

shaoservicenow / sapmodernizationlab Public

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main ▾

...

sapmodernizationlab / guidebook.md



shaoservicenow 20 June



1 contributor

914 lines (499 sloc) | 35.8 KB

...

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 (PI2P) 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 referred 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 its 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

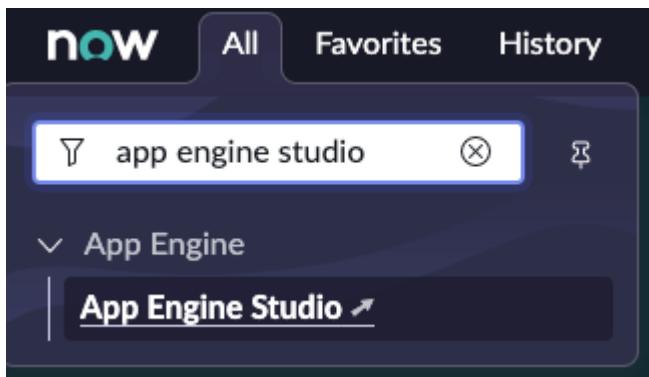
The sample that we will work with, but it usually gets a lot more complicated than this 

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.

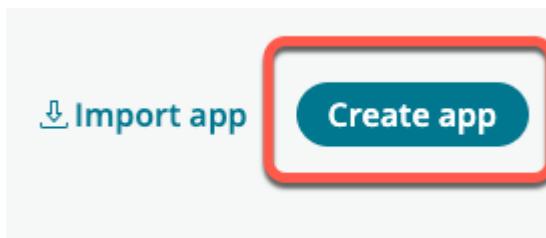
For easy reference, this is how the Standard Order (Sales Order) document looks like in SAP ECC. You navigate to this module by entering the T code VA01. The T Code (Transaction code) is a 4 digit shortcut key to access a requested transaction, sort of like carrying out an *incident.list* command in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

Let's start

- Under All, search and navigate to App Engine Studio



- Click Create app on the top right of the screen



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

CREATE APP

Let's get started on your new app.

Add a name and description that define the purpose of your app.
You can also add a thumbnail image.

Name *

Delegation of Authority

Description

Create an approval matrix for various SAP documents

Drag app logo
or browse to
upload

BMP, GIF, ICO, JPEG,
JPG, PNG, SVG

Cancel

Continue

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

The screenshot shows the SAP App Engine Studio interface with the 'Delegation of Authority' app selected. The top navigation bar includes links for HOME, MY APPS (which is highlighted), TEMPLATES, and RESOURCES. The left sidebar shows 'App Home' and 'Delegation of Authority'. The main content area is titled 'Delegation of Authority' and contains four sections: 'Data', 'Experience', 'Logic and automation', and 'Security'. A red arrow points to the '+ Add' button under the 'Data' section. The 'Data' section also includes a note: 'Store information in your app' and a button to 'Add a table or upload a spreadsheet'. The 'Experience' section has a note: 'Create interfaces for users to interact with the app' and a button to 'Add an interface'. The 'Logic and automation' section has a note: 'Add automated workflows to improve productivity' and a button to 'Add a process or third-party integration'. The 'Security' section includes a toggle switch, a user icon labeled 'admin', a role dropdown, and a note: 'Default admin role'. There are also 'See all (2)' and '...' buttons.

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**

ADD DATA**Which table do you want to use?**

We'll add the table's data to your app. Essentially, you are creating an extension with branching logic.

Table

task	Continue
Analytics Task analytics_task	
Execution Plan Task sc_cat_item_delivery_task	
Planned task planned_task	
Release Task rm_task	
Service Task service_task	
Task task	
Task Relationship task_rel_task	
Template 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**

now. | App Engine Studio

HOME MY APPS **TEMPLATES** RESOURCES

Source **Define** Summary

ADD DATA

Now, let's get more info about your new table

Define the properties of your new table.

Table label * ⓘ SAP Sales Order

Table name * ⓘ x_snc_1006_deleg_0_sap_sales_order

Make extensible ⓘ

Auto number ⓘ

Prefix * ⓘ SAPSO

Starting number * ⓘ 1000

Number of digits * ⓘ 7

Cancel Continue

10. Click **Continue**

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

The screenshot shows the SAP App Engine Studio interface in the 'Define' step. At the top, there are tabs: 'Source' (with a checkmark), 'Define' (highlighted with a blue border and a '2'), and 'Summary'. Below the tabs, the title 'ADD DATA' is followed by the heading 'Let's add permissions to your table.' A note below it says: 'Create roles and define how much control each role has of this table. Note: at least one role needs to have 'read' access for you to 'preview' the data in your table.' A dashed box labeled '+ Add a role' contains two rows of data:

Role Name	Description	All	Create	Read	Write	Delete
admin	Default admin role	<input checked="" type="checkbox"/>				
user	Default user role	<input checked="" type="checkbox"/>				

At the bottom right are 'Cancel' and 'Continue' buttons, and a toggle switch.

12. Click **Edit table**

13. Close the pop-up dialog

14. You should now be on the *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

The screenshot shows the 'Table Fields' configuration page for the 'SAP Sales Order' table. It lists 68 fields with columns for Column label, Column name, Type, Reference, Max length, Default value, Display, and Updated. Fields include comments_and_work_notes (Journal List), company (Reference to Company), contact_type (String), contract (Reference to Contract), document_number (String), document_type (String), amount (Decimal), sales_organization (String), and status (Choice with 5 Choices). A red box highlights the 'Add new field' button at the bottom left of the table.

Column label *	Column name *	Type *	Reference	Max length	Default value	Display	Updated
Comments and Work notes	comments_and_work_notes	Journal List		4,000			2022-03-23 16:12:29
Company	company	Reference	Company	32			2022-03-23 15:47:05
Configuration item	cmdb_ci	Reference	Configuration Item	32			2022-03-23 16:00:44
Contact type	contact_type	String		40			2022-03-23 15:47:05
Contract	contract	Reference	Contract	32			2022-03-23 16:25:39
Document number	document_number	String		40			
Document type	document_type	String		40			
Amount	amount	0.0 Decimal		40			
Sales organization	sales_organization	String		40			
Status	status	Choice 5 Choices		40			

Showing 1-20 of 68 1 2 3 4 → Records per page 20 ▾

16. Click Save

17. Click Form views on the top of the screen

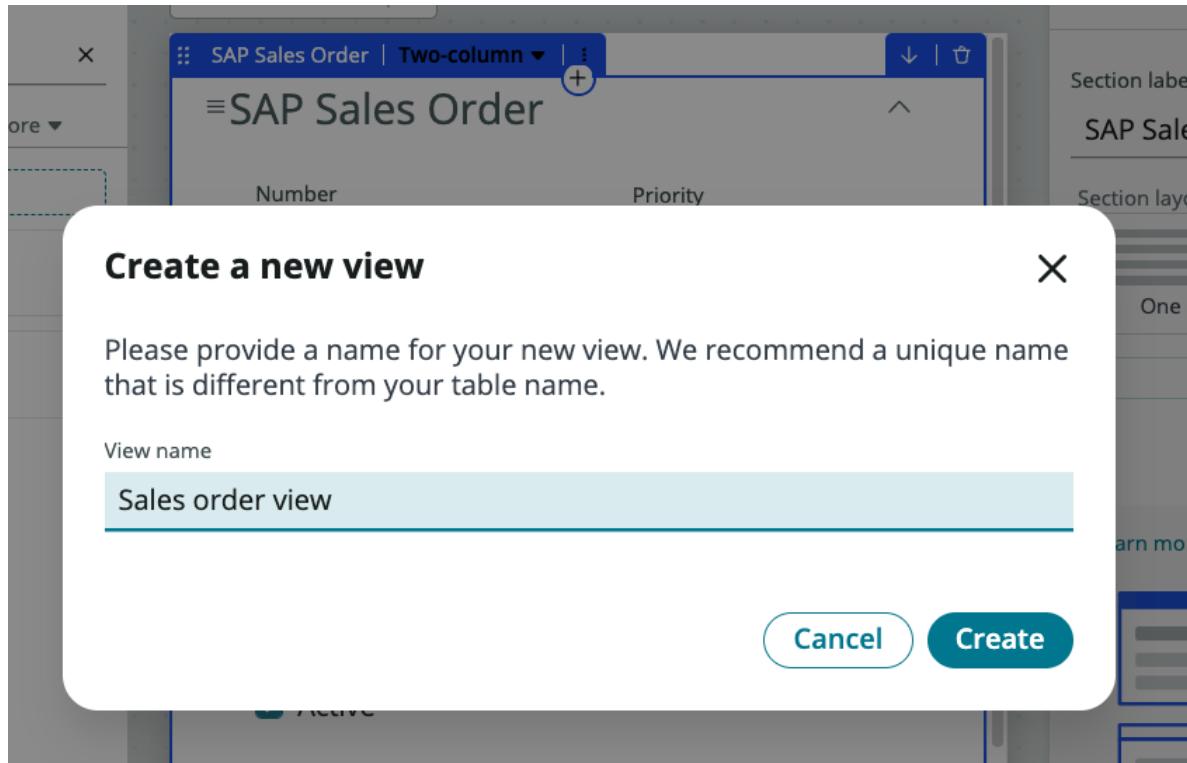
The screenshot shows the top navigation bar with tabs: App Home, Delegation of Authority, Data Table And Forms, SAP Sales Order, Table, Form views (which is highlighted with a red box), Policies and rules. Below the navigation is a 'Table Fields' section with a 'Form views' tab. The table fields listed are approval_set (Datetime), assigned_to (Reference), assignment_group (Reference), business_duration (Duration), and close_notes (String).

Column label *	Column name *	Type *
approval_set	approval_set	Datetime
Assigned to	assigned_to	Reference
Assignment group	assignment_group	Reference
Business duration	business_duration	Duration
Close notes	close_notes	String

18. Click the Default view dropdown

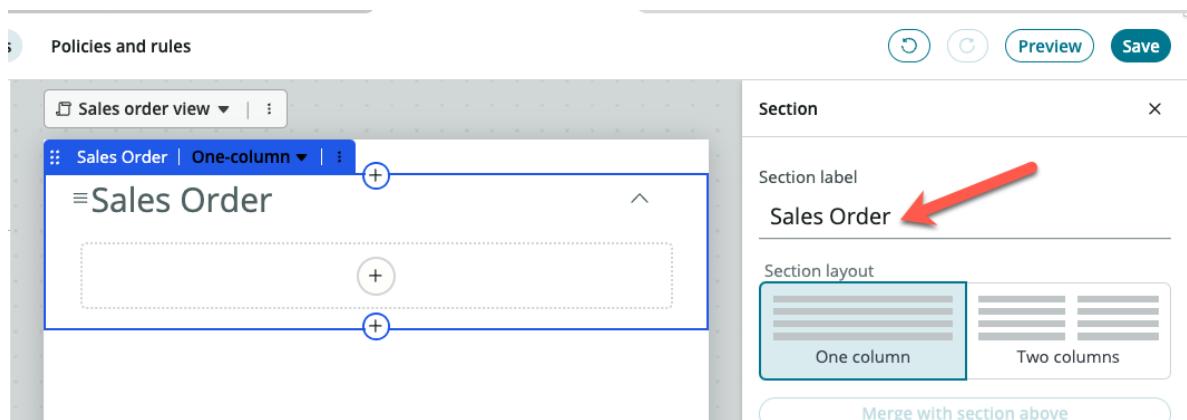
19. Click + Create

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.

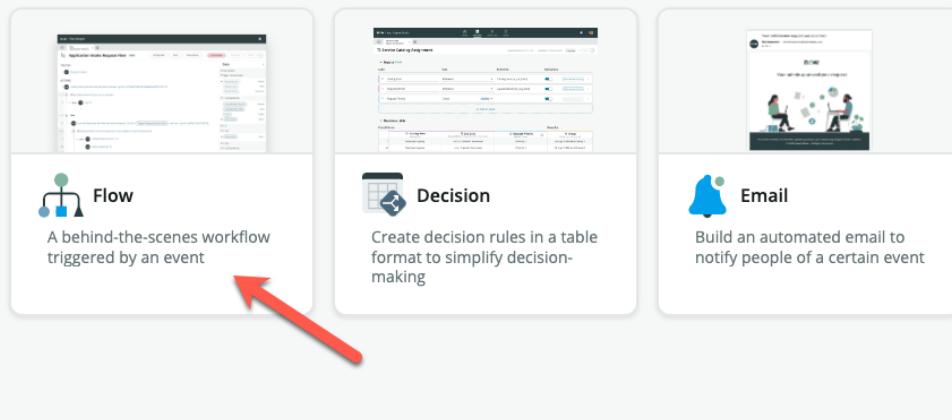
Get Sales Order data via the SAP ERP Spoke

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**

ADD LOGIC AND AUTOMATION

What do you want to add?

Automate your organization's workflows, like automated email notifications or an event-triggered flow, to improve efficiency.



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

The screenshot shows the App Engine Studio interface with the following details:

- Header:** now | App Engine Studio, HOME, MY APPS, TEMPLATES, RESOURCES, ? (Help), User icon.
- Top Bar:** App Home, Delegation of Authority, Data Table And Forms, Flow, Get Sales Orders fro... (inactive).
- Flow Title:** Get Sales Orders from SAP every hour (inactive).
- TRIGGER:** A button labeled "Add trigger".
- ACTIONS:** A button labeled "Add an Action, Flow Logic, or Subflow".
- ERROR HANDLER:** A toggle switch that is off. Below it, text says: "If an error occurs in your flow, the actions you add here will run."
- Data:** A section with a placeholder image of a house and water, and the text "No data yet". Below it, a note says: "Data will appear here as you add steps to your flow".
- Status Bar:** Status: Modified | Application: Delegation of Authority | 0Δ.

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

The screenshot shows the SAP Fiori Launchpad with a search bar at the top containing "SAP". Below the search bar, there are three tabs: "Action", "Flow Logic", and "Subflow". A modal dialog is open, listing various SAP spokes categorized into "INSTALLED SPOKES" and "NOT INSTALLED SPOKES".

INSTALLED SPOKES

- SAP ECC IDoc**
- SAP ECC RFC** (selected)
- 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.

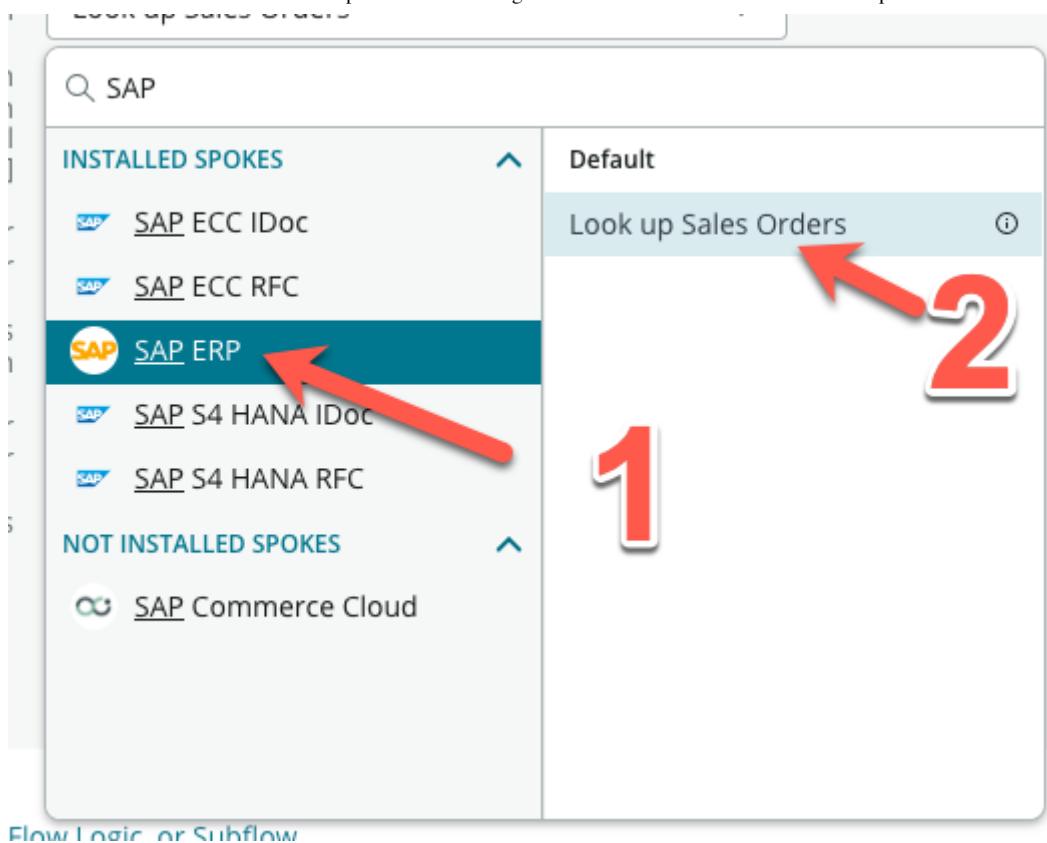
SAP S/4 HANA: This is the latest release of SAP ERP and it can run only on HANA database. With this release, SAP has simplified their core database architecture. This together with in-memory processing enables business to do complex business computation within minutes.

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**



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 accessibility.

Action	Run Start Time UTC	Date/Time
Look up Sales Orders		
* Connection Alias	SAP_S4HANA RFC	Object
Sales Document Type	Standard sales order (OR)	String
Sales Organization	-- None --	Decimal
Delivery Block	ServiceNow approval	String
Material		String
Customer		String
Delivery Date		String

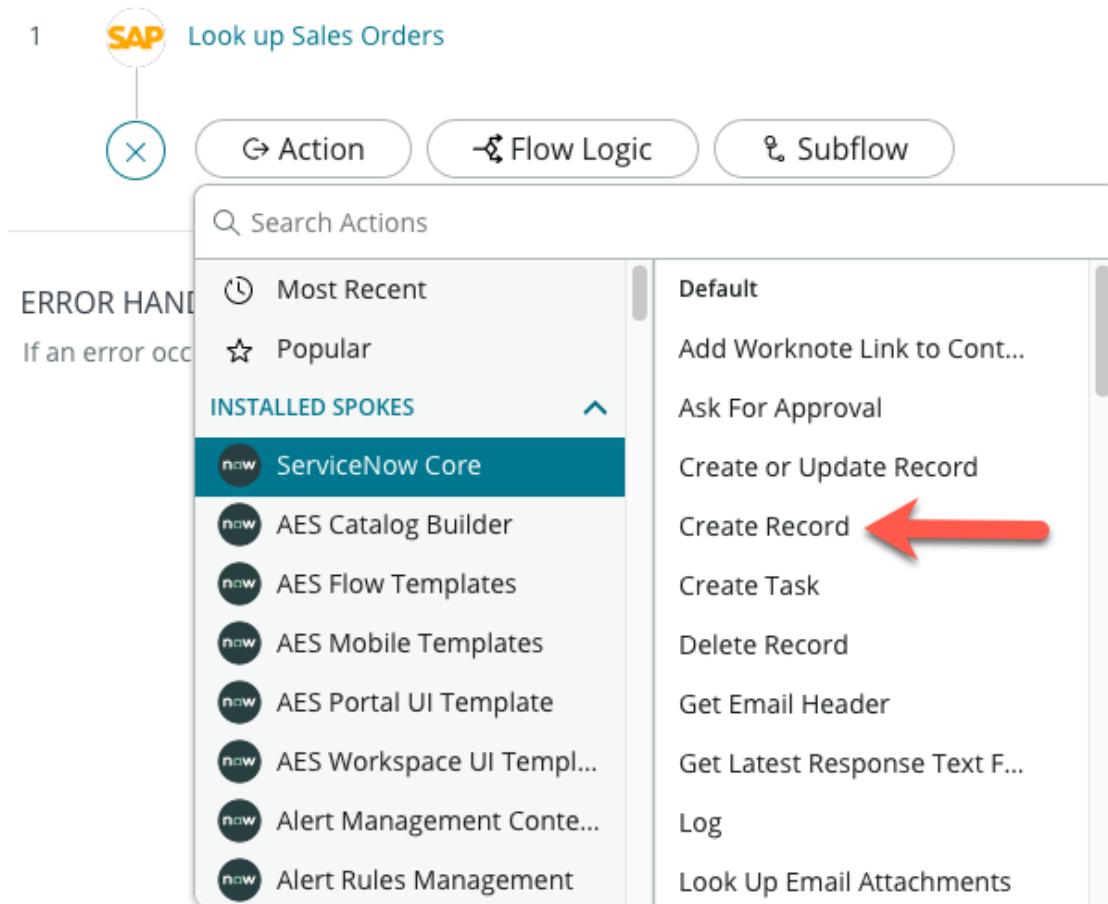
18. Notice how the **Delivery Block** field is populated with **ServiceNow approval**.

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**

The screenshot shows the SAP Flow Designer interface for creating a SAP Sales Order record. The main area displays the "Create SAP Sales Order Record" action. It includes fields for "Action" (set to "Create Record"), "Table" (set to "SAP Sales Order [x_snc_1006_deleg_0_sap_s...]"), and "Fields" (with a dropdown menu open). Below these fields are buttons for "Delete", "Cancel", and "Done". To the right, a sidebar shows the flow structure with steps like "Run Start Time UTC", "1 - Look up Sales Orders" (which further details "Action Status", "Document number", "Sales order amount", "Document type", and "Sales organization"), "2 - Create Record" (which details "SAP Sales Order Record", "SAP Sales Order Table", and "Action Status"), and "SAP Sales Order Record".

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

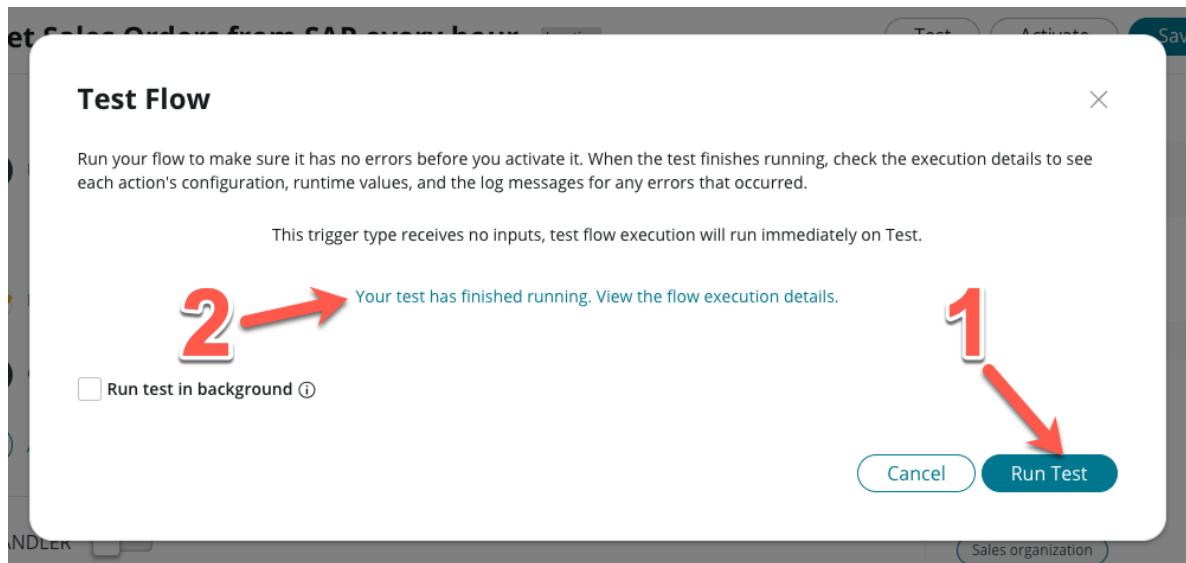
27. Your entire action should look like this

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

1. Click

2. Click

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)

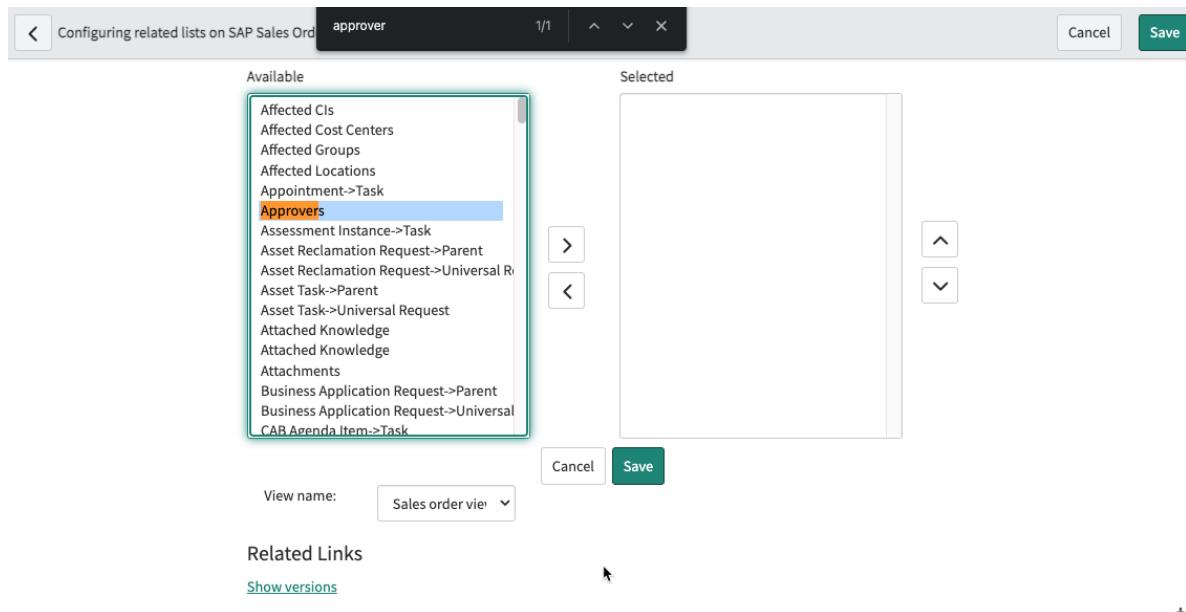
Number	SAPSO0001001
Status	New
Document number	0000013931
Document type	Standard sales order
Amount	50,400
Sales organization	Germany

Update Delete

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

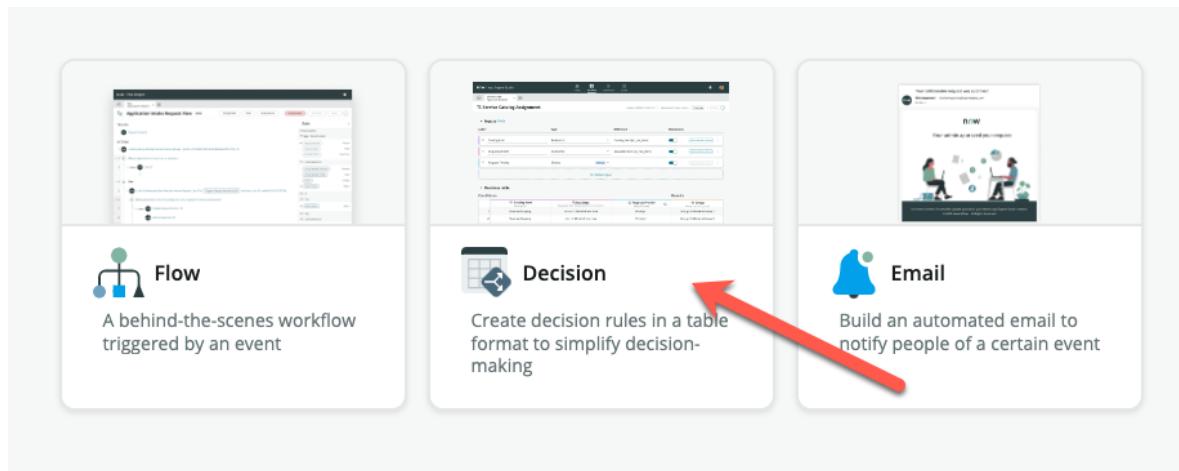
Congrats! You have built the table to store Sales Order data from SAP. Any approvals that you trigger later can also be seen on this view. Go back to the previous browser tab, but don't close anything yet.

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

ADD LOGIC AND AUTOMATION

Let's set up your decision table

After setting this up, you can build your table and add decision rules to define the logic.

Name * ①

Sales Order Approval Matrix

Accessible from ①

All application scopes

Cancel Continue

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

The screenshot shows the 'Sales Order Approval Matrix' configuration page. In the 'Inputs' section, there is a table with four columns: Label, Type, Reference, and Mandatory. A row for 'Sales order record' has 'Reference' selected as the type, pointing to 'SAP Sales Order [x_snc_1006_deleg_0_sap_sale]'. There is also a button to 'Add condition column'.

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

12. Click **Add result column**

The screenshot shows the 'Sales Order Approval Matrix' configuration page. In the 'Decision table' section, under the 'results' column, there is a large rectangular area with rounded corners. In the bottom right corner of this area, a cursor is hovering over the text '⊕ 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**

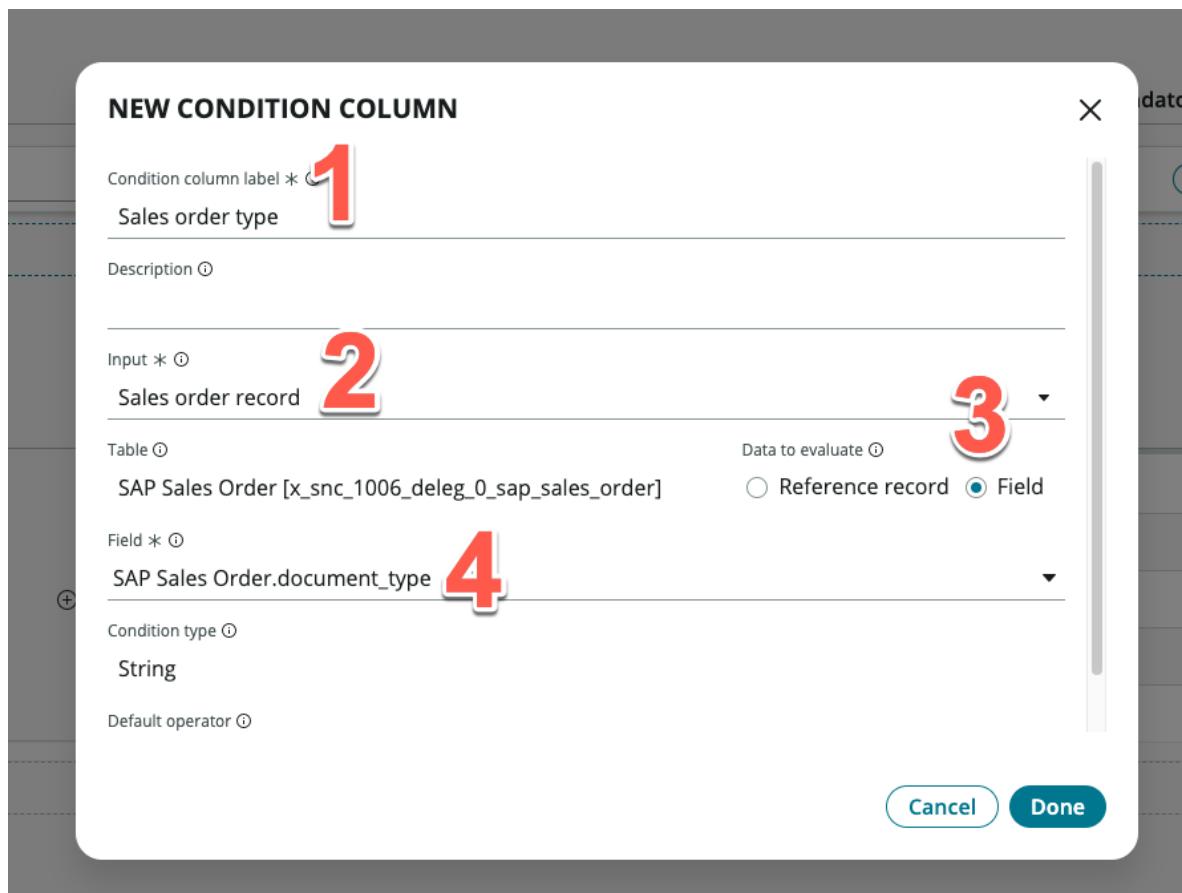
17. Under the **Approval group** column, add these selections as per our matrix at the start of this section: Finance managers, Finance controllers, CFO, CFO (Ensure that CFO is created twice as we have two different scenarios that require CFO approval)

The screenshot shows the App Engine Studio interface for a "Sales Order Approval Matrix" application. At the top, there are tabs for "App Home", "Data Table And Forms", "Flow", "Decision Table", and "Resources". The "Decision Table" tab is active. Below the tabs, the application title is "Sales Order Approval Matrix". The "Inputs" section shows a single input labeled "Sales order record" of type "Reference" pointing to "SAP Sales Order [x_snc_1006_deleg_0_sap_sales_order]". The "Decision table" section shows a condition column with one row containing a value "1" and a result column with a row labeled "Approval group" pointing to "Group [sys_user_group]". A pop-up window titled "Add condition column" is open, showing fields for "Condition column label" (labeled 1), "Input" (labeled 2), "Table" (labeled 3), "Field" (labeled 4), "Condition type" (String), and "Default operator". The "Properties" bar at the bottom indicates the application was created on 2022-06-10 at 07:40:58.

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

The screenshot shows a table titled "results". It has two columns: "Sales order type" and "Approval group". The "Sales order type" column contains a single row: "Sales order record (SAP Sales Order ▶ Document type)". The "Approval group" column contains four rows: "Finance managers", "Finance controllers", "CFO", and "CFO". Below the table, there is a button labeled "+ Add new decision row" and a link "Show more". A red arrow points from the text above to the plus sign (+) icon located between the two columns.

23. Click Add condition column

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

NEW CONDITION COLUMN

Condition column label * ⓘ
Sales order amount

Description ⓘ

Input * ⓘ
Sales order record

Table ⓘ
SAP Sales Order [x_snc_1006_deleg_0_sap_sales_order]

Data to evaluate ⓘ
 Reference record Field

Field * ⓘ
SAP Sales Order.amount

Condition type ⓘ
Decimal

Default operator ⓘ

Cancel Done

25. Click Done

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

▼ Decision table

Conditions

The screenshot shows a decision table with a single row selected. The condition is "Sales order amount" with the value "less than 5000". A dropdown menu is open, showing "less than" and "Value" as options. The value field contains "5000".

		0.0 Sales order amount Sales order record (SAP Sales Order ▶ ...)	Sales order reco
●	1	less than 5000	
●	2	less than	
●	3	Value	
●	4	5000	

+ Add new decision row

Showing 1-4 of 4

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

The screenshot shows a decision table with a single row selected. The condition is "Sales order type" with the value "Standard sales order". A dropdown menu is open, showing "is" and "Value" as options. The value field contains "Standard sales order".

		= Sales order type Sales order record (SAP Sales Order ▶ Document type)	+ Add new decision row
		Standard sales order	
		is	
		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

The screenshot shows the App Engine Studio interface with the title "Sales Order Approval Matrix". The top navigation bar includes links for HOME, MY APPS, TEMPLATES, and RESOURCES. Below the navigation is a toolbar with tabs: App Home, Delegation of Authority, Data Table And Forms, SAP Sales Order, Flow, Get Sales Orders fro..., 27f88b742bc455106591..., Executions, Decision Table, and Sales Order Approva... . The main content area displays the "Inputs" section with a table for a "Sales order record" input, which is a reference to a SAP Sales Order. It also shows the "Decision table" section, which contains a table mapping sales order amounts to approval groups. The table has columns for "Conditions" (Min and Max values) and "results" (Approval group). The rows are:

	Conditions		results
	Min	Max	Approval group
1	less than 5000		Standard sales order Finance managers
2	5000	50000	Standard sales order Finance controllers
3	greater than 50000		Standard sales order CFO
4			Rush order CFO

At the bottom of the decision table section, there are buttons for "+ Add new decision row" and "Show more".

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**

The screenshot shows the 'Sales Order Approval Flow' configuration page. At the top, there is a header with a blue icon, the title 'Sales Order Approval Flow', and a status indicator 'Inactive'. Below the header, there is a section titled 'TRIGGER' containing a configuration for 'SAP Sales Order Created'. The trigger is set to 'Created' and the table is 'SAP Sales Order [x_snc_1006_deleg_0_sap_s...]'.

9. Click Done

10. Click Add an Action, Flow Logic, or Subflow

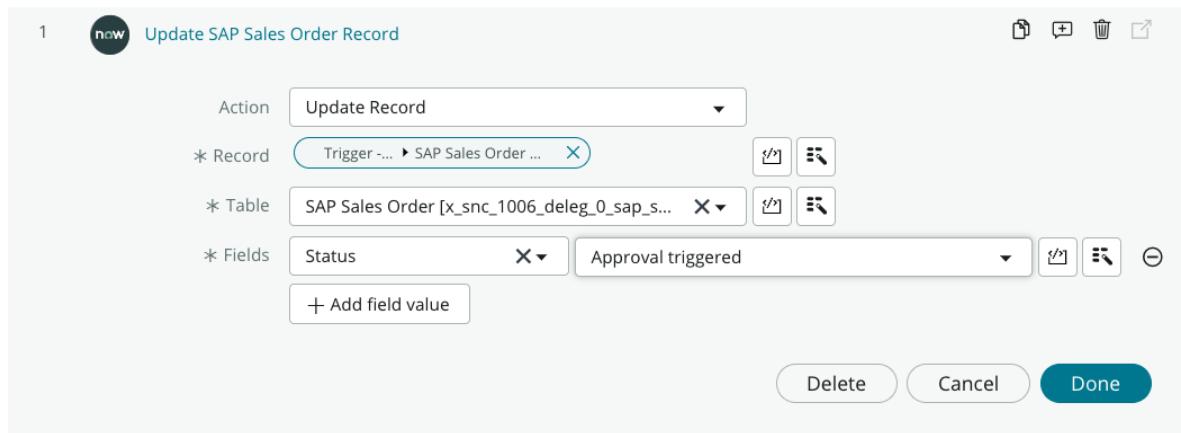
11. Click Action, then select Update Record under ServiceNow Core

The screenshot shows the 'Action' selection dialog. At the top, there are three tabs: 'Action' (selected), 'Flow Logic', and 'Subflow'. Below the tabs is a search bar labeled 'Search Actions'. A sidebar on the left lists 'Most Recent' and 'Popular' actions, and a section for 'INSTALLED SPOKES' with 'ServiceNow Core' selected. The main pane lists various actions: 'Look Up Records', 'Send Email', 'Send Notification', 'Send SMS', 'Update Multiple Records', 'Update Record' (highlighted with a red arrow), 'Wait For Condition', 'Attachments', 'Copy Attachment', 'Delete Attachment', and 'Get Attachments on Record'.

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

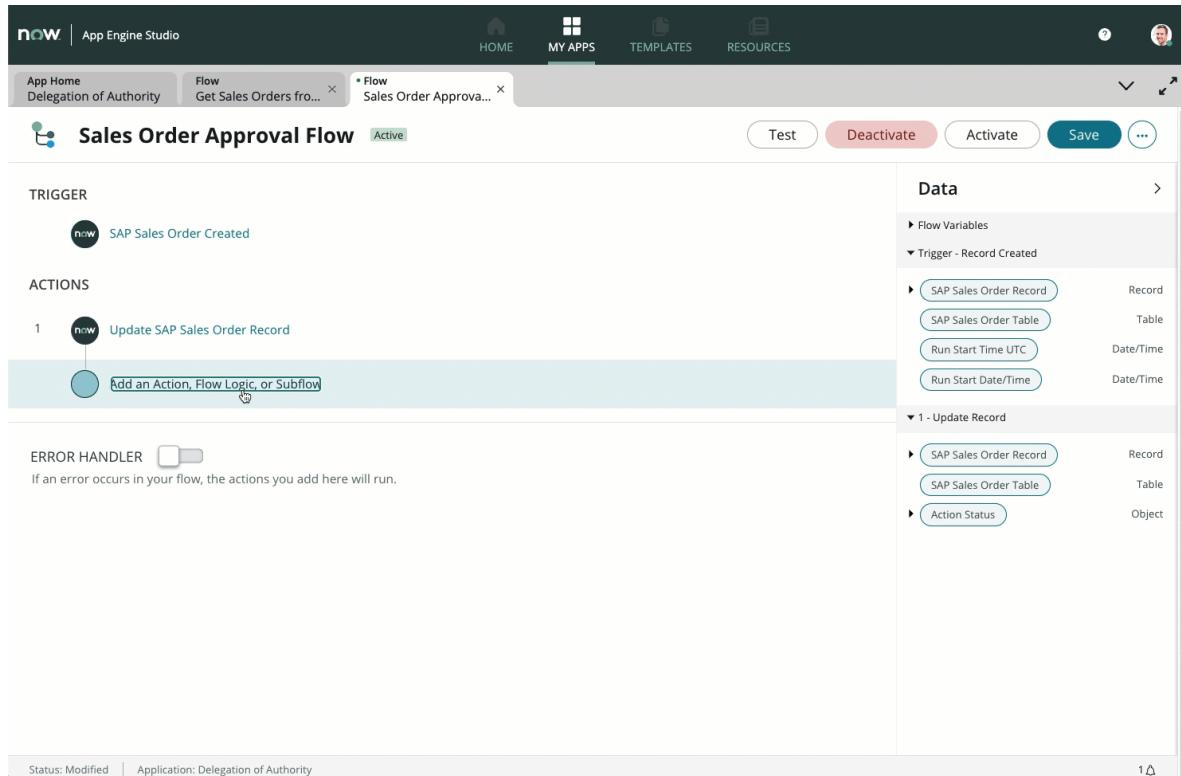
16. Click Add an Action, Flow Logic, or Subflow

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

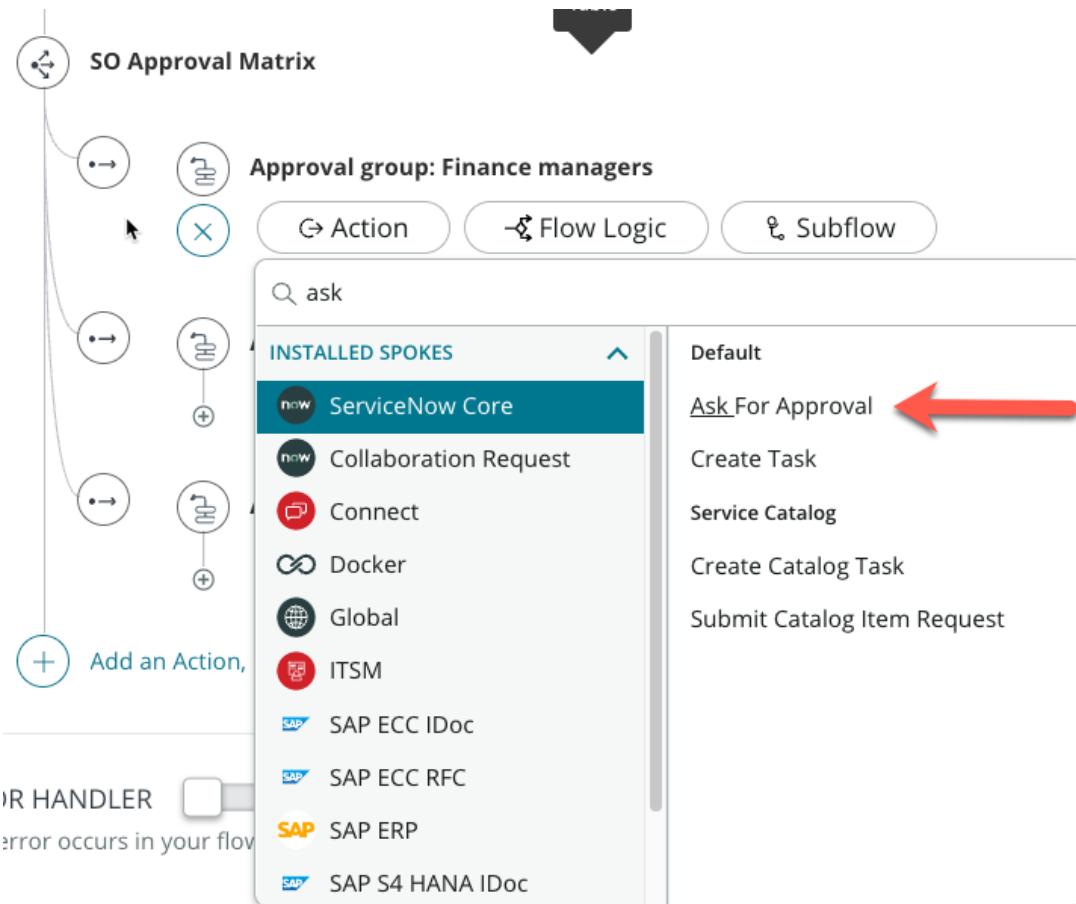
22. You should now see three branches appear for each *Group* we selected as approvers earlier

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

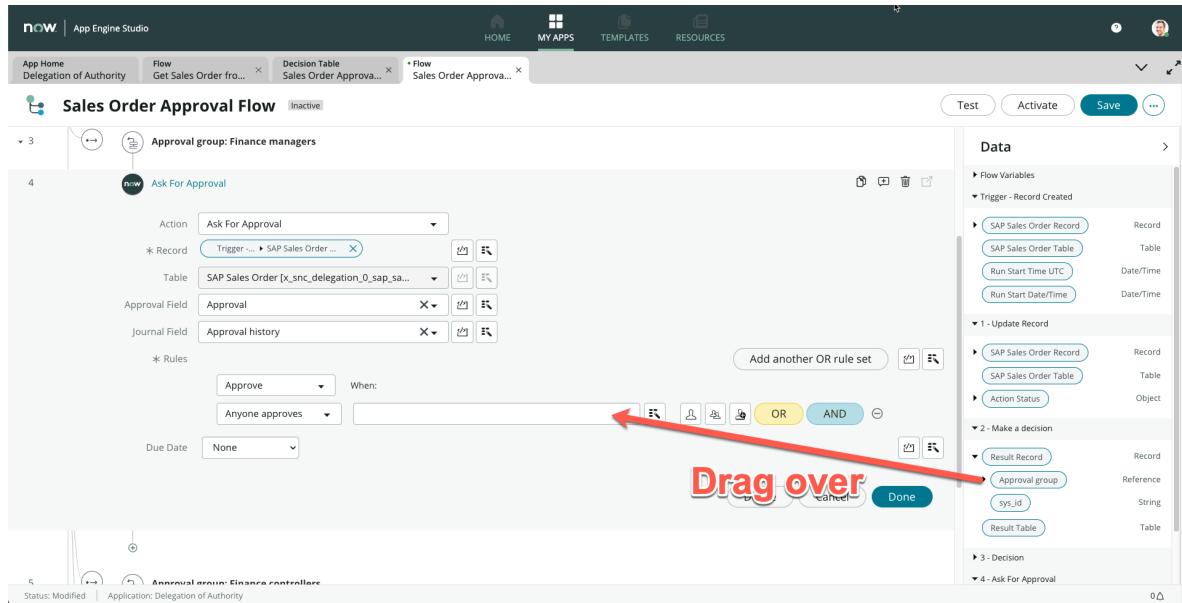


26. Drag **SAP Sales Order Record** from **Trigger - Record Created** within the data column on the right, onto the **Record** field

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)

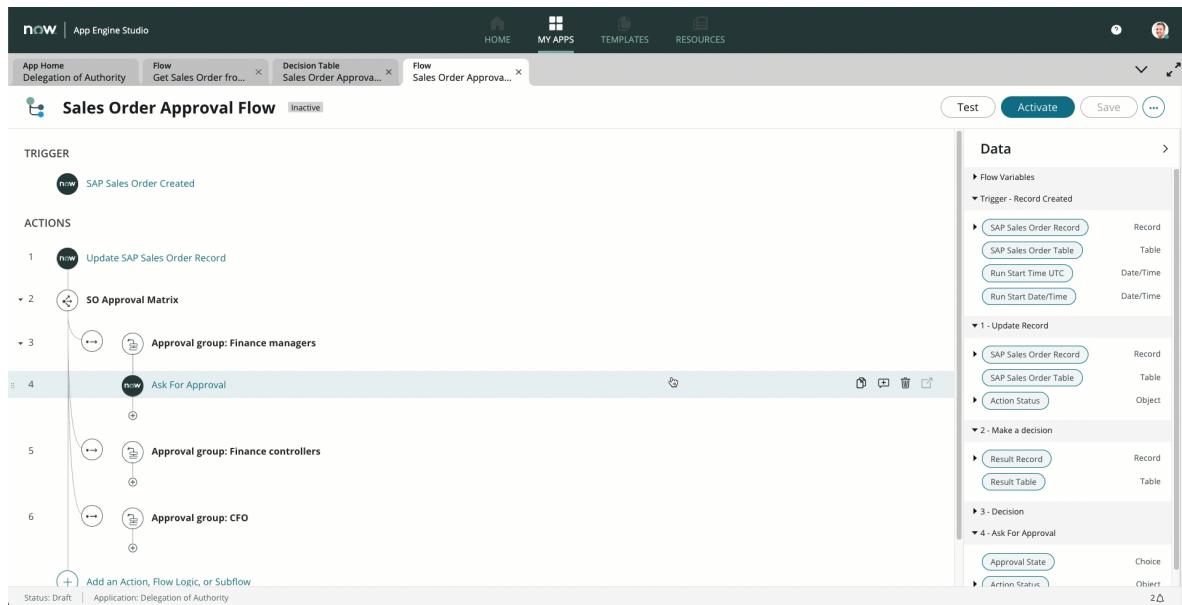
29. Drag over **Approval group** reference from **2 - Make a decision > Result Record** onto the empty field



30. Click Done

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

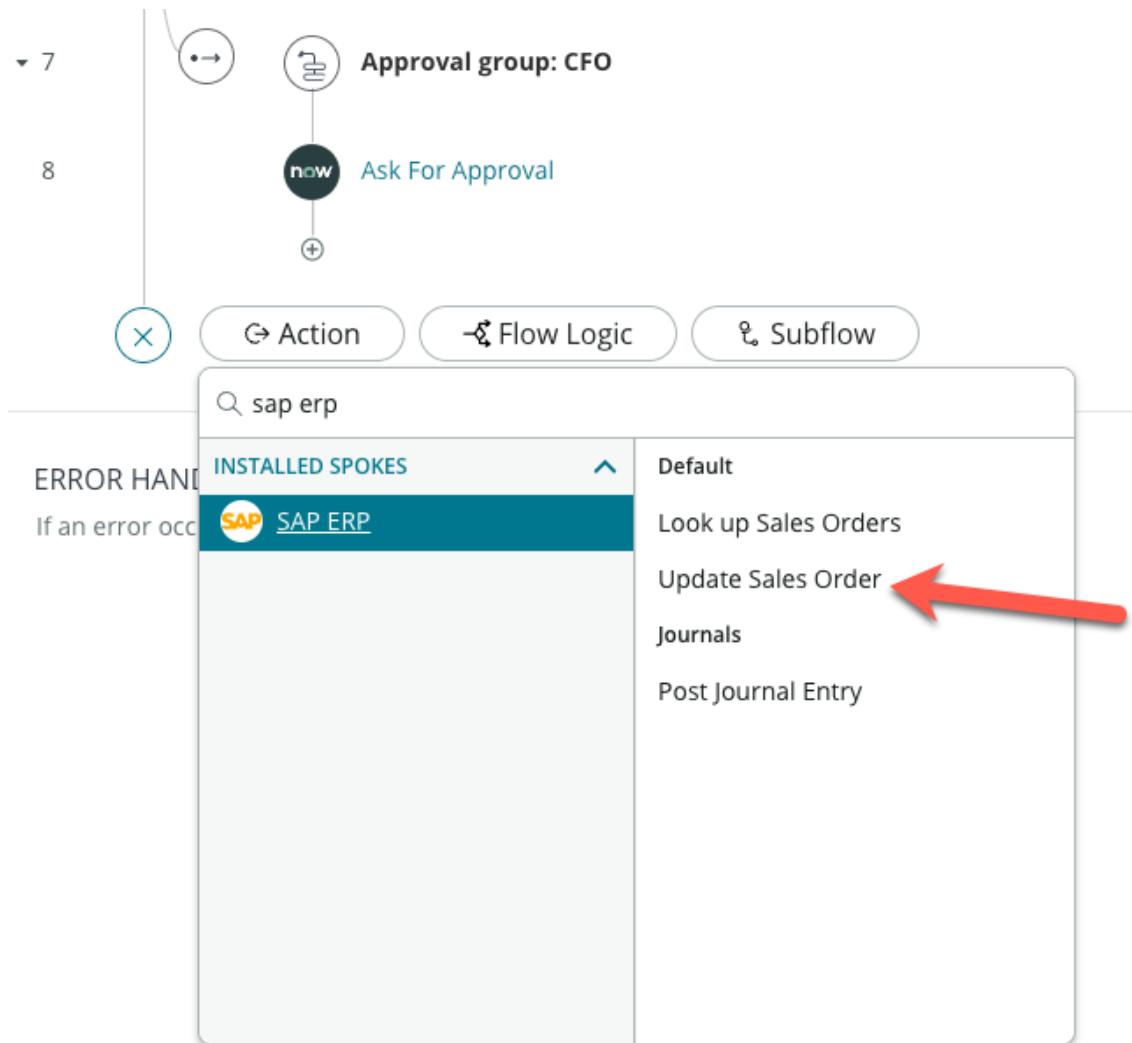
32. Drag each of the 2 duplicated actions under each of the other branches



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**

The screenshot shows the App Engine Studio interface for creating a flow. On the left, a flow diagram with nodes like 'Ask For Approval' and 'Update Sales Order'. The 'Update Sales Order' node has several configuration fields: 'Action' (set to 'Update Sales Order'), 'Connection Alias' (set to 'SAP_S4HANA RFC'), 'Sales Document' (highlighted with a red arrow), 'Delivery Block' (highlighted with a red arrow and set to 'None'), 'Sales Document Type' (set to 'Standard sales order (OR)'), 'Sales Organization' (set to '-- None --'), 'Material' (empty), 'Customer' (empty), and 'Delivery D...' (empty). On the right, a 'Data' panel lists various SAP objects with their data types. Below the flow diagram, there's an 'ERROR HANDLER' section and a status message 'Status: Modified | Application: Delegation of Authority'.

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.

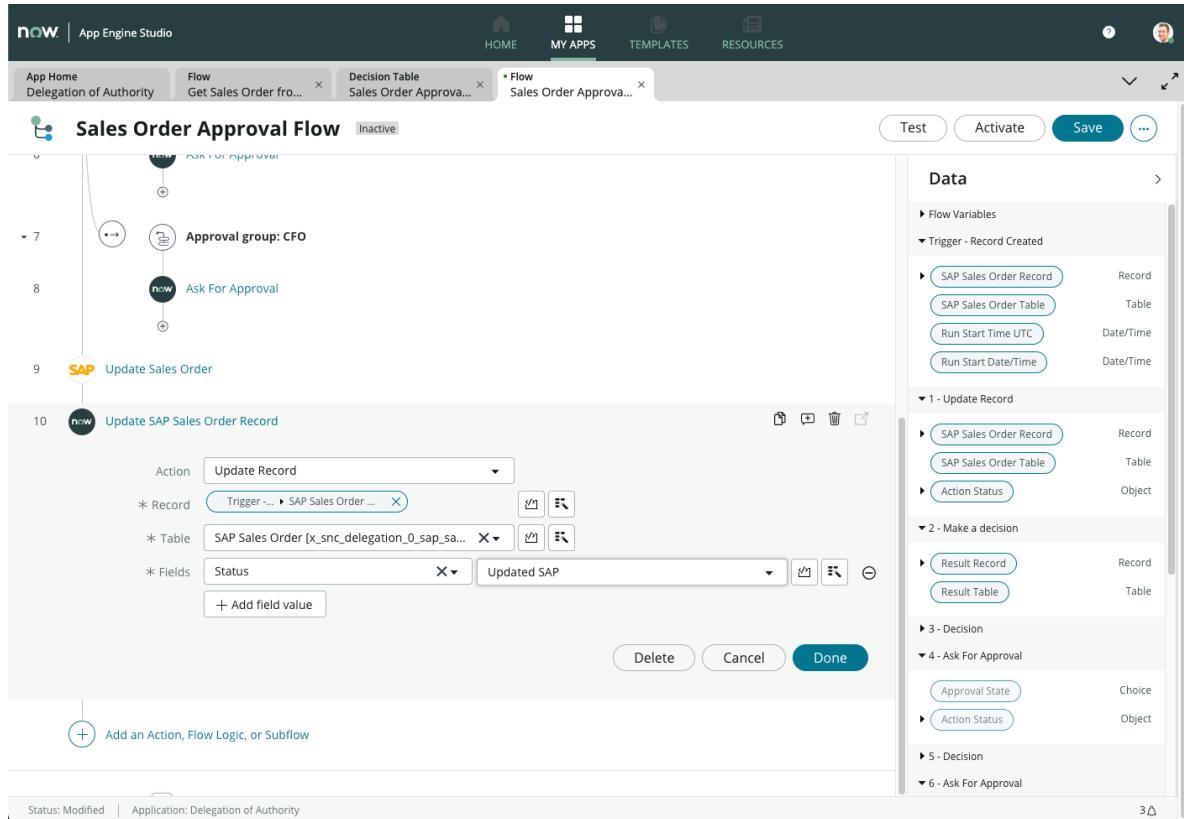
The screenshot shows the SAP ECC 'Change Demo Sales Order 20158: Overview' screen. At the top, there are tabs for Sales, Item overview, Item detail, Ordering party, Procurement, Shipping, and Reason for rejection. The Sales tab is selected. In the main area, there are fields for Demo Sales Order (20158), Net value (100,00 EUR), Sold-To Party (5000 DEMO Customer / D- Frankfurt), Ship-To Party (5000 DEMO Customer / D- Frankfurt), and PO Number. Below these, there are sections for Delivery, Pricing, and Incoterms. Under Delivery, there is a table for All items. The 'Delivery block' field is highlighted with a red box and contains the value '07 Change in quantity'. Other visible values include 'Total Weight' (14 KG) and 'Volume' (0,000). The bottom of the screen shows various SAP navigation icons.

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**

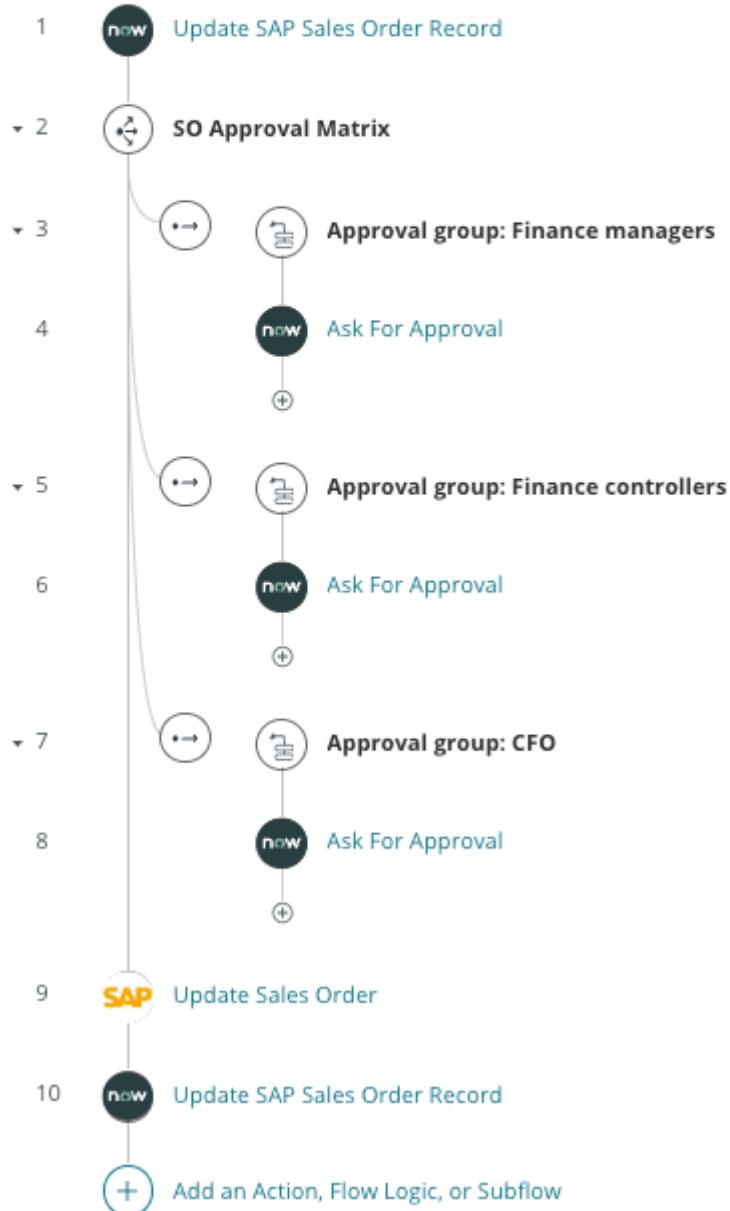


43. Click Done

44. Your entire flow should now look like this

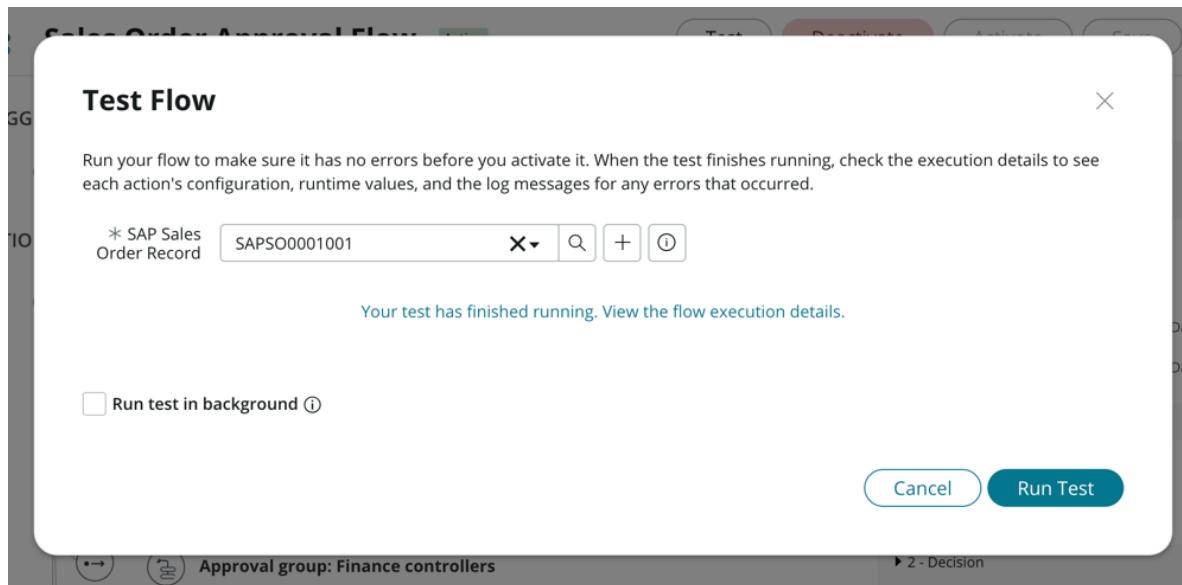


Sales Order Approval Flow

Inactive
TRIGGER**ACTIONS**

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

Step	Action	Type	Status	Start time	Duration
1	now Update Record	Core Action	Completed	2022-06-20 04:25:31	119ms
2	now SO Approval Matrix	Flow Logic	Waiting	2022-06-20 04:25:31	1677ms
3	now Approval group: Finance managers	Flow Logic	Evaluated - False	2022-06-20 04:25:32	3ms
4	now Ask For Approval	Core Action	Not Run		0ms
5	now Approval group: Finance controllers	Flow Logic	Evaluated - False	2022-06-20 04:25:33	0ms
6	now Ask For Approval	Core Action	Not Run		0ms
7	now Approval group: CFO	Flow Logic	Evaluated - True	2022-06-20 04:25:32	1327ms
8	now Ask For Approval	Core Action	Waiting	2022-06-20 04:25:32	1327ms
9	SAP Update Sales Order	Core Action	Not Run		0ms
10	now Update Record	Core Action	Not Run		0ms

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

Number: SAPSO0001001

Status: Approval triggered

Document number: 0000013931

Document type: Standard sales order

Amount: 50,400

Sales organization: Germany

Approver list:

- Fred Luddy (Approved)
- System Administrator (Pending Approval)

Right click on “Requested”, Then Approve

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

Test Run - Completed

Open Flow

Open Context Record

Refresh Flow

EXECUTION DETAILS: Sales Order Approval Flow

FLOW STATISTICS: Run as: System Administrator

TRIGGER: SAP Sales Order Created

ACTIONS:

- 1 now Update Record
- 2 SO Approval Matrix
- 3 Approval group: Finance managers
- 4 now Ask For Approval
- 5 Approval group: Finance controllers
- 6 now Ask For Approval
- 7 Approval group: CFO
- 8 now Ask For Approval
- 9 SAP Update Sales Order
- 10 now Update Record

Core Action: Completed (2022-06-20 04:25:31, 119ms)

Flow Logic: Completed (2022-06-20 04:25:31, 879ms)

Flow Logic: Evaluated - False (2022-06-20 04:25:32, 0ms)

Core Action: Not Run (2022-06-20 04:25:33, 0ms)

Flow Logic: Evaluated - False (2022-06-20 04:25:33, 0ms)

Core Action: Not Run (2022-06-20 04:25:32, 879ms)

Core Action: Completed (2022-06-20 04:25:32, 879ms)

Core Action: Completed (2022-06-20 05:00:14, 1ms)

Core Action: Completed (2022-06-20 05:00:14, 55ms)

ERROR HANDLER: 0

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 capabilities 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

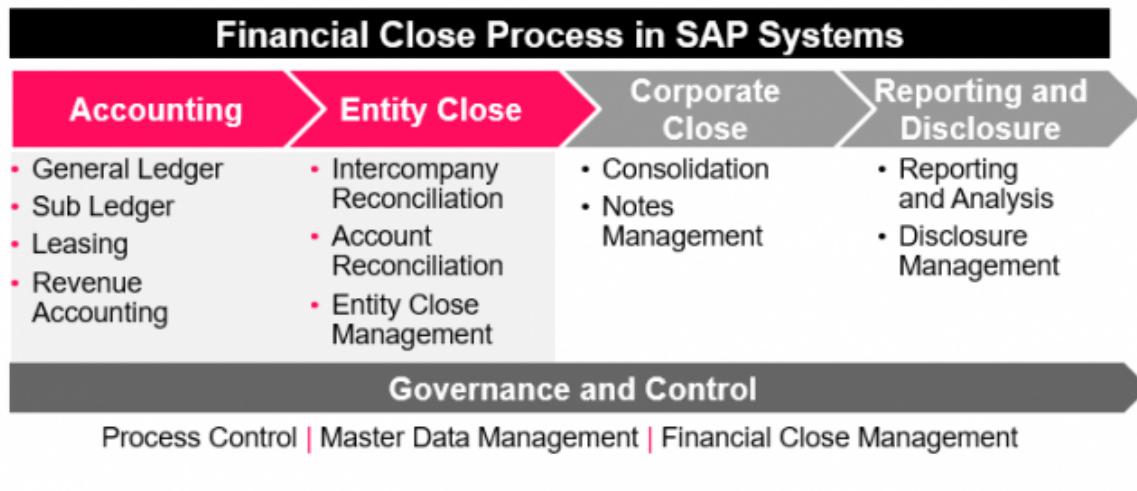
The financial close is a critical process that happens regularly (Monthly, Quarterly, Yearly) in a business with the general goal of producing financial reports representative of the company's true financial position. This is a stressful exercise and there is typically a checklist of activities that the finance team has to undertake, for example: Perform inventory count, reconcile 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 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 activities, 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 see that posting journal entries in the general ledger is among the first tasks in the financial close process. Not only finance users, but sometimes business users as well have to post hundreds or thousands of manual journal entries in each period!



Since you've already gone through the exercise 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**

App Engine Studio

HOME MY APPS TEMPLATES RESOURCES

Hi, System

How do you want to get started?

Quick start

- Add a table
- Add an experience
- Add an automated flow
- Browse templates
- Learn the tools

My recent apps

See all of my apps

EXERCISE 2	Delegation of Authority	SAP ERP	DemoHub PA Data Genera...
Exercise 2: Financial Close ... Last opened 2022-06-12 04:17:19	Delegation of Authority Last opened 2022-06-10 22:43:30	SAP ERP Last opened 2022-06-10 22:38:07	DemoHub PA Data Genera...

Templates

See all templates

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

Document Edit Goto Extras Settings Environment System Help

Display Document: Data Entry View

Display Currency General Ledger View

Data Entry View

Document Number	Company Code	Fiscal Year
300021721	1000	2022
Document Date	Posting Date	Period
03/30/2022	03/30/2022	3
Reference	Cross-Comp.No.	
JE-MARCH 2022		
Currency	Texts Exist	Ledger Group
USD	<input type="checkbox"/>	

Journal Entry Document

CoCode	Account	G/L Account Name	Cost Center	Description	Item Key	Amount Ctry	Local Ctry Amt LCurr	LC3 Amount Text
1000	622210	401K	10001000	401K	1 50	1,000.00- USD	1,000.00- USD	JE-March 2022
	622250	Charit.Contrib.Empl	10001000	Charit.Contrib.Empl	2 40	1,000.00 USD	1,000.00 USD	JE-March 2022

Journal Entry Lines

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 ;)

The screenshot shows the SAP ServiceNow interface for a Journal Entry Document. At the top, there's a header bar with a back arrow, a refresh icon, and tabs for 'Journal Entry Document' and 'JE0001003'. Below the header are several input fields: 'Number' (JE0001003), 'Document number' (30002748), 'Fiscal year' (2022), 'Posting date' (2022-06-12), 'Company code' (1001), 'Updated' (2022-06-12 05:54:06), 'Created' (2022-06-12 03:17:41), and 'Created by' (admin). There are also 'Update' and 'Delete' buttons. Below this is a sub-grid titled 'Journal Entry Lines' with a header row containing 'Number', 'Cost center', 'Credit', 'Debit', and 'GL Account'. The data in the grid includes two rows: JELINE0001004 (Cost center 10001000, Credit 0, Debit 8,000, GL Account 622210) and JELINE0001005 (Cost center 10001000, Credit 8,000, Debit 0, GL Account 622250). A summary row shows 'Sum' for both Credit and Debit as 8,000, and the GL Account for the sum is 8,000. Navigation buttons at the bottom of the grid allow for page navigation.

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 (MRVS)?

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

The screenshot shows the App Engine Studio interface with the following details:

- Header:** now | App Engine Studio, HOME, MY APPS (highlighted), TEMPLATES, RESOURCES.
- Top Bar:** App Home, Exercise 2: Financial Clo..., Source control (with a red arrow pointing to it), Submit, More options.
- Search Bar:** Search all.
- Content Area:**
 - Data (5):** Journal Entry Line (Table), Journal Entry Document (Table).
 - Experience (1):** Journal Entry Line | Default view (Form).
 - Logic and automation (2):** Add a process or third-party integration.

2. Click Open app in Dev Studio

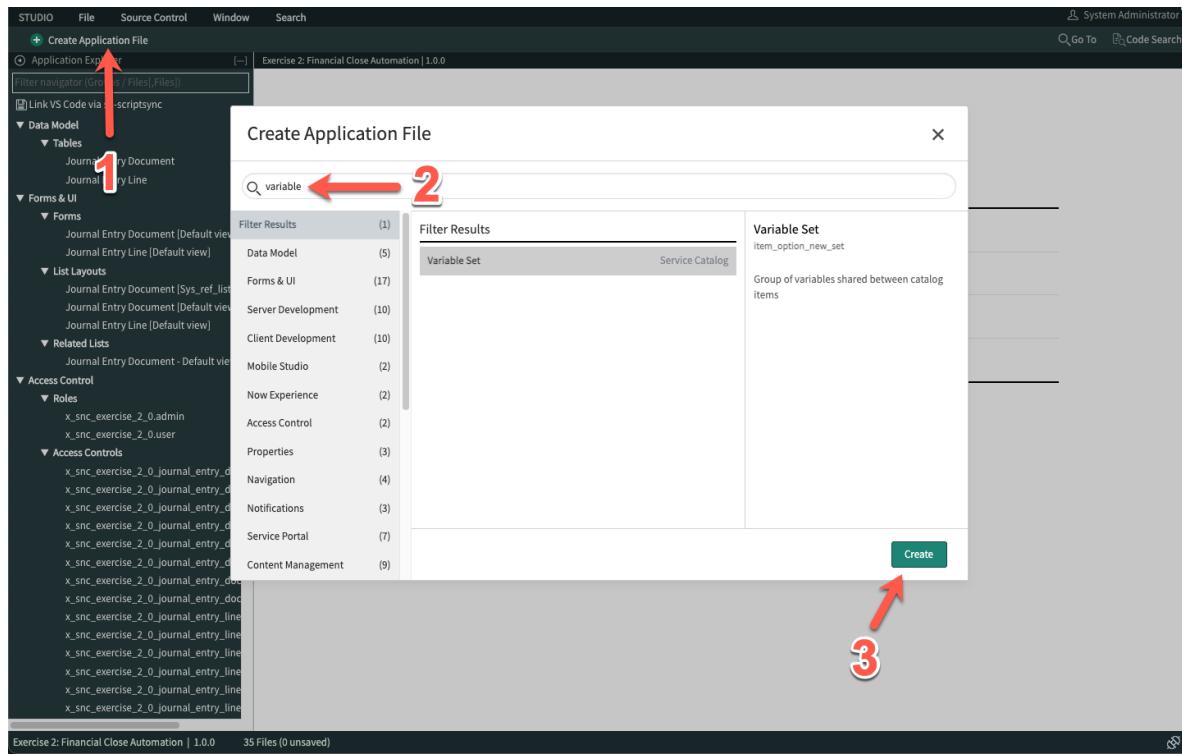
The screenshot shows the Application Properties screen for the 'Exercise 2: Financial Close Automation' app:

- Header:** now | App Engine Studio, HOME, MY APPS (highlighted), TEMPLATES, RESOURCES.
- Title:** Application Properties (with a close button).
- General Tab:** Selected.
- General Section:**
 - Name: Exercise 2: Financial Close Automation
 - Description: Describe your app
 - Scope: X_snc_exercise_2_0
 - Date created: 2022-06-12 10:09:51
 - sys_app ID: 102124611b8c19100c8a6461f54bcbf5 (Copy)
- Buttons:** Open app in Dev Studio (with a red arrow pointing to it), Delete application, Save.

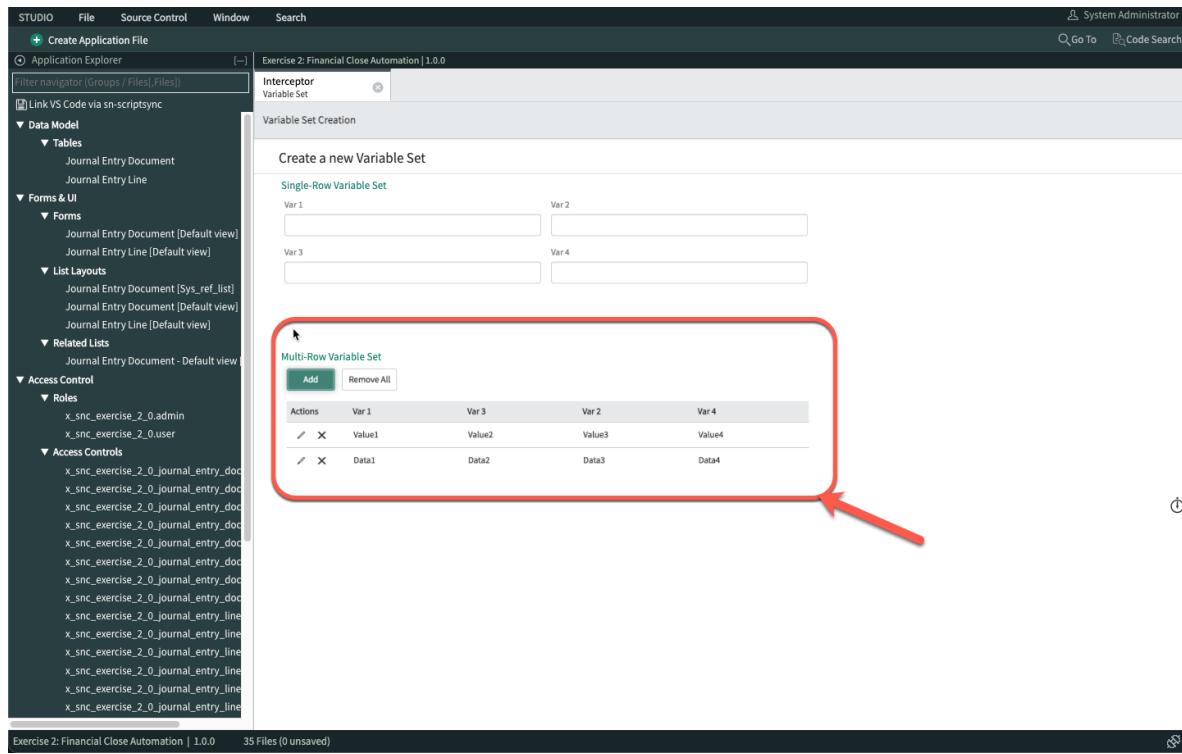
3. A new browser tab will open in Studio IDE

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

The screenshot shows the ServiceNow Studio interface. On the left, there's a sidebar with various application components like Application Explorer, Data Model, Forms & UI, Access Control, and Service Catalog. In the main workspace, a modal window titled 'Exercise 2: Financial Close Automation | 1.0.0' is open. It shows a 'Variable Set' configuration screen. A red arrow labeled '1' points to the 'Title' field, which contains 'Journal Entry'. Another red arrow labeled '2' points to the 'Submit' button in the top right corner of the modal.

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**

The screenshot shows the ServiceNow Studio interface. The left sidebar includes Application Explorer, Access Controls, and Service Catalog. The main area displays a 'Variable' configuration screen for a 'Journal Entry' variable set. A red arrow points to the 'Question' field in the 'Question' tab, which contains 'GL Account'. The 'Name' field is also visible below it. The 'Submit' button is located at the bottom right of the modal.

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

The screenshot shows the ServiceNow Studio interface with the following details:

- Variable Set Details:**
 - Title: Journal Entry
 - Internal name: journal_entry
 - Order: 100
 - Type: Multi Row
 - Application: Exercise 2: Financial Close Automation
 - Variable Set attributes: (empty)
 - Layout: 1 Column Wide
- Variables Catalog:**

Name	Type	Question	Order
gl_account	Single Line Text	GL Account	100
cost_center	Single Line Text	Cost center	200
debit	Single Line Text	Debit	300
credit	Single Line Text	Credit	400

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**

The screenshot shows the SAP Studio interface for configuring a Record Producer. The left sidebar has sections like 'STUDIO', 'File', 'Source Control', 'Window', 'Search', 'Create Application File', 'Application Explorer', 'Link VS Code via sn-scriptsync', 'Access Controls', 'Service Catalog', 'Variable Sets', 'Journal Entry', 'Flow Designer', and 'Actions'. The main area is titled 'Service Catalog > Record Producer > New Record Producer' and shows a 'Record Producer' entry with a status of 'New record'. The form fields include:

- Name:** Po
- Table name:** -- None --
- State:** -- None --
- Checked out:** -- None --
- Owner:** System Administrator
- Application:** Exercise 2: Financial Close Automation
- Active:** checked

Below the form are tabs for 'What it will contain', 'Accessibility', 'Generated Record Data', and 'Portal Settings'. A large text area for 'Description' contains placeholder text: 'Add a short description and a full description'. It includes a rich text editor toolbar. Below the description area is a note: 'Add relevant tags to the Meta field using comma-separated list of tags. These tags will be used while searching the item. Not applicable if AI Search is configured.' The 'Meta' field is empty.

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

The screenshot shows the ServiceNow Studio interface for creating a new variable. The variable is named 'Company code' and is defined as follows:

- Application:** Exercise 2: Financial Close Automation
- Type:** Single Line Text
- Catalog Item:** Post Journal Entry
- Order:** 100
- Field:** x_snc_exercise_2_0_journal_entry_doc
- Active:** Checked
- Map to field:** x_snc_exercise_2_0_journal_entry_doc

The 'Question' tab is selected, containing the following fields:

- * Question:** Company code
- * Name:** company_code
- Tooltip:** (empty)
- Example Text:** (empty)

A 'Submit' button is located at the bottom left of the form.

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*)

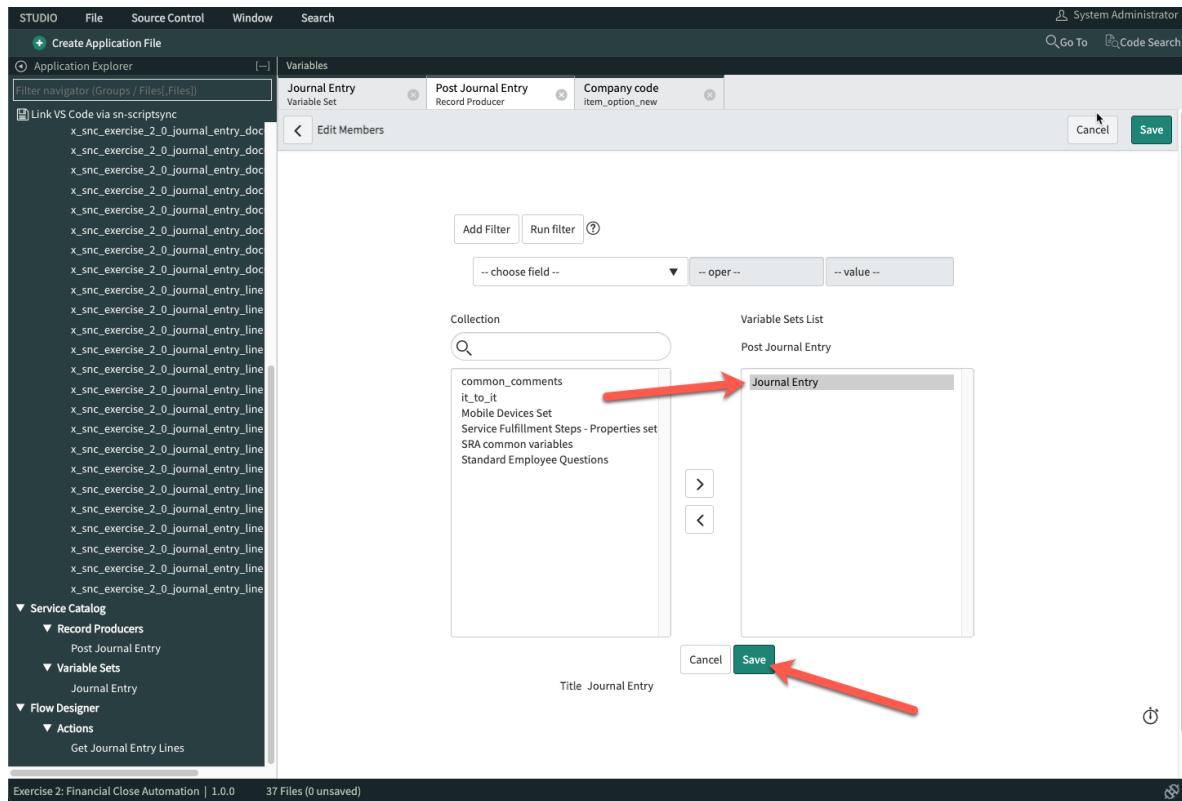
6. Click Submit

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

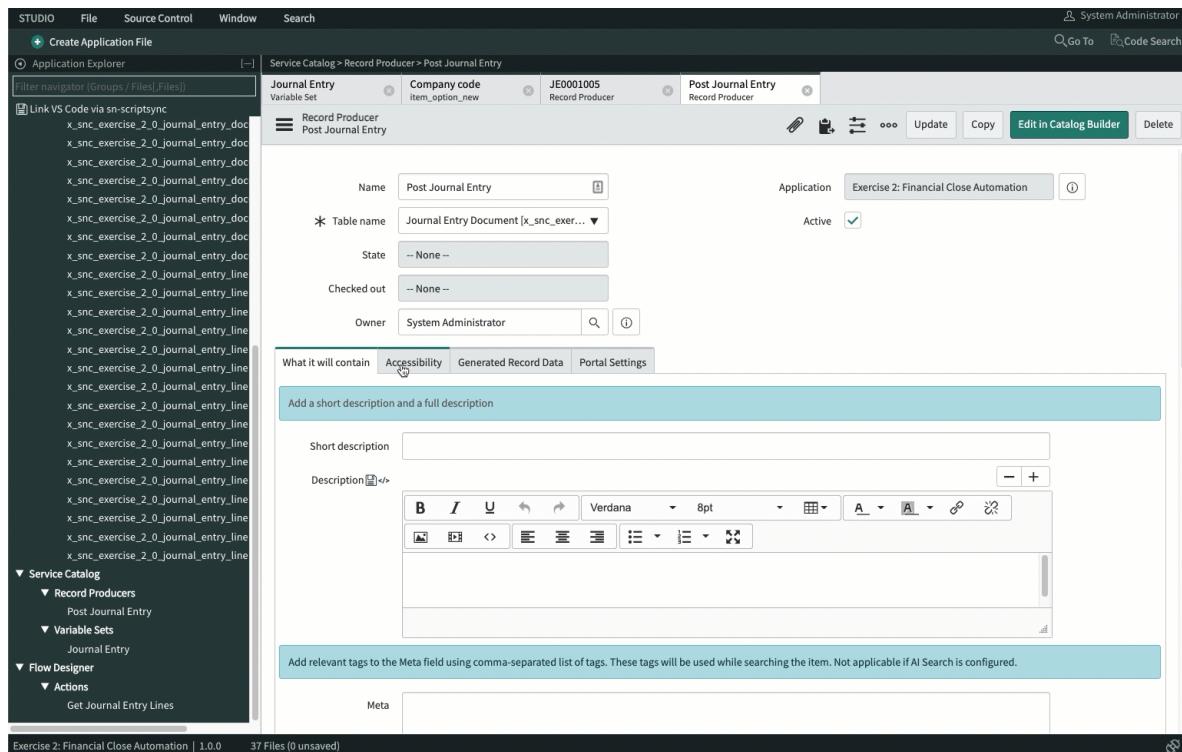


11. Back on the Record Producer screen, click the **Accessibility** tab,

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**

The screenshot shows the SAP Service Catalog interface. On the left, there's a sidebar with categories like 'Service Catalog', 'Record Producers', 'Variable Sets', 'Flow Designer', and 'Actions'. Under 'Actions', there's a sub-item 'Post Journal Entry'. To the right of the sidebar, there's a code editor window with some Java-like code. Below the code editor is a button labeled 'Edit in Catalog Builder'. A red arrow points to this 'Edit in Catalog Builder' button.

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

The screenshot shows a 'Journal Entry' pop-up modal. It has a header 'Journal Entry' and a sub-header 'Add Row'. Inside, there are four input fields: 'GL Account' (value: 622210), 'Cost center' (value: 10001000), 'Debit' (value: 8000), and 'Credit' (value: 0). At the bottom right of the modal are two buttons: 'Cancel' and 'Add', with 'Add' being highlighted with a green box.

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

STUDIO File Source Control Window Search

Create Application File

Application Explorer Variables

Journal Entry Record Producer Company code item_option_new

Link VS Code via sn-scripts: x_snc_exercise_2_0_journal_entry_doc ...

Company code 1002

Journal Entry

Add	Remove All			
Actions	GL Account	Cost center	Debit	Credit
/ X	622210	10001000	8000	0
/ X	622250	10001000	0	8000

Submit

Exercise 2: Financial Close Automation | 1.0.0 37 Files (1 unsaved)

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

The screenshot shows the App Engine Studio interface. In the top left, there's a 'STUDIO' menu with options like 'File', 'Source Control', 'Window', and 'Search'. A 'Create Application File' button is also present. On the right, there are links to 'System Administrator', 'Go To', and 'Code Search'. The main content area is titled 'Service Catalog > Record Producer > Post Journal Entry'. It displays a 'Journal Entry Document' with ID 'JE0001005'. The document has fields for 'Number' (JE0001005), 'Document number' (empty), 'Fiscal year' (empty), 'Posting date' (empty), 'Company code' (1002), 'Updated' (2022-06-12 19:53:57), 'Created' (2022-06-12 19:53:57), and 'Created by' (admin). Below this is a table titled 'Journal Entry Lines' showing two entries:

Number	Cost center	Credit	Debit	GL Account
JELINE0001012	10001000	0	8,000	622210
JELINE0001013	10001000	8,000	0	622250

The bottom of the screen shows a status bar with 'Exercise 2: Financial Close Automation | 1.0.0' and '37 Files (1 unsaved)'.

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

The screenshot shows the configuration interface for a flow named 'Post Journal Entry to ...'. In the 'TRIGGER' section, a new trigger is defined for 'Journal Entry Document Created' on the 'Created' event of the 'Journal Entry Document' table. The 'Actions' section displays a search result for 'sap erp', specifically the 'SAP ERP' spoke. Under this spoke, several actions are listed: 'Look up Sales Orders', 'Update Sales Order', 'Journals', and 'Post Journal Entry', which is highlighted with a red border.

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

The screenshot shows the configuration interface for a flow named 'Post Journal Entry to ...'. In the 'Actions' section, a search for 'sap erp' has been performed, and the results for the 'SAP ERP' spoke are displayed. The 'Post Journal Entry' action is highlighted with a red border.

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.

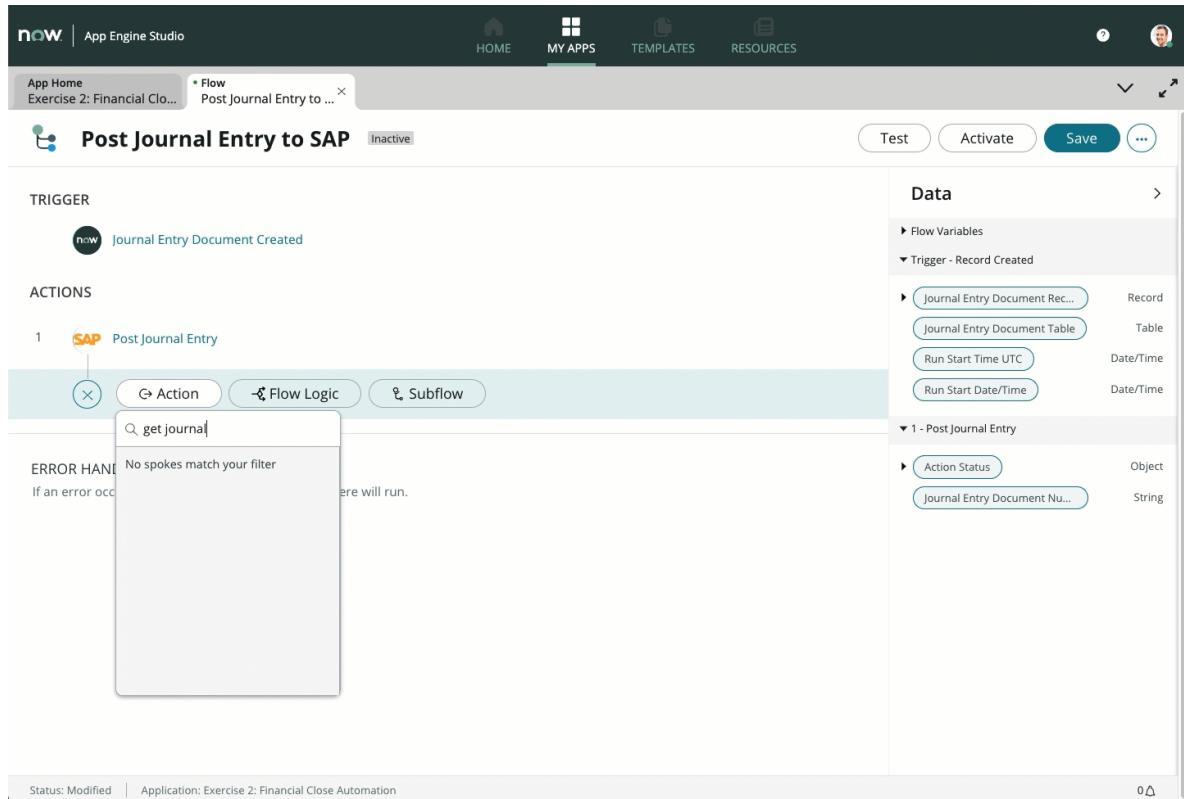
18. Click **Add an Action, Flow Logic, or Subflow**

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**

Action Outline

- 1 Look Up Records step (Look Up Records)
- 2 Script step (Script)

Action Output

Label	Value
Journal Entry Lines	step > ... > Journal Entry Lines
Action Status	Drag and drop object type pill
Code	
Message	

Data

- Input Variables
 - Journal Entry Document (Record)
 - Look Up Records step
 - Count (Integer)
 - records (Records)
 - Table (Table Name)
 - Script step
 - Journal Entry Lines (Array.Object)
 - Step Status (Object)
- Output Variables
 - Action Status (Object)
 - Journal Entry Lines (Array.Object)

javascript:void()

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

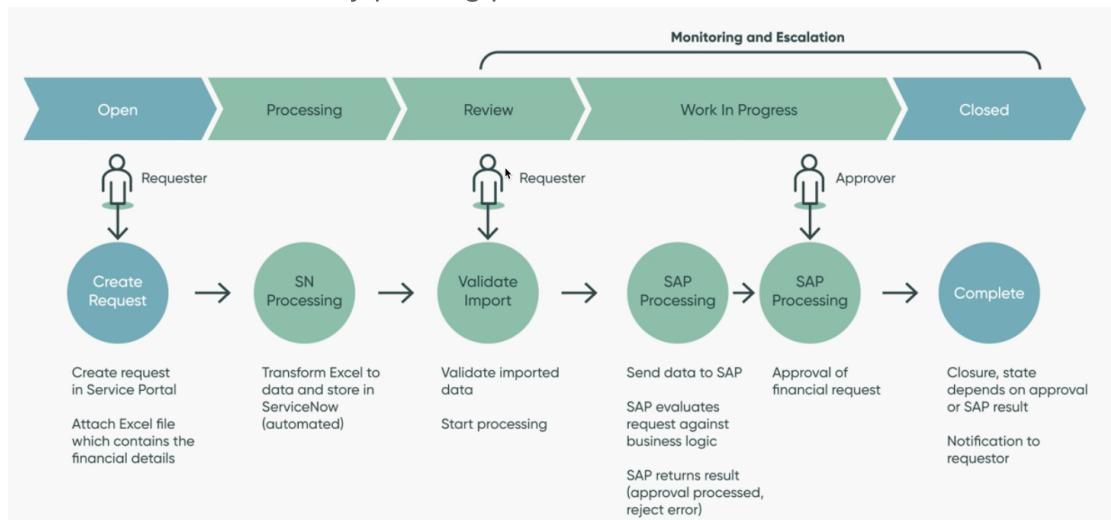
The screenshot shows the App Engine Studio interface with the following details:

- App Home:** Exercise 2: Financial Clo...
- Flow:** Post Journal Entry to ...
- Action:** Get Journal Entry Lin...
- Trigger:** Journal Entry Document Created
- Actions:**
 1. Get Journal Entry Lines
 2. Post Journal Entry
- Error Handler:** If an error occurs in your flow, the actions you add here will run.
- Data:** Shows the flow's variables and their types:
 - Trigger - Record Created
 - Journal Entry Document Rec... Record
 - Journal Entry Document Table Table
 - Run Start Time UTC Date/Time
 - Run Start Date/Time Date/Time
 - 1 - Get Journal Entry Lines
 - Action Status Object
 - Journal Entry Lines Array/Object
 - 2 - Post Journal Entry
 - Action Status Object
 - Journal Entry Document Nu... String

34. Click Activate on the top right of the screen

What are we missing here?

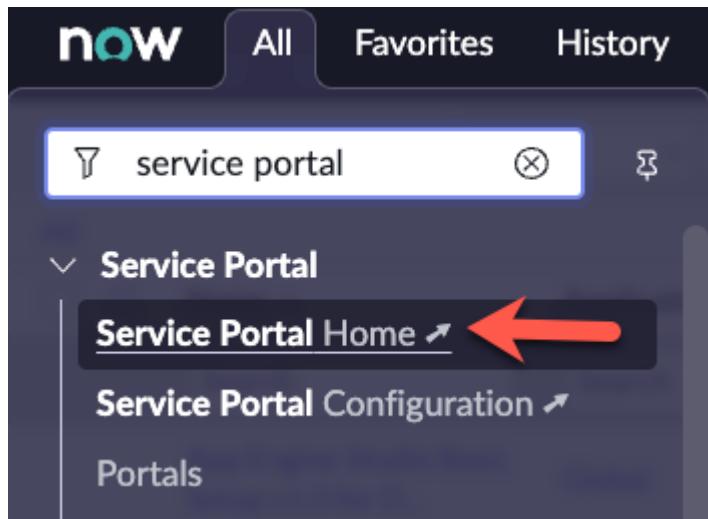
Once again we are skipping the approvals that has to happen before a Journal Entry actually gets posted into SAP in the interest of time. There are also often many validation rules for different Journal Entry types (Intercompany Posting, Asset Depreciation Posting, etc.) that we are not configuring in 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**

A screenshot of the ServiceNow search results page. The search bar at the top contains the text 'post journal entry'. On the left, there are filters for 'Sources' (set to 'All'), 'Knowledge Bases', 'Category', 'Author', 'Last modified', 'View Count', and 'Catalogs'. On the right, the search results are displayed under the heading 'All results for "post journal entry"'. The first result is 'Post Journal entry', which is highlighted with a yellow box and a red arrow. Below it are other results: 'How to Deal with Spam' and 'What is Spam?'. Both of these have small red arrows pointing to them. At the bottom right, it says 'End of results'.

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

Journal Entry

Add Row

GL Account
622210

Cost center
10001000

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

Post Journal Entry

Actions	GL Account	Cost center	Debit	Credit
	622210	10001000	8000	0
	622250	10001000	0	8000

Add attachments

Submit



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**

now | App Engine Studio

App Home Exercise 2: Financial Clo... Flow Post Journal Entry to ... Action Get.Journal Entry Lin... Executions

Test Deactivate Activate Save ...

Post Journal Entry to SAP Active

TRIGGER

Journal Entry Document Created

ACTIONS

- Get Journal Entry Lines
- Post Journal Entry
- Update Journal Entry Document Record
- + Add an Action, Flow Logic, or Subflow

ERROR HANDLER

If an error occurs in your flow, the actions you add here will run.

Properties Executions Flow stages Manage flow catalog variables Flow Variables Configure connections Copy flow Flow preferences Create code snippet

1 - Get Journal Entry Lines

- Action Status Object
- Journal Entry Lines Array.Object

2 - Post Journal Entry

- Action Status Object
- Journal Entry Document Nu... String

3 - Update Record

- Journal Entry Document Rec... Record
- Journal Entry Document Table Table
- Action Status Object

Status: Published Application: Exercise 2: Financial Close Automation

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

The screenshot shows the 'Executions' tab of the App Engine Studio. It lists one execution entry:

- Name:** Post Journal Entry to SAP
- State:** Complete
- Runtime:** 48 ms
- Created by:** admin
- Sys ID:** 011428f1878499103070685156ee9
- Created:** 2022-06-12 22:01:13

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

The screenshot shows the 'Execution Details' view for the 'Post Journal Entry to SAP' flow. It displays the following information:

- Flow Statistics:** Run as: System Administrator, Open Flow Logs, Completed, Start time: 2022-06-12 22:01:17, Duration: 49ms.
- Trigger:** Journal Entry Document Created.
- Actions:**
 - Number: JE0001006
 - Document number: 3000376 (highlighted with a red arrow)
 - Fiscal year:
 - Company code: 1003
 - Created: 3 minutes ago (2022-06-12 22:01:13)
 - Updated: 3 minutes ago (2022-06-12 22:01:17)
 - Created by: admin
- Configuration:**
 - Record: JE0001006 (highlighted with a red arrow)
 - Table: x_snc_exercise_2_0_journal_entry_document
 - Fields: document_number=3000376
 - Trig... ▶ Journal Entry Docu... (highlighted with a red arrow)
 - Core Action: Completed, Start time: 2022-06-12 22:01:17, Duration: 10ms.
 - Completed, Start time: 2022-06-12 22:01:17, Duration: 0ms.
 - Completed, Start time: 2022-06-12 22:01:17, Duration: 36ms.
- Output Data:**

VARIABLE NAME	RUNTIME VALUE	CONFIGURATION	TYPE
Action Status	{"Action Status": {"code": 0, "message": "Success"}}		Object

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

Display Document: Data Entry View

Document Number: 3000021721 Company Code: 1000 Fiscal Year: 2022
 Document Date: 03/30/2022 Posting Date: 03/30/2022 Period: 3
 Reference: JE-MARCH 2022 Cross-Comp.No.:
 Currency: USD Texts Exist: Ledger Group:

Journal Entry Document

CoCode	Account	G/L Account Name	Cost Center	Description	Item Key	Amount Crcy	Local Crcy Amt LCurr	LC3 Amount Text
1000	622210	401K	10001000	401K	1 50	1,000.00 USD	1,000.00 USD	JE-March 2022
	622250	Charit.Contrib.Empl	10001000	Charit.Contrib.Empl	2 40	1,000.00 USD	1,000.00 USD	JE-March 2022

Journal Entry Lines

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.