

Exercise 1 : Creating Project in SAP Build Apps

1. Login to your **SAP BTP Global Account** and select the **sub-account**

The screenshot shows the SAP BTP Cockpit interface. At the top, it displays 'Global Account: VASPP GmbH – Account Explorer' with a note 'All: 0 directories, 1 subaccounts | Subdomain: vasppgmbh'. Below this, there's a search bar and a dropdown for 'Regions'. The main area is titled 'Directories and Subaccounts' and 'Subaccounts (1)'. A single subaccount, 'VASPP BTP Dev', is listed. It includes details: 'Provider: Amazon Web Services (AWS)', 'Region: Europe (Frankfurt)', and 'Environment: Multi-Environment'. A red box surrounds the entire 'Subaccounts (1)' section.

2. Within the sub-account you have selected under **Instances and Subscriptions** select **SAP Build APPS**

The screenshot shows the 'Subaccount: VASPP BTP Dev - Instances and Subscriptions' page. On the left, a sidebar has 'Instances and Subscriptions' selected with a red box. The main area shows a table for 'Subscriptions (1)'. It lists one application, 'SAP Build Apps', with a plan of 'free', changed on '3 Oct 2024', and a status of 'Subscribed'. A red box highlights the 'Subscriptions (1)' section.

3. You will be redirected to login page of SAP Build apps and there select the **custom IAS account**

The screenshot shows the SAP Build Apps login page. It features a welcome message 'Welcome to VASPP BTP Dev!' and a sign-in option 'or sign in with:'. Below this, there's a section for 'Default Identity Provider' with the URL 'axbxdl9i.accounts.ondemand.com' highlighted with a red box.

4. Once you login to SAP Build Apps lobby click on **create** option to create the project

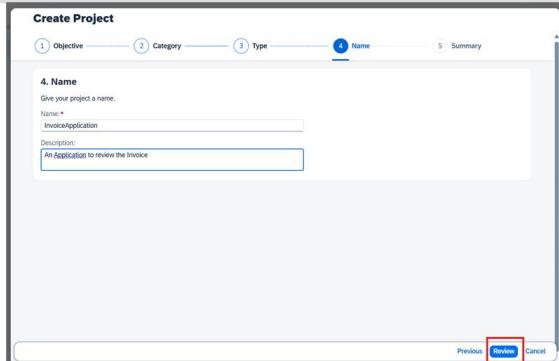
5. Click on Application and Next

6. Click on Front end and Next

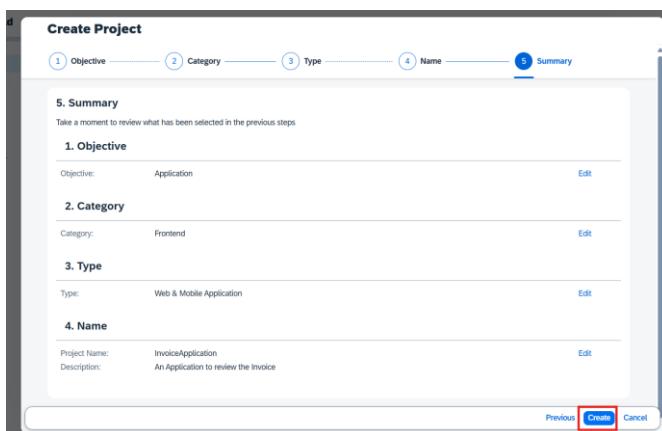
7. Click Web & Mobile Application and click Next.

8. Name the application and provide the description and click on review

Field	Value
Name	InvoiceApplication
Description	An Application to review the invoice



On the **summary** page click on **create**

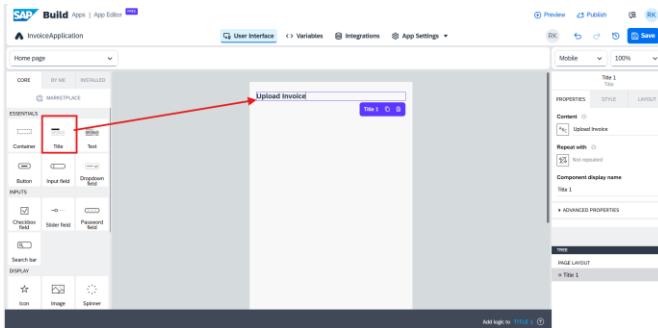


9. Now you have created the project, and you will be able to see the home Page

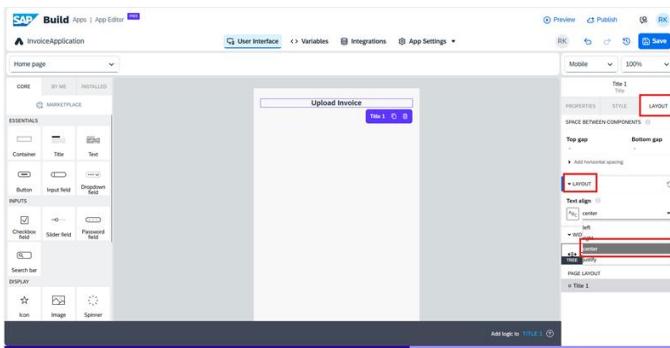
Exercise 2: Build Project in SAP Build Apps

Creating the Upload Invoice Page(Default Home Page)

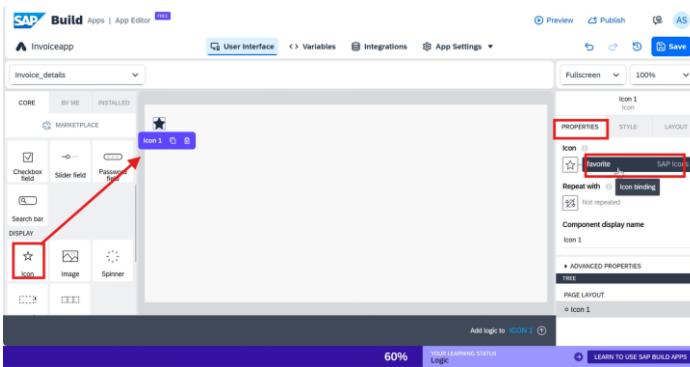
- Under core -> Essentials Drag and drop the **Title**, double click to edit the name as **Upload Invoice**



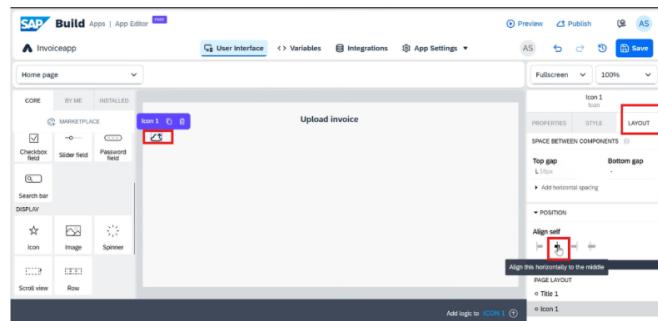
- Towards the right-hand side under layout text align it to centre



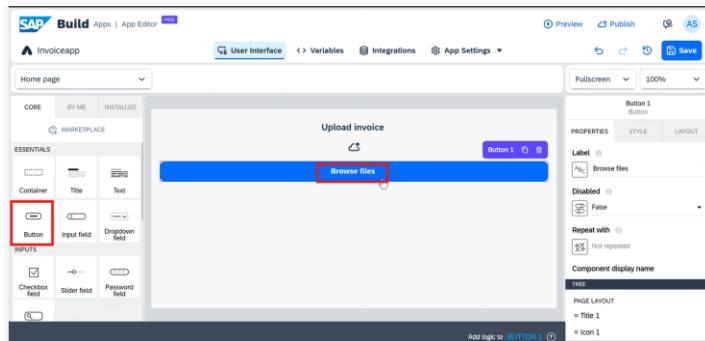
- Under core -> Display Drag and drop the icon, click on properties, under favourites search for upload icon.



Once you select, the upload icon under layout -> position -> align self -> centre



- Under core -> Essentials Drag and drop the **Button**, double click to edit the name as **Browse File**

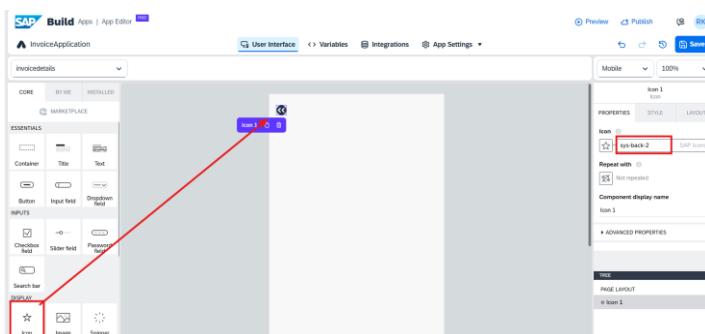


Creating Invoice Review Page

1. Towards the top left-hand side there is drop down displaying **Home page** click on it.
2. Select **add new page** and provide the **Invoice_details**

The top image shows the SAP Build interface with a dropdown menu open, highlighting the 'Add new page' option. The bottom image is a modal dialog titled 'Add new page' with a text input field containing 'Invoice_details' and an 'OK' button, which is also highlighted with a red box.

3. Under core -> Display Drag and drop the **icon**, click on properties, under favourites search for **back icon**.





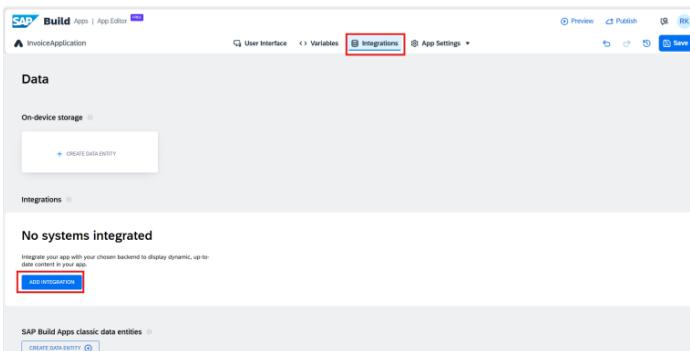
4. Under core -> Essentials Drag and drop the **Input Filed**, double click to edit the name as and provide the input labels and similarly copy/ duplicate for other input fields

The field names are: Vendor, Invoice Number, Invoice Date, Total Amount

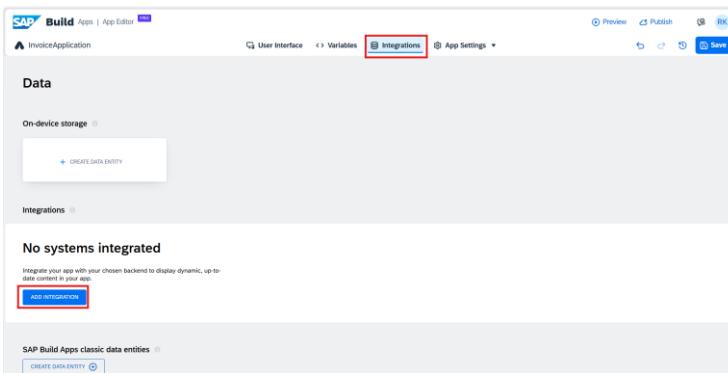
5. Under core -> Essentials Drag and drop the **Button**, double click to edit the name as **Send Mail**

Exercise 3: Integration of SAP Document AI

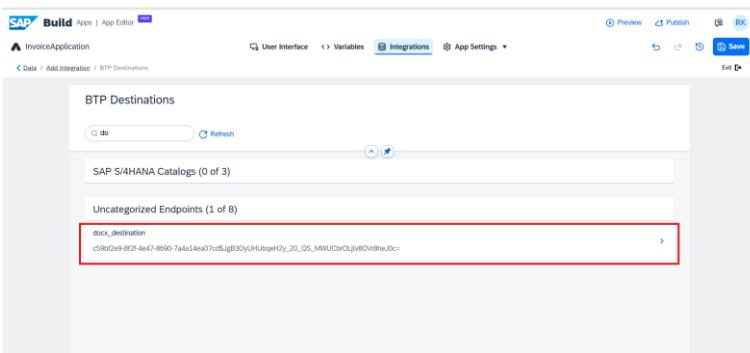
1. Select **Integration tab** and click on **Add Integration**



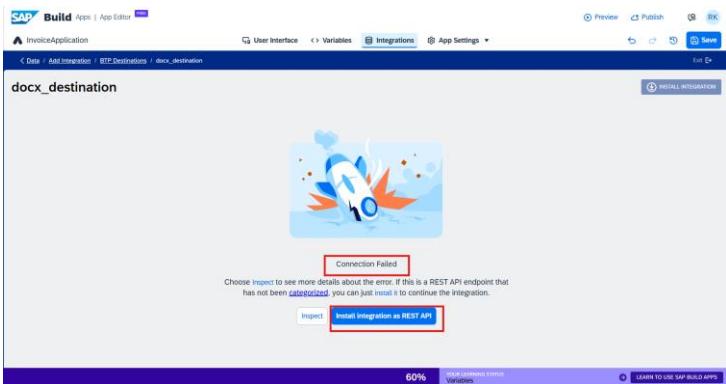
2. Under SAP System select SAP BTP Destinations



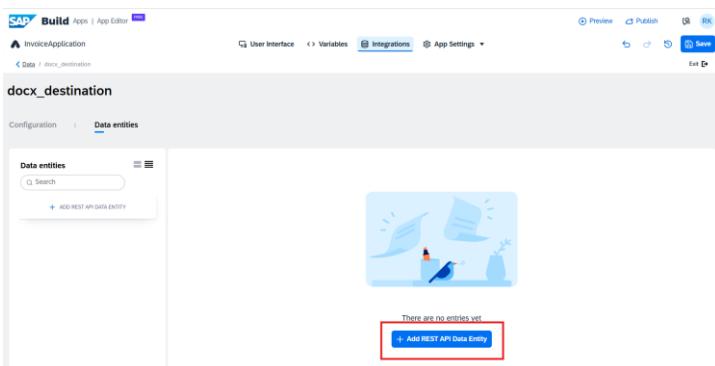
3. We can see many destinations that our created in out BTP Sub-account. Search for **Docx_Destination** select it and save



4. As soon as you click it below screen will be displayed, the connection failed is not an error, minimise the screen so you will be able to see the **Install integration a REST API** option click on it

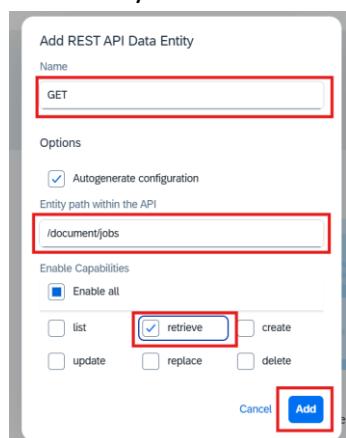


5. Click on + Add REST API Data Entity



6. We will get a pop up to file the below details

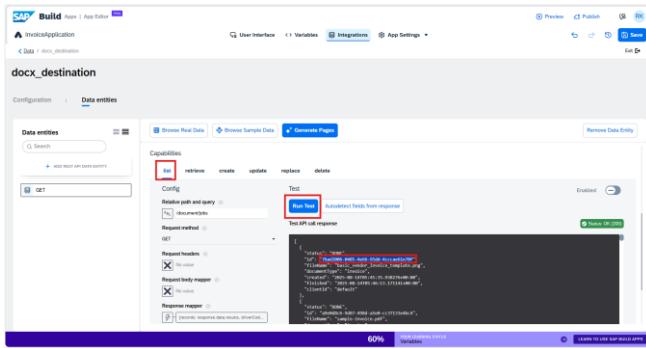
- Name: GET
- Check the Autogenerated configuration option
- Entity path within the API: /document/jobs
- Enable Capabilities: Check only retrieve option
- And finally click on ADD button



7. Once you save the details you will be able to see the updated screen, scroll down to test the API.

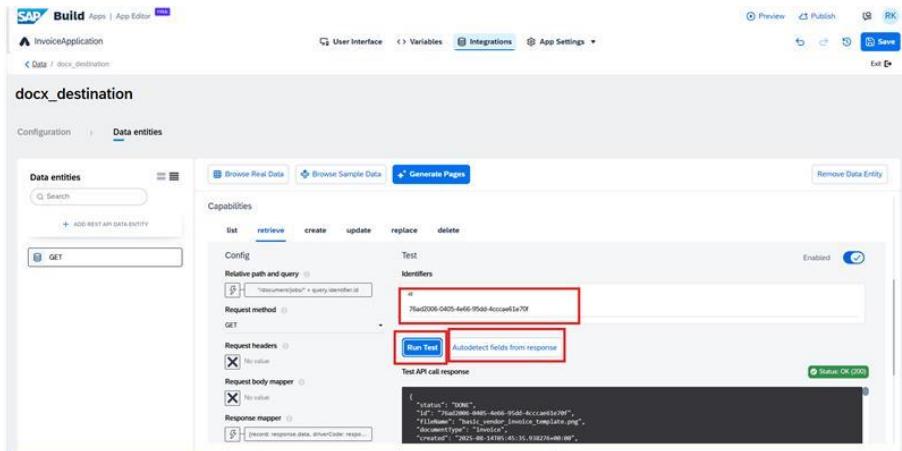
Under **list** click on **Run Test** to check the status and you will be able to see **Status Ok(200)** and the list of posted document's data.

Now copy the ID of one of the document data so we use it next step.



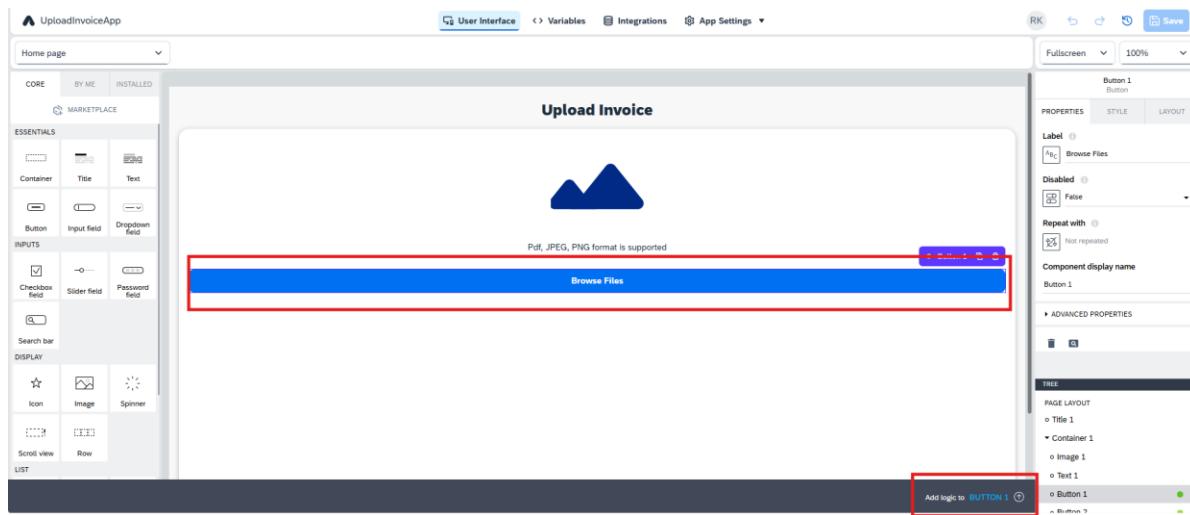
- Now go to **retrieve tab** paste the ID that you have copied previously under ID and click on Run test.

Once you get the run test result as **status Ok (200)** click on **Autodetect fields from response** and finally save it

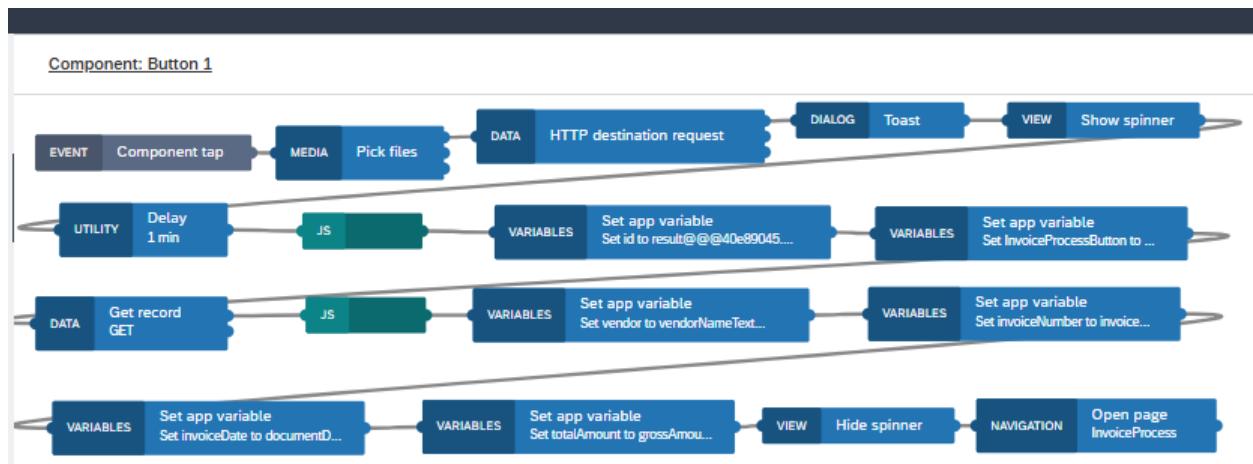


Exercise 4 - Integration of UI

Step 1: Click on Browse Files Button to add custom logic.

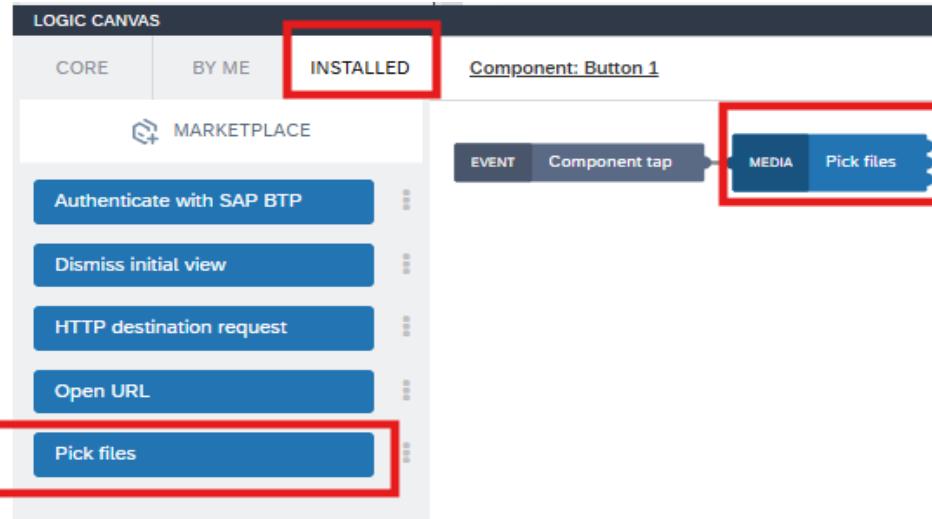


Step2: To Add logic please click on Logic editor with name “Button 1”

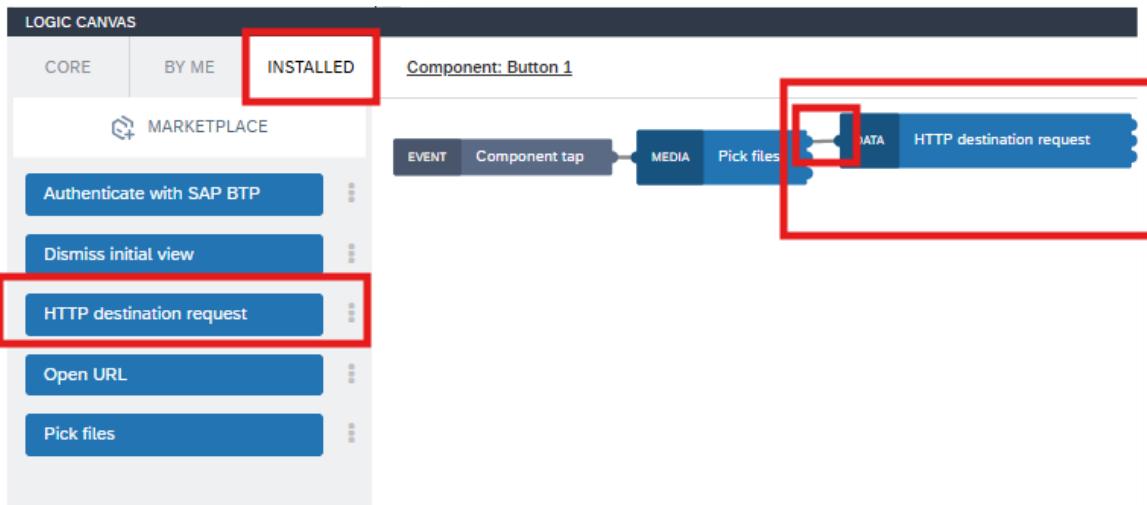


The above are the components I added for browsing File. Now Let us add one by one

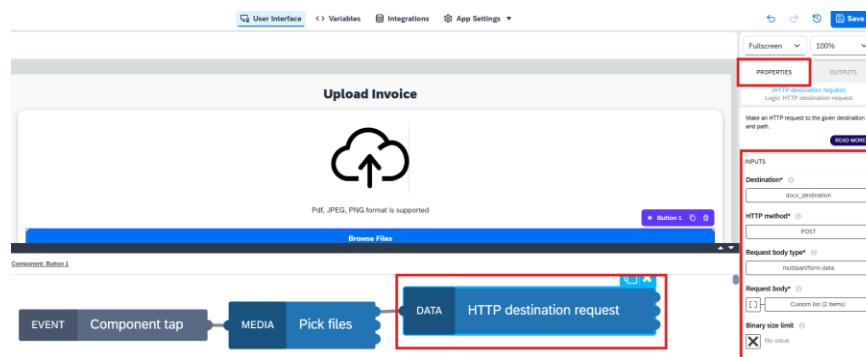
Step3: Under Logic Canvas, Go to marketplace and search for Pick Files install it and similarly install HTTP destination request and Open URL .In Logic Canvas (left side-bottom), go to Installed tab and select “**Pick Files**” and drop it in Logic editor.



Step4: Next drag and drop “HTTP destination request from Installed tab and connect it to the previous node “Pick Files”



Step5: Click on Component “HTTP destination request” and on right side you can view the Properties Panel.



Step6: Now we need to fill in the details, one by one.

The screenshot shows the SAP Build Apps interface for configuring an integration step. The top navigation bar has tabs for 'PROPERTIES' and 'OUTPUTS'. Below the tabs, the logic type is identified as '(HTTP destination request)' with the logic name 'HTTP destination request'. A descriptive note states: 'Make an HTTP request to the given destination and path.' A 'READ MORE' button is available for further details.

INPUTS

- Destination***: docx_destination
- HTTP method***: POST
- Request body type***: multipart/form-data
- Request body***: Custom list (2 items)
- Binary size limit**: No value

OPTIONAL INPUTS

- Path**: /document/jobs
- Headers**: Custom list (0 items)
- Response type**: text

Step7: Destination: On click of destination drop down you see the list already available which we added via integrations. Select the **DocX_destination from the drop down** which we created in SAP BTP Cockpit destination and add in integration in SAP Build Apps.

The screenshot shows the 'Destination*' input field. A red box highlights the dropdown menu, which lists 'docx_destination' as the selected item. Other options like 'docx_destination' are also visible in the list.

Step8: HTTP Method: Select the method "**POST**" from drop down as we upload the invoice to Document AI Service.

HTTP method*

POST
GET
POST
PUT
PATCH
DELETE

Step9: Request Body Type: select type “**multipart/form-data**”.

Request body type*

multipart/form-data
json
x-www-form-urlencoded
multipart/form-data
binary

Step10: Request Body: Click on Custom List a window opens.

Request body*

[]	Custom list (0 items)
-----	-----------------------

Step11: Click on Add Value

Edit binding for
HTTP destination request
list of objects with 2 properties

Select binding type / List of values

A list of items whose schema is defined in-place using the binding editor. A custom list can include both static items and ones bound to dynamic values.

Add a value

SAVE

Step12: Please add below values:

First Value:

Key: file

BACK

Edit binding for
HTTP destination request
list of objects with 2 properties

[Select binding type](#) / List of values

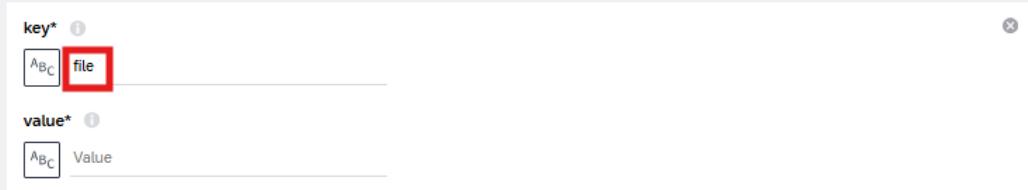
A list of items whose schema is defined in-place using the binding editor. A custom list can include both static items and ones bound to dynamic values.

key* ⓘ
 file

value* ⓘ
 Value

[Add another value](#)

SAVE



Step13: First Click on Value --> Select Formula --> Paste given Value

Code Snippet 1 **Value:**

```
MERGE([outputs["Pick files"].files[0], {"mimeType": outputs["Pick files"].files[0].mimeType}])
```

BACK

Edit binding for
HTTP destination request
list of objects with 2 properties

[Select binding type](#) / List of values

A list of items whose schema is defined in-place using the binding editor. A custom list can include both static items and ones bound to dynamic values.

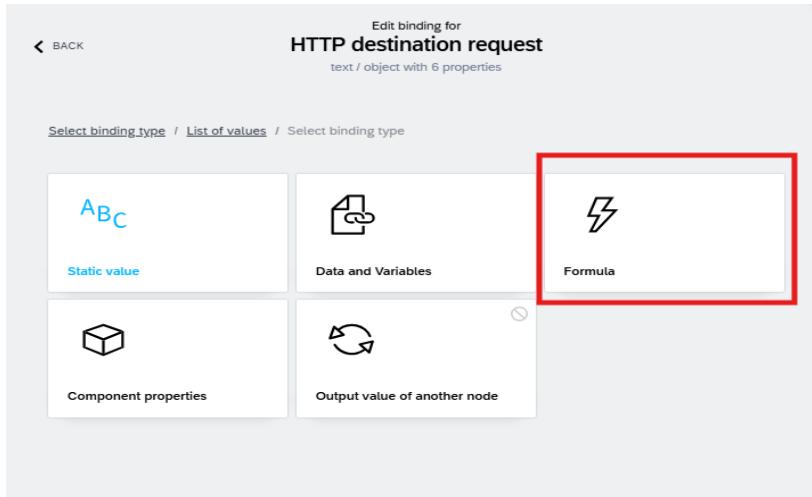
key* ⓘ
 file

value* ⓘ
 Value

[Add another value](#)

SAVE





Step 14: Now add another value just like the previous Step (First Click on Value --> Select Formula --> Paste given Value).

Second Value:

Key: options

Code Snippet 2 Value:

```
ENCODE_JSON({"schemaId":"cf8cc8a9-1eee-42d9-9a3e-507a61baac23","schemaVersion":"1","clientId":"default","documentType":"invoice","enrichment":{}})
```

Edit binding for
HTTP destination request
list of objects with 2 properties

Select binding type / List of values

A list of items whose schema is defined in-place using the binding editor. A custom list can include both static items and ones bound to dynamic values.

key* ABC file

value* MERGE([outputs["3ebc5d04.1d5cf"].files...])

key* ABC options

value* ENCODE_JSON({"schemaId": "cf8cc8a9-1...})

Add another value

SAVE

Step15: In Optional Inputs, give the **relative path** --> **/document/jobs**

▼ OPTIONAL INPUTS

Path ABC /document/jobs

Step16: Next drag and drop toast button from the core and click on ABC option in Properties.

Fullscreen 100%

Properties (Logic-Toast)

Shows a non-blocking toast dialog with the given message. The dialog will disappear automatically, unless otherwise configured.

INPUTS

Toast message* ABC value

LOGIC CANVAS

CORE BY ME INSTALLED MARKETPLACE

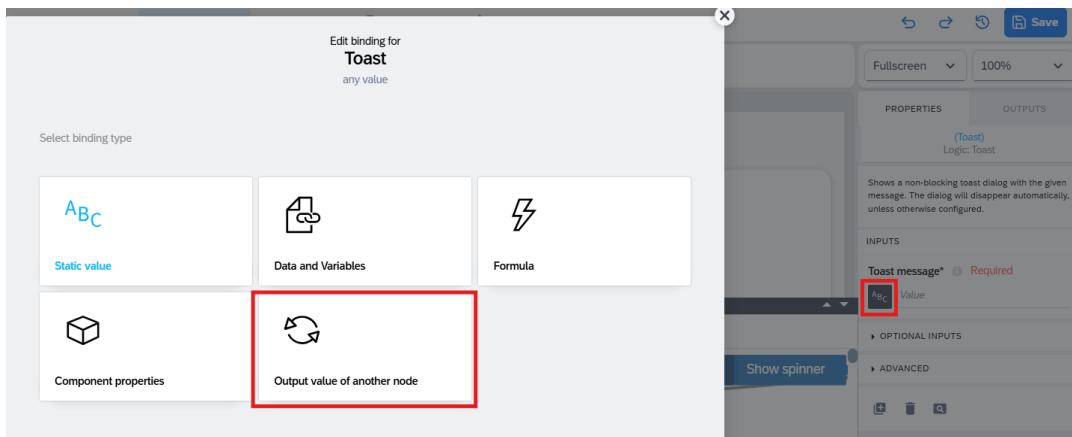
Component: Button 1

Upload Invoice

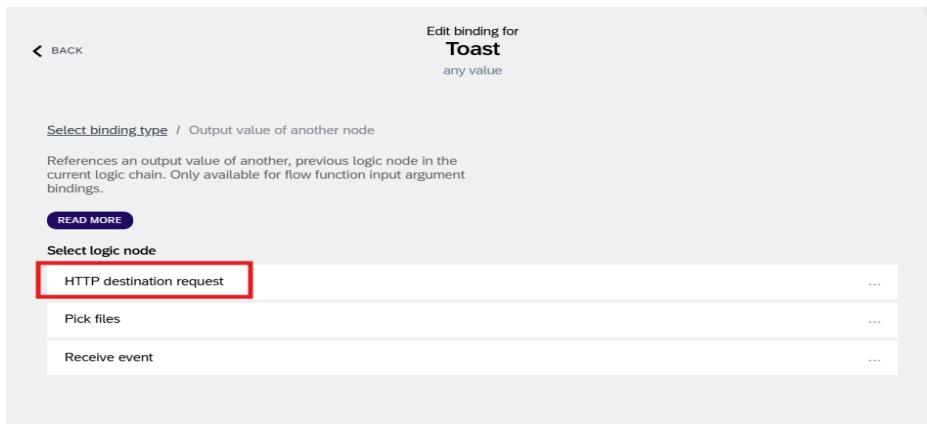
Cloud upload icon

Event: Component tap → Media: Pick files → Data: HTTP destination request → Dialog: Toast

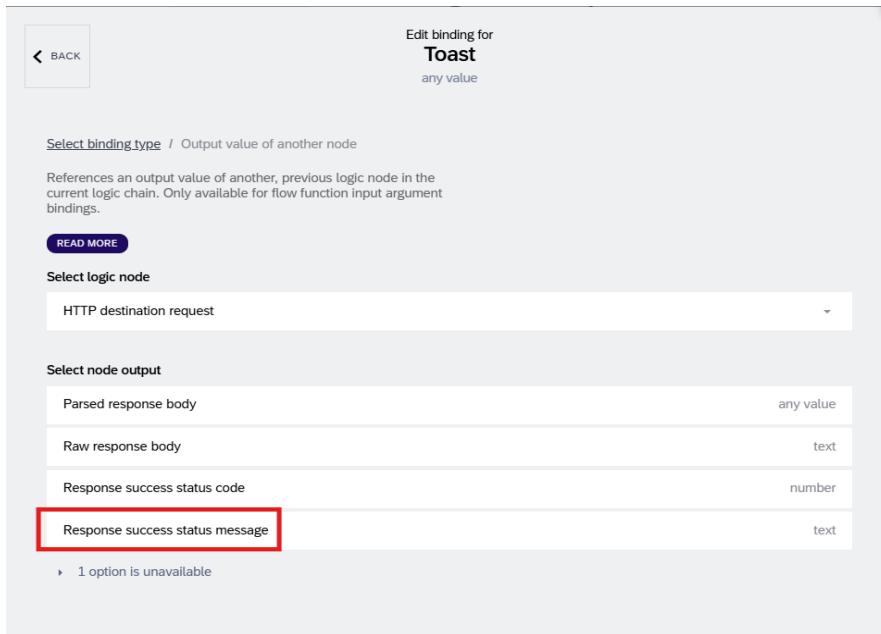
Step17: Click on Output Value of Another node



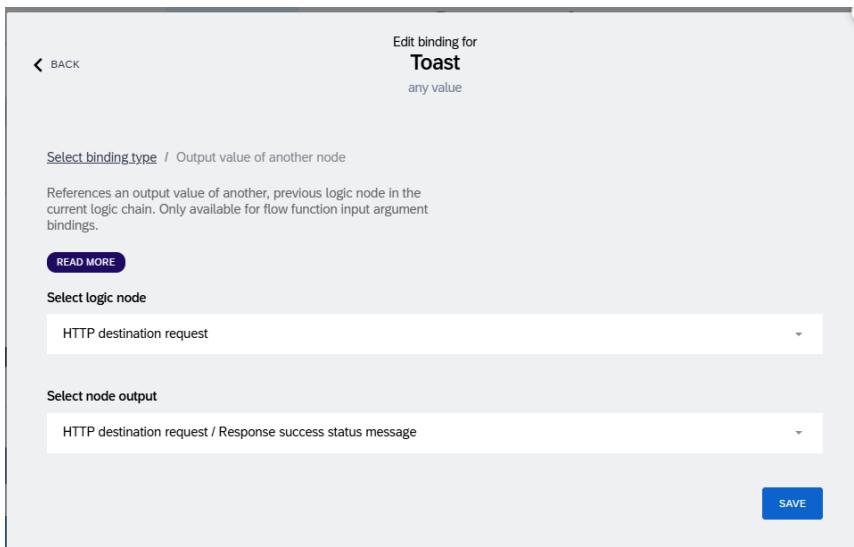
Step18: Click on HTTP destination request



Step19: Click on Response Status Messages. This will toast the success message in successful upload of Document to Document AI.

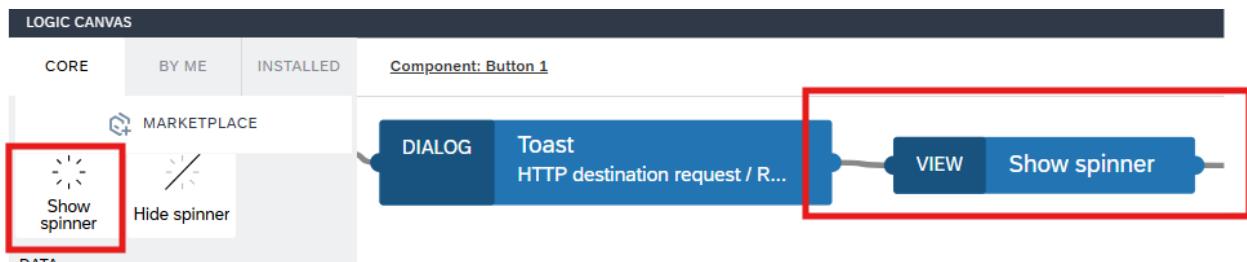


Step 20: Click Save



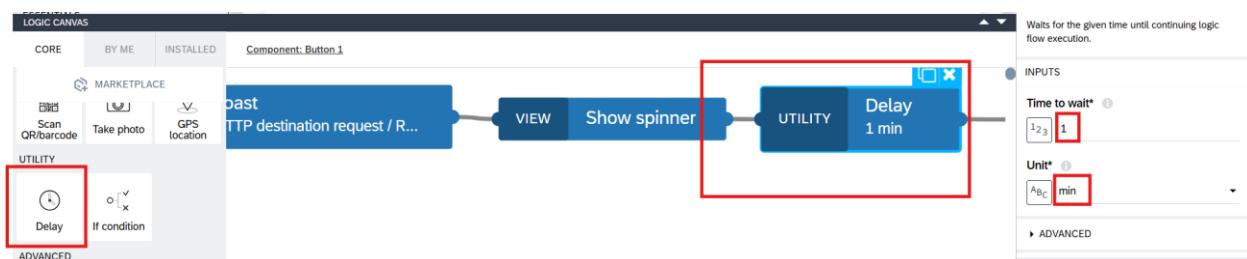
Step 21: In Core, select **show spinner** and connect it to the previous component:

Note: Since we're posting file content, to show process has started we're adding a spinner

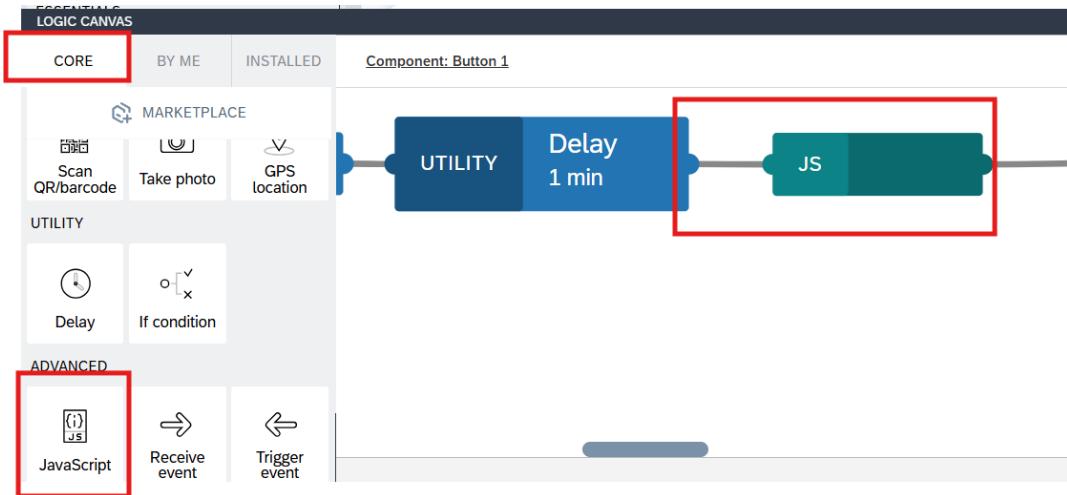


Step22: Now we add a Delay component connect to previous component. Now click on **delay Component** and add Properties for it.

Note: The reason why we need to add delay is , when we upload the file , the DocX service tries to get the content of it and during this time the status of file is "Pending" even though it is posted , so if we immediately look for file content we will not get the full content so this delay helps to get full file content as response.



Step 23: Now from Logic Canvas, select the **JavaScript component**. Drag it and drop it to the logic editor and connect it to the previous component.



Step24: Double click on JS component.



Step25: Now this dialog will open, and we must write the script task in it.

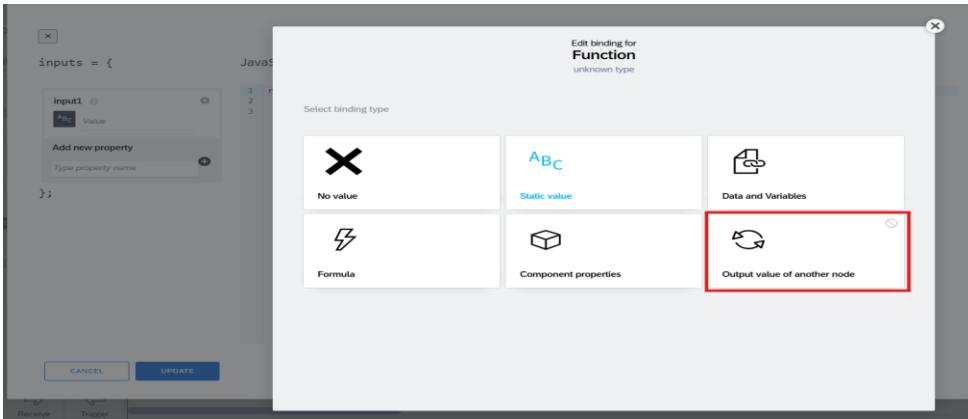
```


inputs = {
    input1: {
        Value: "ABC"
    }
};

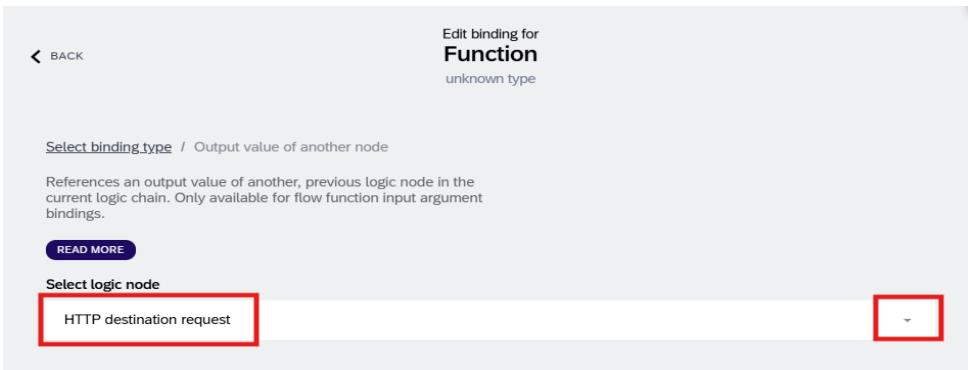
return { result: inputs.input1.id };

```

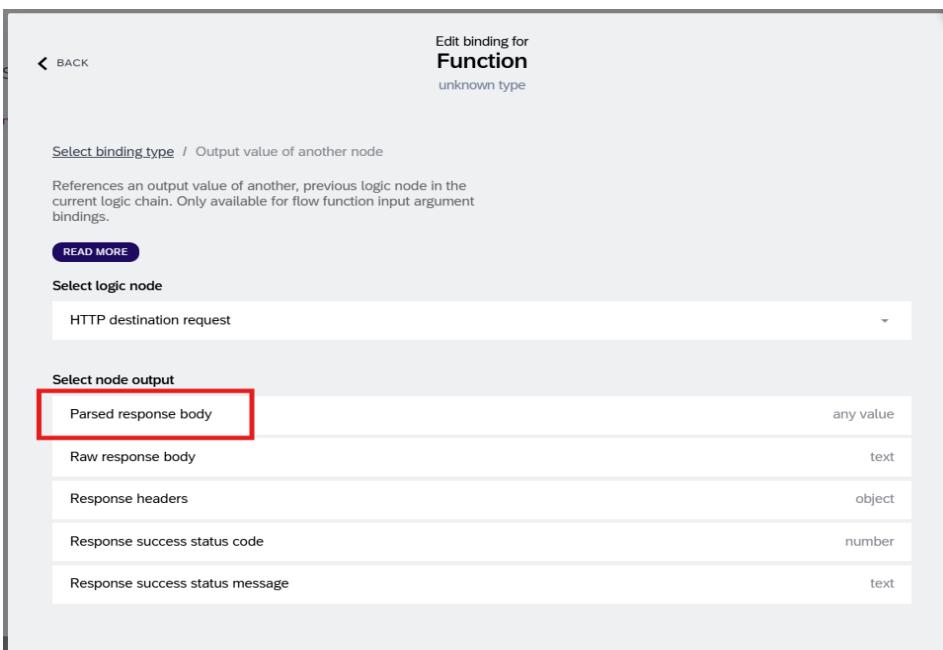
Step26: Select the output value of another node



Step27: Select the HTTP destination request from drop down.



Step28: After selecting it, you will get another node output option to select. Please select the Parsed response body and Save it.



Step29: Under the JavaScript editor remove the default code and add the below code snippet.

Code Snippet 3 Value:

```
return { result: inputs.input1.id };
```

As we're trying to read the Id from response, this id is nothing but UUID type, which is unique key, so set the output type as UUID in right side Options panel

Please Select Property type drop down--> Select UUID and click + icon.

The screenshot shows the JavaScript editor with the following code:

```
1 return { result: inputs.input1.id };
```

To the right is the 'Outputs' panel for the 'result' variable. It shows the 'value type' as 'Object'. Under 'result', the 'Property type' is set to 'UUID'. A red box highlights the 'UUID' option in the dropdown menu. Other settings include 'Example values' (4d7dfe35-9cd3-4eba-805), a checked 'Property is required' checkbox, and a 'Type property name' input field.

Step 30: Creation of Variable.

Now let's create few app variables: We need to create the below app variables, and I will explain why we need it in coming steps

The screenshot shows the 'Variables' tab selected in the navigation bar. Below is a table of app variables:

Variable Name	Type
InvoiceProcessButton	true/false
id	UUID
invoiceDate	text
invoiceNumber	text
totalAmount	text
vendor	text

A red box highlights the 'ADD APP VARIABLE +' button at the top right of the table.

Step31: Click on Add App Variable -> Select from Scratch.

User Interface Variables Integrations App Settings

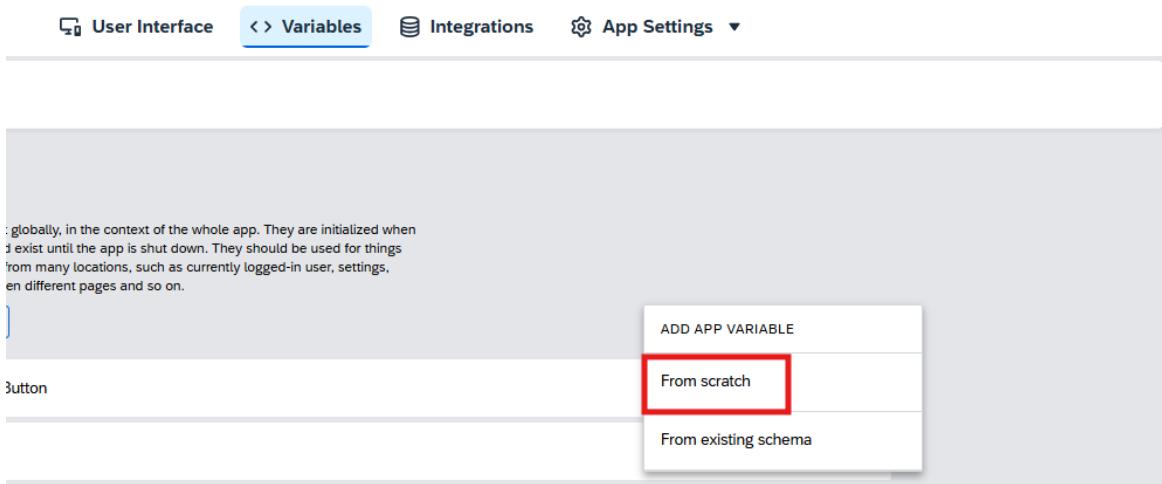
: globally, in the context of the whole app. They are initialized when exist until the app is shut down. They should be used for things from many locations, such as currently logged-in user, settings, en different pages and so on.

Button

ADD APP VARIABLE

From scratch

From existing schema



Step32: We need to give variable name, Value type from drop down. Please refer to the table below for the Variable names and Value Types that need to be created.

variable1
App variable

Variable name ⓘ
 variable1

Variable value type Text ▼

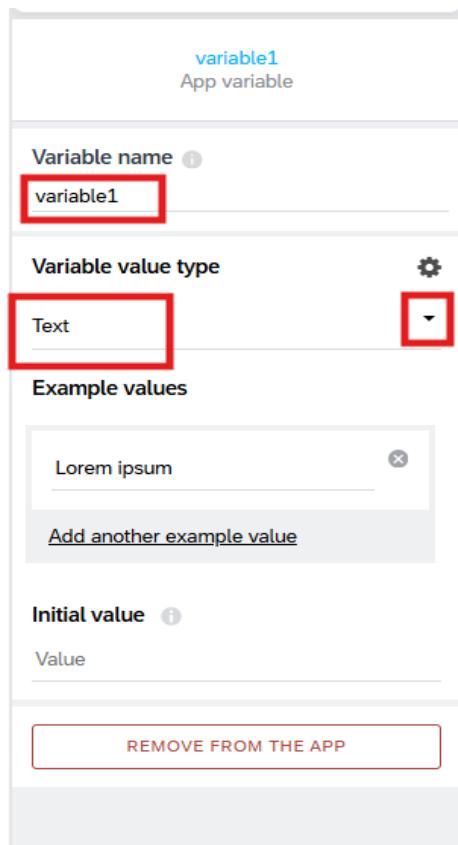
Example values

×

[Add another example value](#)

Initial value ⓘ

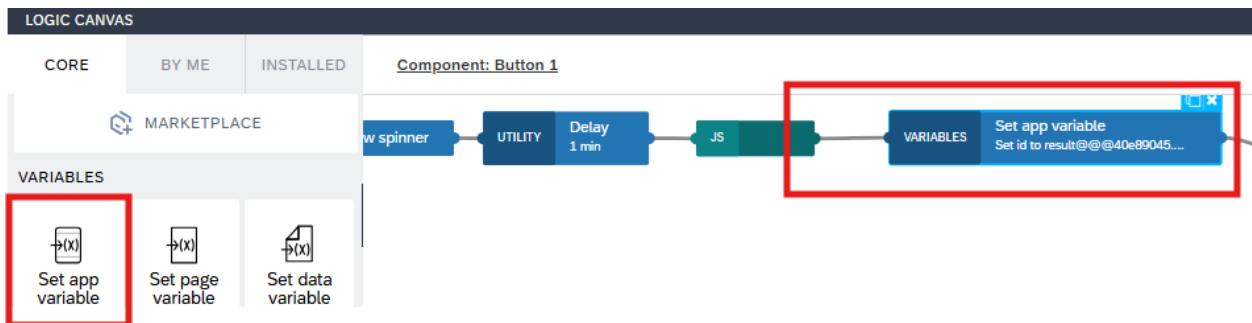
REMOVE FROM THE APP



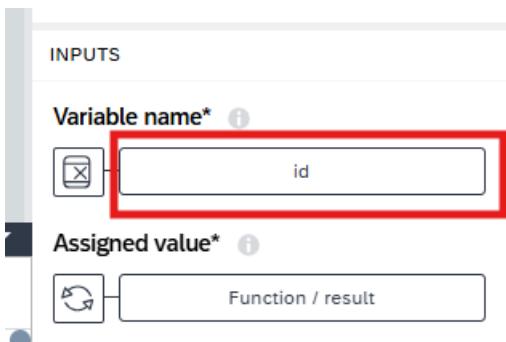
Key	Initial Value	Value Type
-----	---------------	------------

id		UUID
invoiceNumber		Text
grossAmount		Text
documentDate		Text

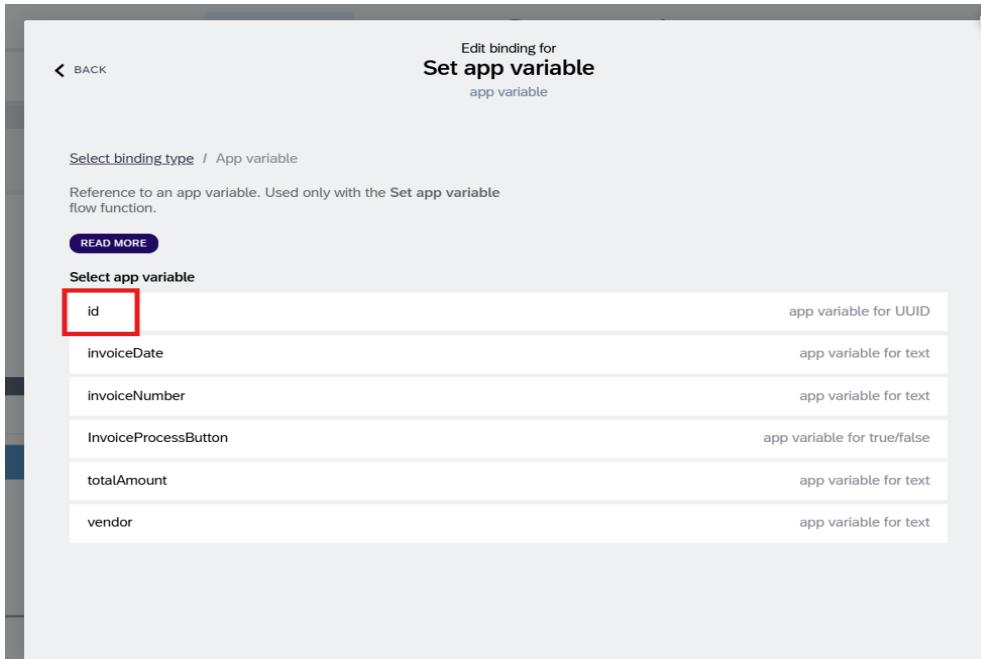
Step 33: Now from Canvas choose **set app Variable**, drag it and drop it. Connect it to JS component.



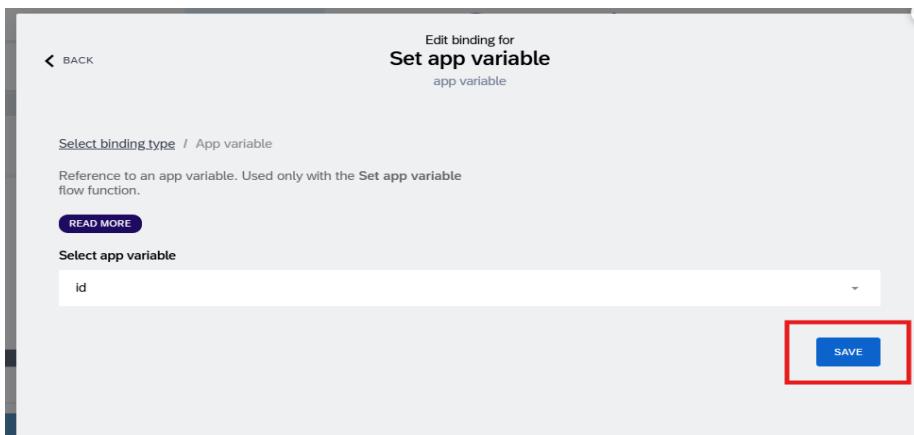
Step34: In properties Panel, by default in variable name it will show the first variable in the list, but you can click on list and choose which variable you want to set now.



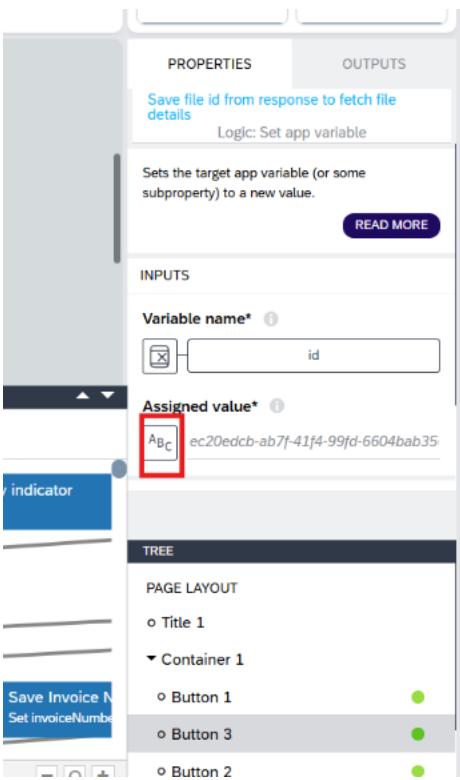
Step35: Click on Drop down and select the id from drop down List.



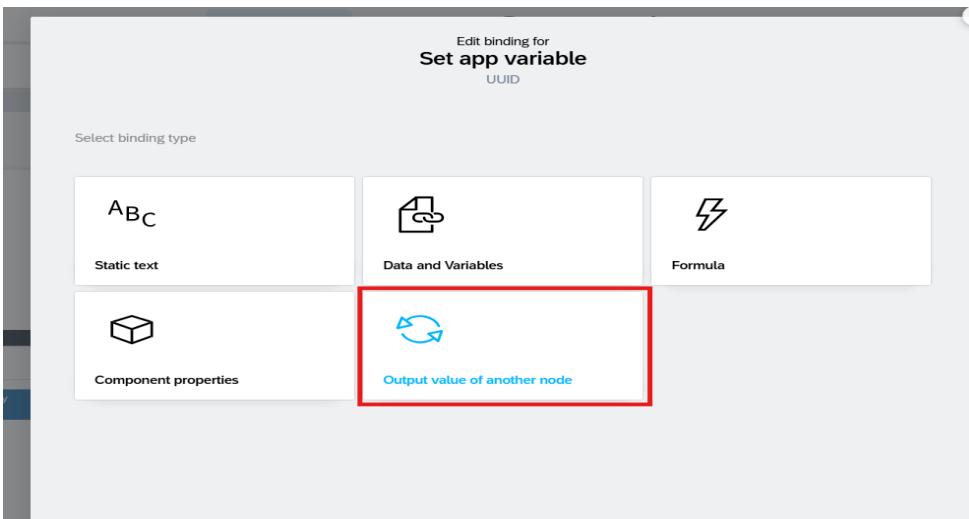
Step36: Save it



Step 37: Let's assign actual ID value to this variable, please select the "ABC" box in assigned value.



Step 38: Select the Output Value of another node.



Step 39: Select the Logic Node Function (previous node JS function we're referring here), after selecting the logic node automatically the node output will pop up and select the result from function from drop down. Save it.

Note: From the POST service response, we're able to save the ID into variable which we use later to fetch the invoice details.

BACK

Edit binding for
Set app variable
UUID

Select binding type / Output value of another node
References an output value of another, previous logic node in the current logic chain. Only available for flow function input argument bindings.

[READ MORE](#)

Select logic node

Function

Select node output

Function / result

SAVE

Step41: Now let's get the file content and store main details like Vendor, Invoice Number, Invoice Date, Total Amount in variables. Before that lets do one small step in Integration.

Go to Integration and select the destination we added -->docX_destination --> Click + Icon

UploadInvoiceApp

User Interface < > Variables Integrations

SAP Systems

docx_destination

SAP Systems

docx_destination

+ ENABLE MORE DATA ENTITIES

Step 42: You will see the below screen and in the left pane select "+"

The screenshot shows a web interface for managing data entities. At the top, there's a breadcrumb navigation: < Data / docx_destination. Below it, the title 'docx_destination' is displayed. Underneath the title, there are two tabs: 'Configuration' and 'Data entities', with 'Data entities' being the active one. A search bar labeled 'Search' is present. At the bottom of the main area, there is a button labeled '+ ADD REST API DATA ENTITY' which is highlighted with a red box.

Step 43: Now fill in the details in the pop up: Give Service name as GET and select only retrieve

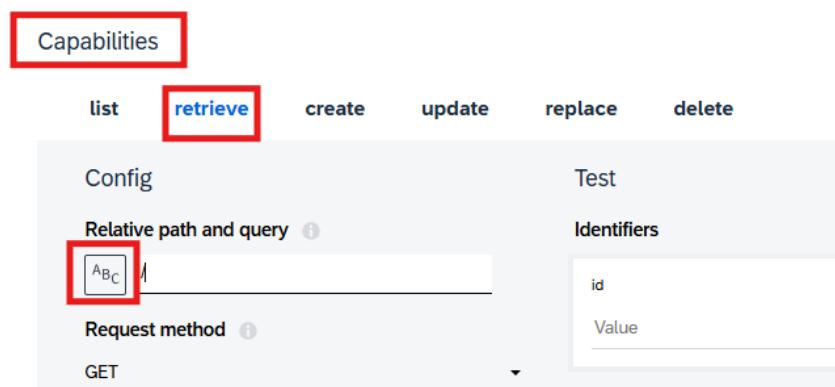
The screenshot shows a modal dialog titled 'Add REST API Data Entity'. It has several sections: 'Name' (containing 'GET'), 'Options' (with a checked checkbox for 'Autogenerate configuration'), 'Entity path within the API' (containing '/get'), 'Enable Capabilities' (with a checked checkbox for 'Enable all' and a selected checkbox for 'retrieve'), and a footer with 'Cancel' and 'Add' buttons. The 'Add' button is highlighted with a blue box.

Step 44: Now you will see the capabilities for this service, as we selected retrieve only retrieve is enabled for now.

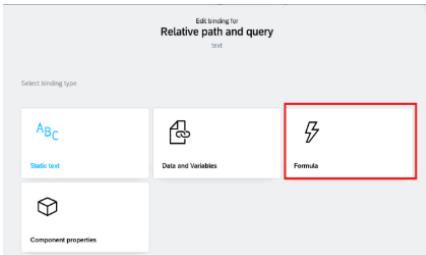
The screenshot shows the 'Data entities' page for 'docx_destination'. It includes a search bar, a '+ ADD REST API DATA ENTITY' button, and three buttons at the top: 'Browse Real Data', 'Browse Sample Data', and 'Generate Pages'. Below these are sections for 'Data entities' (listing 'GET') and 'Capabilities' (showing 'list' and 'retrieve' are selected). There are also 'Config', 'Test', 'Relative path and query', and 'Identifiers' sections, along with an 'Enabled' toggle switch which is turned on. The 'retrieve' button in the capabilities section is highlighted with a red box.

Step 45: Now we need to fill the Config details: In Relative path we need to give the path so Click on

“ABC” option and **select the formula**

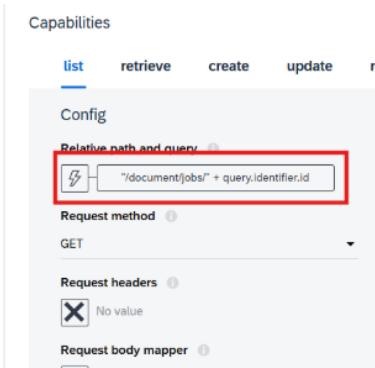


Step46: Select formula



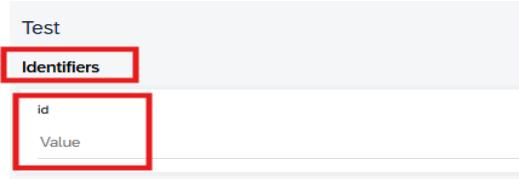
Step47: Now we need to give the relative path + uniquely generated id which comes as response from DocX service when we “Post” Invoice file.

ADD the below snippet and save: "/document/jobs/" + query.identifier.id

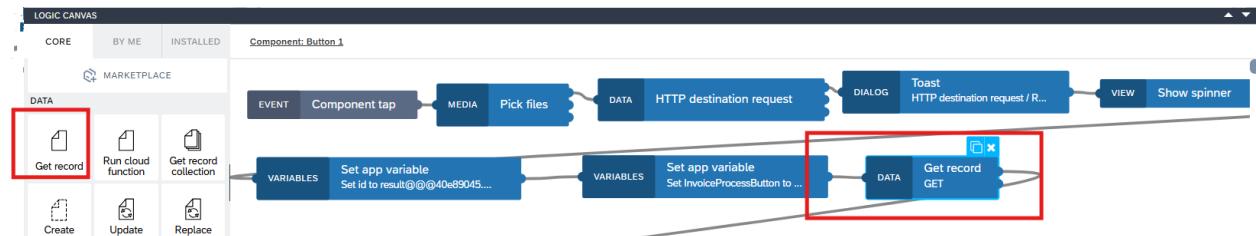


Step 48: Now inside identifiers id we must enter id and when we click the “Run test” we can see the response from DocX service.

Note: You will get the id when you upload file to Document AI, so you can directly open Document AI application and upload fie then you will get the id as response



Step 49: Now in logic canvas, select the **Get record**



Step 50: Click on “ABC” option and **select the formula and paste this formula appVars.id and save it**

PROPERTIES
OUTPUTS

(Get record)
Logic: Get record

Get a single record from a data resource.

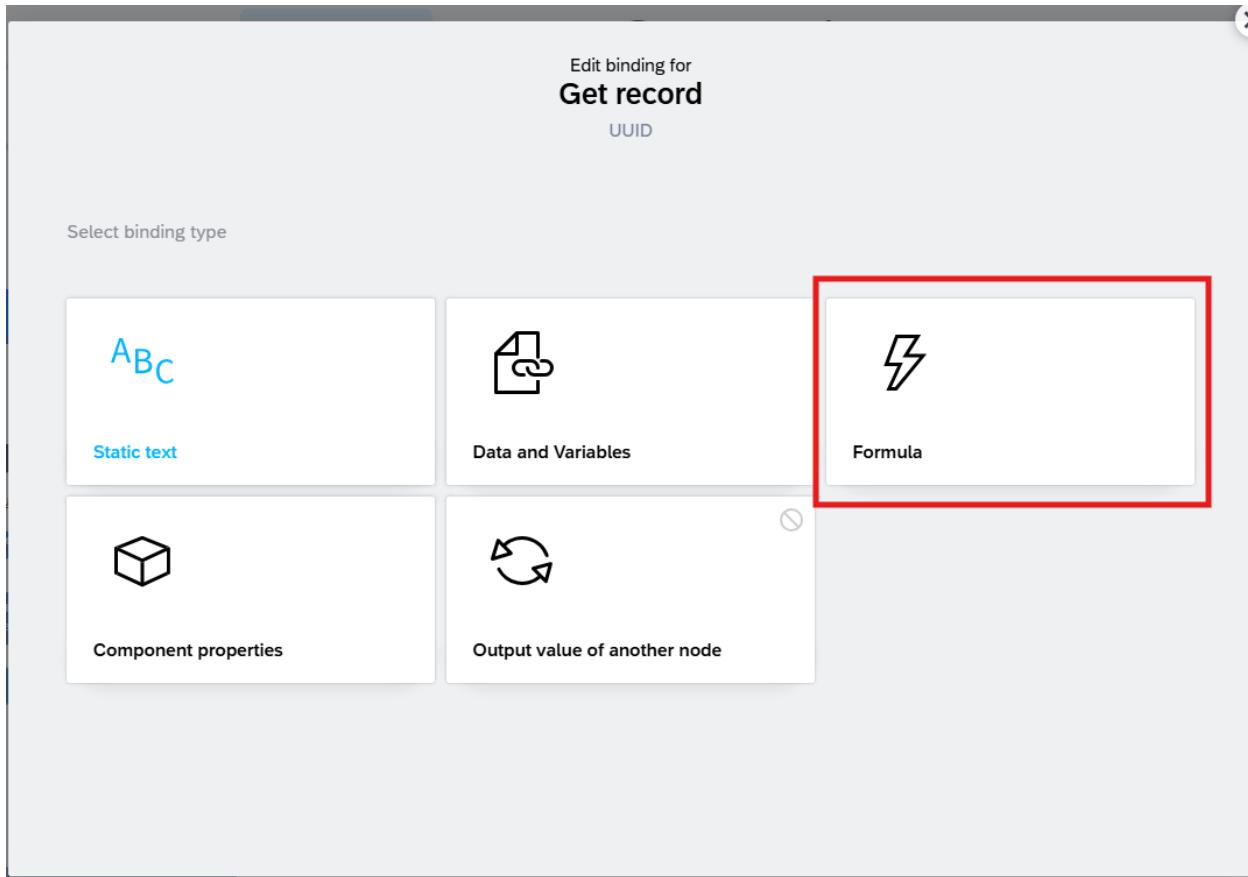
[READ MORE](#)

INPUTS

Resource name* i

id* i ec20edcb-ab7f-41f4-99fd-6604bab35

Query parameters i No value



◀ BACK

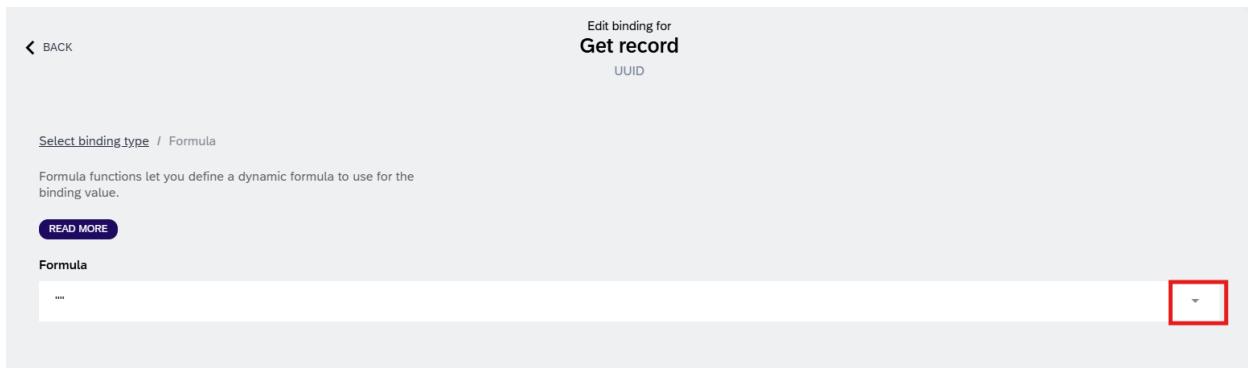
Edit binding for
Get record
UUID

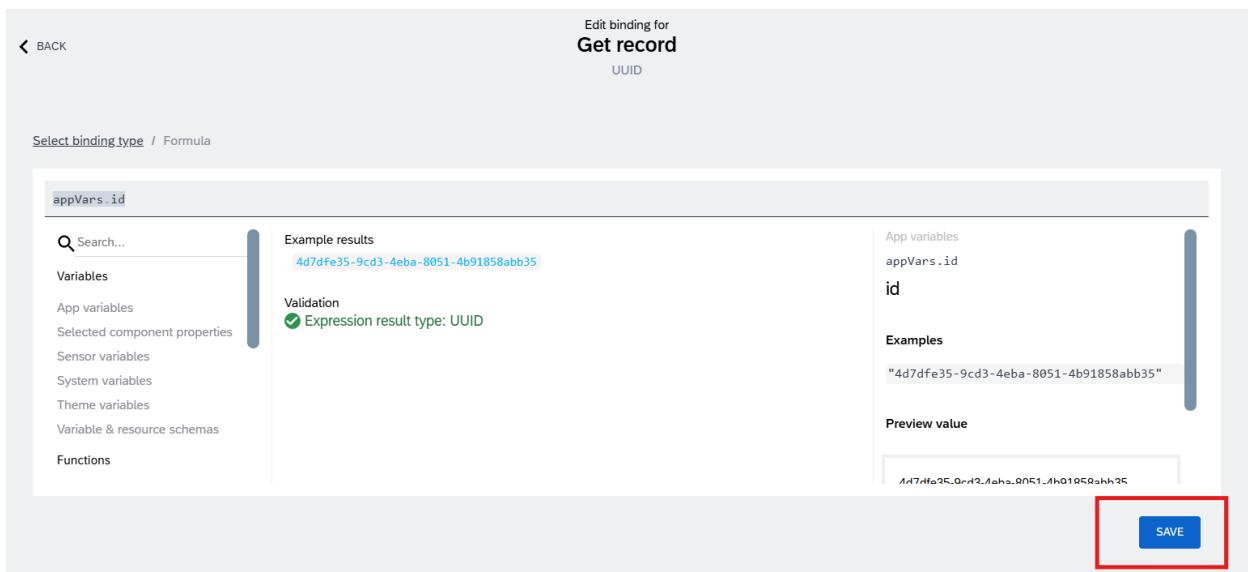
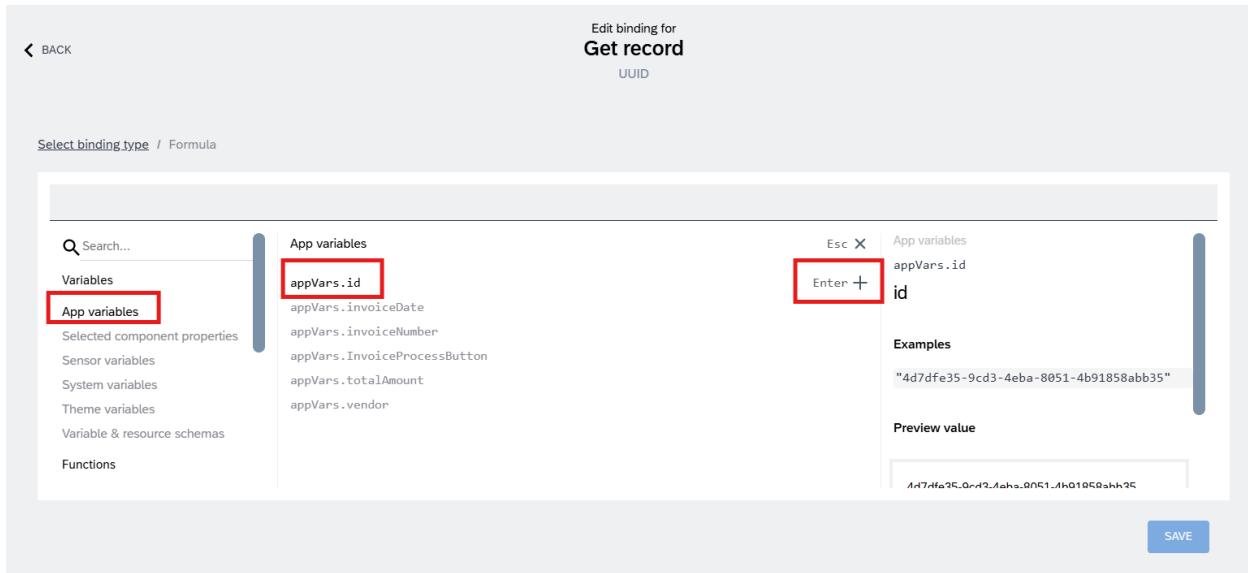
Select binding type / Formula

Formula functions let you define a dynamic formula to use for the binding value.

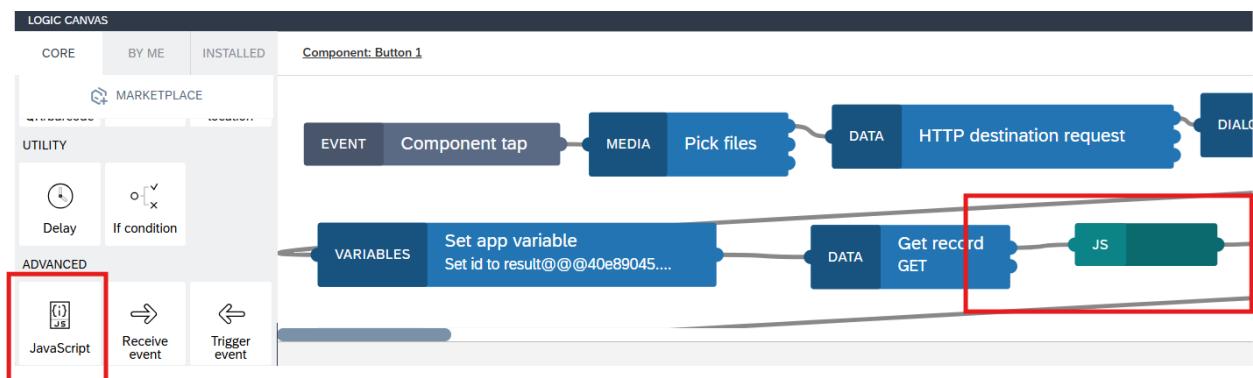
[READ MORE](#)

Formula





Step51: Drag JavaScript from Logic Canvas and drop it next to “Get Record” and connect the nodes.

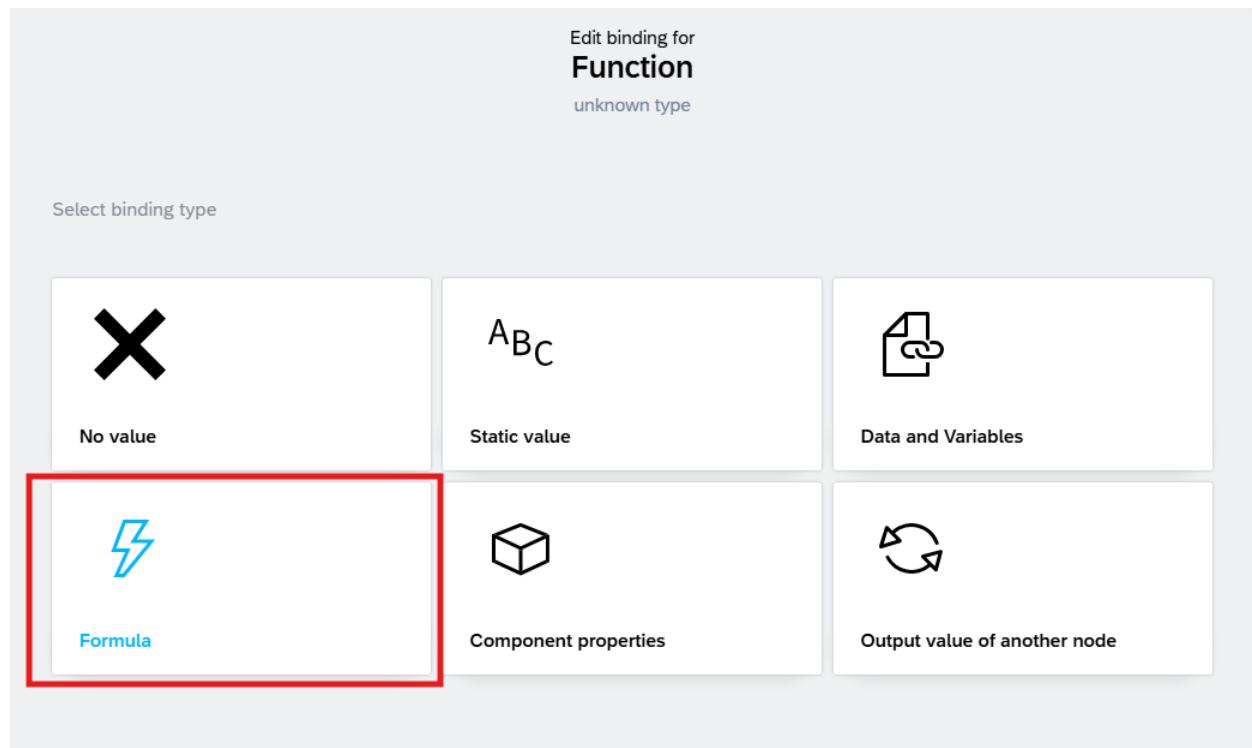


Step 52: Now let's write JavaScript to read the response and store the invoice details in app variables. Under advanced there is a java script: Firstly, select the “ABC” option in input1 and select the Formula



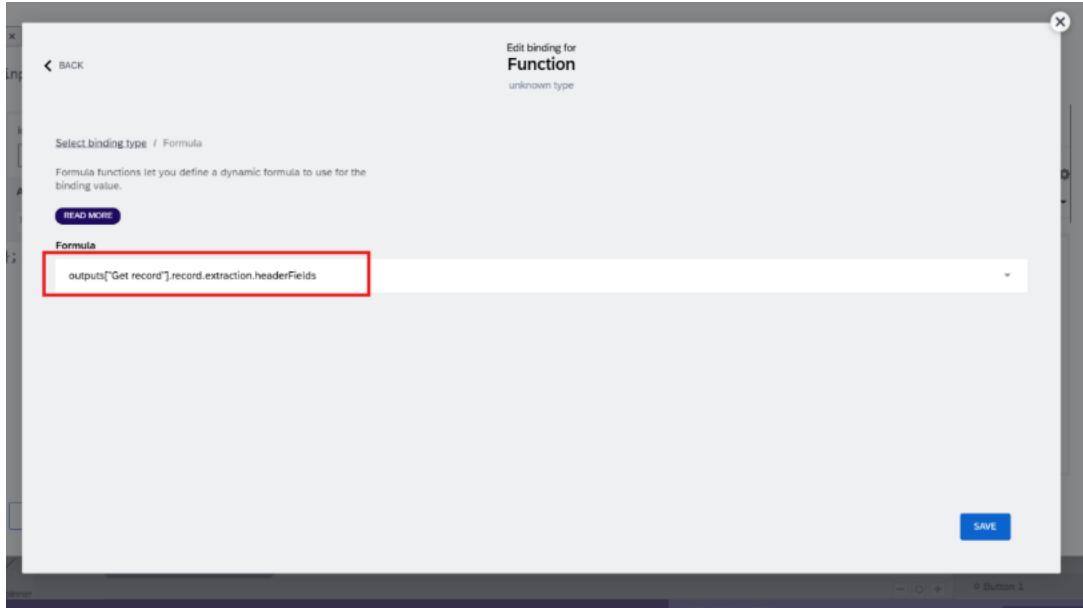
```
inputs = {  
    input1: {  
        value: "ABC"  
    }  
};  
  
function() {  
    return {  
        result: inputs.input1.id  
    };  
}
```

Select Formula



Copy the below snippet and paste it:

```
outputs["Get record"].record.extraction.headerFields
```



Add the below code snippet to JS

```
const headerFields = inputs.input1; const getValue = (name) => {  
  const match = headerFields.find(field => field.name === name);  
  return match ? match.value : "";  
};
```

```
let invoiceNumber = getValue("invoiceId");  
if(!invoiceNumber) {  
  invoiceNumber = getValue("documentNumber");  
}
```

```
const invoiceDate = getValue("documentDate");  
let vendorName = getValue("vendorName");  
if(!vendorName) {
```

```
vendorName = getValue("senderName");

}

let totalDue = getValue("grossAmount");

if(!totalDue){

    totalDue= getValue("netAmount");

}

return {

    invoiceNumberText: String(invoiceNumber),

    documentDateText: String(invoiceDate),

    vendorNameText: String(vendorName),

    grossAmountText: String(totalDue)

};
```

Step 53: We need to make sure we add Output object new properties and value type “Text” in Output (right side). Every time you add one property select Add new property.

Outputs

invoiceNumberText ×

Property type ⚙️

Text ▼

Example values

Lorem ipsum ×

[Add another example value](#)

Property is required

Add new property

Type property name +

invoiceNumberText ×

Property type ⚙️

Text ▼

Example values

Lorem ipsum ×

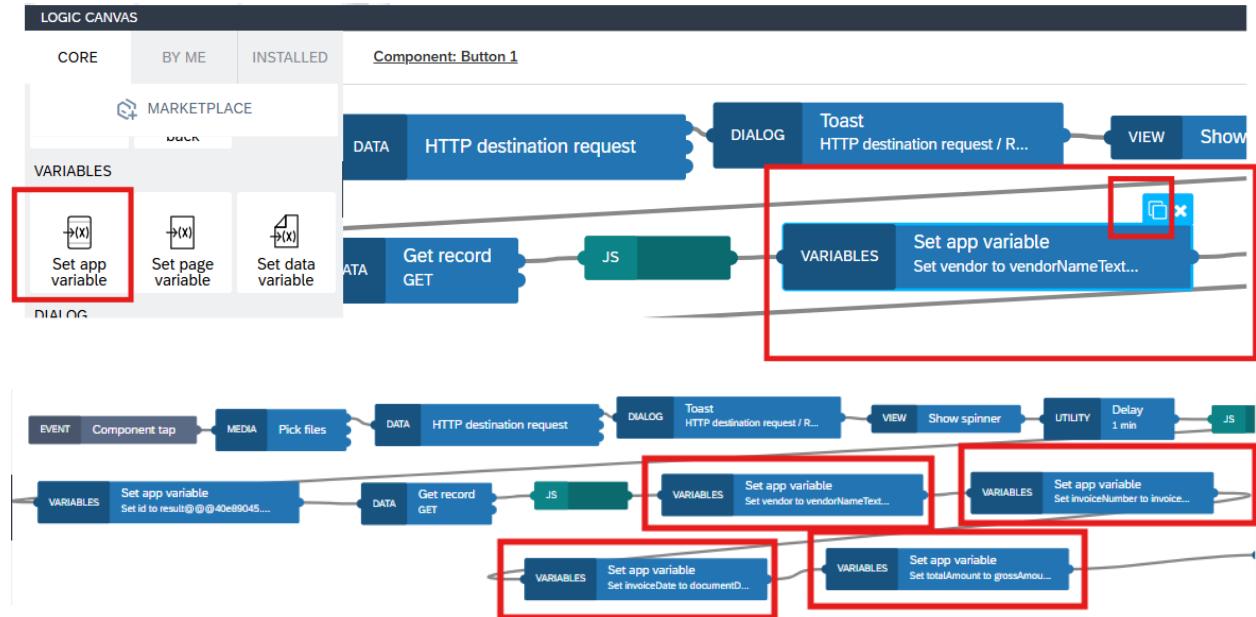
[Add another example value](#)

Property is required

Property
invoiceNumberText

documentDateText
vendorNameText
grossAmountText

Step 54: Now let's drag and drop set app variables and connect to the previous node. Since we need 4 values, you can use Copy icon and create 3 more components and connect it to the previous node.



Step 55: Now lets select one by one all 4 App variables component and add property details to it.

Select Variable Name -> by default it will display first one you can select from drop down the required variable

PROPERTIES

OUTPUTS

(Set app variable)
Logic: Set app variable

Sets the target app variable (or some subproperty) to a new value.

READ MORE

INPUTS

Variable name* 

Assigned value*  

▶ ADVANCED

 **BACK**

Edit binding for
Set app variable
app variable

Select binding type / App variable

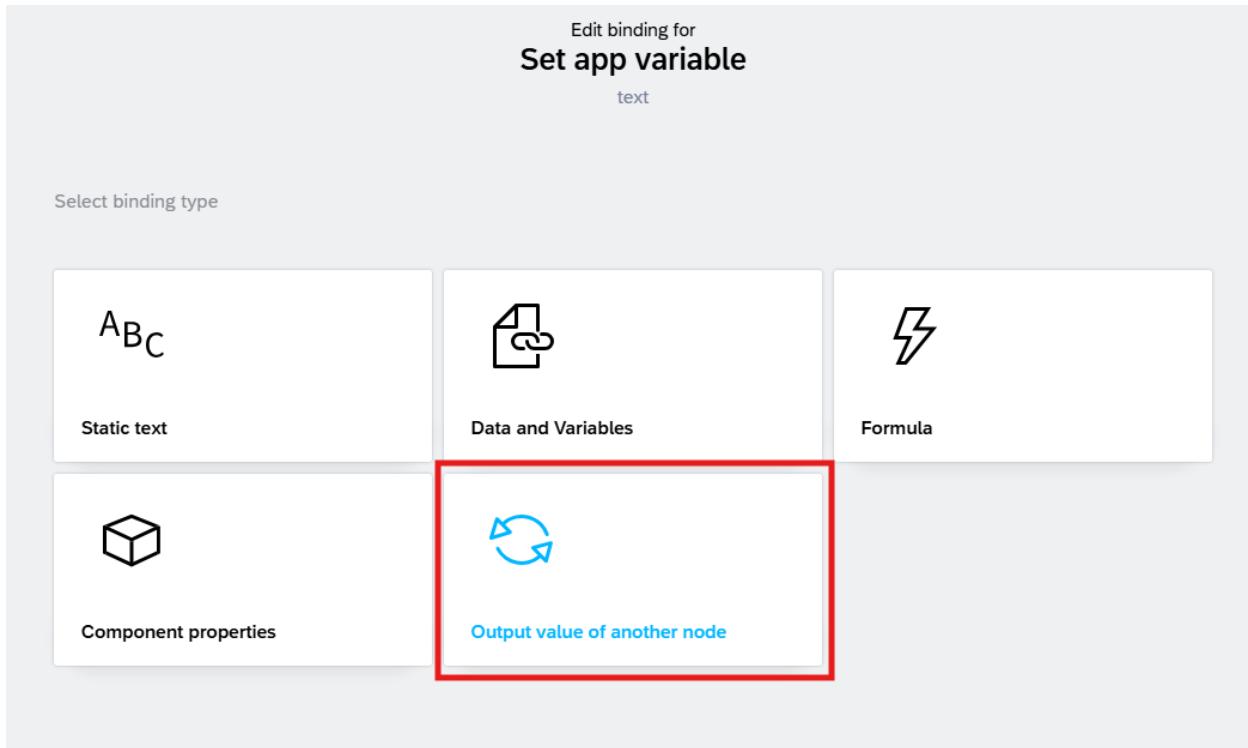
Reference to an app variable. Used only with the **Set app variable** flow function.

READ MORE

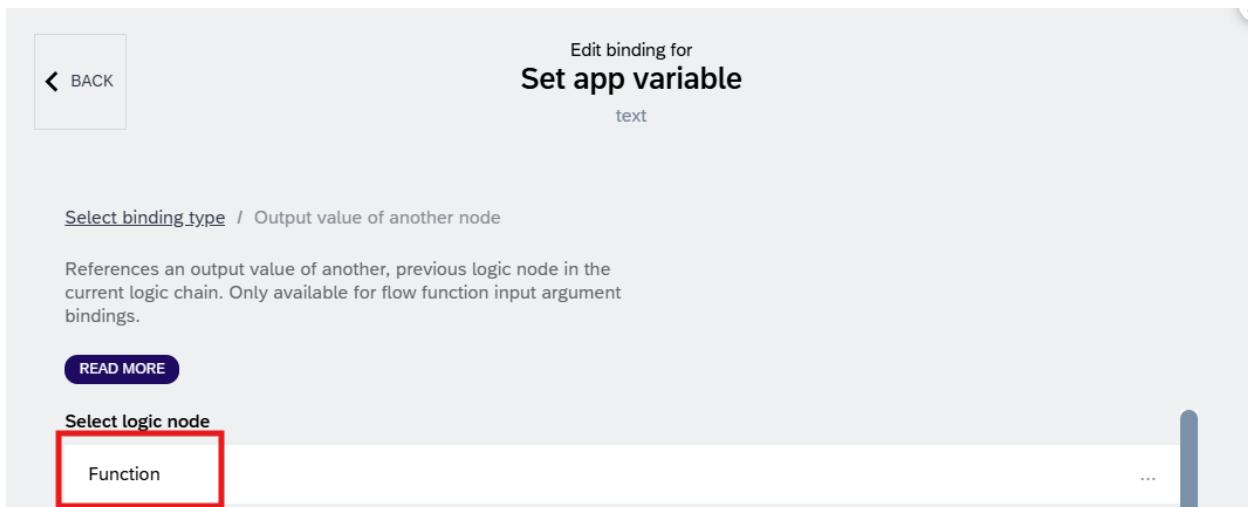
Select app variable

id	app variable for UUID
invoiceDate	app variable for text
invoiceNumber	app variable for text
InvoiceProcessButton	app variable for true/false
totalAmount	app variable for text
<input type="text" value="vendor"/>	app variable for text

Now Click on “ABC” option from Assign to and select the “Output from another node”



Select Function



Select VendorNameText

[BACK](#)

Edit binding for
Set app variable
text

[Select binding type](#) / Output value of another node

References an output value of another, previous logic node in the current logic chain. Only available for flow function input argument bindings.

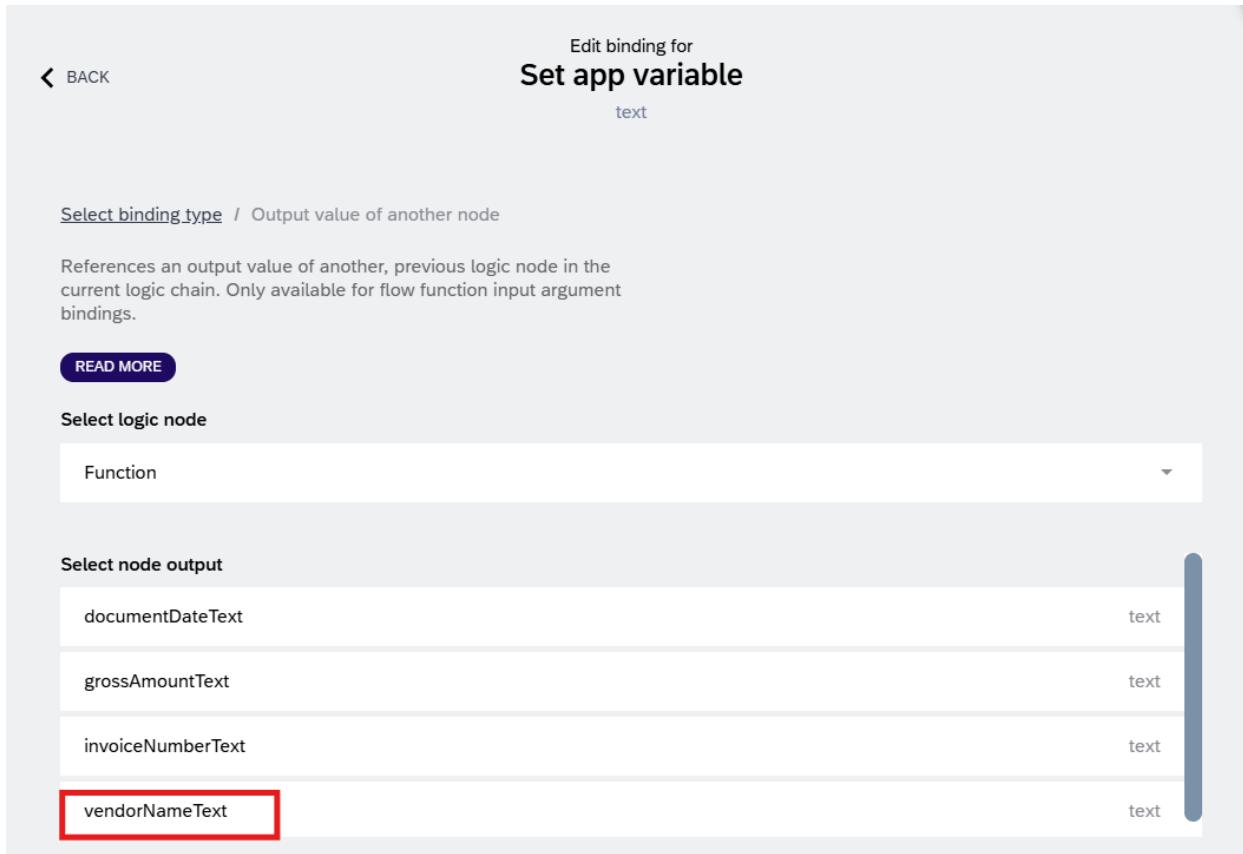
[READ MORE](#)

Select logic node

Function

Select node output

documentDateText	text
grossAmountText	text
invoiceNumberText	text
vendorNameText	text



Step 56: Repeat the previous step for next 3 variables and bind the values

PROPERTIES

OUTPUTS

(Set app variable)
Logic: Set app variable

Sets the target app variable (or some subproperty) to a new value.

[READ MORE](#)

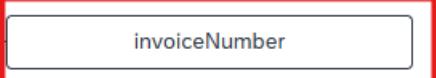
INPUTS

Variable name* 
 invoiceNumber

Assigned value* 
 Function / invoiceNumberText

► ADVANCED



BACK

Edit binding for
Set app variable

text

Select binding type / Output value of another node

References an output value of another, previous logic node in the current logic chain. Only available for flow function input argument bindings.

[READ MORE](#)

Select logic node

Function

Select node output

Output Name	Type
documentDateText	text
grossAmountText	text
invoiceNumberText	text
vendorNameText	text

PROPERTIES

[\(Set app variable\)](#)
Logic: Set app variable

Sets the target app variable (or some subproperty) to a new value.

[READ MORE](#)

INPUTS

Variable name*

Assigned value*

► ADVANCED

PROPERTIES

OUTPUTS

(Set app variable)
Logic: Set app variable

Sets the target app variable (or some subproperty) to a new value.

READ MORE

INPUTS

Variable name* ⓘ
totalAmount

Assigned value* ⓘ
Function / grossAmountText

◀ BACK

Edit binding for
Set app variable

Select binding type / Output value of another node

References an output value of another, previous logic node in the current logic chain. Only available for flow function input argument bindings.

READ MORE

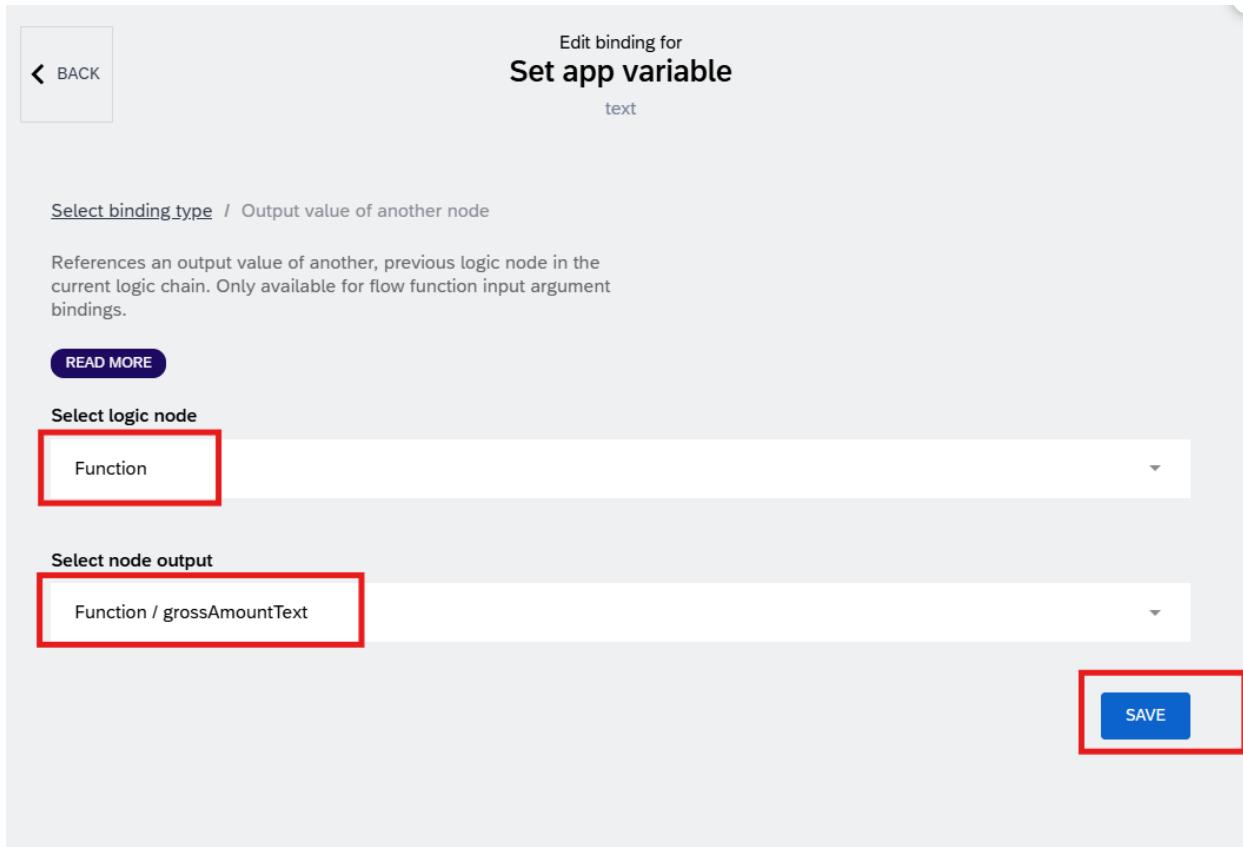
Select logic node

Function

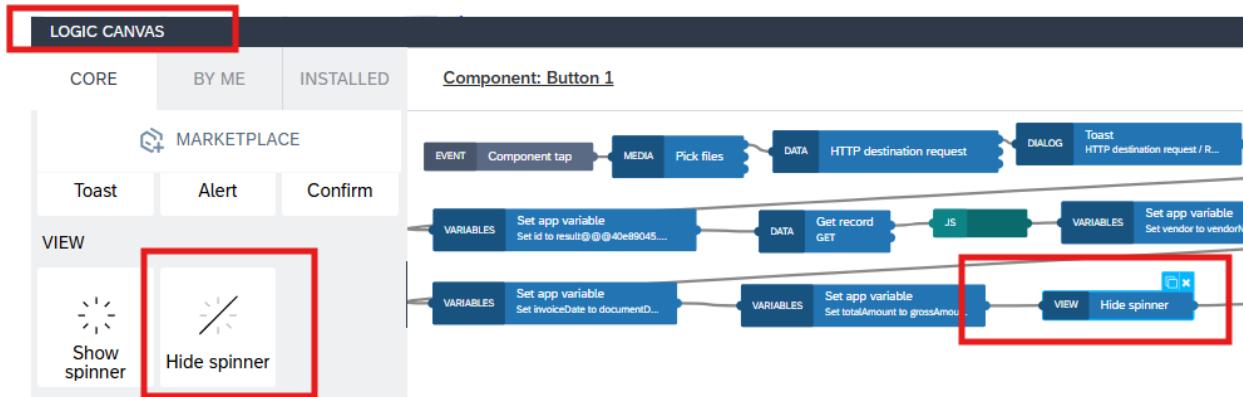
Select node output

Function / documentDateText

SAVE



Step 57: Drag Hide Spinner from Logic Canvas and drop it after set app variable component. Connect Hide Spinner to previous node.

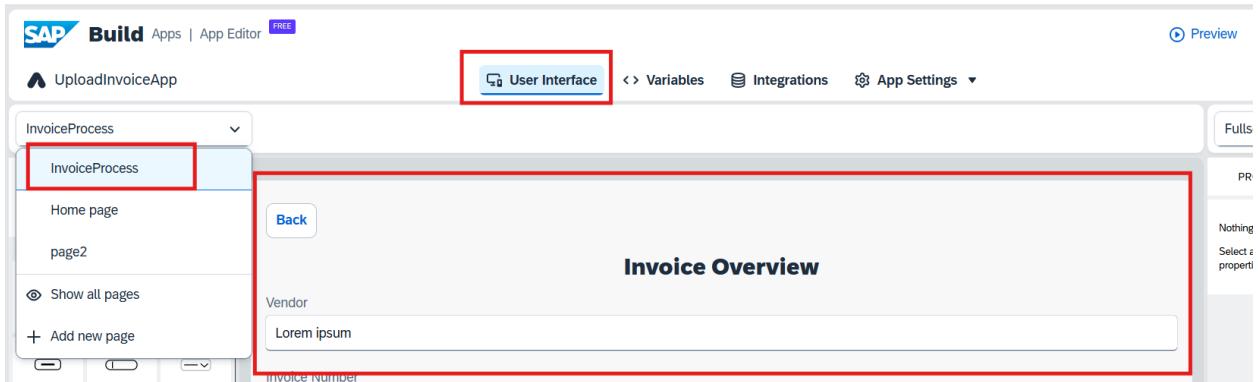


Step58: Drag Open Page from Logic Canvas and drop it after Hide Spinner component. Connect Open page to previous node.



Step59: Add Page detail in Properties. Click on the Open page component and on the right side you can see Properties. If you're not sure about page name, you can find it from drop down (refer second screenshot)

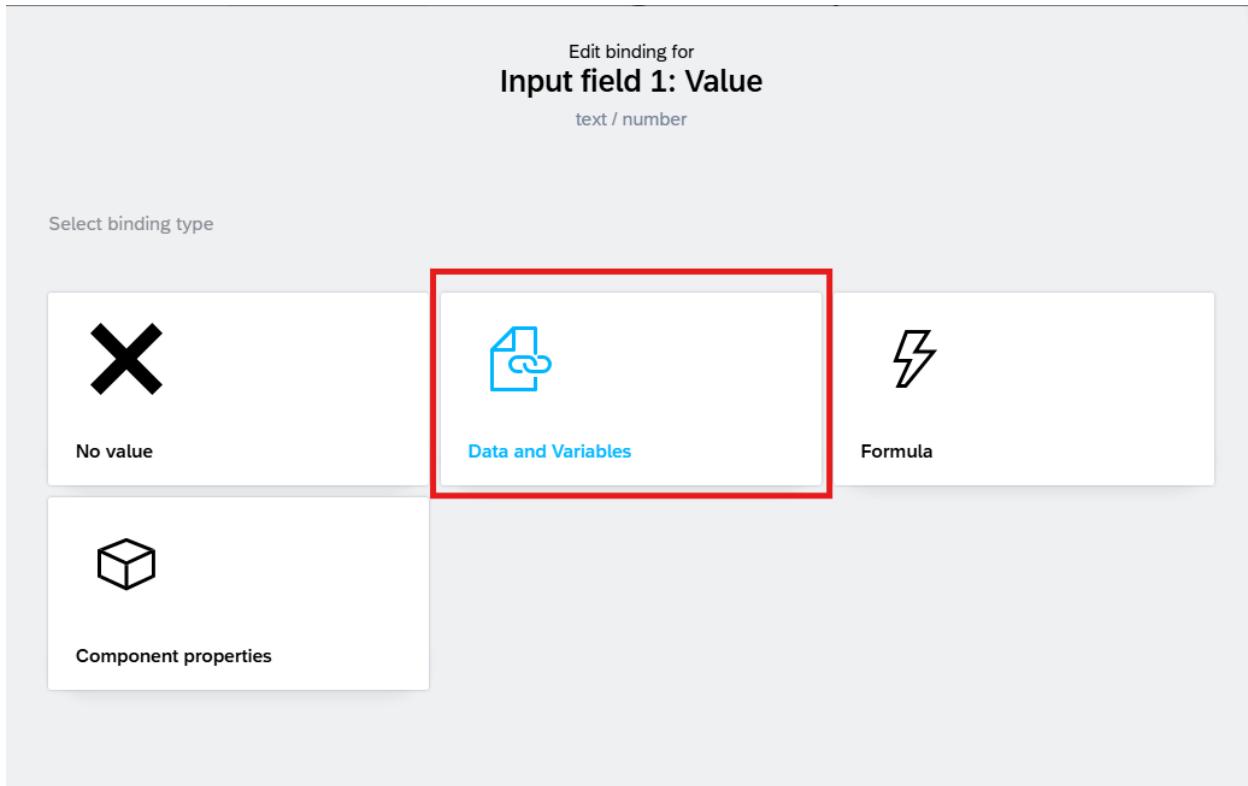
Step60: Select the second Screen from the drop down available at the header of the canvas.



Step 61: Click on First Input Field and in right you see the properties option available for it.
(We must repeat the same step for next 3 variables too)

A screenshot of the SAP Build App Editor showing the properties panel for an input field. The input field is labeled "Input field 1" and has a placeholder "Input field 1". The properties panel is open, showing three tabs: "PROPERTIES" (selected), "STYLE", and "LAYOUT". Under "PROPERTIES", there are sections for "Label" (containing "Vendor") and "Value" (containing "No value"). A red box highlights the "X" icon in the "Value" section, which is used for clearing the value.

Step 62: Click on X icon and select the Data and variables



Step 63: Select the variable

BACK

Edit binding for
Input field 2: Value

text / number

Select binding type / Data and Variables / App variable

App variables are globally available variables that you can use to store and access information between multiple pages.

READ MORE

Select app variable

id	UUID
invoiceDate	text
invoiceNumber	text
totalAmount	text
vendor	text

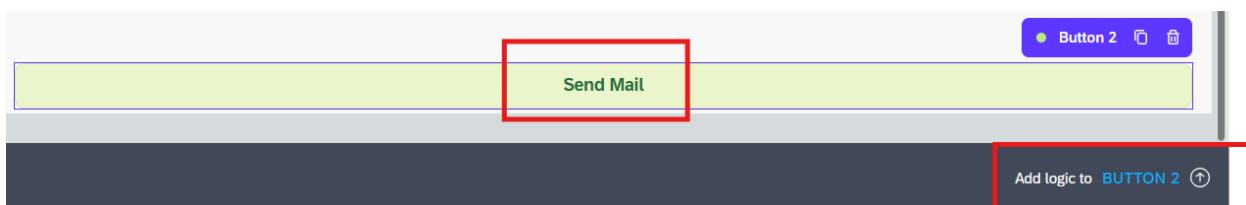
▶ 1 option is unavailable

Step 64: Repeat the step for other input fields too.

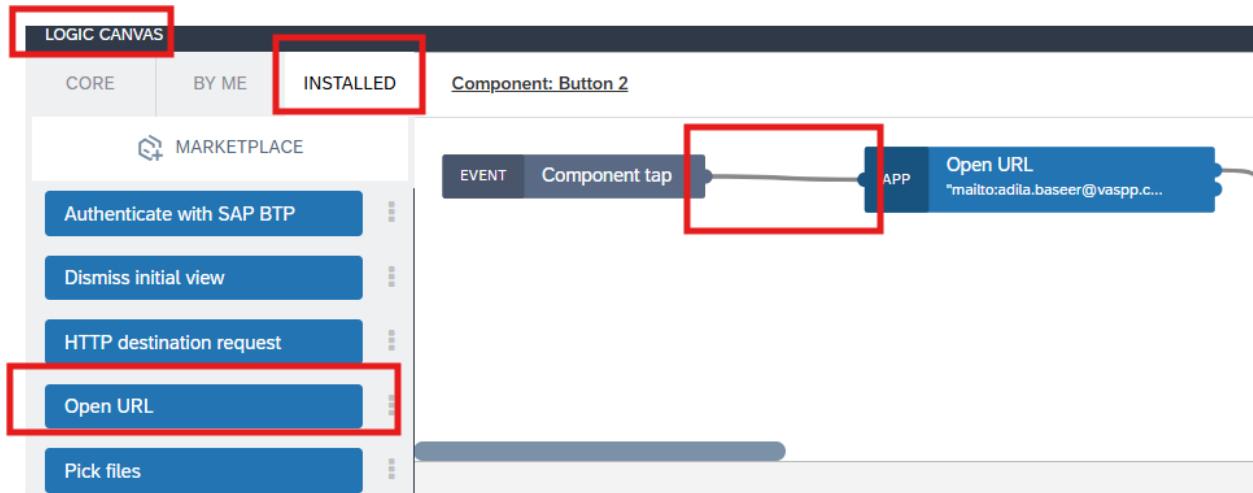
The image consists of three vertically stacked screenshots from a Logic Editor interface, each showing the configuration of an input field. Each screenshot has a header bar with tabs for PROPERTIES, STYLE, and LAYOUT. The first two screenshots also have a sub-header 'Input field' above the main content area.

- Top Screenshot:** The header is 'Input field 2' and 'Input field'. The 'Label' field contains 'Invoice Number' with a placeholder icon 'ABC'. The 'Value' field contains 'invoiceNumber' with a placeholder icon 'ABC'.
- Middle Screenshot:** The header is 'Input field'. The 'Label' field contains 'Invoice Date' with a placeholder icon 'ABC'. The 'Value' field contains 'invoiceDate' with a placeholder icon 'ABC'.
- Bottom Screenshot:** The header is 'PROPERTIES' (highlighted in grey), 'STYLE', and 'LAYOUT'. The 'Label' field contains 'Total Amount' with a placeholder icon 'ABC'. The 'Value' field contains 'totalAmount' with a placeholder icon 'ABC'.

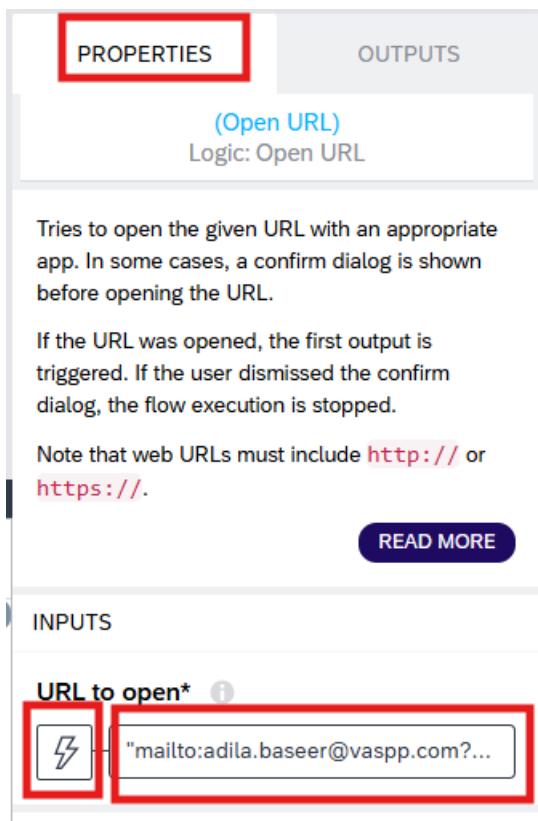
Step65: Click on Send Mail button and click on “Logic editor”

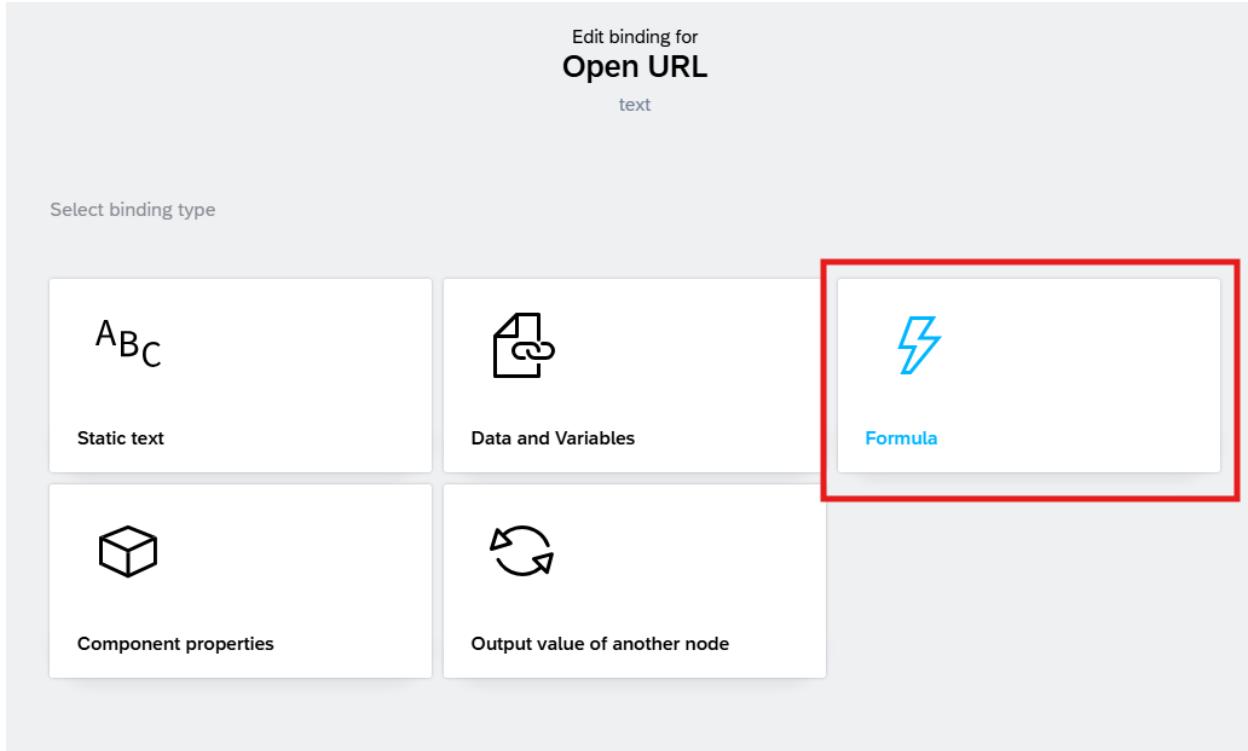


Step66: Go to Logic Canvas -> go to Installed tab --> Drag Open URL and Drop it on logic editor -> Connect it to previous node.



Step67: Select Open URL -> go to right side Properties --> Click on “ABC” option and select the formula option -> Copy and Paste below snippet





BACK

Edit binding for
Open URL
text

Select binding type / Formula

Formula functions let you define a dynamic formula to use for the binding value.

[READ MORE](#)

Formula

```
"mailto:adila.baseer@vaspp.com?subject=Invoice Details&body=" + "Respected Sir/Madam,%0D%0A%0D%0A"+ "Your Invoice has been Extracted Successfully with below details:%0D%0A%0D%0A"+ "Invoice Number: " + appVars.invoiceNumber + "%0D%0A"+ "Invoice Date: " + appVars.invoiceDate + "%0D%0A"+ "Vendor: " + appVars.vendor + "%0D%0A"+ "Total Amount: " + appVars.totalAmount + "%0D%0A%0D%0A"+ "Thank you.%0D%0ABest Regards,"
```

SAVE

Code Snippet: Note: make sure you replace <Enter_Receiver_MailID> with actual mail ID.

```
"mailto:<Enter_Receiver_MailID>?subject=Invoice Details&body=" + "Respected Sir/Madam,%0D%0A%0D%0A"+ "Your Invoice has been Extracted Successfully with below details:%0D%0A%0D%0A"+ "Invoice Number: " + appVars.invoiceNumber + "%0D%0A"+ "Invoice Date: " + appVars.invoiceDate + "%0D%0A"+ "Vendor: " + appVars.vendor + "%0D%0A"+ "Total Amount: " + appVars.totalAmount + "%0D%0A%0D%0A"+ "Thank you.%0D%0ABest Regards,"
```

Step64: Go to “Logic Canvas” --> Drag “Navigate back” component and drop it after “Open URL “ Component --> Connect it with previous Node

