SAP Modernization Lab

Goal

In this lab, you will learn how ServiceNow Creator Workflows can help you to modernize ERP processes with the tools and functionalities available on the ServiceNow platform. Be prepared, there are no-code elements, as well as some basic scripting requirements in this lab, but not to worry, I'll be here to guide you through the process.

You'll learn how the platform can be a catch-all for everything above SAP as a system of engagement and to fuel ERP modernization.

Background

A business process refers to a set of activities that have to be performed to complete an end-to-end business scenario. There are multiple business processes in ERPs, the most common ones are listed below:

Name	What is it?
Order to Cash	Order to cash (OTC or O2C) is a set of business processes that involve receiving and fulfilling customer requests

Name	What is it?
Procure to Pay	Procure to pay (PTP or P2P) is the process of requisitioning, purchasing, receiving, paying for, and accounting for goods and services, covering the entire process from point of order right through to payment.
Hire to Retire	Hire to retire (H2R) is a human resources process that includes everything that needs to be done over the course of an employee's career with a company.
Plan to Produce	Plan-to-Produce (Pl2P) encompasses an end-to-end perspective on producing goods. It depicts the shop-floor procedures and focuses on the integration of surrounding aspects into production processes.
Plan to Inventory	Plan to Inventory (P2I) the process of determining the optimal quantity and timing of inventory for the purpose of aligning it with sales and production capacity.
Quote to Cash	Quote-to-cash (QTC) process encompasses many sales, account management, order fulfillment, billing, and accounts receivables functions.
Record to Report	Record to report (R2R) is a finance and accounting management process that involves collecting, processing and presenting accurate financial data.

Often times, there are multiple exceptions outside of an expected business process flow (sometimes refered to as a "Happy Flow") that organizations need to account for - either by customizing the ERP, or relying on manual processes that sit outside of the ERP.

Then, there is also the issue of data availability. There is a learning curve to navigating ERP systems due to the complexity of the business objects, everyday business users might not have access to the right environment and training to use these ERP systems, and need a friendlier way to interact with the data.

ServiceNow Creator Workflows solves these challenges with it's low code capabilities paired with easy to configure integrations to various ERP systems. We will focus on SAP in this lab.

Exercise 1: Delegation of Authority

Introduction

When it comes to automating ERP workflows, nothing is more painful, or more time consuming than the approval cycle. Couple that with a rigid system like SAP, and it becomes difficult to track, let alone automate approvals on everyday transactions like Purchase Orders, Sales Orders, Invoices, etc. A common term for this process is Delegation of Authority (DoA), or Approval Matrix.

Within SAP, there is such a feature, but it is limited in capability and does not scale for exceptional workflows.

Tales from the field

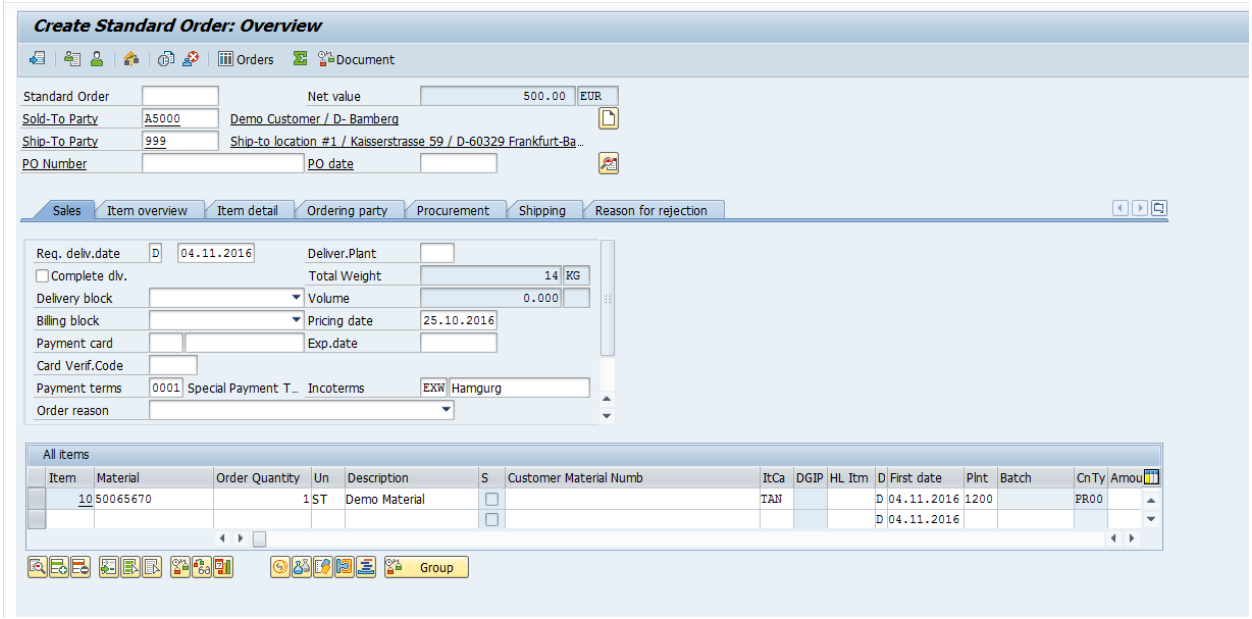
In one of our customer's internal landscape, they reported having 54 different workflows that covers multiple business objects, and their solution to that was over 12,000 lines of code in a custom .NET application for approvals and authority. Some of these workflows require 7 or more levels of approvals, and any changes to personnel or approval logic requires a major rewrite of the code, which they currently spend over 30 days a year doing. Approvers only have one desktop interface they can use to run these approvals. Obviously not scalable, and a terrible user experience!

Sales Order Amount	Sales Order Type	Approval Group
< $5,000	Standard sales order	Finance manager
$5,000 - $50,000	Standard sales order	Finance controller
> $50,000	Standard sales order	CFO
Any amount	Rush order	CFO

> The sample that we will work with, but it usually gets a lot more complicated than this ☠️

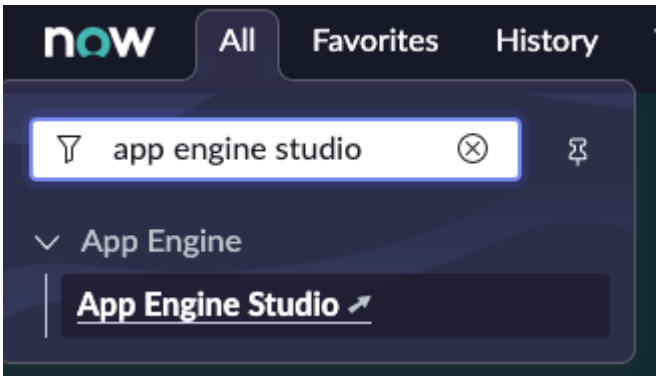
Thankfully, we have the solution for this, so let's start by showing how you can run these approval matrices on top of the Sales Order documents from SAP.

For easy reference, this is how the Standard Order (Sales Order) document looks like in SAP ECC. You navigate to this module by entering the T code *VA01*. The T Code (Transaction code) is a 4 digit shortcut key to access a requested transaction, sort of like carrying out an *incident.list* command in the ServiceNow UI.

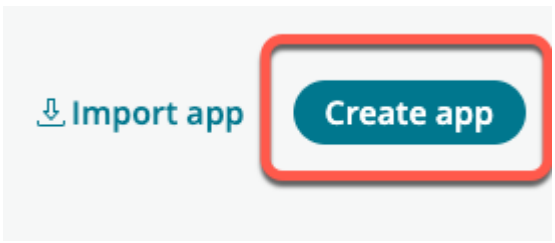


Let's start

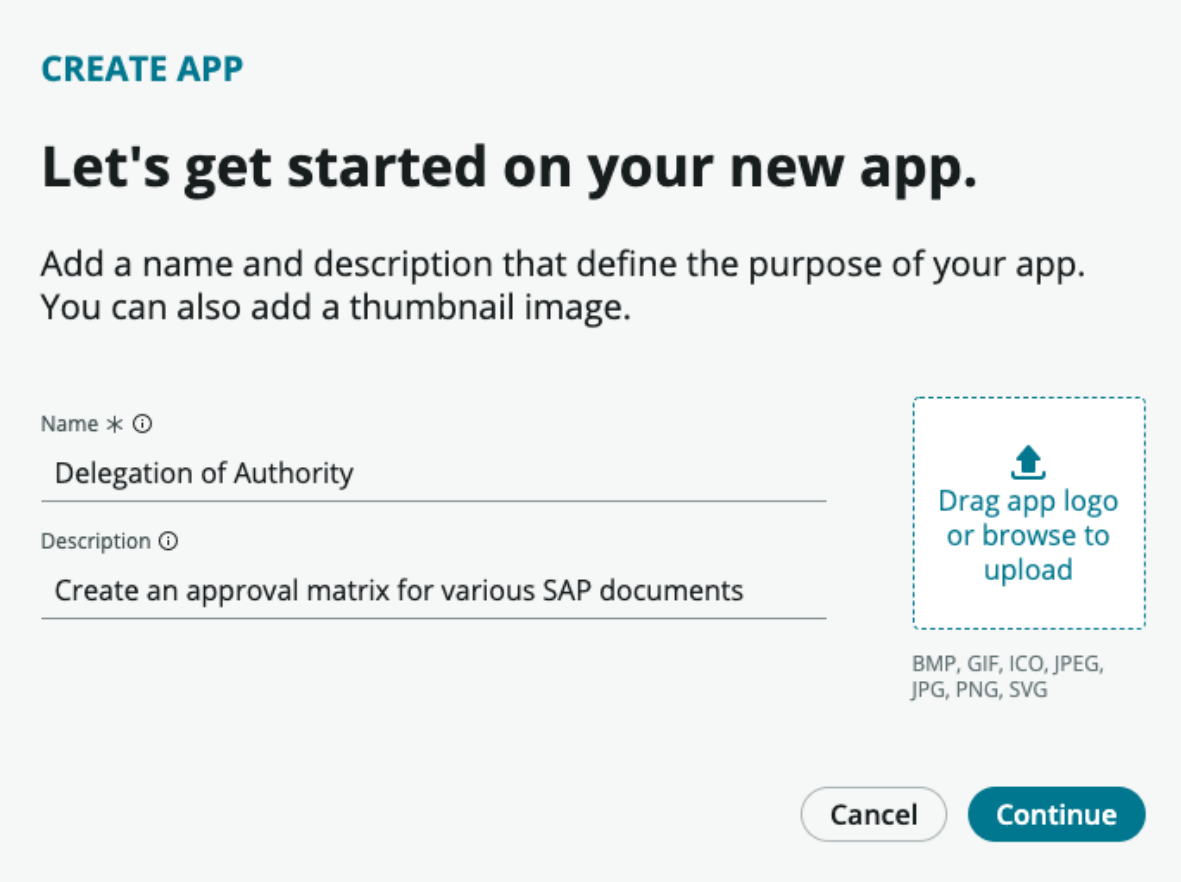
1. Under **All**, search and navigate to **App Engine Studio**



2. Click **Create app** on the top right of the screen



3. On the Create App page, name the app "Delegation of Authority", and for description, enter "Create an approval matrix for various SAP documents"

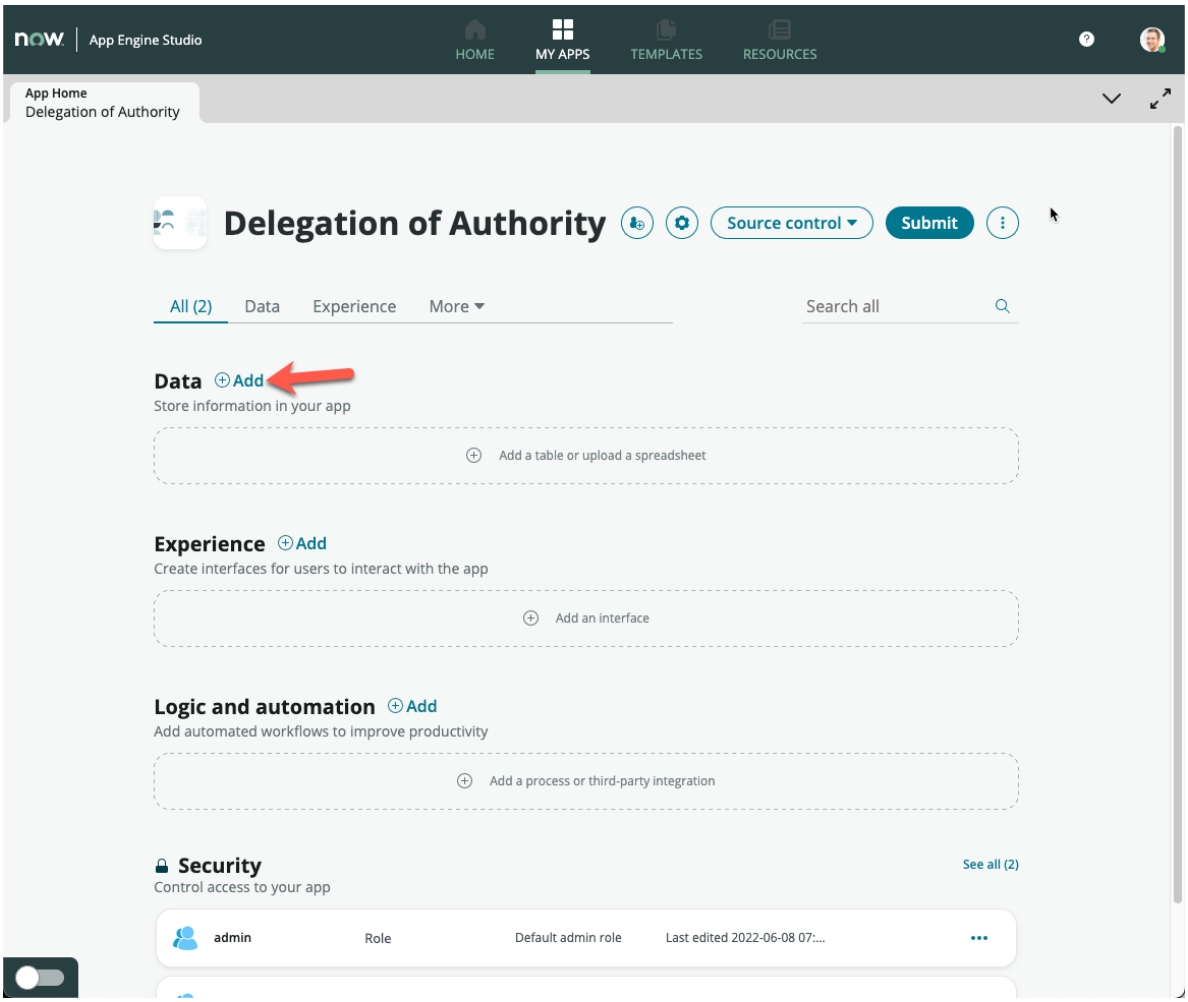


4. Click **Continue**
5. Leave the default roles - *admin* and *user*, and click **Continue**
6. Click **Go to app dashboard**

Create a Sales Order table

We will now create a table to store Sales Order document data from SAP.

1. Under **Data**, click **Add**

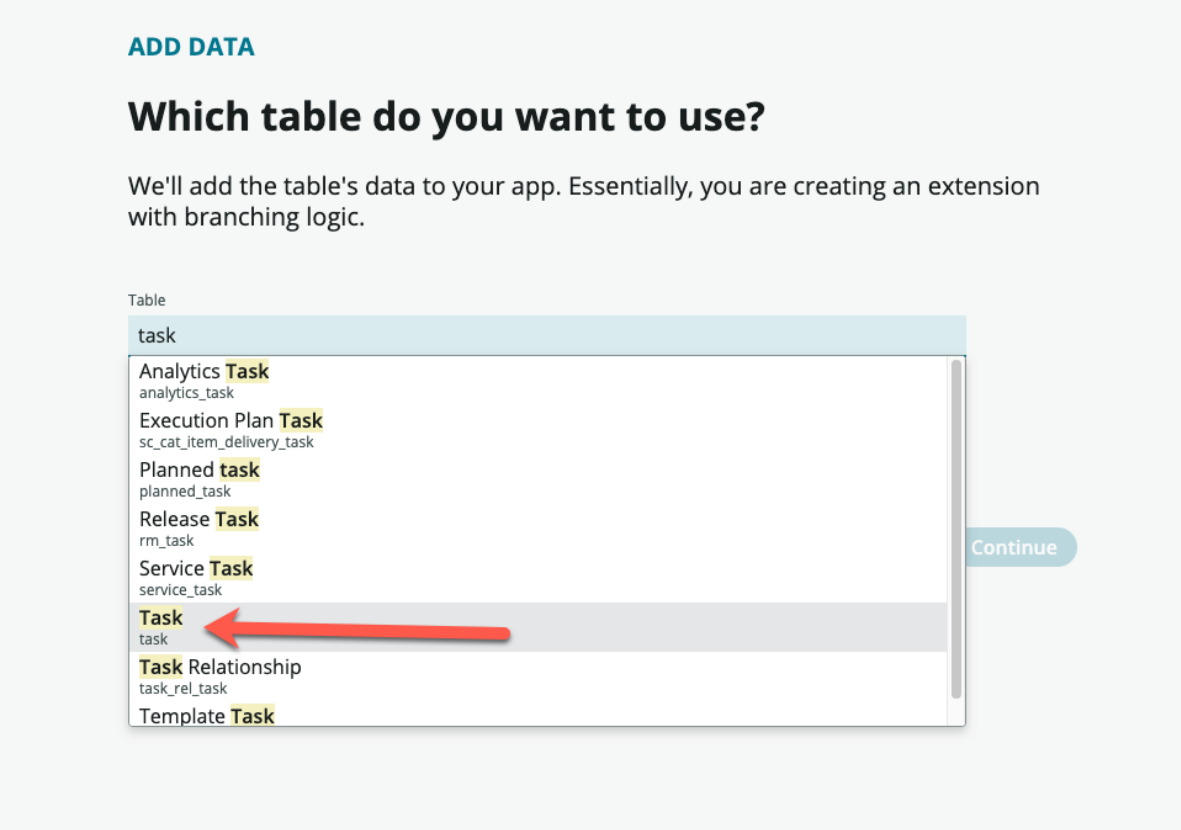


2. Click **Create a table**

3. Click **Get started**

4. On the *Add Data* page, click **Create from an existing table**

5. On the next page, search and select **Task**

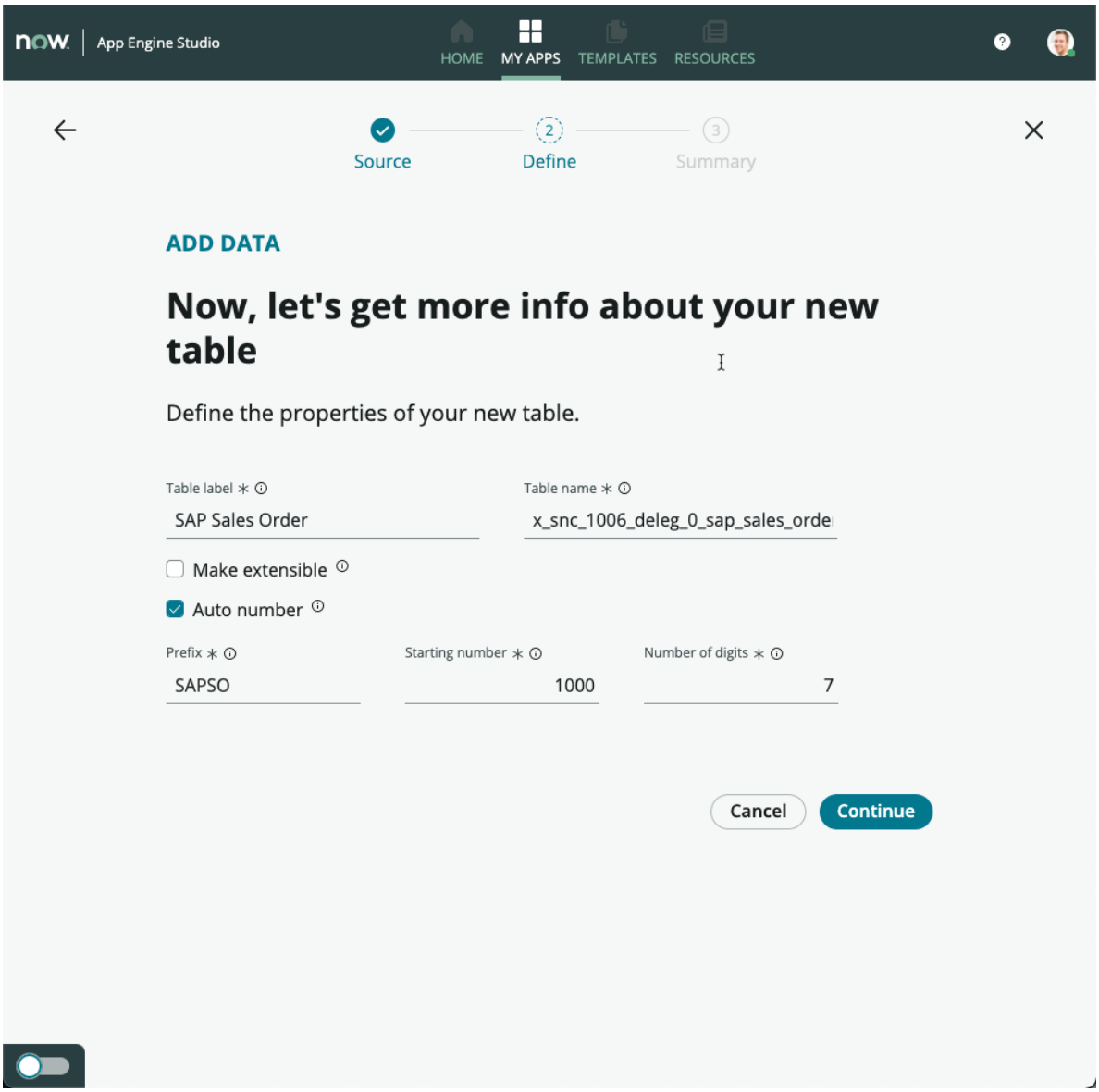


6. Click **Continue**

7. For Table label, enter **SAP Sales Order**. Table name should be auto-populated.

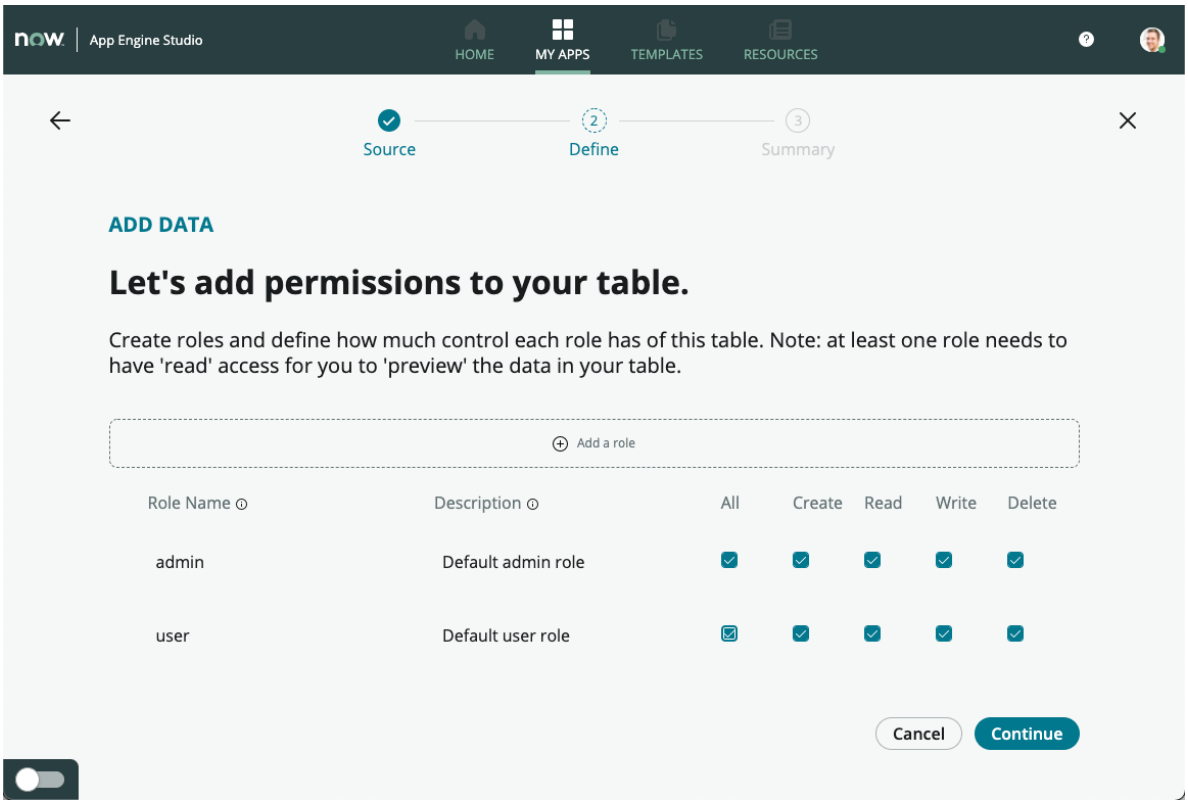
8. Check **Auto number**

9. For Prefix, enter **SAPSO**



10. Click **Continue**

11. Allow all access for *admin* and *user*, then click **Continue**



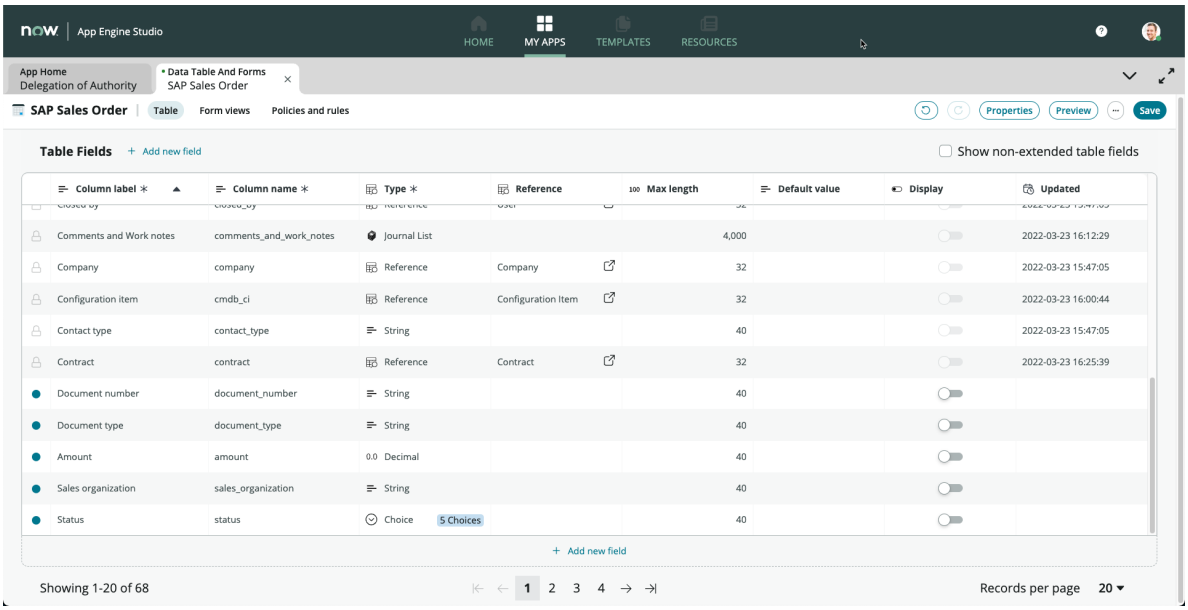
12. Click **Edit table**

13. Close the pop-up dialog

14. You should now be on tha *Table Builder* interface. Click **Add new field**, and add the following fields:

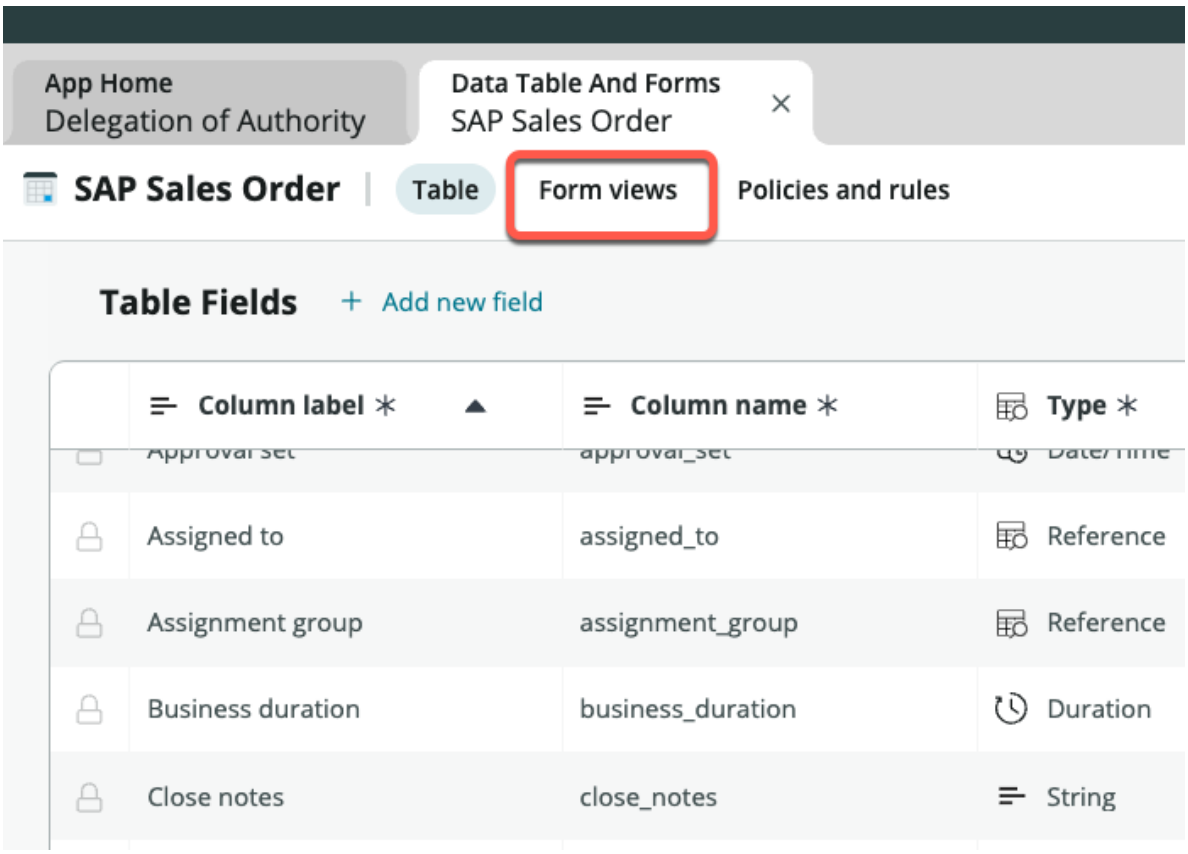
Column label	**Type**
Document number	String
Document type	String
Amount	Decimal
Sales organization	String
Status	Choice (Dropdown with --None--) : New, Approval triggered, Approved, Rejected, Updated SAP

15. Your screen should look like this



16. Click **Save**

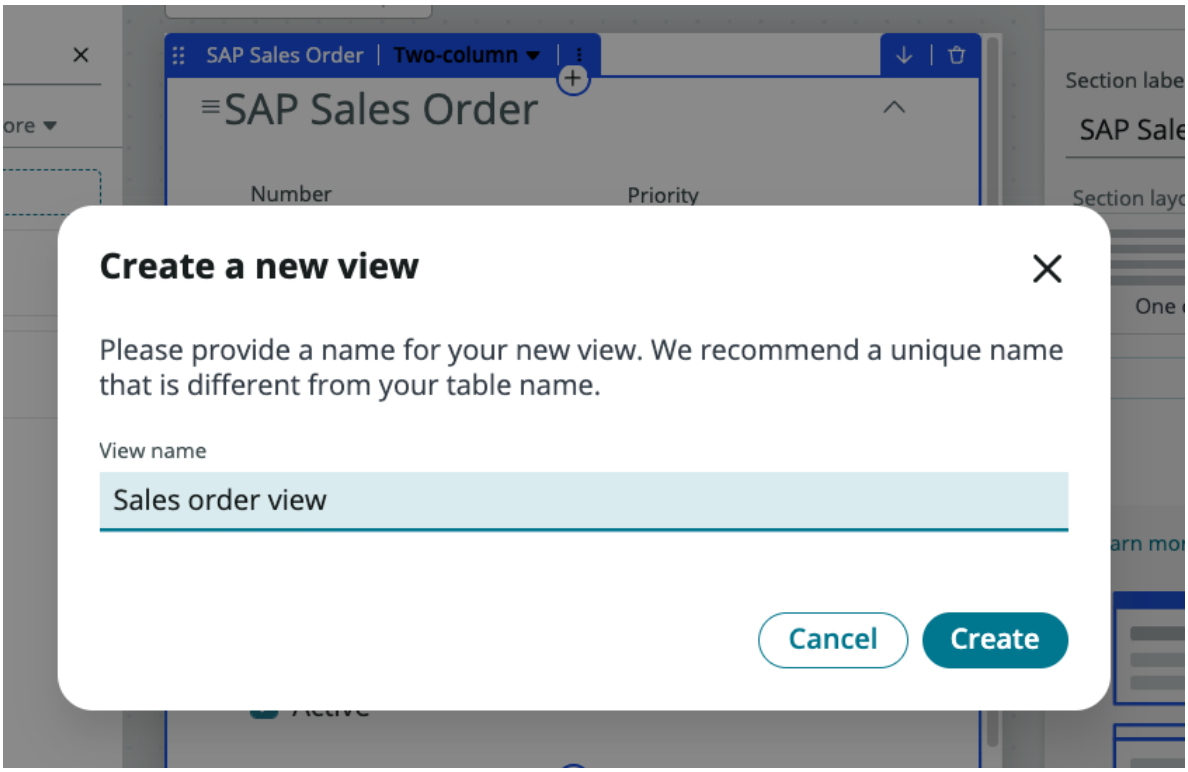
17. Click **Form views** on the top of the screen



18. Click the **Default view** dropdown

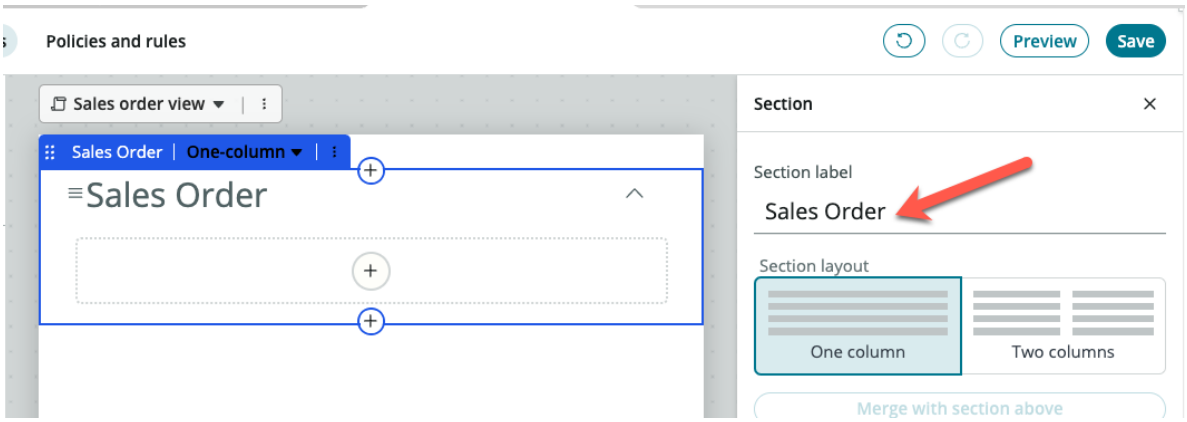
19. Click **+ Create**

20. Enter **Sales order view**



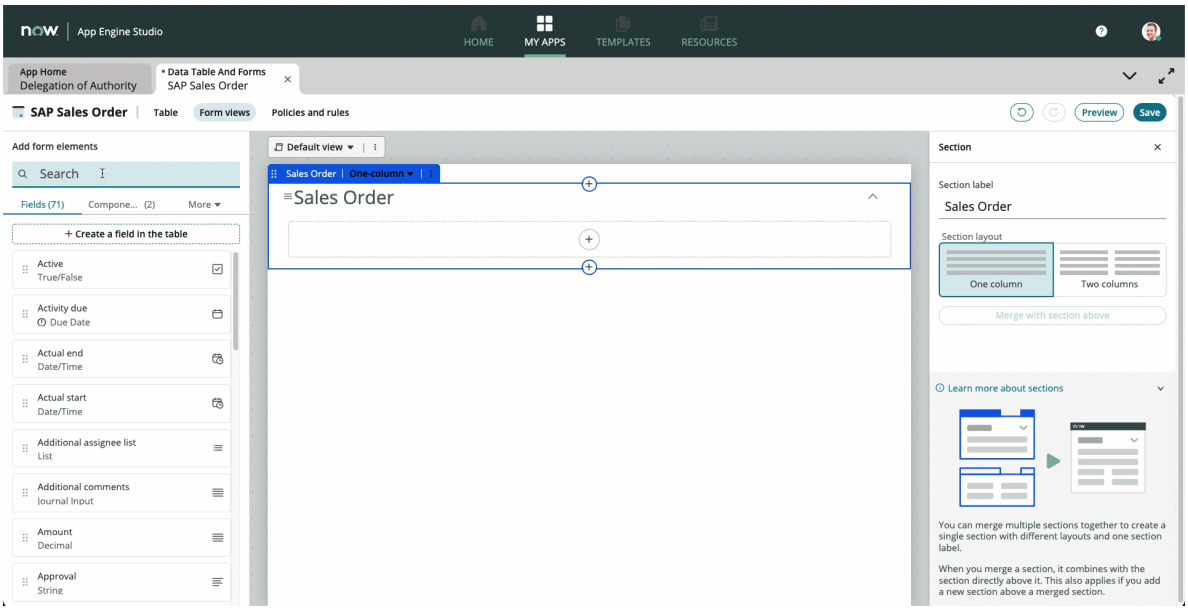
21. Click **Create**

22. Rename the section to **Sales Order** by clicking on the section and editing the **Section label** on the right panel

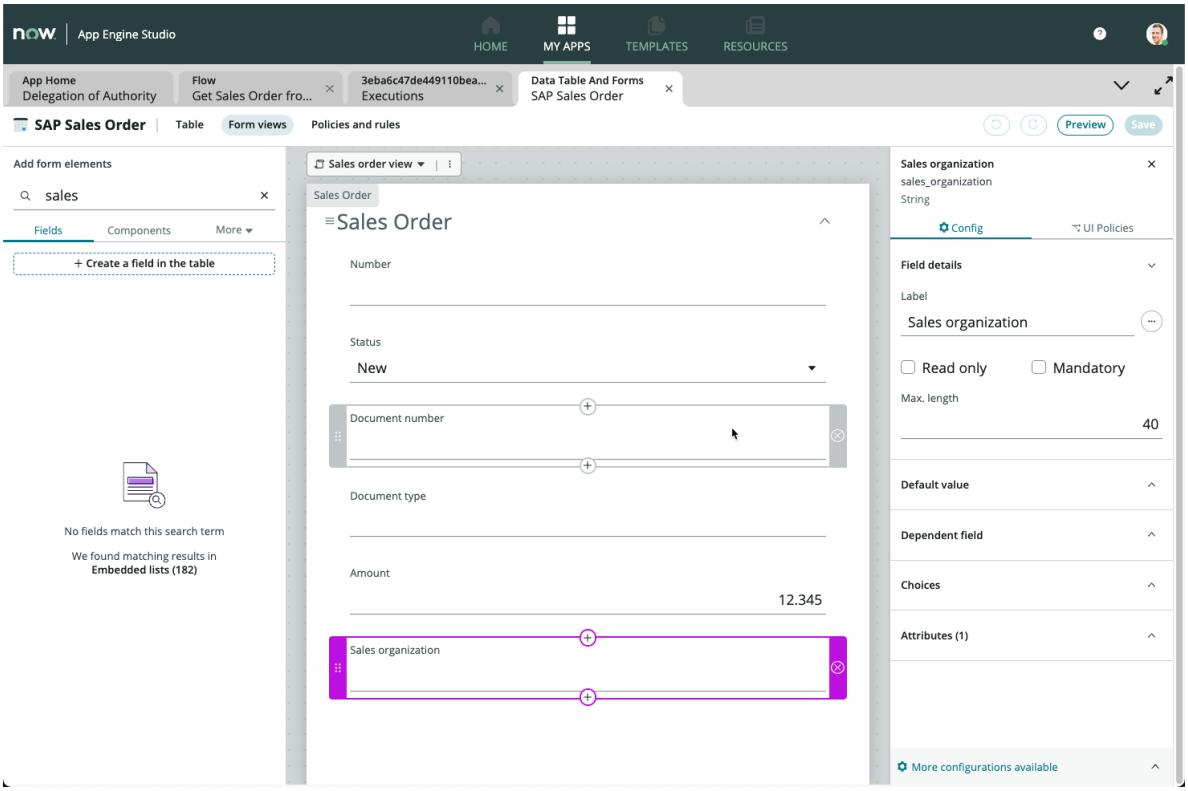


23. Drag the following fields onto the form:

    i. Number
    ii. Status
    iii. Document number
    iv. Document type
    v. Amount
    vi. Sales organization



24. Your completed view should look like this

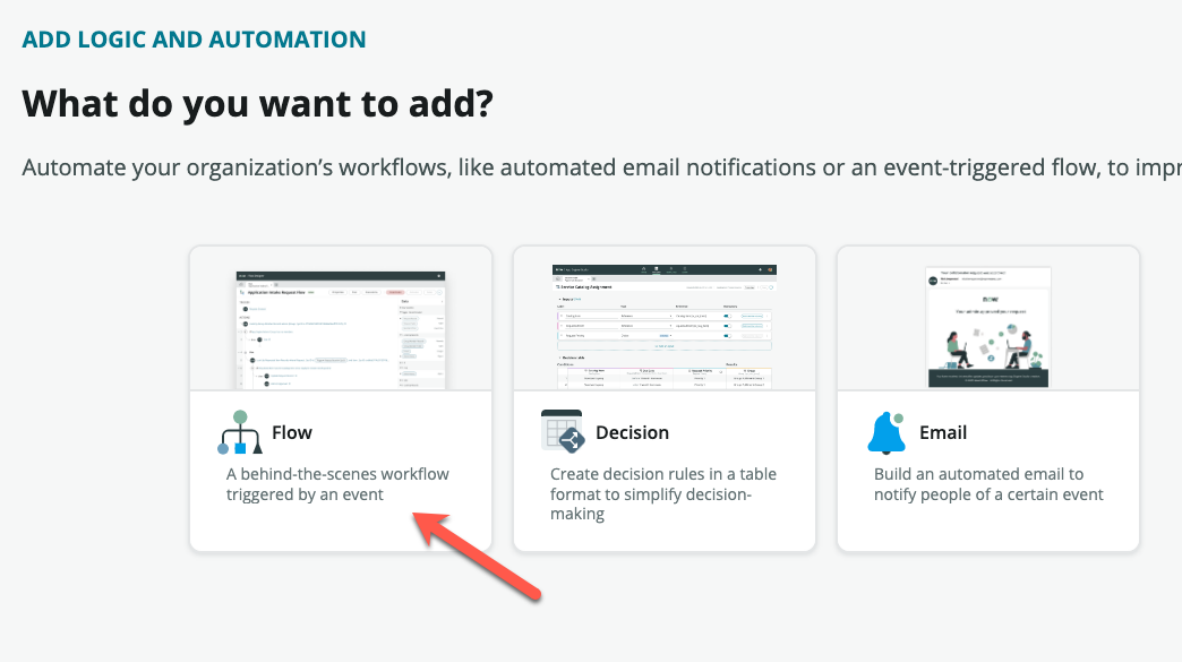


25. Click **Save**

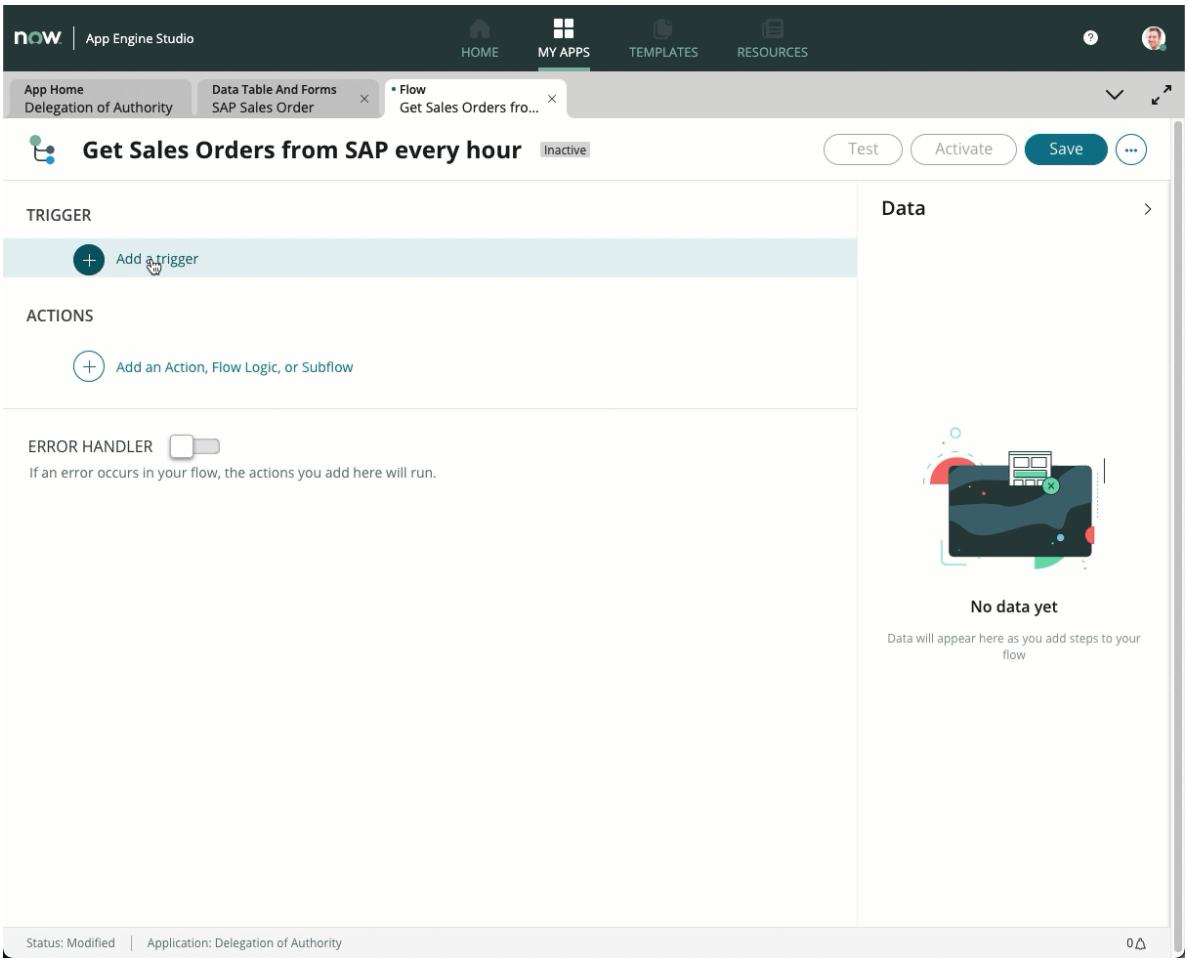
Great, you now have a simple table to store Sales Order data via the SAP integration that we will be configuring next.

Get Sales Order data via the SAP ERP Spoke

1. Go back to the app home page
2. Under *Logic and Automation*, click **Add**
3. In the *What do you want to add* screen, click **Flow**



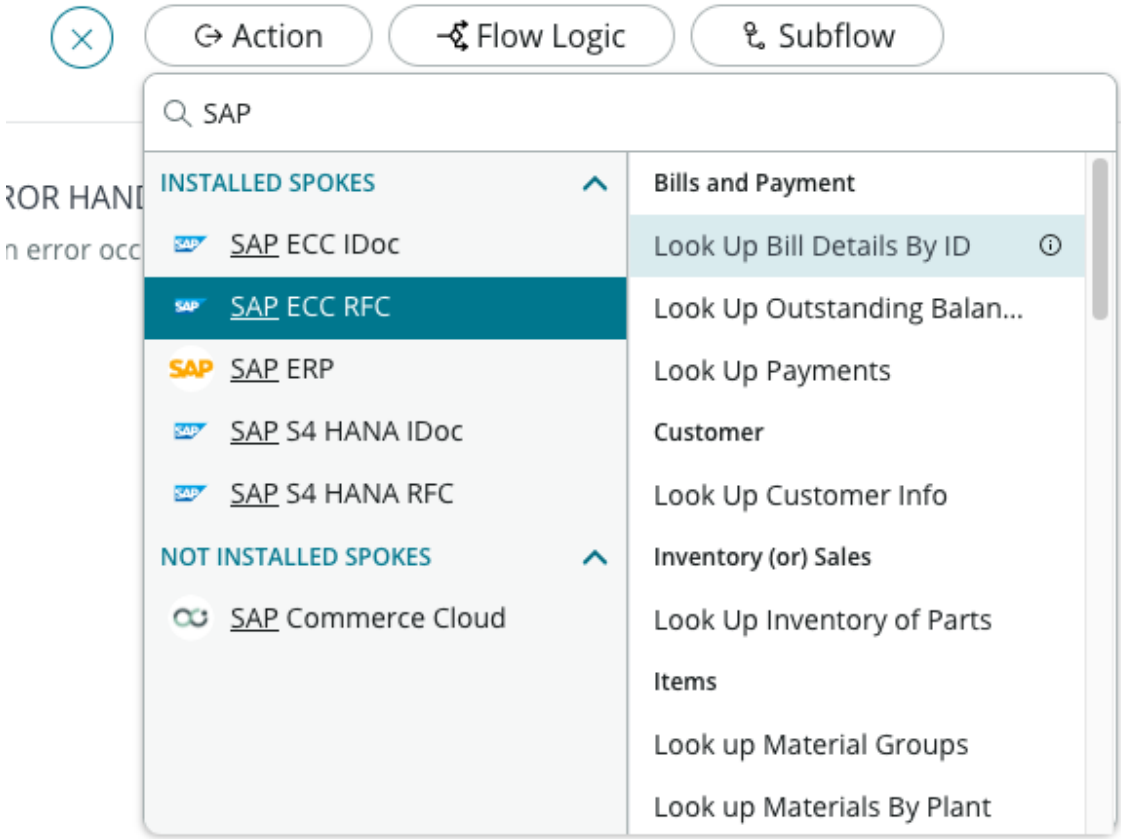
4. Click **Build from scratch**
5. Under **Name**, enter "Get Sales Orders from SAP every 2 hours"
6. Click **Continue**
7. Click **Edit this flow** once you see the *Success! Your flow is ready.* screen
8. Close the pop-up box
9. Click **Add trigger**
10. Under the *Date* section, click **Repeat**
11. Change the repeat duration to 2 hours



12. Click **Add an Action, Flow Logic, or Subflow**

13. Click **Action**, and you should see a dropdown appear

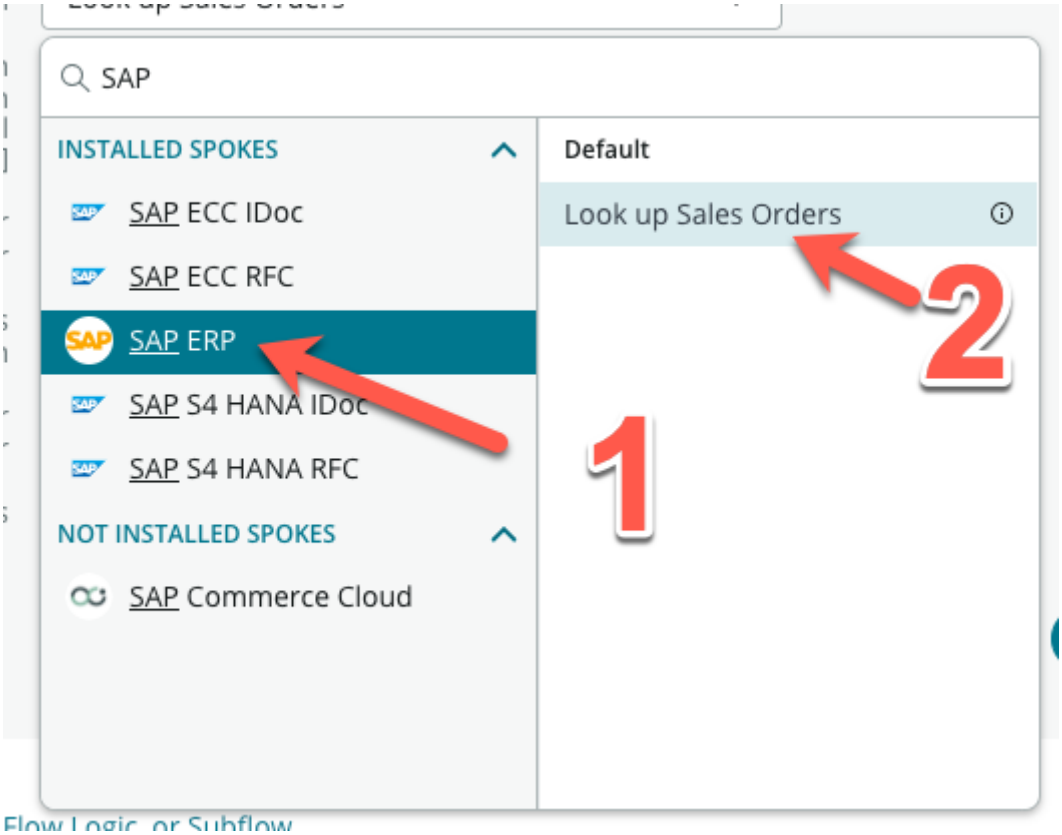
14. In the *Search* bar, enter **SAP**


Action
Flow Logic
Subflow
SAP
INSTALLED SPOKES
SAP ECC IDoc
SAP ECC RFC
SAP ERP
SAP S4 HANA IDoc
SAP S4 HANA RFC
NOT INSTALLED SPOKES
SAP Commerce Cloud
Bills and Payment
Look Up Bill Details By ID
Look Up Outstanding Balan...
Look Up Payments
Customer
Look Up Customer Info
Inventory (or) Sales
Look Up Inventory of Parts
Items
Look up Material Groups
Look up Materials By Plant

> You should be able to see a few different SAP spokes show up. There are 2 different integration methods to both SAP ECC (Older version) and SAP S/4 HANA available in ServiceNow's out of the box integrations available in the Enterprise package of Automation Engine.
>
> Let's break down what these mean:
>
> **SAP ECC**: Most of the companies running on SAP are running on SAP ECC. It is the offering for a large enterprise. Companies with large volume, complex business processes and operating in multiple geographies go for SAP ECC. It is built on ABAP stack.
>
> **SAP S/4 HANA**: This is the latest release of SAP ERP and it can run only on HANA database. With this release, SAP has simplified their core database architecture. This together with in-memory processing enables business to do complex business computation within minutes.
>
> **Integration via IDoc**: IDOCs or intermediary documents are another way to exchange information to and from SAP. If you are more aware of web technologies, consider IDOCs as XML. It consists of neatly defined data segments with parent and child nodes. There are specific steps to configure inbound and outbound IDOCs.
>
> **Integration via RFC**: If you are looking at a real time SAP system integration scenario, RFC is probably the best way to go. In this case, certain functions are enabled for remote call. One such function could be for example sales order creation. Third party applications can integrate with SAP using these RFCs for a real time communication and business process validation (example price computation, minimum order check etc.).

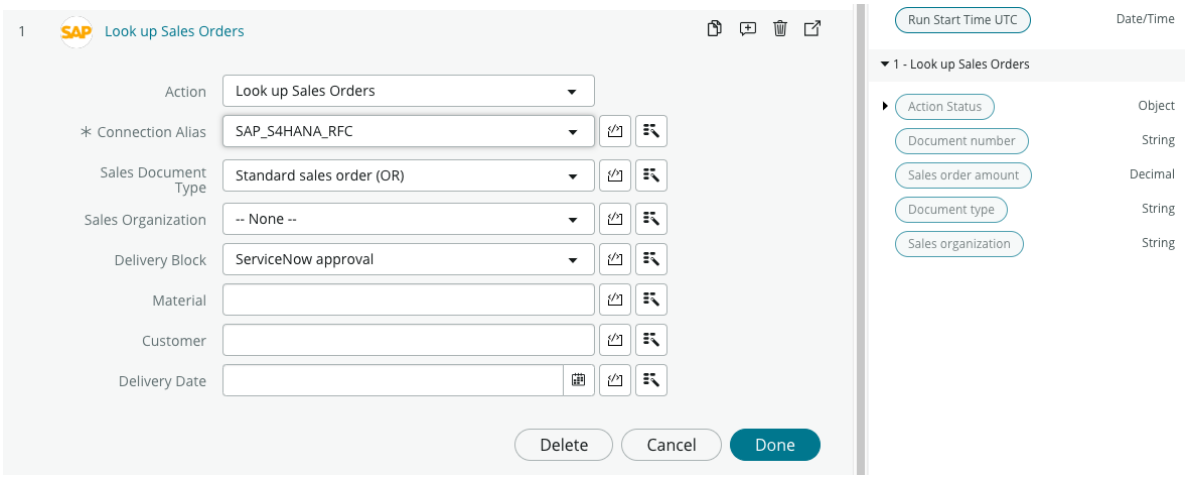
15. Click **SAP ERP**, this is a simulation of the actions available in the other prebuilt spokes.

16. Click **Look up Sales Orders**



Flow Logic, or Subflow

17. In the action that shows on the screen, have a quick look at what shows up. These are a subset of the filter options available in the OOTB spokes to retrieve a list of Sales Orders. In this exercise, we will only return one Sales Order record for easier accesibility.

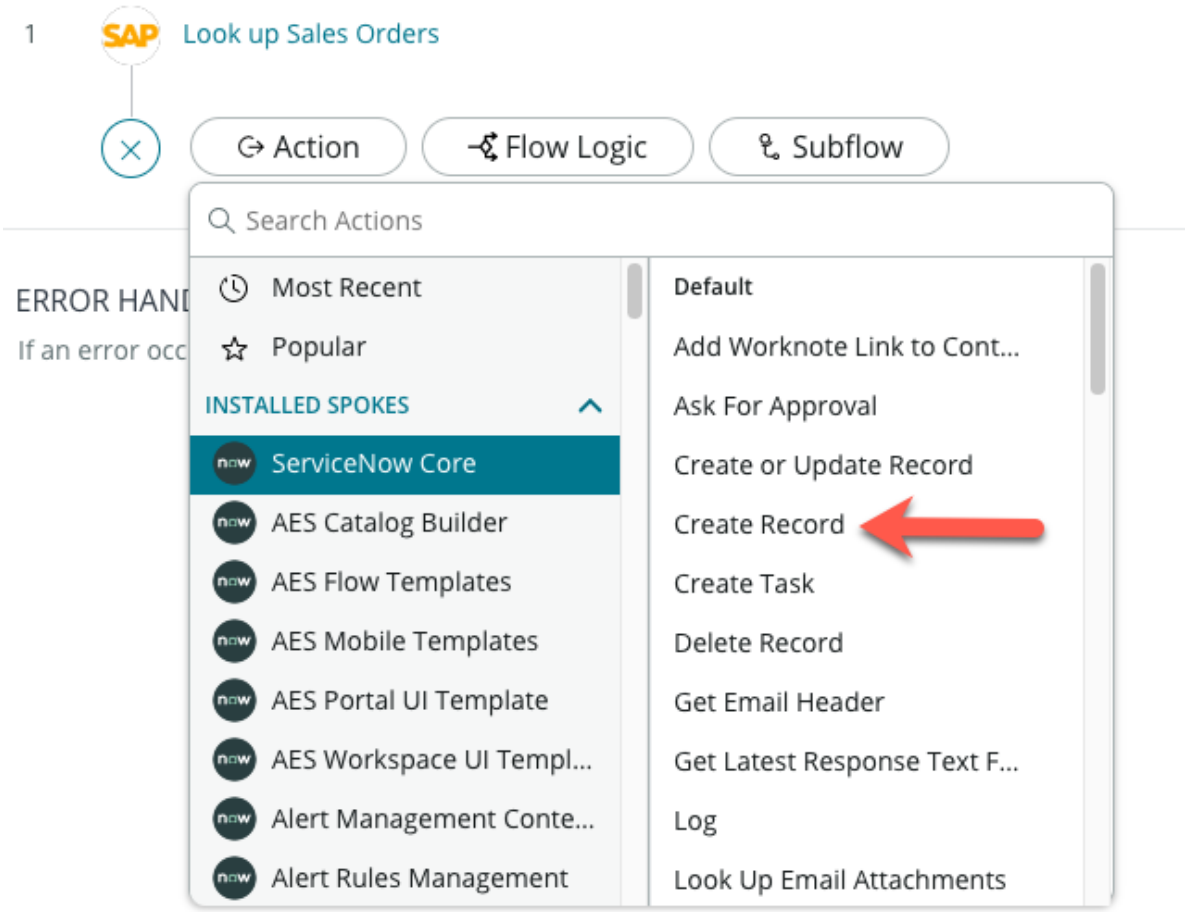


18. Notice how the **Delivery Block** field is populated with **ServiceNow approval**. Delivery block when applied, blocks the sales order for delivery i.e., delivery cannot be created until the delivery block is removed by the authorized person. This list can be customised in SAP, and the logic would be that as Sales Orders are created in SAP, this Delivery Block status is automatically applied until the approval is completed in ServiceNow.

19. Leave everything as it is, and click **Done**

20. Click **Add an Action, Flow Logic, or Subflow**

21. Under **ServiceNow Core**, click **Create Record**

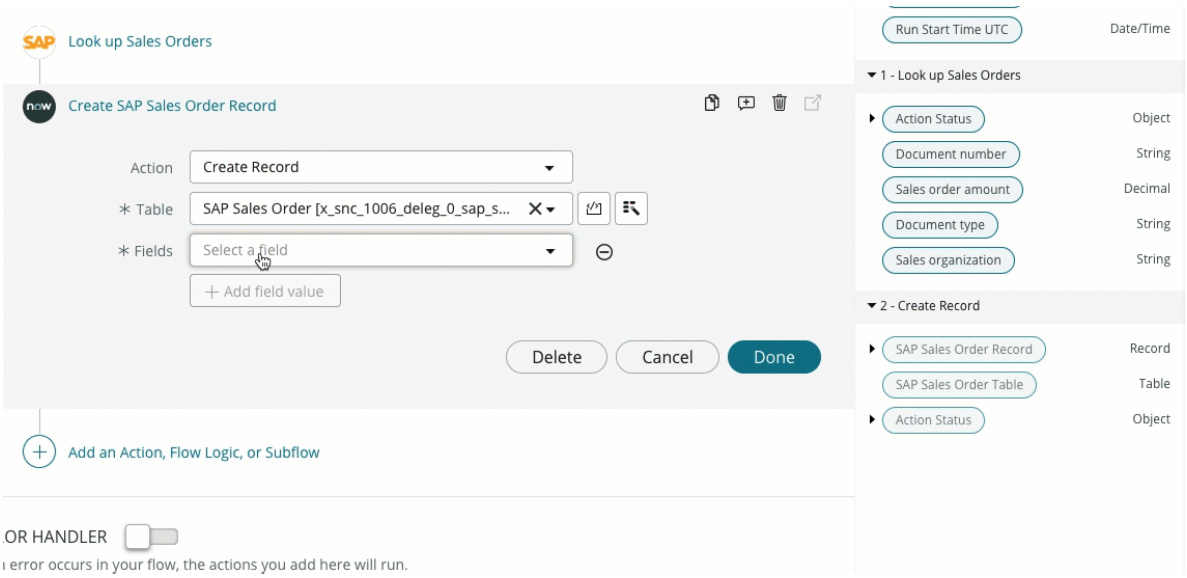


22. On the table field, search and select **SAP Sales Order** (the custom table you created)

23. Click **Add field value**, select **Document number**

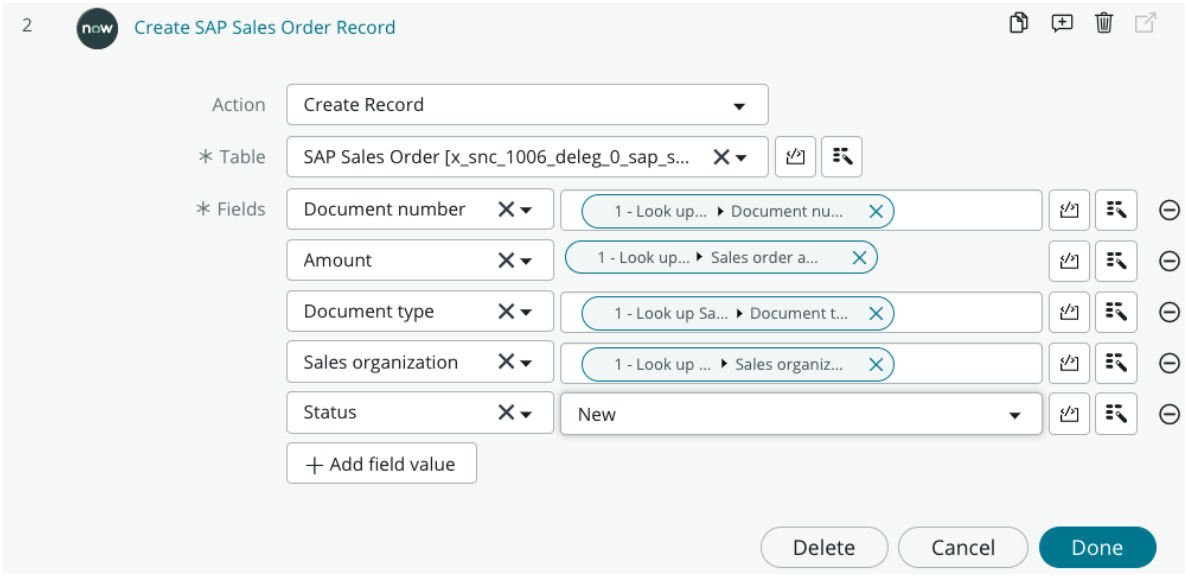
24. Drag the **Document number** data pill into the input field

25. Repeat the above 2 steps for **Amount, Document type** and **Sales organization**



26. Add one final field **Status** and set the value to **New**

27. Your entire action should look like this

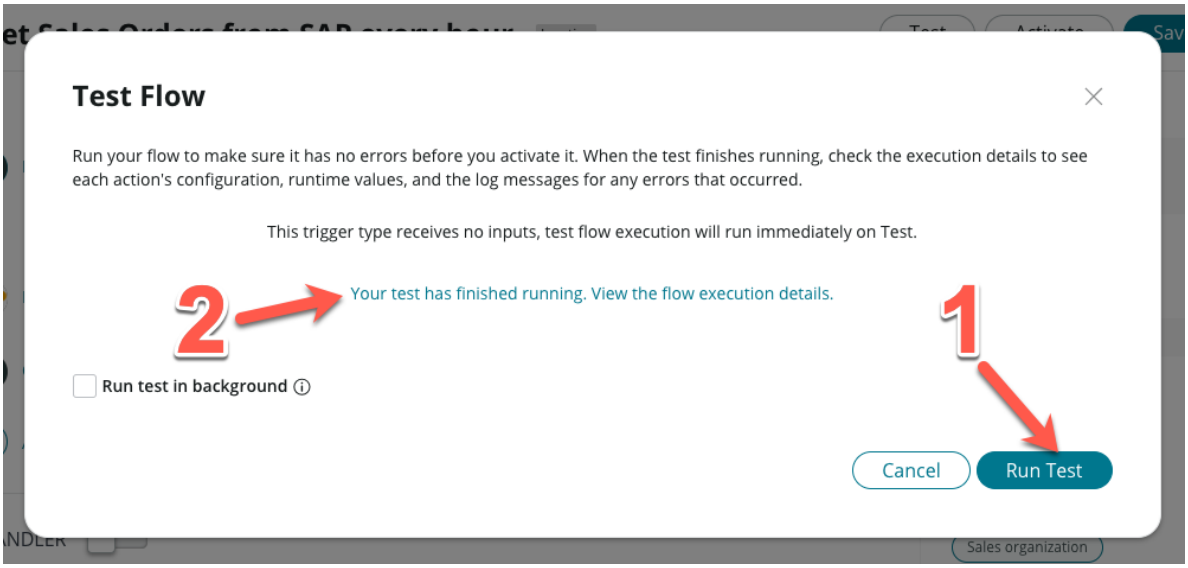


28. Click **Save** on the top right of the screen.

29. Click **Test**

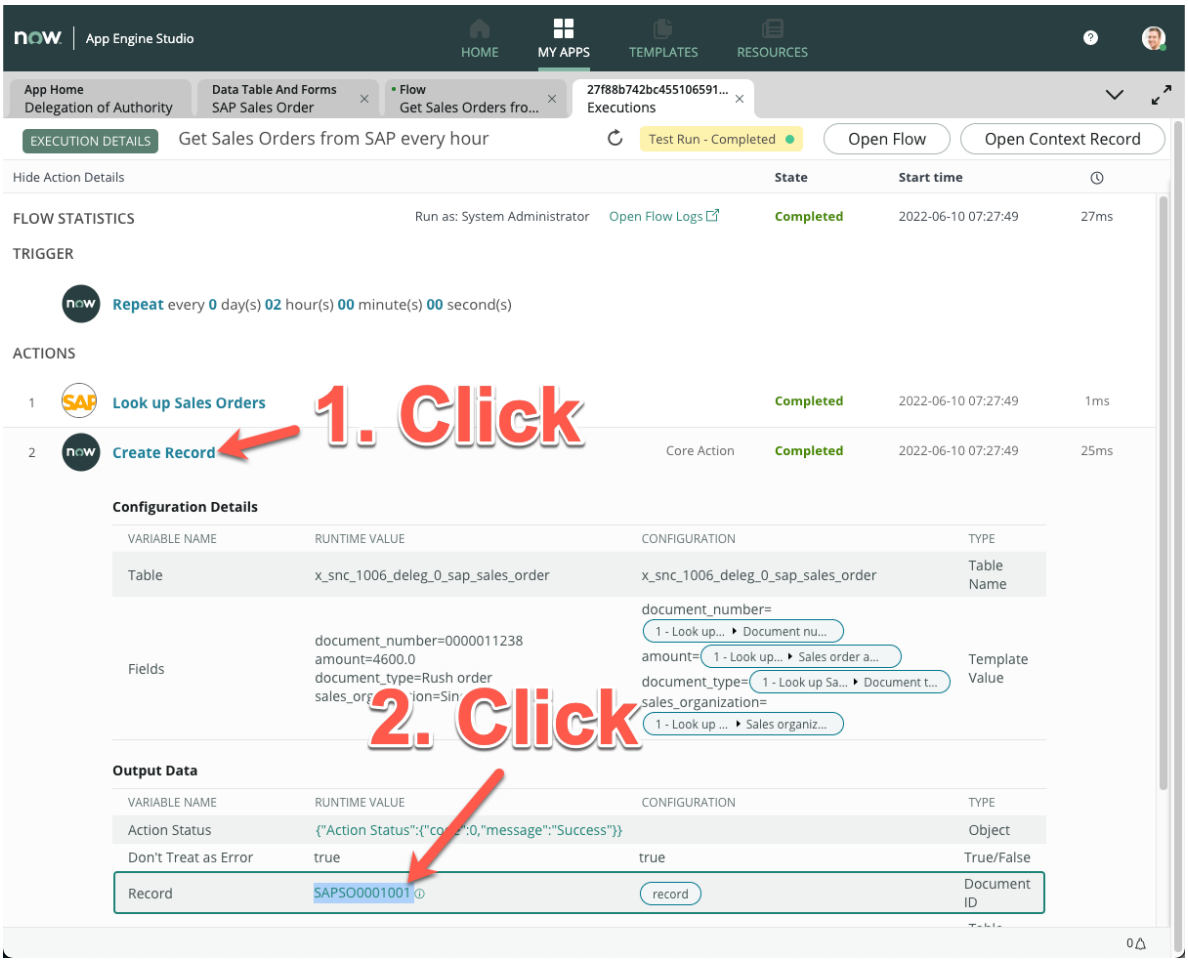
30. Click **Run Test**

31. Click **Your test has finished running. View the flow execution details.**



32. In the new tab that opens, click the **Create Record** action

33. Click the Record, if you followed the guide so far, the record should be **SAPSO0001001**



34. In the pop-up box, click **Open Record**, it should open in a new browser tab.

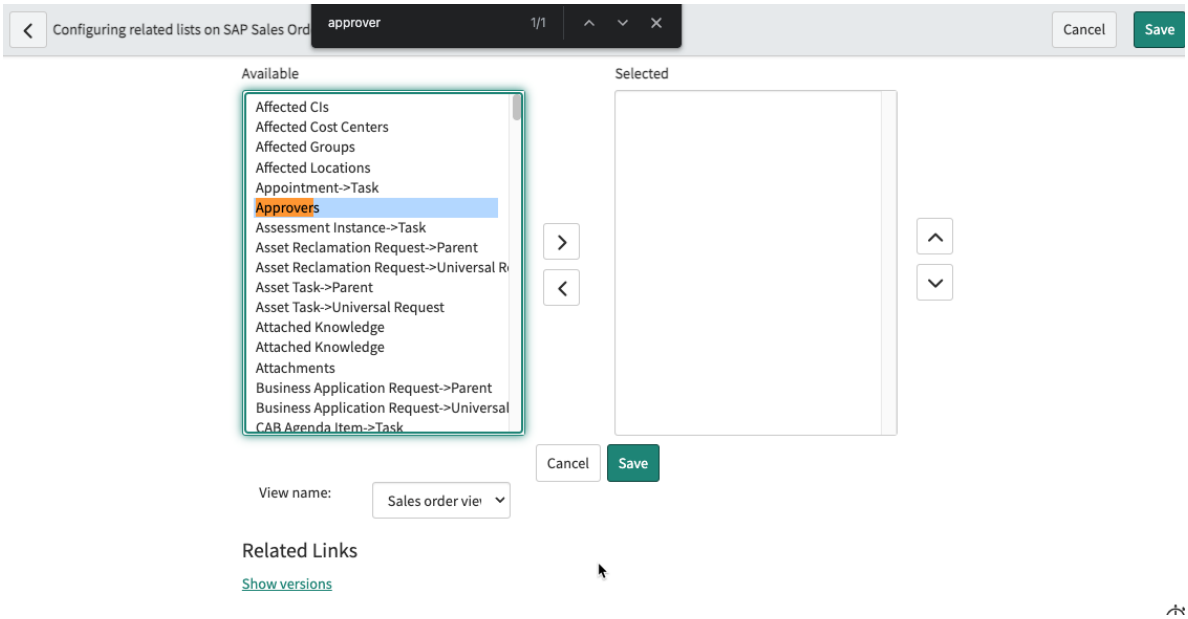
35. Click on the Hamburger icon on the top left, and under **View**, select **Sales order view**

36. Ensure that all the fields are populated. (Your output will be different)

37. Click on the Hamburger icon again

38. Under **Configure**, click **Related Lists**

39. Search for **Approvers**, and shift that into the **Selected** column



40. Click **Save**

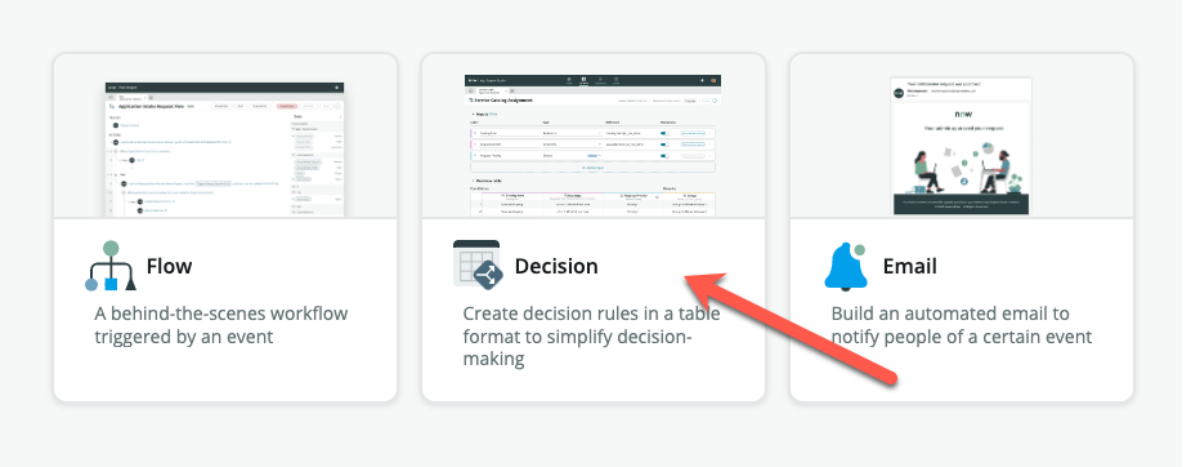
Congrats! You have built the table to store Sales Order data from SAP. Any approvals that you trigger later can also be seen on this view. Go back to the previous browser tab, but don't close anything yet.

Building the approval matrix using decision builder

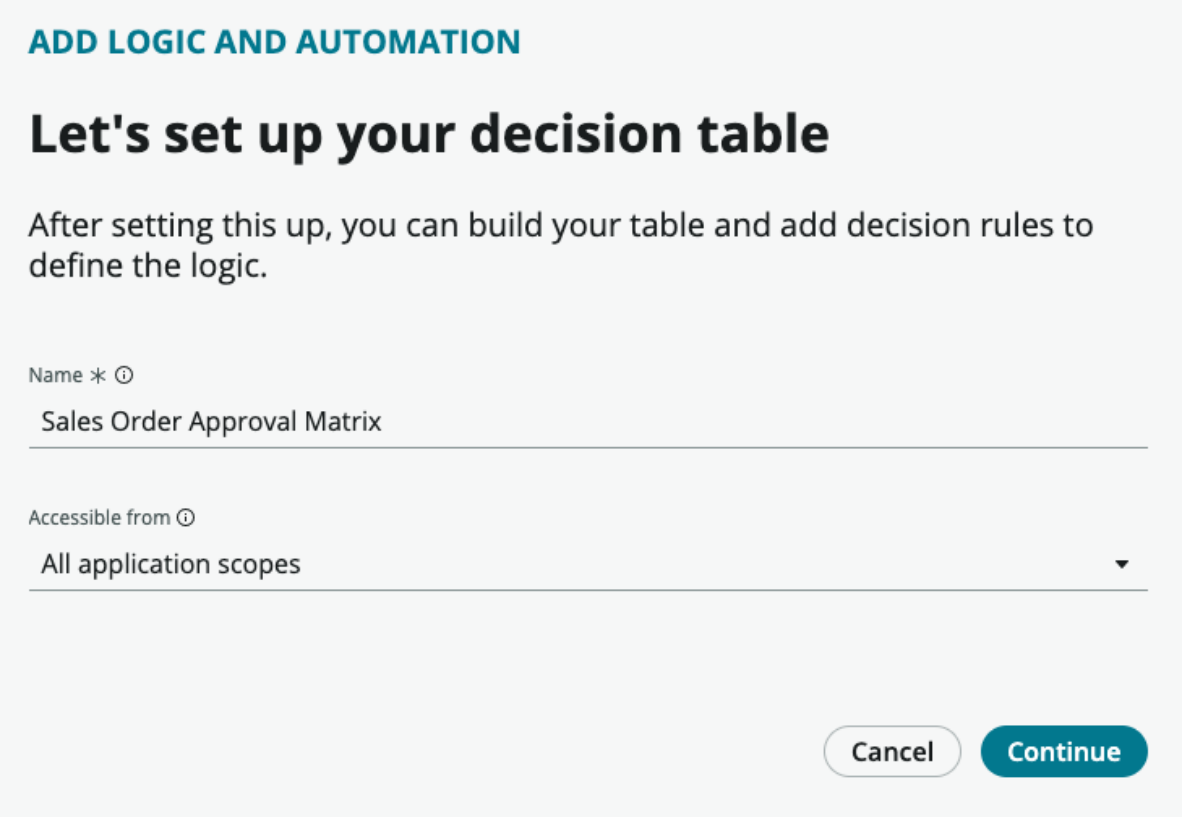
For this part of the exercise, we will rebuild this following approval matrix

Sales Order Amount	Sales Order Type	Approval Group
< $5,000	Standard sales order	Finance manager
$5,000 - $50,000	Standard sales order	Finance controller
> $50,000	Standard sales order	CFO
Any amount	Rush order	CFO

1. Go back to the **App Home** tab
2. Under *Logic and automation*, click **Add**
3. Click **Decision**

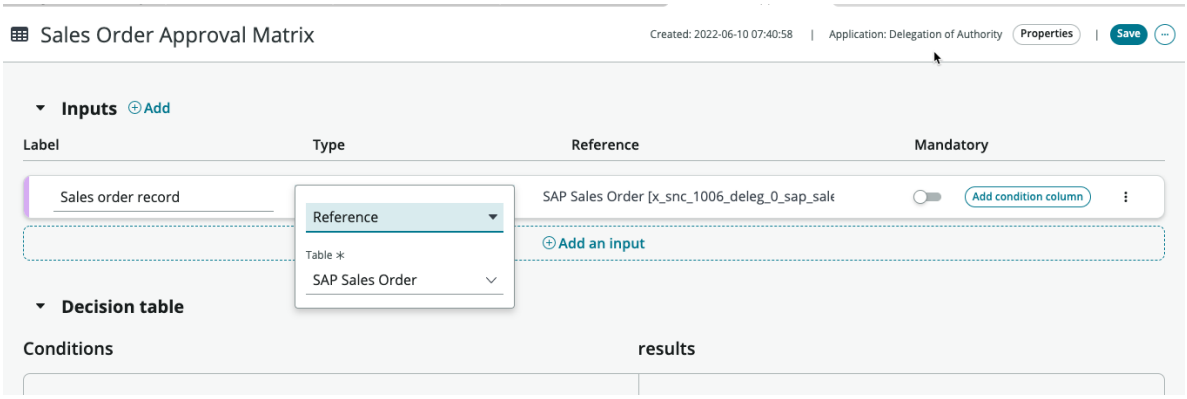


4. Click **Get started**
5. Under Name, enter **Sales Order Approval Matrix**



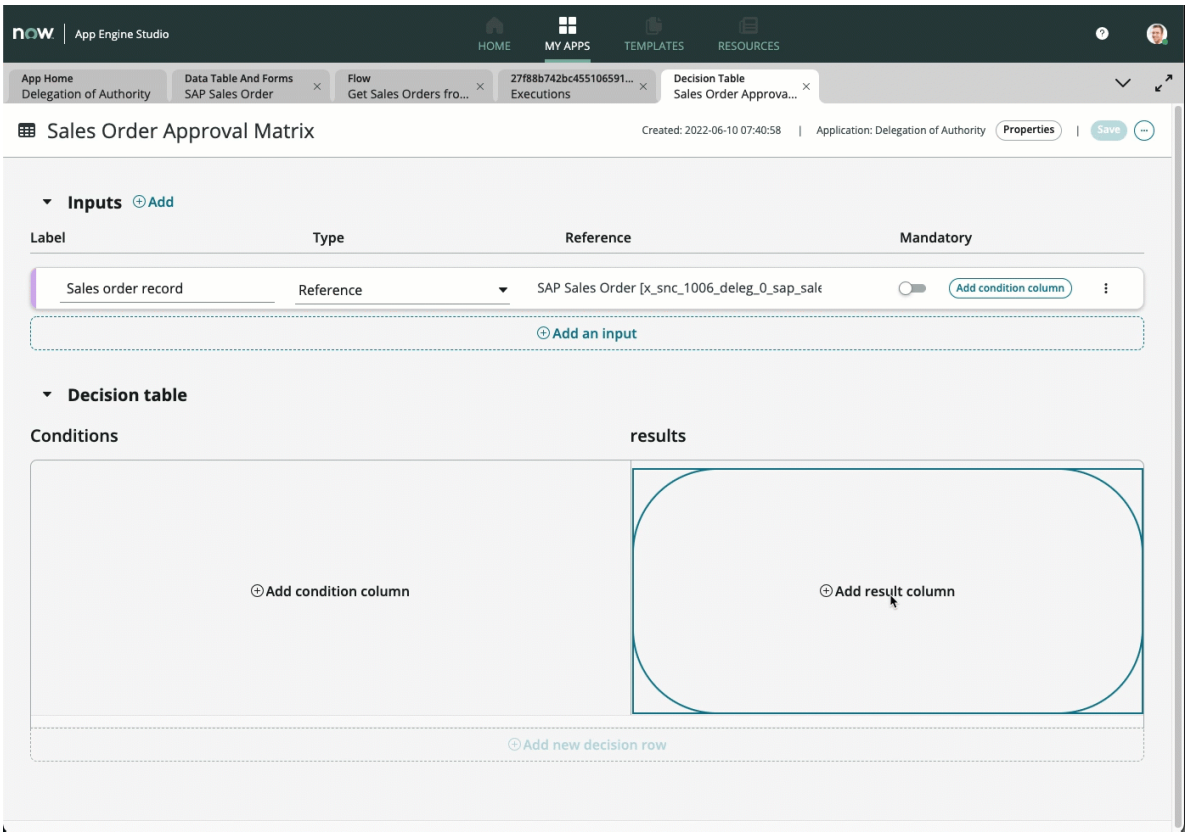
> If you face an issue with the Continue button not turning blue, you might have to close this screen and click Decision again

6. Click **Continue**
7. Click **Edit decision table**
8. Click **Add an input**
9. Under label, enter **Sales order record**
10. Under type, select **Reference**, and under Table, search and select **SAP Sales Order**



11. Click **Save** on the top right

12. Click **Add result column**



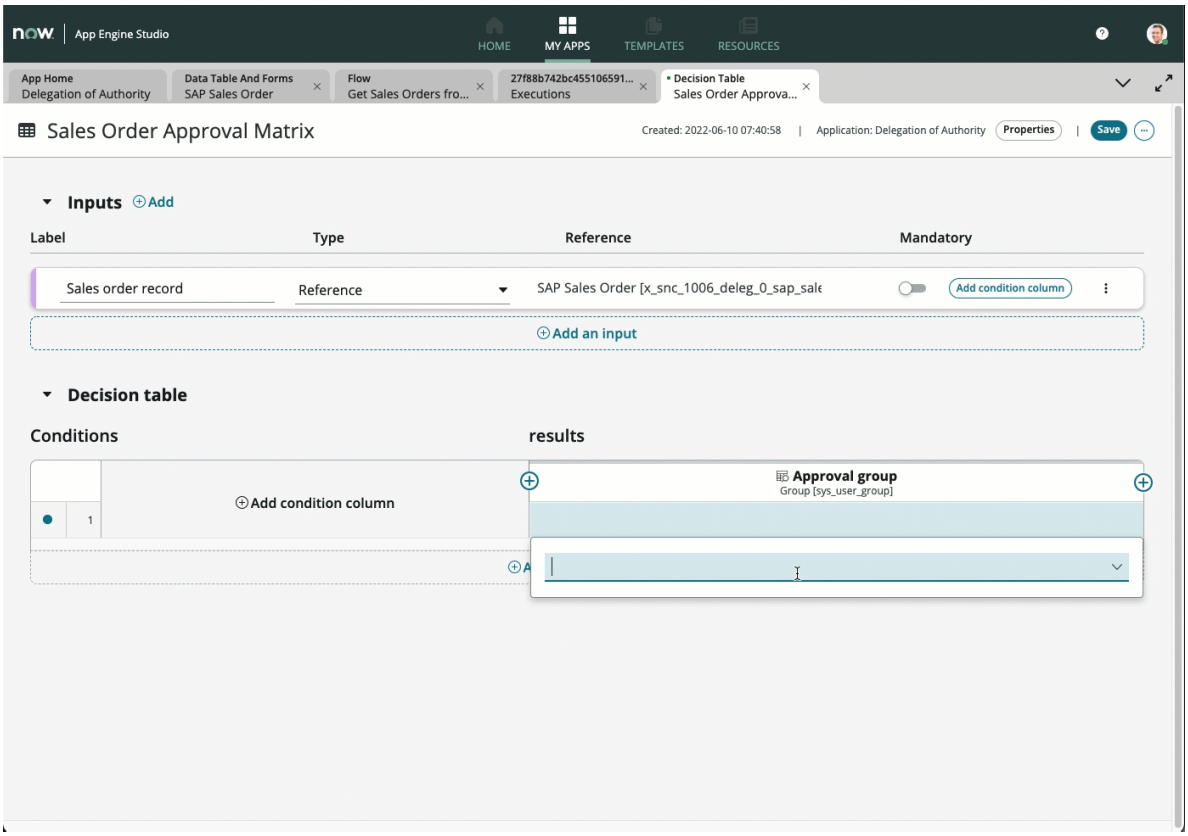
13. Under Result column label, enter **Approval group**

14. Click **Result type**, then select **Reference**

15. Under **Result table**, search **Group** and select **sys_user_group**

16. Click **Done**

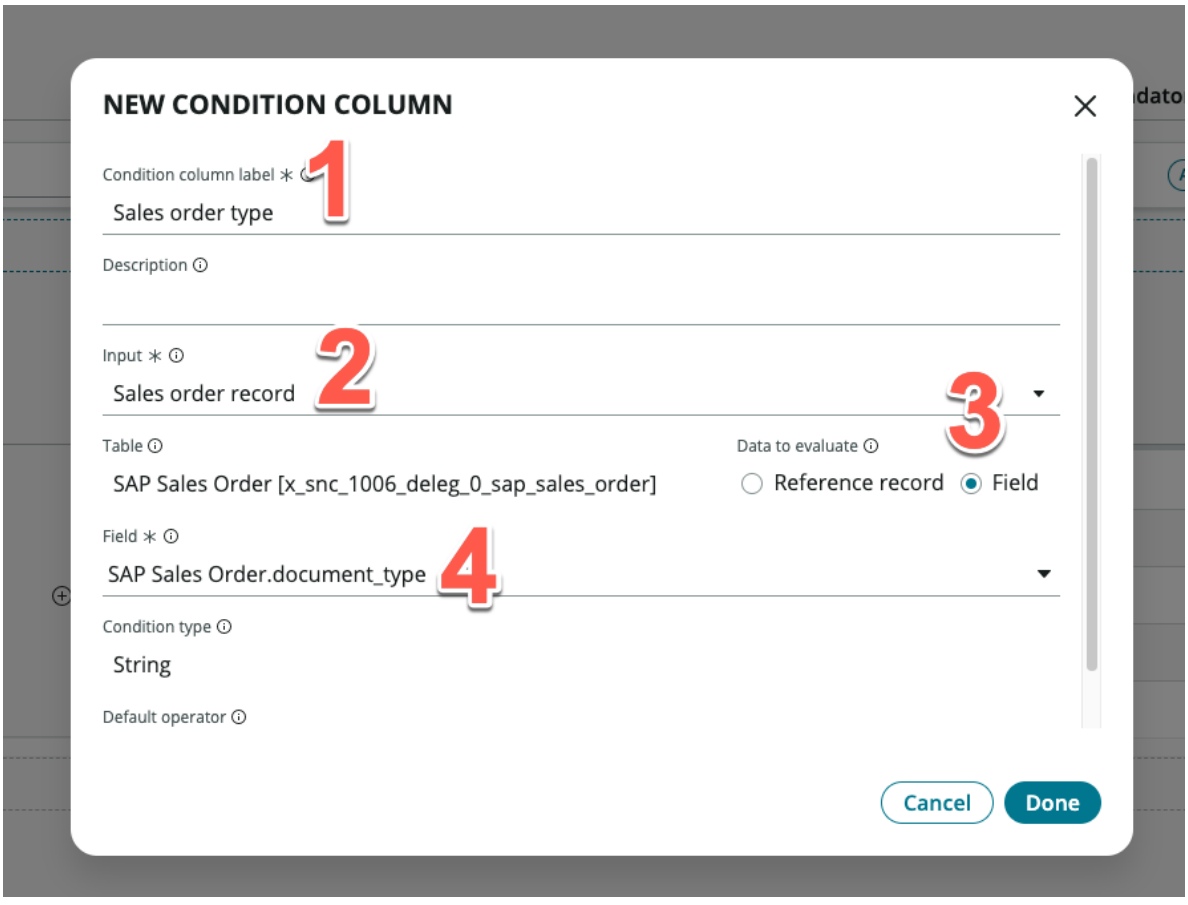
17. Under the **Approval group** column, add these selections as per our matrix at the start of this section: Finance managers, Finance controllers, CFO, CFO (Ensure that CFO is created twice as we have two different scenarios that require CFO approval)



18. Now we have the results, we need to add the conditions.

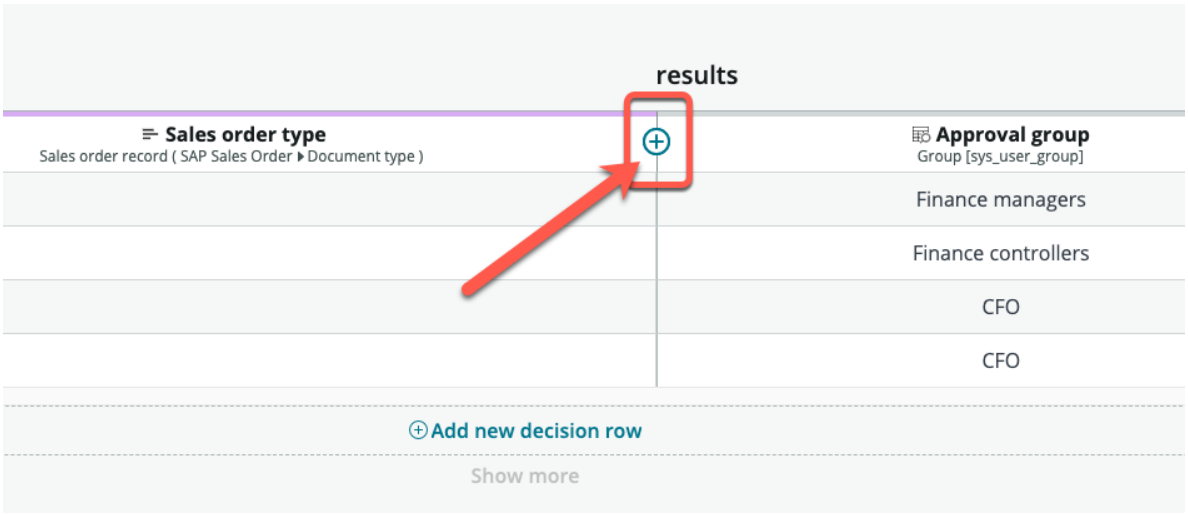
19. Click **Add condition column**

20. In the pop-up, select the fields as per the image below



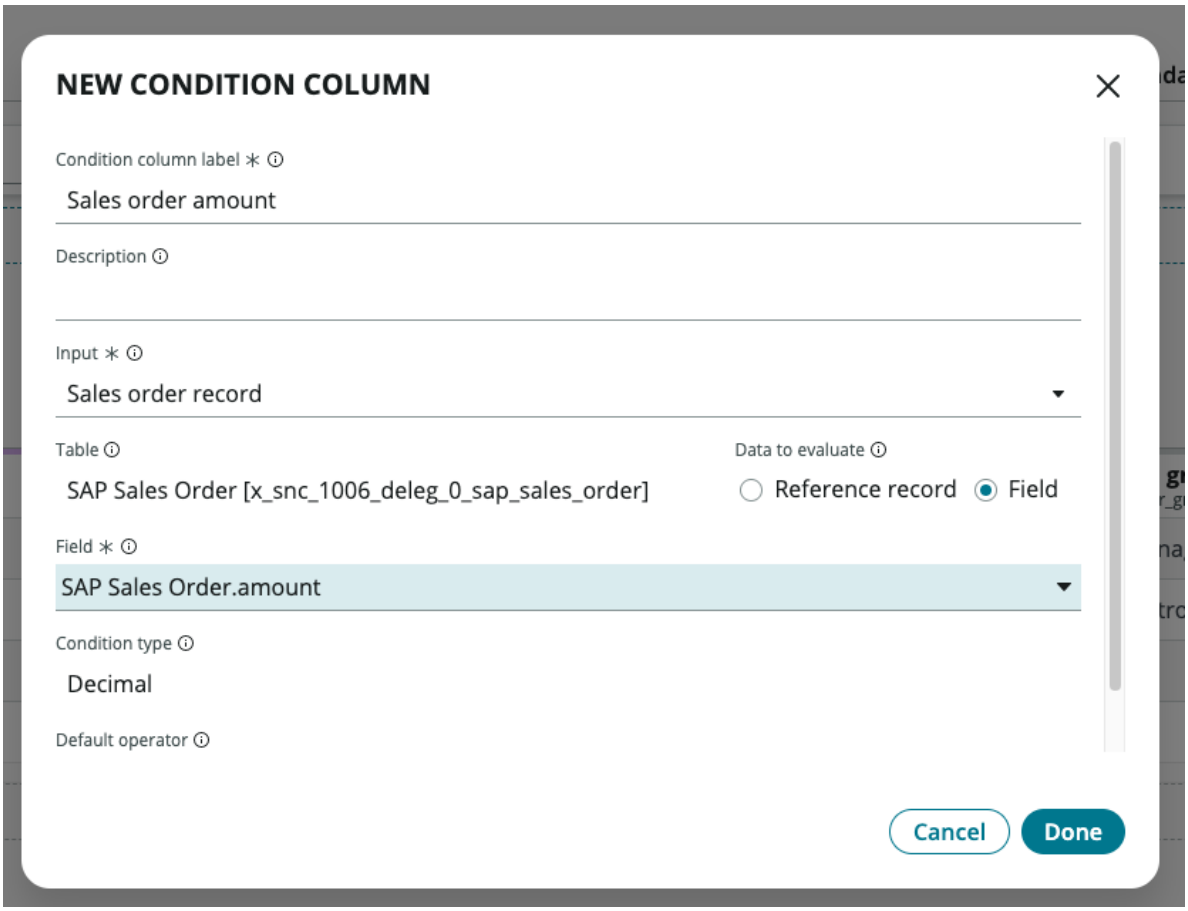
21. Click **Done**

22. Click on the **Plus sign** in between the 2 columns



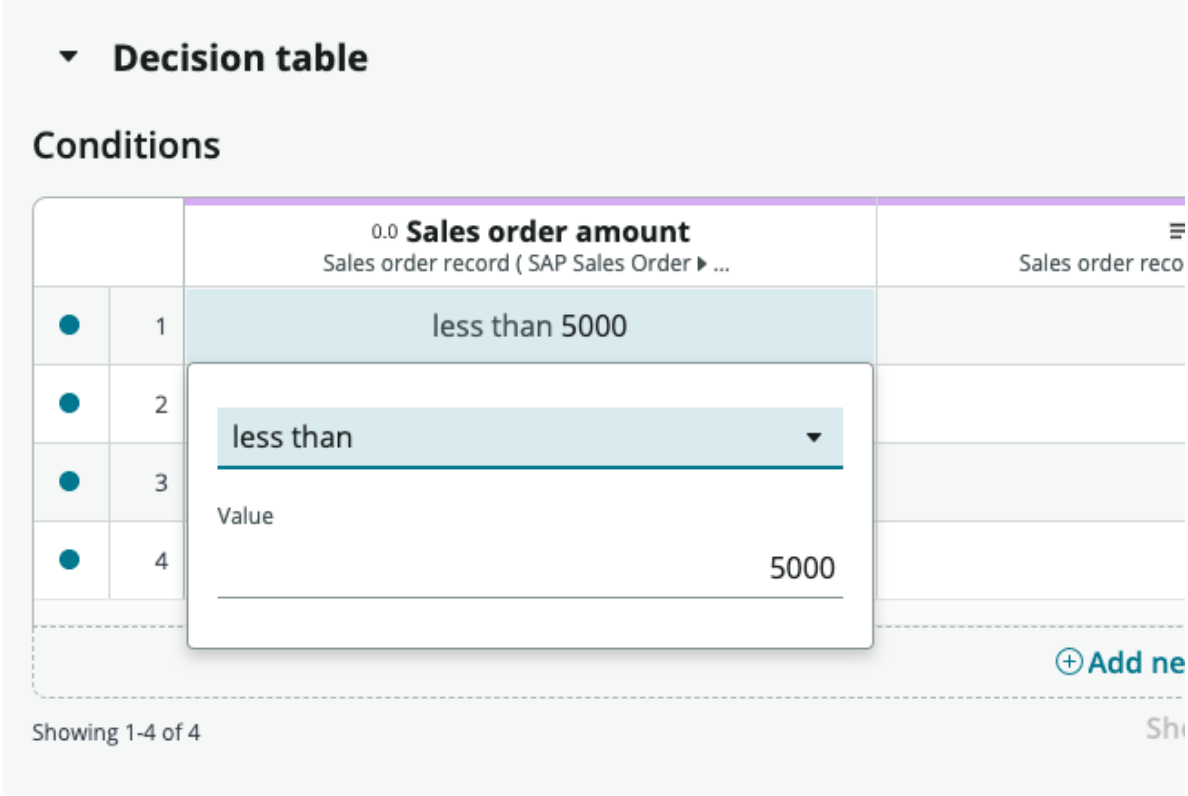
23. Click **Add condition column**

24. Fill in the pop-up modal as per the image below

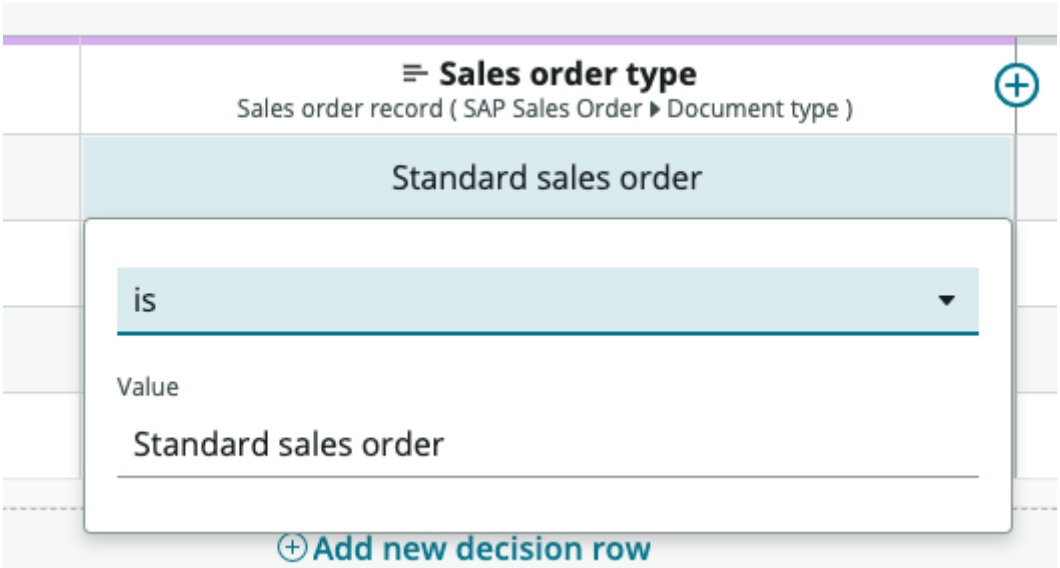


25. Click **Done**

26. Click on the top left selection box, select **less than**, and Value **5000**



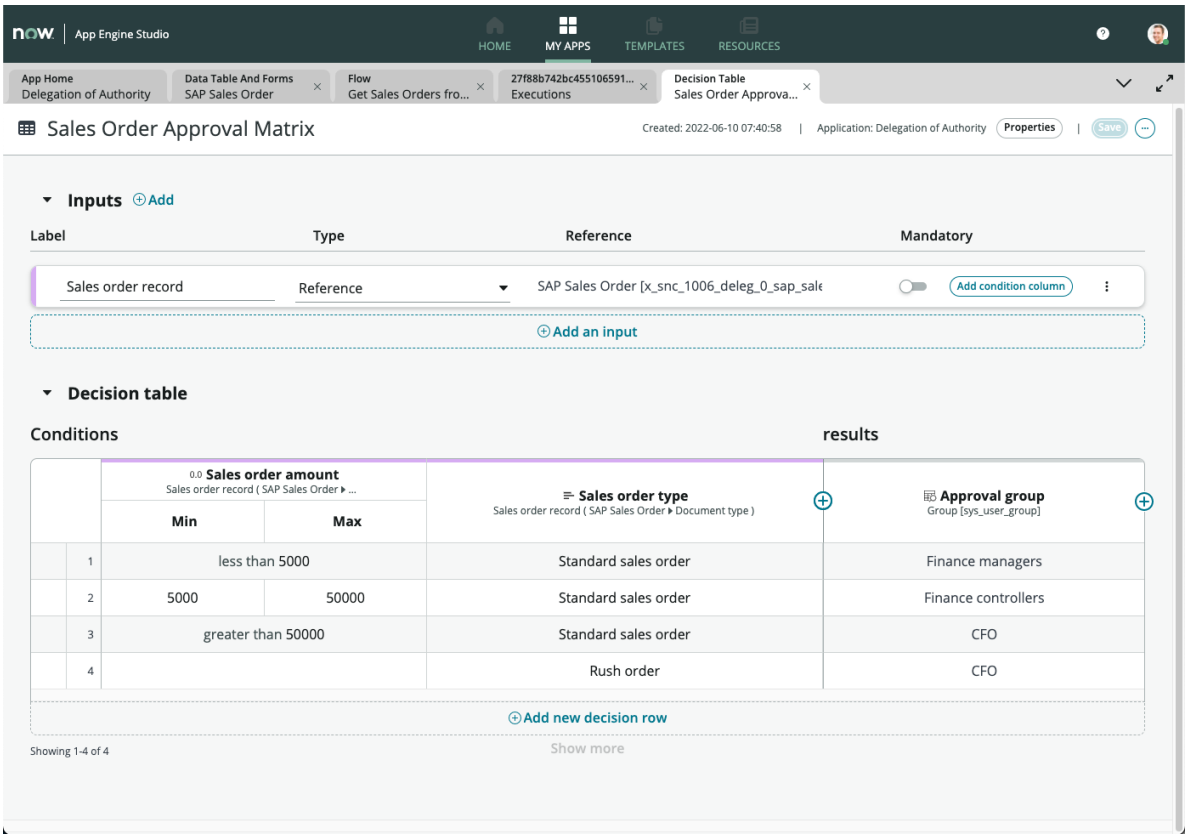
27. On the box to the right, select **is**, and Value **Standard sales order**



28. See if you can fill in the rest according to the matrix below (You have just completed the first row)

Sales Order Amount	Sales Order Type	Approval Group
< $5,000	Standard sales order	Finance manager
$5,000 - $50,000	Standard sales order	Finance controller
> $50,000	Standard sales order	CFO
Any amount	Rush order	CFO

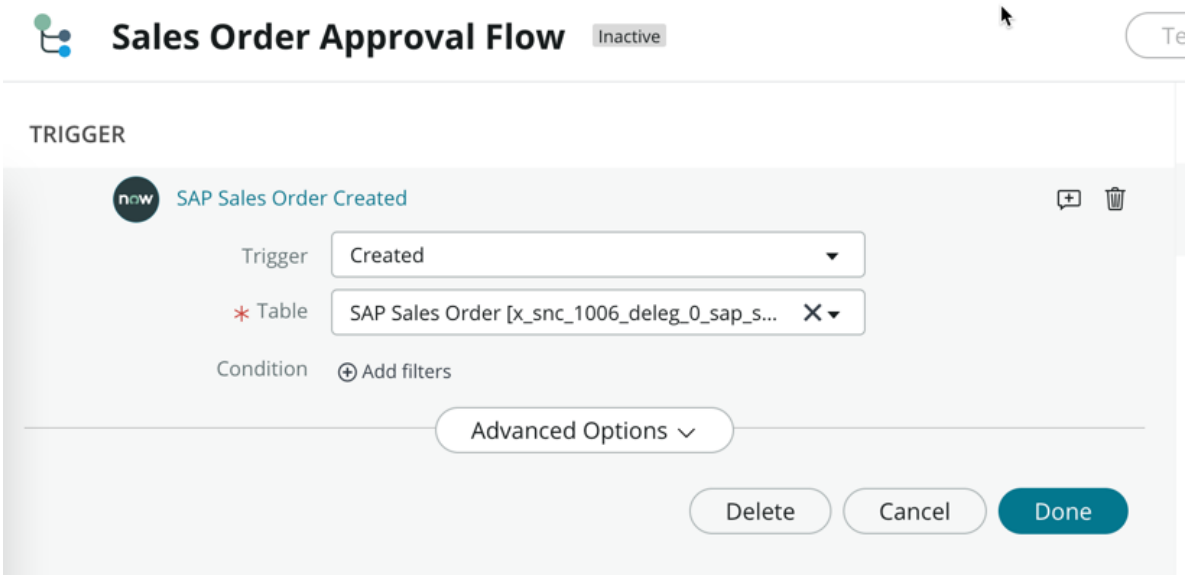
29. Your screen should look like the following once complete



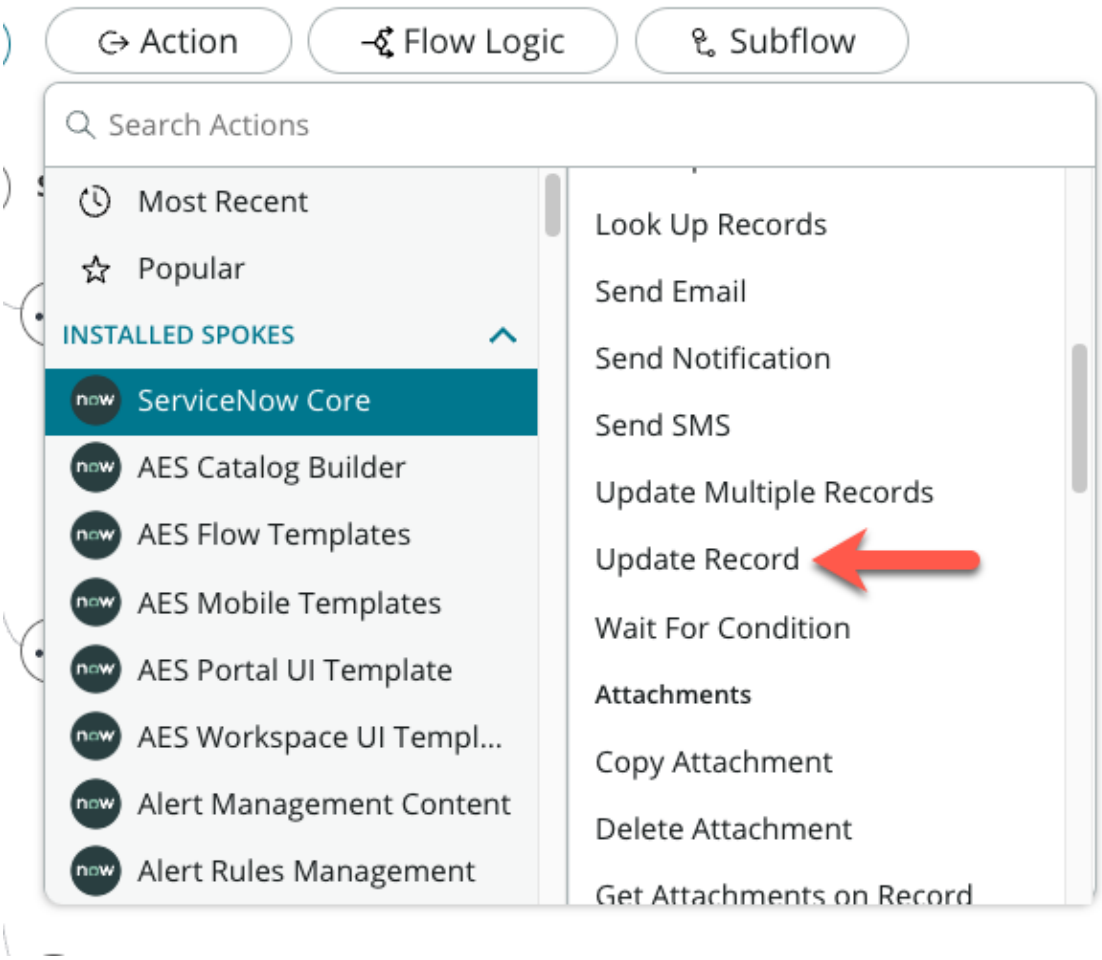
30. Click **Save** on the top right

Using approval decision matrix in a flow

1. Navigate back to **App Home**
2. Under *Logic and automation*, click **Add**
3. Click **Flow**
4. Click **Build from scratch**
5. Under **Name**, type **Sales Order Approval Flow**, then click **Continue**
6. Click **Edit this flow**
7. Click on **Add a trigger**, and under *Record*, select **Created**
8. Under *Table*, search and select **SAP Sales Order**

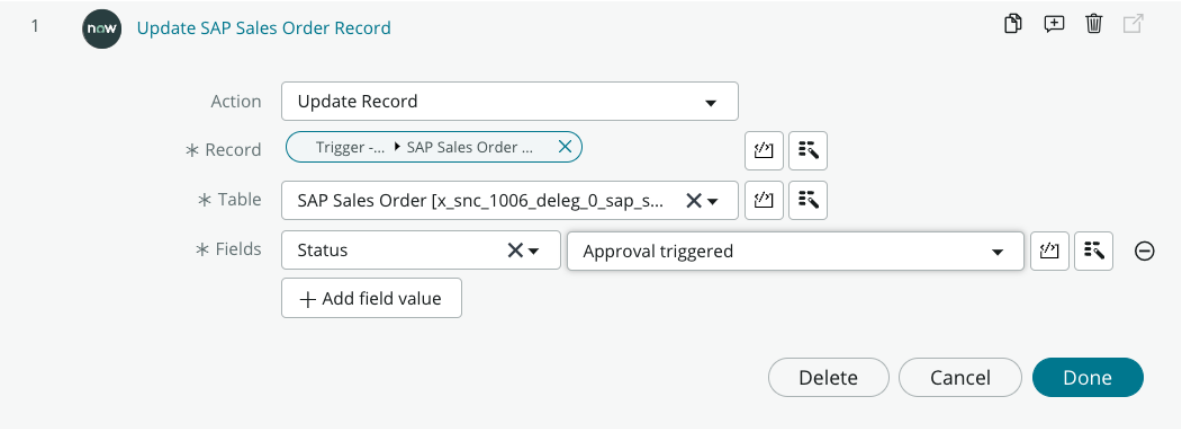


9. Click **Done**
10. Click **Add an Action, Flow Logic, or Subflow**
11. Click **Action**, then select **Update Record** under **ServiceNow Core**



12. Drag the **SAP Sales Order Record** from **Trigger - Record Created** within the data column on the right, onto the **Record** field
13. Click **Add field value**

14. Set field to **Status** and select **Approval triggered**



15. Click **Done**

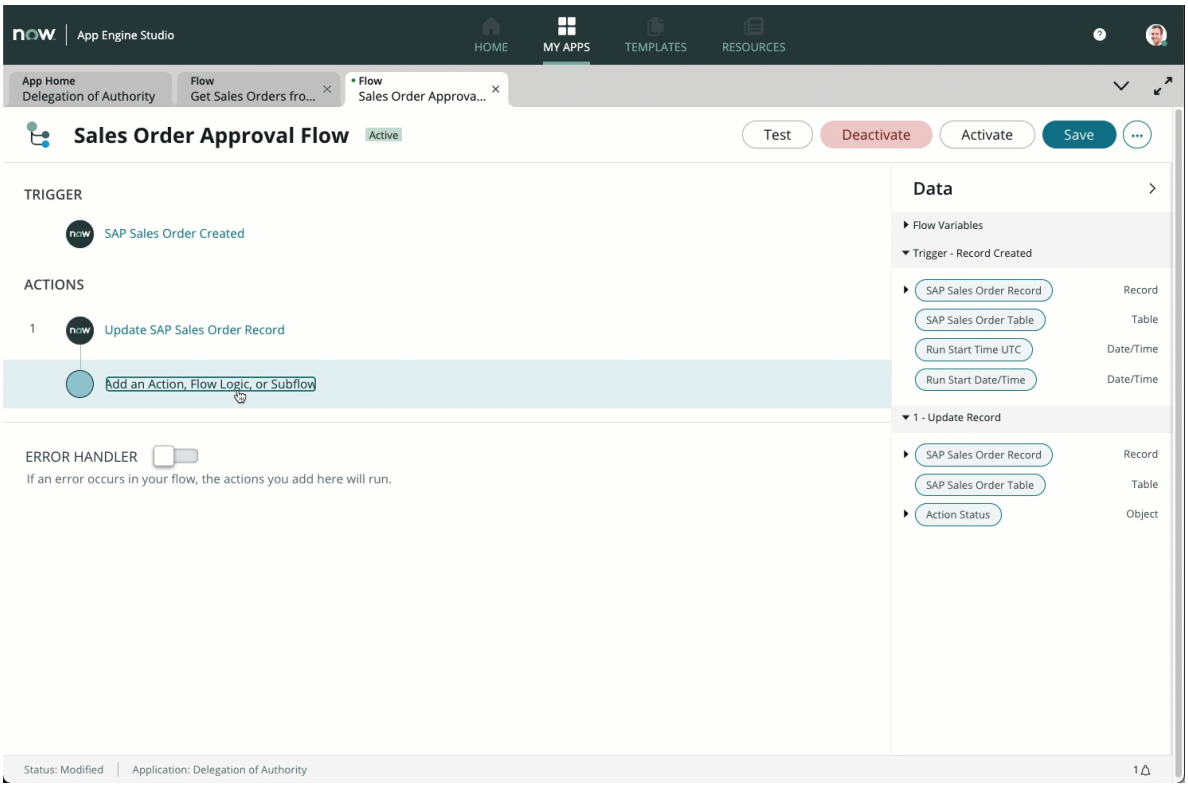
16. Click **Add an Action, Flow Logic, or Subflow**

17. Click **Flow Logic**, then **Make a decision**

18. Under *Decision Label*, type **SO Approval Matrix**

19. Under **Decision Table*, search for the decision table we just created, **Sales Order Approval Matrix**

20. Drag in the **SAP Sales Order Record** data pill from **Trigger - Record Created** within the data column on the right, onto **Sales order record**



21. Click **Done**

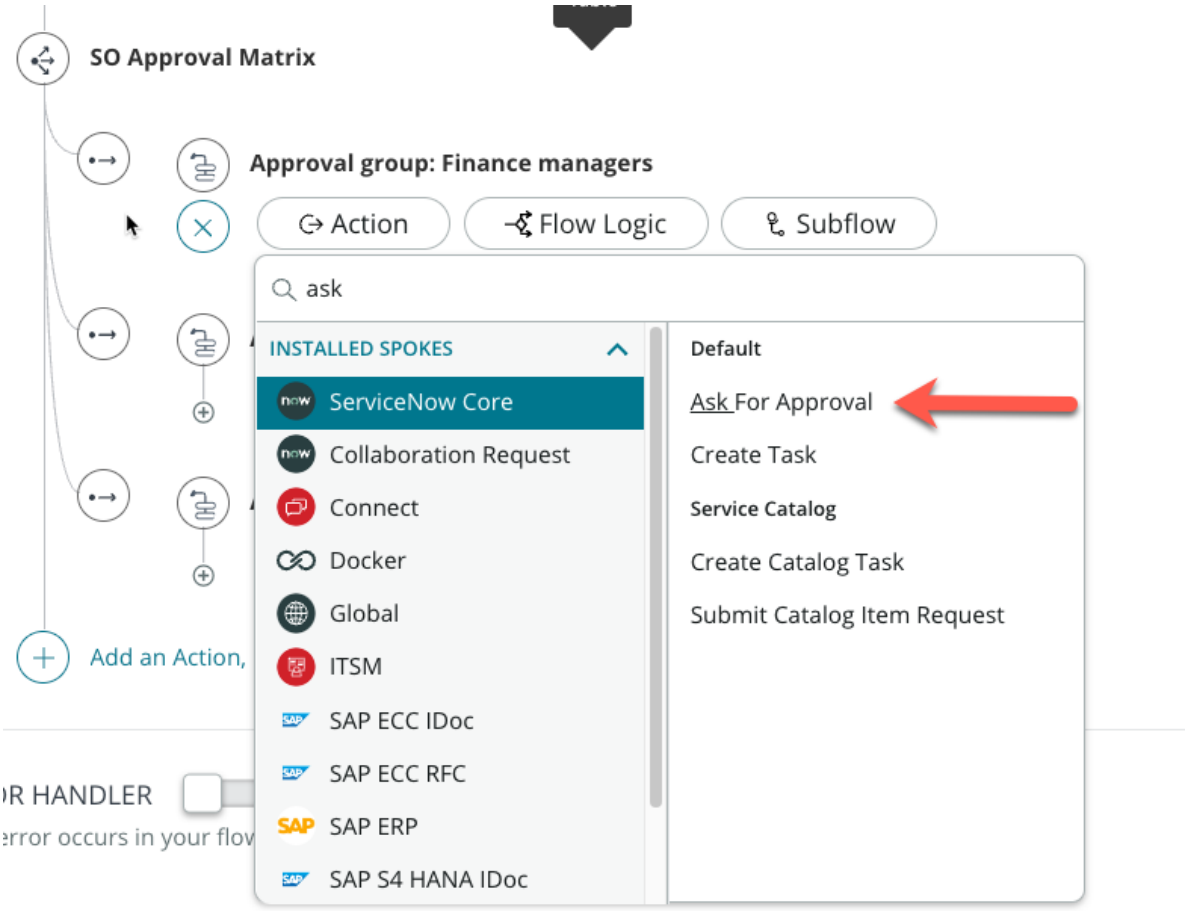
22. You should now see three branches appear for each *Group* we selected as approvers earlier

> **IMPORTANT** For this lab, we will not spend time building out the complete approval workflow for each branch. The idea here is that you now have the ability to cater to any specific workflow approval pattern using the ServiceNow Core *Ask for Approval* action.

23. Click on **+** under the first branch

24. Click **Action**

25. Search for **Ask For Approval**, then select the action

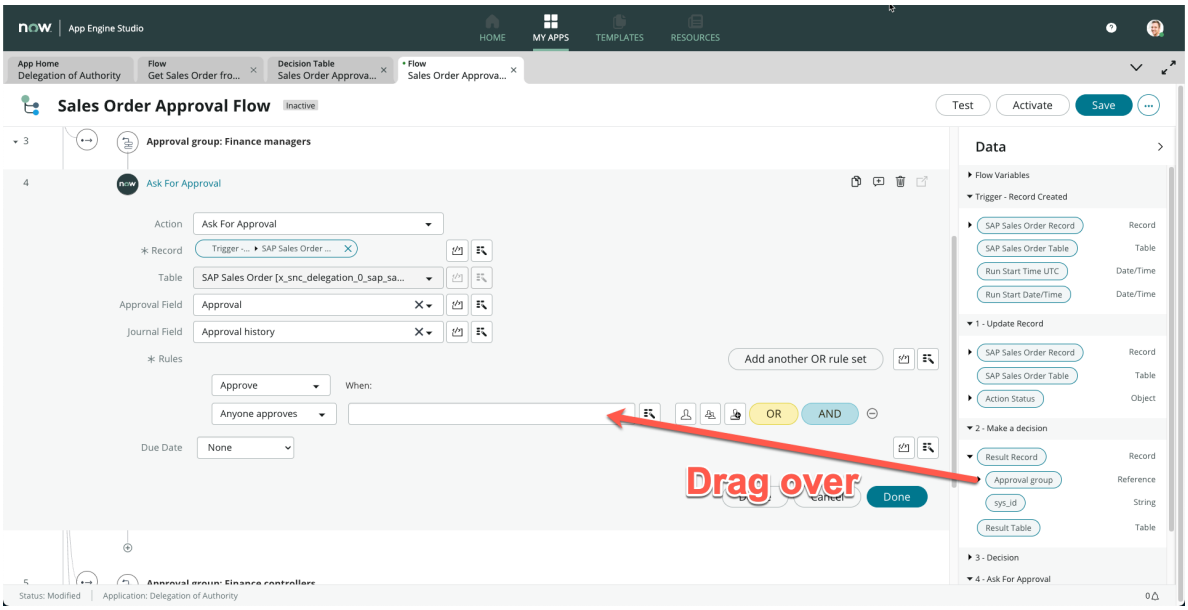


26. Drag **SAP Sales Order Record** from **Trigger - Record Created** within the data column on the right, onto the **Record** field

27. Since we extended the **Task** table, the **Approval Field** and **Journal Field** should be automatically populated

28. Under rules, change **-Choose approval rule** to **Anyone approves** (We are not factoring an Rejection flow here)

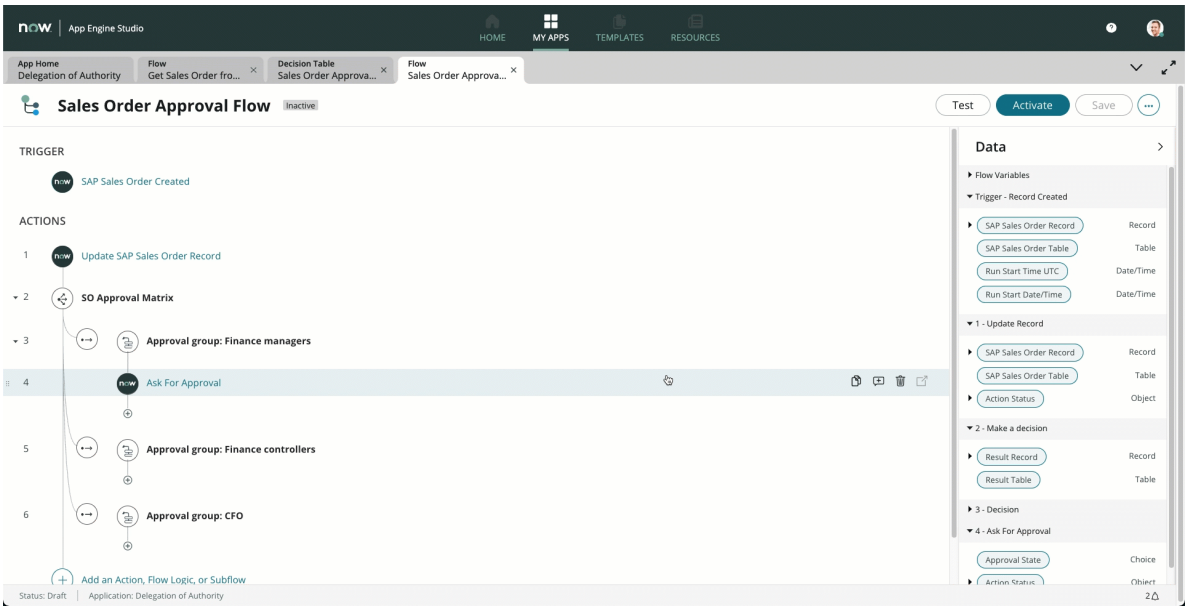
29. Drag over **Approval group** reference from **2 - Make a decision > Result Record** onto the empty field



30. Click **Done**

31. On the **Ask For Approval** action you just added, click on the **Duplicate** icon twice

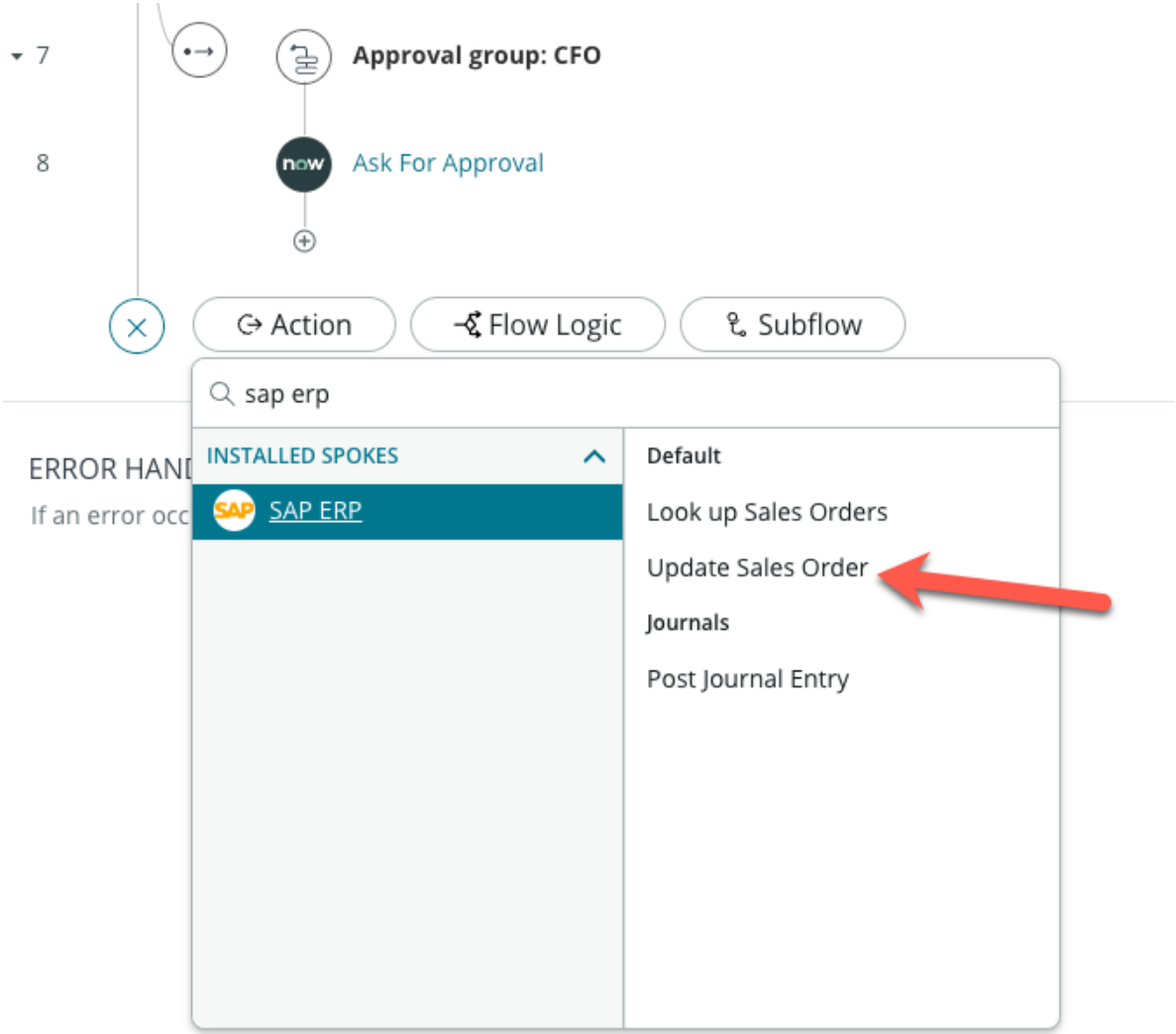
32. Drag each of the 2 duplicated actions under each of the other branches



33. Outside of the Decision flow, click **Add an Action, Flow Logic, or Subflow**

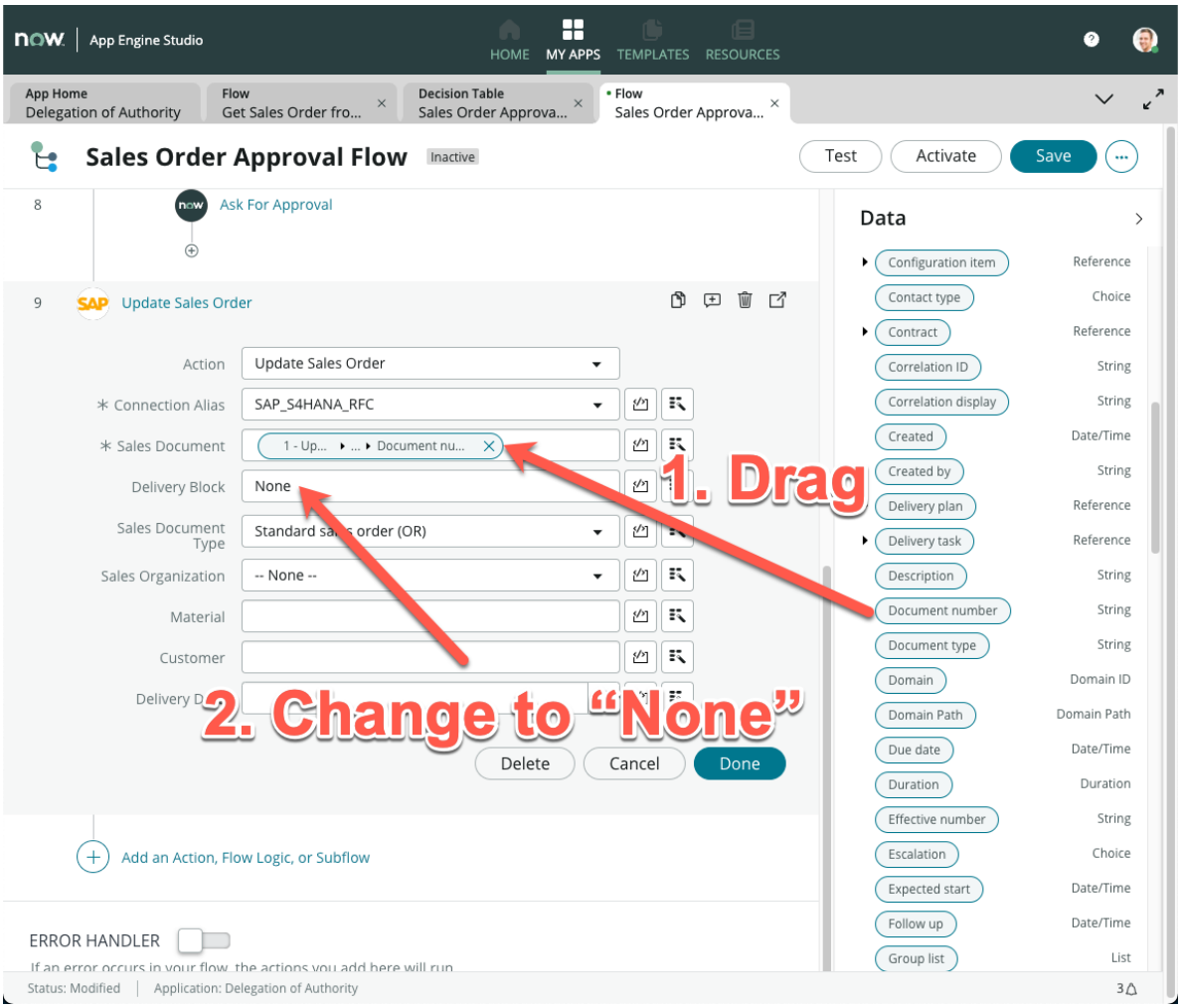
34. Click **Action**

35. Search for **SAP ERP**, then select **Update Sales Order**

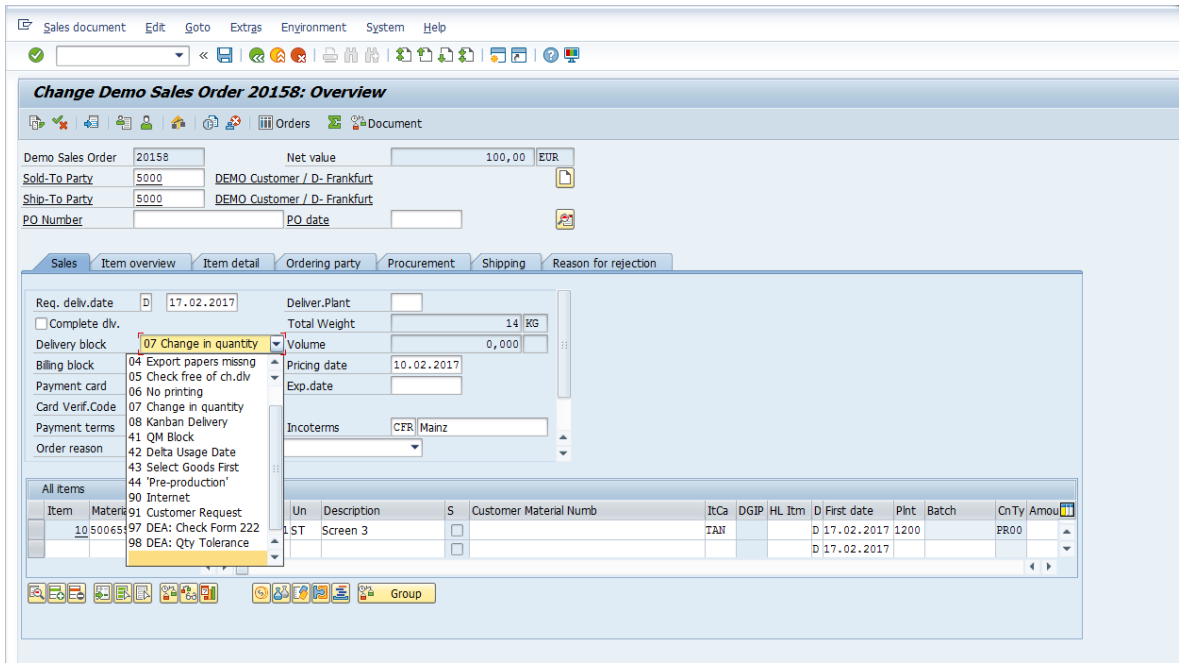


36. Under the **Sales Document** field, drag the **Document number** pill from **Trigger - Record Created > SAP Sales Order Record** within the data column on the right, onto the field

37. Change the **Delivery Block** field to **None**



38. Click **Done**

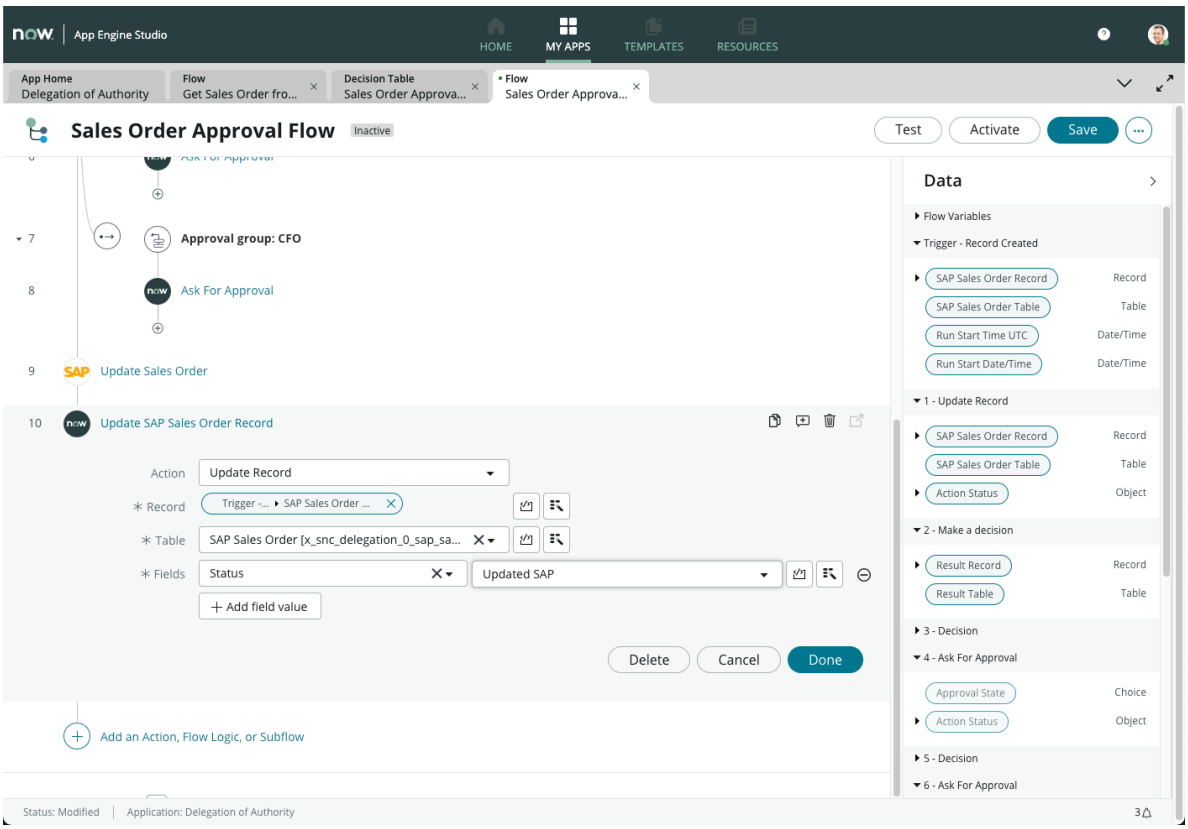
> For reference, this action will remove the delivery block as you see in the image from SAP ECC below, which will allow the order to move to the next stage, which is generally creating a delivery document.
>
> 
>

39. Under the *Update Sales Order* action, add a new **Action**

40. Select **Update Record** under **ServiceNow Core**

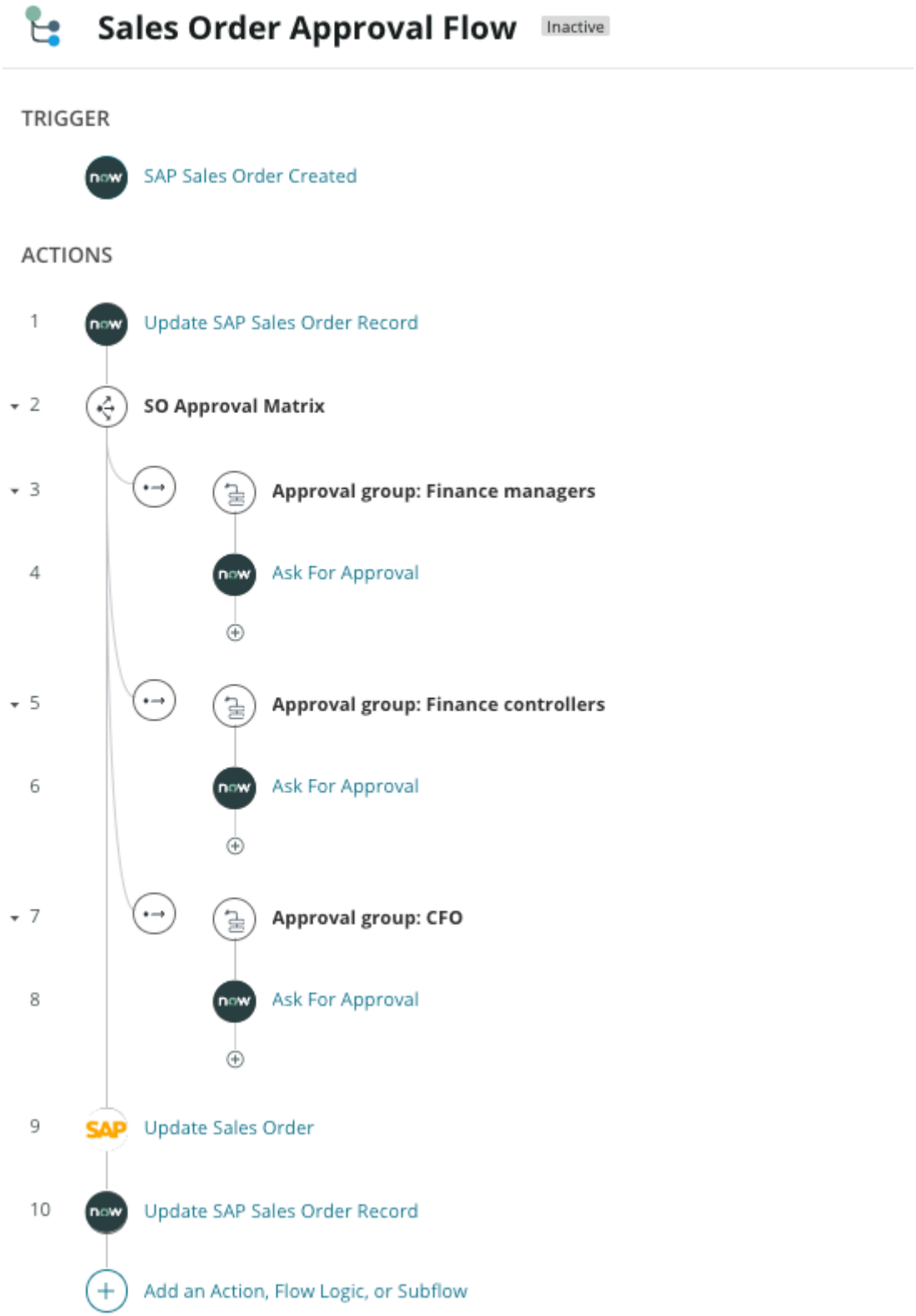
41. Drag the **SAP Sales Order Record** pill from **Trigger - Record Created** within the data column on the right, onto the **Record** field

42. Add a field, select **Status** and select **Updated SAP**



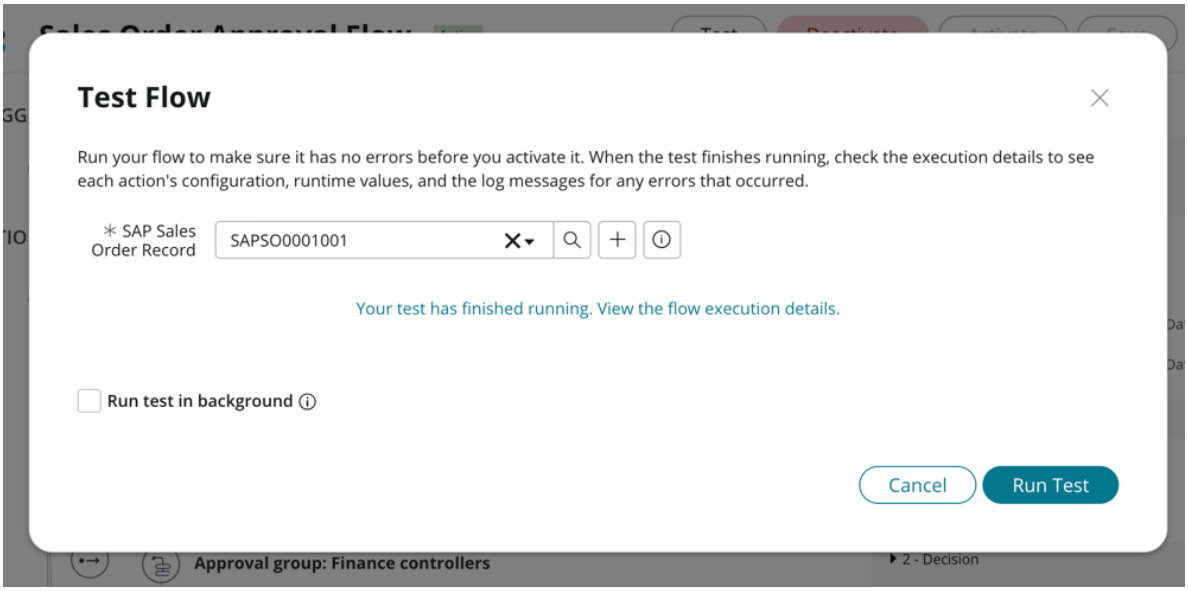
43. Click **Done**

44. Your entire flow should now look like this



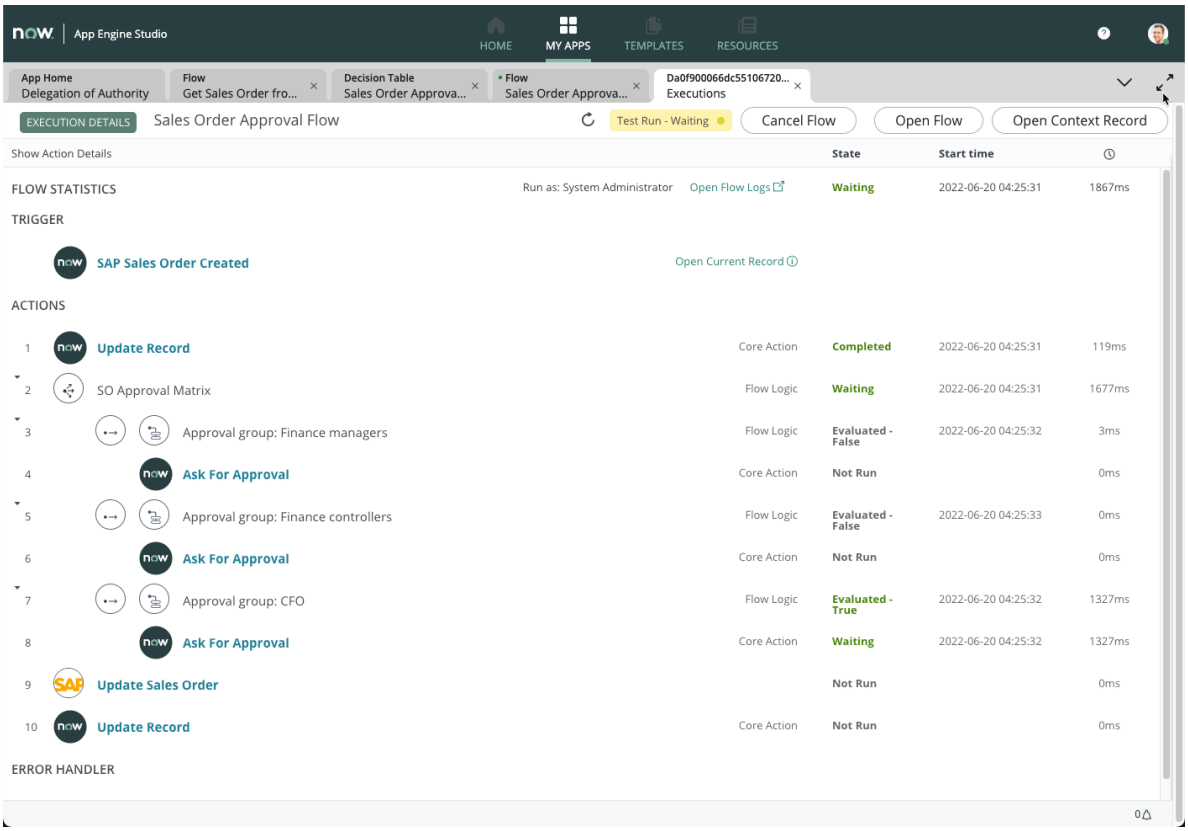
45. Click **Activate** on the top right

46. Click **Test**, and select the record that was created via the first flow we created, it should be **SAPSO0001001**



47. Click **Your test has finished running. View the flow execution details**

48. Check the execution flow, did it match our Approval Matrix? Your results will vary based on your record and should now be in Waiting for approval state after identifying the right approval group



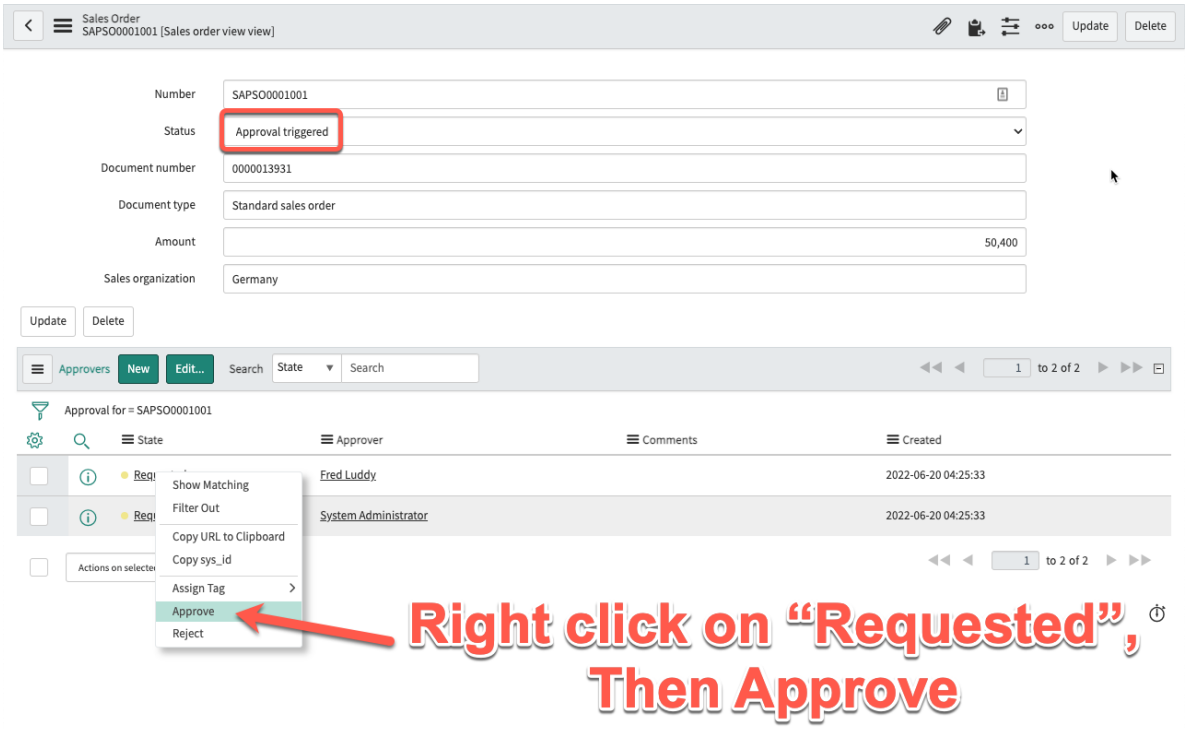
49. If it did not, examine your Decision Builder table again to see if there was anything you missed.

50. Open up the Sales Order record tab (SAPSO0001001)

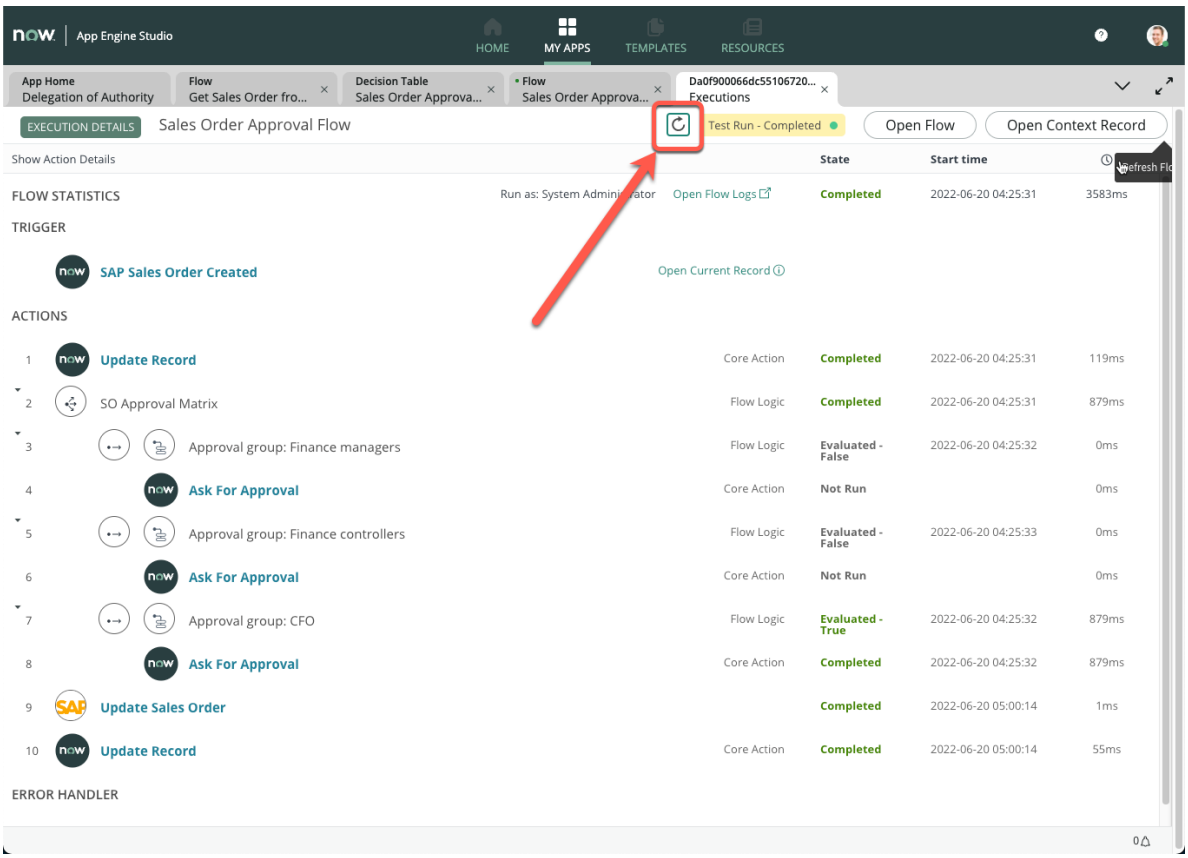
51. **Refresh** the page

52. Confirm that the **Status** has changed to **Approval triggered**

53. Right click on any **Requested** approval records, and click **Approve**



54. Return to the flow execution screen and click the **Refresh** icon



55. Your flow should now be completed and the record updated

As you can imagine, now that we have these 2 workflows designed, one will essentially trigger off the other: Our custom app will pull Sales Orders from SAP every 2 hours, and for any new Sales Orders picked up, it will be processed via our Decision Table, before routing to the right parties for Approval. Only after approval, will we then update the same Sales Order back in SAP (to remove the delivery block).

Well done! You've come to the end of Exercise 1, where you've built a very powerful approval matrix alternative that sits on top of any SAP document to drive the Delegation of Authority process. Of course, this is only the very start, and you can start to include platform capabilties such as SLAs, Approval Delegation periods, Conversational integrations, Mobile Approvals and more that makes the entire process seamless.

Exercise 2: Automating Financial Close - The Journal Entry experience

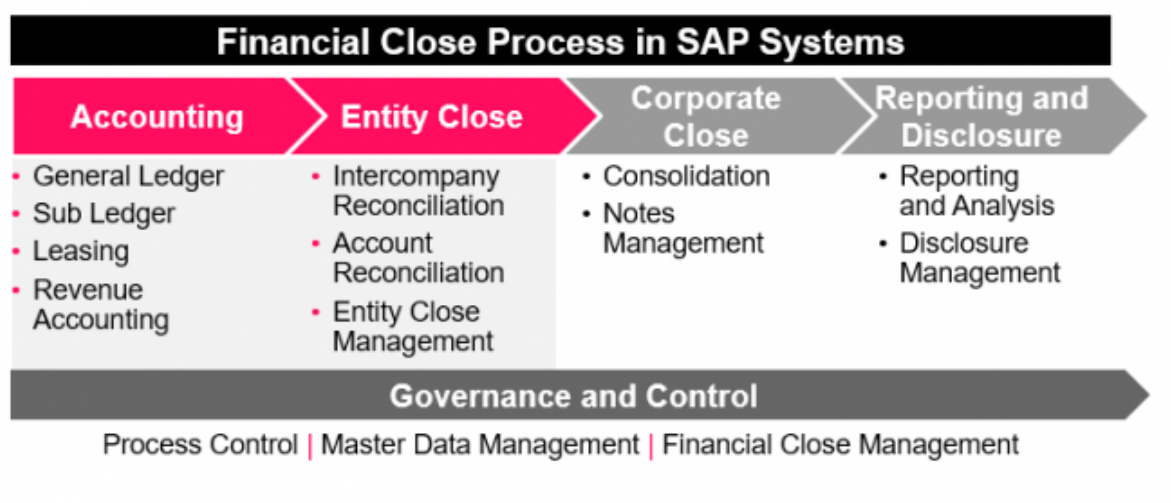
Introduction

The financial close is a critical process that happens regularly (Monthly, Quarterly, Yearly) in a business with the general goal of producing financial reports representative of the company's true financial position. This is a stressful exercise and there is typically a checklist of activities that the finance team has to undertake, for example: Perform inventory count, reoncile account and post Journal Entries, update cash flow statements, balance petty cash funds, any many more.

Many of these activities also tie back to SAP as the system of record, but this is essentially is a workflow activity, not a record centric one! Navigating across SAP, updating, checking then reporting then becomes a huge headache. In fact, in a 2021 SAPinsider report, 58% of organizations see financial close as the biggest pain point for financial processing, and only 30% of organizations have implemented solutions to automate the SAP financial close process, far below areas such as Account Receivable or Accounts Payable. So can we make this better? Obviously!

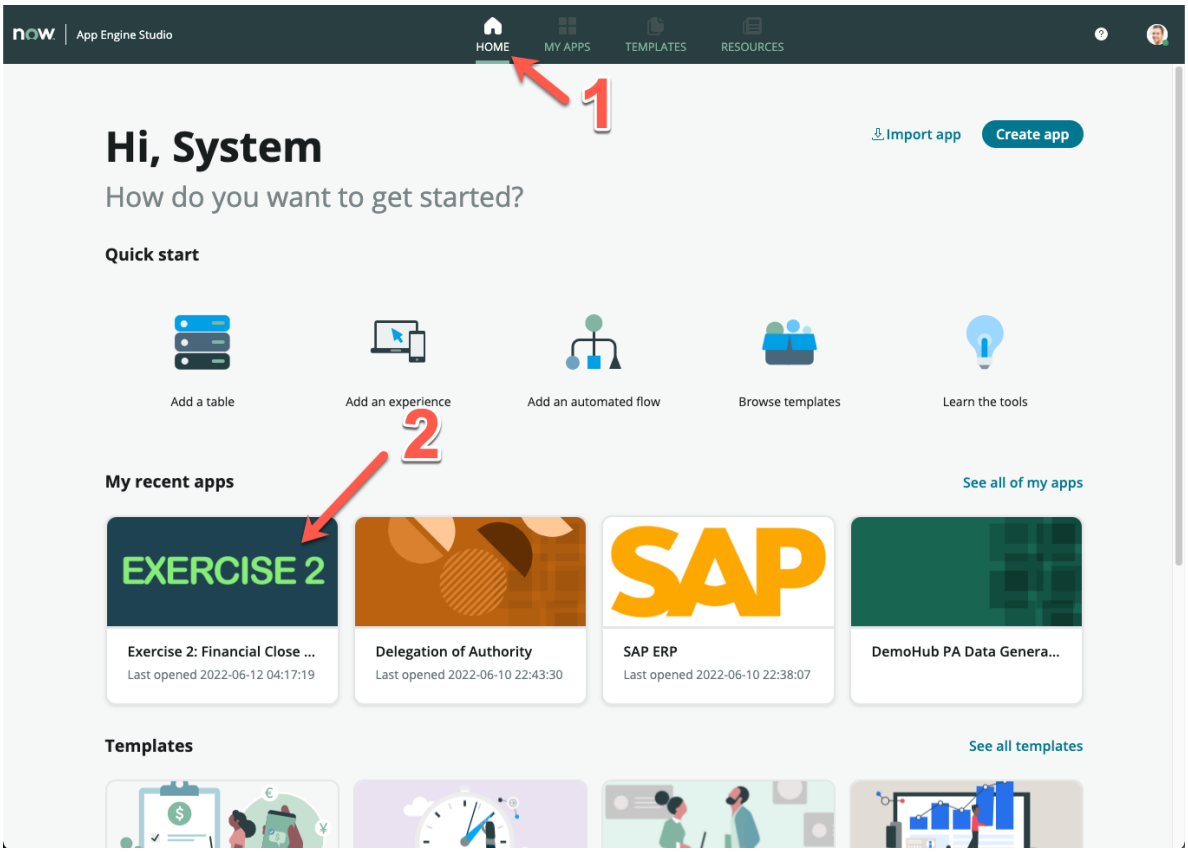
In this exercise, let's focus on one of the key activites, which is Journal Entry posting from ServiceNow to SAP.

> A journal entry is used to record a business transaction in the accounting records of a business. A journal entry is usually recorded in the general ledger; which is then used to create financial statements for the business.The logic behind a journal entry is to record every business transaction in at least two places (known as double entry accounting).
> In the following diagram, you can se that posting journal entries in the general ledger is among the first tasks in the financial close process. Not on finance users, but sometimes business users as well have to post hundreds or thousand of manual journal entries in each period!



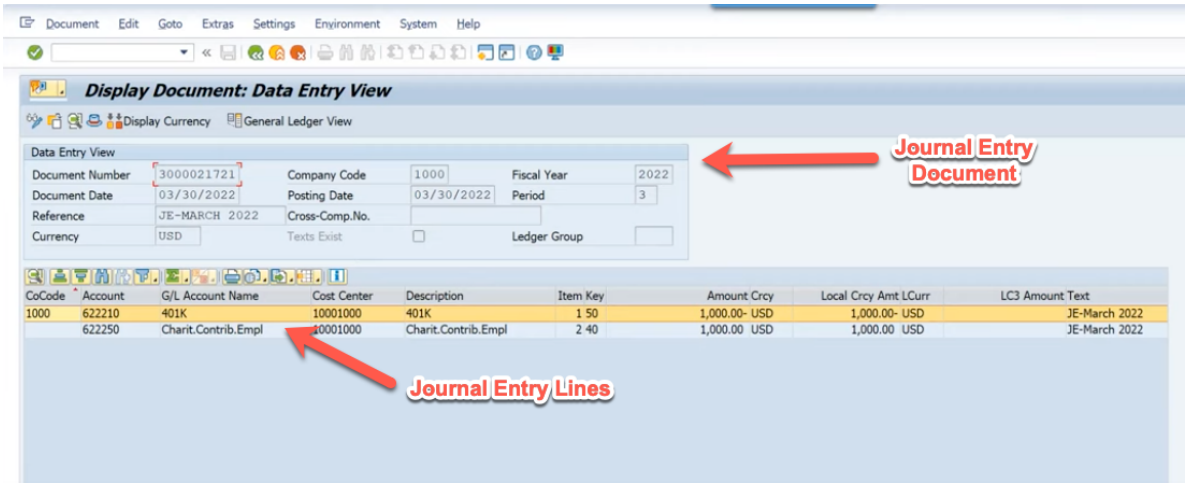
Since you've already gone through the excercise of creating a new app and tables and forms, the setup of these application files for this exercise is already done for you.

1. Navigate back to *App Engine Studio Home* by clicking on the **Home** icon
2. Under *My recent apps*, look for **Exercise 2: Financial Close ....** If not found, click on **See all of my apps**, then search for **Exercise 2**



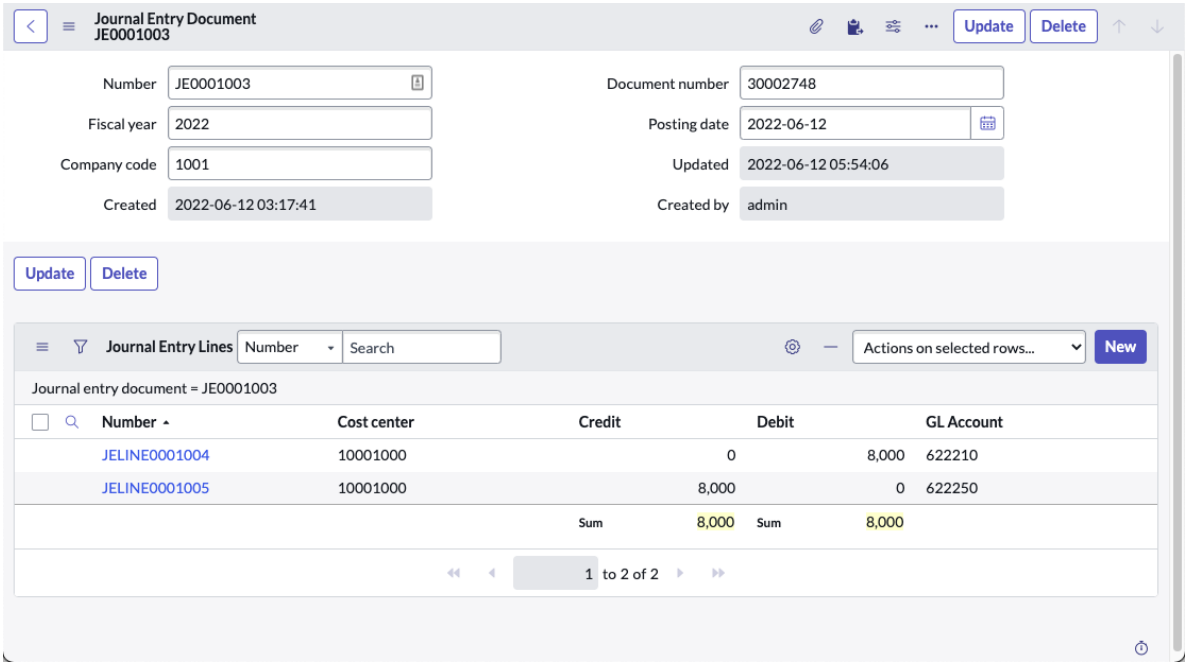
3. Under *Data*, you can see 2 tables added in this application: **Journal Entry Document** and **Journal Entry Lines**

4. First, let's see what a Journal Entry Document looks like in SAP



> The Journal Entry Document in our scenario refers to the "Document Header", and Journal Entry Lines are row items nested under the Document Header.

5. Now let's see how this works with our two related tables. Similar? You could have built this in App Engine Studio in less than 10 minutes ;)



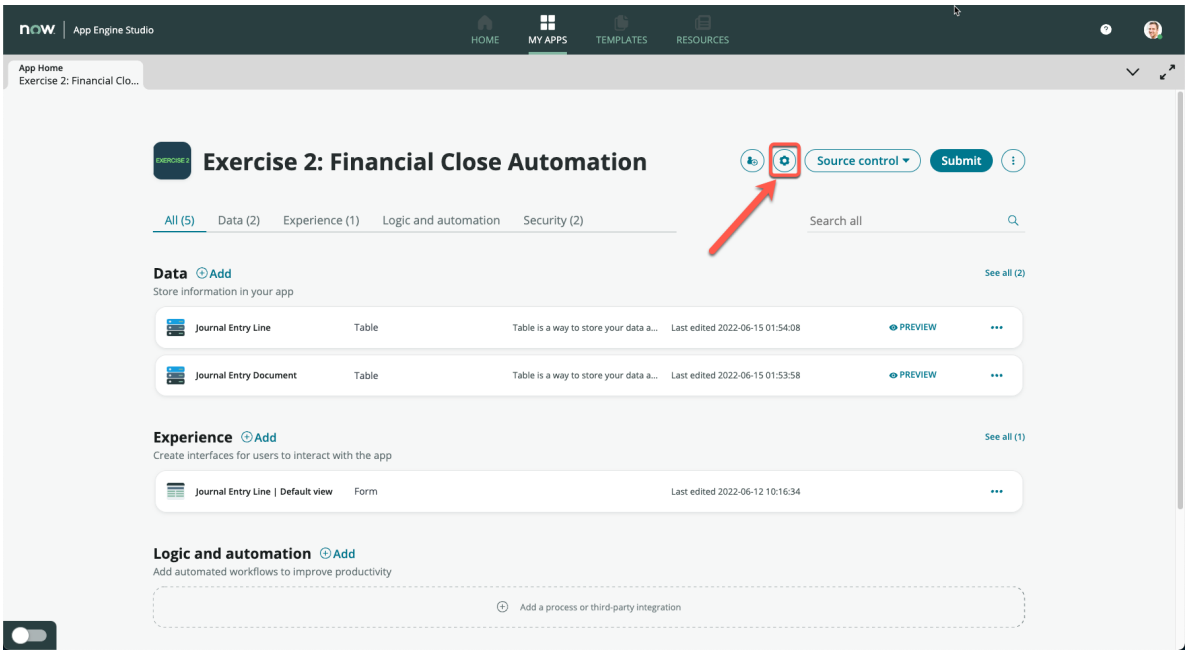
Building the user experience layer

For this part of the exercise, we will get our hands a little dirtier in building the Record Producer for a user to enter the Journal Entry record. As we now need to enable the user to attach a dynamic number of Journal Entry Lines against a parent Journal Entry Document, there is some scripting needed as part of implementing this feature via a Multi-row Variable Set.

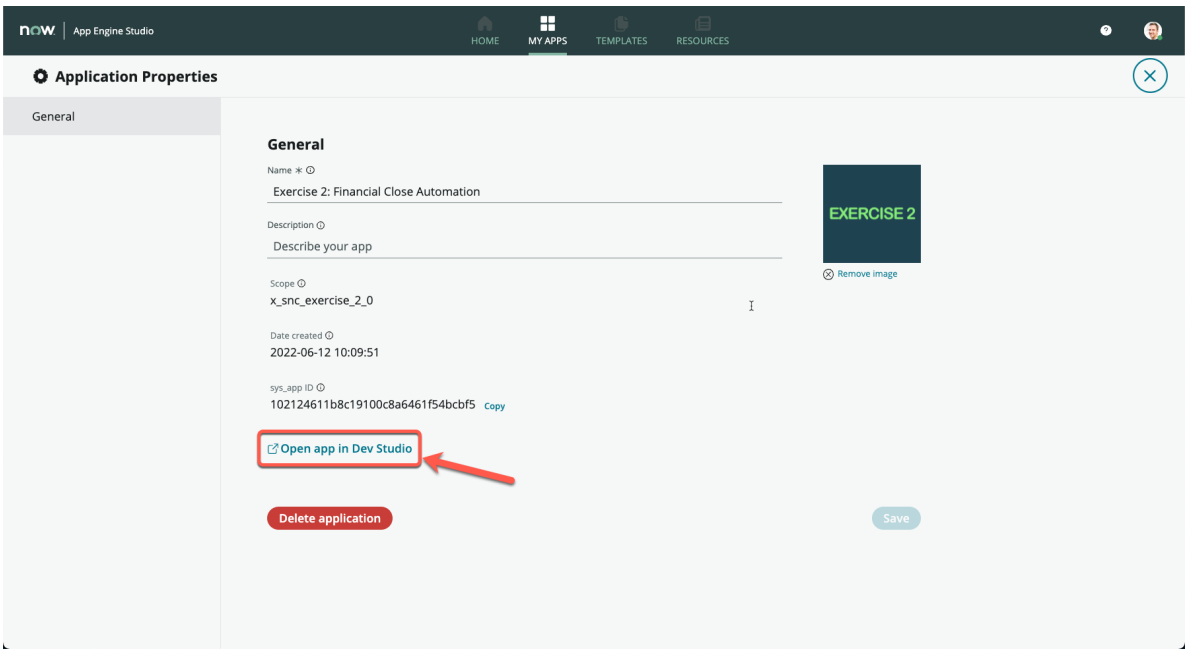
> **What is a Multi-row Variable Set (MVRS)?**
> From ServiceNow documentation: Use a multi-row variable set (MRVS) to capture variable data in a grid layout while submitting a catalog item request for a group of entities. For example, for HR during the reorganization of employees, a single record producer should be able to capture the relevant information such as the department and manager for a group of employees

First, let's create the variable set. This time, for this part, we will be working within Studio IDE.

1. On the App Home Screen, click the **Gear** icon



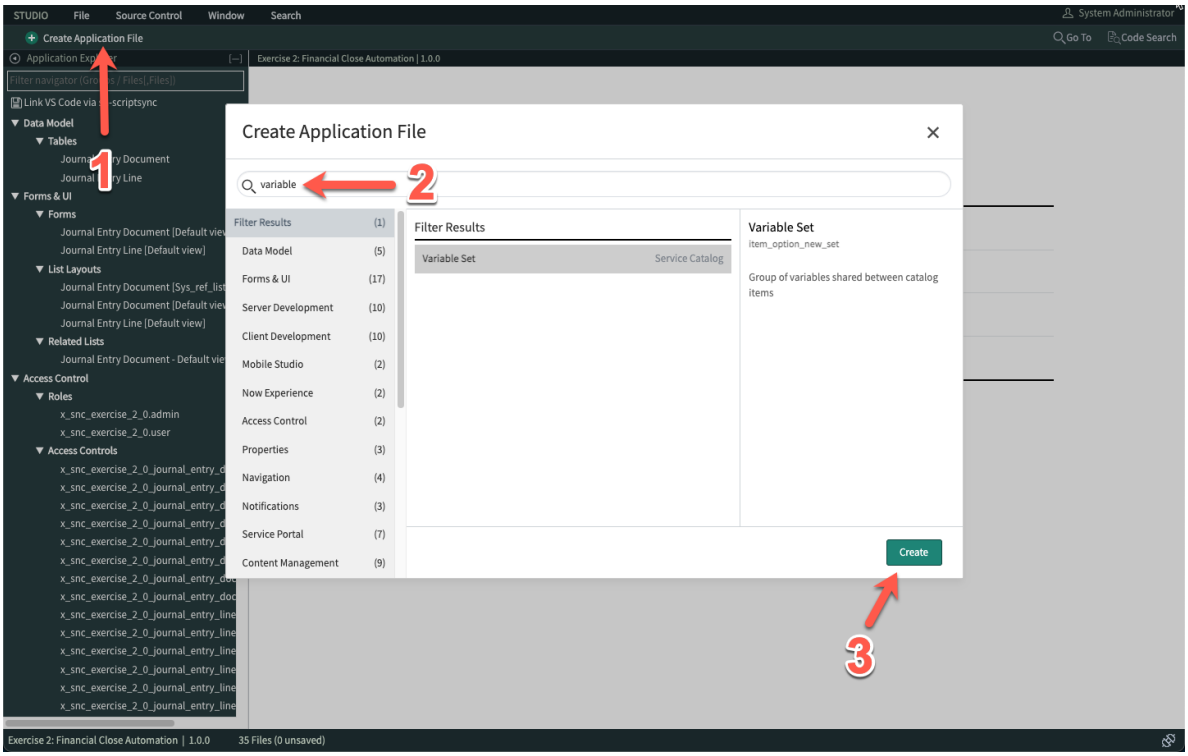
2. Click **Open app in Dev Studio**



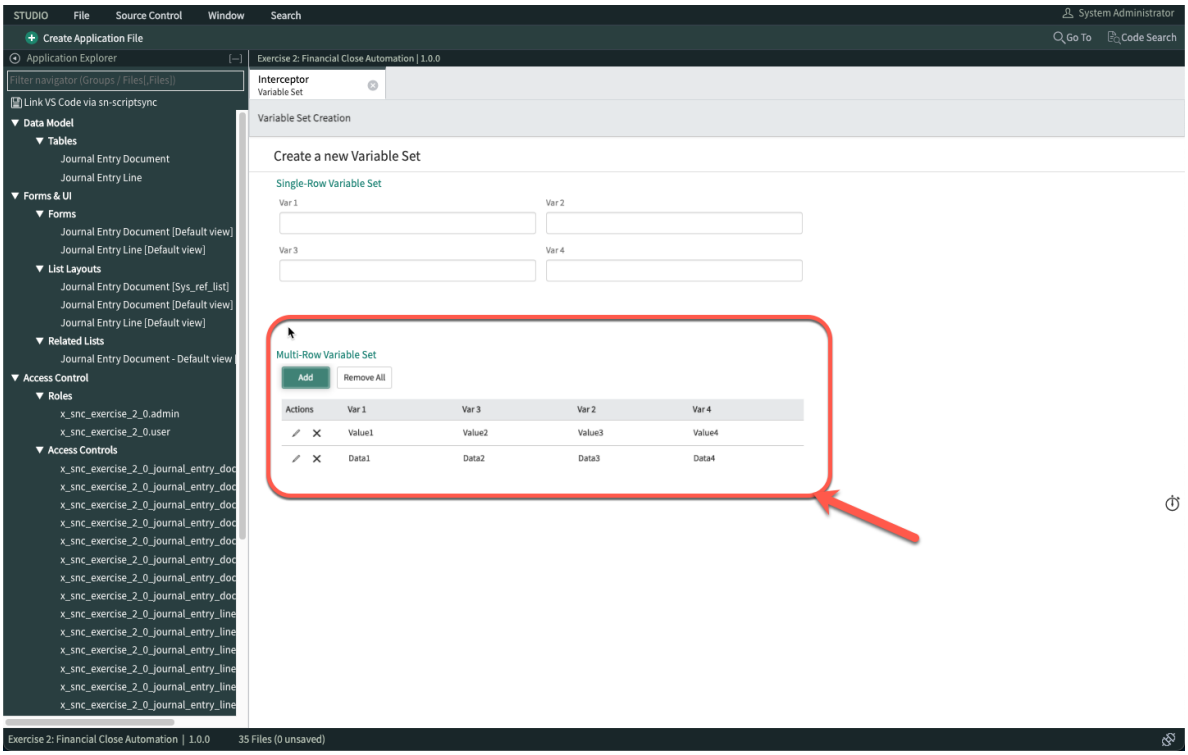
3. A new browser tab will open in *Studio IDE*

4. Click **Create Application File** on the top left

5. Search **Variable Set**, then click the **Create** button on the bottom right

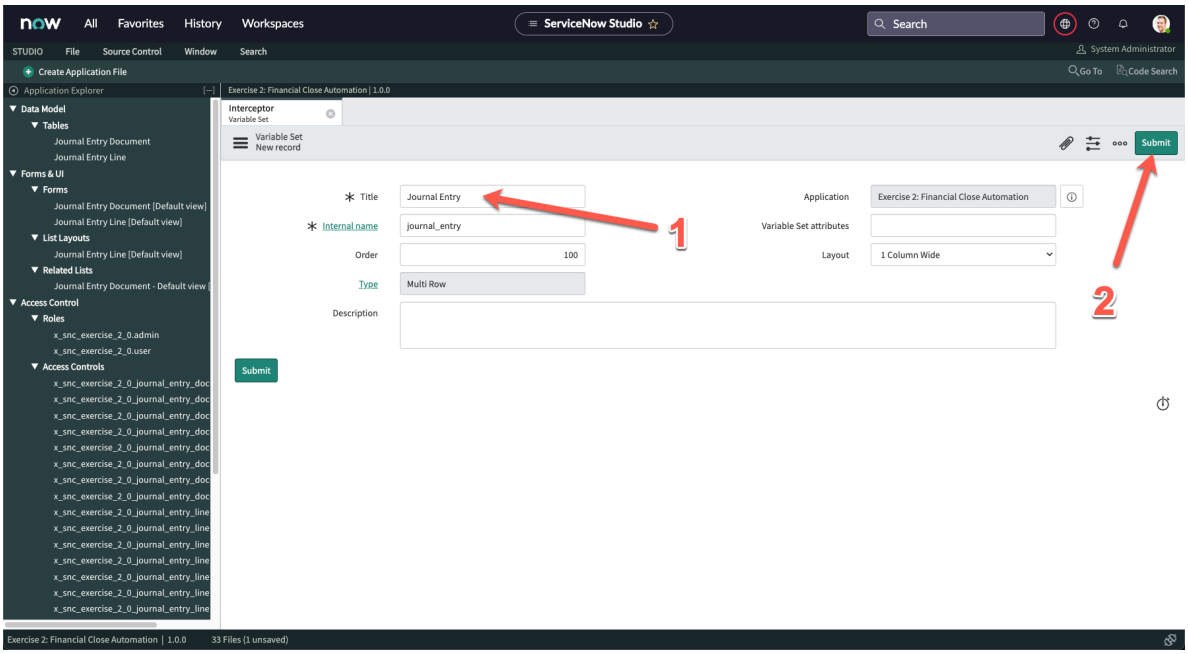


6. Click **Multi-Row Variable Set**



7. On the **Variable Set** form, fill in the Title as **Journal Entry**, then click out of the field. The **Internal name** should be automatically populated as **journal_entry**. Leave it as such.

8. Click **Submit**



9. On the left panel, scroll down until you see **Service Catalog > Variable Sets > Journal Entry**

10. Click on **Journal Entry**

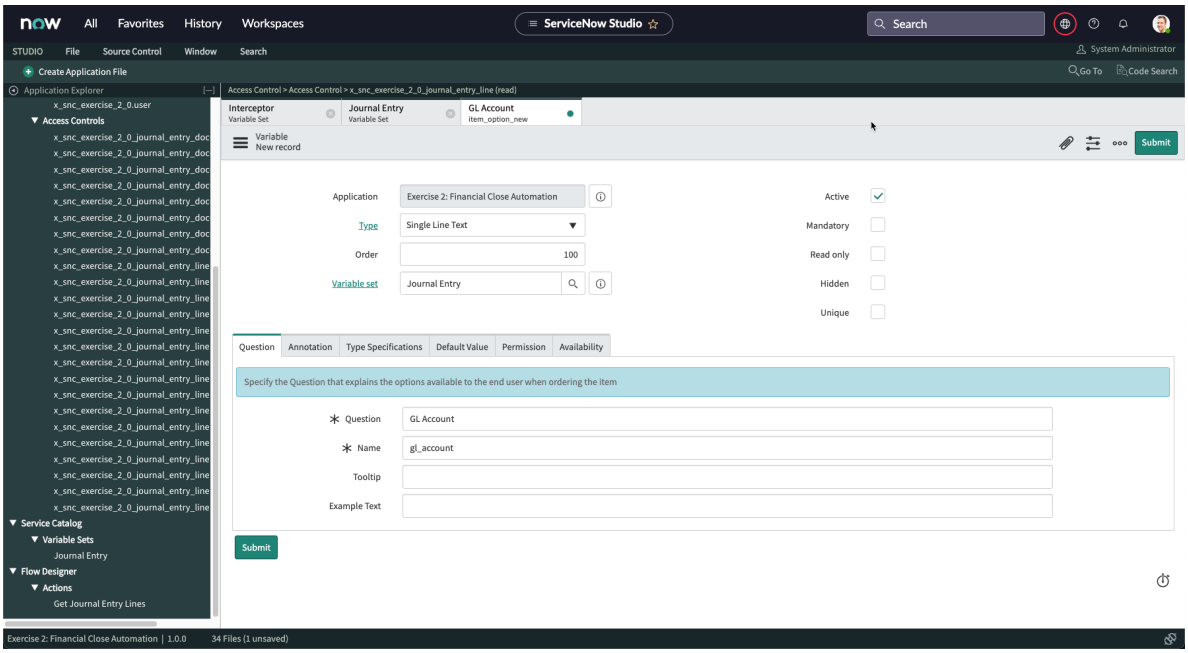
11. On the Variable Set, you should now see more related lists below.

> **NOTE:** It is important you follow the next few steps accurately, if not there might be some problems in the later part of this lab.

12. Under **Variables**, click **New**

13. On the new **Variable** screen, enter **100** under *Order*

14. Under the **Question** tab, enter **GL Account** under *Question*

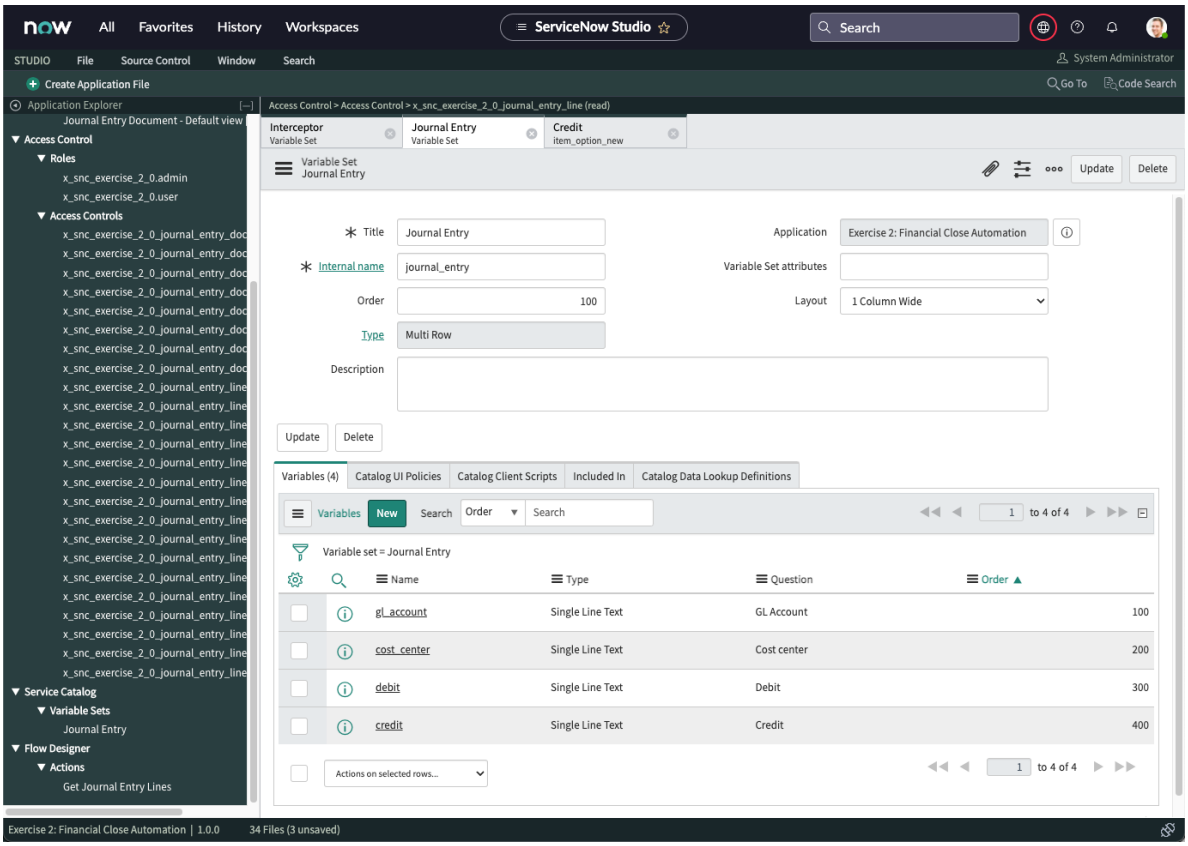


15. Click **Submit**

16. Create 3 more variables under the **Journal Entry** Variable Set (Repeat steps 12 to 14)

Order	Question	Name (autofilled)
200	Cost center	cost_center
300	Debit	debit
400	Credit	credit

17. Your Variable Set should now look like this



Creating a Record Producer

Now that we have the variable set to capture Journal Entry Lines, let's use this in a new Record Producer

> We will do this section in ServiceNow Studio IDE for quicker access, but you could definitely use catalog builder as well.

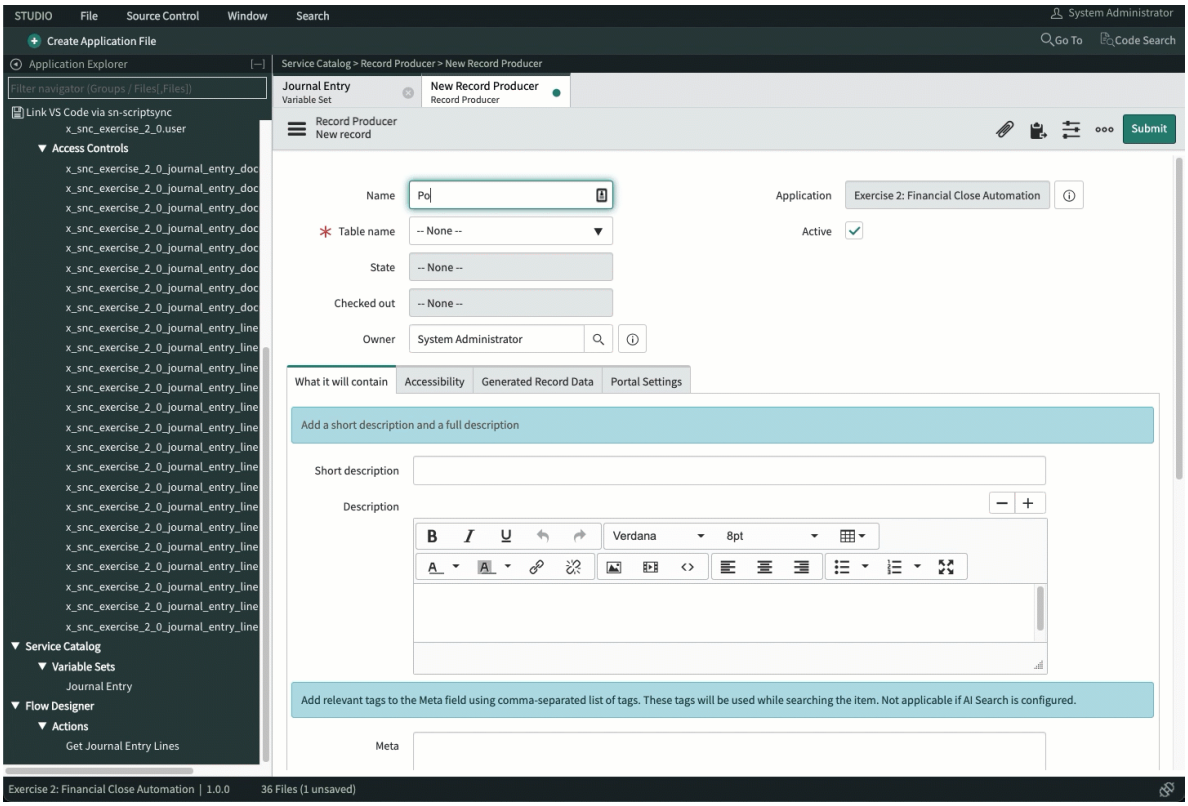
1. Click **Create Application File**, then search for and create **Record Producer**
2. In the **Record Producer** screen, fill in the form as follows

Field	Entry
Name	Post Journal Entry

Field	Entry
Table name	Journal Entry Document
Script (Scroll down)	Copy and paste the script below (Overwrite default script)

```
(function() {

var mrvs = producer.journal_entry;

var l = mrvs.getRowCount();
for(var i = 0; i < l; i++) {
        var row = mrvs.getRow(i);
        var cells = row.getCells();

        var journaldocument = new
GlideRecord('x_snc_exercise_2_0_journal_entry_line');
        journaldocument.initialize();
        journaldocument.setValue('journal_entry_document',
current.getUniqueValue());
        journaldocument.setValue('gl_account', cells[0]);
        journaldocument.setValue('cost_center', cells[1]);
            journaldocument.setValue('debit', cells[2]);
            journaldocument.setValue('credit', cells[3]);
        journaldocument.insert();
}

})();
```

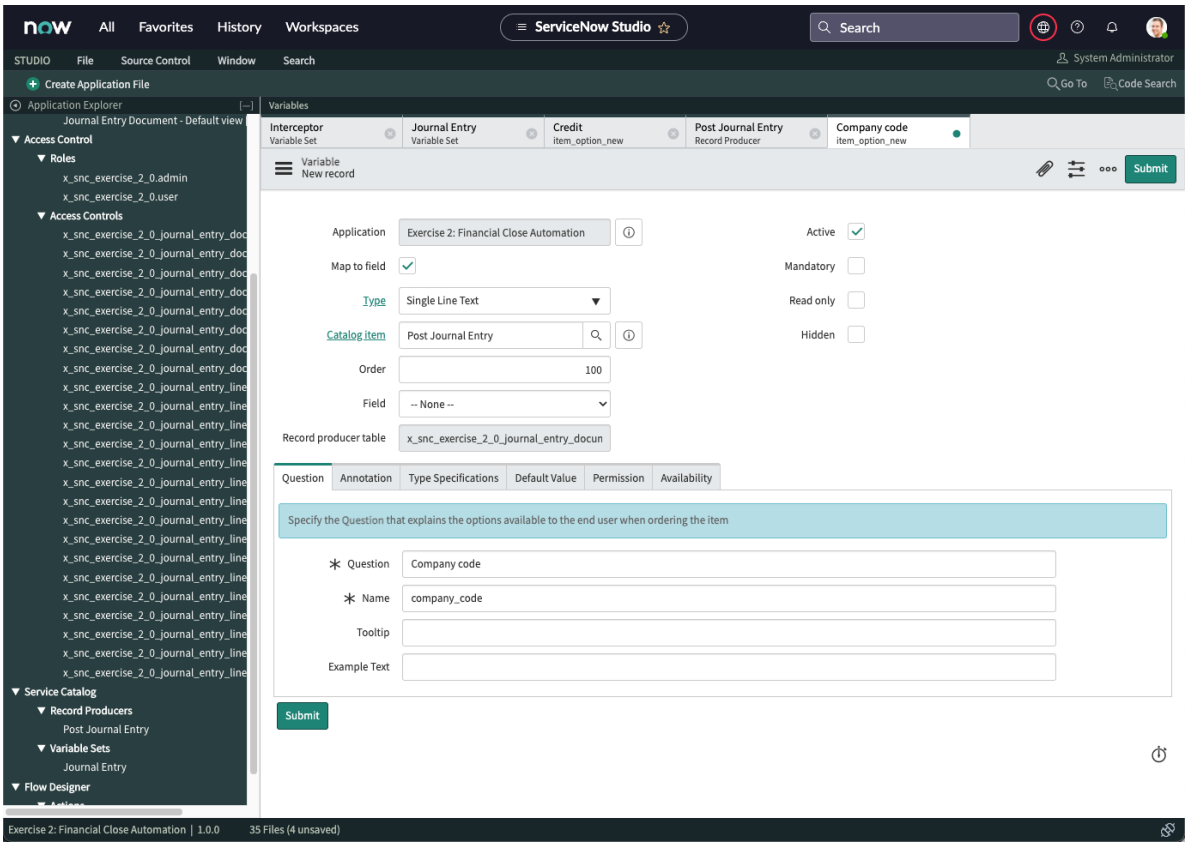
3. Right click on the form header and click **Save**



4. Scroll down to the **Variables** tab and click **New**

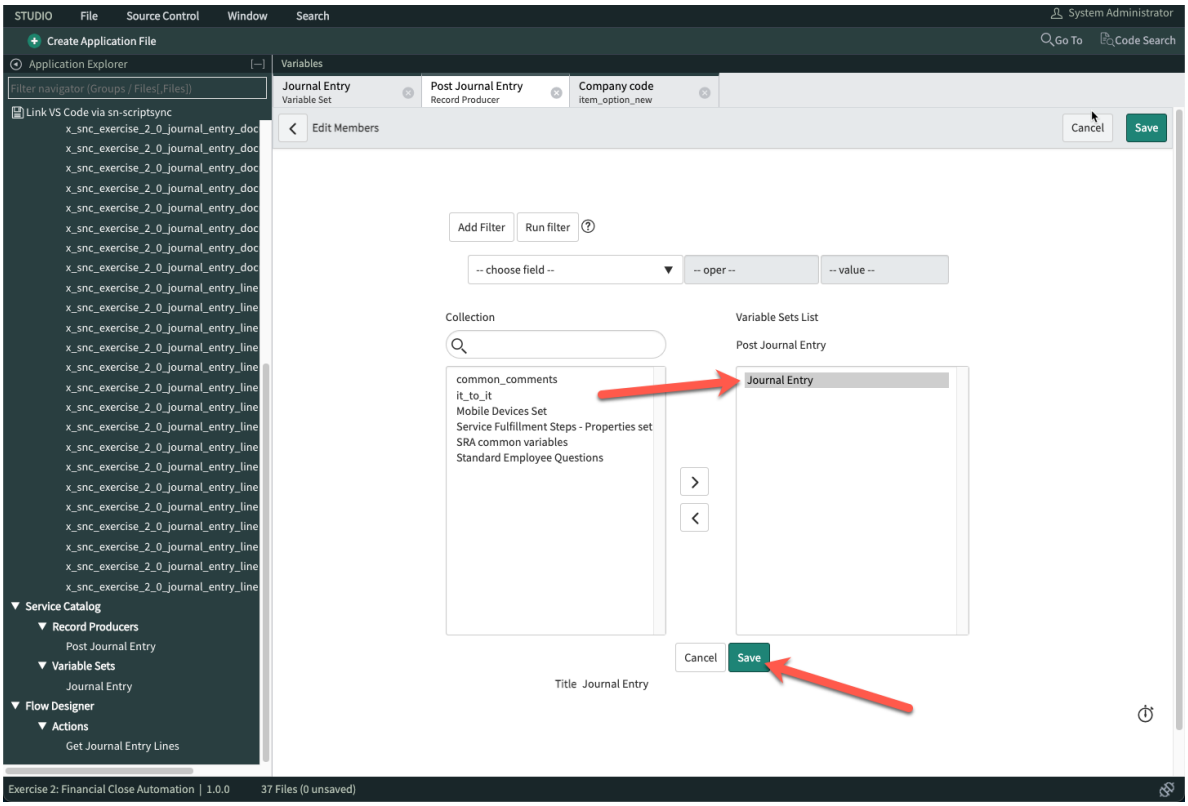
5. Fill in the form as follows

Field	**Entry**
Map to field	**Checked**
Order	100
Field	Company code
Question	Company code

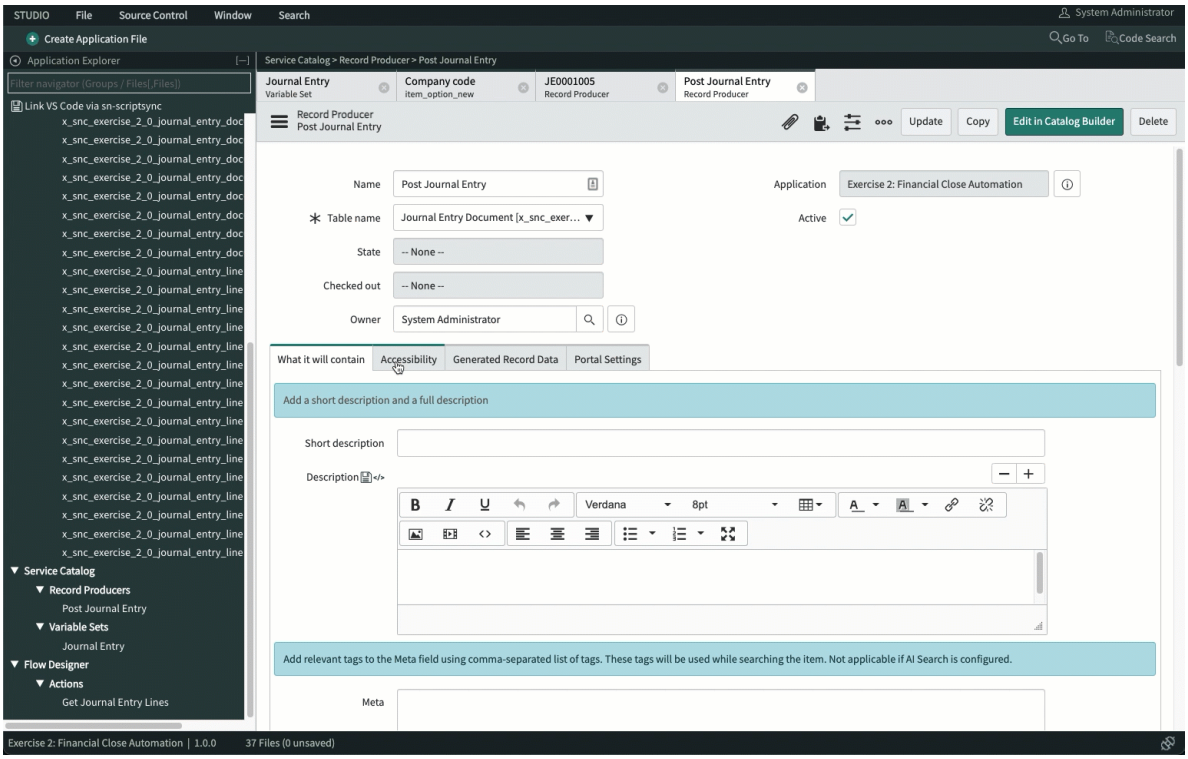


> We will only work on this one variable for the sake of time, but you could fill out the rest of the fields if you feel necessary. (*Fiscal year, Posting date*)

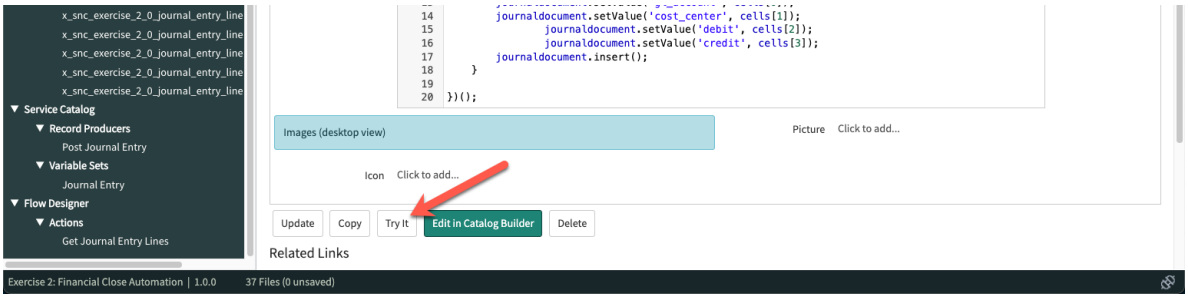
6. Click **Submit**
7. Navigate back to the **Post Journal Entry** Record Producer tab
8. Click the **Variable Sets** tab
9. Click **Edit...**
10. On the **Edit Members** screen, move **Journal Entry** to the right column, then click **Save**



11. Back on the Record Producer screen, click the **Accessibility** tab,
12. Next to **Catalog**, click on the **Lock** icon
13. Where it says *Select target record*, search and add **Service Catalog**
14. Under category, search and select **Can We Help You?**



15. Right click the form header and click **Save**
16. We can now test our Record Producer. Click **Try It**



17. For *Company Code*, enter **1002**

18. Click **Add**

19. On the pop-up modal, enter the following details

Field	Entry
GL Account	622210
Cost center	10001000
Debit	8000
Credit	0

20. Click **Add**

21. Click **Add** once again for a new Journal Entry

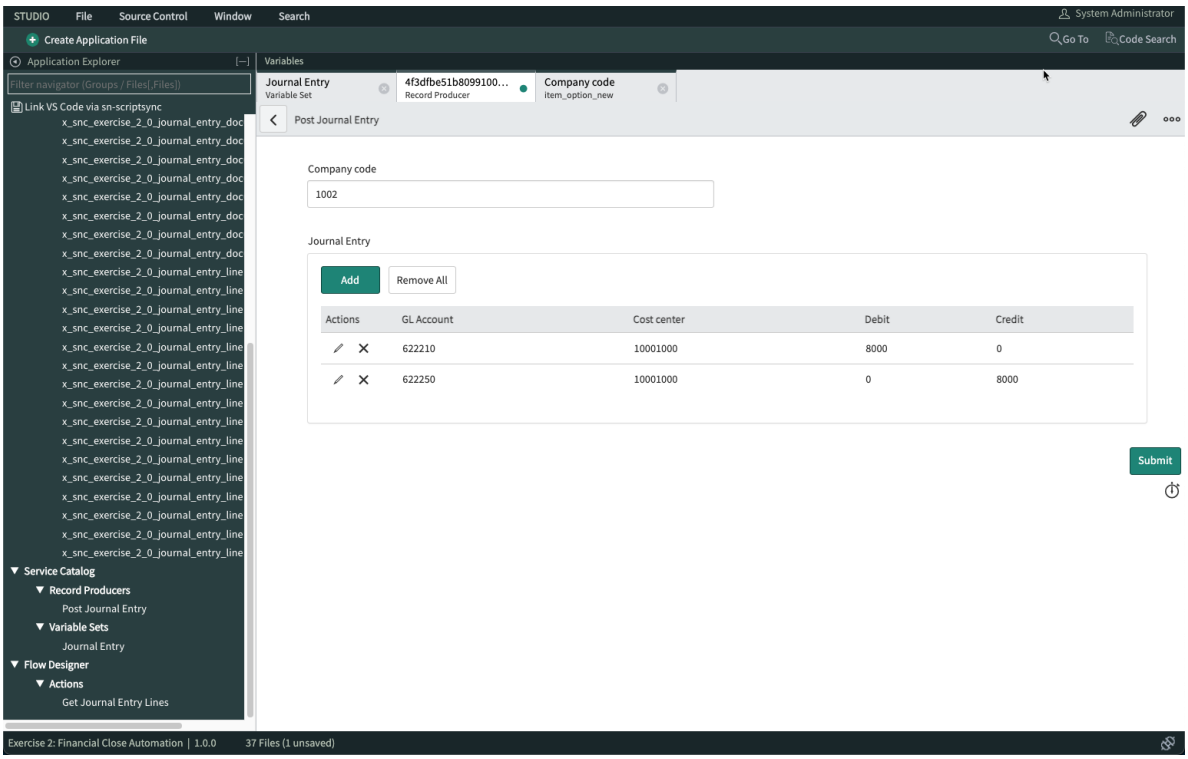
22. This time, enter these details

Field	Entry
GL Account	**622250**
Cost center	10001000

Field	Entry
Debit	0
Credit	8000

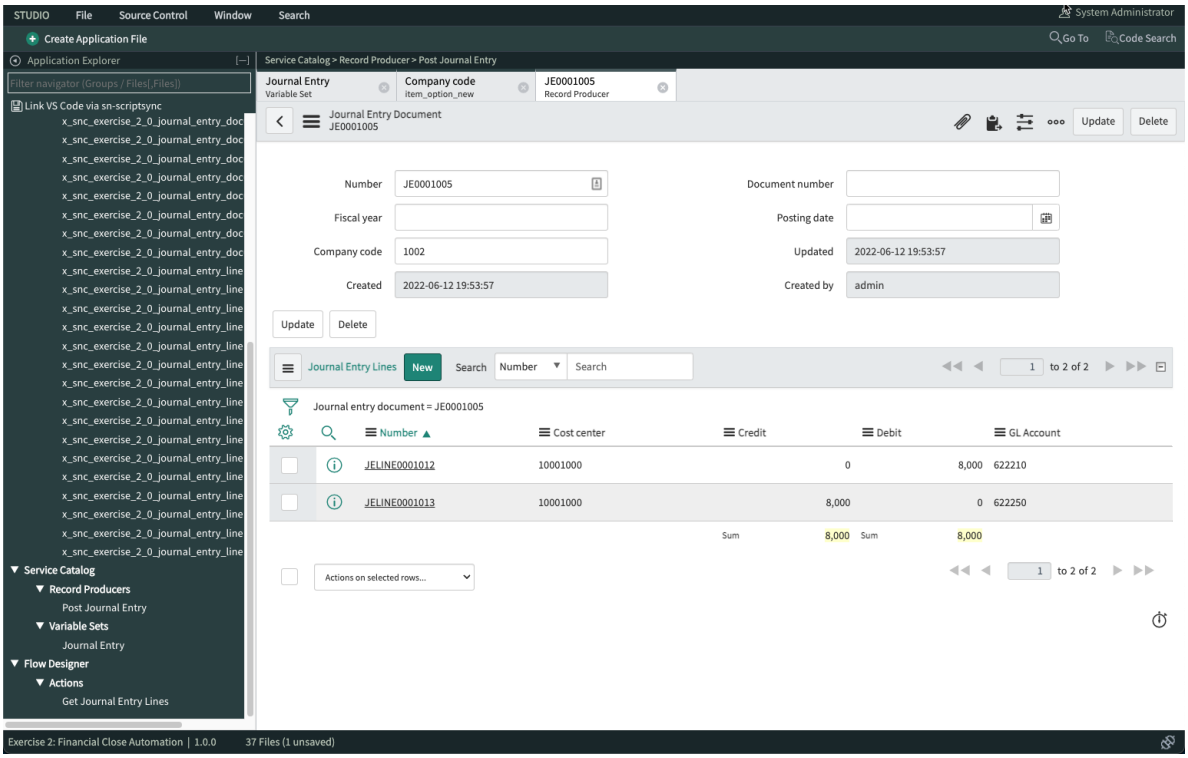
23. Click **Add**

24. Your screen should now look like the following



25. Click **Submit**

26. Your screen should now show the submitted record. Ensure that the *Company code* is captured, and you have the two Journal Entry Lines under the related table



Congratulations! You have now built the front end user experience to capture Journal Entries, and this record producer can be added to any catalog on any portal. It can even be used on the mobile app. Obviously, this is only one approach, and in reality, some of our customers still prefer to have everything entered via an excel template, which is uploaded against the record producer/catalog item before being processed through an import set.

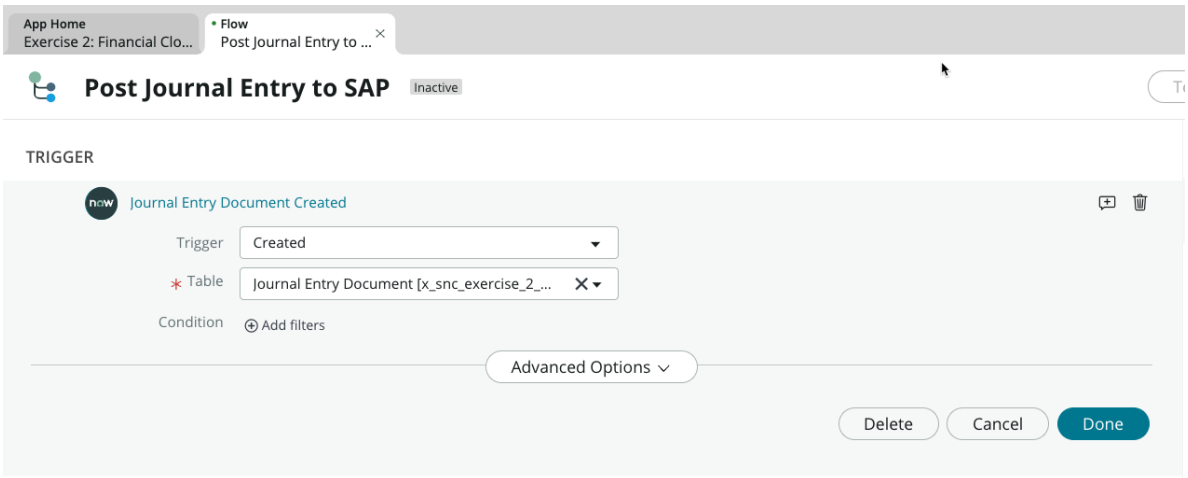
I personally prefer the MVRS approach as there is little room for error. You can also build validation logic in directly via a client script or business rule.

Integrating to SAP

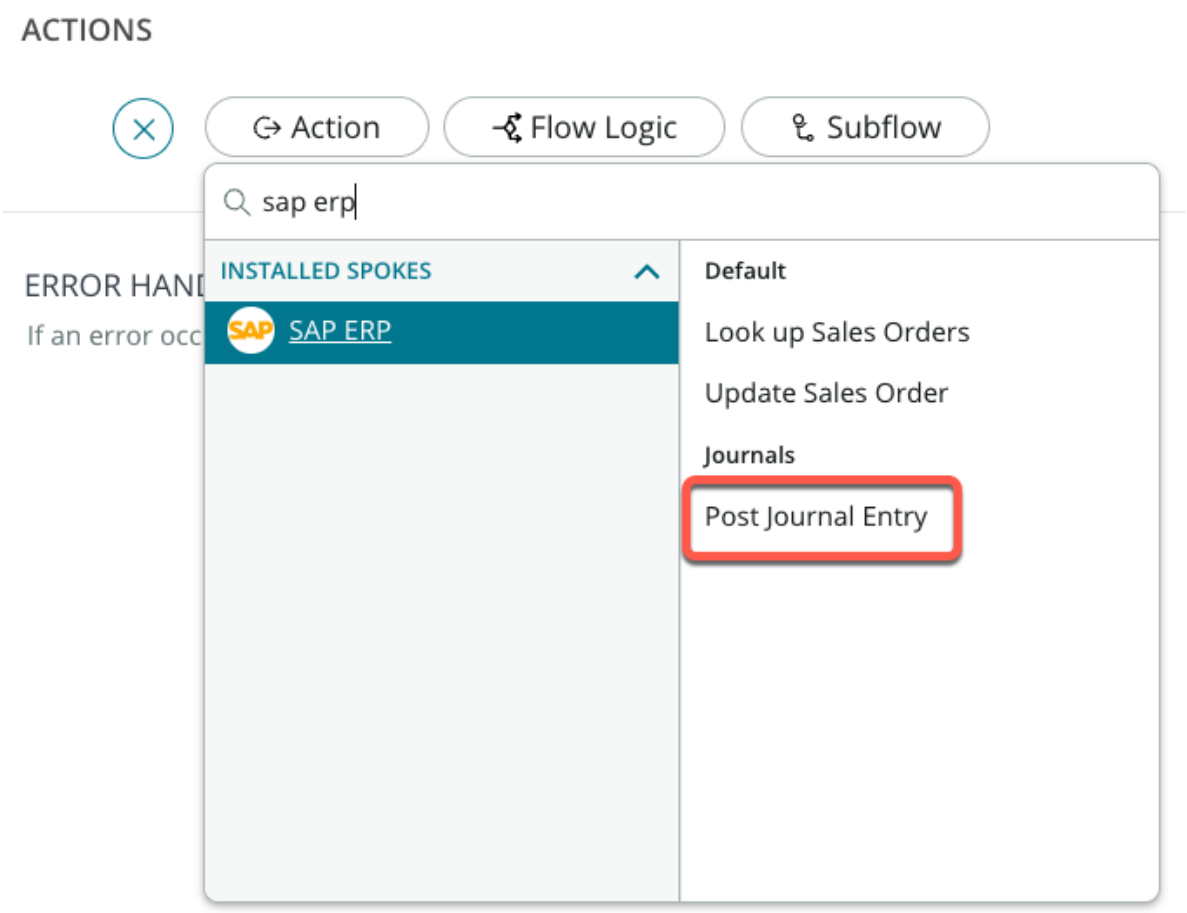
Now that we can capture this data, let's close off the process by building the integration to SAP

1. Navigate back to the App Engine Studio interface, if you have closed it before, open the **Exercise 2: Financial Close Automation** app
2. Under *Logic and automation*, click **Add**
3. Click **Flow**
4. Click **Build from scratch**
5. Under *Name*, enter **Post Journal Entry to SAP**
6. Click **Continue**
7. Click **Edit this flow**

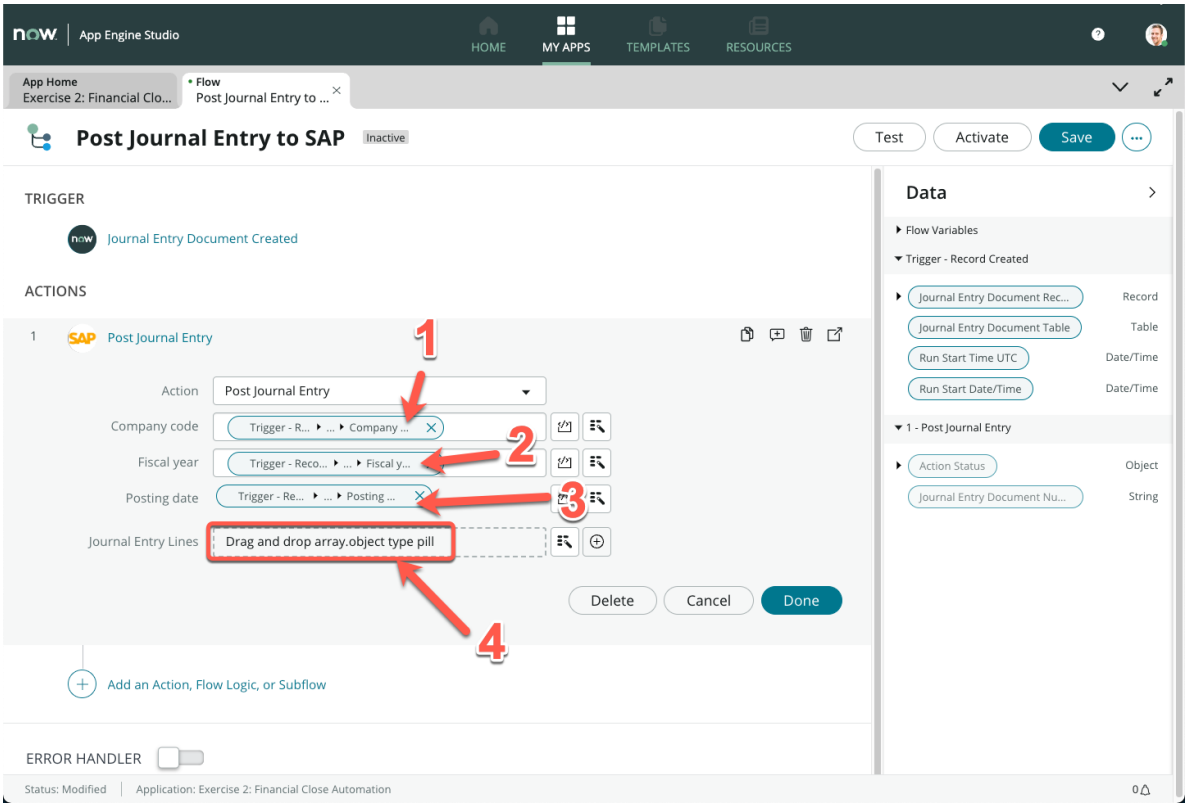
8. Click **Add a trigger**
9. Under **Record**, select **Created**
10. Under **Table**, search and select **Journal Entry Document**



11. Click **Done**
12. Under **Actions**, click **Add an Action, Flow Logic, or Subflow**
13. Click **Action**
14. Search for **sap erp**, then select **Post Journal Entry**



15. Map the data pills for each of **1. Company code, 2. Fiscal year** and **3. Posting date** from the **Trigger - Record Created** step Record to the corresponding fields in the action



16. However, notice that for the **Journal Entry Lines** field, it only accepts **array.object** type pills. Recall how we structured our table earlier, there are multiple Journal Entry Lines against one Journal Entry Document. What we need to do here is get all the Journal Entry Lines linked to this trigger record.

17. Fortunately, the action to convert a list of records into an array object has already been configured for you.

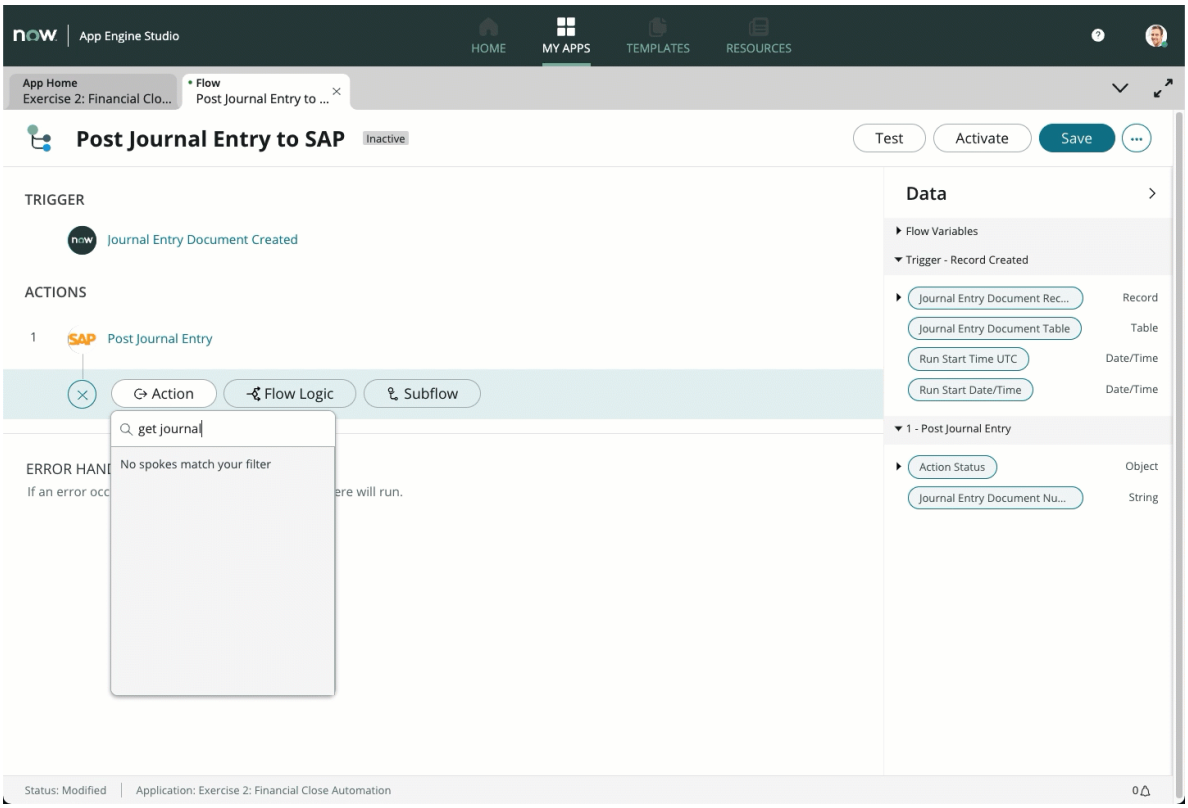
18. Click **Add an Action, Flow Logic, or Subflow**

19. Click **Action**

20. Search **Get Journal Entry Lines**, and select the one action under the Exercise 2 spoke

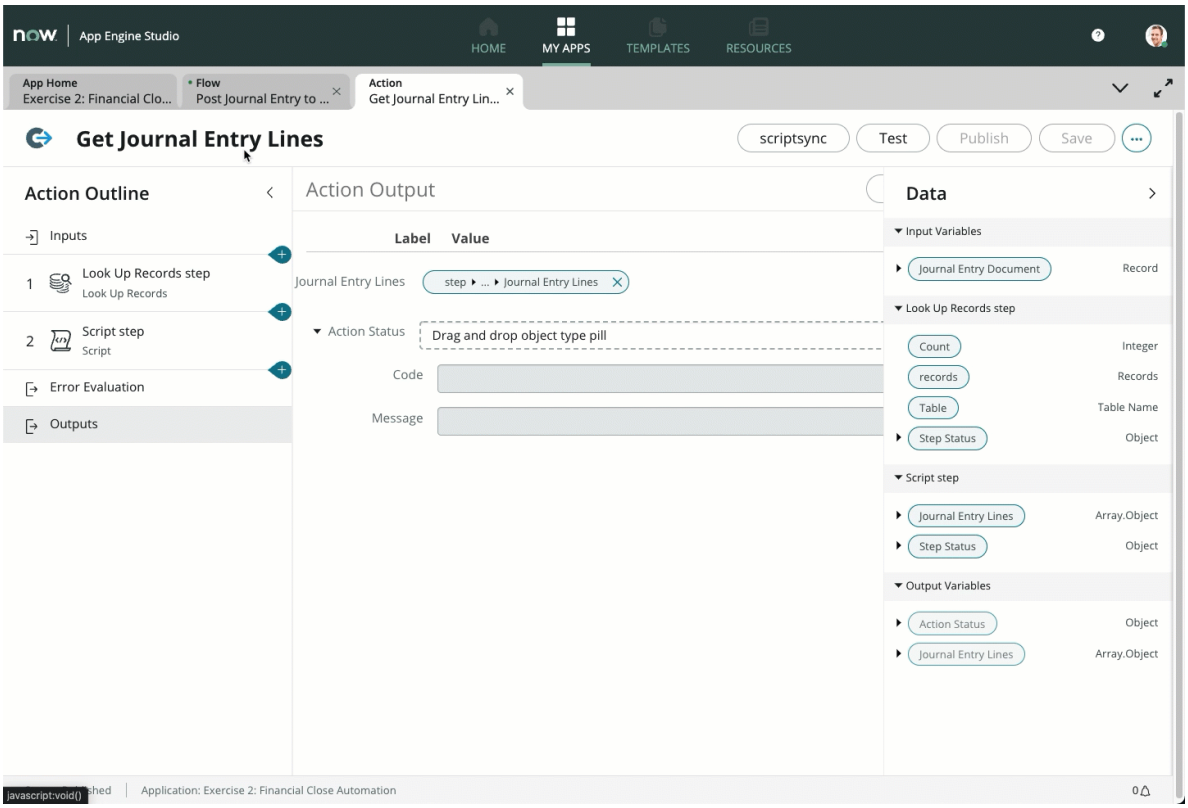
21. On the right of the action, open the Action by clicking on the **Open Action in Action Designer icon**

22. Work through the different action steps here to understand what is being done



> This action takes the input of the Journal Entry Document, then finds all related Journal Entry Lines associated with the Journal Entry Document and converts each record to an Array.Object

23. Navigate back to the Flow
24. Drag the **Journal Entry Document Record** data pill onto the **Journal Entry Document** field under *Get Journal Entry Lines*
25. Move this action before the **Post Journal Entry** action so that we have access to the **Journal Entry Lines Array.Object** to pass into the **Post Journal Entry** action
26. Click the **Post Journal Entry** action
27. Drag the **Journal Entry Lines Array.Object** onto **Journal Entry Lines**



28. Click **Done**

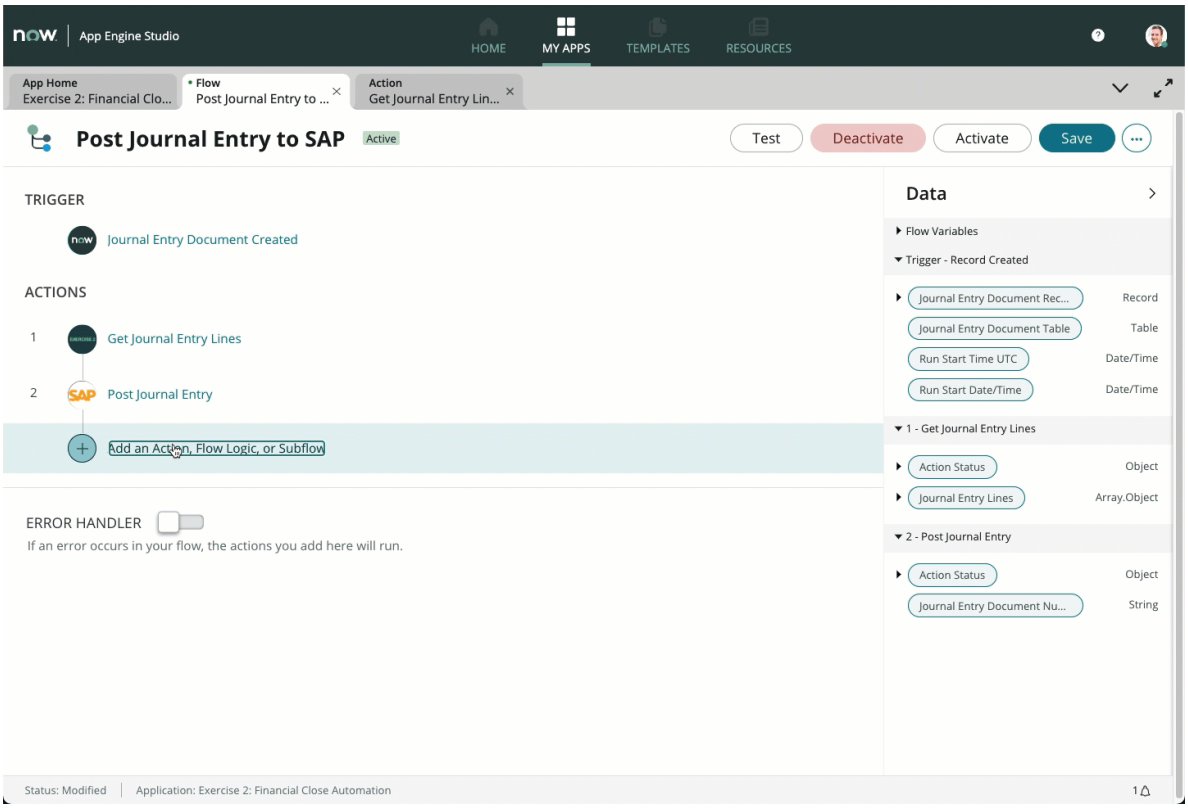
29. Click **Add an Action, Flow Logic, or Subflow**

30. Click **Action**, then under **ServiceNow Core**, search and select the **Update Record** action

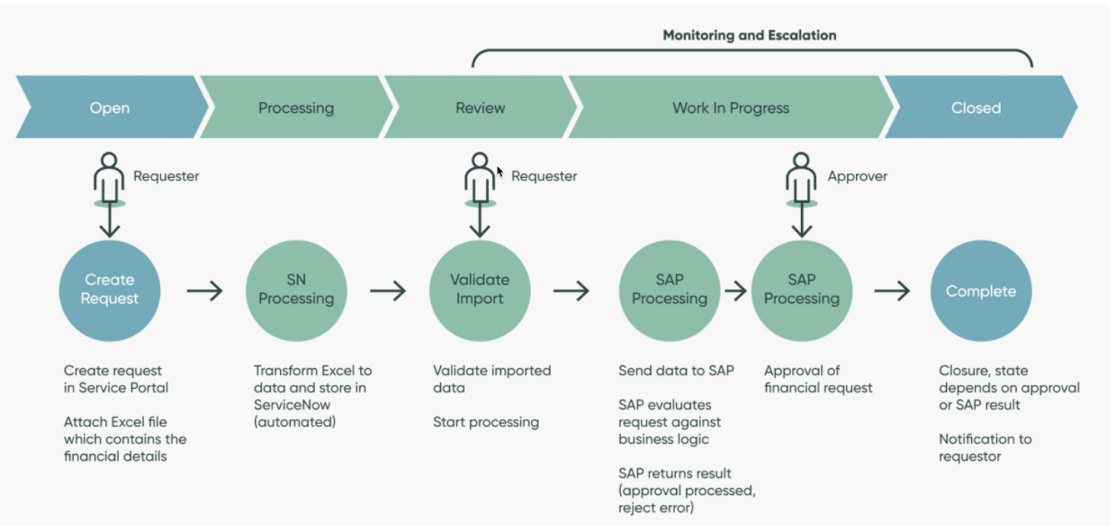
31. For *Record*, drag in the **Journal Entry Document Record** data pill

32. Click **Add field value**

33. Select **Document number**, and under *2 - Post Journal Entry*, drag in the **Journal Entry Document Number** data pill



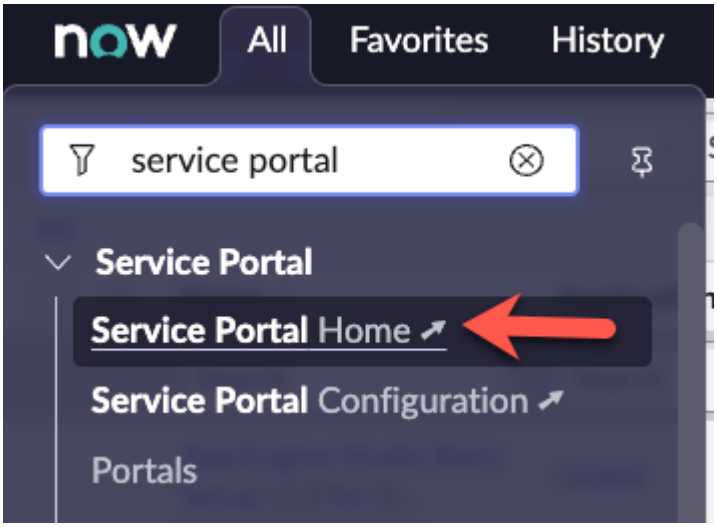
34. Click **Activate** on the top right of the screen

> **What are we missing here?**
> Once again we are skipping the approvals that has to happen before a Journal Entry actually gets posted into SAP in the interest of time. There are also often many validation rules for different Journal Entry types (Intercompany Posting, Asset Depreciation Posting, etc.) that we are not configuring in this exercise. Here is a common workflow that you will see across this Journal Entry posting process:
>
> 
>
>
> The method used (Excel template) is slightly different but the overall approach is the same.

Testing the experience and flow

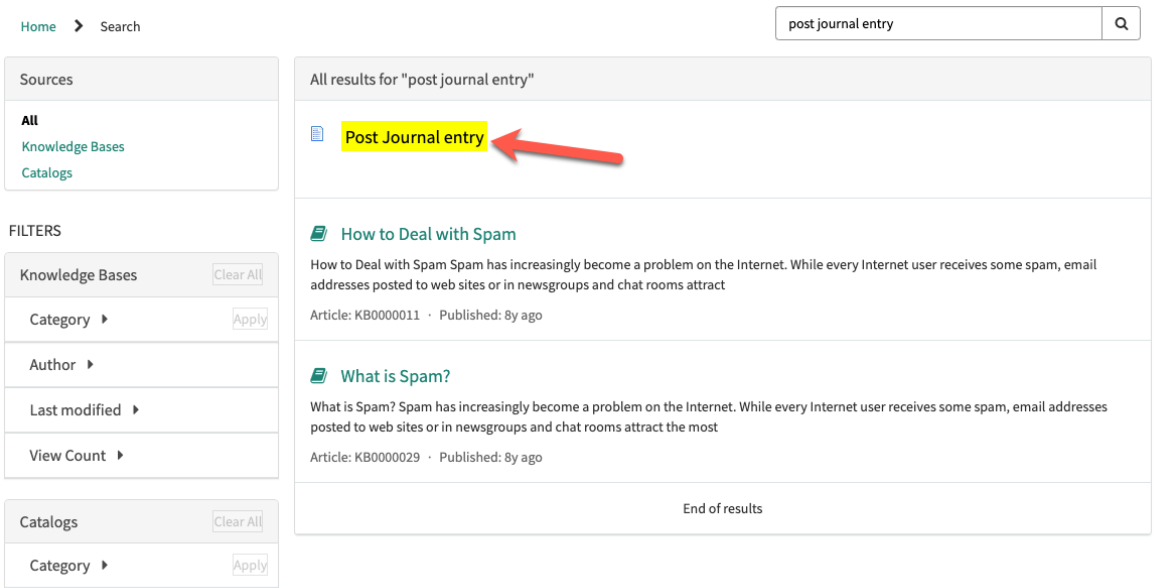
1. Navigate back to the ServiceNow Polaris UI, and search for **Service Portal** under *All*

2. Click **Service Portal Home**, and it will open in a new browser tab



3. Under the search bar, enter **post journal entry**, then hit search

4. Click **Post Journal Entry**



5. This was the record producer you created earlier.

6. Enter **1003** for Company code

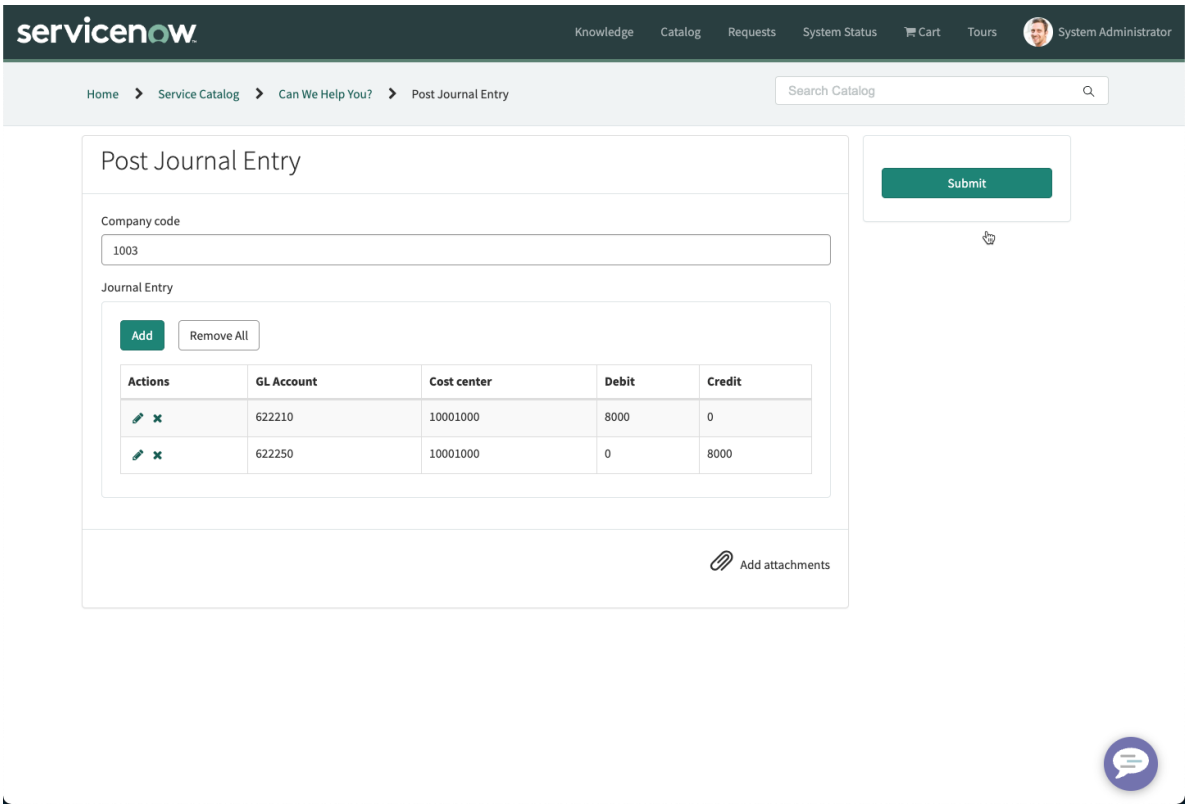
7. Click **Add**, then enter the following

Field	Entry
GL Account	622210
Cost center	10001000

Field	Entry
Debit	8000
Credit	0

8. Click **Add**
9. Click **Add** once again for a new Journal Entry
10. This time, enter these details

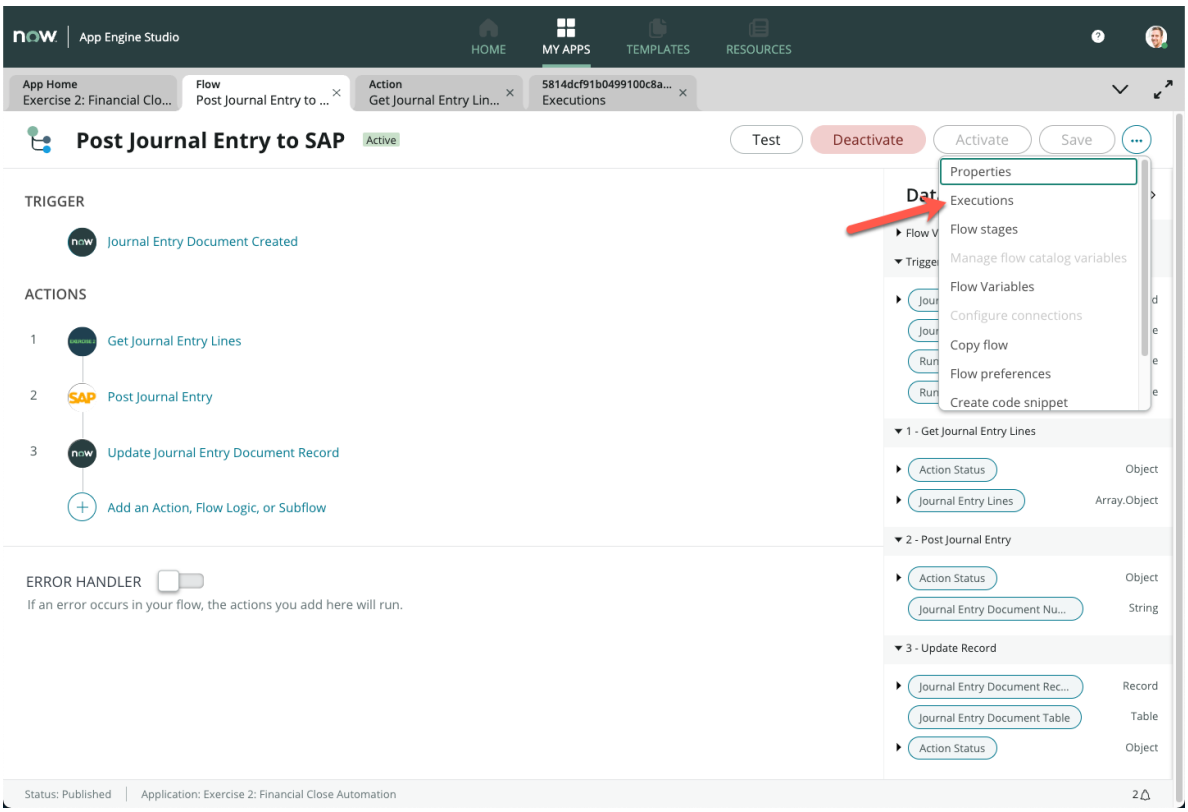
Field	Entry
GL Account	**622250**
Cost center	10001000
Debit	0
Credit	8000



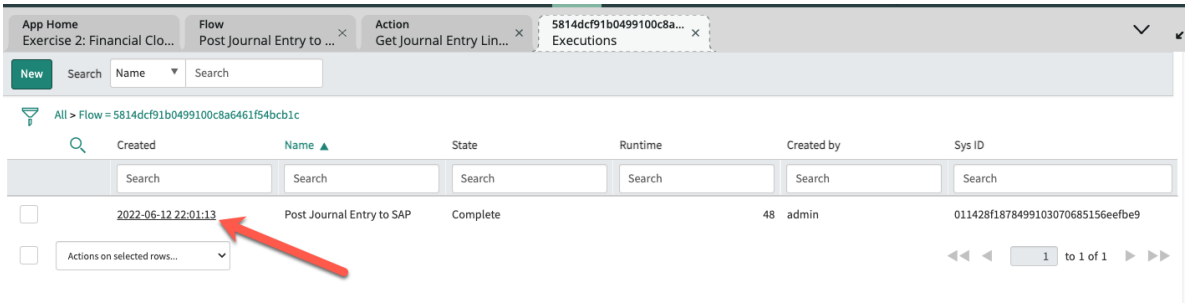
11. Click **Submit**

12. Go back to App Engine Studio, with the **Post Journal Entry to SAP** flow open

13. On the top right, click on **more** then click **Executions**



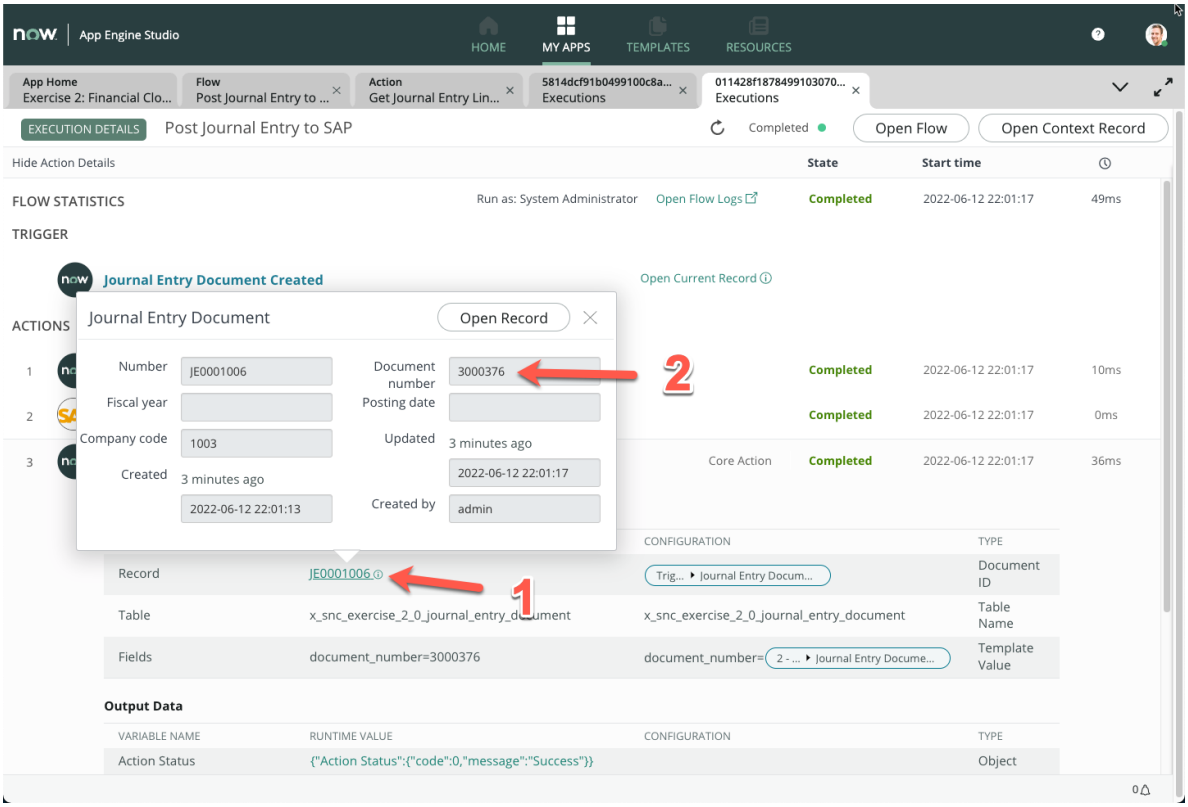
14. You should now see one execution record, click it



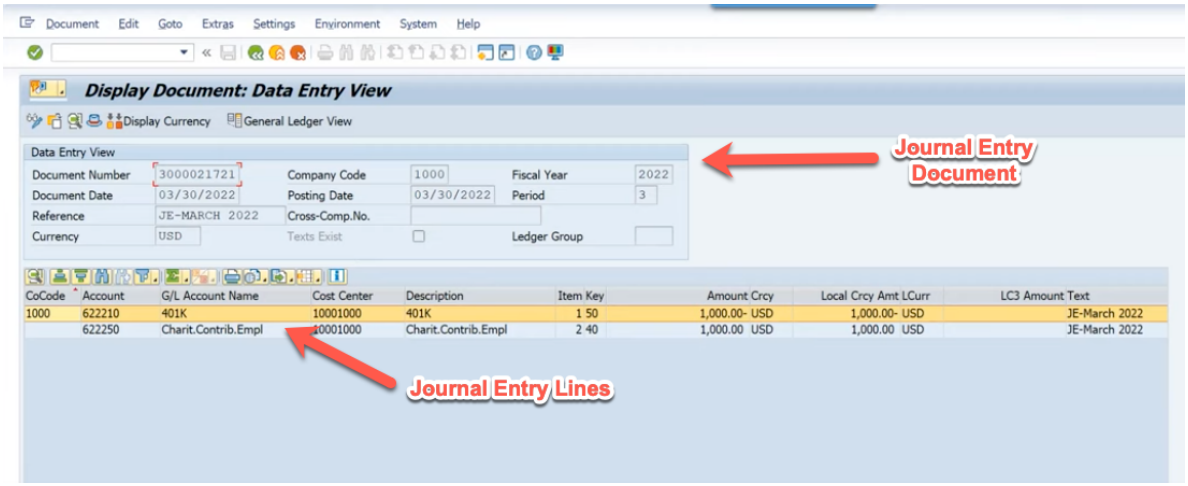
15. Confirm that the workflow was executed successfully, then click **Update Record** to expand it

16. Click the **Record** number, and confirm that the **Document number** field is now populated

17. This is the document number you can use to identify the Journal Entry Document in SAP



18. For your reference, the Journal Entry Document will show up as such in SAP



Summary

Congratulations on completing this lab! You've built 2 apps that will help Modernize SAP. One for automating approvals on any object from SAP, and another for assisting in the Financial Close process by giving users a better experience when posting Journal Entries. Just these alone will tremendously alleviate a great amount of manual effort that was previously done through SAP + multiple channels.

There are obviously so many more areas we can help with, from P2P to OTC, shifting customizations out of SAP is the number 1 priority for many customers, and ServiceNow Creator Workflows allows you to do that all on our industry leading Low-code Application Development Platform.