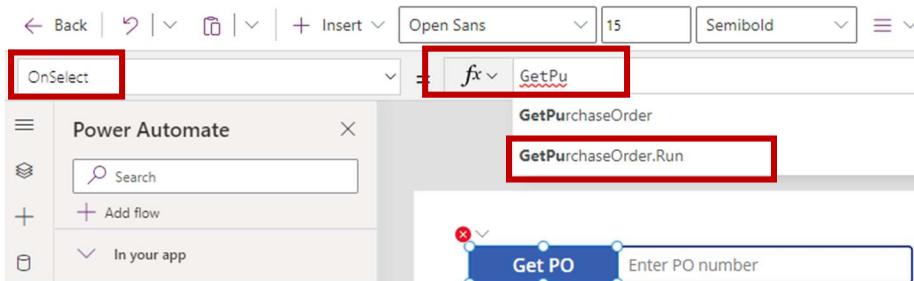
10. Click on + Add Flow. Search for GetPurchase and then select GetPurchaseOrder flow.



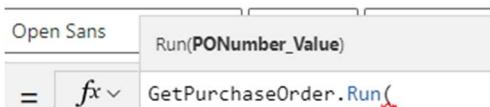
The flow is now in your application and is ready to use.



11. Configure the button to call the flow with the text input. On the canvas, click on the button to select it. In the upper ribbon, the **OnSelect** property is selected. **Delete False** from the **Formula bar** and start typing **GetPurchase**. You will see **Intellisense** pop up the Run Command.



Select that command. You will see information above the formula bar to guide you on what variables the flow is expecting.



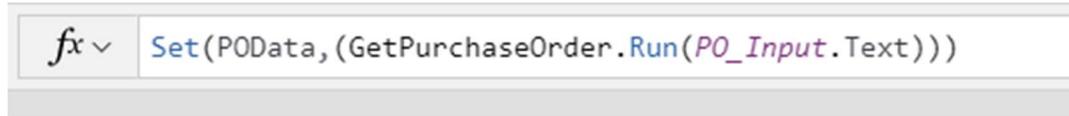
We want to pass the PO number that was typed in by the user in the Text Input box, so type **PO_Input.Text** as the variable and then **close the parenthesis**.

You will see Intellisense helping out again as you start typing.



12. Create a variable to store the PO information. We will be modifying the OnSelect formula for the button. Click on the Home key to bring the cursor back to the front of the formula. Hit the space bar to unselect GetPurchaseOrder. Type in **Set(POData**, in

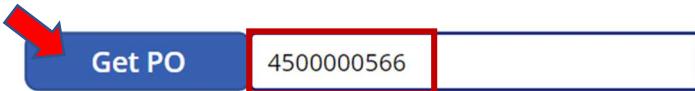
the front of your formula and then a **final close parenthesis** at the end. Final formula should look like the screen shot below. This will create a variable called POData to store all the information that comes back from our flow.



13. Test the App. Click on the Play button in the upper right of the ribbon.



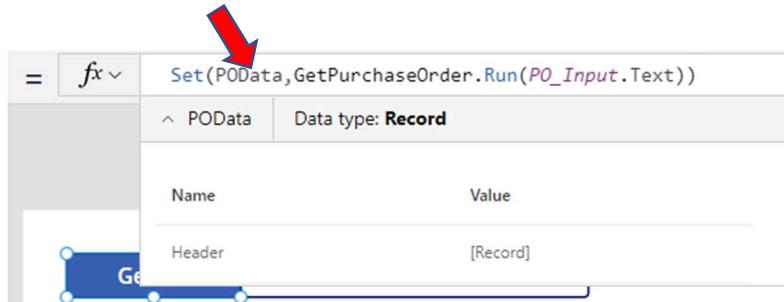
14. Type our sample PO number into the text box – 4500000566 – and click on the Get PO Button.



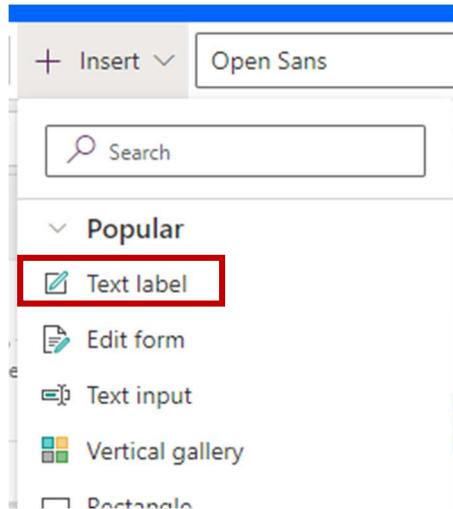
Click the X in the upper right-hand corner to close the app.



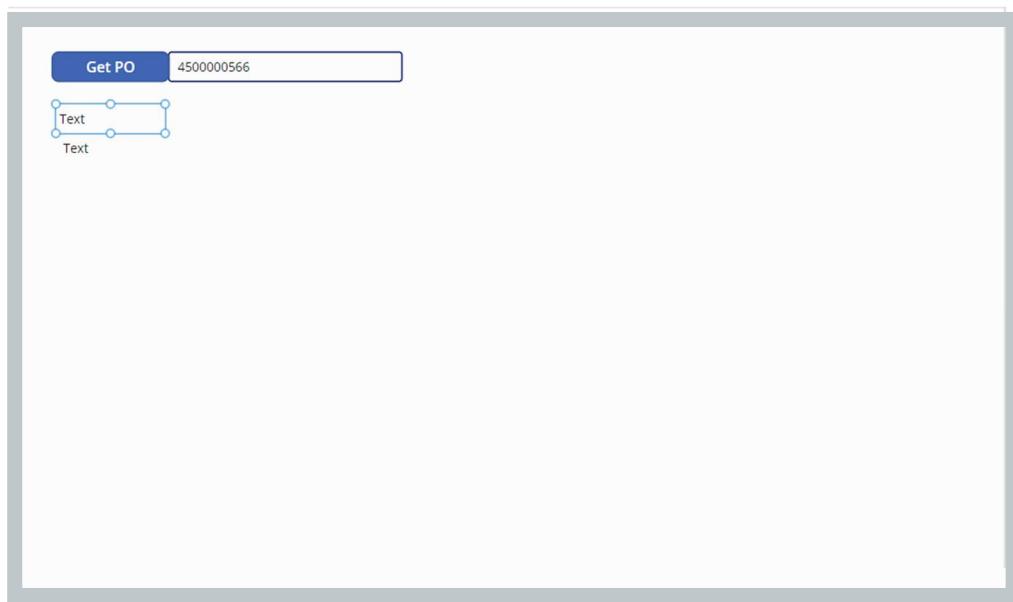
15. Verify there is data in our PODATA Variable. Click on the GetPO Button to select and then double click on the **POData** variable in our formula bar. We see that we now have a record associated called Header.



16. We will now modify the app to create places for the SAP PO Data to show up in the app. Click **Insert** and select **Text label**. Repeat this so there are (2) new Text labels on the canvas.



17. Configure the Text Labels. Drag and drop the labels so they are lined up vertically on the left-hand side of the canvas under the GetPO Button. Your canvas should look like the screen shot below.



18. Configure the Text Labels. We will make some property changes to the text labels to beautify the app. Select the top Text Label and make the following changes to the properties.

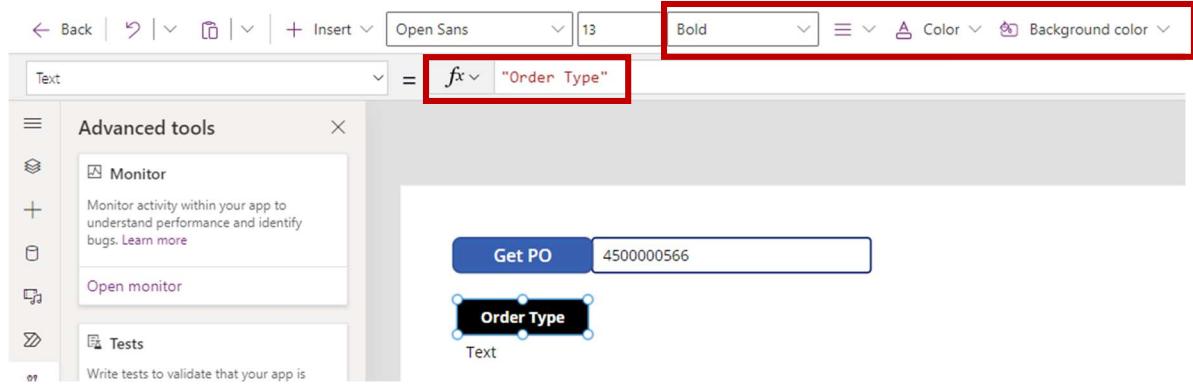
Text – “Order Type”

Font Weight- Bold

Alignment – Center

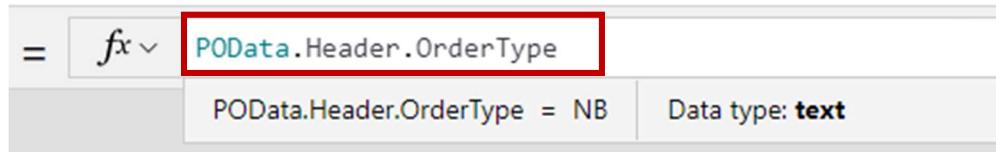
Font Color – White

Background Color – Black



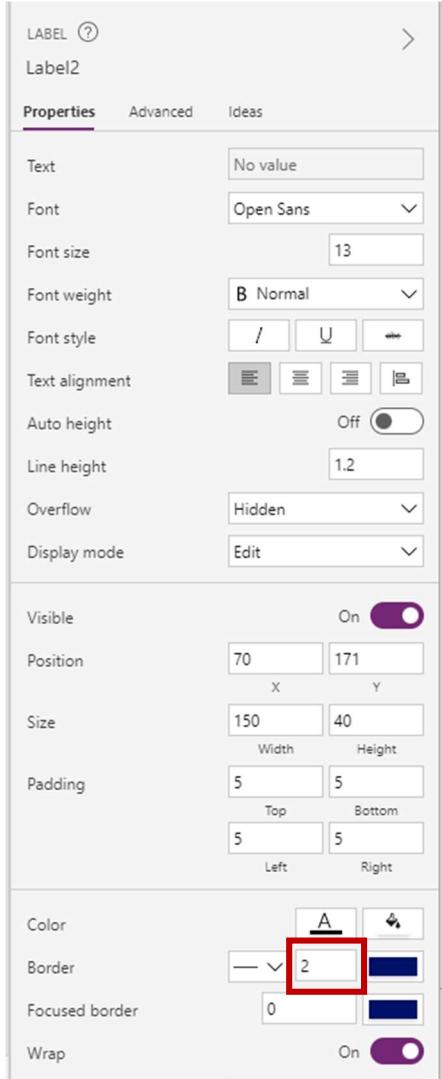
19. Select the bottom Text label and make the following changes to the properties.

Text – `POData.Header.OrderType`

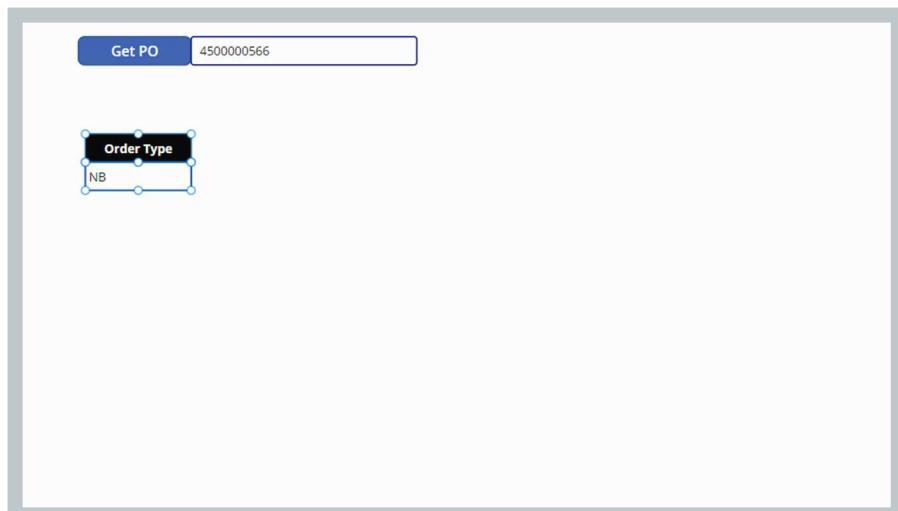


NOTE: This is our variable data from SAP. When you click on the period after Header, Intellisense will list the other data that is being brought back from SAP. You will also see the value of the data – “NB”.

In the right-hand properties, change **Border Thickness** to 2.



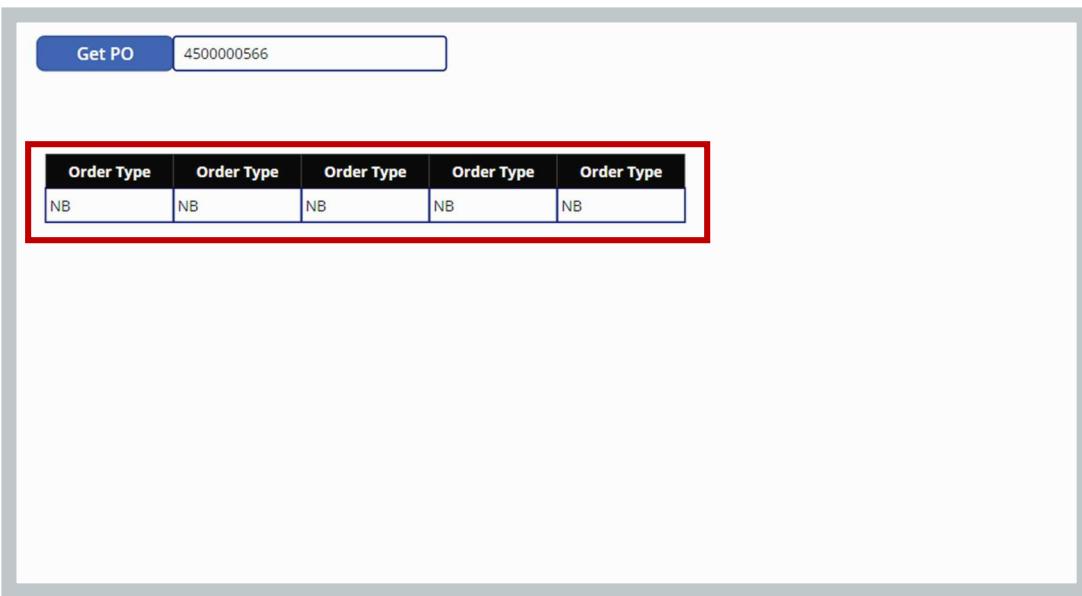
Once you have the border around the 2nd Text Label, line them up using the grid lines. Your canvas should now look like the screen shot below.



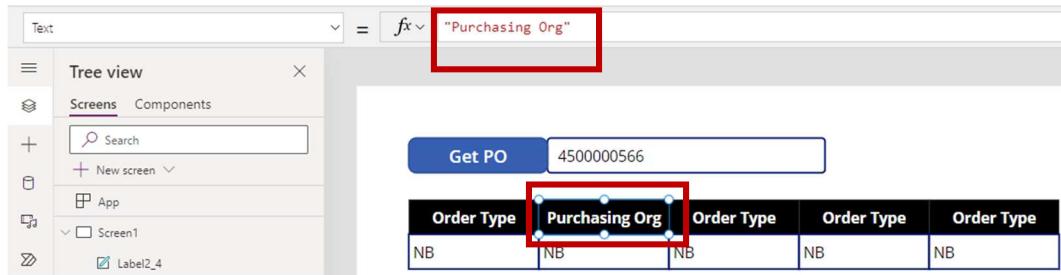
20. Create 4 more columns of Text Labels for the rest of the date. Select **both** Text Labels and **Ctrl-C** to copy and then **Ctrl-V** to paste. Once you have copied the pair of labels, move them as a group so they line up horizontally on the canvas.



21. Type **Ctrl-V 3 more times** so you have 5 sets of labels. Move the labels as pairs to the canvas looks like the screen shot below.



22. Change the data of the columns to pull in the other PO Header data. Now we need to change that data for the other columns to reflect the other data fields coming back from the PO Header in SAP. **Click** on the **2nd header label** to select it and change the **Text to Purchasing Org**.



Follow the same steps for the other column headers using the text below:

Column 3 – "Vendor"

Column 4 – "Created By"

Column 5 – "Created Date"

Your canvas should now look like the screen shot below.



23. Change the data fields. Currently all the data fields are set to OrderType, so we need to change that to reflect the correct data field for that column. Click on the Purchasing Org data field to select it. Change the text to `POData.Header.PurchasingOrganization`.

The screenshot shows the SAP Power Platform canvas interface. On the left, there's a tree view with 'Screens' selected, showing 'Search' and 'New screen'. In the center, there's a table with columns: Order Type, Purchasing Org, Vendor, Created By, and Created Date. The 'Purchasing Org' column has a red box around it. At the top right, there's a search bar with 'POData.Header.PurchasingOrganization' and a dropdown showing 'POData.Header.PurchasingOrganization = HLTH Data type: text'. Below the table is a button labeled 'Get PO' and a text input field with '4500000566'.

Follow the same steps to change the other data fields using the text below:

Vendor – `POData.Header.Vendor`

Created By – `POData.Header.CreatedBy`

Created Date – `POData.Header.CreatedDate`

Your canvas should now look like the screen shot below.

The screenshot shows the SAP Power Platform canvas interface with a gray border. Inside, there's a table with the following data:

Order Type	Purchasing Org	Vendor	Created By	Created Date
NB	HLTH	0000000088	JON	2022-08-19

A 'Get PO' button and a text input field with '4500000566' are also visible at the top.

24. Save the App. Click on the **Save** icon in the upper-right hand corner.



Exercise 6: Shape the data returned from SAP

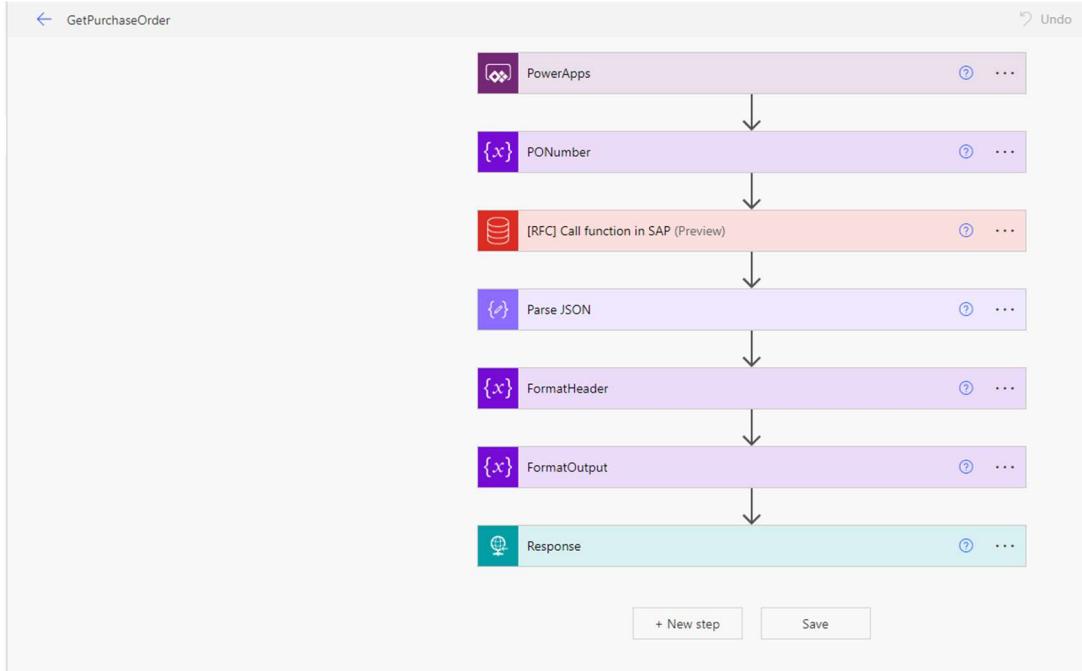
We are now going to shape the data being returned from SAP. Many times, the data is not in the format we need to use or there is too much data, and we need to only pass a limited set of data.

Below is a table of the most common Data Operations that you may be using to manipulate data from SAP.

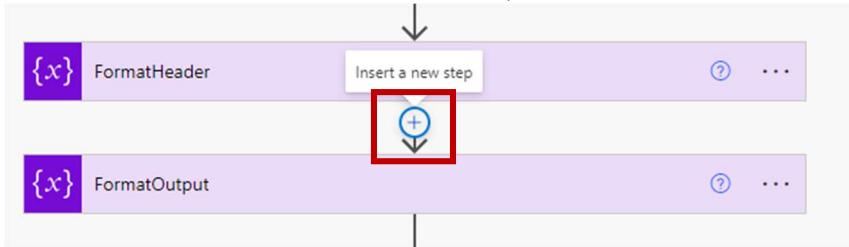
Action Name	Description
Compose	Use the Compose action when you need to enter data multiple times in a cloud flow – for example creating an array. Use with another operation like Join to access the contents.
Join	Use the Join action to delimit an array with the separator of your choice. For example, change an array separated by commas to be separated by semicolons.
Select	Use the Select action to transform the shaped of objects in an array. For example, you can add, remove, or rename elements in each object in an array.
Filter Array	Use the Filter array action to reduce the number of objects in an array to a subset that matches the criteria you provide.
Create CSV Table	Use the Create CSV table action to change a JSON array input into a comma-separated value (CSV) table.
Create HTML Table	Use the Create HTML table action to change a JSON array input into an HTML table.

For more detailed information on the Data Operations and specific examples, please refer to this document - [Use data operations in Power Automate \(contains video\) - Power Automate | Microsoft Docs](#)

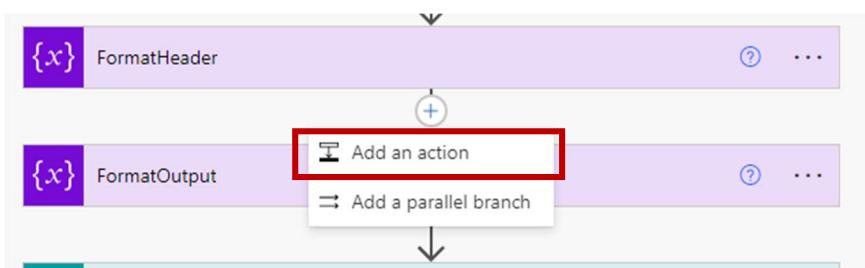
1. Pass back the PO line item details from SAP. We will now manipulate the line item details into an array so we can name the data to something that makes sense and only pass back the limited amount of data we need. Click back on the tab that is editing your Power Automate flow – **GetPurchaseOrder**. You may need to click Edit again.



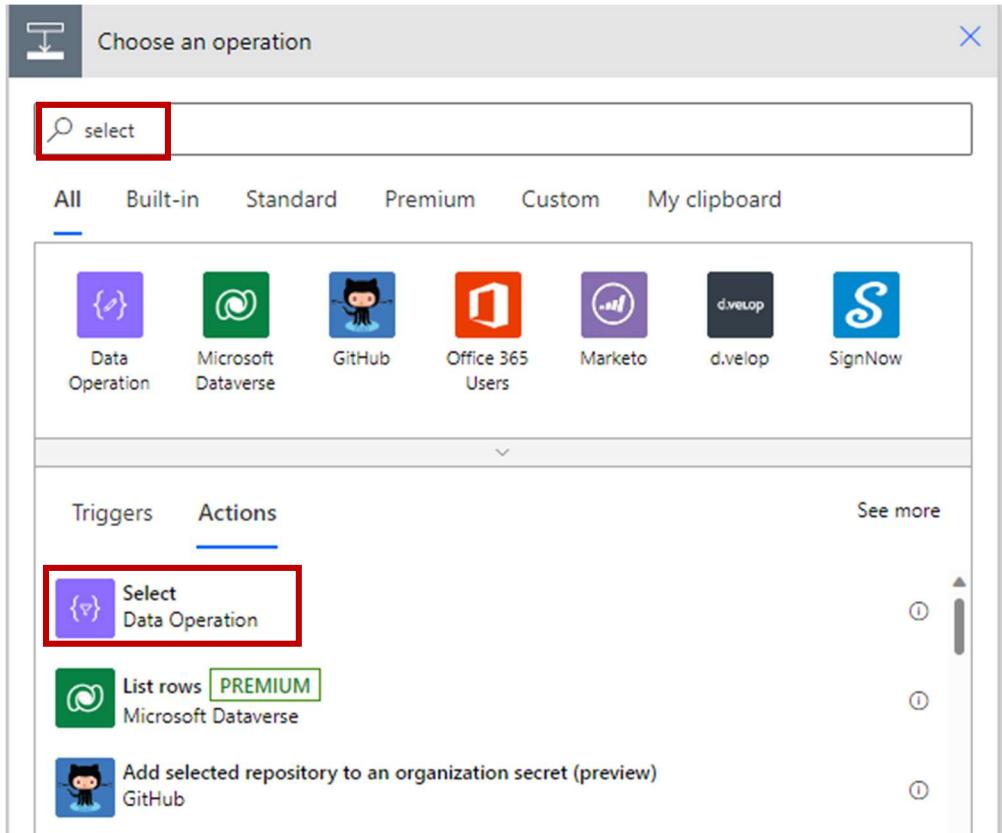
2. Add a new step after the FormatHeader action. Mouse over the arrow between the FormatHeader action and the FormatOutput action to add a new step.



3. Click on the + to add a new step and then Add an action.



4. We will use the Select action to format the elements of the array. Type in **Select** in the search box in Operations and select **Select – Data Operations** action.

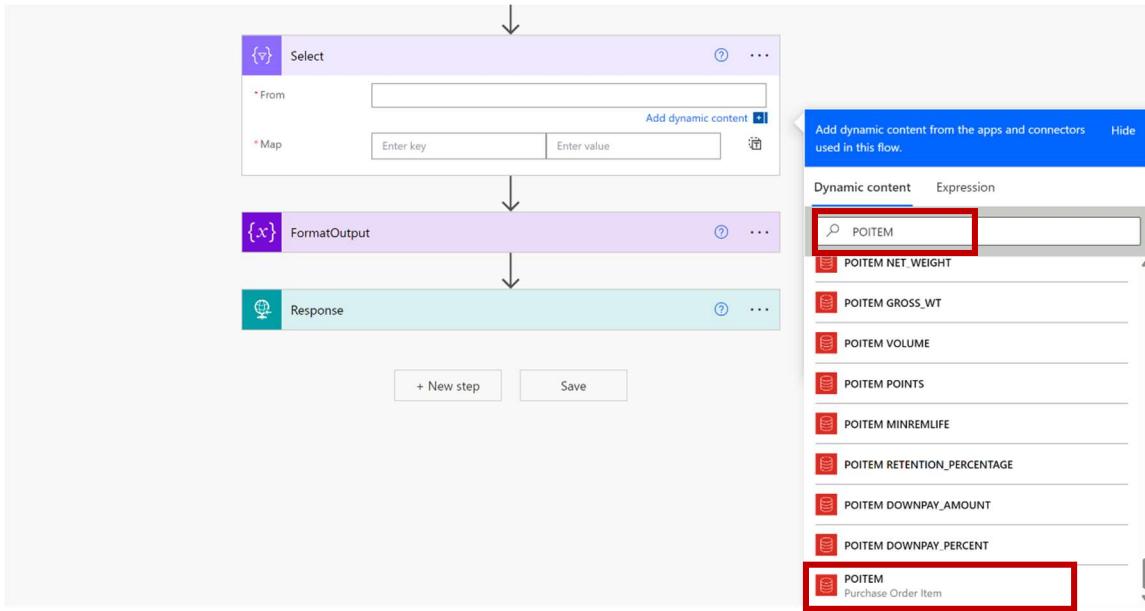


5. Rename the Select Action. Click on the ellipsis (...) and then **Rename**. Rename the action **FormatItems**.



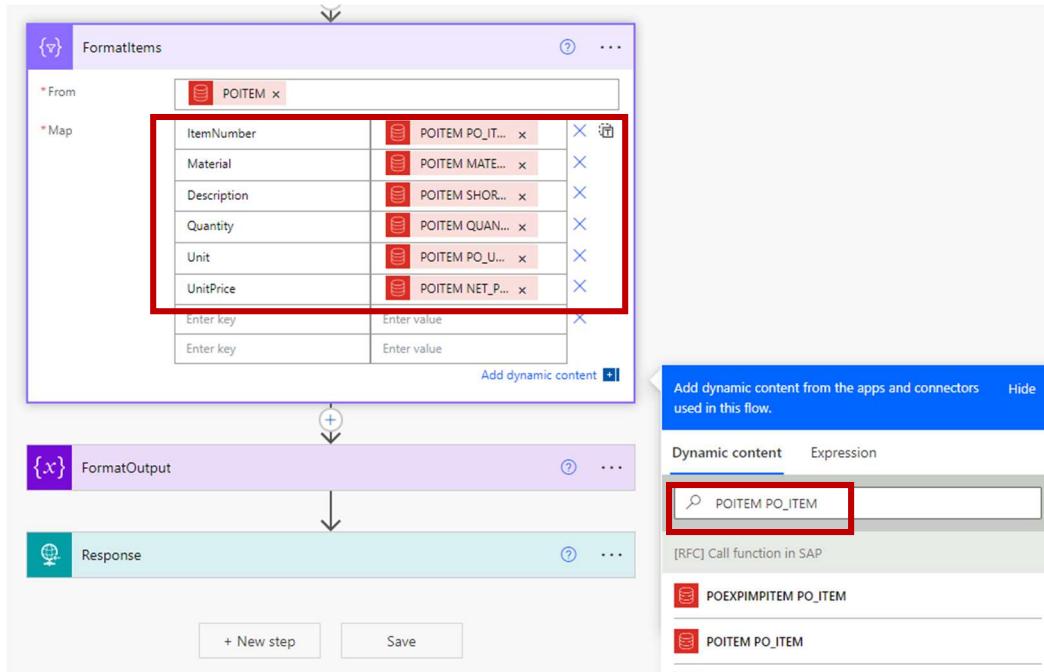
6. Configure the Select Action. Click in the **From** box. Search for POITEM in the Dynamic Content and select **POITEM (Purchase Order Item)** from the [RFC] Call function in SAP section.

You will have to scroll all the way down to the bottom to get the correct PO ITEM (Purchase Order Item)



- We will now create a map to bring back only the data we need and with a data key that will be more recognizable in Power Apps. Click in the **Map** section and enter the following **Key** and **Value** pairs for the map using Dynamic Content for the Value. Use search to find the values more easily and pull from the [RFC] Call function SAP section.

Key	Value
ItemNumber	POITEM PO_ITEM
Material	POITEM MATERIAL
Description	POITEM SHORT_TEXT
Quantity	POITEM QUANTITY
Unit	POITEM PO_UNIT
UnitPrice	POITEM NET_PRICE



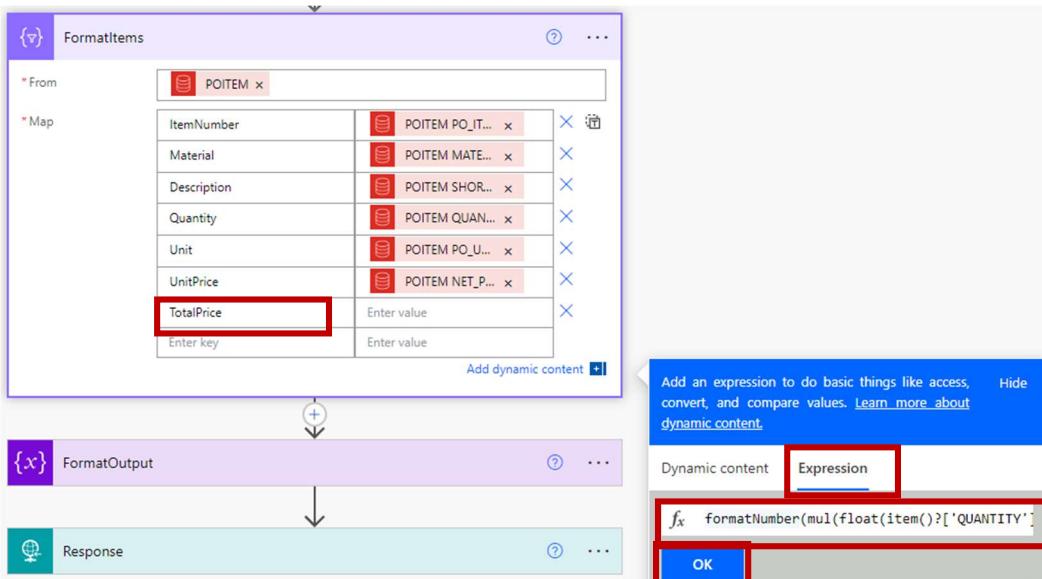
8. Create a calculated field for the Total Price field. We will now use an Expression to create a calculated field that adds up total of this line item. We will calculate this by multiplying the NetPrice by the Quantity. Since SAP values are coming back as Text values, we will need to convert them to numeric values.

Back in our Map Table, type in **TotalPrice** in the **Key Field**. Click in the **Value field** and select **Expression** from the Dynamic Content. **Paste** the formula below into the formula bar and then click on **Okay**.

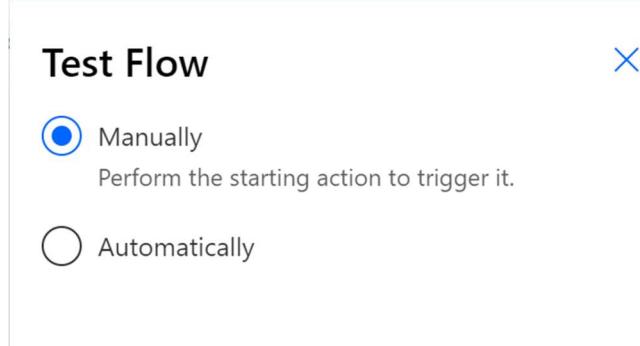
```
formatNumber(mul(float(item()?'QUANTITY'),float(item()?'NET_PRICE'))), 'F2')
```

This expression multiples Quantity by NetPrice. Float converts the string to a number and FormatNumber at the beginning of the formula and F2 at the end of the formula format

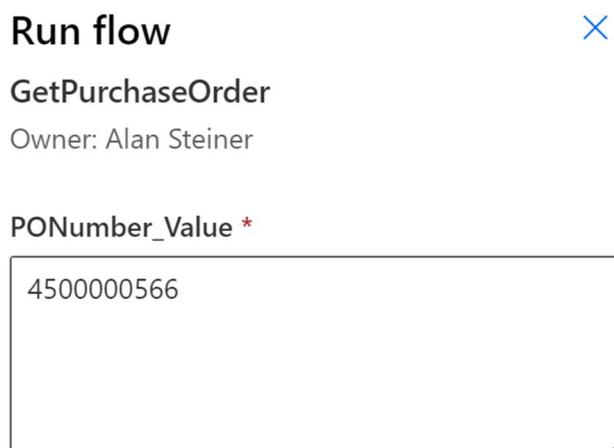
the TotalPrice to have 2 decimals.



9. Save the Flow and Test with a Manual Trigger.



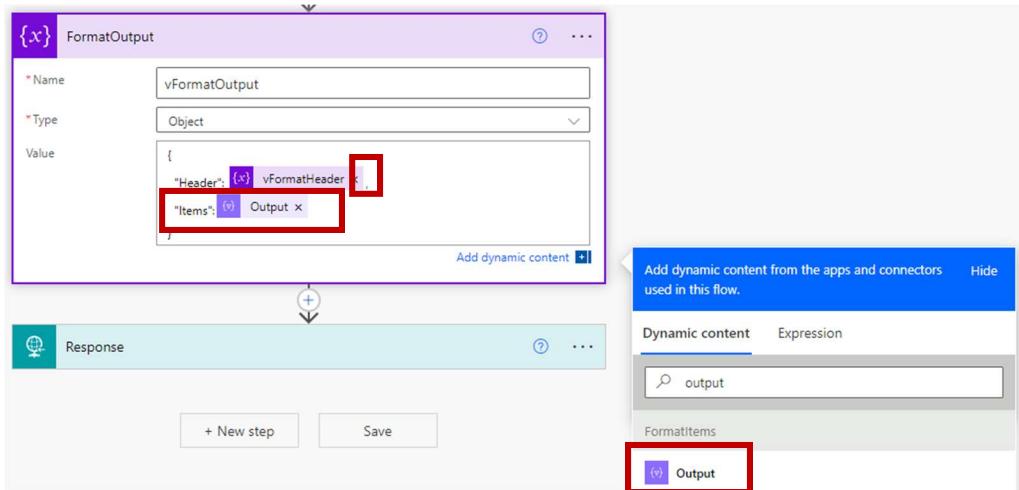
Enter 4500000566 for the PO number.



10. Verify the data is being pulled back properly and our calculated field is correct.
Expand the **FormatItems** action and scroll through the Outputs section.

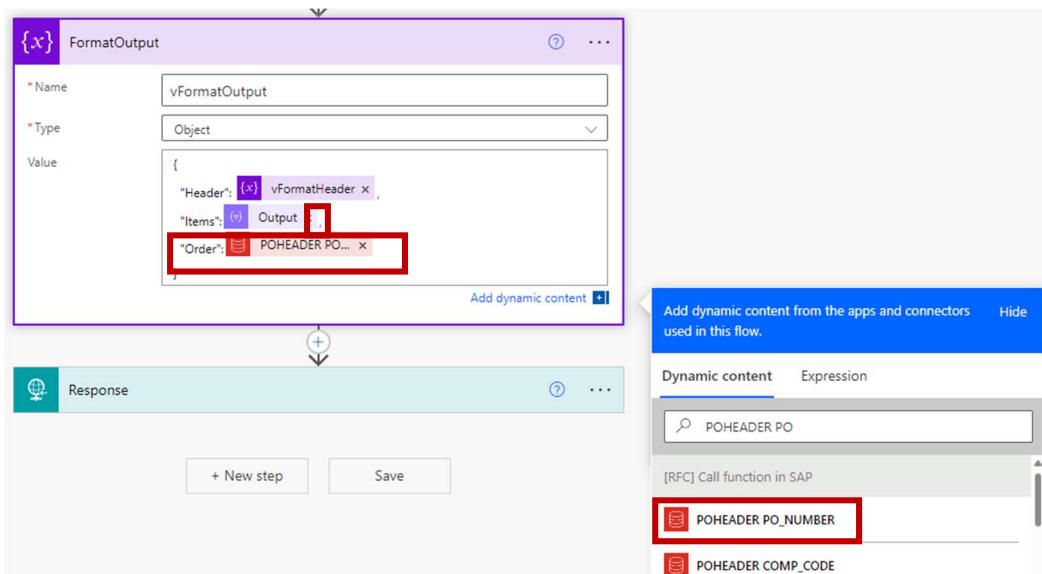
The screenshot shows the SAP Integration Studio interface with the 'FormatItems' action selected. The 'From' input pane displays a JSON object with several fields. The 'Body' output pane displays a JSON object with several fields, some of which are highlighted with a red box. The highlighted fields are: "ItemNumber": "00001", "Material": "OFFICE SUPPLIES", "Description": "HP Printer", "Quantity": 20, "Unit": "EA", "UnitPrice": "200.17000000", and "TotalPrice": "4003.40".

11. Update the schema for the **FormatOutput** section to include the **Format Items** data.
Click on **Edit**. Expand the **FormatOutput** action. In the **value** section, make the following changes.
- ❑ Add a **comma** behind the name/value pair and then enter
 - ❑ On the next line type "**Items**": **Output** – where Output is from Dynamic Content under **FormatItems**



12. Add data directly from SAP in the FormatOutput Section. Add the PO Number directly from the SAP by adding a simple name/value pair. In the **Value** section, add a **comma** behind the **Items** name/value pair to add another set of data and hit **enter**. On the next line, add the following text:

"Order": POHEADER PO_NUMBER – where POHEADER PO_NUMBER is from Dynamic Content under [RFC] Call function in SAP.

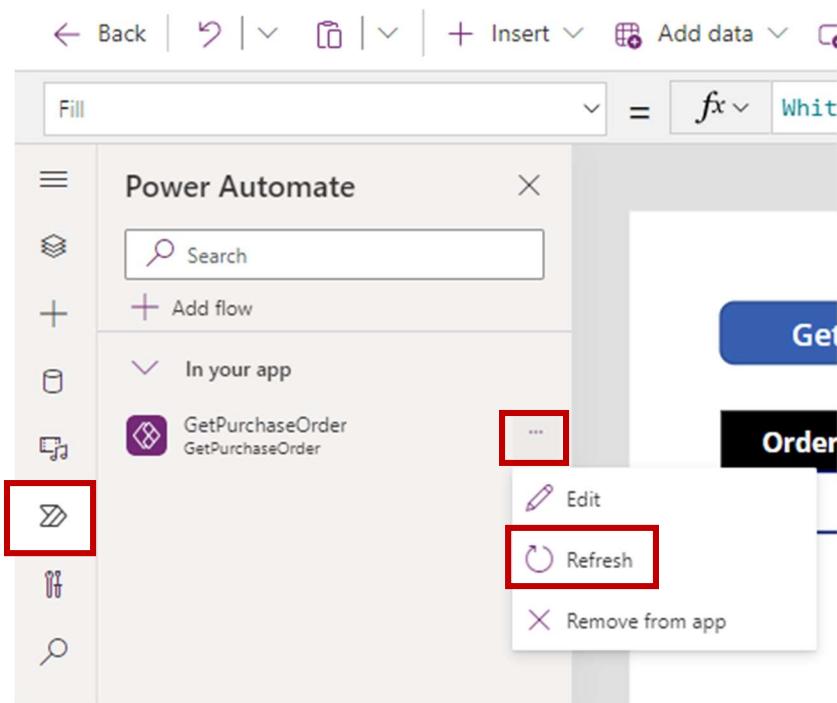


13. Update the Response Section. We now need to get the new schema from the FormatOutput action and generate from sample to create the new schema like we have done several times already.

- Save and Test the flow using an automatic trigger.
- Expand the FormatOutput action and copy the entire Value section.

- Edit the flow.
- Expand the Response action.
- Click on Advanced Options.
- Click on Generate from sample
- Paste the new schema
- Save the flow

14. Update the Power App to show our item list and PO number. Click back to your tab where you are editing your **Power App**. First, we need to update our Power Automate flow since we have made changes. Click on the **Power Automate** tab in the tree on the left-hand side. Click on the ellipse (...) and then Refresh.



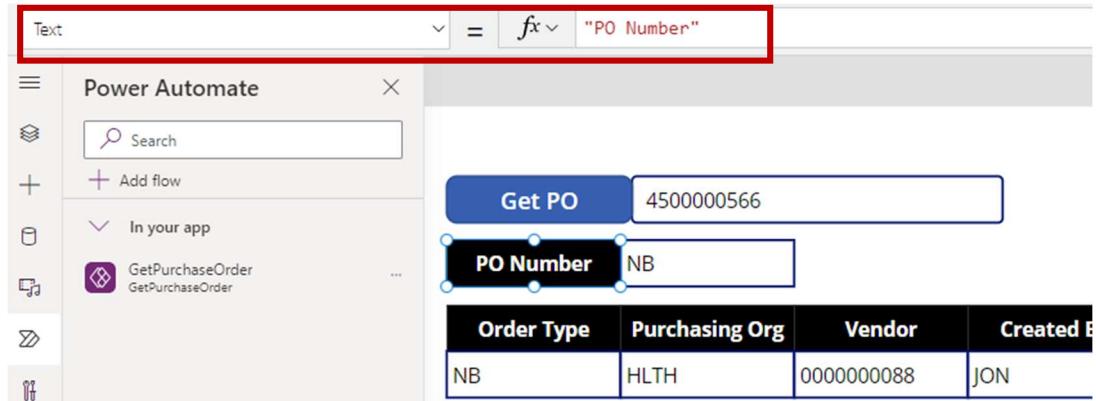
15. Play the app to load in the new data. Another way to play the app inside the canvas is to hold down the **Alt-key** and then click on the **GetPO Button**. This will cause the cloud flow to get the new line item and PO number values.

If needed, use PO # 4500000566

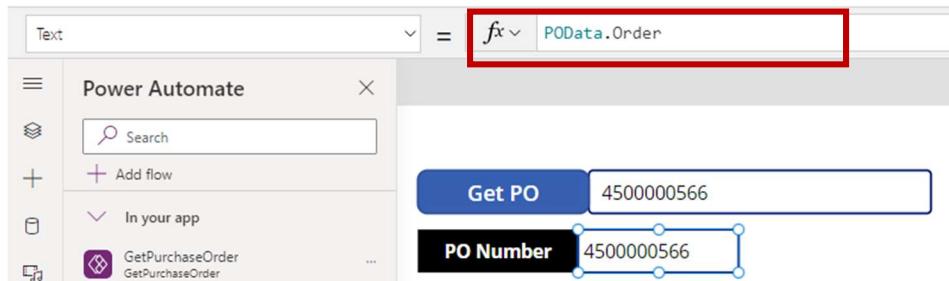
16. Add the PO Number to the App. We will use **Ctrl-C** to copy the **OrderType Text label** and **OrderType Value field** to make things easier. **Ctrl-V** to paste them on the canvas. **Drap and Drop** them above the POHeader information.



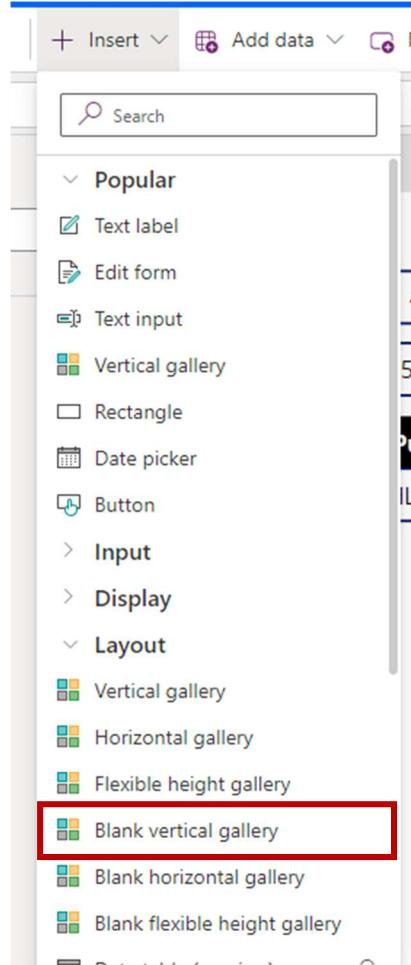
17. Update the fields to display the PO number data. Click on the new OrderType Header to select it. Update the Text to "PO Number"



Click on the new Text Order Type value field to select it. Change the Text formula to `POData.Order`. We don't need to have Header in the formula.



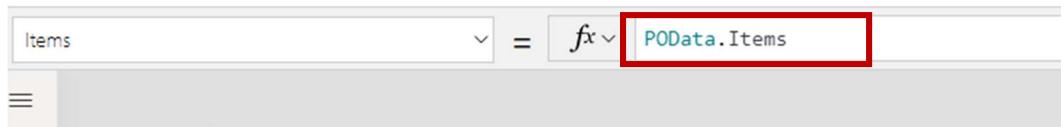
18. Insert the Line Items into the Canvas App. Click on **Insert** and select **Blank Vertical Gallery**. You may have to expand the Layout section to find it.



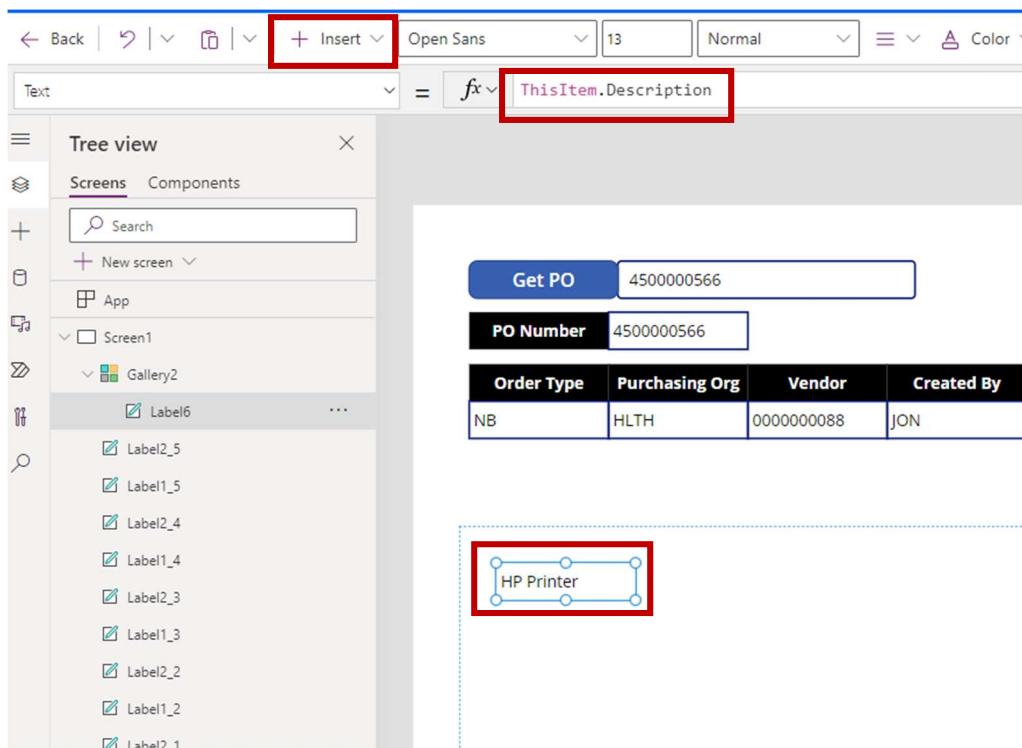
Drag and drop the gallery below your PO Header information and leave room for column labels above.

A screenshot of the Microsoft Power Apps canvas. At the top, there is a header with 'Get PO' and a value '4500000566'. Below it is a table with columns: Order Type, Purchasing Org, Vendor, Created By, and Created Date. The table has one row with values: NB, HLTH, 000000088, JON, and 2022-08-19. Below the table is a large rectangular area with a blue border and rounded corners. Inside this area, there is a small circular icon with a dot and a horizontal line through it, followed by the text 'Add an item from the Insert pane or connect to data'.

19. Change the Items formula to point to the **Line Items** data by changing the text to **POData.Items**



20. Configure the Gallery. We need to add the fields we want to display in the gallery. With the Gallery still selected, click on **Insert** and then **Text Label**. Power Apps will bring in what it thinks you may want for the first item. It brought in the Description field.



Drop and drag to the middle of the gallery.

Get PO 4500000566

PO Number	4500000566			
Order Type	Purchasing Org	Vendor	Created By	Created Date
NB	HLTH	0000000088	JON	2022-08-19

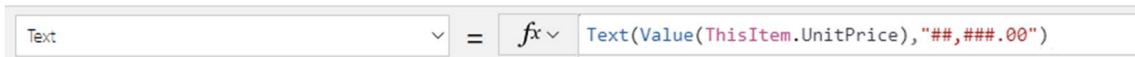


21. Repeat the previous step until you have brought in all the data fields listed in the table below and have the columns in the same order of the screen shot below.
If the short cut does not bring in all the fields for you like Quantity, select the Text Label and then set the Text formula to match the table below. You can reduce the size of the Quantity and Unit fields.

Item	Text Formula
Item Number	ThisItem.ItemNumber
Quantity	ThisItem.Quantity
Unit	ThisItem.Unit
Description	ThisItem.Description
Material	ThisItem.Material
Unit Price	ThisItem.UnitPrice
Total Price	ThisItem.TotalPrice

22. Format the Unit Price and Total Price fields. Change the format of the unit price field to only have 2 decimals. Select the **Unit Price field** and then change the **Text formula** to:

```
Text(Value(ThisItem.UnitPrice),"##,###.00")
```



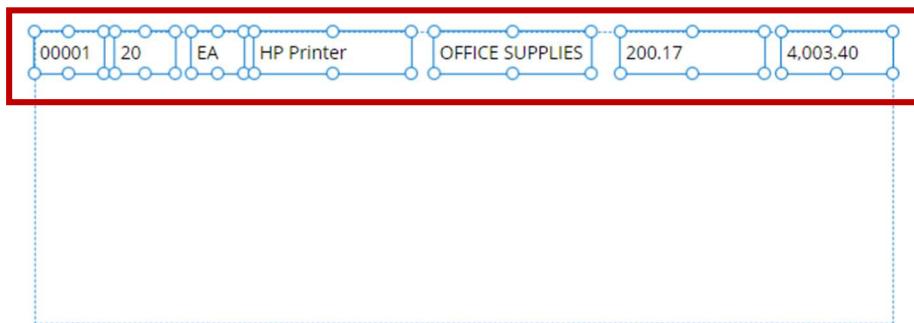
Repeat the same process for Total Price field.

```
Text(Value(ThisItem.TotalPrice),"##,###.00")
```

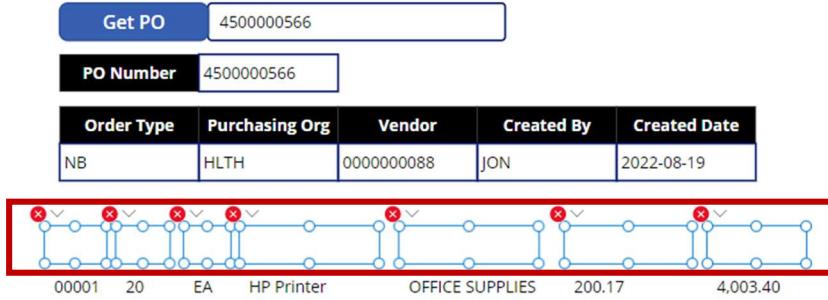


23. Create Headers for our Line Item Gallery. A quick way to create headers for all the columns in the gallery is to **select all the labels in the gallery** and then **Ctrl-C** to copy them.

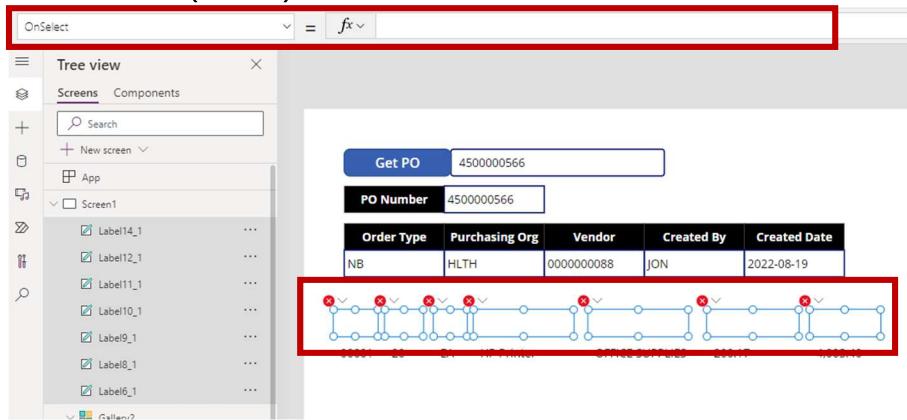
Get PO	4500000566			
PO Number	4500000566			
Order Type	Purchasing Org	Vendor	Created By	Created Date
NB	HLTH	0000000088	JON	2022-08-19



24. Paste the labels. **Click outside the gallery**, click **Ctrl-V** to paste and then **drag and drop** them as a group to line up above the line item gallery.

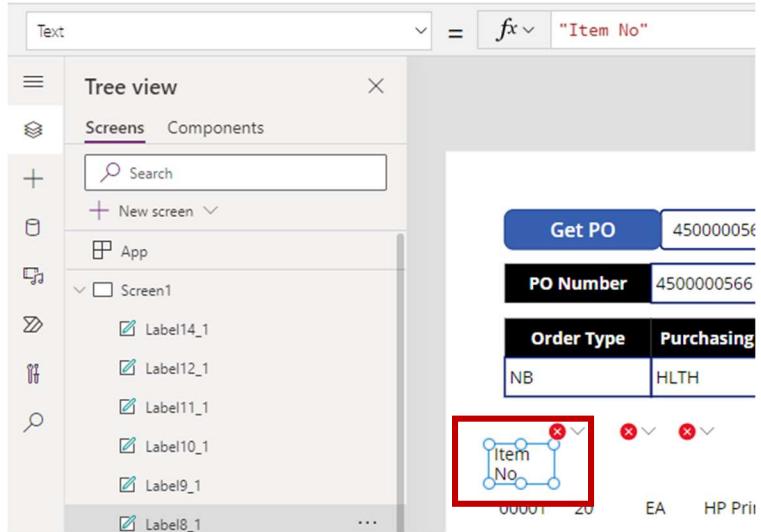


25. Configure the Header Labels. Clear out the OnSelect formula. Keeping all the labels selected, scroll down to the **On-Select** property in the property drop down list and delete Select (Parent). This removes the link to the data.



26. Individually select each header label and set the **Text Property** based on the table below:

Header Name	Text Property
Item Number	"Item No"
Quantity	"Qty"
Unit	"U/M"
Description	"Description"
Material	"Material"
Unit Price	"Unit Price"
Total Price	"Total Price"



27. Select all the header labels again and change the following properties:

Font Weight – Bold

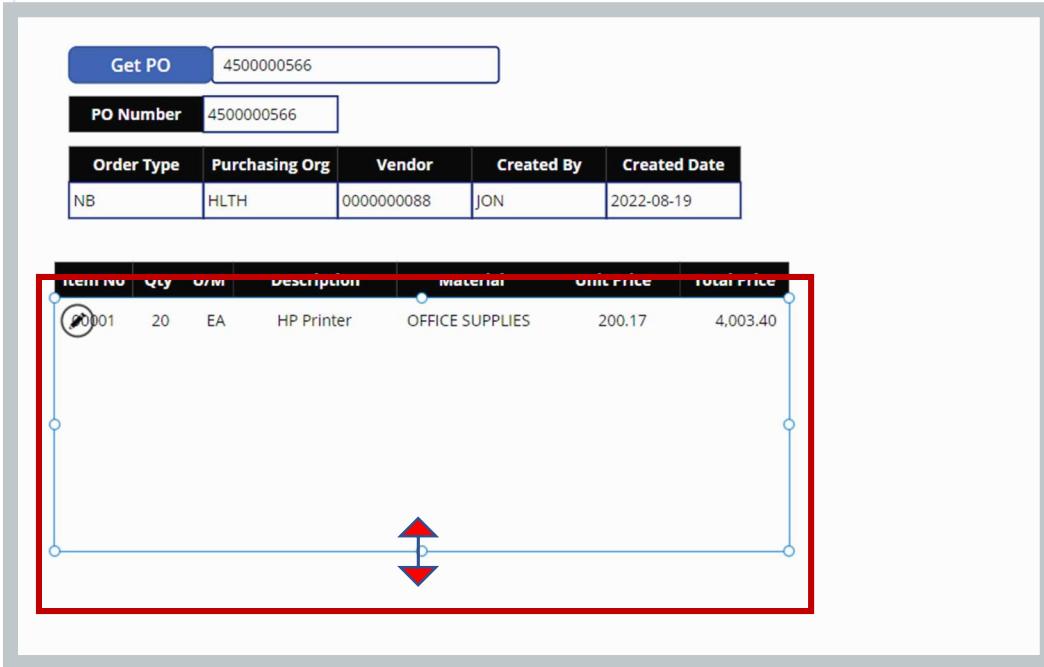
Text Alignment – Center

Text Color – White

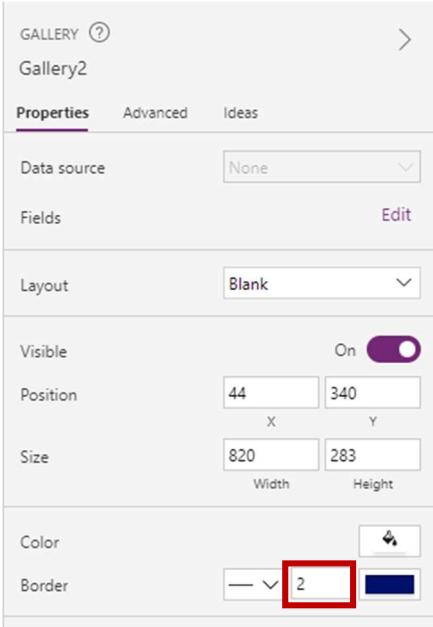
Text Fill – Black

Item No	Qty	U/M	Description	Material	Unit Price	Total Price
00001	20	EA	HP Printer	0000000088	JON	2022-08-19

28. Configure the Gallery. Select the gallery and reduce the size of the gallery so we can add another text label for total at the bottom of the screen.



29. Add a border around the gallery. On the **properties** on the right-hand side, change **border** to 2.



30. Reduce the size of the template cell. Click on the pencil icon in the gallery to edit the template cell.

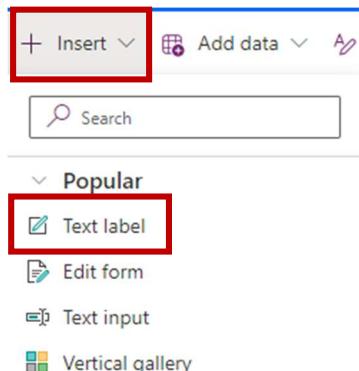
Item No	Qty	U/M	Description	Material	Unit Price	Total
00001	20	EA	HP Printer	OFFICE SUPPLIES	200.17	4,

Reduce the size of the template cell. This will be the size of each row in the gallery so we can have multiple rows showing on the screen.

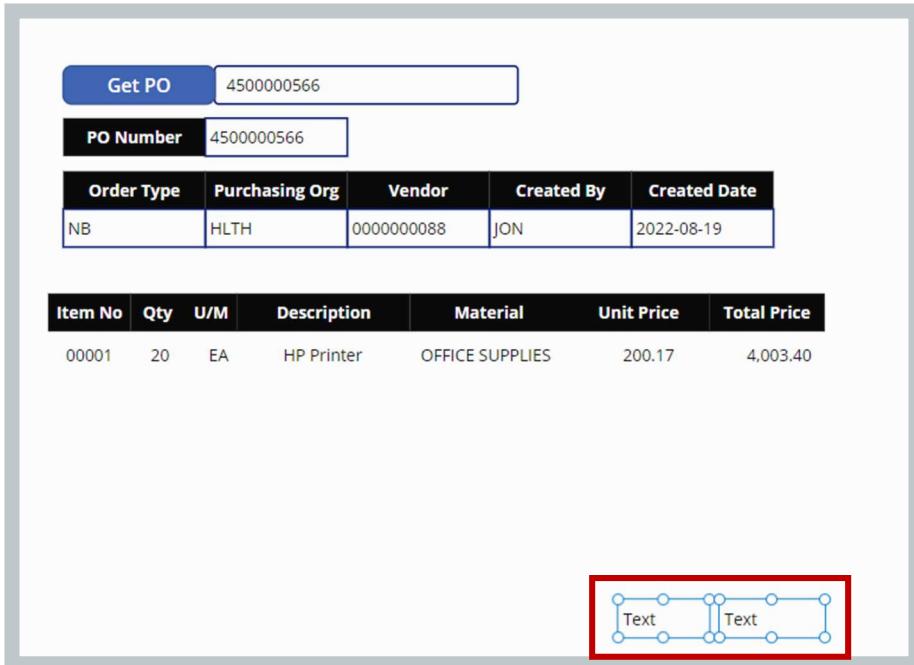
Get PO	4500000566			
PO Number	4500000566			
Order Type	Purchasing Org	Vendor	Created By	Created Date
NB	HLTH	0000000088	JON	2022-08-19

Item No	Qty	U/M	Description	Material	Unit Price	Total Price
00001	20	EA	HP Printer	OFFICE SUPPLIES	200.17	4,003.40

31. Add Text Labels for Total Price of Invoice. Click outside the gallery to unselect it. Click on Insert then Text label. Repeat so you have (2) new text labels.



32. Drag and drop the new text labels under the gallery on the right-hand side.



33. Configure the text labels. Change the **Text Property** for the new labels as follows:

Text Box 1 – “PO Total:”

`Text = fx "PO Total"`

Text Box 2 – `Text(Sum(Gallery1.AllItems,TotalPrice),"###,###.##")`

If you get an error message when you paste, try typing in the formula using Intellisense.

`Text = fx Text(Sum(Gallery1.AllItems, TotalPrice), "###,###.00")`

The screenshot shows a SAP Fiori application interface. At the top, there is a header bar with a search input field containing "Get PO" and a value "4500000566". Below the header is a table with the following data:

Order Type	Purchasing Org	Vendor	Created By	Created Date
NB	HLTH	0000000088	JON	2022-08-19

Below this is another table showing the items on the purchase order:

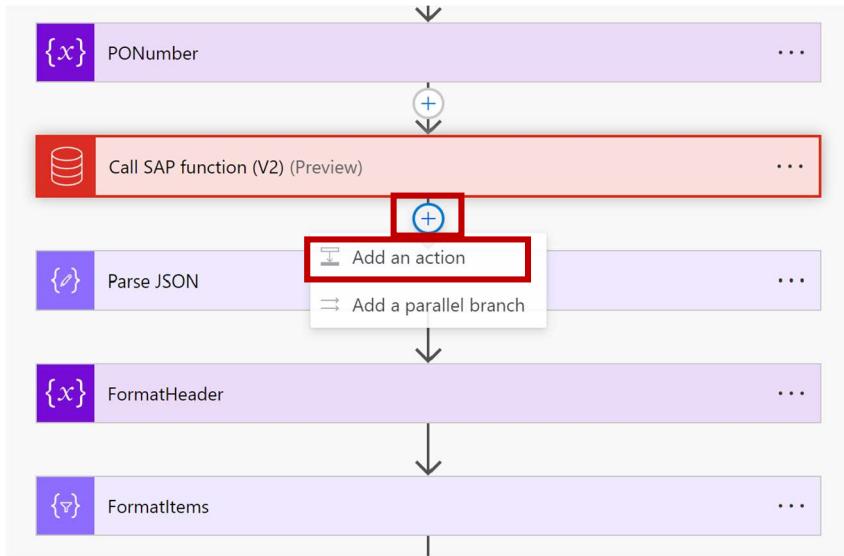
Item No	Qty	U/M	Description	Material	Unit Price	Total Price
00001	20	EA	HP Printer	OFFICE SUPPLIES	200.17	4,003.40

At the bottom of the table, it says "PO Total \$4,003.40".

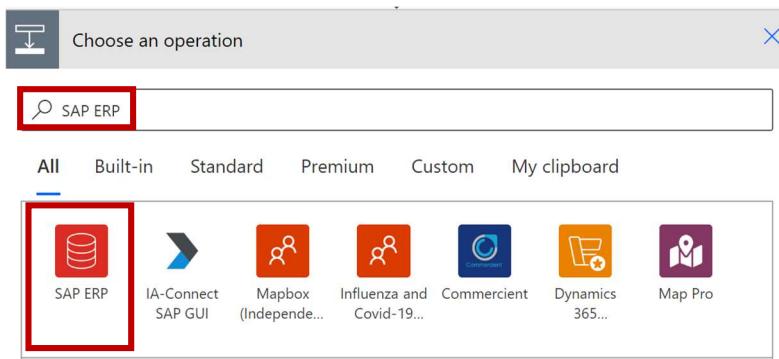
34. Save your app.
35. Modify our cloud flow to use SAP Read Table action to bring back the vendor name rather than the vendor number. Click over to your tab that has your solution open. Select the **GetPurchaseOrder** cloud flow, click on the ellipse (...) and then **Edit**.

The screenshot shows the Power Platform Studio interface. On the left, there is a sidebar with a tree view of objects. Under "Cloud flows", there is one item named "GetPurchaseOrder". This item is highlighted with a red box. To the right of the sidebar, the main area shows a list of cloud flows. One specific cloud flow, "GetPurchaseOrder", is also highlighted with a red box. A context menu is open next to this item, with the "Edit" option highlighted by another red box. Other options in the menu include "Details", "See analytics", "Turn off", "Advanced", and "Remove".

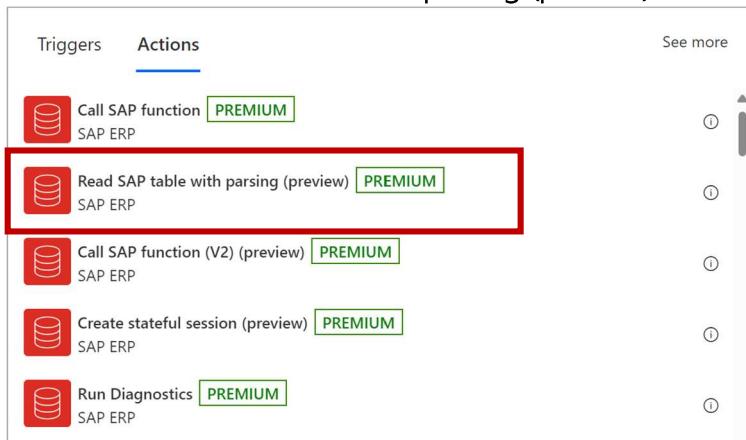
36. Add an action. Hover over the arrow after the [RFC] Call function in SAP and click on the + to add a new action. Click on Add an Action.



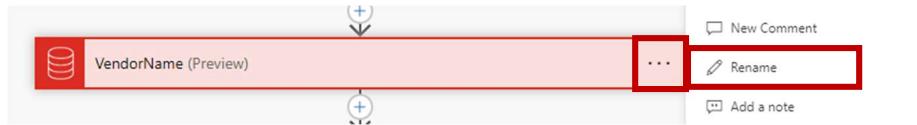
37. Search for SAP ERP and select the SAP ERP action.



38. Select the Read SAP Table with parsing (preview) action.



39. Rename the action. Click on the ellipse (...) and select Rename. Rename the action VendorName.



40. Configure the action:

- ☒ SAP system - use the environment variable we created in an earlier exercise from Dynamic Content – sap_sap_application
- ☒ Table name, type in LFA1
- ☒ Fields to read – Item – 1 – NAME1
- ☒ Count of rows to read – 1
- ☒ Filter – copy and paste where highlighted text is from Dynamic Content.
LIFNR = 'POHEADER VENDOR'

NOTE: You must have a space before and after the = for the filter and single quotes outside the Dynamic Content.

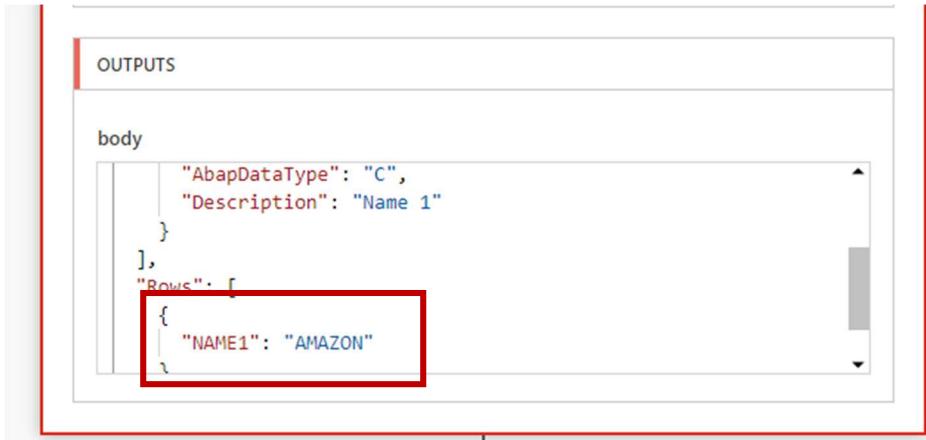
The screenshot shows the configuration of the 'VendorName (Preview)' action. The main configuration area includes:

- SAP system:** Set to 'sap_application (sap_sap_application)'. A red box highlights this section.
- Table name:** Set to 'LFA1'.
- Fields to read Item - 1:** Set to 'NAME1'.
- Count of rows to read:** Set to '1'.
- Where filters Item - 1:** Set to 'LIFNR = POHEADER VEN...' (with a red box highlighting the filter condition).
- Starting row index:** Set to 'Starting row index, e.g. 0'.

On the right side, there is a sidebar with the following sections:

- Add dynamic content from the apps and connectors used in this flow.** (button)
- Dynamic content** (selected tab)
- Search dynamic content** input field
- Environment Variables** list:
 - [@] SAP Application Server (sap_SAPIAddress)
 - [@] SAP Development (sap_SAPDevelopment)
 - [@] SAP SSO with Load Balancing (sap_SAPSSOwithLoadBal...)
 - [@] sap_application (sap_sap_application)

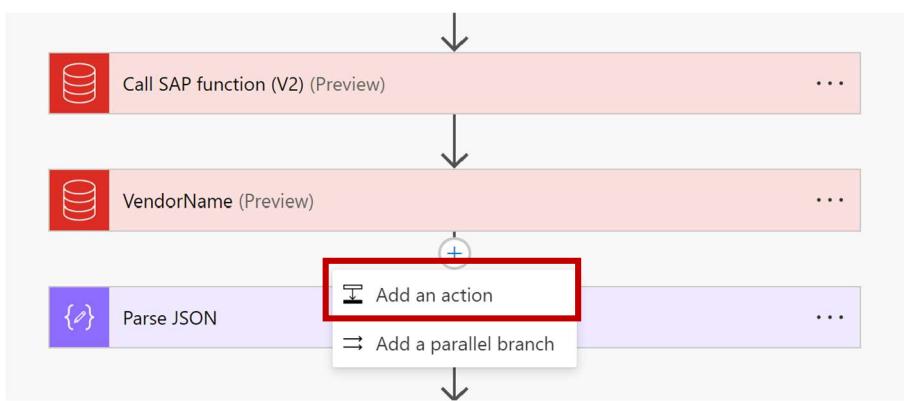
41. Save the flow. Click on Save.
 42. Test the flow. Use Automatically with a recently used trigger.
 43. Expand the **VendorName** action and look at what is being returned in the **Outputs** section. An object called rows which is an array with a single name/value pair.



```
body
  [
    {
      "AbapDataType": "C",
      "Description": "Name 1"
    }
  ],
  "Rows": [
    {
      "NAME1": "AMAZON"
    }
  ]
}
```

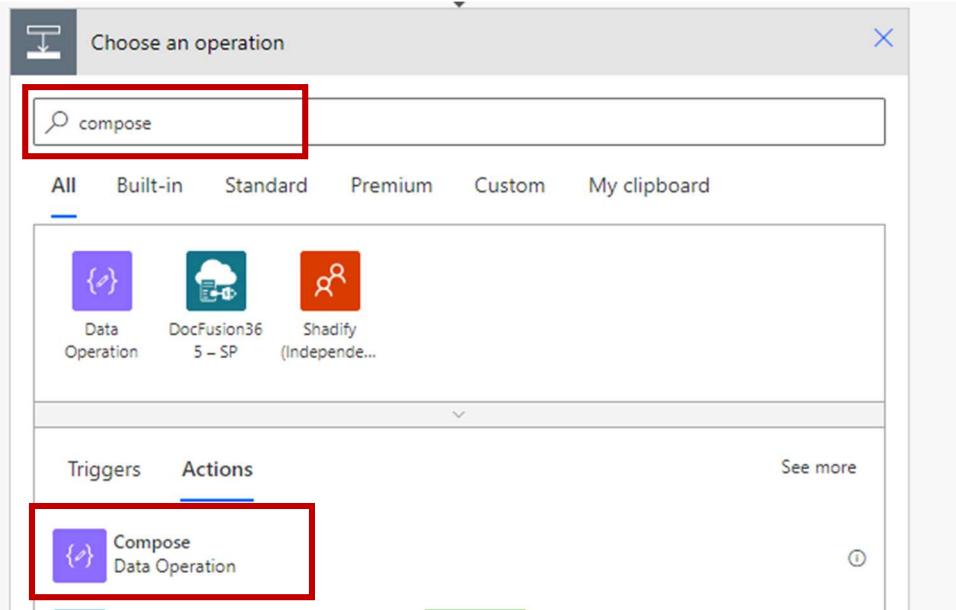
44. Create new action. Click on **Edit** to add a new action.

45. Add a new action below **VendorName** action.



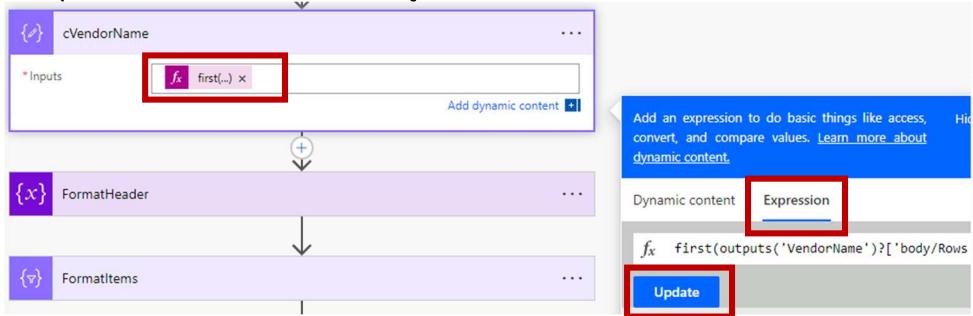
46. Add the Compose Action. Another Data Operation we can use is called Compose.

The Compose action allows you to give it a formula and it will evaluate the formula and return the result in the output. Search for **Compose** and select the **Compose Data Operation**.

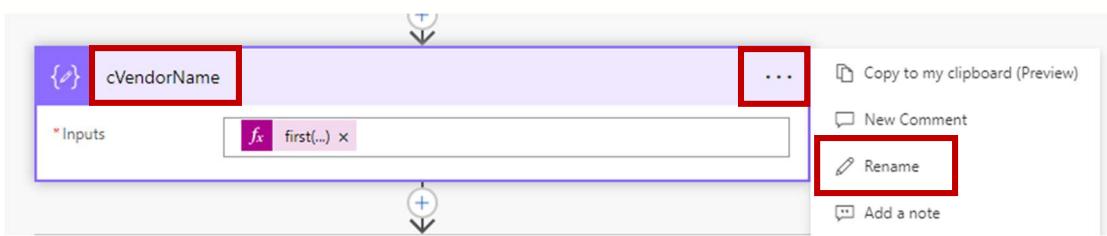


47. Click in **Inputs** and paste in the following **Dynamic Expression** and click on **Update**.

`first(outputs('VendorName')?['body/Rows'])['NAME1']`

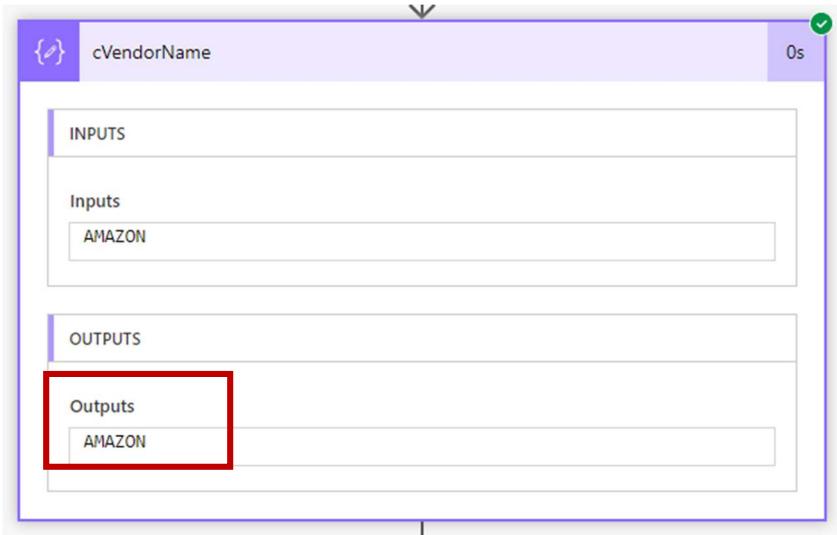


48. Rename the action. Click on the **ellipsis (...)** and then **Rename**. Rename to **cVendorName**.



49. Test the flow again using Automatic with recently used Trigger.

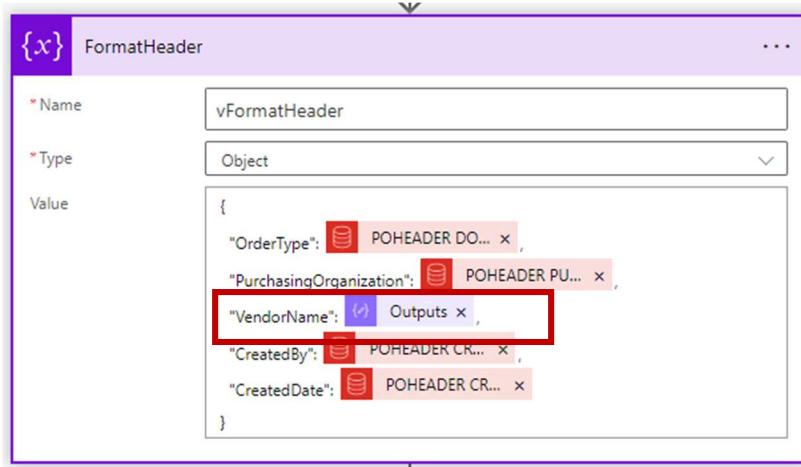
50. Expand the **cVendorName** action and verify that the **Output** has the name of the vendor – Amazon.



51. Update FormatHeader action. We are currently returning the Vendor number and we need to change to the Vendor Name which is coming from the output of our Compose action. Click on **Edit** to edit the Flow. **Expand the FormatHeader**.

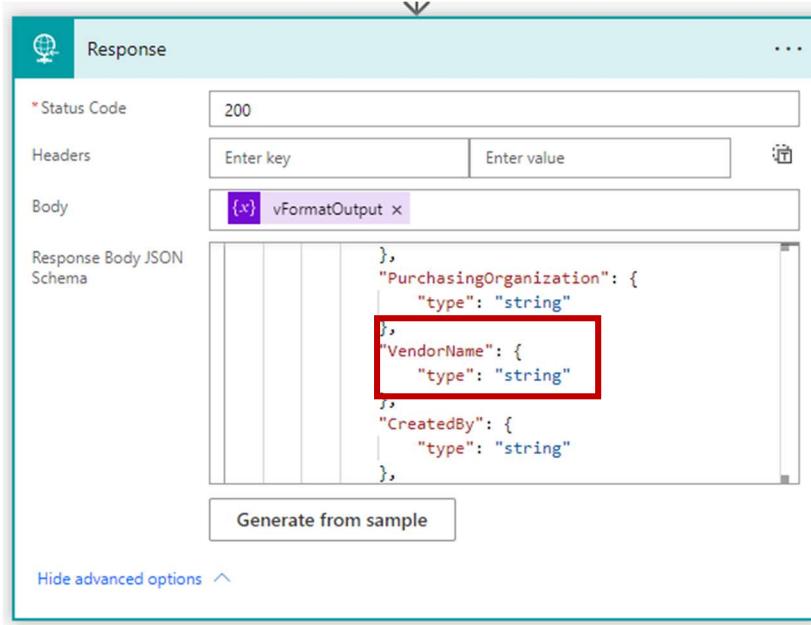
Update the value section to now have the following name/value pair where the highlighted text is from Dynamic Content so we no longer have the Vendor field.

"VendorName": Outputs

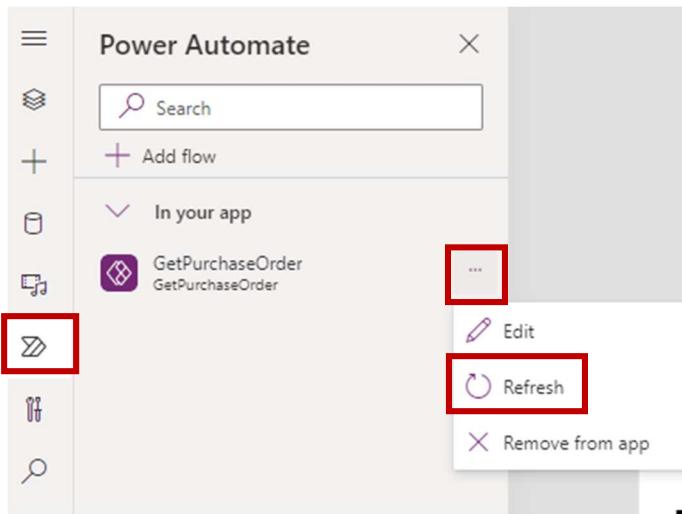


52. Update the Response action. Now we need to update the schema in the Response action to include the new VendorName in the Header section.

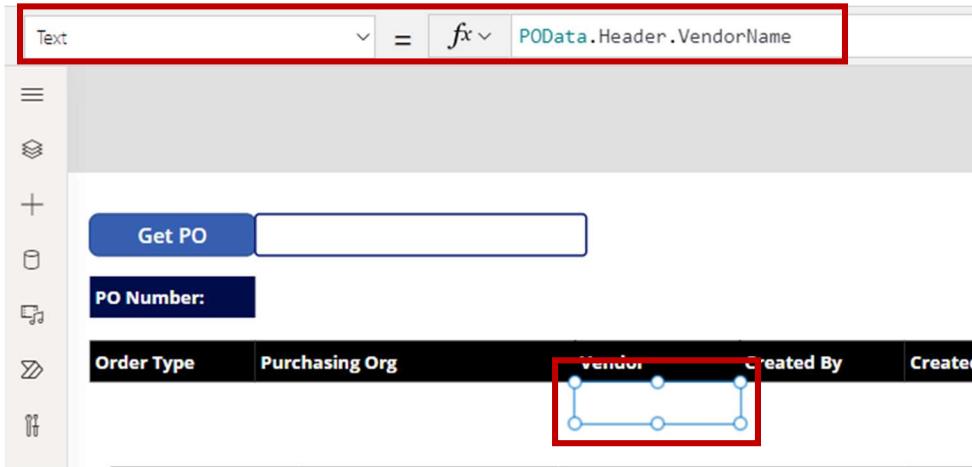
In the **Response Body JSON schema**, scroll down to the **Header Section**. Update the schema to remove the Vendor field and change to **VendorName**.



53. Save your cloud flow.
54. Click back to your tab with your Power App – Purchase Order Viewer.
55. Refresh your cloud flow. Click on the **Power Automate tab** on the left-hand side ribbon, click on the **ellipsis (...)** for GetPurchaseOrder cloud flow and then click on **Refresh**.



56. Edit the Vendor Text field. We now need to update the Vendor text field to reflect our new variable – VendorName. Select the **Vendor Text** field. Change the **text** property field to **POData.Header.VendorName**.



57. Play the app to reload the data. Hold down the **Alt key** and click on the **GetPO** button to reload the data and notice you now have the vendor name in your app.

Order Type	Purchasing Org	Vendor	Created By	Created Date
NB	HLTH	AMAZON	JON	2022-08-19

Item No	Quantity	U/M	Description	Material	Unit Price	Total Price
00001	20	EA	HP Printer	OFFICE SUPPLIES	200.17	4,003.40

PO Total: \$4,003.40

Congratulations on completing this app that retrieves PO information from SAP!

Exercise 7: Procure to Pay Example

In this next group of exercises, we will be working in some of the existing canvas app and cloud flows in the SAP Integration (Public Preview) solution that were built to create purchase orders in SAP. We will walk through the procure to pay steps that are accomplished with this solution.

1. Explore the solution. In your existing environment, **select the SAP Integration (Public Preview) solution**, select the solution to **edit**. You can see there are many objects inside the solution.

The screenshot shows the Power Automate interface with the 'Solutions' tab selected. A red box highlights the 'SAP Integration' row in the list, which is currently selected. The list includes various other solutions like 'Power Platform Conference Workshop - UserXX', 'Power Platform Conference Workshop - BAB', and 'Power Platform Conference Workshop BB'. The 'SAP Integration' row has columns for Display name, Name, Created, Version, and Managed.

Display name	Name	Created	Version	Managed
Power Platform Conference Workshop - UserXX	PowerPlatformConfe...	3 days ago	1.0.0.0	No
Power Platform Conference Workshop - BAB	PowerPlatformConfe...	3 days ago	1.0.0.0	No
SAP Integration	SAPSalesOrders	4 days ago	0.87	No
Power Platform Conference Workshop BB	PowerPlatformConfe...	1 week ago	1.0.0.0	No
Power Platform Conference Workshop	PowerPlatformConfe...	3 weeks ago	1.0.0.0	No
SAPUserSearch	SAPUserSearch	1 month ago	1.0.0.0	No
DVTest Solution	DVTestSolution	1 month ago	1.0.0.0	No
SAP Connector Demo	SAPConnectorDemo	1 month ago	1.0.0.0	No

2. Create a new Purchase Order in SAP. We will use the canvas app Purchase Orders to create a new purchase order. Click on the **ellipsis (...)** for the **Purchase Order App** and click on **Play**.

The screenshot shows the Purchase Orders canvas app. A red box highlights the 'Purchase Orders' item in the list. To the right, a context menu is open for the 'Purchase Orders' item, with a red box highlighting the 'Play' option under the 'Edit' section.

- Equipment Search
- Financial Documents Search
- GL Account Search
- Goods Movement Search
- Header and Menu Components
- Material Search
- Profit Center Search
- Purchase Order Search
- Purchase Orders**

Edit

▶ Play

Monitor

Details

Share

Settings

Add to Teams

Advanced

Remove

3. You may be prompted to enter your SAP credentials. Use the **SAP username** and **password** supplied by the instructor and then click on **Create**.

SAP ERP

SAP ERP
Premium

SAP ERP is an enterprise resource planning software developed by SAP SE. SAP ERP incorporates the key business functions of an organization. The SAP ERP connector for Power Automate and Power Apps allows you to invoke RFC and BAPI functions using on-premises data gateway. [Learn more](#)

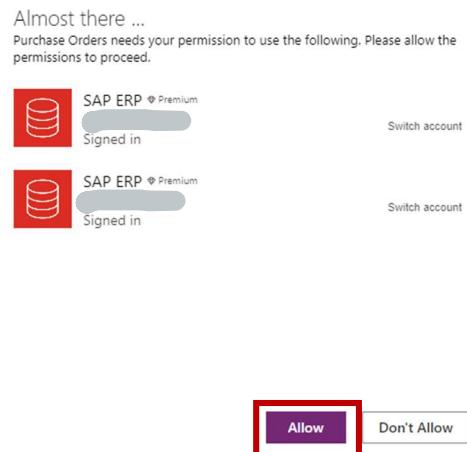
Authentication Type: SAP Authentication

SAP Username *: [Redacted]

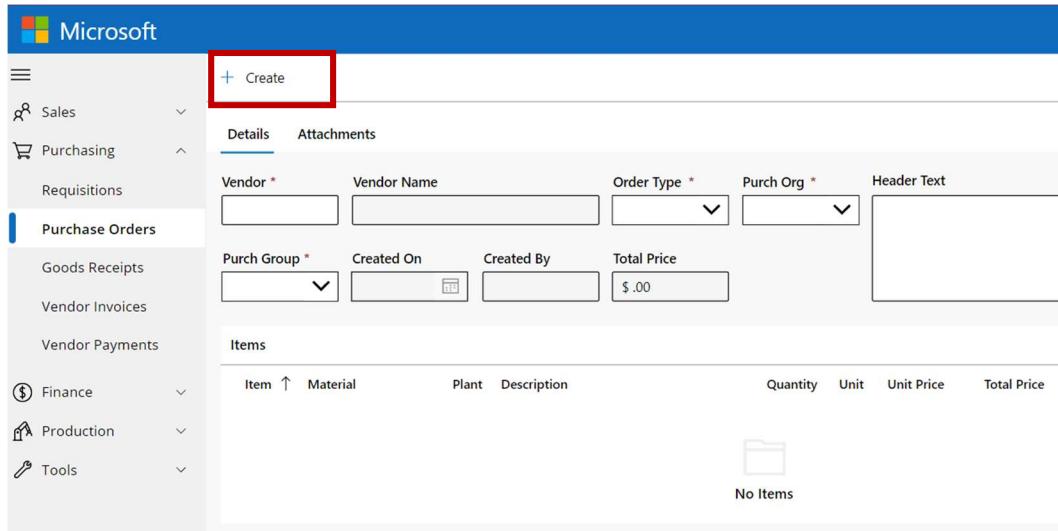
SAP Password *: [Redacted]

Create **Don't Allow**

4. Once you have been authenticated in the SAP System, you need to allow Purchase Orders to use the SAP ERP Connector. Click on **Allow**.



5. Create a new purchase order. Click on **+ Create** in the upper left-hand corner.



6. Search for the vendor. In the **Vendor** field, click on the magnifying glass to search for a vendor.

7. Click on **All** to search through all the vendors and then in the **City** field, type **REDMOND** (all capital letters – SAP is case sensitive) and then click on **Search** to search for vendors in Redmond.

Vendor	Name	Street	City	State	Country	Zip Code
197	VANDELAY INDUSTRIES	123 MAIN ST	REDMOND	WA	US	98502
MICROSOFT	MICROSOFT	234 PINE ST	REDMOND	WA	US	98765

8. Select your vendor. Select the vendor **197 Vandelay Industries** by clicking on the vendor number.

Vendor	Name	Street	City	State	Country	Zip Code
197	VANDELAY INDUSTRIES	123 MAIN ST	REDMOND	WA	US	98502
MICROSOFT	MICROSOFT	234 PINE ST	REDMOND	WA	US	98765

9. In our sample app, we only have 1 order type, purchasing org and purchasing group. You can configure this to have the drop downs reflect the different options for each customer's SAP system.

Details Attachments

Vendor *	Vendor Name	Order Type *	Purch Org *
197	VANDELAY INDUSTRIES	NB	HLTH
Purch Group *	Created On	Created By	Total Price
HLT			\$.00

10. Add some line items to the PO. Click on + Add

Items	+ Add	Delete	Material	Plant	Description	Quantity	Unit	Unit Price	Total Price	Requisition	Details

No Items

11. Click on the magnifying glass in the Material field to search for materials.

New Item

Item	Material *	Description		
1	<input type="text"/>	<input type="text"/>		
Plant *	Quantity *	Unit *	Unit Price *	Total Price
0001	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Requisition	Goods Receipt	Vendor Invoice	Payable	<input type="text"/>
Item Text 				
Save Cancel				

12. Click on the All tab to search through all the materials. In the description field, type OFFICE (all capital letters – SAP is case sensitive) and then click on the Search.

← Material Search

Recently viewed	All		
Description	<input type="text" value="OFFICE"/>	Material	Type
Search	Clear		
Material	Description	Unit	Type
OFFICE SUPPLIES	OFFICE SUPPLIES	EA	FERT

13. Click on Office Supplies in the Material list.

Material	Description	Unit	Type
OFFICE SUPPLIES	OFFICE SUPPLIES	EA	FERT

14. Fill out the information for the first line item. Change the Description field to LASER PRINTER, enter 1 for Quantity, select EA for Unit and type 250.00 for Unit Price. Finally, click on Save.

New Item

Item	Material *	Description		
1	OFFICE SUPPLIES	LASER PRINTER		
Plant *	Quantity *	Unit *	Unit Price *	Total Price
0001	1	EA	250.00	
Requisition	Goods Receipt	Vendor Invoice	Payable	
Item Text				
Save Cancel				

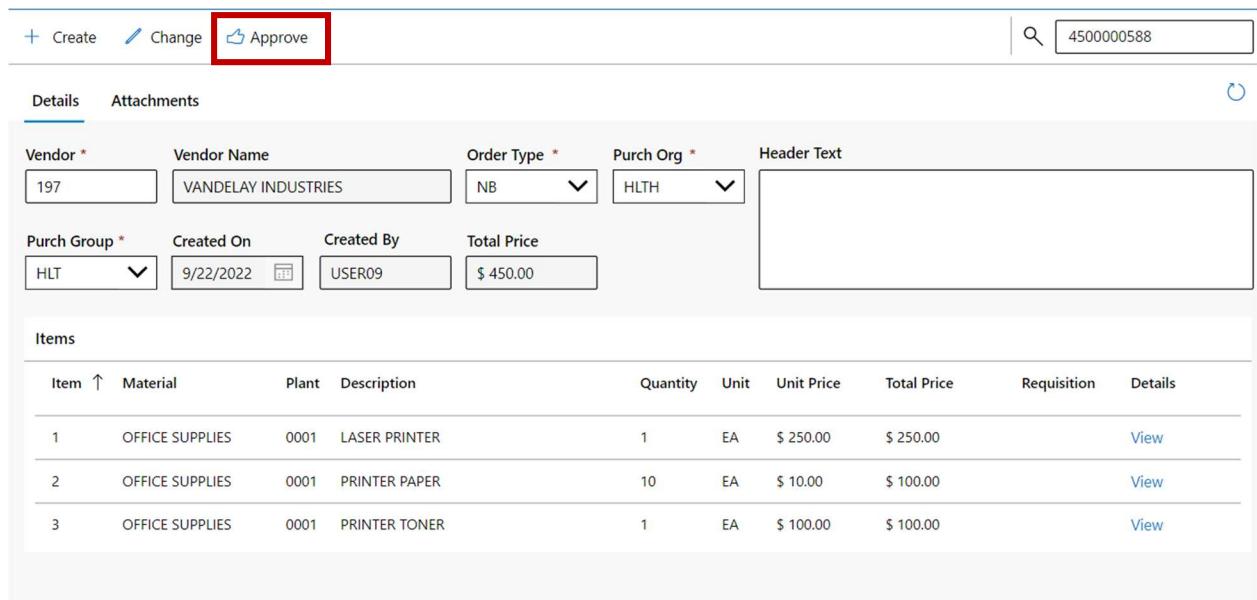
15. Follow steps 10 – 14 again to add the remaining items to the purchase order based with the information in the table below. You don't have to do steps 11-13 since we will be using the Office Supplies material for the rest of the line items.

Description	Quantity	Unit	Unit Price
PRINTER PAPER	10	EA	10.00
PRINTER TONER	1	EA	100.00

16. Save the entire purchase order. Click on **Save**.

Save	Cancel	Enter Order...							
Details Attachments									
Vendor * Vendor Name Order Type * Purch Org * Header Text 197 VANDELAY INDUSTRIES NB HLTH									
Purch Group * Created On Created By Total Price HLT									
Items + Add Delete									
Item ↑	Material	Plant	Description	Quantity	Unit	Unit Price	Total Price	Requisition	Details
1	OFFICE SUPPLIES	0001	LASER PRINTER	1	EA	\$ 250.00	\$.00	Edit	
2	OFFICE SUPPLIES	0001	PRINTER PAPER	10	EA	\$ 10.00	\$.00	Edit	
3	OFFICE SUPPLIES	0001	PRINTER TONER	1	EA	\$ 100.00	\$.00	Edit	

17. When you click on the Save button, this kicks-off a cloud flow, **CreatePurchaseOrder**, to send the purchase order information to SAP and then return to the PO number assigned in SAP which you will see in the success message in the green bar at the top and be populated in the Search bar at the top right hand of the screen.
18. Approve the Purchase Order. The second step in the process is to have someone approve the purchase order. Once the purchase order has been created in SAP, you can see we have a new option at the top of our screen – Approve. For our example, we can approve any purchase orders under \$ 500, so click on **Approve**.



The screenshot shows a Fiori application interface for a Purchase Order. At the top, there are three buttons: '+ Create', 'Change', and 'Approve' (which is highlighted with a red box). To the right is a search bar with the value '4500000588'. Below the buttons, there are two tabs: 'Details' (selected) and 'Attachments'. The 'Details' section contains fields for Vendor (197), Vendor Name (VANDELAY INDUSTRIES), Order Type (NB), Purch Org (HLTH), Header Text (empty), Purch Group (HLT), Created On (9/22/2022), Created By (USER09), and Total Price (\$ 450.00). The 'Items' section lists three items: Item 1 (OFFICE SUPPLIES, Plant 0001, Description LASER PRINTER, Quantity 1, Unit EA, Unit Price \$ 250.00, Total Price \$ 250.00), Item 2 (OFFICE SUPPLIES, Plant 0001, Description PRINTER PAPER, Quantity 10, Unit EA, Unit Price \$ 10.00, Total Price \$ 100.00), and Item 3 (OFFICE SUPPLIES, Plant 0001, Description PRINTER TONER, Quantity 1, Unit EA, Unit Price \$ 100.00, Total Price \$ 100.00). Each item row has a 'View' link on the right.

19. Receive the Goods. Behind the scenes, SAP releases that purchase order to the vendor. The vendor will then ship the goods to us and then we will receive the goods. You can see once the purchase order was approved; we now have a new button on the top of our app – Receive. Click on **Receive**.

20. Fill out the Goods Receipt. The Goods Receipt is pre-populated with the items and quantities from our purchase order. We can type in a **delivery note** and then click on **Save**. This will cause a Goods Receipt to be created in SAP.

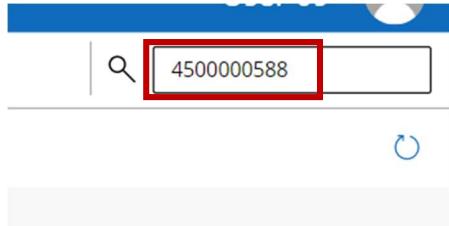
Description	Quantity Received *
LASER PRINTER	1
PRINTER PAPER	10
PRINTER TONER	1

You can click on **View** for one of the line items of the PO to see the Goods Receipt number.

View Item

Item	Material *	Description		
1	OFFICE SUPPLIES	LASER PRINTER		
Plant *	Quantity *	Unit *	Unit Price *	Total Price
0001	1	EA	250.00	\$ 250.0
Requisition	Goods Receipt	Vendor Invoice	Payable	
	5000000289			
Item Text				
LASER PRINTER				

21. Create our invoice for this purchase order. We need to create a vendor invoice with our new purchase order number for one of our next exercises. Copy the PO number in the search box in the upper right-hand corner of the screen.



NOTE: Make a note of this purchase order number on your credentials sheet or you will have to open up your vendor invoice for a later exercise.

22. Open the **vendor invoice template** stored in the **Files** on the **SAP Workshop Teams** site. See exercise 0 on how to access this site. This was also emailed to you a few days before the workshop. Replace the **purchase order number** with your newly created purchase order number. Click on **File, Save** and save as a **.pdf** on your desktop for one of the next exercises.

INVOICE

11144

Vandelay Industries
123 Main St
Redmond, WA 98502

Bill To: Kruger Industrial Smoothing
456 Alpine Dr
Indianapolis, IN 46054

Purchase Order: 4500000588

Date: September 23, 2022

Due Date: October 23, 2022

Balance Due: **\$450.00**

Vendor Id: US-104

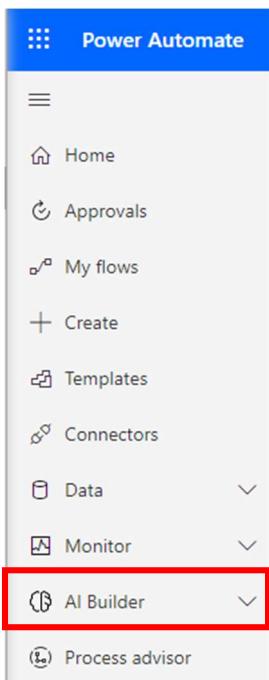
Item	Product	Description	Quantity	Unit	Unit Price	Amount
1	B07QPWKYG9	HP LASER PRINTER	1	EA	\$250.00	\$250.00
2	B07N1DF9VV	PRINTER PAPER	10	EA	\$10.00	\$100.00
3	W07N1DF9VV	PRINTER TONER	1	EA	\$100.00	\$100.00

Subtotal: \$450.00
Tax (0%): \$0.00
Amount: \$450.00

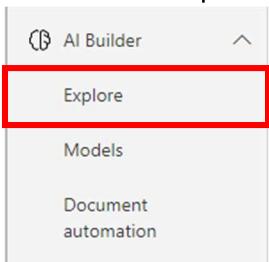
Exercise 8: Create and Add AI Builder Model

In the next exercise, we will create and train an AI Builder model to capture information from our vendor invoices.

1. Browse to <https://make.power automate.com> and confirm you are still signed in with the credentials supplied by the instructor and you are in the correct environment.
2. On the left-hand side, expand the **AI Builder** section.



3. Click on the **Explore** section.

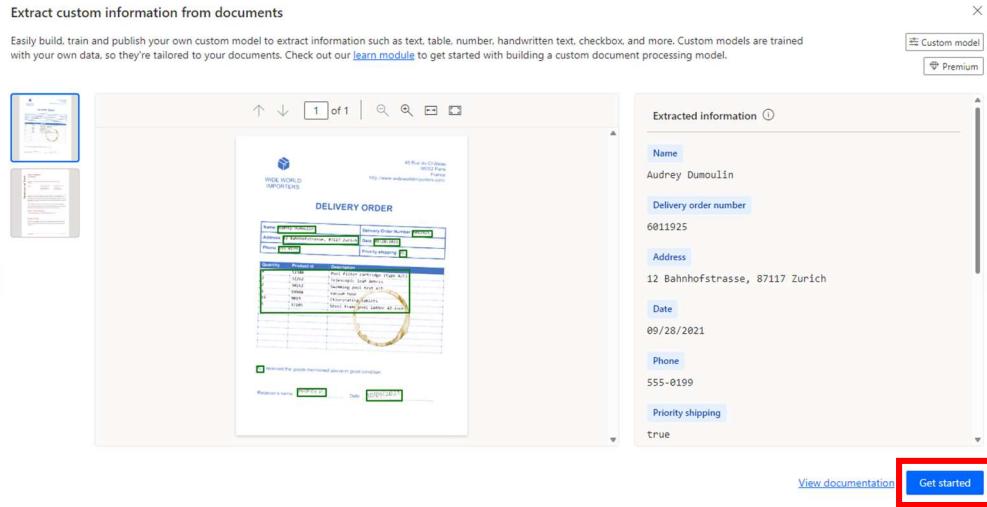


4. Select the Document Processing Model.

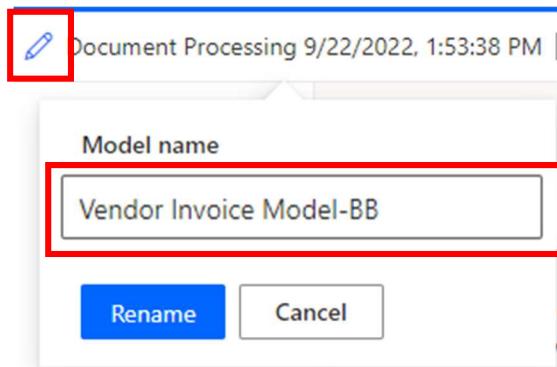
The screenshot shows a grid of document processing models. Each model has a preview icon, a name, and a brief description. The 'Document processing' model, which is highlighted with a red box, includes a 'Custom model' button. Other models shown include Invoice processing, Text recognition, Receipt processing, Identity document reader, Business card reader, Sentiment analysis, Category classification, Entity extraction, and Key phrase extraction.

Invoice processing Extract information from invoices	Text recognition Extract all the text in photos and PDF documents (OCR)	Receipt processing Extract information from receipts	Identity document reader Extract information from identity documents	Business card reader Extract information from business cards
Document processing Extract custom information from documents	Sentiment analysis Detect positive, negative, or neutral sentiment in text data	Category classification Classify customer feedback into predefined categories	Entity extraction Extract key elements from text, and classifies them into predefined categories	Key phrase extraction Extract most relevant words and phrases from text

5. Click on **Get Started**.

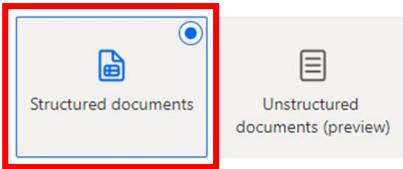


6. Rename the model. In the upper right-hand corner click on the **pencil icon** to rename the model. **Rename** the model **Vendor Invoice Model-XX** where XX is your initials. Click on **Rename**.



7. Choose the **Structured Documents** for extracting information and click on **Next**.

Select the type of documents your model will process



Structured documents are those where for a given layout, the fields, tables, checkboxes, and other information are defined. Unstructured documents are those where the layout can vary from document to document.

Examples of structured documents are invoices, purchase orders, delivery orders, tax documents, and contracts.

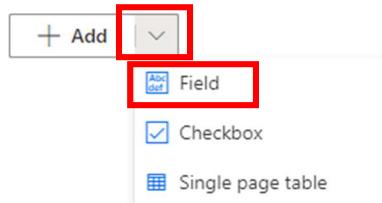
Here you can see some examples of structured documents:

The image shows a preview of structured documents. On the left, there are three small thumbnail images of different document layouts. On the right, a larger preview window shows a bill of materials for 'Contoso, Ltd.' with items like 'Complete golf set. Left-handed', 'Portable clip-on umbrella', 'MicroFiber Golf Towels 3 Pack', and 'Golf Tees 25 Pack'. At the bottom left of the preview window, a blue 'Next' button is highlighted with a red box.

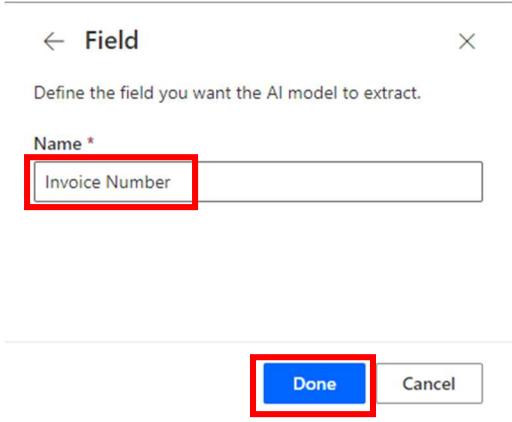
8. Add the Fields for Extraction. We will now add the fields we want to extract from the documents we are processing. Click on the drop-down menu in the + Add box and select Field.

Choose information to extract

List all pieces of information that you want the AI model to extract from your documents. For example: Name, Address, Total amount, Line items... You'll tag them in the documents.



9. Type Invoice Number as the Name for the first field and click Done.



10. Repeat the last 2 steps to add the rest of the fields we want to extract from the invoice.

Name	Type
Purchase Order	Field
Invoice Date	Field
Total Amount	Field
Vendor Name	Field

11. Add the table for the line items. Click on the drop-down in the +Add box and select Single page table.

Choose information to extract

List all pieces of information that you want the AI model to extract from your document. For example: Name, Address, Total amount, Line items... You'll tag them in the next step.

12. Name the table Line Items

← Single page table

X

Give your table a name and define the columns you want the AI model to extract. This capability is currently in preview.

Column 1	+

Done **Cancel**

13. Rename the columns. Click on Column 1 and **Rename column**.

← Single page table

Give your table a name and define the columns yo
preview.

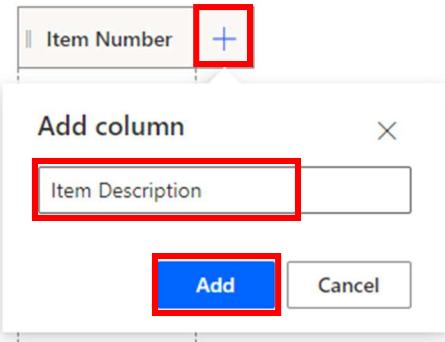
Column 1	+
Rename column	
Delete column	

14. Type **Item Number** for the name and then click on **Rename**.

Rename column

Rename **Cancel**

15. Add another column. Click on the + and name the column **Item Description**. Click on **Add**.



16. Repeat the last step to add the following columns to our table.

Name
Quantity
Unit
Unit Price
Total

Click on Done when you have added all the columns.

17. Click on Next.



18. Train the model. We are now ready to train the mode with some sample invoices. Click on New collection.

Add collections of documents

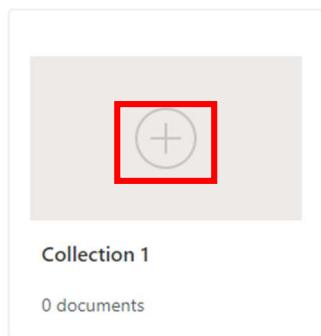
Add sample documents for your model to study. Put similar documents into the same collection.

Create a collection for each layout ⓘ

New collection

19. Click on + to add your invoices.

Add documents ⓘ



20. Click on Add documents.

Collection 1
0 documents

×



Add 5 or more documents with the same layout

JPG, .PNG or .PDF — up to 50 MB or 500 pages in total.

Add documents

21. Select the source for your documents. Click on **My device** and browse to where you downloaded the documents at the beginning of the workshop.

Select source

Select a data source



My device



SharePoint



Azure Blob Storage

Premium

22. Select all 5 Vendor Invoices from the Train folder.

A screenshot of a list interface showing five vendor invoices. Each invoice has a checkbox next to it, which is checked for all five items. A red box highlights the first five rows of the list.

Name
Vendor Invoice 11113
Vendor Invoice 11115
Vendor Invoice 11117
Vendor Invoice 11128
Vendor Invoice 11133

23. Click on Upload 5 documents.

Upload documents

These documents will be used to train your model.

	Name	Size	Status
✓	Vendor Invoice 11113.pdf	103.2 KB	
✓	Vendor Invoice 11115.pdf	103.1 KB	
✓	Vendor Invoice 11117.pdf	103.6 KB	
✓	Vendor Invoice 11128.pdf	108.3 KB	
✓	Vendor Invoice 11133.pdf	104.7 KB	

Upload 5 documents

Cancel

24. Click on Done.

Upload documents

These documents will be used to train your model.

X

	Name	Size	Status
	Vendor Invoice 11113.pdf	103.2 KB	Uploaded
	Vendor Invoice 11115.pdf	103.1 KB	Uploaded
	Vendor Invoice 11117.pdf	103.6 KB	Uploaded
	Vendor Invoice 11128.pdf	108.3 KB	Uploaded
	Vendor Invoice 11133.pdf	104.7 KB	Uploaded

Done

25. Click on **Next**.



26. Tag the Documents. Next, we will tag the areas on the invoice and associate them with our fields. The first document has been selected. Right-Click and select the area around the invoice number and then map to the **Invoice Number** field. You can see the value of the invoice number to confirm it has been selected properly.

INVOICE

Value
11115

Vandelay Indu
123 Main St
Redmond, WA

Bill To:
Kruger Indust
456 Alpine Dr
Indianapolis, IN 46054

Purchase Order:
Date:
Due Date:
Balance Due:

Vendor Id: US-104

27. Repeat step 26 to tag the rest of the fields. When complete, you should have green boxes around all the areas in the invoice we have tagged and green check marks showing all the fields have been tagged.

INVOICE

11115

Vandelay Industries
123 Main St
Redmond, WA 98502

Bill To:
Kruger Industrial Smoothing
456 Alpine Dr
Indianapolis, IN 46054

Purchase Order: 9400000050
Date: September 29, 2022
Due Date: October 30, 2022
Balance Due: \$119.89

Vendor Id: US-104

Item	Product	Description	Quantity	Unit	Unit Price	Amount
1	K22HIAUTF2	DESKJET PRINTER	1	EA	\$119.89	\$119.89

Subtotal: \$119.89
Tax (0%): \$0.00
Amount: \$119.89

Fields
 Invoice Number ...
 Purchase Order ...
 Invoice Date ...
 Total Amount ...
 Vendor Name ...
Tables
 Line Items ...

28. Tag the Line-Item table. Right-click and drag the cursor over the whole Line Items table to select it.

Vendor Id: US-104

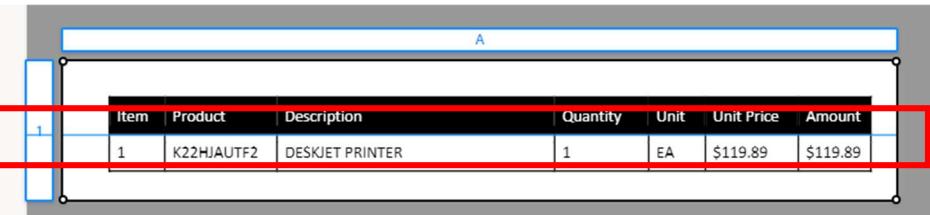
Item	Product	Description	Quantity	Unit	Unit Price	Amount
1	K22HIAUTF2	DESKJET PRINTER	1	EA	\$119.89	\$119.89

29. Select the Line Items table.

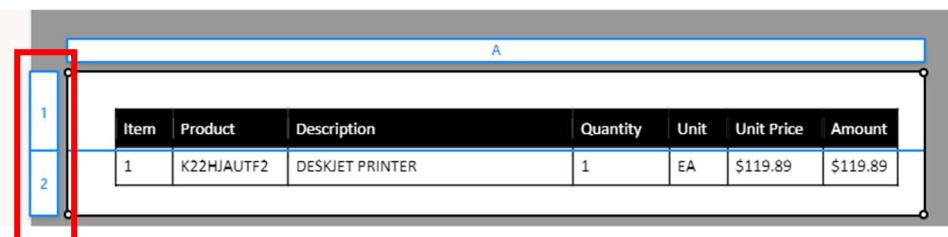
The screenshot shows a Fiori application interface. At the top left, it says "Redmond, WA 98502". Below that, "Bill To:" information is listed: "Kruger Industrial Smoothing", "456 Alpine Dr", and "Indianapolis, IN 46054". To the right, there's a "Value" section with a table containing one row: "Item Product Description Quantity Unit Unit Price Amount" with values "1 K22HJAUTF2 DESKJET PRINTER 1 EA \$119.89 \$119.89". Next to it are fields for "45000000560" (highlighted in green), "September 29, 2022" (highlighted in green), "October 30, 2022", and "\$119.89". A red box highlights the "Tables" section, which contains a radio button for "Line Items". Below this is a "Continue tagging" button and a right-pointing arrow. A large blue box encloses the "Line Items" table, which has the following data:

Item	Product	Description	Quantity	Unit	Unit Price	Amount
1	K22HJAUTF2	DESKJET PRINTER	1	EA	\$119.89	\$119.89

30. Map out the rows of the table. There are great pop up instructions for this step you can click through the first time. To map where the rows are, move the cursor so the blue line is between the first and second row and then click your mouse.



31. You should now see 2 rows identified in the outside blue box.



32. Map out the columns of the table. We will now follow a similar process to map out the columns. **Hold down the Ctrl key** and move the cursor between the first and second column and then click your mouse. You should see 2 columns identified in the outside blue box.

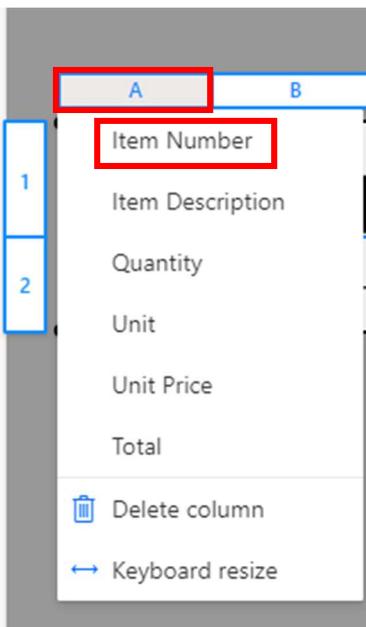
	A	B	
1	Item	Product	Description
2	1	K22HJAUTF2	DESKJET PRINTER

	A	B	C	D	E	F	G
1	Item	Product	Description	Quantity	Unit	Unit Price	Amount
2	1	K22HJAUTF2	DESKJET PRINTER	1	EA	\$119.89	\$119.89

33. Repeat step 32 to map the rest of the columns.

	A	B	C	D	E	F	G
1	Item Number	Item Description					
2	1	K22HJAUTF2	DESKJET PRINTER	1	EA	\$119.89	\$119.89

34. Map the column headings to the columns we identified in the beginning. Click in the blue box for column A and select Item Number to map.



35. Map the rest of the columns. **Repeat step 33** and map the rest of the columns according to the table below. Note, we will be skipping column B as we are not mapping the product number.

Column	Mapping
Column C	Item Description
Column D	Quantity
Column E	Unit

Column F	Unit Price
Column G	Total

Item Number	B	Item Description	Quantity	Unit	Unit Price	Total
Item	Product	Description	Quantity	Unit	Unit Price	Total
1	K22HJAUTF2	DESKJET PRINTER	1	EA	\$119.89	\$119.89
2						

36. Verify data. You can see on the right-hand side that data that is being pulled from the document based on our tagging of the table. We want to turn on **Ignore the first row** since that is our header information. Click on **Done**.

Line Items

022 11:48:58 AM Advanced tagging mode ⓘ Off

Ignore first row ⓘ On

	Item Num...	Item Desc...	Quantit
1	1	DESKJET PRINTER	1

Done

37. Repeat steps 26-36 for the remaining 4 documents. Click on **Next** when finished tagging all the documents.

Back

Next

38. Train the model. We are now ready to train the model. Click on **Train**. It may take a few minutes to finish training.

Model summary

Review your model's details below. If everything looks good, select Train. [Learn more about training](#)

Overview

Owner Barb Borrowman	Model type Document Processing	Document type Structured and semi-structured documents	Collections 1
-------------------------	-----------------------------------	---	------------------

Document sources

Data source	Number of documents
My device	5 documents

Information to extract

Type	Details
Fields	Invoice Number, Purchase Order, Invoice Date, Total Amount, Vendor Name
Line Items	Item Number, Item Description, Quantity, Unit, Unit Price, Total

Actions: Back, **Train**

39. Test your model. When the model is finished training, you can see the results of accuracy with more details on the specific items we are extracting. Click on Quick Test.

[Edit model](#) [Share](#) [Settings](#) [Delete](#)

Your model isn't published yet. Publish to use it in apps and flows. [Learn more](#)

Models > Document Processing 9/17/2022, 11:48:58 AM
Document Processing • Not published • Barb Borrowman

Accuracy score [More details](#)

92%
Good

This model correctly predicted 92% of actual results and may be ready to be used. To improve the accuracy score, [review full evaluation](#).

Actions: Publish, **Quick test**

Information to extract [More details](#)

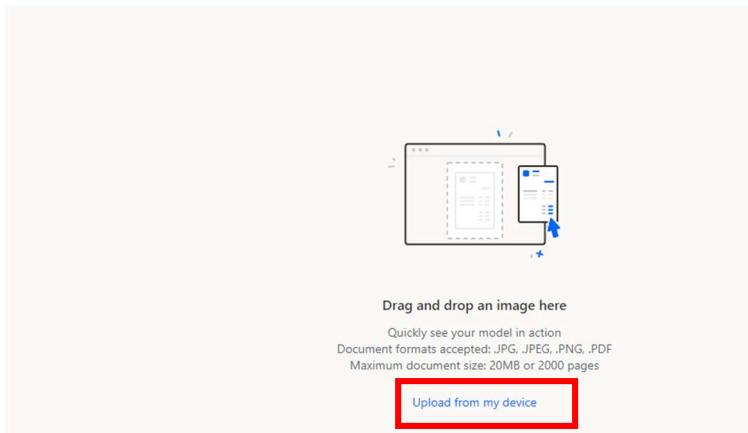
Invoice Number	99
Purchase Order	99
Invoice Date	68
Total Amount	99
Vendor Name	99
Line Items	93

How to use your model

Power Automate	Model isn't published
Power Apps	Model isn't published

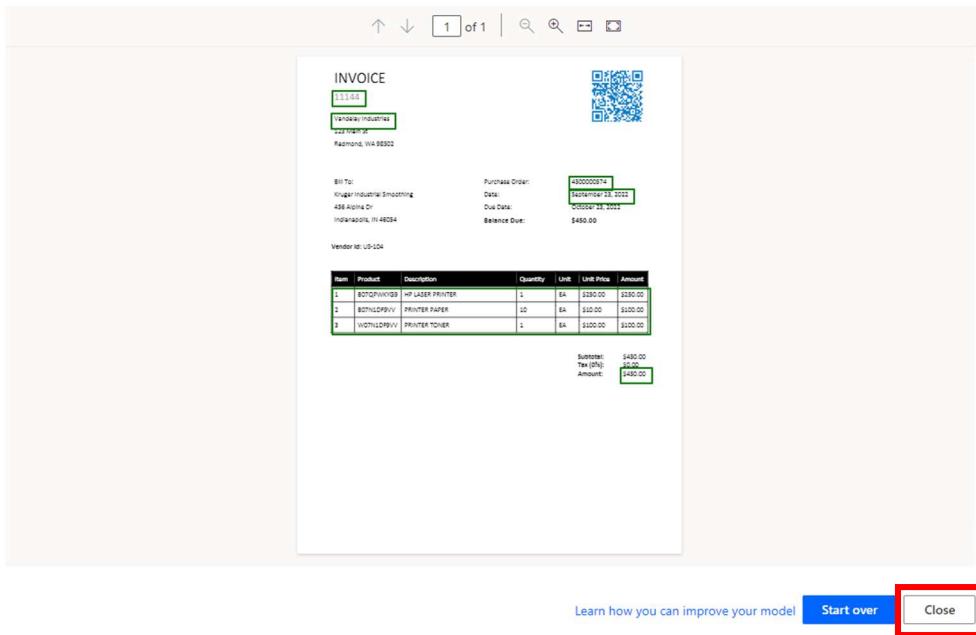
- 40 Click on Upload from my device and select the invoice that is in your Test folder.

Quick test



40. Verify the test. You can mouse over the green boxes and confirm the data is correct.
Click on **Close**.

Quick test



41. Publish the model. We are now ready to publish the model. Click on **Publish**.



42. Add AI Builder Model to our solution. Click on **Solutions**, select the SAP Integration solution, click on the **ellipsis (...)** and click on **Edit**.

The screenshot shows the Power Automate interface with the 'Solutions' tab selected. A list of solutions is displayed, including 'Power Platform Conference Workshop - UserXX', 'Power Platform Conference Workshop - BAB', 'SAP Integration', 'Power Platform Conference Workshop BB', 'Power Platform Conference Workshop', 'SAPUserSearch', 'DVTest Solution', 'SAP ConnectorDemo', 'IDW-BB2', 'IDW-BB', 'Integration Developer Workshop', and 'Power Apps Checker Base'. A context menu is open over the 'SAP Integration' row, with the 'Edit' option highlighted. Other options in the menu include 'Delete', 'Export solution', 'Solution checker', 'Show dependencies', 'See history', 'Clone', 'Apply Upgrade', 'Translations', and 'Settings'. The 'Solutions' button in the left sidebar is also highlighted.

- 43 Edit the cloud flow. Click on the CreateVendorInvoice cloud flow, the ellipse (...) and then Edit.

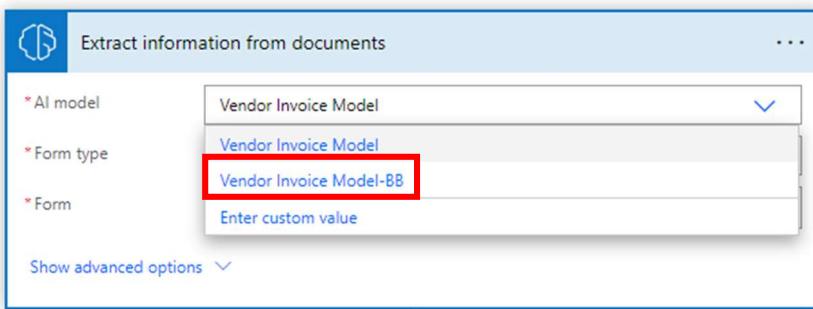
The screenshot shows the Cloud Flow list. A context menu is open over the 'CreateVendorInvoice' cloud flow, with the 'Edit' option highlighted. Other options in the menu include 'Details', 'See analytics', 'Turn off', 'Advanced', and 'Remove'.

Overview of Cloud Flow. This flow is triggered when an email arrives in the Inbox. We use variables to capture information about each invoice that comes from the data

extracted with our AI Builder Model. In the interest of time, we will update certain parts of this flow.

43. Expand the **Apply to each ATTACHMENT** action. Add your new AI Model into the Extract Information from documents action. In the **AI Model** field, select the **Vendor Invoice Model-XX** you just created.

NOTE: We have a completed AI Builder Model in the solution for you to use if you were unsuccessful in building your model. Please choose **Vendor Invoice Model** if needed.



44. Scroll down to the **Apply to each Attachment** action and expand the **Set HEADER** Action. Update the schema in the Value section to pull data from our AI model. In the **Order** name/value pair, insert **Purchase Order Value** from Dynamic Content. In the **Total Price** name/value pair, insert **Invoice Total Value** from Dynamic Content.

Name	Value
Header	{ "Order": "Purchase Order Value", "TotalPrice": "Invoice Total Value" }

Dynamic content pane:

- Invoice Total value
- Purchase Order value

45. Expand the **Apply to each item** action and then expand the **Append to ITEMS** action. We will update the name/value pair the same way as the step above. Using the table below, update values with the following Dynamic Content from the AI Model.

Name	Dynamic Content
Description	Line Items Description Value
Quantity	Line Items Quantity Value
Item	Line Items Item Number Value
Unit	Line Items Unit Value
TotalPrice	Line Items Total Value



The screenshot shows the SAP Cloud Flow editor. A central action card for 'Append to ITEMS' is open. In the 'Value' field of this card, a JSON object is defined:

```
{
  "Description": "Line Items Des...",
  "Quantity": "Line Items Qua...",
  "Item": "Line Items Item...",
  "Unit": "Line Items Unit...",
  "TotalPrice": "Line Items Tota..."
}
```

To the right of the action card, a sidebar titled 'Add dynamic content from the apps and connectors used in this flow.' lists several items. One item, 'Line Items Total value', is highlighted with a red box. Below it, four other items are also highlighted with red boxes: 'Line Items Unit value', 'Line Items Quantity value', 'Line Items Description value', and 'Line Items Item Number value'.

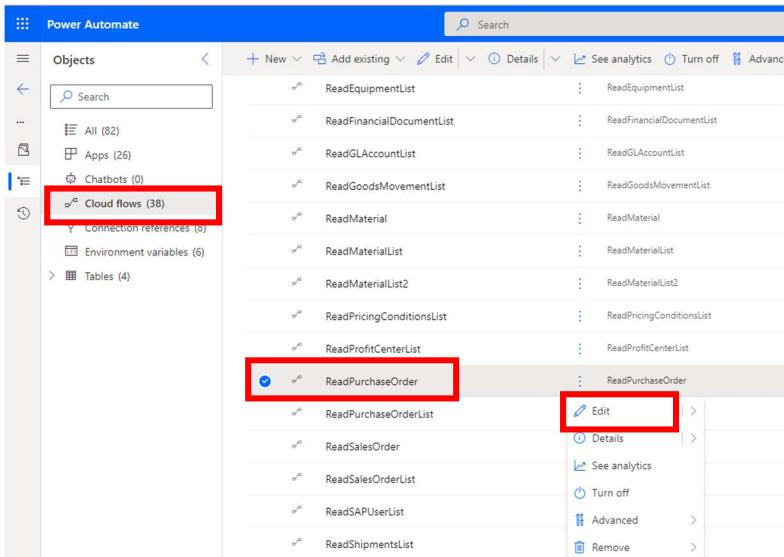
46. Save the Cloud flow. Click on **Save**.

We have now finished updating the cloud flow that will use the AI Builder model to extract data from invoice that are received via email

Exercise 9: Update purchase order cloud flow to extract more information.

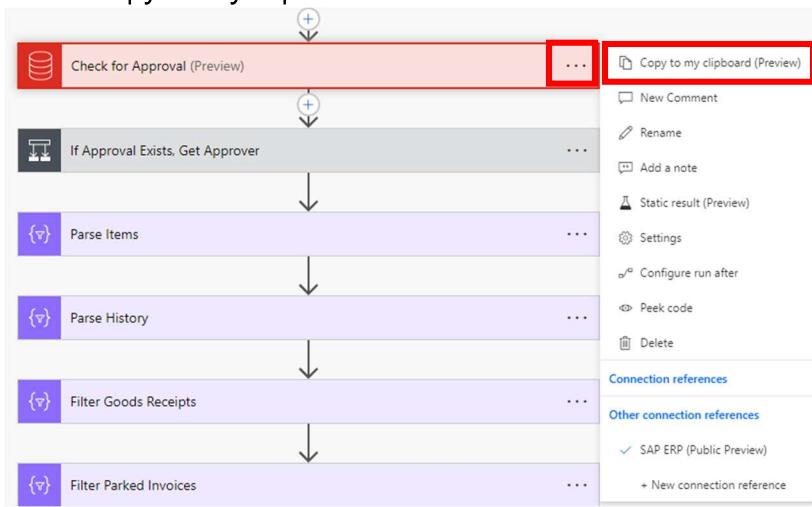
We are now going to update the ReadPurchaseOrder cloud flow to extract the name of the person who approved the Purchase Order.

1. In the SAP Integration solution, filter on Cloud Flows and select the ReadPurchaseOrder cloud flow. Click on the ellipse (...) and then Edit.

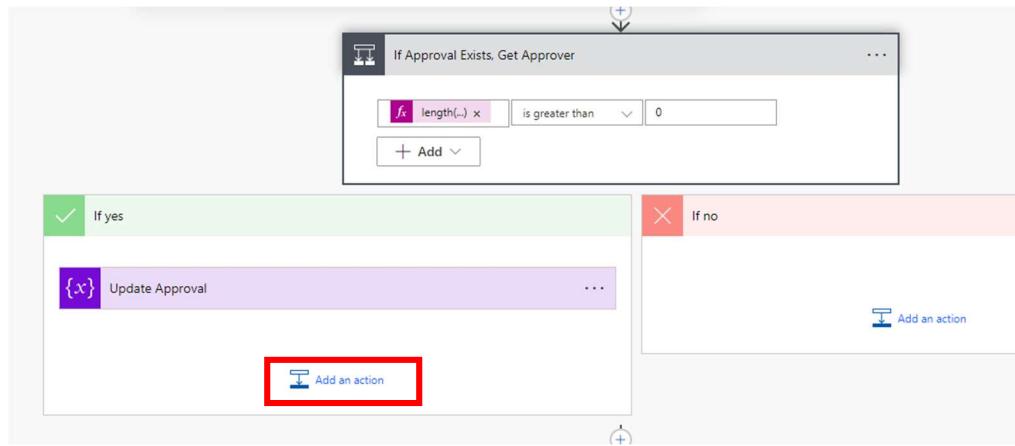


2. Overview - This cloud flow has 2 variables that will capture information if a purchase order has been approved. The Check for Approval action connects to SAP to check if the PO has been approved based on the value for the field being greater than 0. If it is, then we update the Approval variable. We are now going to add a step to get the name of the Approver and update that variable.
3. Copy the Check for Approval Action. We will modify this action that we have already created that read a database table in SAP. Scroll down through the cloud flow to find the

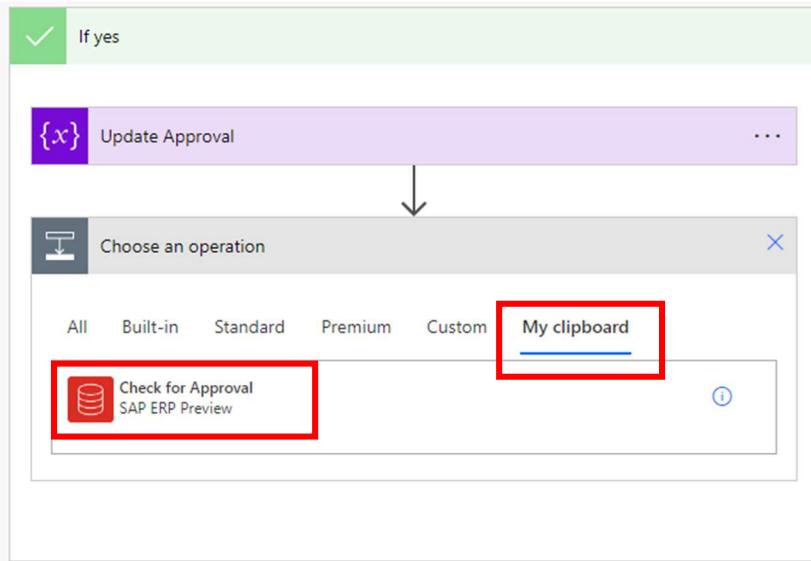
Check for Approval action. Click on the ellipse (...) for the Check for Approval action and select Copy to my clipboard.



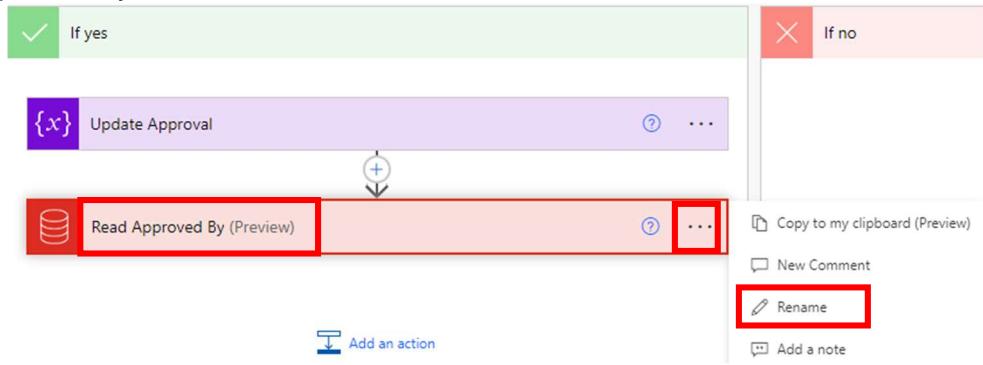
4. Expand the If Approval Exists, Get Approver action. Click on the Add an Action under the If Yes condition.



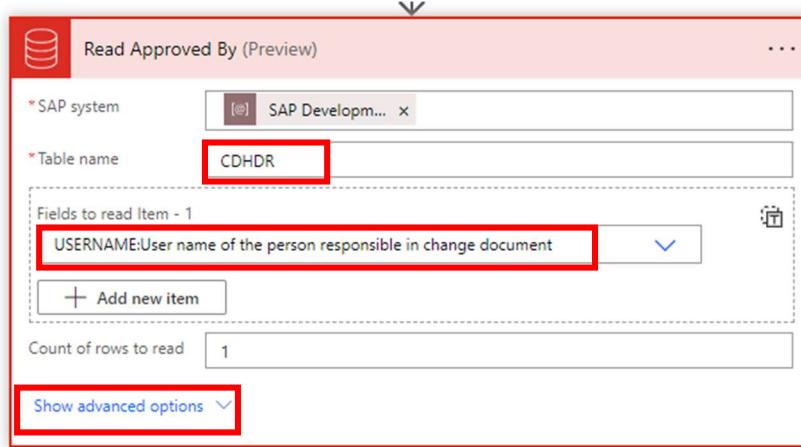
5. Select My Clipboard and then select the Check for Approval action that was copied to the clipboard.



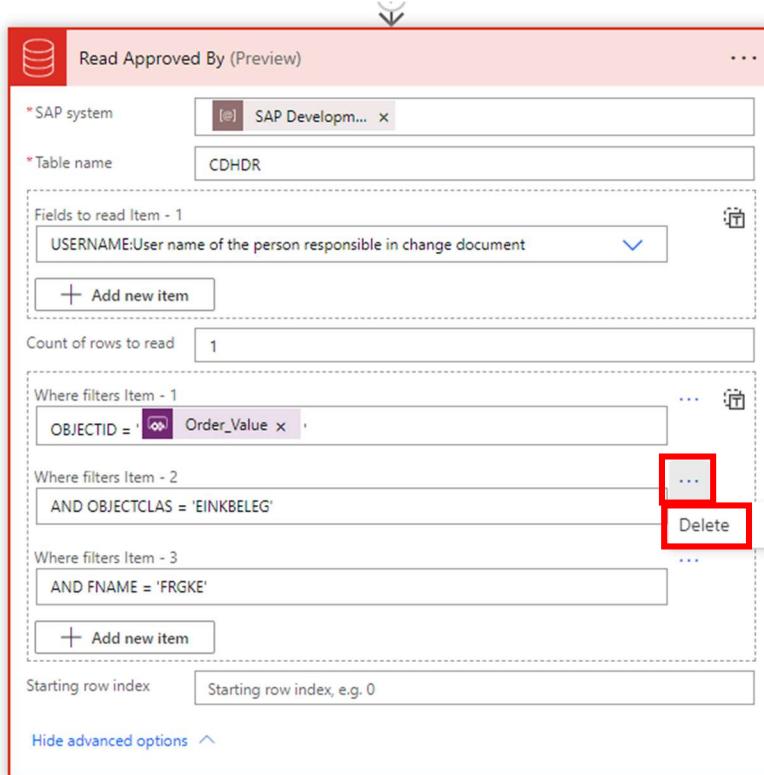
6. Rename the Action. Click on the ellipse (...) and then **Rename**. Rename to Read Approved By.



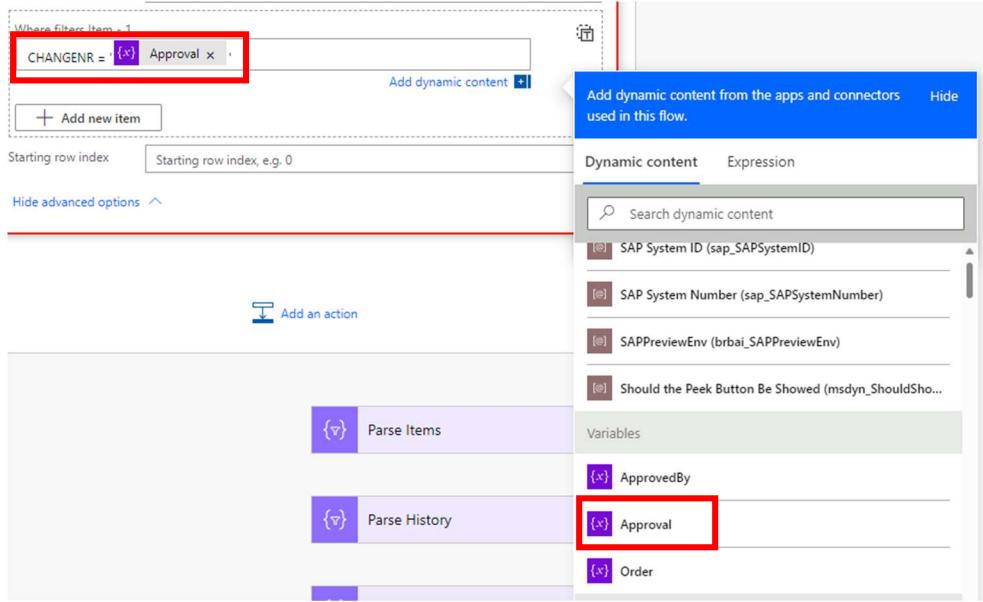
7. Configure the action. Expand the Action and update the fields to the content below:
- Change the **Table name** to **CDHDR** (Change Header Table in SAP)
 - Change the **Fields** to **Read Item – 1** to **USERNAME**
 - Click on **Show advanced options**.



8. Delete additional criteria. Click on the ellipse (...) for the Where filters Item - 2 and then Delete. Repeat for Where filters Item – 3.



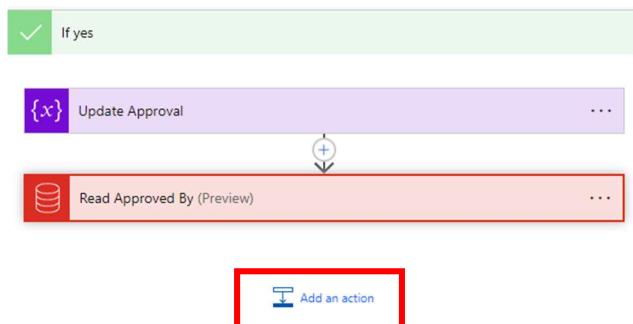
9. Update main criteria. Type in the following for the Where filters Item -1.
CHANGENR = 'Approval' where Approval is coming from Variables in Dynamic Content.



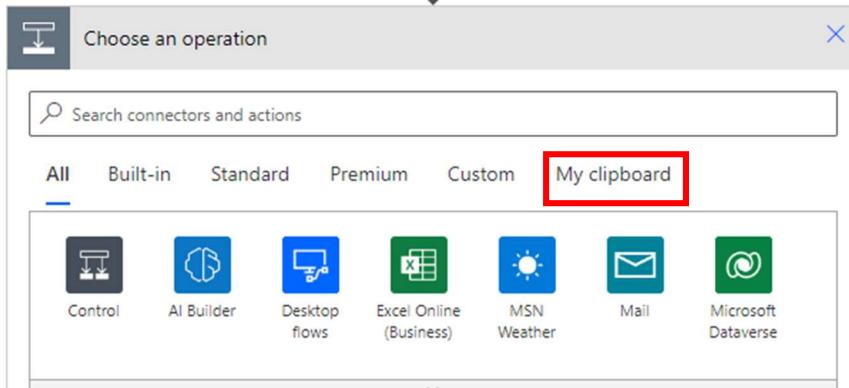
10. Add new action to update Approved By variable. We can copy the Update Approval action and modify. Click on the ellipse (...) for Update Approval and click on Copy to my clipboard.



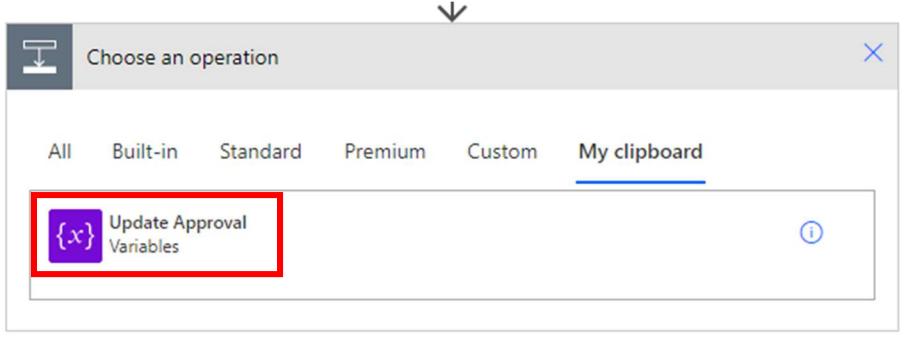
11. Click on Add an Action.



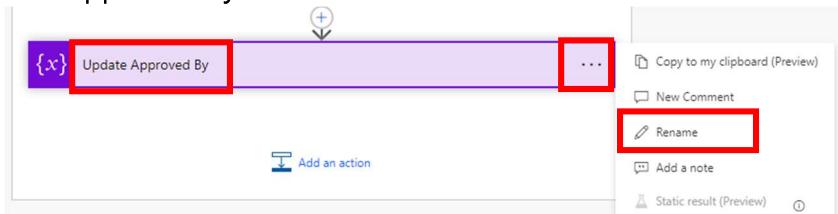
12. Click on My clipboard



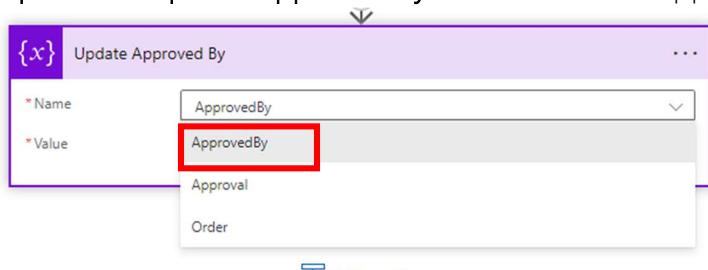
13. Select the **Update Variable** action from the clipboard.



14. Rename the action. Click on the ellipse (...) and then **Rename**. Rename the action **Update Approved By**.



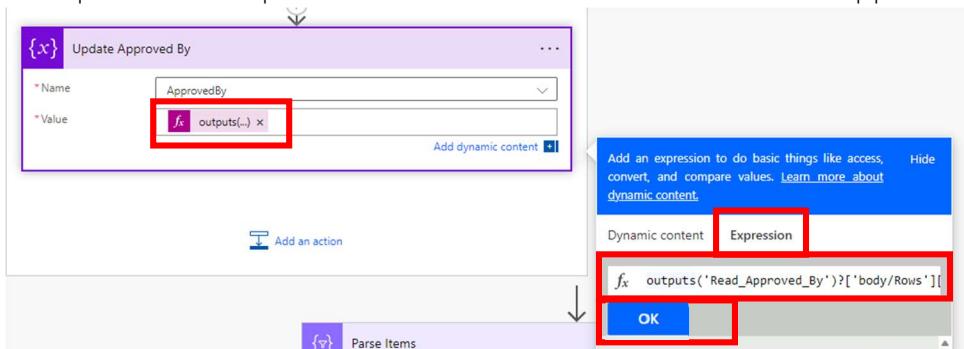
15. Expand the **Update Approved By** action. Choose **Approved By** from the drop down.



16. Modify the Expression for Value. Click on the current **Outputs formula** to pull up the Expression. **Delete** the current expression and **Type in** the expression below and then click on **OK**.

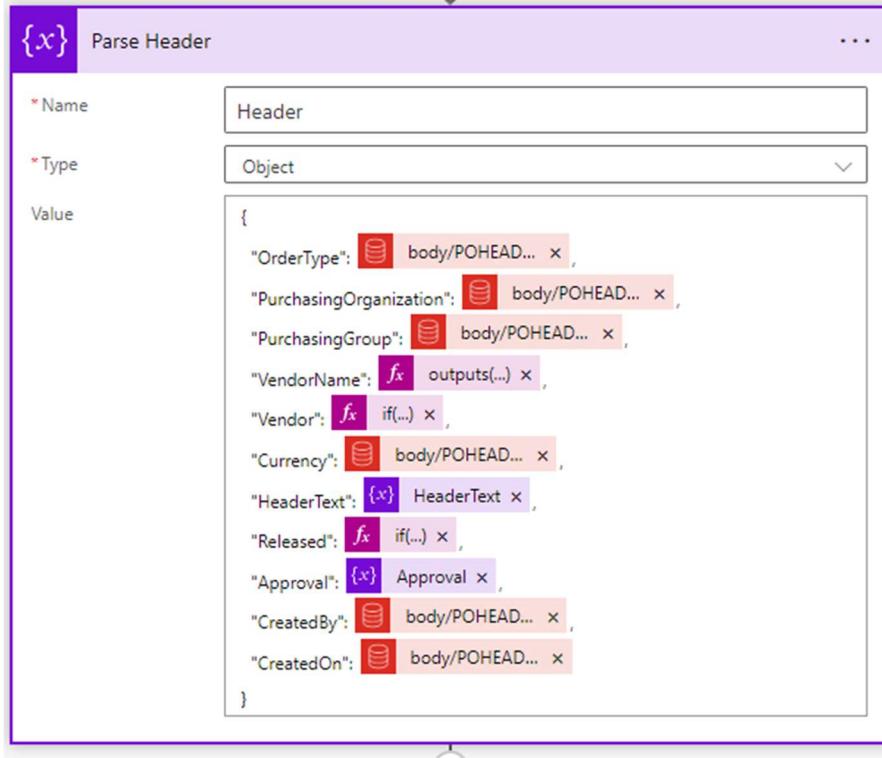
```
outputs('Read_Approved_By')?['body/Rows'][0]['USERNAME']
```

We updated the expression to match information in our Read Approved By action.



17. Save the Flow. Click on **Save**.

18. Update the Parse Header Schema to include Approved By. Now we need to update the Parse Header Schema to include our Approved By variable. Scroll down in the cloud flow and **Expand** the Parse Header action.



NOTE: If you get an error message in the Parse Header, refresh your browser to reload the components. You may have to get back into your SAP Integration solution and ReadPurchaseOrder cloud flow.



19. Copy the Approval name/value pair and paste below the existing Approval.

The screenshot shows the SAP Integration Studio interface for configuring a 'Parse Header' step. The 'Name' is set to 'Header' and the 'Type' is 'Object'. The 'Value' field contains the following JSON code:

```
{
  "OrderType": body/POHEAD... ,
  "PurchasingOrganization": body/POHEAD... ,
  "PurchasingGroup": body/POHEAD... ,
  "VendorName": outputs(...) ,
  "Vendor": if(..) ,
  "Currency": body/POHEAD... ,
  "HeaderText": {x} HeaderText ,
  "Released": if(..) ,
  "Approval": {x} Approval ,
  "Approval": {x} Approval ,
  "CreatedBy": body/POHEAD... ,
  "CreatedOn": body/POHEAD... ,
}
```

The 'Approval' field is highlighted with a red box.

20. Update the copied name/value pair to the following where the highlighted text is coming from Dynamic Content.

"ApprovedBy": ApprovedBy,

The screenshot shows the SAP Integration Studio interface for configuring a 'Response' step. The 'Value' field contains the following JSON code:

```
{
  "VendorName": outputs(...) ,
  "Vendor": if(..) ,
  "Currency": body/POHEAD... ,
  "HeaderText": {x} HeaderText ,
  "Released": if(..) ,
  "Approval": {x} Approval ,
  "ApprovedBy": {x} ApprovedBy ,
  "CreatedBy": body/POHEAD... ,
  "CreatedOn": body/POHEAD... ,
}
```

A dynamic content search dialog is open on the right, showing results for 'approv'. The variable '{x} ApprovedBy' is highlighted with a red box.

NOTE: Reminder format is “Name”: Value,
Make sure you have a comma after the new name/value pair.

21. Update the Response action. The response section is what Power Apps will receive, so we need to update the schema to include our ApprovedBy variable. Expand the **Response** section and click on **Show Advanced Options**.

The screenshot shows the 'Response' configuration screen in Power Apps Studio. The 'Body' schema is expanded to show a JSON object with various properties: Order, Header, Items, Invoices, ParkedInvoices, GoodsReceipts, and Payments. Below the schema, the 'Response Body JSON Schema' is displayed as a JSON object with properties for Order and Header. A yellow box highlights the 'Header' property in the schema. At the bottom, there is a 'Generate from sample' button and a 'Hide advanced options' link.

```

{
    "Order": PO_NUMBER,
    "Header": {x} Header,
    "Items": (v) Output,
    "Invoices": (v) Body,
    "ParkedInvoices": (v) Body,
    "GoodsReceipts": (v) Body,
    "Payments": {x} Payments
}

{
    "type": "object",
    "properties": {
        "Order": {
            "type": "string"
        },
        "Header": {
            "type": "object",
            "properties": {
                "OrderType": {
                    "type": "string"
                }
            }
        }
    }
}

```

22. Update the Response Body JSON Schema. We need to update the Header section of the schema to add the Approved by information. Click in the **Response Body JSON Schema** and scroll down to the bottom of the **header** section and copy the **CreatedBy** section based on the screen shot below.

```
Response Body JSON Schema
{
    "CreatedOn": {
        "type": "string"
    },
    "CreatedBy": {
        "type": "string"
    }
},
"Items": {
    "type": "array",
    "items": [
        {
            "type": "string"
        }
    ]
}
```

23. Paste below CreatedBy

Response Body JSON Schema	
	<pre> "CreatedOn": { "type": "string" }, "CreatedBy": { "type": "string" }, "CreatedBy": { "type": "string" } } }</pre>

24. Update the section. Change the name to **ApprovedBy**.

Response Body JSON Schema		
<pre> "CreatedOn": { "type": "string" }, "CreatedBy": { "type": "string" }, "ApprovedBy": { "type": "string" } } }</pre>		

25. Save the cloud flow. Click on **Save**.

Exercise 10: Update Power App

We are now going to update the Purchase Orders canvas app to include the Approved By field.

1. Go back to your **SAP Integration** solution. Filter on **Apps**, select the **Purchase Orders** canvas app and click on **edit**.

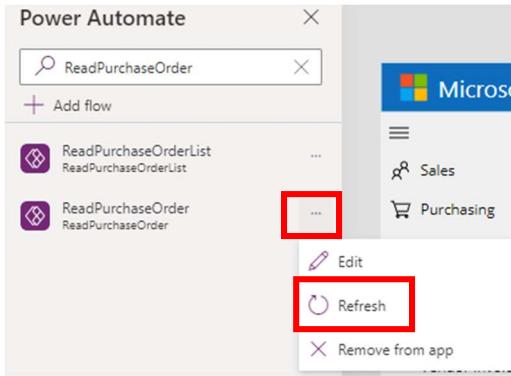
The screenshot shows the SAP Power Platform Objects list. On the left, there's a sidebar with categories like 'All (82)', 'Apps (26)' (which is highlighted with a red box), 'Chatbots (0)', 'Cloud flows (38)', 'Connection references (8)', 'Environment variables (6)', and 'Tables (4)'. The main area lists various SAP applications with their names and IDs. One item, 'Purchase Orders' (sap_purchaseorders_977e2), is selected and highlighted with a red box. Below it are 'Sales Order Search' and 'Sales Orders'. At the bottom right of the list, there are buttons for 'Edit', 'Play', and 'Monitor'.

Object	ID
Customer Search	sap_customersearch_28642
Delivery Search Default	sap_deliverysearchdefault_5328b
Equipment Search Default	sap_equipmentsearchdefault_6a403
Financial Documents Search Default	sap_financialdocumentssearchdefault_759cf
GL Account Search	sap_glaccountsearch_33c72
Goods Movement Search Default	sap_goodsmovementsearchdefault_17b08
Header and Menu Components	sap_headerandmenucomponents_ae407
Material Search	gilman_materialsearch_c63b3
Pricing Condition Search Default	sap_pricingconditionsearchdefault_f1c6a
Profit Center Search	sap_profitcentersearch_c6694
Purchase Order Search	sap_purchaseordersearch_53731
Purchase Orders	sap_purchaseorders_977e2
Sales Order Search	
Sales Orders	

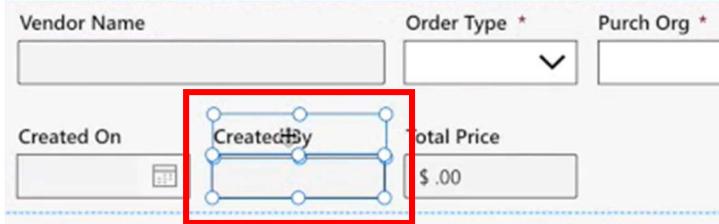
2. Refresh Power Automate flow. Anytime you have updated a flow, you need to refresh it in your Power App. Click on **Power Automate icon** on the left-hand side and search for **ReadPurchaseOrder**.

The screenshot shows the Power Automate interface. On the left, there's a sidebar with icons for 'Power Automate', 'Add flow', 'ReadPurchaseOrderList', 'ReadPurchaseOrder', and a refresh button (indicated by a double arrow icon, which is highlighted with a red box). The main area has a search bar where 'ReadPurchase' is typed (also highlighted with a red box). Below the search bar, there are two flow items: 'ReadPurchaseOrderList' and 'ReadPurchaseOrder'.

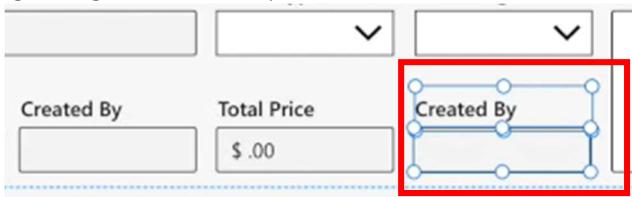
3. Click on the **ellipsis (...)** and then **Refresh**.



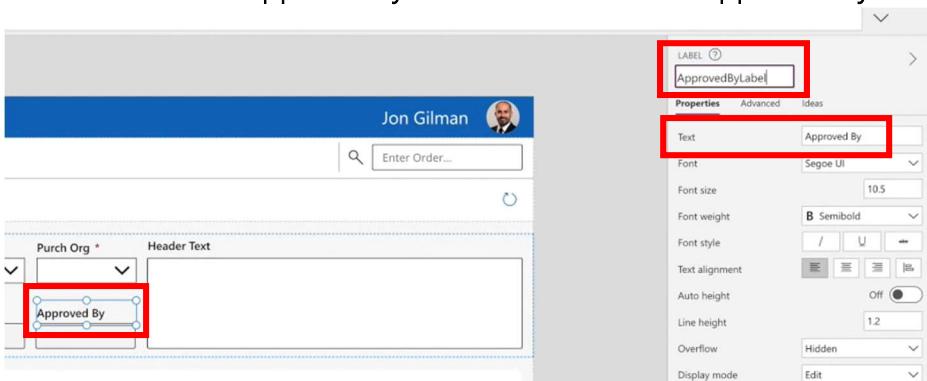
- Add a new set of labels for the Approved By information. Select both the **CreatedBy** label and **CreatedBy** Text Input box and copy them (Ctrl-C).



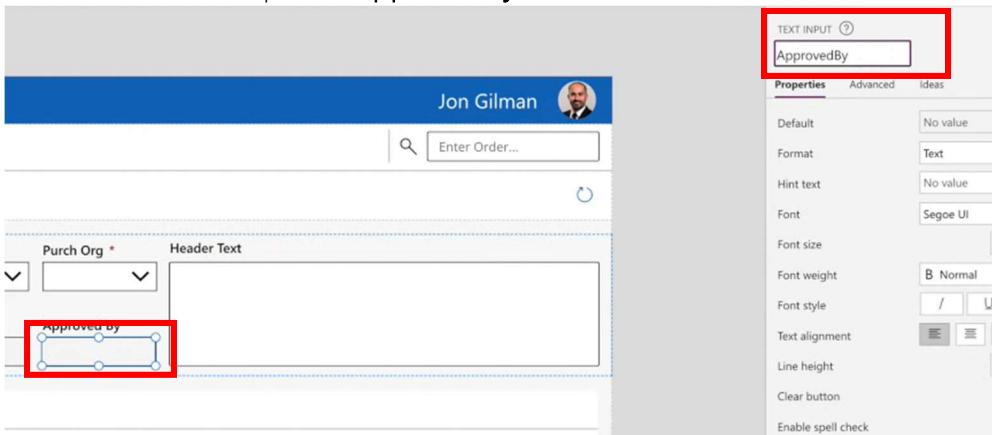
- Paste (Ctrl-V) them and then drag and drop both items to the right of the Total Price using the guidelines to position in-line with the rest of the fields.



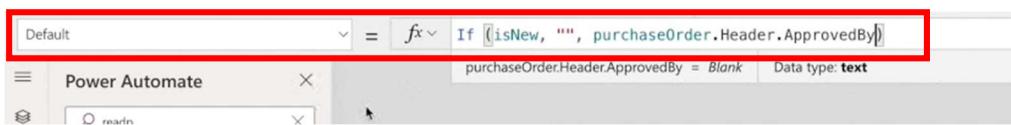
- Select the copied **CreatedBy** label. In the right-hand properties section, change the Name of the label to **ApprovedByLabel** and the Text to **Approved By**.



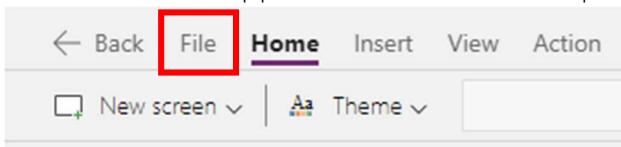
7. Select the copied **CreatedBy** Text Input. In the right-hand properties section, change the Name of the text input to **ApprovedBy**.



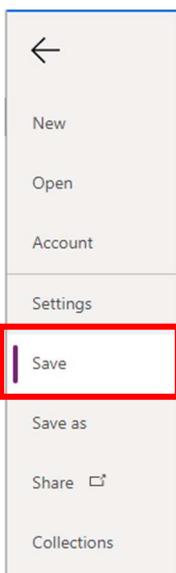
8. With the **CreatedBy** Text Input still selected, update the **Default formula** to **ApprovedBy** as indicated in the screen shot below.



9. Save the canvas app. Click on **File** in the top ribbon.



Then click on **Save**.



10. Publish the canvas app. Click on Publish.

Purchase Orders

Environment: ArmyDemo

Saved: 9/15/2022, 2:47:59 PM

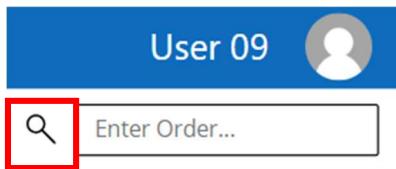
✓ All changes are saved.



11. Play the canvas app. Click on the Play button to test the app.



12. Select the Purchase Order and verify the Approved By field is filled in. You can click on the Magnifying Glass to find the PO number you created in the previous exercise.



Select the purchase order from Vandelay Industries that you created in the earlier exercise..

Recently viewed	All	
Order	Vendor Name	Last Viewed
4500000588	VANDELAY INDUSTRIES	9/22/2022 12:28 PM

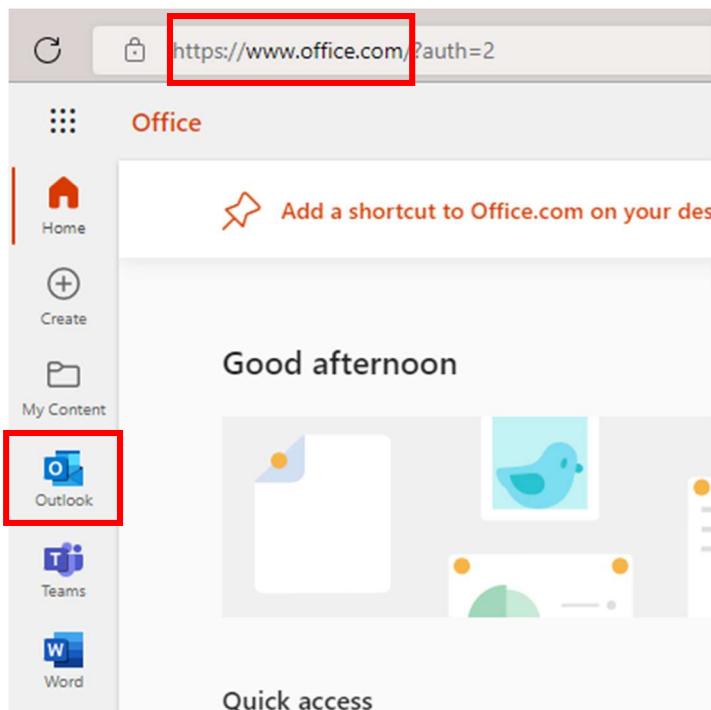
13. Confirm the Approved By field is filled out with your SAP user name.

A screenshot of a purchase order creation form. It includes fields for Vendor (197), Vendor Name (VANDELAY INDUSTRIES), Order Type (NB), Purch Org (HLTH), Purch Group (HLT), Created On (10/16/2022), Created By (USER01), Total Price (\$ 450.00), and Approved By (USER01). The "Approved By" field is highlighted with a red box.

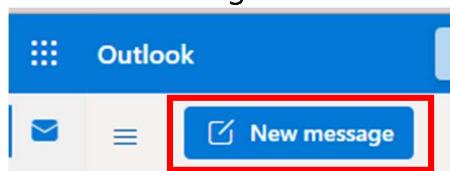
Exercise 11: Finish the Procure to Pay journey

We are now going to finish the Pay to Procure journey by simulating the vendor sending us an email with the invoice as an attachment. This will kick off our cloud flow to run our AI Builder Model to pull the information from the invoice. This information is returned to SAP and a vendor invoice is created in SAP. We will then go back into our Power App and refresh the screen to see a new button is available to confirm the vendor invoice against the purchase order and the goods receipt.

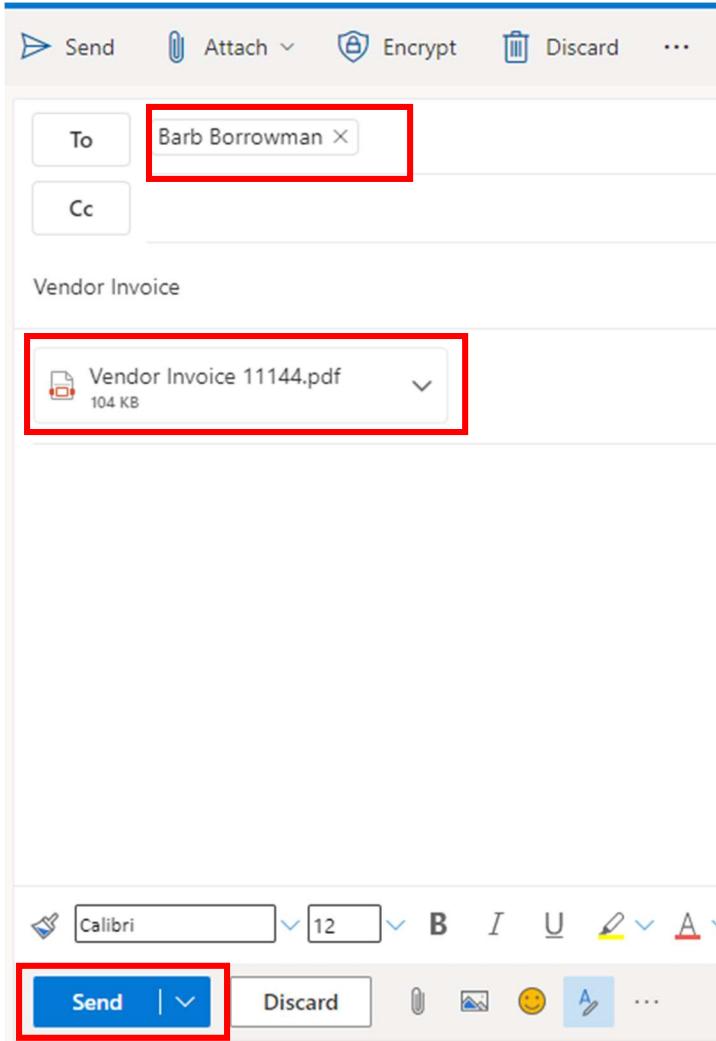
1. Open the O365 portal to access Outlook. Using an InPrivate or Incognito session, browse to <https://www.office.com>. Ensure you are still using the credentials supplied by the instructor. Click on **Outlook**.



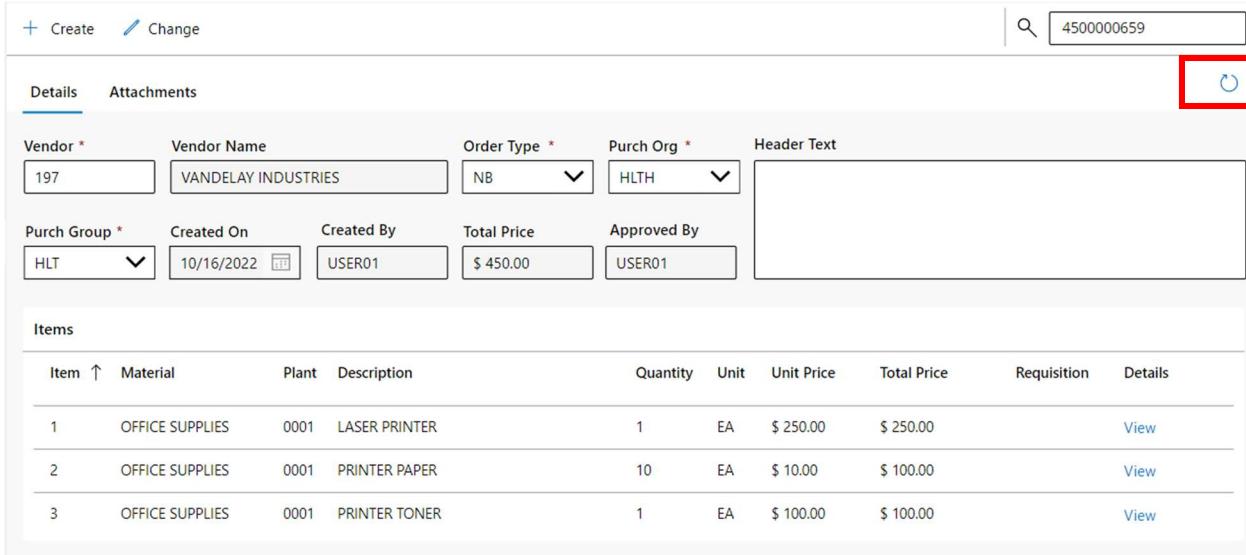
2. Click on **New Message**.



3. Compose email message. Send the message to yourself using the credentials supplied by the instructor, fill out the subject line and attach the vendor invoice we created in the previous exercise. Click on **Send**.



4. Go back to the Power App. Click on the tab that has the Purchase Order canvas app open. Click on the **Refresh** button in the upper right-hand corner.

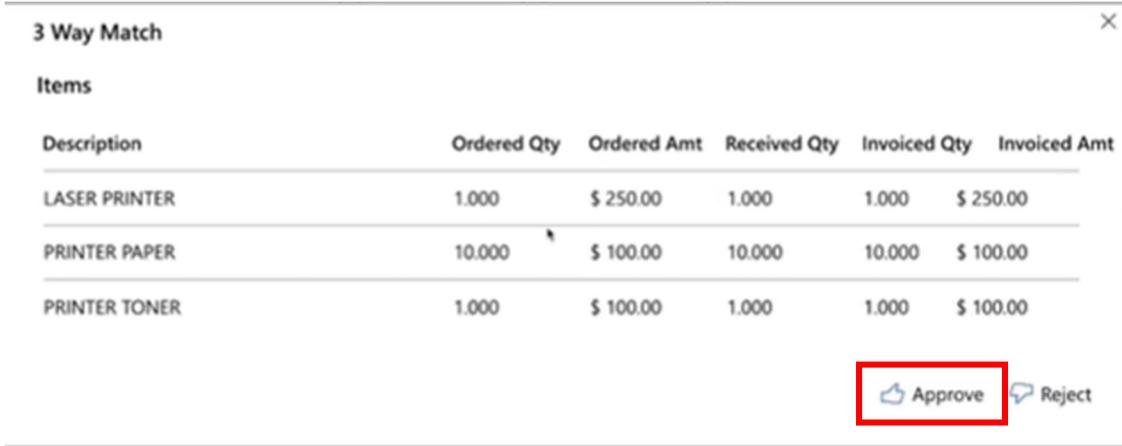


The screenshot shows the SAP Purchase Order creation interface. At the top, there are buttons for 'Create' and 'Change'. A search bar contains the number '4500000659'. Below the search bar are tabs for 'Details' and 'Attachments', with 'Details' being the active tab. The 'Details' section contains fields for Vendor (197), Vendor Name (VANDELAY INDUSTRIES), Order Type (NB), Purch Org (HLTH), Header Text, Purch Group (HLT), Created On (10/16/2022), Created By (USER01), Total Price (\$ 450.00), and Approved By (USER01). A large 'Items' section below lists three items: OFFICE SUPPLIES (Laser Printer, 1 unit, \$ 250.00), OFFICE SUPPLIES (Printer Paper, 10 units, \$ 10.00), and OFFICE SUPPLIES (Printer Toner, 1 unit, \$ 100.00). The 'Details' section ends with a 'Save' button, which is highlighted with a red box.

5. Click on the **Confirm** button.



6. This pulls up the screen that shows the information from the purchase order, goods receipt and vendor invoice. We verify that all quantities match and click on **Approve**.



The screenshot shows the '3 Way Match' screen. It has a header '3 Way Match' and a section titled 'Items'. The table below shows the comparison between Ordered Qty, Ordered Amt, Received Qty, Invoiced Qty, and Invoiced Amt for three items: LASER PRINTER, PRINTER PAPER, and PRINTER TONER. At the bottom right, there are 'Approve' and 'Reject' buttons, with 'Approve' being highlighted with a red box.

Description	Ordered Qty	Ordered Amt	Received Qty	Invoiced Qty	Invoiced Amt
LASER PRINTER	1.000	\$ 250.00	1.000	1.000	\$ 250.00
PRINTER PAPER	10.000	\$ 100.00	10.000	10.000	\$ 100.00
PRINTER TONER	1.000	\$ 100.00	1.000	1.000	\$ 100.00

You have now completed the Using Power Automate End-to-End: Procure to Pay in SAP hands on lab. Congratulations!