

Hands-on Lab: Building Joule Skills in SAP Build

Aleksandrs Antonuks, Vass, aleksandrs.antonuks@vasscompany.com

Michael Pytel, Vass, michael.pytel@vasscompany.com

Sheldon Lipshitz, Vass, sheldon.lipshitz@vasscompany.com

<https://vasscompany.com>

Index

Table of Contents

Environment Information	2
Lab Goal	3
Create Actions for Joule	3
Create a Joule Skill	14
Test your Joule Skill	30

Environment Information

1. Please use GOOGLE CHROME as your browser
2. SAP Build App URLs
 - o [SAP Build Apps](#)
3. Lab User Information
 - o User: labuser<##>@vasscompany.com
 - o Password: Newpa55!
4. Replace <##> with your Lab User Number assigned by the instructors

The screenshot shows the SAP ID Service sign-in interface. At the top, it displays the SAP logo and the text "Universal ID". Below this, there is a "Sign In" button. The main area contains fields for "E-Mail, ID, or Login Name" (containing "labuser01@vasscompany.com") and "Password" (containing "Newpa55!"). To the right of the password field is a "Forgot password?" link. At the bottom of the form is a "Continue" button.

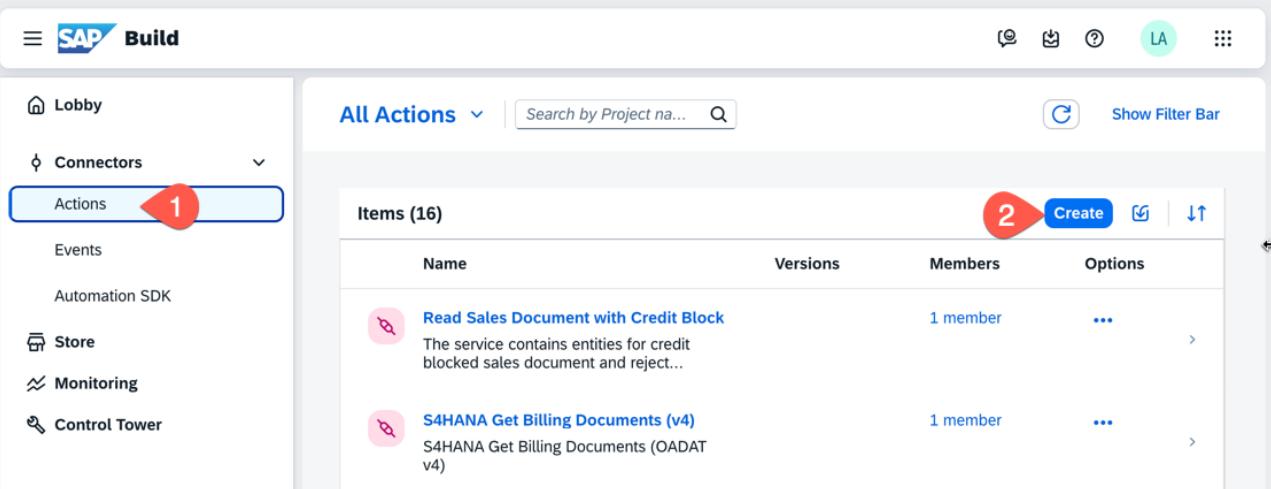
Lab Goal

We will be building a Joule Skill that can review Sales Orders with the status 'Credit Hold' and then help the business user draft an email to notify the customer of their open Invoices to be paid before the Sales Order will be processed.

Create Actions for Joule

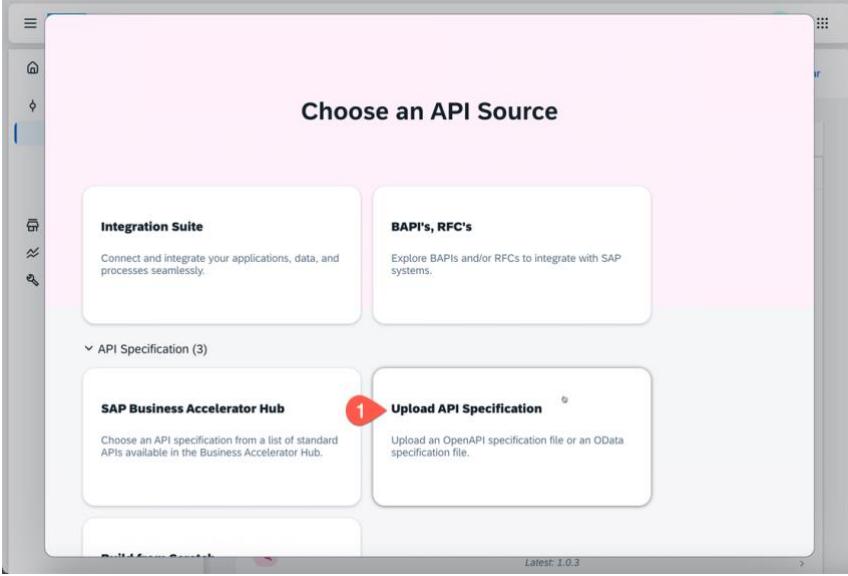
Our first step is to create Actions for Joule to use this lab. An action connects to an ODATA API.

1. Open an Chrome Browser and navigate to [SAP Build Apps](#)
2. Enter your Lab UserID and Password
3. Click on Actions, then the Create button



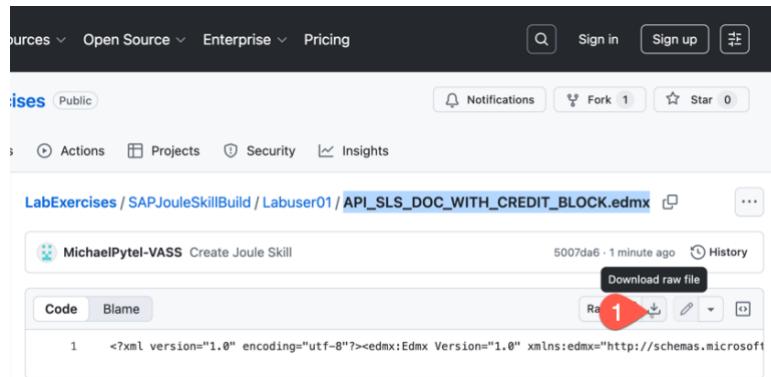
The screenshot shows the SAP Build interface. On the left, there's a sidebar with options like 'Lobby', 'Connectors', 'Actions' (which is highlighted with a red circle and the number 1), 'Events', 'Automation SDK', 'Store', 'Monitoring', and 'Control Tower'. The main area is titled 'All Actions' with a dropdown and a search bar. Below it, there's a table titled 'Items (16)' with columns for 'Name', 'Versions', 'Members', and 'Options'. Two items are listed: 'Read Sales Document with Credit Block' and 'S4HANA Get Billing Documents (v4)'. At the top right of the main area, there's a 'Create' button with a red circle and the number 2.

- 4.
5. On choose API Source, scroll down and select *Upload API Specification*



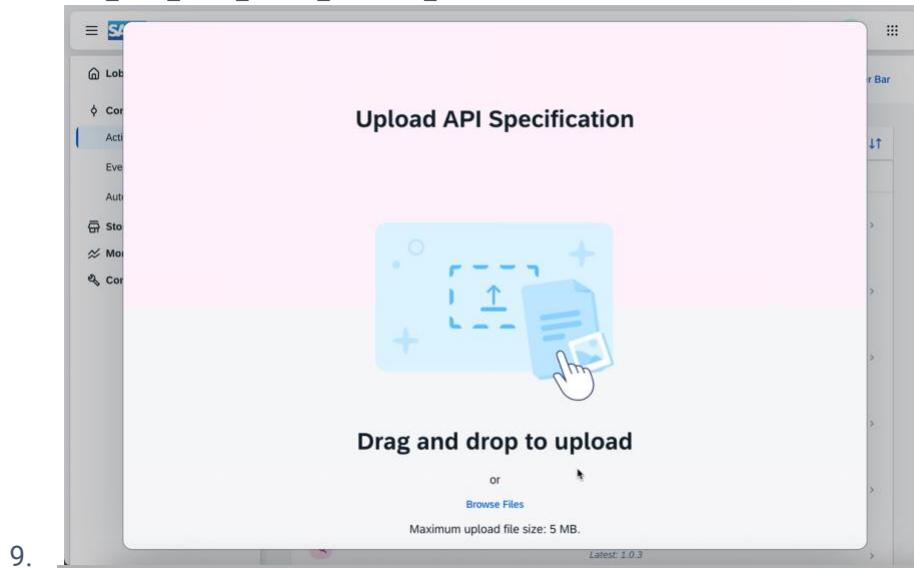
The screenshot shows a modal dialog titled 'Choose an API Source'. It has two main sections: 'Integration Suite' and 'BAPI's, RFC's'. Below these, there's a section titled 'API Specification (3)' with two options: 'SAP Business Accelerator Hub' and 'Upload API Specification'. The 'Upload API Specification' option is highlighted with a red circle and the number 1. A tooltip for this option says: 'Upload an OpenAPI specification file or an OData specification file.'

- 6.
7. Browse to your Labuser## folder on Github to download the two EDMX files
 - a. <https://github.com/MichaelPytel-VASS/LabExercises/tree/main/main/SAPJouleSkillBuild>
 - b. Open you labuserXX folder
 - c. Download the EDMX files to your local computer

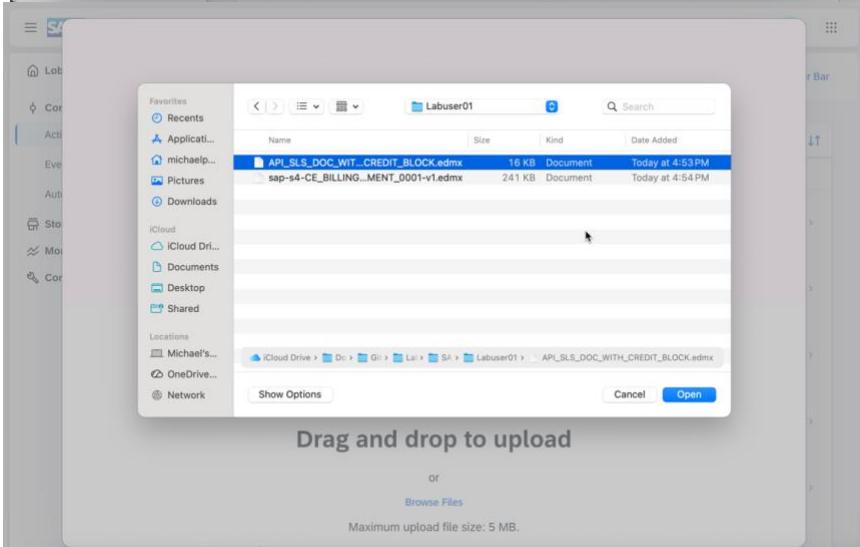


d.

8. When prompted upload the first API we will activate 'API_SLS_DOC_WITH_CREDIT_BLOCK.edmx'

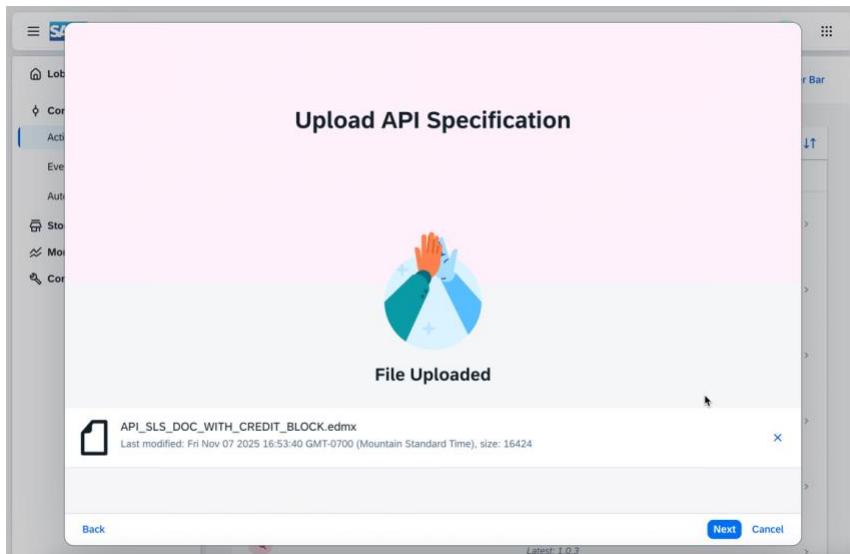


9.



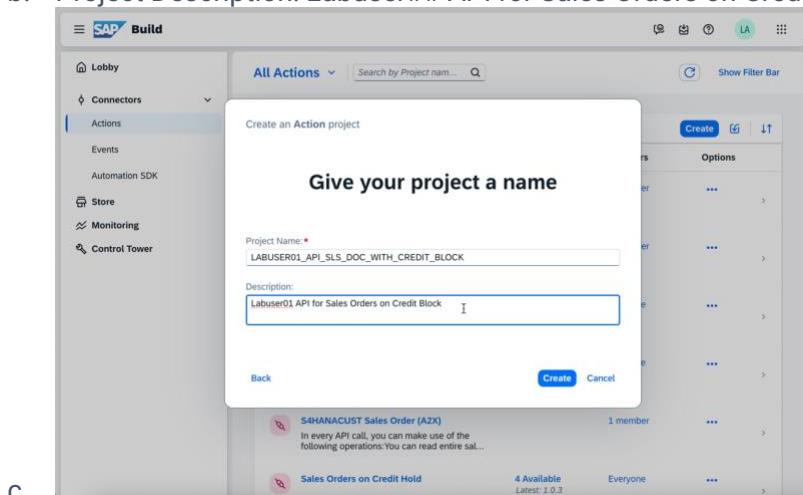
10.

11. When you complete the upload, review and click Next



12.

13. Enter the project name as shown below
- Project Name: Labuser##_API_SLS_DOC_WITH_CREDIT_BLOCK
 - Project Description: Labuser## API for Sales Orders on Credit Block

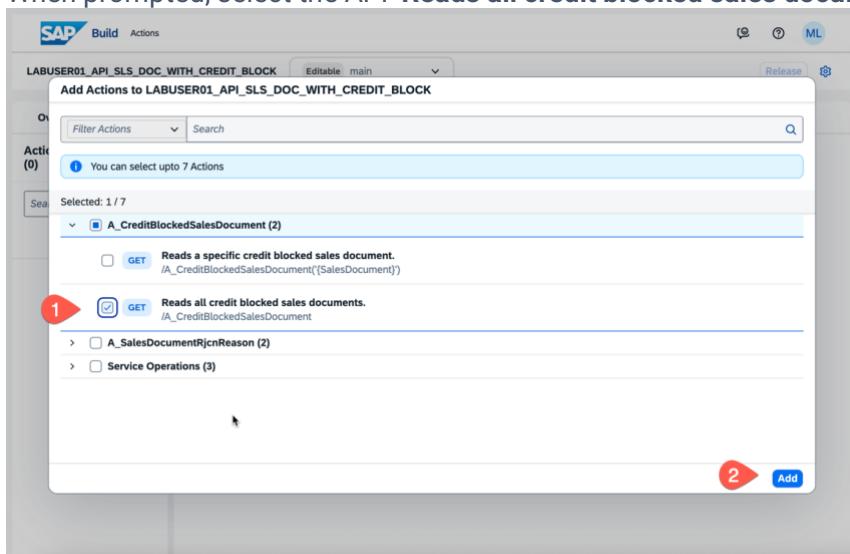


C.

- d. Click Create after entering a name

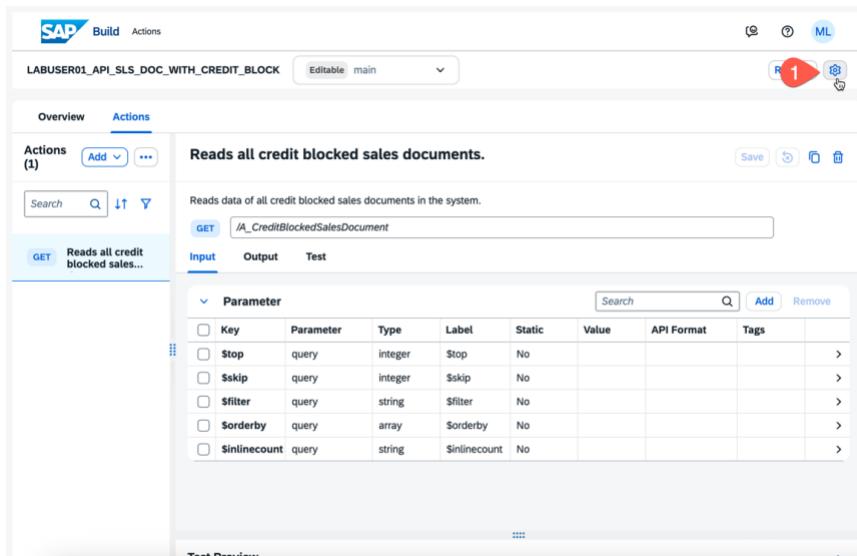
14. A new browser tab will open; check your pop-up blocker.

15. When prompted, select the API '**Reads all credit blocked sales documents.**' And Click Add.



16.

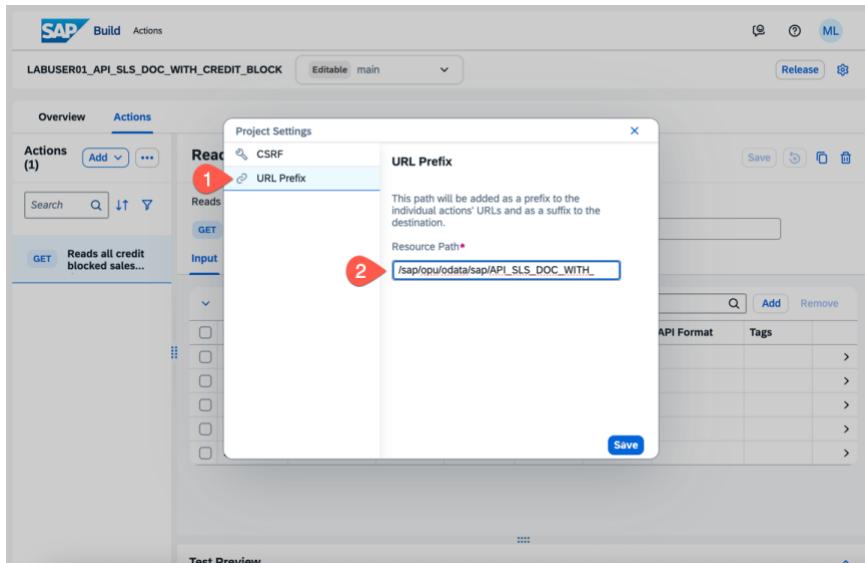
17. Click the Settings Button



The screenshot shows the SAP Build Actions interface for a project named LABUSER01_API_SLS_DOC_WITH_CREDIT_BLOCK. A single GET action is listed under the 'Actions' tab, titled 'Reads all credit blocked sales documents.' The action description states: 'Reads data of all credit blocked sales documents in the system.' The URL path is /A_CreditBlockedSalesDocument. Below the action, there is a table for parameters:

Key	Parameter	Type	Label	Static	Value	API Format	Tags
<input type="checkbox"/>	Stop	query	Stop	No			>
<input type="checkbox"/>	\$skip	query	\$skip	No			>
<input type="checkbox"/>	\$filter	string	\$filter	No			>
<input type="checkbox"/>	\$orderby	array	Sorderby	No			>
<input type="checkbox"/>	\$inlinecount	string	\$inlinecount	No			>

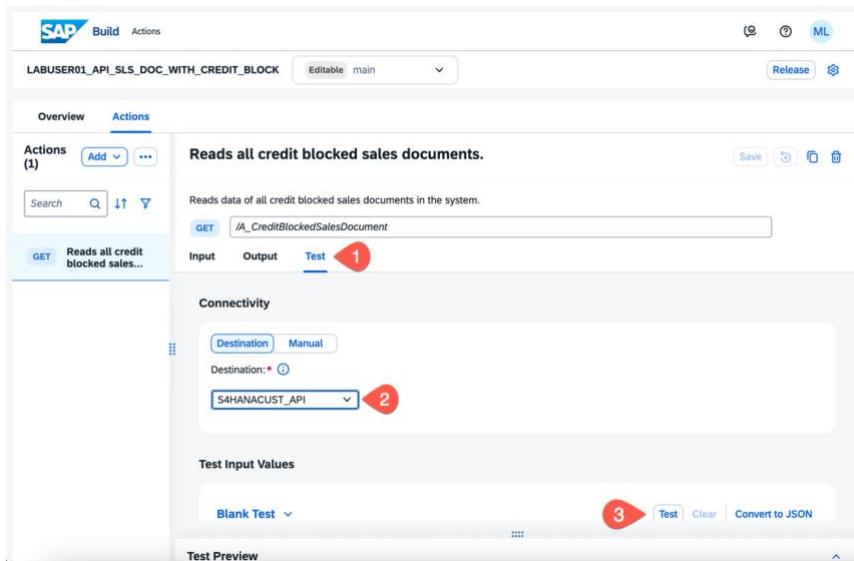
18. []
 19. Select URL Prefix and then Resource Path



The screenshot shows the SAP Build Actions interface for the same project. The 'Actions' tab is selected, showing the GET action from the previous screenshot. A modal dialog titled 'Project Settings' is open, specifically the 'URL Prefix' tab. The dialog contains the following information:

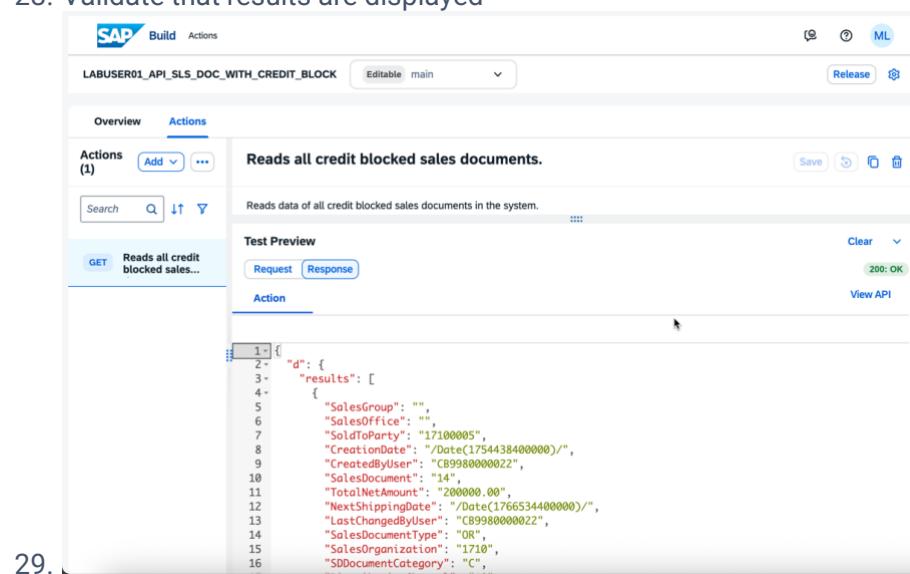
- Project Settings: CSRF, URL Prefix
- URL Prefix: This path will be added as a prefix to the individual actions' URLs and as a suffix to the destination.
- Resource Path*: /sap/opu/odata/sap/API_SLS_DOC_WITH_
- Buttons: Save, Cancel

20. []
 21. Enter in the URL prefix for the API
 a. Path: /sap/opu/odata/sap/API_SLS_DOC_WITH_CREDIT_BLOCK
 22. Click Save
 23. Next, we're going to test the API.
 24. Click the tab for Test
 25. Select the Destination: S4HANACUST_API
 26. And then select the Test button



27.

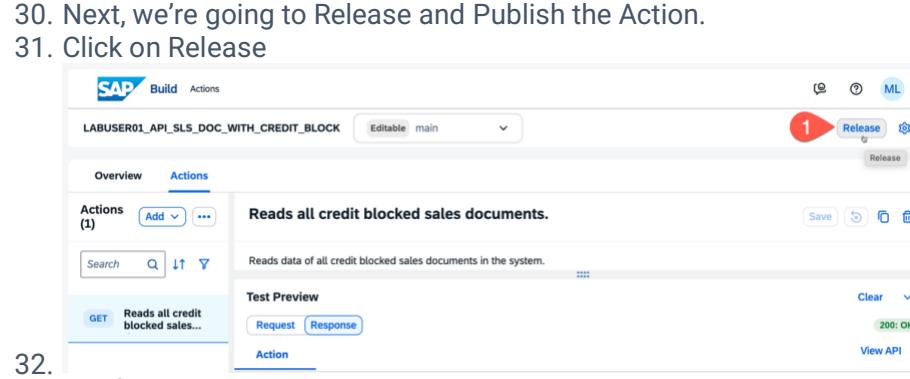
28. Validate that results are displayed



29.

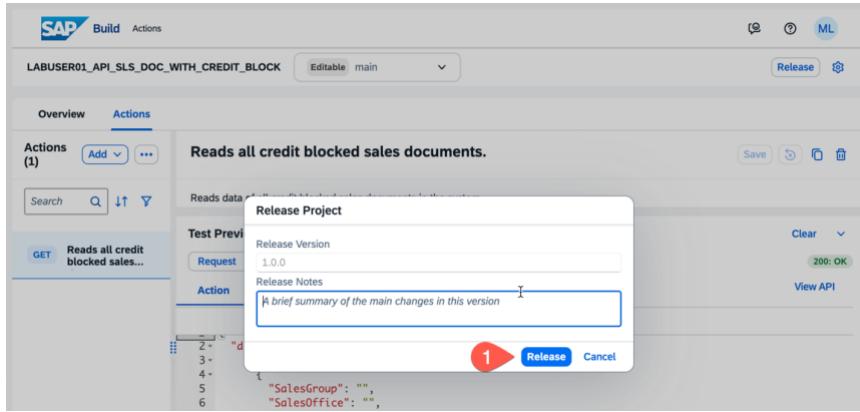
30. Next, we're going to Release and Publish the Action.

31. Click on Release



32.

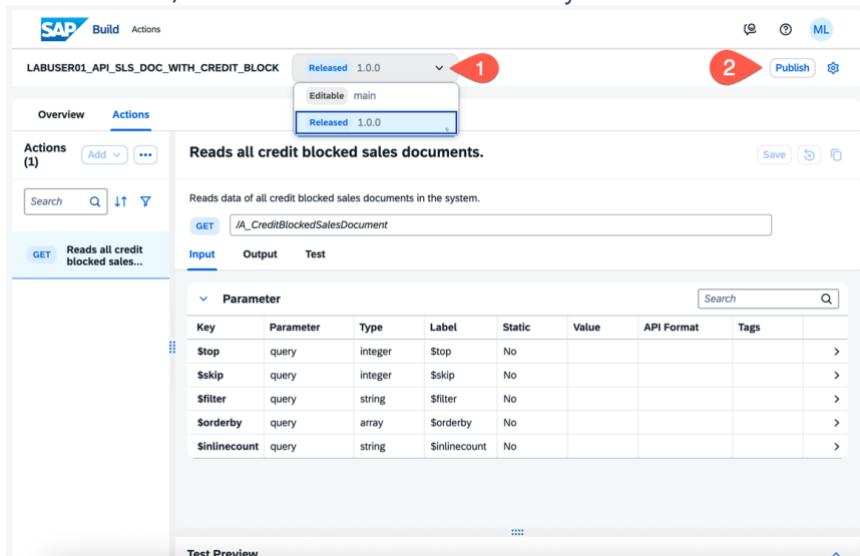
33. Confirm the Release



The screenshot shows the SAP Build Actions interface for a project named LABUSER01_API_SLS_DOC_WITH_CREDIT_BLOCK. A modal dialog titled 'Release Project' is open, prompting for a 'Release Version' (set to 1.0.0) and 'Release Notes' (containing a brief summary). A red arrow labeled '1' points to the 'Release' button at the bottom right of the dialog.

34.

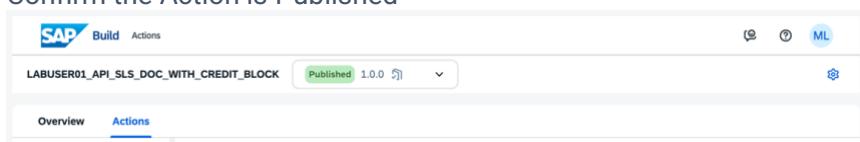
35. After Release, select the Released version of your Action. Then click Publish.



The screenshot shows the SAP Build Actions interface for the same project. The 'Test Preview' dialog is open, showing the released version '1.0.0' selected in a dropdown menu (indicated by a red arrow labeled '1'). A red arrow labeled '2' points to the 'Publish' button at the top right of the dialog.

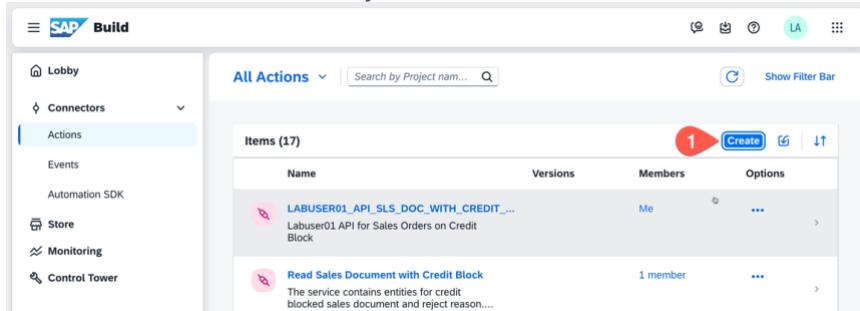
36.

37. Confirm the Action is Published



The screenshot shows the SAP Build Actions interface for the project. The 'Published' button is highlighted (indicated by a red arrow labeled '1'), confirming the action is now published.

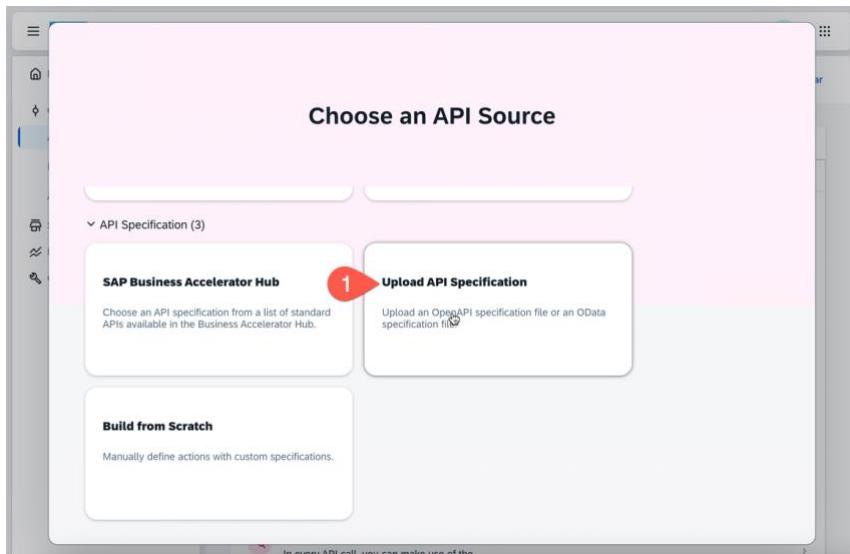
38.

39. Next, we're going to create the second Action to read open Customer Invoices
40. Return to the SAP Build Lobby, Actions, and click Create


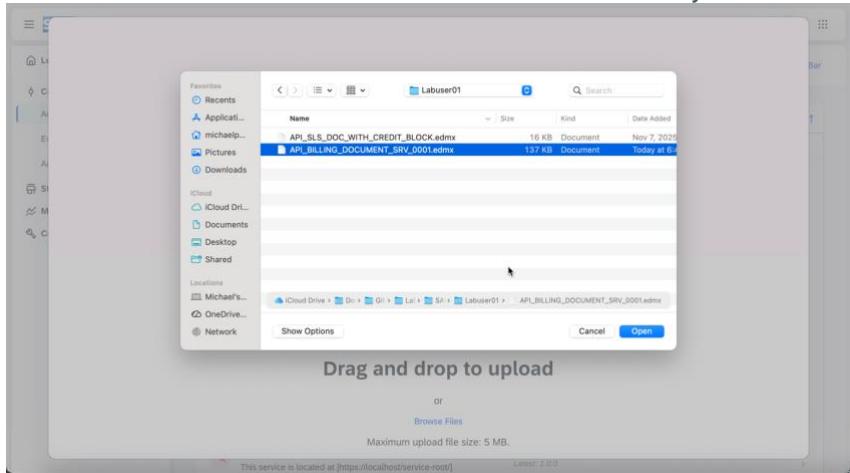
The screenshot shows the SAP Build lobby interface under the 'Actions' section. A modal dialog titled 'All Actions' is open, displaying a list of items (17 total). A red arrow labeled '1' points to the 'Create' button at the top right of the list.

41.

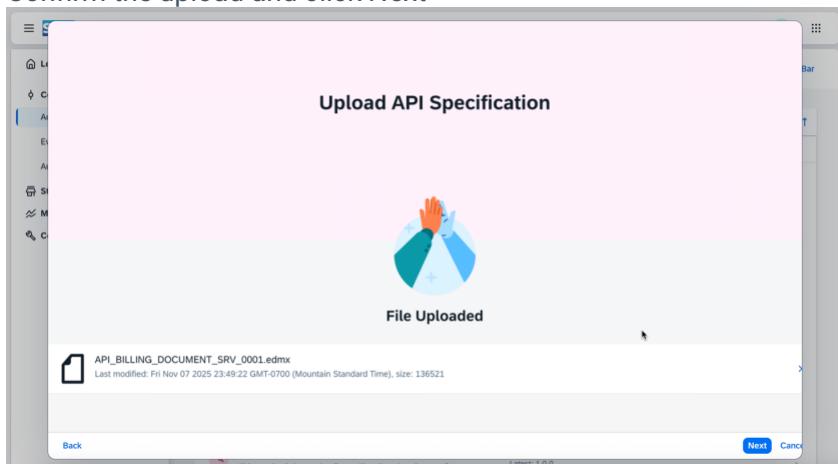
42. Choose Upload API Specification



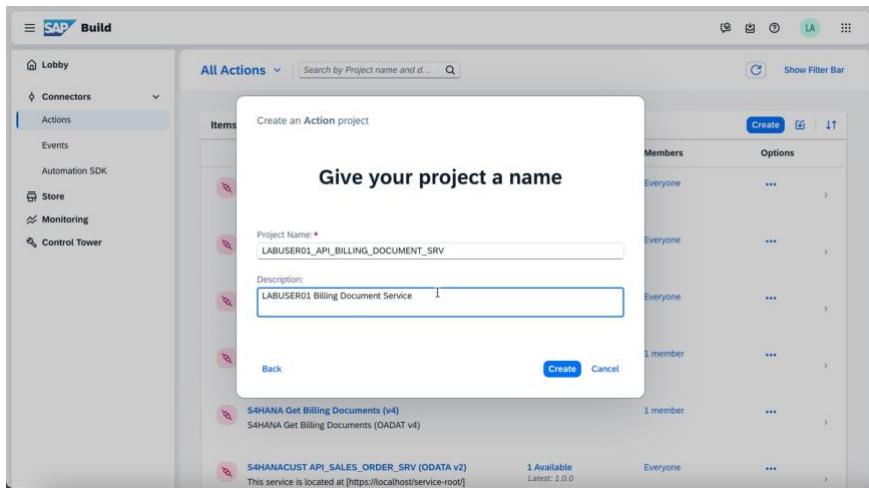
43.
44. Select the second EDMX file from the Github Folder for your Labuser##



45.
46. Confirm the upload and click Next



47.
48. Enter a Project Name and Project Description
a. Project Name: LABUSER##_API_BILLING_DOCUMENT_SRV
b. (replace ## with your labuser number)



49.

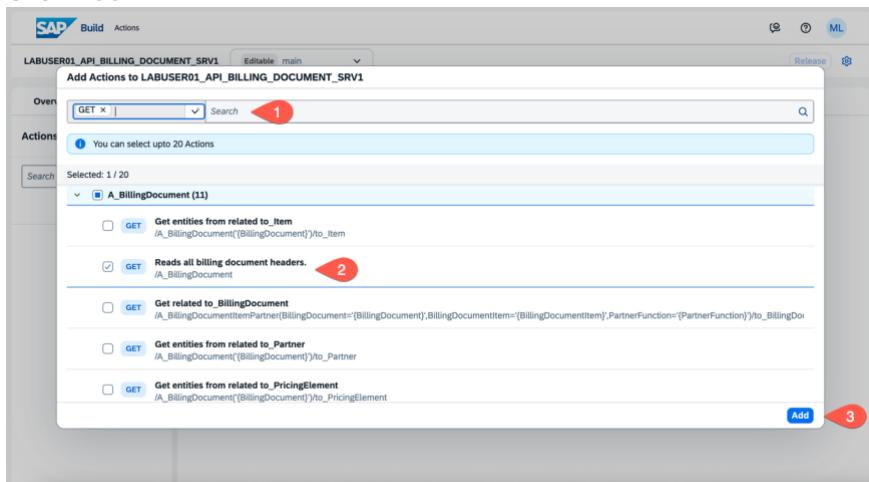
50. Click Create

51. A new browser tab will open (confirm your popup blocked is disabled)

52. When prompted to add Actions, set a filter for 'GET' actions.

53. Scroll down and select the API 'Reads all billing document headers'

54. Click Add

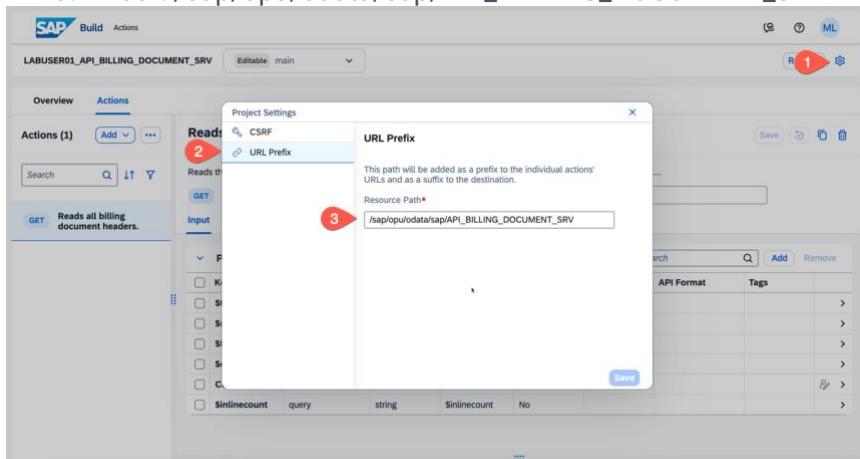


55.

56. Click the Settings icon on the upper right

57. Select URL Prefix then Enter the Resource Path below:

a. Path: /sap/opu/odata/sap/API_BILLING_DOCUMENT_SRV

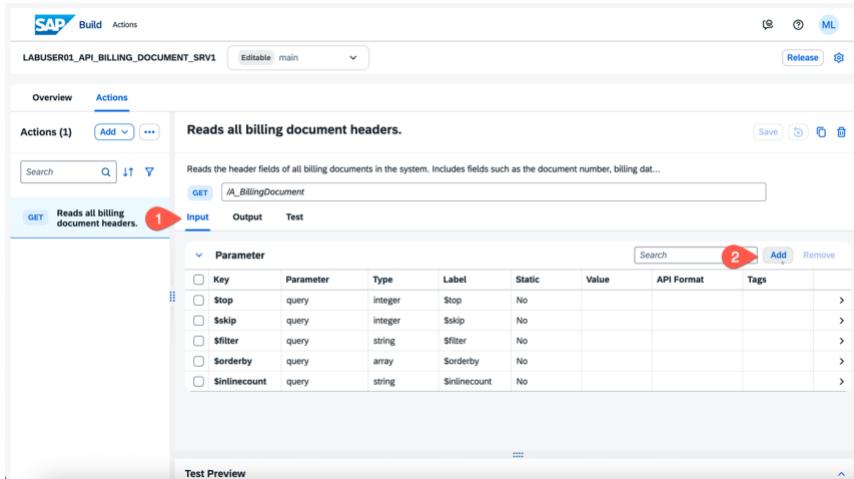


58.

59. Click Save

60. Next, we're going to add a field to help us build the filter for the ODATA Service.

61. On the Input Tab, Click Add

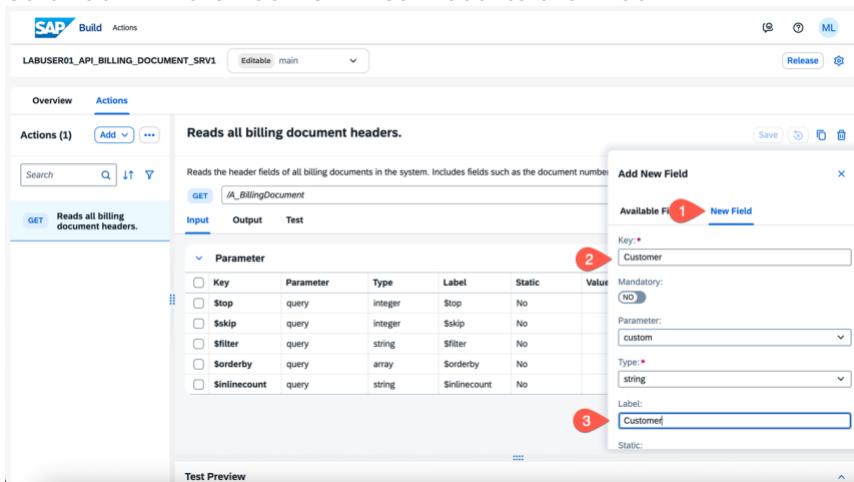


Actions (1) Add

Parameter

Key	Parameter	Type	Label	Static	Value	API Format	Tags
Stop	query	integer	Stop	No			>
\$skip	query	integer	\$skip	No			>
\$filter	query	string	\$filter	No			>
\$orderby	query	array	\$orderby	No			>
\$inlinecount	query	string	\$inlinecount	No			>

- 62.
63. Click New Field and then enter the following
- Key: Customer
 - Type: string
 - Label: Customer
 - Max Length: 20
64. Scroll down in the Add New Filed modal to click Add



Add New Field

Available Fields 1 **New Field**

Key: * **Customer**

Mandatory: **NO**

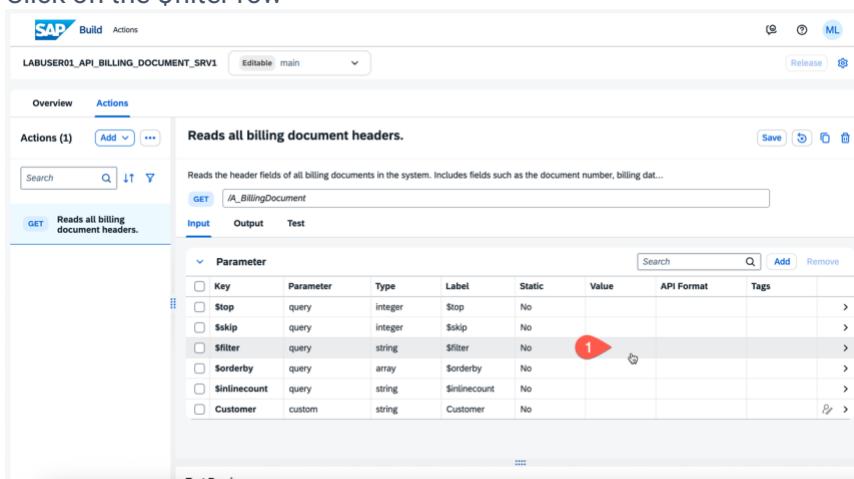
Parameter: **custom**

Type: * **string**

Label: **Customer**

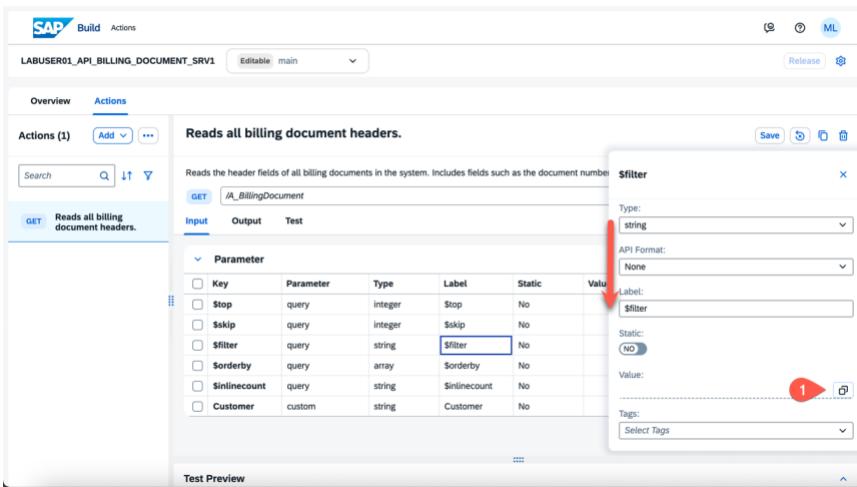
Static:

- 65.
66. Next, we're going to apply the filter.
67. Click on the \$filter row



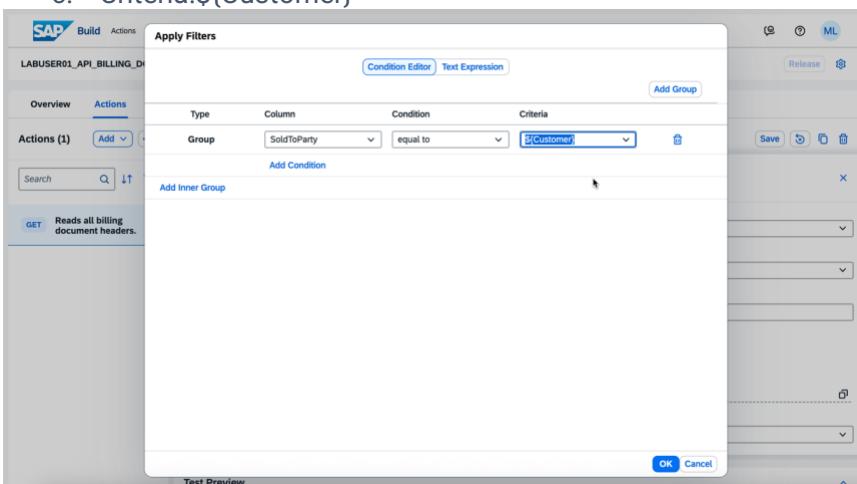
Key	Parameter	Type	Label	Static	Value	API Format	Tags
Stop	query	integer	Stop	No			>
\$skip	query	integer	\$skip	No			>
\$filter	query	string	\$filter	No			>
\$orderby	query	array	\$orderby	No			>
\$inlinecount	query	string	\$inlinecount	No			>
Customer	custom	string	Customer	No			>

- 68.
69. Scroll down and click the button  in the Value: field



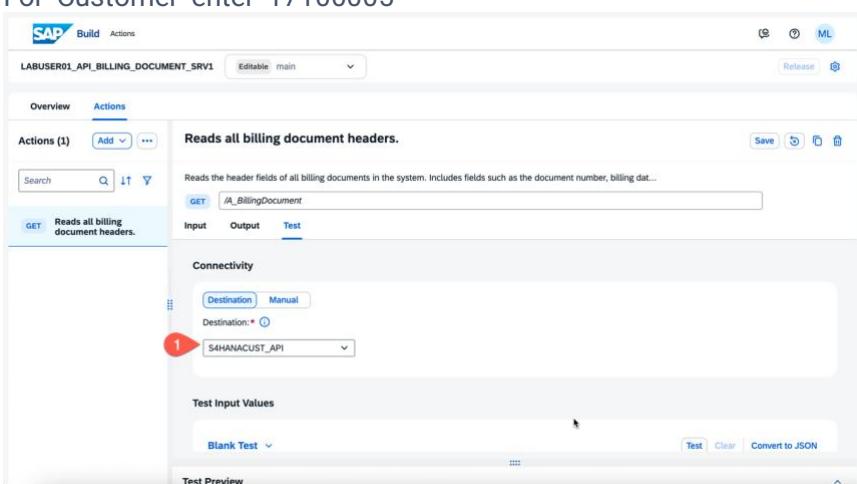
The screenshot shows the SAP Build Actions interface for a service named LABUSER01_API_BILLING_DOCUMENT_SRV1. An action named 'Reads all billing document headers.' is selected. In the 'Parameter' section, there is a table with columns: Key, Parameter, Type, Label, Static, and Value. One row has 'Parameter' set to '\$filter' and 'Value' set to '\$filter'. A red arrow points to the 'Value' field.

- 70.
71. Add a Condition as shown below. Then click OK
- Column: SoldToParty
 - Condition: equal to
 - Criteria: \${Customer}



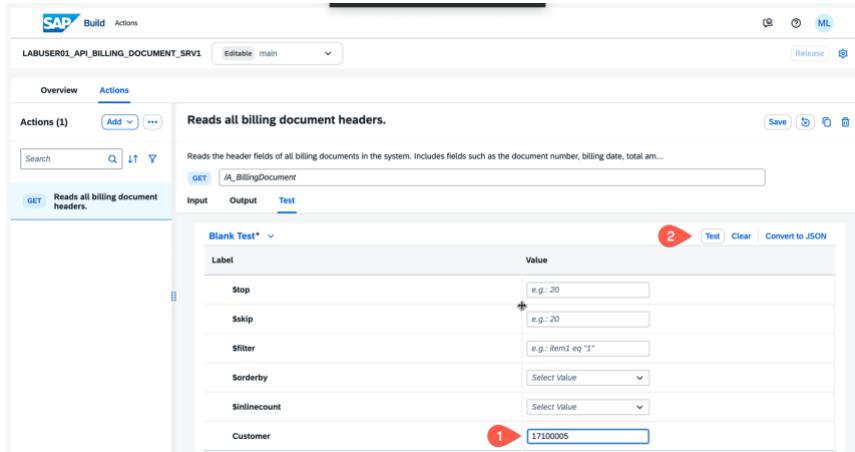
The screenshot shows the 'Apply Filters' dialog in SAP Build Actions. It displays a condition group for 'SoldToParty' set to 'equal to' with criteria '\${Customer}'. A red arrow points to the 'OK' button at the bottom right of the dialog.

- 72.
73. Close the \$filter panel by click the
74. Select the Test tab
75. Select the Destination: S4HANACUST_API; scroll down to test values
76. For 'Customer' enter '17100005'



The screenshot shows the SAP Build Actions interface for the same service. The 'Test' tab is selected. In the 'Destination' section, the dropdown is set to 'S4HANACUST_API'. A red arrow points to the 'Destination' dropdown.

- 77.

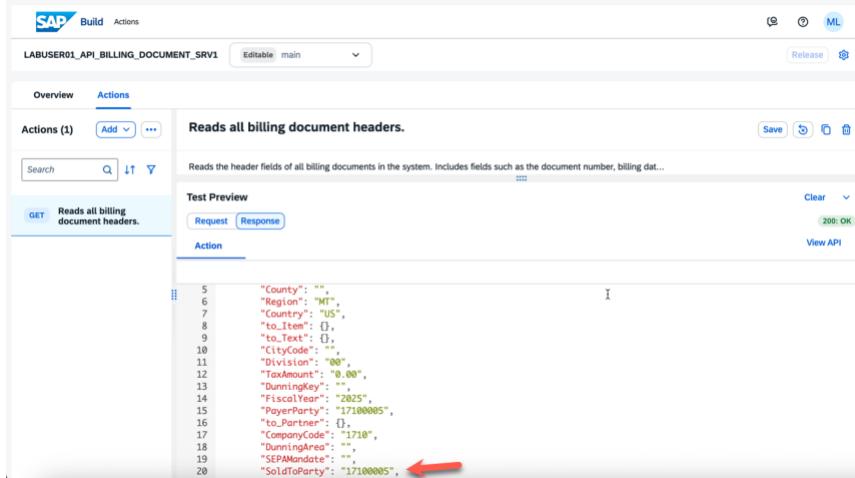


Actions (1) **GET** Reads all billing document headers.

Blank Test* **2**

Label	Value
Stop	e.g.: 20
\$skip	e.g.: 20
\$filter	e.g.: item1 eq "1"
\$orderby	Select Value
\$inlinecount	Select Value
Customer	17100005

78.
79. Click Test and confirm results are displayed (similar to below)



Test Preview

Request Response **200: OK**

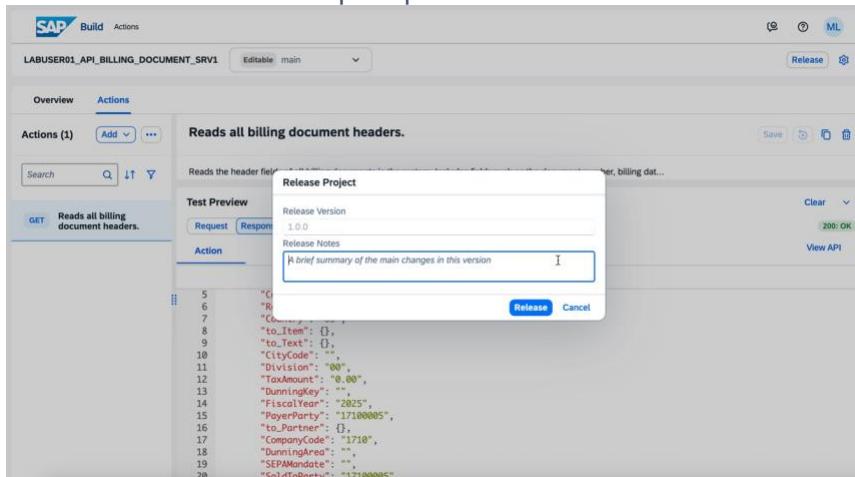
Action

```

5   "Country": "*",
6   "Region": "MT",
7   "Country": "US",
8   "to_Item": {},
9   "to_Text": {},
10  "CityCode": "*",
11  "Division": "00",
12  "TaxAmount": "0.00",
13  "DunningKey": "*",
14  "FiscalYear": "2025",
15  "PayerParty": "17100005",
16  "to_Partner": {},
17  "CompanyCode": "1710",
18  "DunningArea": "*",
19  "SEPMandate": "*",
20  "SoldToParty": "17100005",

```

80.
81. After confirm results, click the Save Button.
82. Click the Release button.
83. Confirm the Release when prompted.



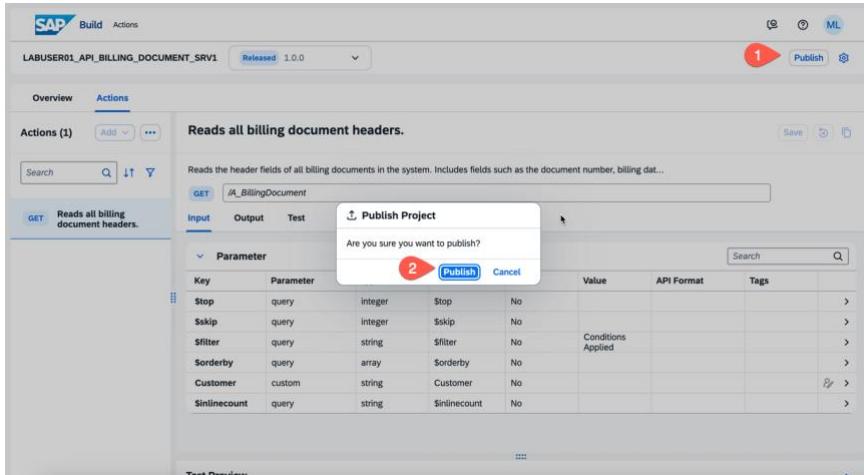
Release Project

Release Version
1.0.0

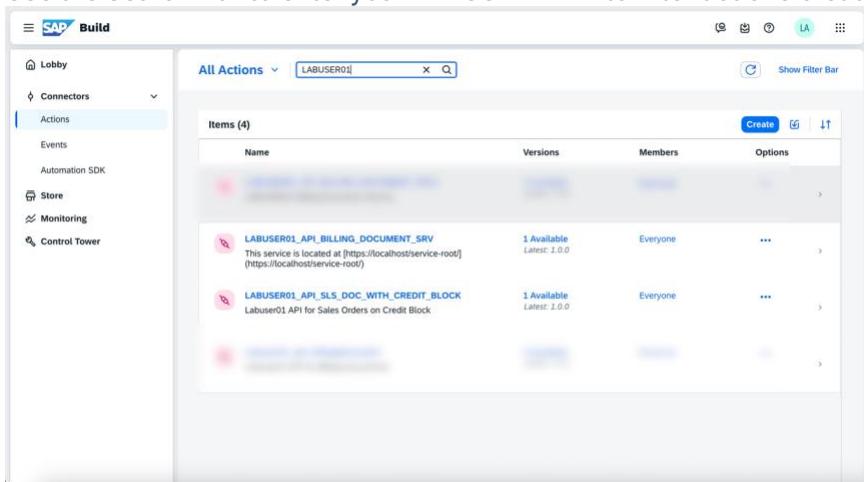
Release Notes
A brief summary of the main changes in this version

Release Cancel

84.
85. Next, we're going to click Publish and then confirm.



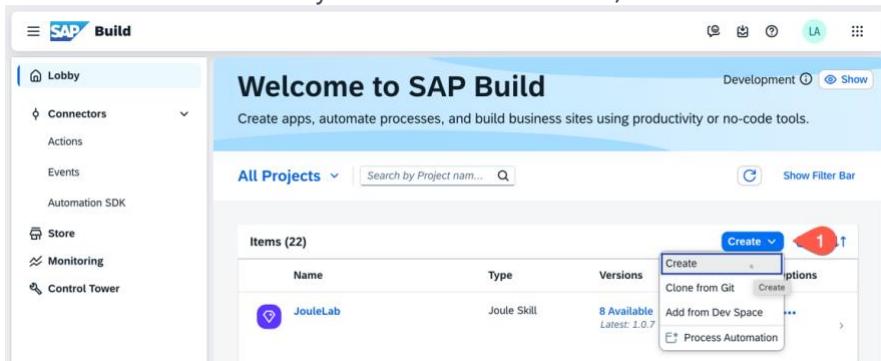
86. []
 87. We've now publish two actions we can leverage within a Joule Skill.
 88. Close the browser tab for the Action and return to the SAP Build Lobby
 89. Use the Search Bar to enter your LABUSER## ID to filter actions created.



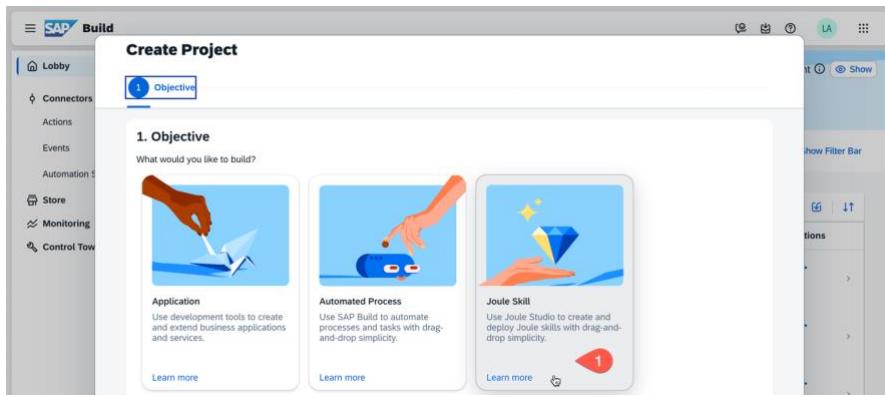
90. []

Create a Joule Skill

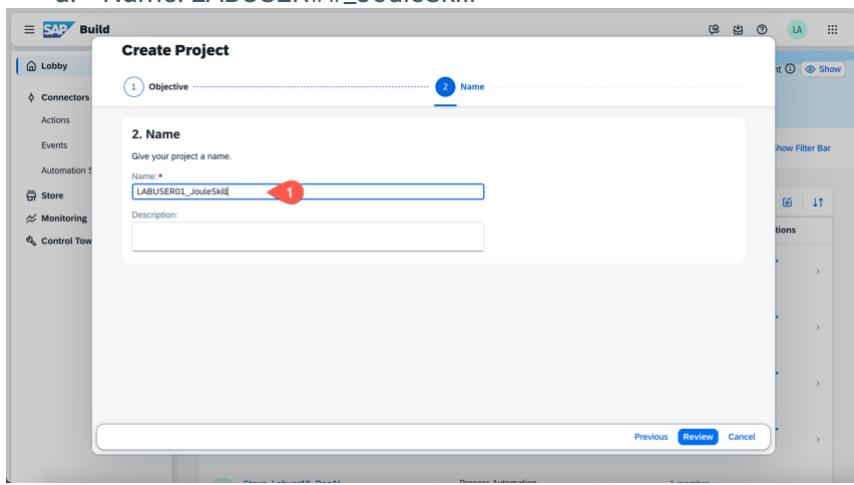
1. From the SAP Build Lobby click the Create button, then select Create



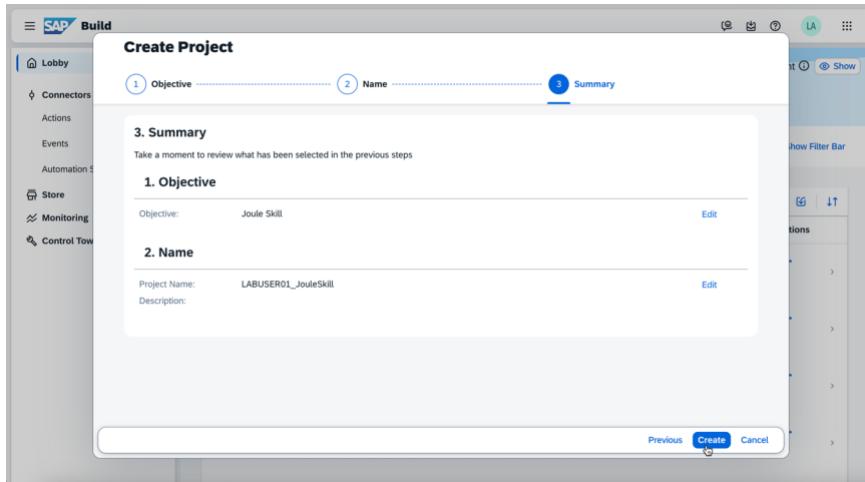
2. []
 3. Select Joule Skill



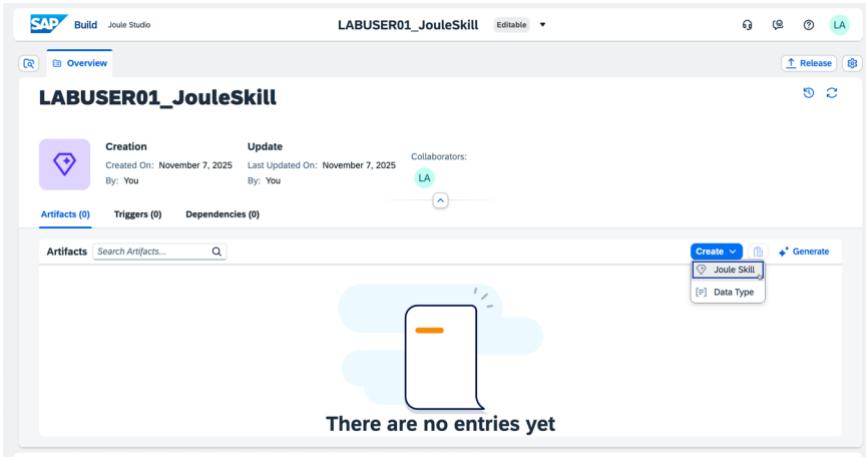
- 4.
5. Enter a name for your Joule Skill in the following format
a. Name: LABUSER##_JouleSkill



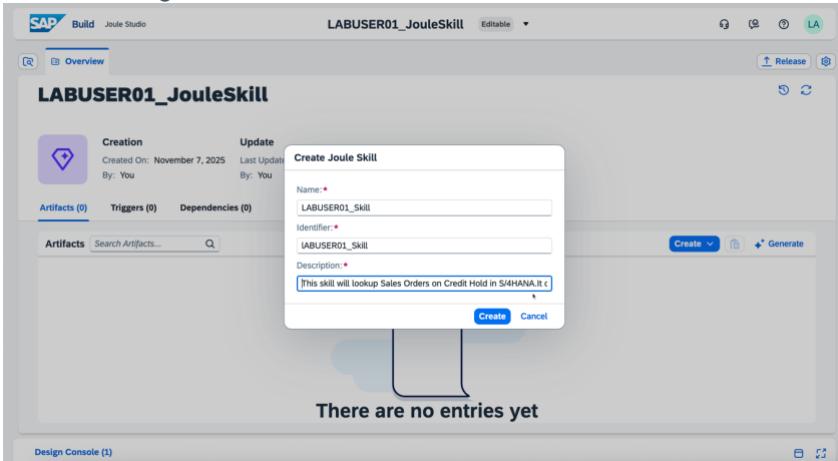
- 6.
7. Click Review. Then click Create



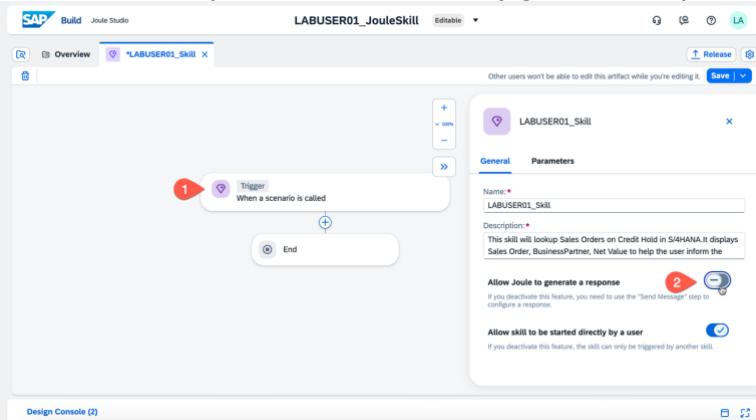
- 8.
9. A new browser tab will open (suppress the pop-up blocker)
10. Click Create > Joule Skill



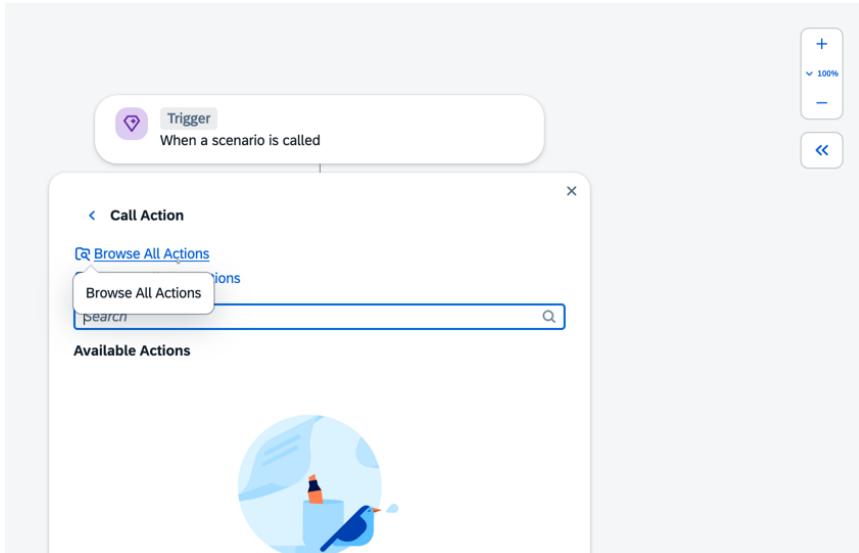
11. ..
12. On the *Create Joule Skill* modal, enter the following data
- Name: LABUSER##_Skill
 - Identifier: <Generated by SAP>
 - Description: This skill will look up Sales Orders on Credit Hold in S/4HANA. It displays Sales Order, Business Partner, Net Value to help the user inform the customer of their order status.
13. IMPORTANT: The 'Description' is interpreted by the LLM to understand the task to be completed. The 'Description' is very important and should be informative.
14. After entering the data above, click *Create*



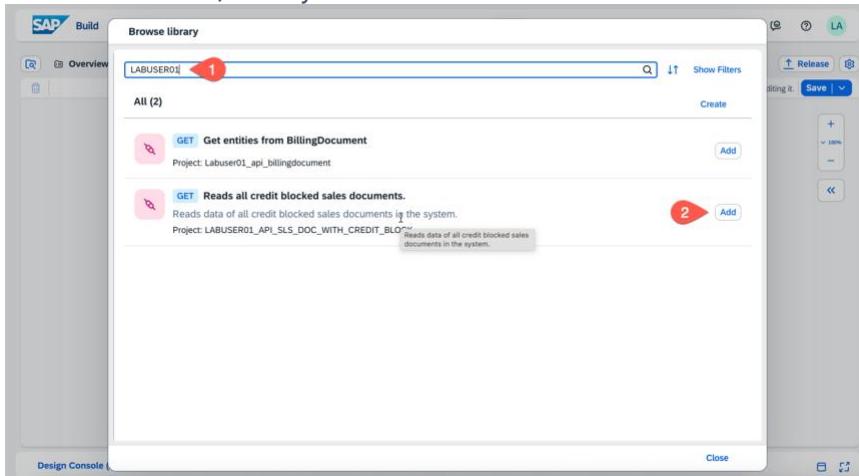
15. ..
16. When the canvas opens, select the Trigger and disable "Allow Joule to Generate a Response"
- We disable this option to more accurately guide the response from Joule.



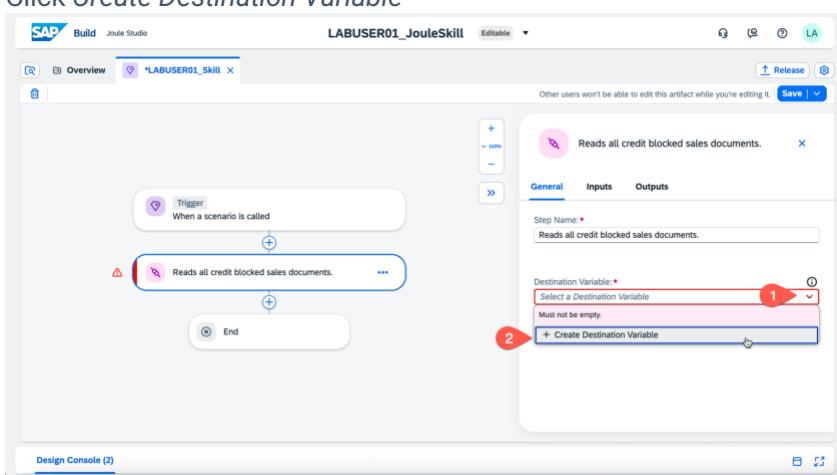
- Click the  to close the details
17. Click the plus sign under trigger to add an Action
18. Click Action
19. Click Browse All Actions



20.
21. In the search box, enter your LABUSER##



22.
23. Click Add next to the Action 'Reads all credit blocked documents'
24. After adding the Action, click the dropdown for Select a Destination Variable
25. Click Create Destination Variable



26.
27. Enter the Identifier and Description as shown below. Click Create

Create Destination Variable

The destination variable is created as an environment variable and can be managed in the project properties.

Identifier*

S4HANA

Description

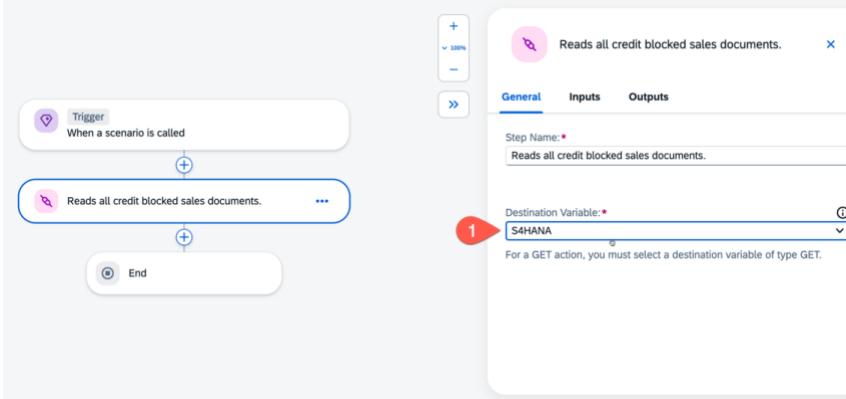
S4HANA System

Type*

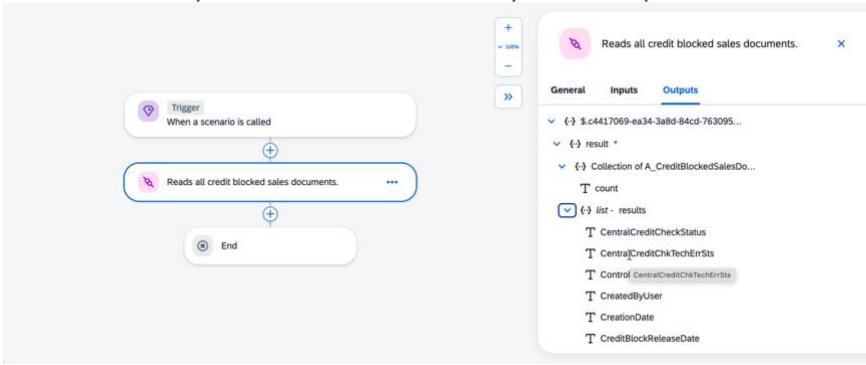
Destination

Create **Cancel**

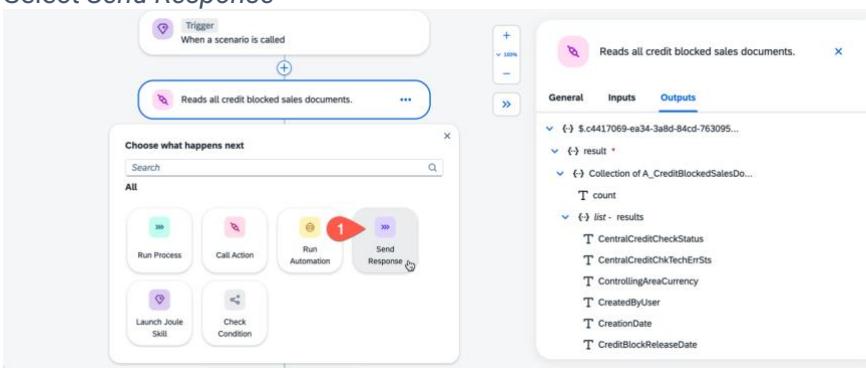
28.
29. Validate the Destination Variable is set correctly.



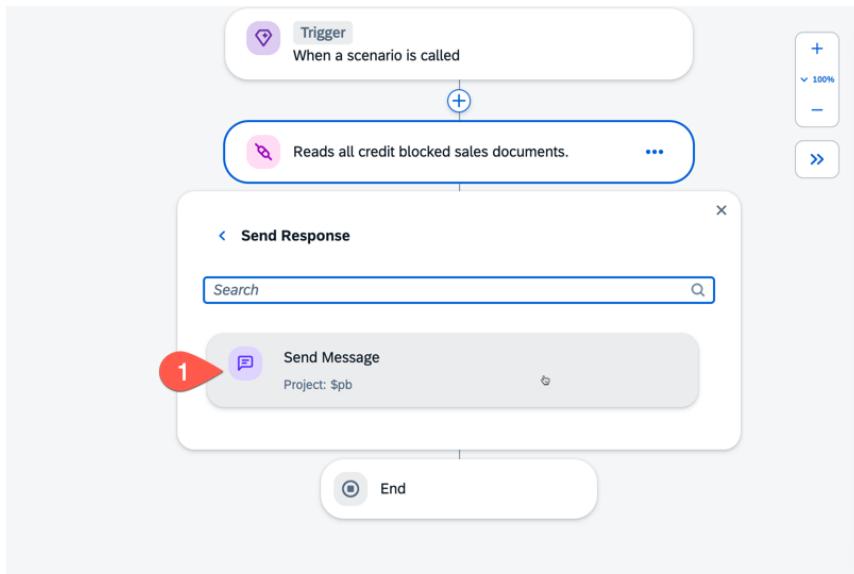
30.
31. Click on the *Outputs* tab to browse the expected output from the Action



32.
33. Return to the canvas and click the plus sign under the Action
34. Select Send Response

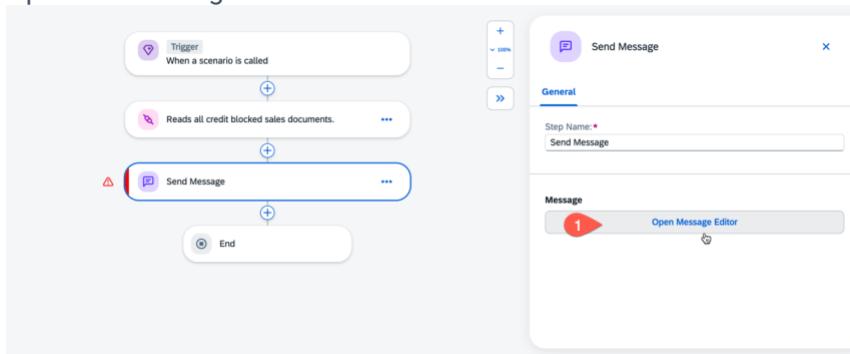


35.
36. Select Send Message



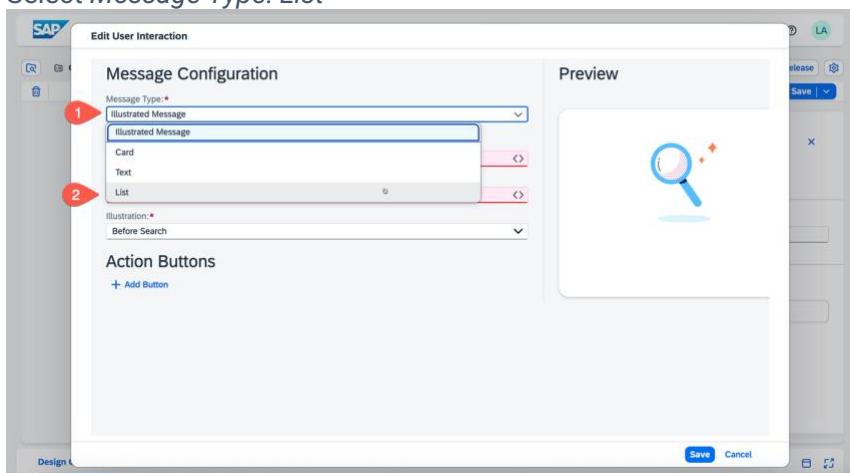
37.

38. Open the Message Editor



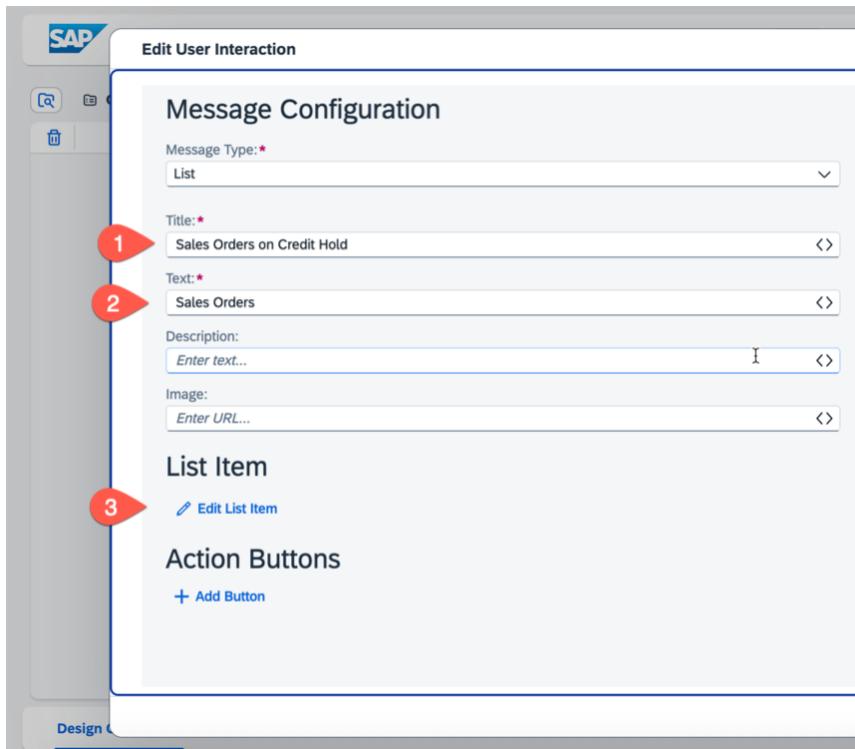
39.

40. Select Message Type: List

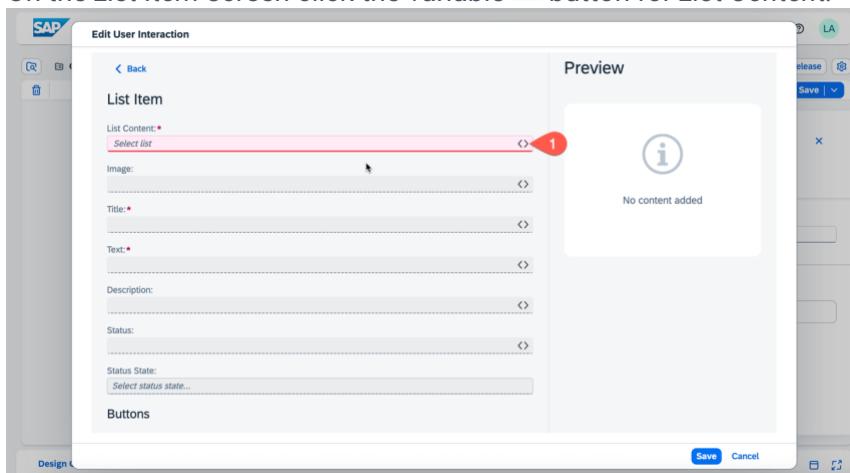


41.

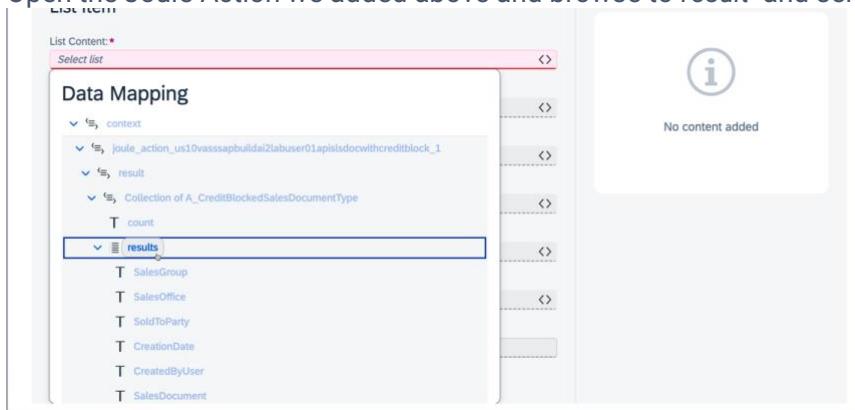
42. Enter the Title: Sales Orders on Credit Hold



43.

44. Select *Edit List Item*45. On the *List Item* screen click the variable $\langle\rangle$ button for *List Content*:

46.

47. Open the Joule Action we added above and browse to *result* and select it

48.

49. Enter the following data

a. Title: **Sales Order** and open the variable results to SalesDocument

1. Title: *
Sales Order \${context.joule_action_us10vasssapbuildai2labuser01apislsdocwithcreditblock_1.result}

2. SalesDocument

b. c. In the Text: field enter **Customer** and select the field **SoldToParty**

Customer:

d.

50. Scroll down to *Section Details*
 51. Enter the Title: *Order Details* and click *Add Attribute*

Sales Document \${context.joule_action_us10vasssapbuilddai2labuser01apislsdocwithcreditblock_1.r... <>}

Text:*
Customer: \${context.joule_action_us10vasssapbuilddai2labuser01apislsdocwithcreditblock_1.result.... <>}

Description:
Enter text... <>

Status:
Enter text... <>

Status State:
Select status state... <>

Buttons

+ Add Button

Detail View

Section Details

Title:
Order Details 1

Attributes

+ Add Attribute 2

52. Add two attributes

a. Label: Order Date
i. Value: Creation Date from Action T CreationDate

b. Label: Total Value
i. Value: TotalNetAmount from Action T TotalNetAmount

Status State:
Select status state... <>

Buttons

+ Add Button

Detail View

Section Details

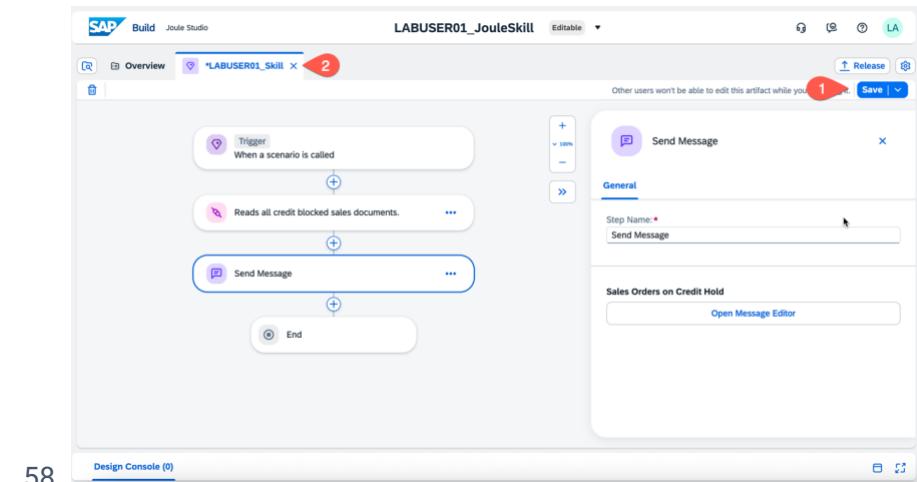
Title:
Order Details

Attributes

Type:*	Label:*	Value:
Text	Order Date	\${context.joule_action_u... <>} 
Type:*	Label:*	Value:
Text	Total Value	\${context.joule_action_u... <>} 

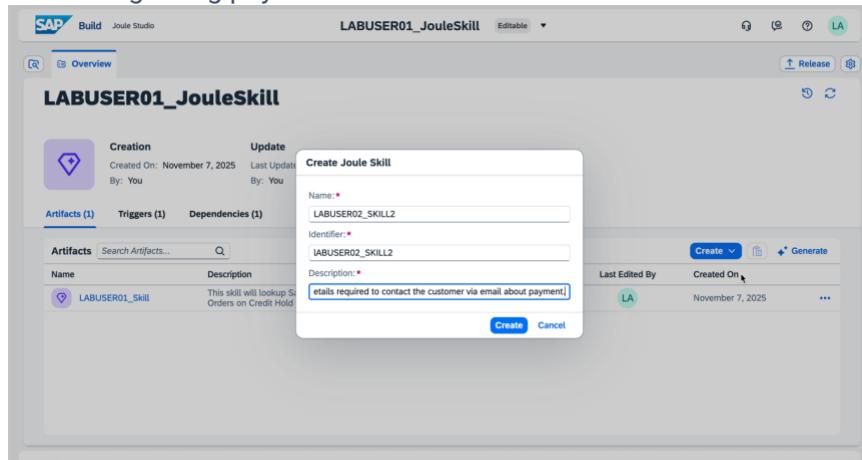
+ Add Attribute

- 54.
55. Click the Save button when complete
56. Click the Save button again on the canvas.
57. Close the canvas for this skill.

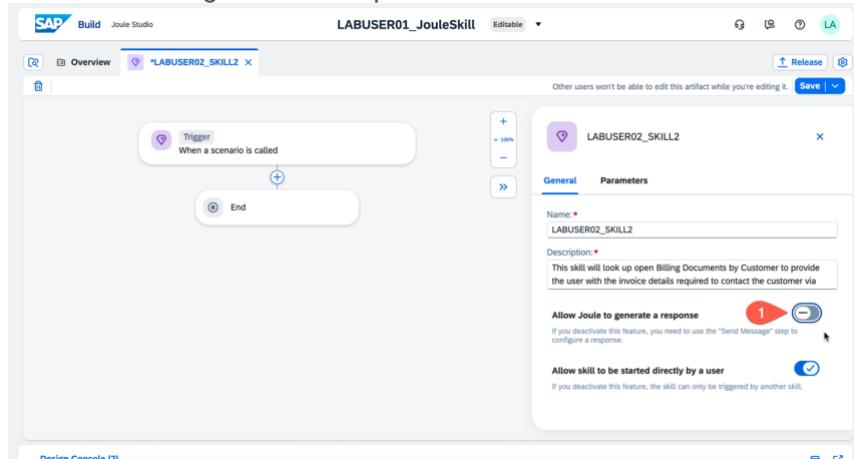


Next, we're going to create another skill to view open invoices. From the Overview, click *Create > Joule Skill* again

1. Enter the following details on Create Joule Skill
 - a. Name: LABUSER##_SKILL2
 - b. Description: This skill will look up open Billing Documents by Customer. Billing Documents are Invoices sent to the Customer awaiting payment. The output will provide the user with the invoice details required to contact the customer via email regarding payment.

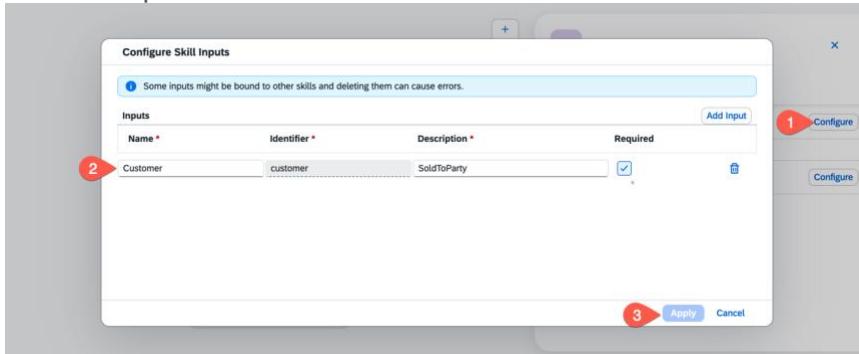


3. Disable Joule's generated response

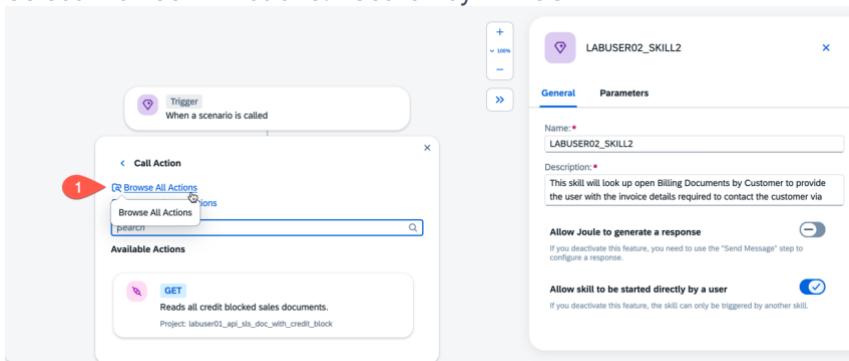


5. Click the Parameters Tab
6. Click Configure for Skill Inputs
7. Enter the following details

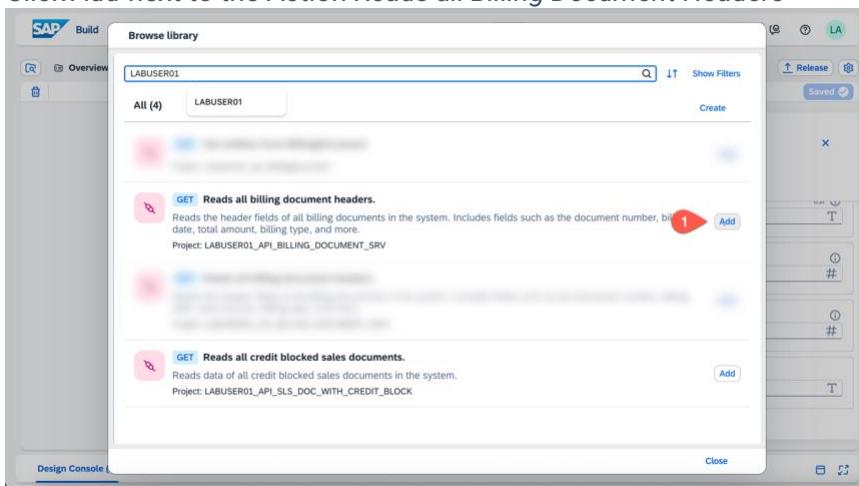
- Name: Customer
- Description: SoldToParty
- Required: Checked



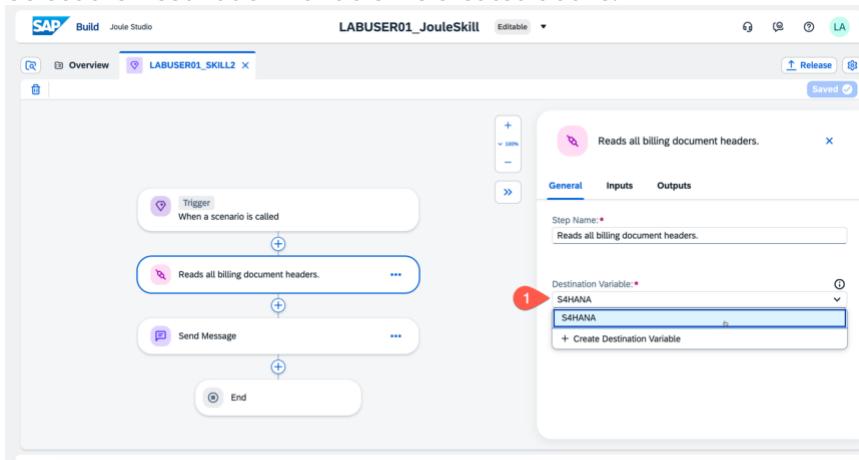
8.
9. Click Apply
10. Return to the canvas, Click the plus sign to add a Call Action under Trigger
11. Select Browse All Actions. Search by LABUSER##



12.
13. Click Add next to the Action Reads all Billing Document Headers

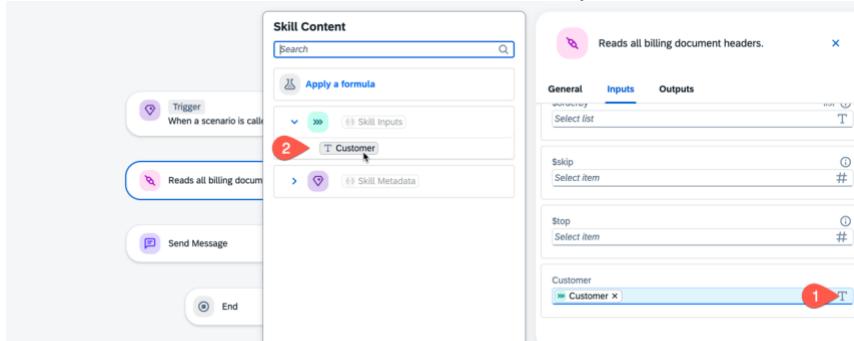


14.
15. Select the Destination Variable we created above.



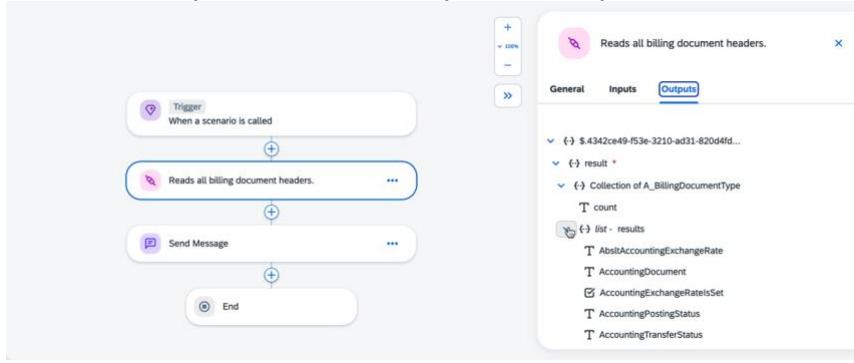
- 16.

17. Next, we're going to add the Input Variable
18. Click the Inputs tab, scroll down to 'Customer' (the field we added to the API call)
19. Click into the field and select the 'Customer' input variable we added.



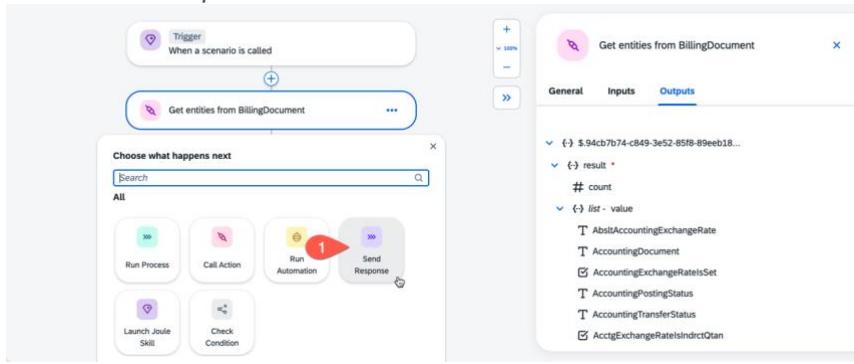
The screenshot shows the 'Skill Content' editor with the 'Inputs' tab selected. A red arrow points to the 'Customer' input variable in the list. The 'Customer' variable is highlighted with a red border.

- 20.
21. Browse the *Outputs* tab to review a potential response from the API



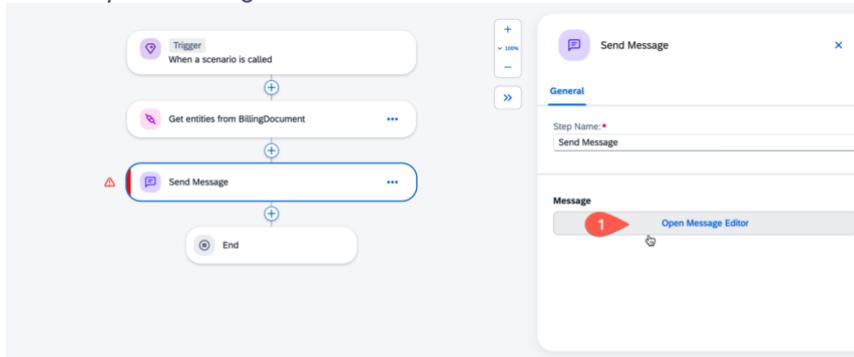
The screenshot shows the 'Skill Content' editor with the 'Outputs' tab selected. A red arrow points to the 'Customer' output variable in the list. The 'Customer' variable is highlighted with a red border.

- 22.
23. Click the plus sign under the Action we just added.
24. Select *Send Response*



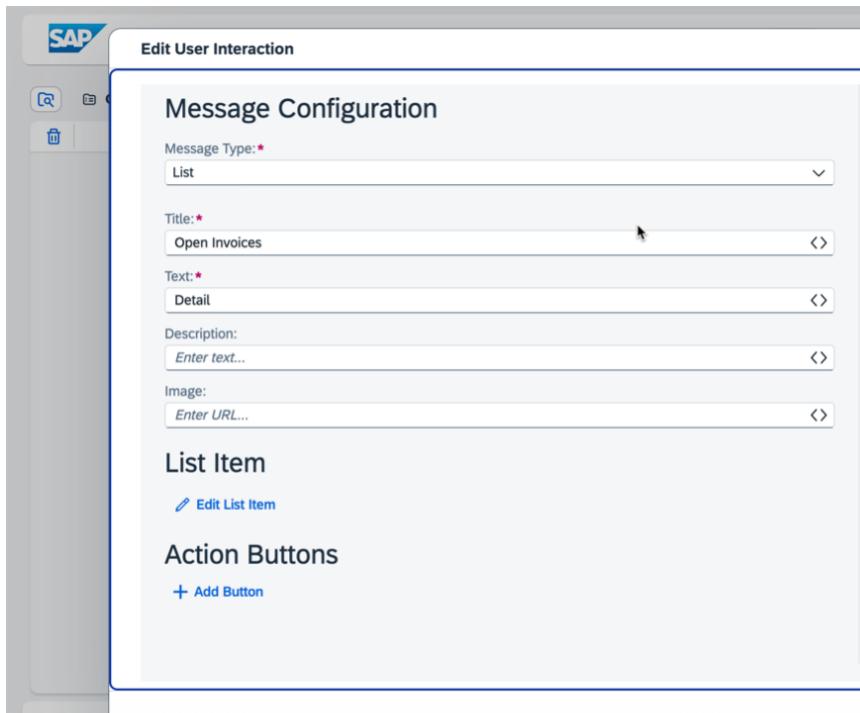
The screenshot shows the 'Skill Content' editor with the 'Outputs' tab selected. A red arrow points to the 'Customer' output variable in the list. The 'Customer' variable is highlighted with a red border.

- 25.
26. Select *Send Message*
27. Select *Open Message Editor*

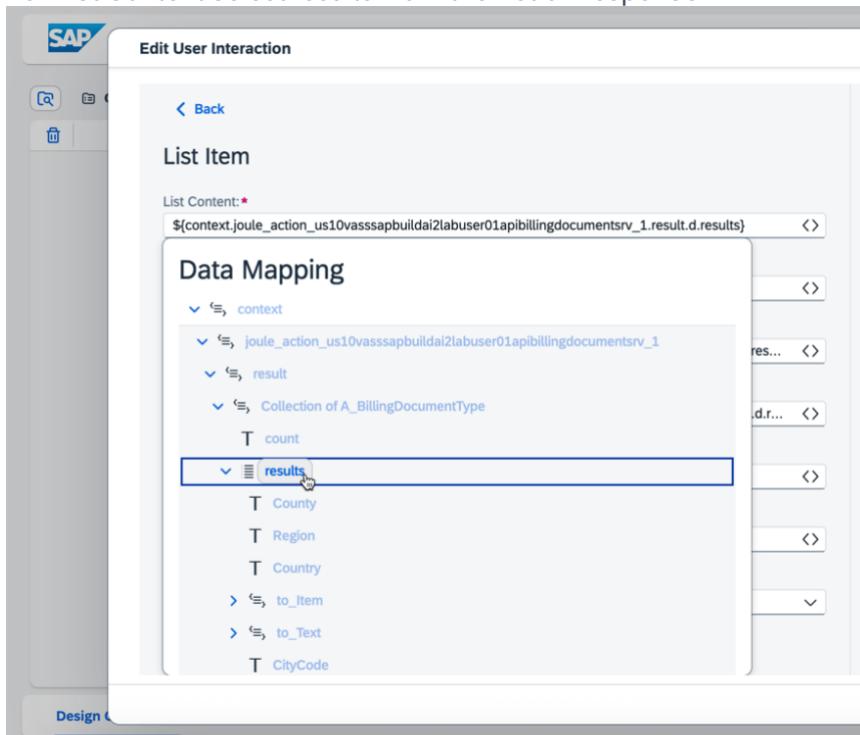


The screenshot shows the 'Skill Content' editor with the 'Outputs' tab selected. A red arrow points to the 'Customer' output variable in the list. The 'Customer' variable is highlighted with a red border.

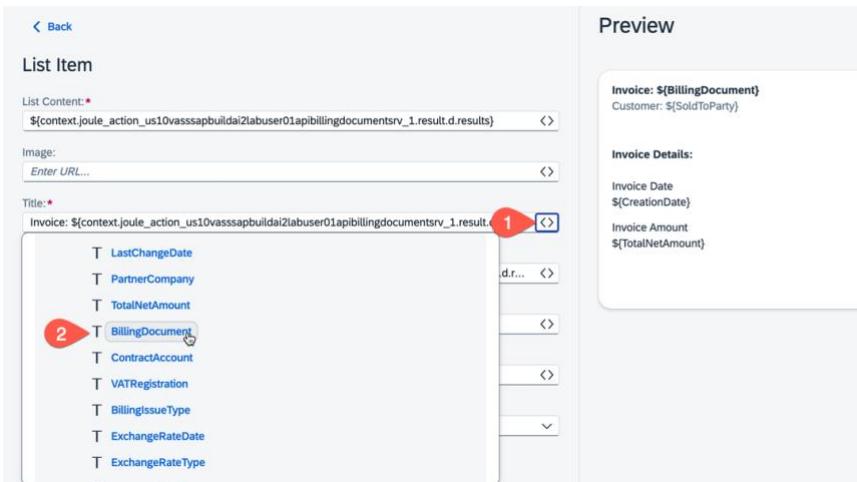
- 28.
29. Select Message Type: *List*
30. Enter Title: *Open Invoices*
31. Enter Text: *Detail*



- 32.
33. Click *Edit List Item* on screen
34. For List Content select *results* from the Action response

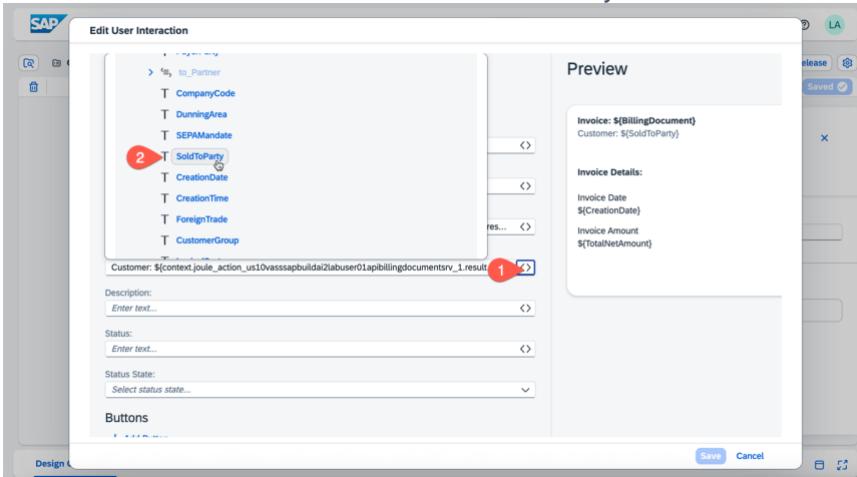


- 35.
36. Enter Title: *Invoice:* and select the BillingDocument value from Action response.



37.

38. Enter Text: *Customer:* and select the *SoldToParty* from the Action response

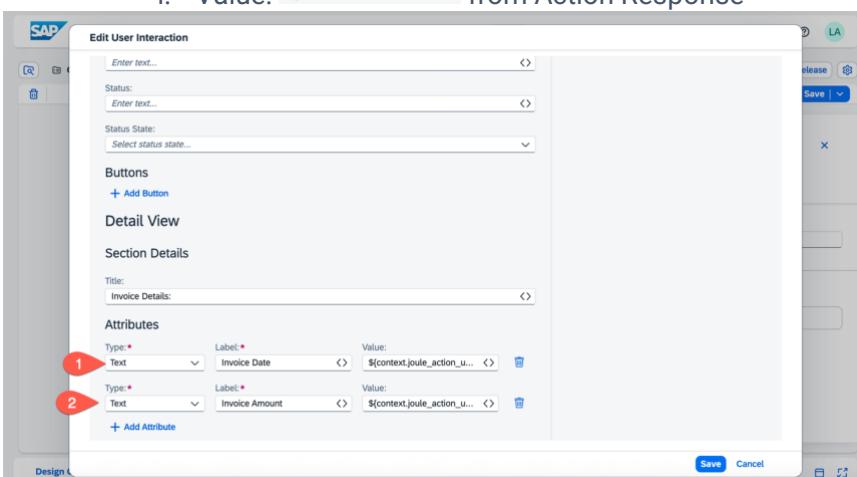


39.

40. Scroll down to Detail View

41. Add two Attributes

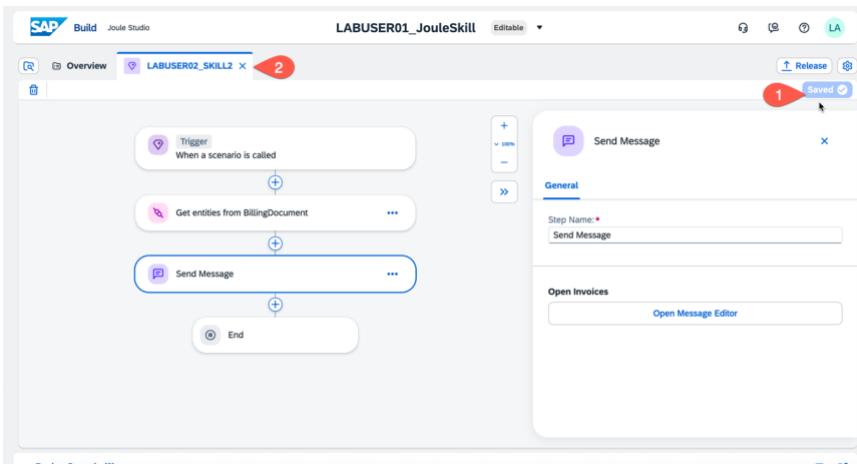
- Label: Invoice Date
 - Value: T **CreationDate** from Action Response
- Label: Invoice Amount
 - Value: T **TotalNetAmount** from Action Response



42.

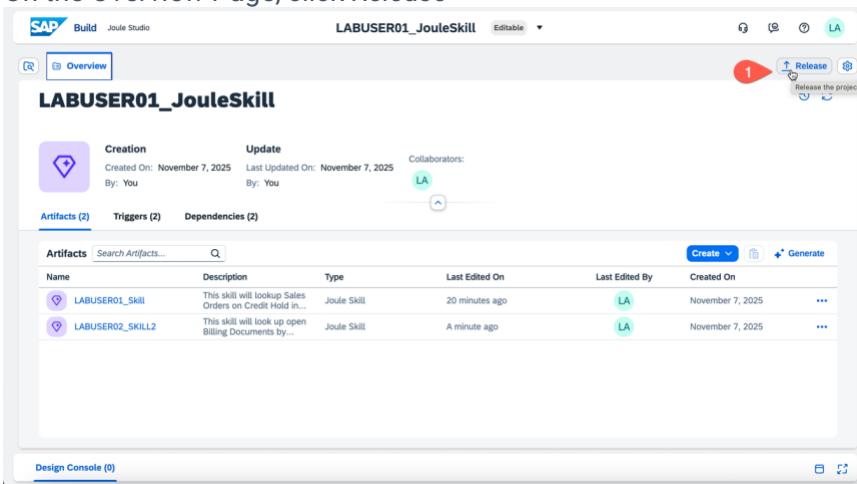
43. Click Save

44. Click Save again and close the Skill



45.

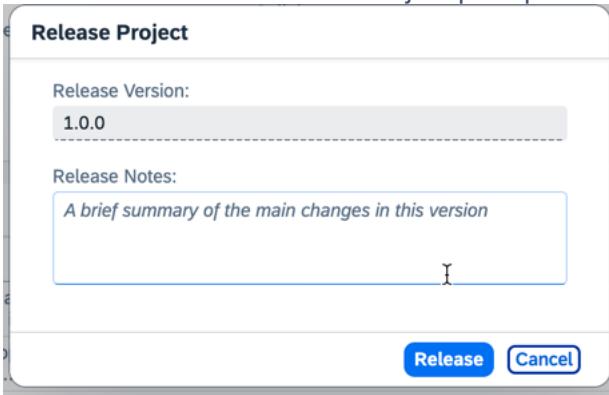
46. On the Overview Page, click Release



The screenshot shows the Overview page for the project LABUSER01_JouleSkill. It includes sections for Creation (Created On: November 7, 2025, By: You) and Update (Last Updated On: November 7, 2025, By: You). A red circle labeled '1' points to the 'Release' button in the top right corner.

47.

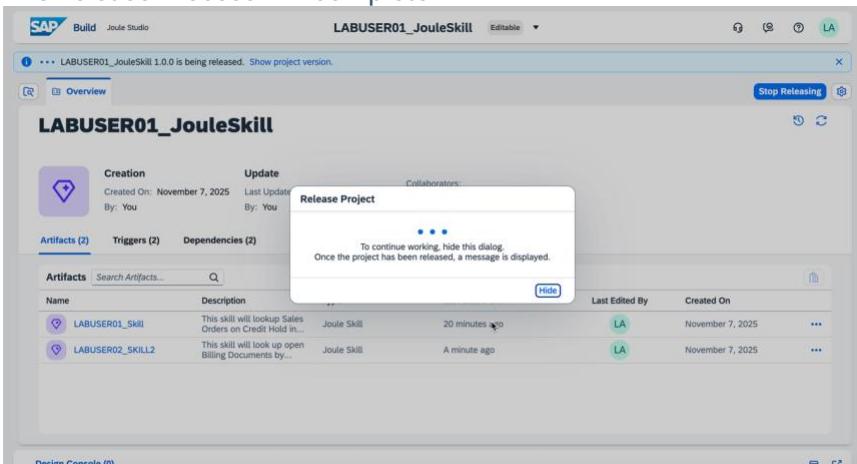
48. Click Release on the Release Project prompt



The screenshot shows the 'Release Project' dialog box. It contains fields for 'Release Version' (set to 1.0.0) and 'Release Notes' (containing the text 'A brief summary of the main changes in this version'). At the bottom are 'Release' and 'Cancel' buttons.

49.

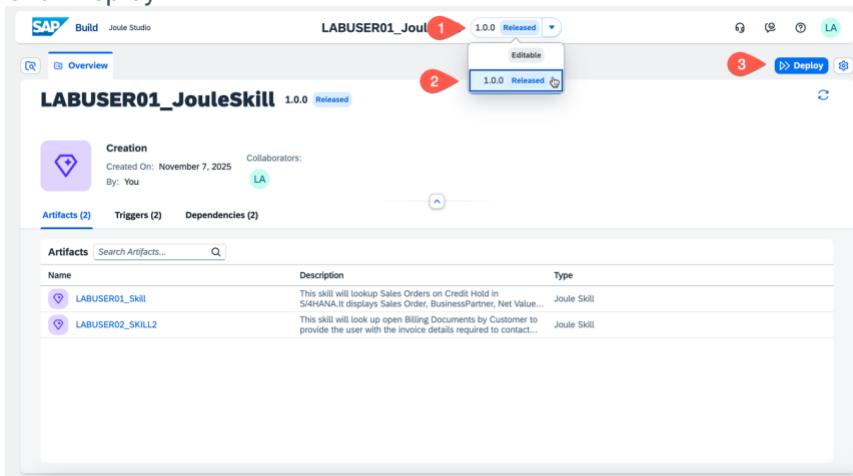
50. The Release Process will complete



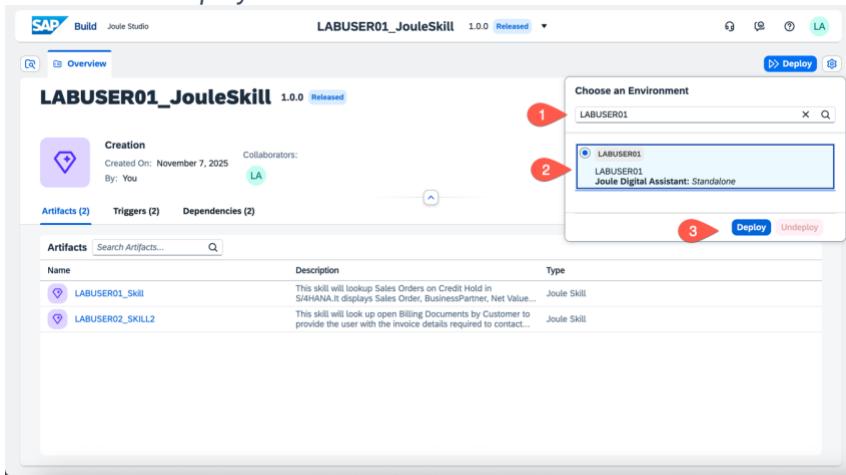
The screenshot shows the Overview page for the project LABUSER01_JouleSkill after the release process has completed. It includes sections for Creation (Created On: November 7, 2025, By: You) and Update (Last Updated On: November 7, 2025, By: You). A red circle labeled '1' points to the 'Stop Releasing' button in the top right corner. A message box is visible in the center of the screen.

51.

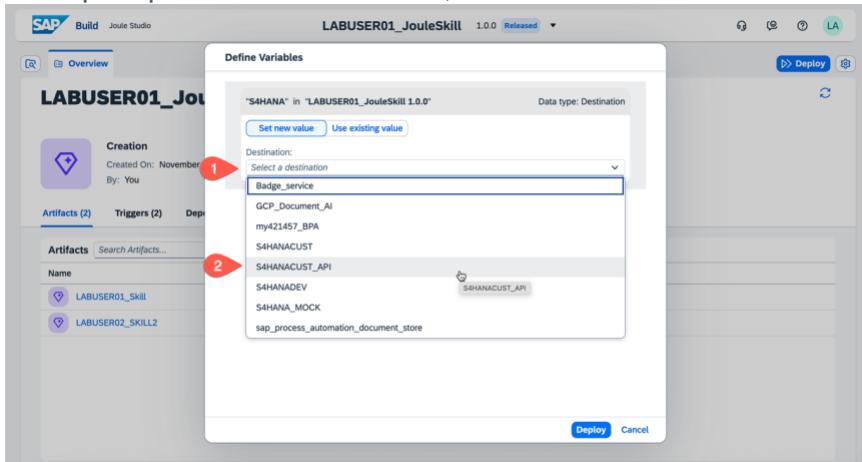
52. From the upper middle drop down, select your *Released* version
 53. Click Deploy



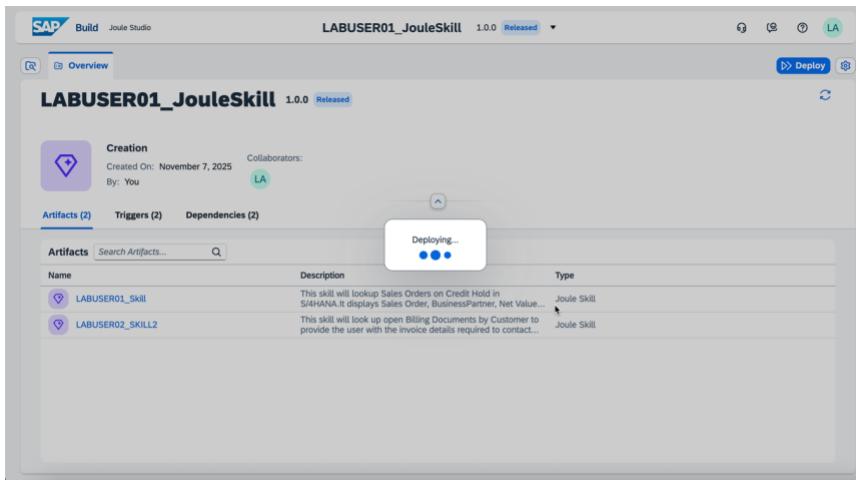
54. []
 55. Select the Environment for your LABUSER##
 a. Search *LABUSER##*
 b. Click Deploy



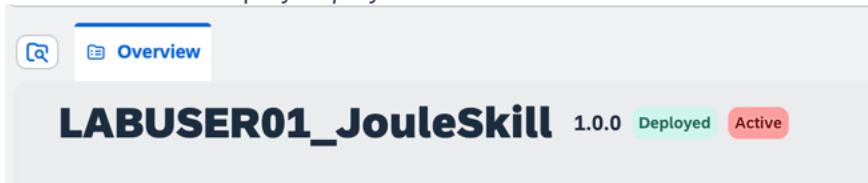
56. []
 57. When prompted to Define Variables, select the Destination: S4HANACUST_API



58. []
 59. Click Deploy after selecting the destination



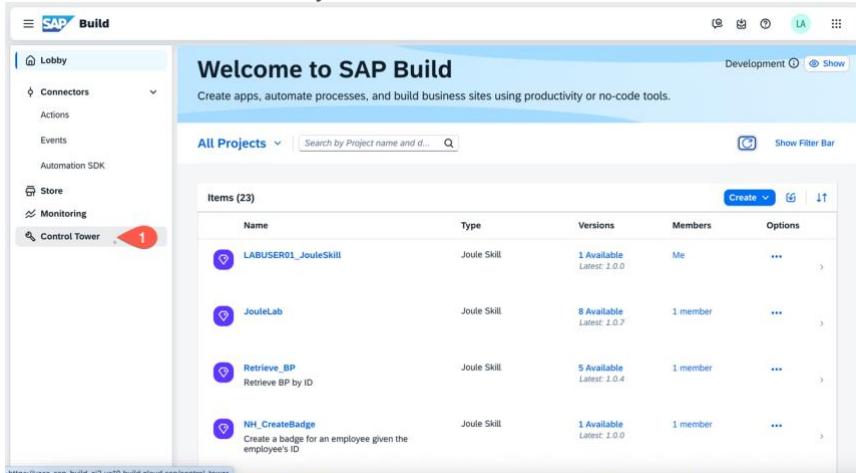
60.
61. The skill should display *Deployed* and *Active*



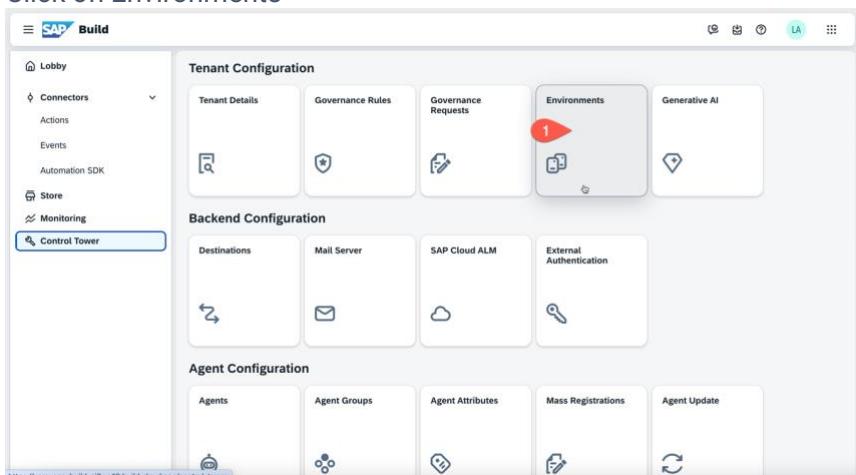
62.
63. Close this Browser Tab and return to the SAP Build Lobby

Test your Joule Skill

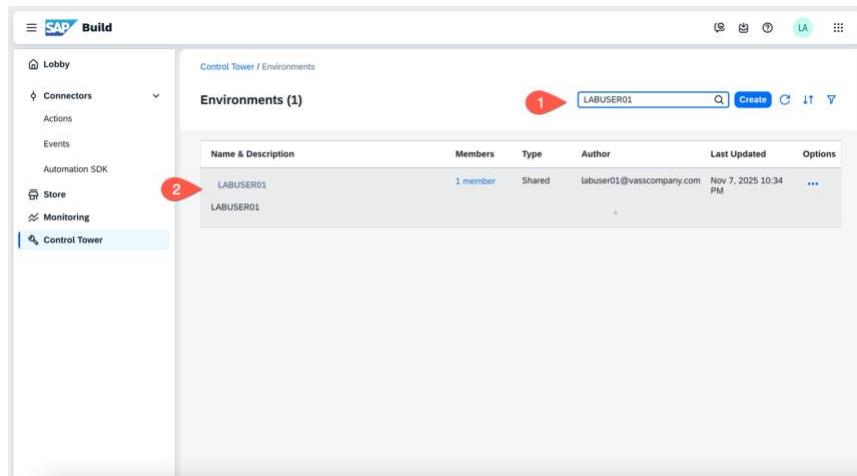
1. From the SAP Build Lobby, click Control Tower



2. <https://vass-sap-build-a12.us10.build.cloud.sap/control-tower>
3. Click on Environments

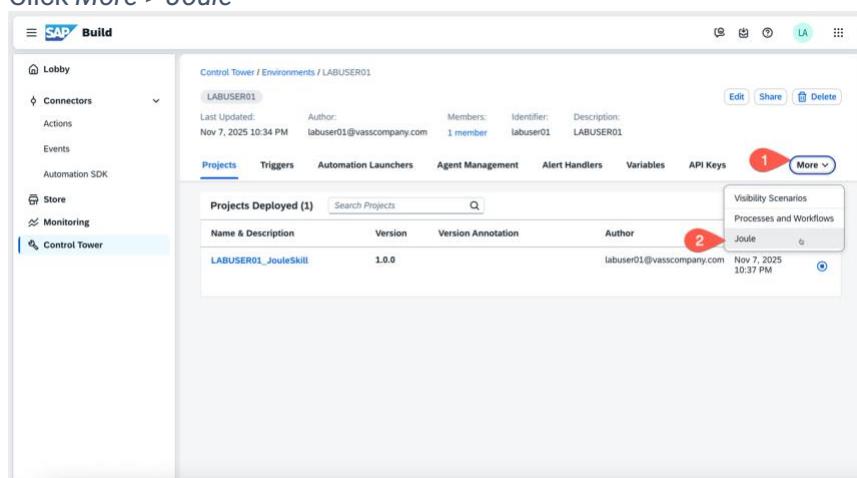


4. <https://vass-sap-build-a12.us10.build.cloud.sap/control-tower>
5. Search LABUSER## in the Search field



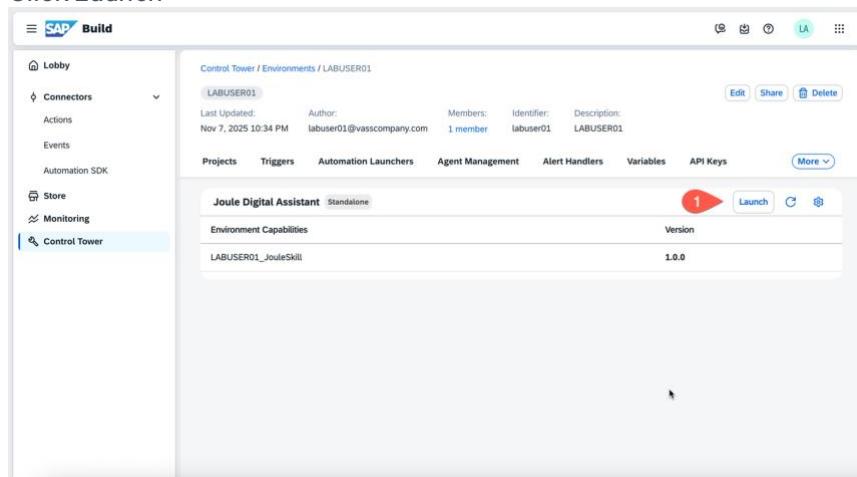
The screenshot shows the SAP Build interface under the Control Tower. In the left sidebar, 'Control Tower' is selected. The main area displays the 'Environments' section with one environment named 'LABUSER01'. A red arrow labeled '1' points to the search bar at the top right of the table, and another red arrow labeled '2' points to the 'More' button in the top right corner of the table header.

- 6.
7. Select your Environment
8. Click *More > Joule*



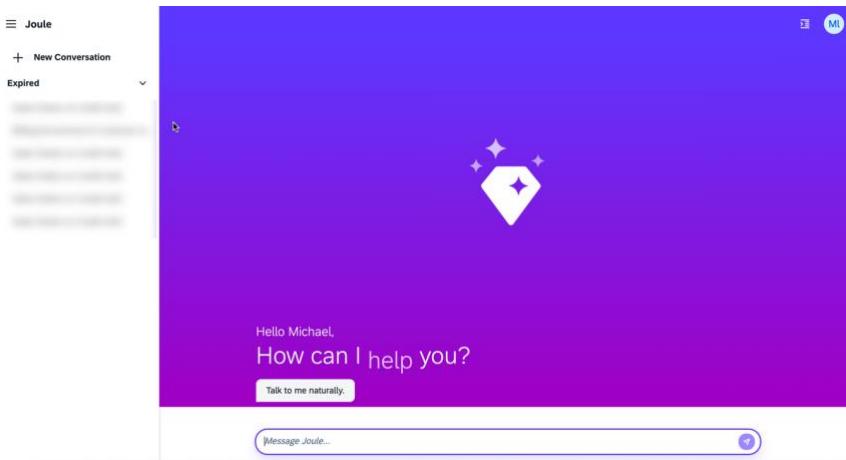
The screenshot shows the SAP Build interface under the Control Tower, specifically for the 'LABUSER01' environment. The 'Projects' tab is selected. A red arrow labeled '1' points to the 'More' button in the top right corner of the table header. A red arrow labeled '2' points to the 'Joule' tab in the dropdown menu.

- 9.
10. Click *Launch*

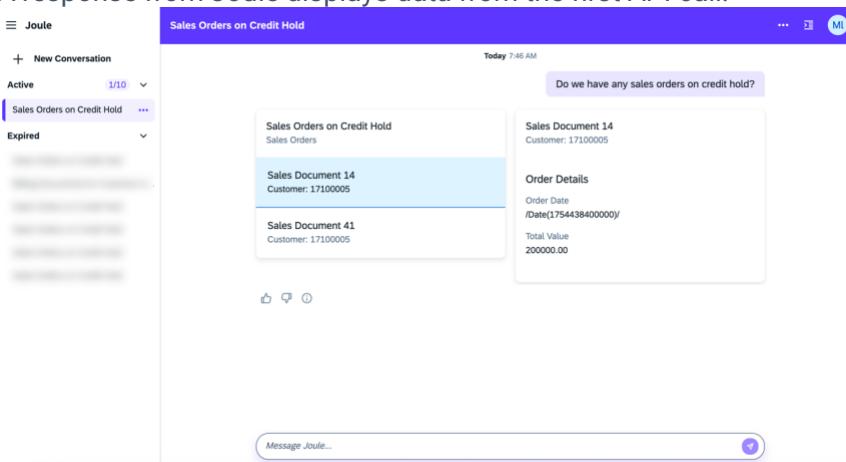


The screenshot shows the SAP Build interface under the Control Tower, specifically for the 'LABUSER01' environment. The 'Joule' tab is selected. A red arrow labeled '1' points to the 'Launch' button in the top right corner of the table header.

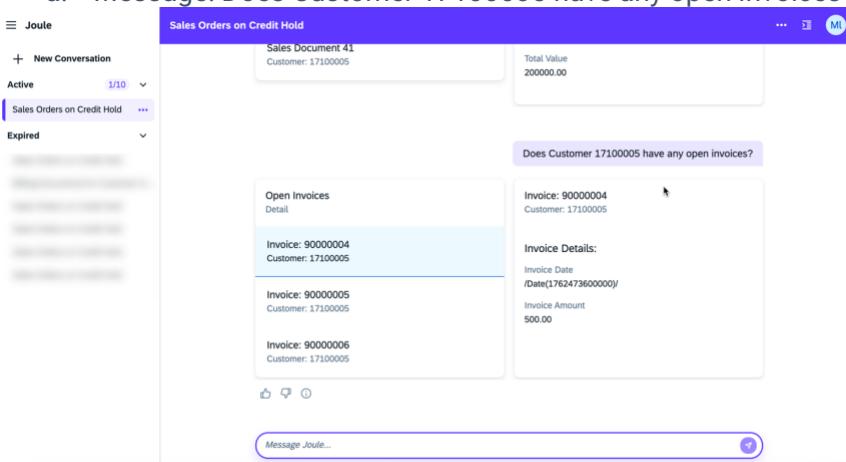
- 11.
12. A new browser tab will open (verify popup blocker)
13. A new conversation will be started. You can also click New Conversation to start a new dialog.



- 14.
15. Enter the following prompt for Joule
- Message: Do we have any sales orders on credit hold?
16. A response from Joule displays data from the first API call.



- 17.
18. Note the Order Date field. It's the unmodified field from the JSON response
- Trivia: Where would you fix that field?
 - Hint: It's in the Action configuration
19. Next, message Joule about any open invoices for the customer.
- Message: Does Customer 17100005 have any open invoices?



- 20.
21. You can click on an invoice to view details from the API response.
22. Next, we're going to ask Joule to draft an email asking the customer for payment.
- Message: Can you help me draft an email to the customer providing the open invoice details? And ask for a status on payment before Sales Order 41 can be released?
23. Validate the response provided by Joule.

The screenshot shows the Joule application interface. On the left, there's a sidebar with navigation options like 'New Conversation', 'Active' (1/10), and a selected item 'Sales Orders on Credit Hold'. The main area is titled 'Sales Orders on Credit Hold' and contains a message template:

```

Can you help me draft an email to the customer providing the open invoice details? And ask for a status on payment before Sales Order 41 can be released?

Subject: Follow-Up on Open Invoices and Sales Order Release

Dear [Customer Contact Name].  

I hope this message finds you well.  

I am writing to follow up regarding the open invoices associated with your account (Customer ID: 17100005). Below are the details of the outstanding invoices:  


- Invoice Number: 90000004  
Invoice Date: [Insert Date]  
Amount Due: $500.00
- Invoice Number: 90000005  
Invoice Date: [Insert Date]  
Amount Due: $200.00
- Invoice Number: 90000006  
Invoice Date: [Insert Date]  
Amount Due: $300.00


As per our records, these invoices remain unpaid. Kindly confirm the status of the payments at your earliest convenience. Please note that Sales Order 41 (Total Value: $400.00) is currently on hold and will be released.

Message Joule...

```

At the bottom of the message, it says 'Joule uses AI. Verify results.'

24.

Where can we go from here?

1. Enrich the responses with more data, like Customer Name, Contact Name!
2. Resolve the date display issue in Joule response.
3. Link this Skill to an Automation in SAP Build to send emails!
4. For example, we could automate this skill to draft emails at the beginning of every week
5. Add buttons with links to S/4HANA Fiori Apps for Invoices or Sales Orders
6. Retrieve PDF copies of Invoices and attach to an email
7. And more!

Thank you for completing our Hands-On Lab! Please email if you have any questions!