

Contents

| | |
|---|----|
| Goal of the Workshop | 1 |
| Pre-requisites | 1 |
| Creating the Build App Project..... | 2 |
| Step 1 – Create a new Build App Project | 2 |
| Step 2 – Design the Product Pricing Form..... | 8 |
| Step 3 – Adding Logic to our Form to Publish an Event | 41 |
| Step 4 – Loading the Javascript Libraries | 68 |
| Step 5 – Previewing the Application | 88 |
| Step 6 – Testing the Application..... | 93 |

Goal of the Workshop

Guide participants through the process of creating a SAP Build Apps Application that can connect to the Advanced Event Mesh and demonstrate the ability to publish a message.

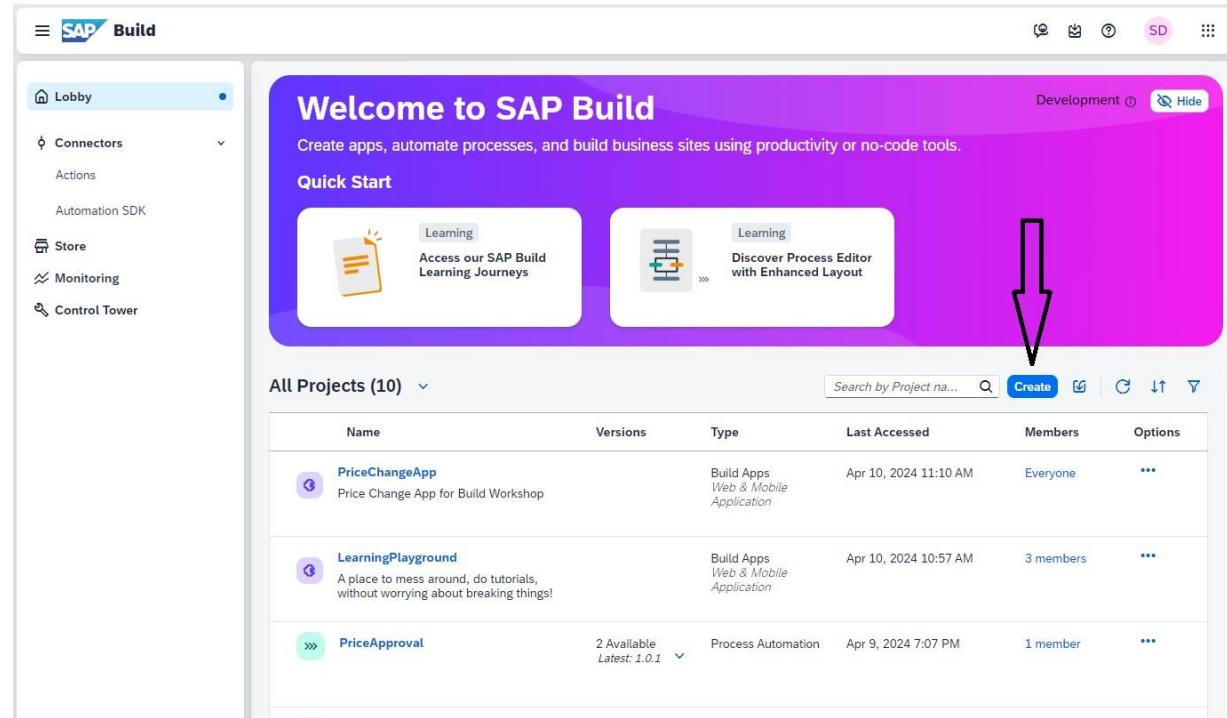
Pre-requisites

Access to the SAP Build Apps environment and access to a Advanced Event Mesh Broker.

Creating the Build App Project

Step 1 – Create a new Build App Project

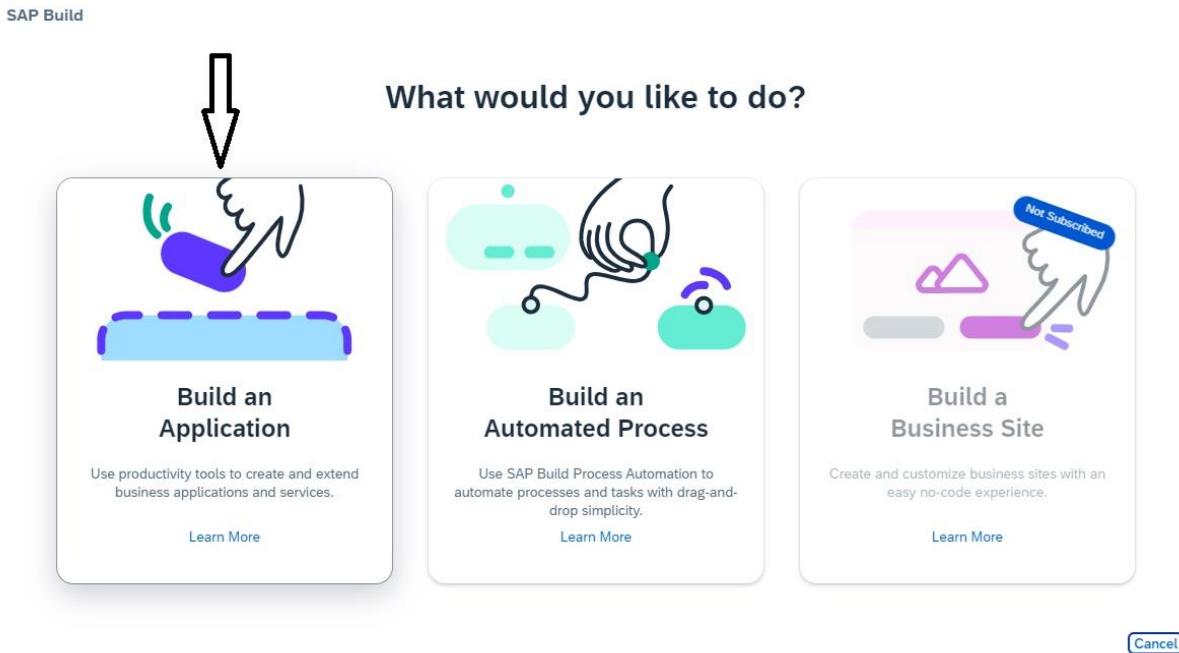
From the Lobby of the SAP Build, we will select the create option.



The screenshot shows the SAP Build lobby interface. On the left, there's a sidebar with links like 'Lobby' (which is selected), 'Connectors', 'Actions', 'Automation SDK', 'Store', 'Monitoring', and 'Control Tower'. The main area has a purple header 'Welcome to SAP Build' with the sub-header 'Create apps, automate processes, and build business sites using productivity or no-code tools.' Below this is a 'Quick Start' section with two cards: 'Access our SAP Build Learning Journeys' (with a document icon) and 'Discover Process Editor with Enhanced Layout' (with a database icon). A large black arrow points downwards from the top of the 'Discover Process Editor' card towards the 'Create' button. At the bottom, there's a table titled 'All Projects (10)' showing three projects: 'PriceChangeApp', 'LearningPlayground', and 'PriceApproval'. The 'Create' button is located at the top right of the project list table.

| Name | Versions | Type | Last Accessed | Members | Options |
|---|------------------------------|--|-----------------------|-----------|---------|
| PriceChangeApp Price Change App for Build Workshop | | Build Apps Web & Mobile Application | Apr 10, 2024 11:10 AM | Everyone | ... |
| LearningPlayground A place to mess around, do tutorials, without worrying about breaking things! | | Build Apps Web & Mobile Application | Apr 10, 2024 10:57 AM | 3 members | ... |
| PriceApproval 2 Available Latest: 1.0.1 | 2 Available Latest: 1.0.1 | Process Automation | Apr 9, 2024 7:07 PM | 1 member | ... |

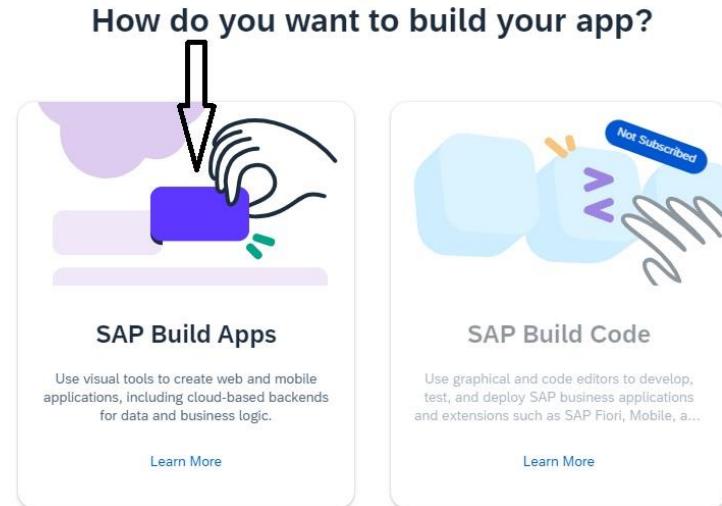
From there select “Build an Application”



From there select “SAP Build Apps”

(***My option for SAP Build Code is grayed out because I have not activated this service***)

Build an Application



[Back](#)

[Cancel](#)

Select “Web & Mobile Application”

SAP Build Apps

Which type of application would you like to build?





Web & Mobile Application

Create a multi-platform frontend app with powerful interaction logic and seamless integration with SAP and other systems.

[Learn More](#)



Application Backend

Create a cloud-deployed application backend with custom data and business logic that can be shared across multiple frontends.

[Learn More](#)

[Back](#)

[Cancel](#)

Give your name a project and an amazing description.

Create a **Build Apps** project

Give your project a name

Project Name:*

EDAWithBuild

Description:

Your first project to send events from SAP Build Apps. An easy way to bring agility and real time responsiveness to your business via a low code no code solution.

Back

Create

Cancel

It may take a few minutes for the project to be created. When ready, click on the project in the list to open it.

Your starting point for the project should look like this.

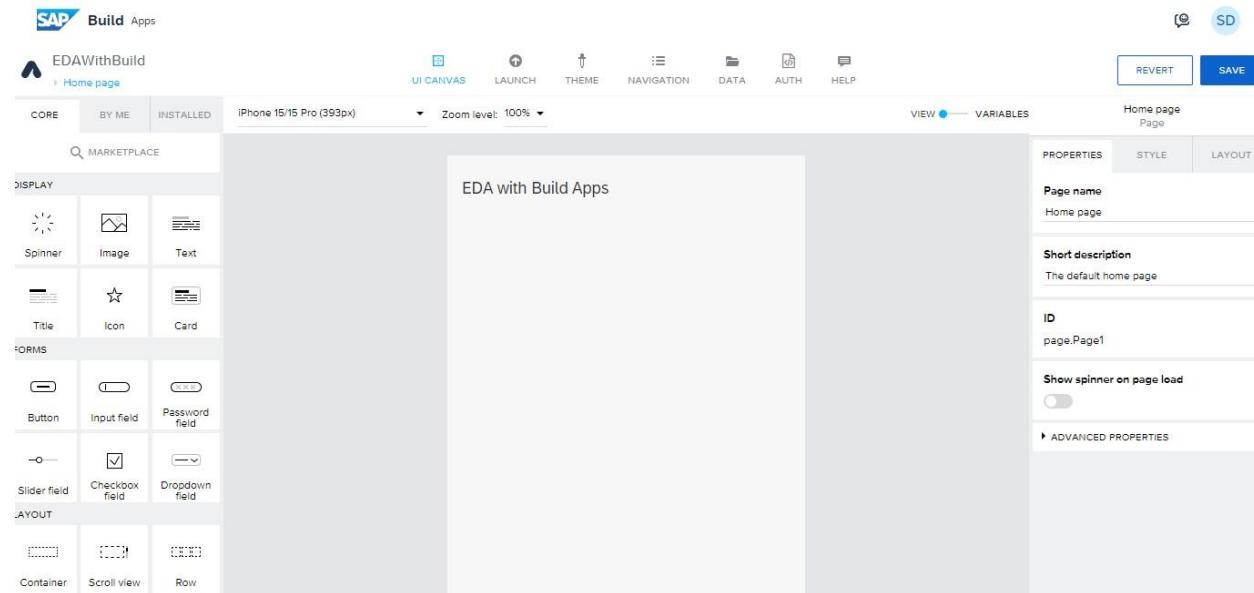
The screenshot shows the SAP Build Apps interface. On the left, there's a sidebar with categories: CORE, BY ME, and INSTALLED. Under CORE, there are sections for MARKETPLACE, DISPLAY (Spinner, Image, Text, Title, Icon, Card), FORMS (Button, Input field, Password field, Slider field, Checkbox field, Dropdown field), LAYOUT (Container, Scroll view, Row), and LISTS (List divider, List section header, Toggle list item, Search bar, Icon list item, List item, Large image list item, Image list item). A 'PRIMITIVES' section is also present. The main area shows a UI canvas with a page template. The template has a 'Headline' section containing placeholder text: 'Lorem ipsum dolor sit amet, consetetur sadipscing elitr.' To the right of the canvas is a properties panel for a 'Home page' page. The properties panel includes tabs for PROPERTIES, STYLE, and LAYOUT. Under PROPERTIES, fields are shown for 'Page name' (Home page) and 'Short description' (The default home page). An 'ID' field contains 'page.Page1'. A toggle switch for 'Show spinner on page load' is turned off. Below the properties panel is a tree view under the heading 'PAGE LAYOUT', which lists 'Title 1' and 'Text 1'. The top navigation bar includes UI CANVAS, LAUNCH, THEME, NAVIGATION, DATA, AUTH, and HELP, along with HISTORY and other icons.

Step 2 – Design the Product Pricing Form

The first thing we will do is rename the headline and remove the 2nd line of text.

Simply click on the Headline which will highlight the properties on the right hand side of the screen. Change the name to a title of your choice or “EDA with Build Apps” by editing the properties in the pane on the right.

Next, click the 2nd link of text and remove it by clicking on the x in the blue field that shows up.



At this point, you will see a component palette on the left side with the various widgets you can add to your form. Now we will add 4 “Input Field”, 4 “DropDown Field” and 1 “Button” by dragging and dropping these field from the palette on the left to your form in the middle (one by one) so the next iteration of your form will look like in the next screen shot.

Arrange the fields in the order shown on the screen.

The screenshot shows the SAP Build Apps interface. On the left, there is a component palette with categories: CORE, MARKETPLACE, DISPLAY, FORMS, LAYOUT, LISTS, and others. Under FORMS, three components are highlighted with red boxes: "Button", "Input field", and "Dropdown field". In the center, a form titled "EDA with Build Apps" is displayed. It contains several input fields and dropdowns, each with a label and a corresponding input box. At the bottom of the form is a large blue button labeled "Button". On the right, there is a properties panel for the current page, which includes fields for "Page name" (set to "Home page") and "Short description" (set to "The default home page"). Below the properties panel is a tree view showing the page structure: "Title 1" has four children: "Input field 1", "Dropdown field 1", "Dropdown field 2", and "Dropdown field 3".

Now before we make any further changes to the form, we will switch our view to create variables to which we will bind the various input fields. On the top right of the screen, there is slider bar and it's currently sitting on "View" -> Click on it to slide it to the right to access the variables screen.

The screenshot shows the SAP Build Apps interface for an application named "EDAWithBuild". The left sidebar has sections for APP VARIABLES, PAGE VARIABLES, PAGE PARAMETERS, DATA VARIABLES, and TRANSLATION VARIABLES. The main content area is titled "App variables" and describes them as existing globally. It includes a "READ MORE" button and an "ADD APP VARIABLE" button with a plus sign. At the top right, there is a navigation bar with icons for UI CANVAS, LAUNCH, THEME, NAVIGATION, DATA, AUTH, and HELP. A red box highlights the "VIEW" and "VARIABLES" buttons in the top right corner. Below them, the text "Nothing selected. Select a variable to edit it." is displayed.

Now we will add all the variables required for the application. On the left side of the page, click "PAGE VARIABLES". Start by clicking the "ADD PAGE VARIABLE". From here you will be creating a series of variables (10 in total).

The first one is called “currency” (rename the variable in the pane on the right).

Page variables

Page variables exist in the context of the current page. They are initialized when the page opens, and removed from app state when the page is closed. They should be used for things that exist in the context of the current page, such as form data, loading state of the current page, selected filters and so on.

READ MORE

ADD PAGE VARIABLE +

currency text

Variable name currency

Variable value type Text

Example values

Now you will create the next 9 variables....all with the “Text” Data Type (you should have 10 in total at the end of this) and rename each of them to match the following names. ***important*** make sure the spelling is exactly the same for the below variables, lowercase and uppercase matter. Use the same spelling every time they are referenced as well.

- topic
- origin
- pricing
- product_description
- product_id
- region
- specification

- Be sure to hit the “SAVE” button in the top right when you are done.

The screenshot shows a software interface for managing variables. On the left, there is a vertical sidebar with the following categories:

- APP VARIABLES
- PAGE VARIABLES
- PAGE PARAMETERS
- DATA VARIABLES
- TRANSLATION VARIABLES

The main panel is titled "Page variables". It contains the following text:

Page variables exist in the context of the current page. They are initialized when the page opens, and removed from app state when the page is closed. They should be used for things that exist in the context of the current page, such as form data, loading state of the current page, selected filters and so on.

Below this text are two buttons: "READ MORE" and "ADD PAGE VARIABLE" with a plus sign icon.

A table lists the page variables:

| Variable Name | Type |
|---------------------|------|
| currency | text |
| origin | text |
| pricing | text |
| product_description | text |
| product_id | text |
| region | text |
| specification | text |
| topic | text |

After saving, we will return to the form and configure our Form Fields and Map them to these App variables. From this screen, move the slider back to the “VIEW” position by clicking on it.

From here, you will now label our first field and map it to the corresponding page variable.

Click on the Label Property and Specify Topic. Next you will click on the Value and now we will bind it.

The screenshot shows the SAP Build Apps interface for creating a mobile application. The top navigation bar includes the SAP logo, project name "EDAWithBuild", and tabs for "UI CANVAS", "LAUNCH", "THEME", "NAVIGATION", "DATA", "AUTH", and "HELP". On the right, there are "REVERT" and "SAVE" buttons, along with a "SD" icon. The main workspace displays a mobile UI design for an "iPhone 15/15 Pro (393px)" at 100% zoom. The UI consists of several input fields and dropdowns. To the left, a sidebar provides a catalog of UI components categorized into "DISPLAY", "FORMS", "LAYOUT", and "LISTS". On the right, the "PROPERTIES" tab is selected for the first input field, which is currently labeled "Topic". The "Label" property is set to "ABC Topic" and the "Value" property is set to "No value". A note below states "Not bound to any value". Other properties shown include "Auto-capitalization" (none), "Disabled" (False), "Keyboard type" (default), "Multiline input" (False), and "Password input" (TREE). The "PAGE LAYOUT" section shows "Title 1" is selected. The "DATA" tab is highlighted in blue, indicating it is the active tab for this step.

When you click on the field (Click on the X), you will choose what the field is being bound to, we will choose “Data and Variables”.

The screenshot shows the SAP Build Apps interface for a project titled "EDAWithBuild". On the left, there's a sidebar with categories like CORE, BY ME, INSTALLED, MARKETPLACE, DISPLAY, FORMS, LAYOUT, LISTS, and PRIMITIVES. The main area displays a UI component titled "EDA with Build Apps" with a modal dialog titled "Edit binding for Input field 1: Value". The dialog has three options: "No value" (selected), "Data and Variables" (highlighted with a red box), and "Formula". To the right of the modal, the component properties are listed, including "Label" (Topic), "Value" (No value), "Placeholder text" (Type here...), "Auto-capitalization" (none), "Disabled" (False), "Keyboard type" (default), "Multiline input" (False), and "Password input" (TREE). At the bottom, there are buttons for "REVERT" and "SAVE". A progress bar at the bottom indicates "80%".

There are several types of variables but as per the earlier section, we created Page Variables so click “Page Variable”.

Edit binding for
Input field 1: Value
text / number

Select binding type / Data and Variables

Dynamic bindings whose values can change while the app is running.

[READ MORE](#)

 App variable

Available on all pages in your app

 Page variable

Available only on the current page

 Data variable

Access data from a backend

You will then be prompted with the list of all the variables you created earlier, Select "Topic" for this first binding.

Build Apps will then show you your selection, if you've picked the correct variable, Push the "Save" button at the bottom.

The screenshot shows the SAP Build Apps interface for a project named 'PriceChangeApp'. The central focus is a modal dialog titled 'Edit binding for Input field 1: Value' with the subtitle 'text / number'. The dialog lists various page variables: currency, origin, pricing, product_description, product_id, region, specification, and topic. The 'topic' entry is highlighted with a red box. In the background, the main workspace displays the Logic Canvas with components like 'Component: Input field 1' and 'Component tap'. The right side of the screen features the Properties, Style, and Layout tabs for the selected input field component.

 BACK

Edit binding for
Input field 1: Value

text / number

[Select binding type](#) / [Data and Variables](#) / Page variable

Page variables are variables that are only available on the page on which they have been defined.

[READ MORE](#)

Select page variable

topic

Optional preview value 

Lorem ipsum

SAVE

Now you should see that the Value of the Input Field is now bound to the “topic” page variable.

The screenshot shows the SAP Build Apps interface for a project named "EDAWithBuild". The UI canvas displays a screen titled "EDA with Build Apps" with several input fields. One input field is highlighted with a blue border and has the label "Input field 1" and the value "topic". The properties panel on the right shows the configuration for this input field, specifically the "Value" section which is set to "topic". The "Label" section also shows "topic". The "Value" section is circled in red, indicating the bound variable.

UI Canvas: EDA with Build Apps

Properties Panel:

- Label:** topic
- Value:** topic
- Placeholder text:** Type here...
- Auto-capitalize:** none
- Disabled:** False
- Keyboard type:** default
- Multiline input:** False
- Password input:** TREE
- PAGE LAYOUT:**
 - Title 1
 - Input field 1
 - Dropdown field 1
 - Dropdown field 2
 - Dropdown field 3

Now you will repeat the same process for the other 3 Input Fields (the ones that say “Label”):

- Price
- Origin
- Specification

So, once you are finished labelling and binding those input fields on the form, you should have something that looks like this:

The screenshot shows the SAP Build Apps interface for creating a home page. On the left, there's a sidebar with categories: CORE, BY ME, and INSTALLED. Under CORE, there are sections for DISPLAY (Spinner, Image, Text; Title, Icon, Card), FORMS (Button, Input field, Password field; Slider field, Checkbox field, Dropdown field), LAYOUT (Container, Scroll view, Row), LISTS (List divider, List section header, Toggle list item; Search bar, Icon list item, List item), and PRIMITIVES (Large image list item, Image list item). The main area shows a form titled "EDA with Build Apps" with four input fields: "Topic" (placeholder: "Lorem ipsum"), "Price" (placeholder: "Lorem ipsum"), "Origin" (placeholder: "Lorem ipsum"), and "Specification" (placeholder: "Lorem ipsum"). A blue button labeled "Button" is at the bottom. To the right, there's a properties panel for "Page name" (Home page), "Short description" (The default home page), "ID" (page.Page1), and "Show spinner on page load" (disabled). Below that is an "ADVANCED PROPERTIES" section. At the bottom, there's a tree view under "PAGE LAYOUT" and a progress bar indicating "80%".

EDA with Build Apps

Topic
Lorem ipsum

Region
Select option

Dropdown label
Select option

Dropdown label
Select option

Price
Lorem ipsum

Dropdown label
Select option

Origin
Lorem ipsum

Specification
Lorem ipsum

Button

Properties

Page name

Home page

Short description

The default home page

ID

page.Page1

Show spinner on page load

ADVANCED PROPERTIES

TREE

PAGE LAYOUT

- o Title 1
- Input field 1
- Dropdown field 1
- Dropdown field 2
- Dropdown field 3
- Input field 2
- Dropdown field 4
- Input field 3
- Input field 4
- o Button 1

Add logic to HOME PAGE

80%

YOUR LEARNING STATUS

Data

Now we will take care of the dropdown fields. Change the label to Region for the first field and then click to set the “Option List”. Click on the Box where it says “Custom List(0 Items)...”

The screenshot shows the SAP Build Apps interface. On the left, there's a sidebar with categories like CORE, DISPLAY, FORMS, LAYOUT, and LISTS, each with sub-options. The main area displays a form titled "EDA with Build Apps". It contains several input fields: a text input "Topic" with placeholder "Lorem ipsum", a dropdown field "Region" with placeholder "Select option", a text input "Price" with placeholder "Lorem ipsum", a dropdown field "Origin" with placeholder "Lorem ipsum", and a text input "Specification" with placeholder "Lorem ipsum". To the right, the "PROPERTIES" panel is open for the "Region" dropdown field. The "Label text" is set to "Region". The "Option list" section is highlighted with a red box; it shows a placeholder "Custom list (0 items)" with a small icon. Other properties shown include "Selected value" (No value), "Placeholder text" (Select option), "Selected label" (No value), and "Show value as label" (False). At the bottom of the properties panel, there's a "TREE" section listing various input fields.

Once you click on the custom list, you will be presented with this screen where you can add your options. You will create 3:

- EMEA
- North America
- APJ

The screenshot shows the SAP Fiori Launchpad interface. A central modal dialog is open, titled "Dropdown field 1: Option list". The dialog displays three entries, each consisting of a label and a value. The first entry has "EMEA" as both the label and value. The second entry has "North America" as both the label and value. The third entry has "APJ" as both the label and value. Below these entries is a link "Add another value". At the bottom right of the dialog is a blue "SAVE" button, which is highlighted with a red rectangle. To the right of the dialog, a sidebar titled "Dropdown field" contains various configuration settings, such as "Label text" (set to "Region"), "Option list" (set to "Custom list (1 item)"), and "Selected value" (set to "No value"). The "Component display name" section at the bottom lists several items under "TREE", including "PAGE LAYOUT", "Input field 1", "Dropdown field 1", "Dropdown field 2", "Dropdown field 3", "Input field 2", "Dropdown field 4", "Input field 3", and "Input field 4".

dropdown field 1: Option list

list of objects with 2 properties

Select binding type / List of values

label* ABC EMEA

value* ABC Value

label* ABC North America

value* ABC Value

label* ABC APJ

value* ABC Value

Add another value

SAVE

Label text ABC Region

Option list Custom list (1 item)

Selected value No value

Placeholder text ABC Select option

Selected label No value

Show value as label False

Repeat with Not repeated

Component display name

TREE

- PAGE LAYOUT
- Input field 1
- Dropdown field 1
- Dropdown field 2
- Dropdown field 3
- Input field 2
- Dropdown field 4
- Input field 3
- Input field 4

Once you click Save, you will be returned back to the previous screen and if you remembered to Save, you will now also see "Custom list (3 items)" as in the this screenshot.

The screenshot shows the SAP Build Apps interface for creating a mobile application. On the left, there's a sidebar with categories like CORE, DISPLAY, FORMS, LAYOUT, LISTS, and a MARKETPLACE search bar. The main area displays a page titled "EDA with Build Apps" with several input fields. One of these fields is a dropdown labeled "Region". The properties panel on the right shows the configuration for this dropdown. Two specific settings are highlighted with red boxes: "Label text" containing "Region" and "Option list" containing "Custom list (3 items)". Other visible properties include "Selected value" (No value), "Placeholder text" (Select option), "Selected label" (No value), "Show value as label" (False), "Repeat with" (Not repeated), and "Component display name" (TREE). The "PAGE LAYOUT" section lists components like Title 1, Input field 1, and three dropdown fields (dropdown field 1, dropdown field 2, dropdown field 3).

| Category | Component | Description |
|----------|---------------------|----------------|
| DISPLAY | Spinner | |
| | Image | |
| | Text | |
| | Title | Icon |
| | Card | |
| | Form | Button |
| | Input field | Input field |
| | Password field | Password field |
| | Slider field | Slider field |
| | Checkbox field | Checkbox field |
| Form | Dropdown field | |
| Layout | Container | |
| Layout | Scroll view | |
| Layout | Row | |
| Lists | List divider | |
| Lists | List section header | |
| Lists | Toggle list item | |
| Lists | Search bar | |
| Lists | Icon list item | |
| Lists | List item | |
| Lists | Large image | |
| Lists | Image list | |

UI CANVAS LAUNCH THEME NAVIGATION DATA AUTH HELP

VIEW VARIABLES

REVERT SAVE

Dropdown field 1
Dropdown field

Properties Style Layout

Label text ABC Region

Option list [] Custom list (3 items)

Selected value No value

Placeholder text Select option

Selected label No value

Show value as label False

Repeat with Not repeated

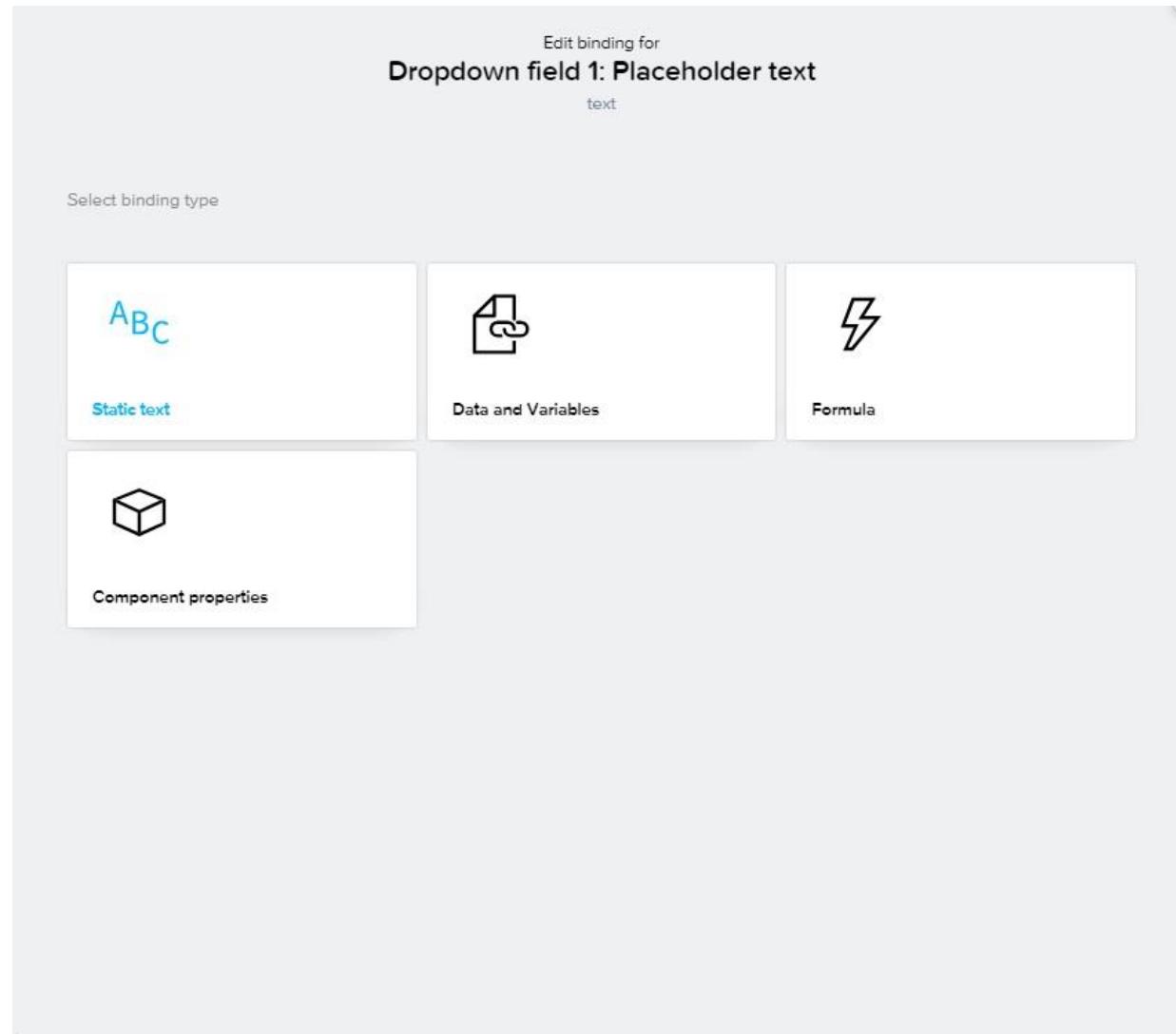
Component display name TREE

PAGE LAYOUT

- o Title 1
- o Input field 1
- o Dropdown field 1
- o Dropdown field 2
- o Dropdown field 3

Now you will be following a similar process for setting the Placeholder Text and mapping it to the Region Page Variable.

Just notice as you click through, there are a few more options but your path will be the same. Start by clicking on the “ABC” box in front of “Select Option” under “Placeholder text” in the properties pane. You will then see the following 4 options, Choose “Data and Variables”.



Select “Page variable”

BACK

Edit binding for
Dropdown field 1: Placeholder text
text

[Select binding type](#) / Data and Variables

Dynamic bindings whose values can change while the app is running.

[READ MORE](#)

| | | |
|--|---|---|
|  App variable Available on all pages in your app |  Page variable Available only on the current page |  Page parameter Pass data to a page when it's opened |
|  Data variable Access data from a backend |  Translation variable Translate text to the current language |  Sensor variable Access device sensor information |
|  System variable Access device and platform information | | |

Again you will see your page variables, select “region”.

The screenshot shows the SAP Build Apps interface for a project named "PriceChangeApp". A modal dialog is open titled "Edit binding for Dropdown field 4: Selected value". The dialog has a sub-section "Select binding type / Data and Variables / Page variable". It lists several page variables: currency, origin, pricing, product_description, product_id, region, specification, and topic. The "region" variable is highlighted with a red box. The right side of the screen shows the properties panel for "Dropdown field 4" with various configuration options like "Label text" set to "Region" and "Selected value" set to "region".

Dropdown field 4: Selected value

Select binding type / Data and Variables / Page variable

Page variables are variables that are only available on the page on which they have been defined.

currency

origin

pricing

product_description

product_id

region

specification

topic

Label text: Region

Selected value: region

Placeholder text: region

Selected label: region

Show value as label: True

Repeat with: Not repeated

Component display name: Dropdown field 4

TREE

- PAGE LAYOUT
 - Title 2
 - Input field 1

Build Apps will show you the selected option, Click “SAVE”.

The screenshot shows a configuration interface for a dropdown field. At the top right, it says "Edit binding for" followed by "Dropdown field 1: Placeholder text" and "text". A "BACK" button is located at the top left. Below the title, there's a breadcrumb navigation: "Select binding type" / "Data and Variables" / "Page variable". A descriptive text states: "Page variables are variables that are only available on the page on which they have been defined." A "READ MORE" button is present. The main section is titled "Select page variable" and contains a dropdown menu with the option "region". An "Optional preview value" input field contains the placeholder text "Lorem ipsum". At the bottom right is a blue "SAVE" button.

BACK

Edit binding for
Dropdown field 1: Placeholder text
text

Select binding type / Data and Variables / Page variable

Page variables are variables that are only available on the page on which they have been defined.

READ MORE

Select page variable

region

Optional preview value ⓘ

SAVE

Now we will assigned the “Selected Label”, using the same process as above to find our Region Page Variable. Click on the X for Selected Label.

The screenshot shows the SAP Build Apps interface for the "EDAWithBuild" app. The left sidebar contains categories like CORE, DISPLAY, FORMS, LAYOUT, and LISTS, each with sub-components. The main area displays a UI component titled "EDA with Build Apps" with several input fields: "Topic" (dropdown field), "Region" (dropdown field), "Price" (input field), "Origin" (input field), "Specification" (input field), and a "Button" (button). On the right, the "VARIABLES" tab is selected, showing properties for a "Dropdown field 1". The "Selected value" property is set to "region", and the "Selected label" property is highlighted with a red box around its icon. The "Component display name" is set to "TREE".

Click "Data and Variables"

Edit binding for
Dropdown field 1: Selected label
text

Select binding type



No value



Data and Variables



Formula



Component properties

Click on “Page variable”

Edit binding for
Dropdown field 1: Selected label
text

[Select binding type](#) / Data and Variables

Dynamic bindings whose values can change while the app is running.

[READ MORE](#)

 App variable

Available on all pages in your app

 Page variable

Available only on the current page

 Data variable

Access data from a backend

Click on Region

The screenshot shows the SAP Build Apps interface for a project named "PriceChangeApp". A modal dialog is open, titled "Edit binding for Dropdown field 4: Selected value". The dialog displays a list of page variables under the heading "Select page variable". The variable "region" is selected and highlighted with a red box. Other variables listed include currency, origin, pricing, product_description, product_id, specification, and topic. To the right of the modal, the properties panel for "Dropdown field 4" is visible, showing settings for label text ("Region"), option list ("Custom list (3 items)"), selected value ("region"), placeholder text ("region"), selected label ("region"), show value as label ("True"), repeat with ("Not repeated"), and component display name ("Dropdown field 4").

Edit binding for
Dropdown field 4: Selected value

any text

Select binding type / Data and Variables / Page variable

Page variables are variables that are only available on the page on which they have been defined.

READ MORE

Select page variable

| | |
|---------------------|------|
| currency | text |
| origin | text |
| pricing | text |
| product_description | text |
| product_id | text |
| region | text |
| specification | text |
| topic | text |

Add logic to **DROPDOWN FIELD 4**

PROPERTIES **STYLE** **LAYOUT**

Label text Region

Option list Custom list (3 items)

Selected value region

Placeholder text region

Selected label region

Show value as label True

Repeat with Not repeated

Component display name Dropdown field 4

TREE

PAGE LAYOUT

- o Title 2
- Input field 1

Once "Region" is selected, click on "SAVE"

BACK

Dropdown field 1: Selected label

binding for
text

Select binding type / Data and Variables / Page variable

Page variables are variables that are only available on the page on which they have been defined.

[READ MORE](#)

Select page variable

region

Optional preview value ⓘ

Lorem ipsum

SAVE

Make sure you set “Show value as label” to “True”.

Your properties should now look similar to what is displayed below in RED, double check that you have formatted them properly.

Welcome to AEM

Topic
try-me

Region
Lorem ipsum

Product ID
Lorem ipsum

Product Description
Lorem ipsum

Price
Lorem ipsum

Currency
Lorem ipsum

Origin
Lorem ipsum

Specification
Lorem ipsum

Publish Price Change

PROPERTIES STYLE LAYOUT

Label text ABC Region

Option list [] Custom list (3 items)

Selected value region

Placeholder text region

Selected label region

Show value as label True

Repeat with Not repeated

Component display name TREE

PAGE LAYOUT

- Title 2
- Input field 1
- Dropdown field 1
- Dropdown field 2
- Dropdown field 3

Now you will follow the same process to populate the next dropdown:

Label – Product ID

Option List

- 51001
- 52005
- 62009
- 95004

And bind all the various fields to your product_id page variable as before for region.

And also set “Show value as label” to “True”.

Double check that your configuration looks like this:

The screenshot shows the SAP Build Apps interface for a 'PriceChangeApp'. The UI canvas displays a form with fields for Topic, Region, Product Description, Price, and Currency. A red arrow points from the 'Region' dropdown to the configuration panel on the right. The configuration panel for 'Dropdown field 2' is highlighted with a red box and contains the following settings:

- Label text**: ABC Product ID
- Option list**: Custom list (4 items)
- Selected value**: product_id
- Placeholder text**: product_id
- Selected label**: product_id
- Show value as label**: True
- Repeat with**: Not repeated
- Component display name**: Dropdown field 2

The left sidebar shows various component categories: DISPLAY (Spinner, Image, Text, Title, Icon, Card), FORMS (Button, Input field, Password field, Slider field, Checkbox field, Dropdown field), LAYOUT (Container, Scroll view, Row), and LISTS.

Now you will follow the same process to populate the next dropdown:

Label – Product Description

Option List

- Milk Chocolate
- Apple
- Bread
- Cheese

Again, bind all the various fields to your product_description page variable as before for region & productid.

And also set “Show value as label” to “True”.

Double check that your configuration looks like this:

The screenshot shows the SAP Build Apps interface for a 'PriceChangeApp' project. The left sidebar contains toolbars for CORE, BY ME, and INSTALLED components, along with sections for MARKETPLACE, DISPLAY, FORMS, LAYOUT, and LISTS. The UI CANVAS tab is active, displaying a mobile preview for an iPhone 15/15 Pro (393px) at 100% zoom level. The preview shows a form titled 'Welcome to AEM' with fields for Topic (try-me), Region (Lorem ipsum), Product ID (Lorem ipsum), Product Description (dropdown field), Price (Lorem ipsum), and Currency (Lorem ipsum). The Product Description field is highlighted with a red box. On the right, the VARIABLE panel is open, showing properties for 'Dropdown field 3'. A red box highlights the 'Label text' field, which contains 'ABC Product Description'. Another red box highlights the 'Selected value' field, which contains 'product_description'. The 'Repeat with' field is set to 'Not repeated'.

Properties for Dropdown field 3

- Label text: ABC Product Description
- Option list: Custom list (4 items)
- Selected value: product_description
- Placeholder text: product_description
- Selected label: product_description
- Show value as label: True
- Repeat with: Not repeated

Now you will follow the same process to populate the next dropdown:

Label – Currency

Option List

- GBP
- EUR
- USD
- YEN

Again, bind all the various fields to your currency page variable as before for region & productid.

And also set “Show value as label” to “True”.

Double check that your configuration looks like this:

EDA with Build Apps

Topic
Lorem ipsum

Region
Lorem ipsum

Product ID
Lorem ipsum

Product Description
Lorem ipsum

Price
Lorem ipsum

Currency
Dropdown field 4

Origin
Lorem ipsum

Specification
Lorem ipsum

Button

PROPERTIES **STYLE** **LAYOUT**

Label text ABC Currency

Option list Custom list (2 items)

Selected value currency

Placeholder text currency

Selected label currency

Show value as label True

Repeat with Not repeated

Component display name TREE

PAGE LAYOUT

- Title 1
- Input field 1
- Dropdown field 1
- Dropdown field 2
- Dropdown field 3
- Input field 2
- Dropdown field 4

Just one small change left
and the form is complete,
change the label on the
Button to read “Publish
Price Change”.

EDA with Build Apps

Topic

Region

Product ID

Product Description

Price

Currency

Origin

Specification

Publish Price Change

Properties Style Layout

Label ABC Publish Price Change

Disabled False

Repeat with Not repeated

Component display name Button 1

ADVANCED PROPERTIES

TREE

PAGE LAYOUT

- o Title 1
- Input field 1
- Dropdown field 1
- Dropdown field 2
- Dropdown field 3

Congratulations, your form is now complete and we must implement the logic behind the “Publish Price Change” button.

Step 3 – Adding Logic to our Form to Publish an Event

At the moment, SAP Build Apps does not natively support the publication of Events, however they do support the use of Javascript Libraries. So in this next section, we will be taking advantage of this feature to load the libraries necessary to publish an event to the Advanced Event Mesh. When you click on the button, you will see text at the bottom of the screen “Add Logic to Button1”, click this section to reveal another panel.

EDA with Build Apps

Topic
Lorem ipsum

Region
Lorem ipsum

Product ID
Lorem ipsum

Product Description
Lorem ipsum

Price
Lorem ipsum

Currency
Lorem ipsum

Origin
Lorem ipsum

Specification
Lorem ipsum

Publish Price Change

Properties **Style** **Layout**

Label ABC Publish Price Change

Disabled False

Repeat with Not repeated

Component display name Button 1

ADVANCED PROPERTIES

TREE

- PAGE LAYOUT
 - o Title 1
 - Input field 1
 - Dropdown field 1
 - Dropdown field 2
 - Dropdown field 3
 - Input field 2
 - Dropdown field 4
 - Input field 3
 - Input field 4
 - o Button 1

Add logic to **BUTTON 1** ⓘ

DISPLAY

| | | |
|---------|-------|------|
| Spinner | Image | Text |
| Title | Icon | Card |

FORMS

| | | |
|--------------|----------------|----------------|
| Button | Input field | Password field |
| Slider field | Checkbox field | Dropdown field |

LAYOUT

| | | |
|-----------|-------------|-----|
| Container | Scroll view | Row |
|-----------|-------------|-----|

LISTS

LOGIC CANVAS

CORE BY ME INSTALLED Component: Button 1

MARKETPLACE

Delete record

DEVICE

| | | |
|------------|------------|--------------|
| QR/barcode | Take photo | GPS location |
|------------|------------|--------------|

UTILITY

| | |
|-------|--------------|
| Delay | If condition |
|-------|--------------|

ADVANCED

| | | |
|------------|---------------|---------------|
| JavaScript | Receive event | Trigger event |
|------------|---------------|---------------|

The Logic Canvas interface shows a workflow step. An 'Event' block labeled 'Component tap' is connected to a teal-colored 'JS' block. The 'JS' block has a small red 'X' icon in its top right corner.

EDA with Build Apps

Topic
Lorem ipsum

Region
Lorem ipsum

Product ID
Lorem ipsum

Product Description
Lorem ipsum

Price
Lorem ipsum

Currency
Lorem ipsum

On the left hand side, you will scroll down to the “Advanced” group of widgets and you will drag and drop the Javascript widget onto the pallet as in the screenshot.

The screenshot shows the SAP Build Apps interface. At the top, there's a navigation bar with UI CANVAS, LAUNCH, THEME, NAVIGATION, DATA, AUTH, and HELP. Below the navigation is a toolbar with REVERT and SAVE buttons. The main area is divided into two sections: UI CANVAS and LOGIC CANVAS.

UI CANVAS: This section displays a form titled "EDA with Build Apps". It contains several input fields: Topic (text input), Region (dropdown), Product ID (dropdown), Product Description (dropdown), Price (text input), and Currency (dropdown). To the left of the form is a sidebar with categories like CORE, DISPLAY, FORMS, LAYOUT, LISTS, and LOGIC CANVAS. Under CORE, there are links for MARKETPLACE, Delete record, and DEVICE (Scan QR/barcode, Take photo, GPS location).

LOGIC CANVAS: This section shows a logic flow diagram. A blue rounded rectangle labeled "Component: Button 1" is connected to a green rounded rectangle labeled "JS". An event "Component tap" is associated with the button component. To the right of the logic canvas is a tree view showing the structure of the page layout, including components like Title 1, Input field 1, and various dropdown fields.

Now we will convert that JS component into a Flow function. With the Javascript widget selected, you should see on the upper right side a little “+” symbol. Click the Plus symbol to add a flow function that we will now configure. You will notice that your JS object has magically changed as per the next screenshot.

EDAWithBuild

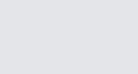
[Home page](#)
[UI CANVAS](#) [LAUNCH](#) [THEME](#) [NAVIGATION](#) [DATA](#) [AUTH](#) [HELP](#)
[REVERT](#)[SAVE](#)
[CORE](#) [BY ME](#) [INSTALLED](#)

iPhone 15/15 Pro (393px)

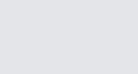
Zoom level: 100% ▾

[VIEW](#) [VARIABLES](#)
[PROPERTIES](#) [OUTPUTS](#)

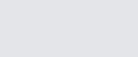
DISPLAY



FORMS



LAYOUT



LISTS

LOGIC CANVAS

[CORE](#) [BY ME](#) [INSTALLED](#)

Component: Button 1



You have not created any composite flow functions for this app yet.

You can do so by selecting one or more flow functions and clicking Turn selection into a new function from the properties panel.

Nothing selected.

Select a node, resource or variable to view its properties.

TREE

PAGE LAYOUT

- o Title 1
- Input field 1
- Dropdown field 1
- Dropdown field 2
- Dropdown field 3
- Input field 2

Voila, you have a Flow Function to work with. Now “double click” on the Flow Function.

Double Click on the “Input” Box (the flow function graphic) to reveal a screen where you will specify the input parameters.



From this next screen, you will be defining the same properties that we defined on the form and the same properties we defined for page variables.

- topic Value Type “Text”
- region Value Type “Text”
- origin Value Type “Text”
- specification Value Type “Text”
- production_description Value Type “Text”
- product_id Value Type “Text”
- pricing Value Type “Text”
- currency Value Type “Text”

Click on “Edit Properties” to start.

Then click the X next to “input1” to remove the default property that is created.

Flow function properties

Name
Flow function 5

Extra label
You can show input values with the mustache syntax
e.g. Set {{variableName}} to {{inputValue}}

Intro (short description)
Max 140 characters
Type here

Description
Type a description for this flow function. You can use Markdown syntax for formatting and paragraphing.

Explainer video URL
Type here

Category

INPUTS

input1 

Title
Type title for this value

Description
Type a description for this value. You can use Markdown syntax for formatting and paragraphing.

Value type

Value is required

Example values

Type a text 

Flow function properties

Name
Flow function 7

Extra label
You can show input values with the mustache syntax
e.g. Set {{variableName}} to {{inputValue}}

Intro (short description)
Max 140 characters
Type here

Description
Type a description for this flow function. You can use Markdown syntax for formatting and paragraphing.

Explainer video URL
Type here

Category

INPUTS

ADD NEW PROPERTY 

CANCEL

OK

After you remove the default property, you will then enter “topic” as your first property to be added.

*****DO NOT CLICK OK****** instead click the small plus sign to the right of the property input field to add the next field.

Once you click the “+” button, the property “topic” has been added and you can repeat the process for all of the other fields. Here’s the full list again:

- topic Value Type “Text”
- region Value Type “Text”
- origin Value Type “Text”
- specification Value Type “Text”
- production_description Value Type “Text”
- product_id Value Type “Text”
- pricing Value Type “Text”
- currency Value Type “Text”

Only click the OK, once all the properties have been added.

The screenshot shows the 'Flow function properties' dialog box. At the top right is a close button (X). Below it are sections for 'Name' (Flow function 7), 'Extra label' (with placeholder text for Mustache syntax), 'Intro (short description)' (Max 140 characters), 'Description' (placeholder text about Markdown syntax), 'Explainer video URL' (Type here), and 'Category' (dropdown menu). The 'INPUTS' section is expanded, showing a property named 'topic' with a value type of 'Text' (selected), a checked 'Value is required' checkbox, and an 'Example values' section with a 'Add example value' button. A red oval highlights the 'ADD NEW PROPERTY' button at the bottom of the inputs list, which contains the placeholder 'Type property name'. At the very bottom are 'CANCEL' and 'OK' buttons.

Once you click “OK” you will be back to this screen. From here you will double click on the JS Box to enter the Javascript code that will be executed when the button is pressed and also define the variables that the Javascript will use.

The screenshot shows the SAP Build Apps interface. The top navigation bar includes the SAP logo, 'Build Apps', and tabs for 'UI CANVAS', 'LAUNCH', 'THEME', 'NAVIGATION', 'DATA', 'AUTH', and 'HELP'. Below the navigation is a breadcrumb trail: 'EDAWithBuild' > 'Home page'. The main area is divided into two sections: 'UI CANVAS' on the left and 'LOGIC CANVAS' on the right.

UI CANVAS:

- CORE:** Contains 'Button', 'Input field', 'Form field', 'Slider field', 'Checkbox field', and 'Dropdown field'.
- LAYOUT:** Contains 'Container', 'ScrollView', and 'Row'.
- LISTS:** Contains 'List divider', 'List section header', 'Toggle list item', 'Search bar', 'Icon list item', and 'List item'.
- MARKETPLACE:** Contains 'Large image list item' and 'Image list item'.
- PRIMITIVES:** Contains a placeholder 'Placeholder'.

The 'UI CANVAS' section displays a form titled 'EDA with Build Apps' with fields for Topic, Region, Product ID, Product Description, Price, and Currency, all populated with 'Lorem ipsum'.

LOGIC CANVAS:

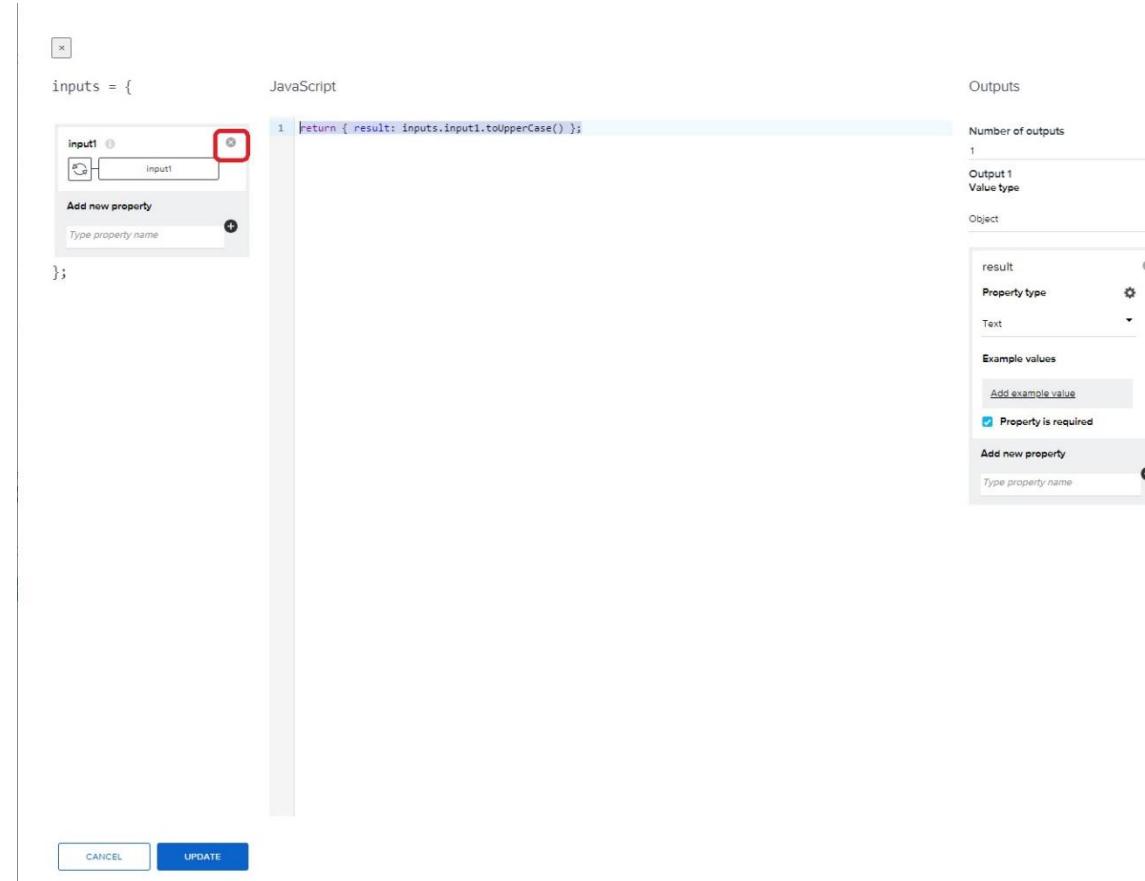
The 'LOGIC CANVAS' section shows a flow function named 'Flow function 7' in isolation mode. It contains a sequence of nodes: 'input' -> 'JS' (highlighted with a red box) -> 'output 1'. The 'JS' node is connected to the 'output 1' node. The 'JS' node has an 'Edit Properties' button, an 'Advanced' dropdown, and an 'Exit Isolation' button.

Now that you have the JS window open, we need to do 2 things:

1. Recreate all the same variables as above. Prior to creating the variables, **REMOVE** the default input parameter by clicking the little “x” in red.

- topic Value Type “Text”
- region Value Type “Text”
- origin Value Type “Text”
- specification Value Type “Text”
- product_description Value Type “Text”
- product_id Value Type “Text”
- pricing Value Type “Text”
- currency Value Type “Text”

You will do the same process as before, type the variable name into the field and hit the little “+” sign.



2. Paste the JavaScript Code Below into the JavaScript window and replace what is there.

**** JAVASCRIPT BEGINS****

```
const { topic, pricing, region, product_id, product_description, origin, specification, currency } = inputs;

const { session } = self.solace;

const destination = createDestination(topic, 'topic');

publishMessage(destination, pricing, region, product_id, product_description, origin, specification, currency);

function createDestination(name, type) {

    if (type === 'queue') {

        return solace.SolclientFactory

            .createDurableQueueDestination(name);

    }

    if (type === 'topic') {

        return solace.SolclientFactory

            .createTopicDestination(name);

    }

}

function publishMessage(publishDestination, pricing, region, product_id, product_description, origin, specification, currency) {

    const message = solace.SolclientFactory.createMessage();

    const payload = {

        pricing: pricing,

        region: region,

        product_id: product_id,

        product_description: product_description,

        origin: origin,

    }

    publishDestination.publishMessage(message, payload);
}
```

```
specification: specification,  
currency: currency  
};  
message.setDestination(publishDestination);  
message.setBinaryAttachment(JSON.stringify(payload));  
session.send(message);  
}  
  
****JAVASCRIPT ENDS*****
```

Once the variables are created and the Javascript is pasted in, you should have a screen that looks like this.

You should have 8 variables defined on the left as input to the function.

BUT, if you notice there is nothing bound to the function at this point. Now we will select each of the variables and bind them to the flow function variables you defined in the previous step. You will start by clicking on the ABC that is highlighted. Yours will be unbound to start.

The screenshot shows the SAP Build Apps interface for a project named "PriceChangeApp". The main area displays a logic function with the following code:

```
inputs = {  
    currency: currency,  
    product_id: product_id,  
    specification: specification,  
    product_description: product_description,  
    topic: topic  
};  
  
const { topic, pricing, region, product_id, product_description, origin, specification, currency } = inputs;  
const session = self.solace;  
  
const destination = createDestination(topic, 'topic');  
publishMessage(destination, pricing, region, product_id, product_description, origin, specification, currency);  
  
function createDestination(name, type) {  
    if (type === 'queue') {  
        return solace.SolclientFactory  
            .createDurableQueueDestination(name);  
    }  
    if (type === 'topic') {  
        return solace.SolclientFactory  
            .createTopicDestination(name);  
    }  
}  
  
function publishMessage(publishDestination, pricing, region, product_id, product_description, origin, specification, currency) {  
    const message = solace.SolclientFactory.createMessage();  
  
    const payload = {  
        pricing: pricing,  
        region: region,  
        product_id: product_id,  
        product_description: product_description,  
        origin: origin,  
        specification: specification,  
        currency: currency  
    };  
}
```

The "topic" variable is highlighted with a red box and labeled "ABC Value". On the left, a sidebar lists various UI components like Spinner, Title, and Button. On the right, the "Outputs" panel shows one output named "Output 1" with a value type of "Object". A modal window titled "Add new property" is open, prompting for a "Type property name".

Select “Flow Function Input”

Your flow function will likely be labelled Flow Function 1 (***(I've done 7 iterations 😊 ***)).

Edit binding for
Function
unknown type

Select binding type

| | | |
|--|---|---|
|  No value |  Static value |  Data and Variables |
|  Formula |  Component properties |  Output value of another node |
|  Flow function input | | |

Click the Flow Function to reveal the variables available.

The screenshot shows a user interface for editing a binding. At the top right, it says "Edit binding for **Function** unknown type". On the left, there's a "BACK" button with a left arrow icon. Below the title, the text "Select binding type / Flow function input" is displayed. A descriptive note below it states: "References an input argument passed to a flow function. Only available inside a flow function's component template editor." A dropdown menu is open, titled "Select flow function input". It contains one visible option: "Flow function 7". To the right of the dropdown, there is a small ellipsis (...).

From the list, select the input field being mapped, in this case “topic”.

The screenshot shows the SAP Build Apps interface for a "PriceChangeApp". On the left, there's a sidebar with various UI component icons like Spinner, Title, Button, Slider field, Container, etc. The main area displays a JavaScript code editor with the following content:

```
inputs = {  
    topic: topic,  
    origin: origin,  
    region: region,  
    pricing: pricing,  
    currency: currency,  
    product_id: product_id  
};
```

Below the code editor, there are "CANCEL" and "UPDATE" buttons. A modal window titled "Edit binding for Function" is open, showing the following details:

- Select binding type:** Flow function input
- References:** An input argument passed to a flow function. Only available inside a flow function's component template editor.
- Select flow function input:** Flow function 1
- Select flow function input field:** A list of fields from the code:
 - topic (highlighted with a red border)
 - origin
 - pricing
 - product_description
 - product_id
 - region
 - specification
- Outputs:** Shows "Number of outputs: 1", "Output 1 Value type: Object", and an "Add new property" button.

You have a chance to validate here before pressing “SAVE”

 BACK

Edit binding for

Function

unknown type

Select binding type / Flow function input

References an input argument passed to a flow function. Only available inside a flow function's component template editor.

Select flow function input

Flow function 7



Select flow function input field

topic



SAVE

After saving the first one, it should look like this and if you hover over the topic you should see that it is mapped to the flow function input.

The screenshot shows a configuration interface for a flow function. On the left, under 'inputs', there are five fields: currency, product_id, specification, product_description, and topic. The 'topic' field is highlighted with a red oval. In the center, the 'JavaScript' tab displays the following code:

```
1 const { topic, pricing, region, product_id, product_description, origin, specification, currency } = inputs;
2 const { session } = self.solace;
3
4 const destination = createDestination(topic, 'topic');
5 publishMessage(destination, pricing, region, product_id, product_description, origin, specification, currency);
6
7 function createDestination(name, type) {
8   if (type === 'queue') {
9     return solace.SolclientFactory
10    .createDurableQueueDestination(name);
11  }
12  if (type === 'topic') {
13    return solace.SolclientFactory
14    .createTopicDestination(name);
15  }
16 }
17
18 function publishMessage(publishDestination, pricing, region, product_id, product_description, origin, specification, currency) {
19   const message = solace.SolclientFactory.createMessage();
20
21   const payload = {
22     pricing,
23     region,
24     product_id,
25     product_description: product_description,
26     origin,
27     specification: specification,
28     currency: currency
29   };
30 }
```

On the right, under 'Outputs', it shows 'Number of outputs: 1'. Under 'Output 1', 'Value type' is set to 'Object'. A button 'Add new property' is visible. At the bottom, there are 'CANCEL' and 'UPDATE' buttons.

Once completed, it should look like this but one final update. Each time you add a JS widget, it adds a default Output Parameter which you can see highlighted in RED. You will need to remove this.

The screenshot shows a configuration interface for a JS widget. On the left, there is a list of inputs:

- topic
- region
- origin
- specification
- productDescription
- productId
- productPrice
- currency

Below this list is an "Add new property" section with a text input field labeled "Type property name" and a plus sign button.

In the center, there is a "JavaScript" code editor containing the following code:

```
1 const { topic, productPrice, region, productId, productDescription, origin, specification, currency } = input
2 const { session } = self.solace;
3 const destination = createDestination(topic, 'topic');
4 publishMessage(destination, productPrice, region, productId, productDescription, origin, specification, currency);
5 function createDestination(name, type) {
6   if (type === 'queue') {
7     return solace.SolclientFactory
8       .createDurableQueueDestination(name);
9   }
10  if (type === 'topic') {
11    return solace.SolclientFactory
12      .createTopicDestination(name);
13  }
14}
15 function publishMessage(publishDestination, productPrice, region, productId, productDescription, origin, specification, currency) {
16   const message = solace.SolclientFactory.createMessage();
17   const payload = {
18     pricing: productPrice,
19     region: region,
20     product_id: productId,
21     product_description: productDescription,
22     origin: origin,
23     specification: specification,
24     currency: currency
25   };
26   message.setDestination(publishDestination);
27   message.setBinaryAttachment(JSON.stringify(payload));
28   session.send(message);
29 }
30
```

On the right, there is an "Outputs" panel:

- Number of outputs: 1
- Output 1: Value type: Object
- result:
 - Property type: Text
 - Example values: Add example value (button), Property is required (checkbox checked)
 - Add new property: Type property name (input field)

At the bottom, there are "CANCEL" and "UPDATE" buttons.

This is the final result that you should see.

The screenshot shows a configuration interface with three main sections: Inputs, JavaScript, and Outputs.

Inputs: A list of variables with their types and values. The variables are:

- currency (String): currency
- product_id (String): product_id
- specification (Object): specification
- product_description (String): product_description
- topic (String): topic

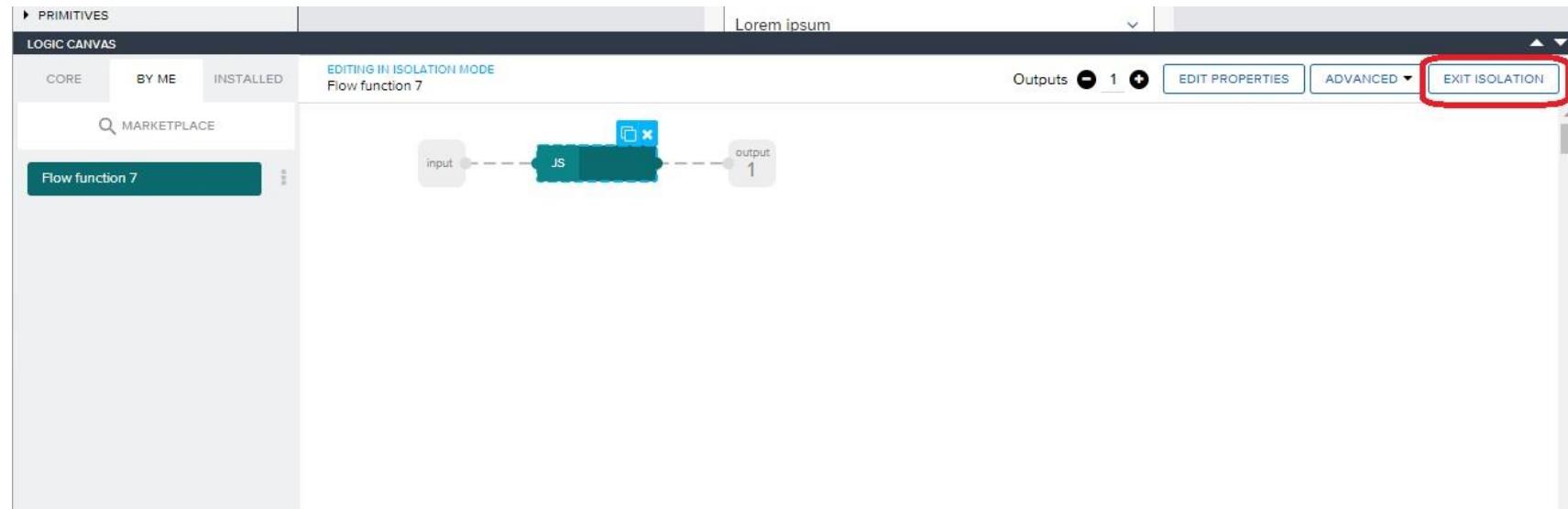
JavaScript: A block of code defining a function to create destinations and another to publish messages.

```
1 const { topic, pricing, region, product_id, product_description, origin, specification, currency } = inputs;
2 const { session } = self.solace;
3
4 const destination = createDestination(topic, 'topic');
5 publishMessage(destination, pricing, region, product_id, product_description, origin, specification, currency);
6
7 function createDestination(name, type) {
8     if (type === 'queue') {
9         return solace.SolclientFactory
10            .createDurableQueueDestination(name);
11     }
12     if (type === 'topic') {
13         return solace.SolclientFactory
14            .createTopicDestination(name);
15     }
16 }
17
18 function publishMessage(publishDestination, pricing, region, product_id, product_description, origin, specification, currency) {
19     const message = solace.SolclientFactory.createMessage();
20
21     const payload = {
22         pricing,
23         region,
24         product_id,
25         product_description,
26         origin,
27         specification,
28         currency
29     };
30 }
```

Outputs: A section for defining output properties. It shows one output named "Output 1" with a value type of "Object". There is also a button to "Add new property".

Buttons: At the bottom left are "CANCEL" and "UPDATE" buttons.

After pressing “UPDATE”, you will return to the JS function. Press the “Exit ISOLATION” button.



From here, we have 2 things to do:

- 1) Bind all of the Flow Function Inputs to our page variables (click on the flow function to reveal this properties pane).
- 2) Connect our “Component Tap” event to the Flow Function by simply dragging a line between the dot that is highlighted by the red square to the dot on the flow function.

The screenshot shows the SAP Build Apps interface. On the left, there's a sidebar with categories like CORE, BY ME, and INSTALLED, and sections for MARKETPLACE, LAYOUT, LISTS, and PRIMITIVES. The main area displays a form titled "EDA with Build Apps" with fields for Topic, Region, Product ID, Product Description, Price, and Currency, each containing placeholder text "Lorem ipsum". Below this is the "LOGIC CANVAS" section, which contains a "Component tap" event connected to a "Flow function 7". A red box highlights the connection point between the event and the flow function. To the right, the "PROPERTIES" pane is open for "Flow function 7", showing inputs for various variables like topic, region, origin, specification, productionDescription, productId, productPrice, and currency, all marked as required and currently empty. A red box also highlights this properties pane.

EDAWithBuild

Home page

UI CANVAS LAUNCH THEME NAVIGATION DATA AUTH HELP

REVERT SAVE

VIEW VARIABLES

INPUTS

topic* ⓘ Required
No value

region* ⓘ Required
No value

origin* ⓘ Required
No value

specification* ⓘ Required
No value

productionDescription* ⓘ Required
No value

productId* ⓘ Required
No value

productPrice* ⓘ Required
No value

currency* ⓘ Required
No value

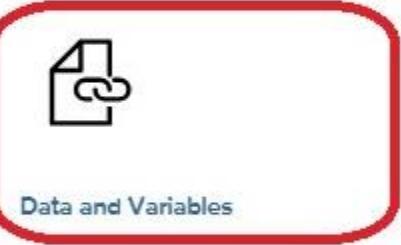
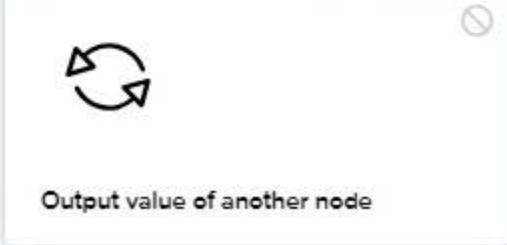
ADVANCED

TREE

Let's start by mapping the first variable. Click the "X" just underneath the topic input. Select Data and Variables.

Edit binding for
Flow function 7
text

Select binding type

| | | |
|---|--|---|
| A_{B_C} Static text |  Data and Variables |  Formula |
|  Component properties |  Output value of another node | |

Select "Page variable"

Edit binding for
Flow function 7
text

Select binding_type / Data and Variables

Dynamic bindings whose values can change while the app is running.

[READ MORE](#)

| | | |
|--|---|---|
|  App variable Available on all pages in your app |  Page variable Available only on the current page |  Page parameter Pass data to a page when it's opened |
|  Data variable Access data from a backend |  Translation variable Translate text to the current language |  Sensor variable Access device sensor information |
|  System variable Access device and platform information | | |

Select the page variable for “topic”

Edit binding for
Publish
text

[BACK](#)

[Select binding type](#) / [Data and Variables](#) / Page variable

Page variables are variables that are only available on the page on which they have been defined.

[READ MORE](#)

Select page variable

| | |
|---------------------|------|
| currency | text |
| origin | text |
| pricing | text |
| product_description | text |
| product_id | text |
| region | text |
| specification | text |
| topic | text |

The system will prompt you before you click “SAVE”

Edit binding for
Flow function 7
text

[Select binding type](#) / [Data and Variables](#) / Page variable

Page variables are variables that are only available on the page on which they have been defined.

[READ MORE](#)

Select page variable

topic

SAVE

Now if you hover over the topic, SAP Build will display which page variable is currently bound to that input

The screenshot shows the SAP Build UI for a 'PriceChangeApp'. On the left, there's a sidebar with categories like CORE, DISPLAY, FORMS, LAYOUT, and LISTS, each containing various UI components with small icons. The main area displays a form titled 'Product ID' with several input fields, each containing placeholder text ('Lorem ipsum'). Below the inputs is a blue button labeled 'Publish Price Change'. To the right of the form is a 'VARIABLES' panel. A red arrow points from the 'topic' variable in the 'OPTIONAL INPUTS' section of the panel towards the 'Product ID' field in the form. The 'topic' variable is bound to the 'Product ID' input field. The 'VARIABLES' panel also lists other optional inputs: 'region', 'origin', 'specification', 'product_description', 'product_id', and 'pricing', each with its respective binding icon.

Repeat the process for all of the remaining variables.

You should now have a screen that looks like this. Remember to connect the Component Tap Event to the flow function.

The screenshot shows the SAP Build Apps interface. The top navigation bar includes UI CANVAS, LAUNCH, THEME, NAVIGATION, DATA, AUTH, and HELP. The left sidebar has sections for CORE, BY ME, and INSTALLED, with MARKETPLACE and a search icon. The main area displays a mobile view for an iPhone 15/15 Pro (393px) at 100% zoom level. The UI consists of several input fields: Price (text), Currency (dropdown), Origin (text), Specification (text), and a blue "Publish Price Change" button. To the right is the PROPERTIES panel for a flow function named "Publish". The "VIEW" tab is selected. The "OPTIONAL INPUTS" section contains fields for topic, region, origin, specification, product_description, product_id, and pricing, each with a small icon and a placeholder text box. A red curved arrow points from the "topic" input field in the PROPERTIES panel back to the "topic" input field in the UI canvas. Below the UI canvas is the LOGIC CANVAS, which shows a flow diagram starting with an EVENT node labeled "Component tap", followed by a FLOW node labeled "Publish*", and an END node. A red circle highlights the connection between the "Component tap" event and the "Publish*" flow node. The bottom left sidebar includes NAVIGATION, OPEN PAGE, and NAVIGATE BACK options.

Hit Save at the top right when you are done.

Step 4 – Loading the Javascript Libraries

In order to be able to publish the events as per the Javascript code, we need to load the library that is used by AEM.

We will do this when the application starts. At the top left, you will see the HomePage is highlighted.

The screenshot shows the SAP Build Apps interface. At the top, there's a navigation bar with 'SAP Build Apps' logo, user profile, and various menu items like UI CANVAS, LAUNCH, THEME, NAVIGATION, DATA, AUTH, and HELP. Below the navigation is a toolbar with 'CORE', 'BY ME', 'INSTALLED', 'iPhone 15/15 Pro (393px)', 'Zoom level: 100%', and a search bar labeled 'MARKETPLACE'. The main area is divided into two sections: 'UI CANVAS' and 'LOGIC CANVAS'. The 'UI CANVAS' section displays a form titled 'EDA with Build Apps' with fields for Topic, Region, Product ID, Product Description, and Price. The 'LOGIC CANVAS' section shows a flow diagram starting from an 'EVENT Component tap' node, which has a 'FLOW Flow function 1*' connection pointing to another node. On the right side, there are 'PROPERTIES' and 'OUTPUTS' tabs, and a 'TREE' panel showing a hierarchical structure of page layout components: PAGE LAYOUT, Title 1, Input field 1, Dropdown field 1, Dropdown field 2, Dropdown field 3, Input field 2, Dropdown field 4, Input field 3, Input field 4, and Button 1. A status message at the bottom right says 'Nothing selected. Select a node, resource or variable to view its properties.'

Click On Homepage to reveal the screen for all pages that are available in the application.

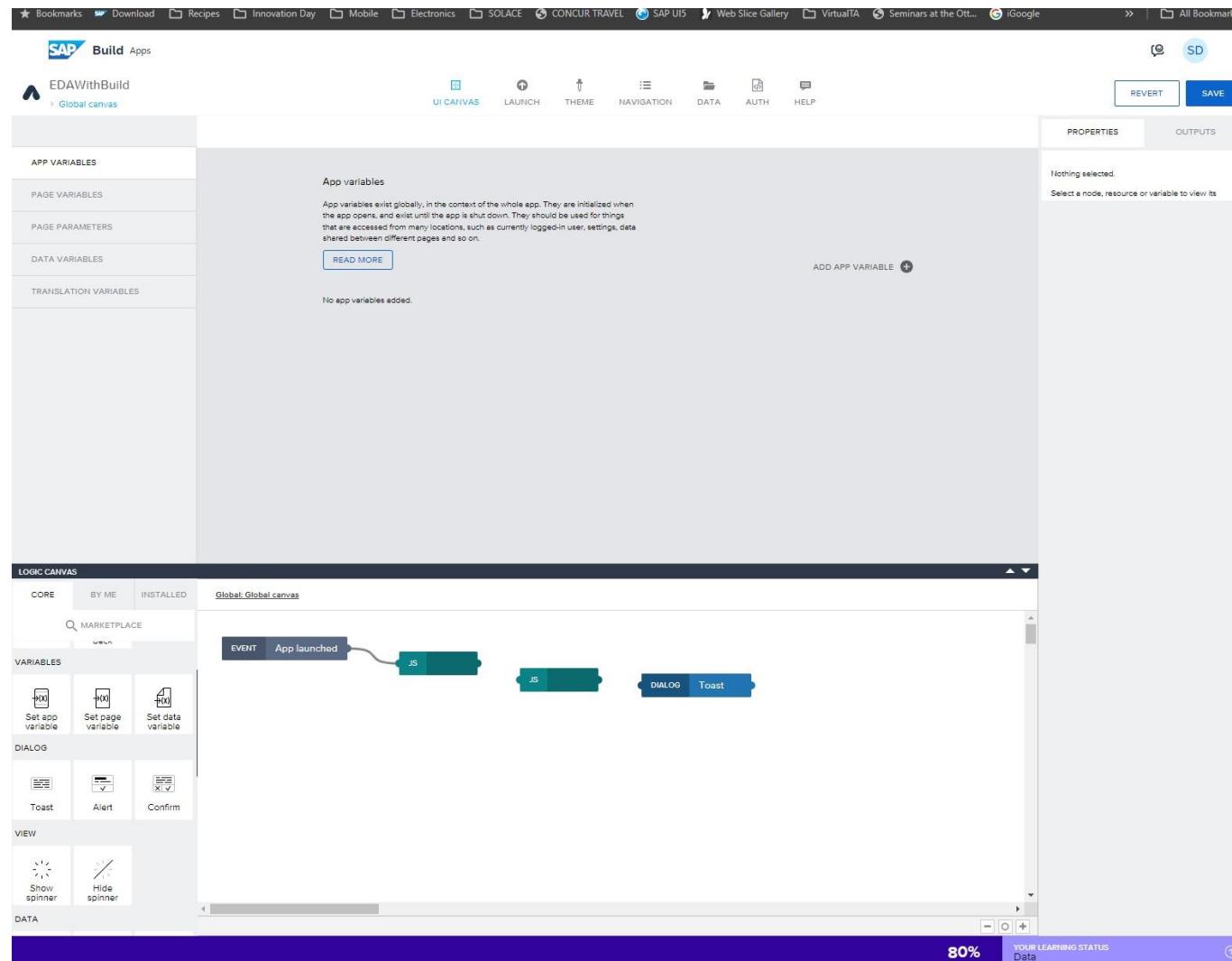
Then click on the Global Canvas.

The screenshot shows the SAP Build Apps interface. At the top, there is a header with the SAP logo and the text "Build Apps". Below the header, the page title is "EDAWithBuild" and the current page is "Home page". There are two buttons: "UI CANVAS" and "LAUNCH".

The main content area is titled "PAGES". It contains two cards:

- Home page**: This card displays several input fields with placeholder text "Lorem ipsum":
 - Topic
 - Region
 - Product ID
 - Product Description
 - Price
 - Currency
- Global canvas**: This card contains the text "Global canvas does not have a user interface."

From the component palette in the lower left, click on “Core”, then let's start by dragging 2 JavaScript widgets and 1 Toast widget onto the canvas as show.



From here, first make sure your App Launched event is connected by dragging a line from App Launchpad Event to the first JavaScript box.

Then single click the first Javascript box to reveal the property editor on the right. Enter “Load Solace Library” for the Name (under Advanced).

The screenshot shows the SAP Build Apps interface. On the left, there's a sidebar with sections like APP VARIABLES, PAGE VARIABLES, PAGE PARAMETERS, DATA VARIABLES, and TRANSLATION VARIABLES. The main area is titled "LOGIC CANVAS" and shows a workflow: "EVENT App launched" connects to a "JS Load Solace Library" block, which then connects to another "JS" block. The "Load Solace Library" block is highlighted with a red rectangle. On the right, the "PROPERTIES" tab is selected for the "Load Solace Library" variable. The "Name" field contains "Load Solace Library" and is also highlighted with a red rectangle. The "Logic" dropdown shows "function2". Below the properties, the "ADVANCED" section is expanded, showing the "Name" field again with its value "Load Solace Library".

Now double click on the Load Solace Library so we can copy paste the following code.

****JAVASCRIPT BEGINS****

```
if (self.solace) {  
    return;  
}  
  
const { exports } = self;  
  
self.exports = {};  
  
await import('https://cdn.jsdelivr.net/npm/solclientjs@10.15.0/lib-browser/solclient.js');  
  
self.solace = self.exports.solace;  
  
self.exports = exports;  
  
****JAVASCRIPT ENDS****
```

There are no inputs for this JavaScript but we do have to make one small change. Notice that an output structure has been created. You need to remove this.

VA

EV

EP

AV

NSI

LAM

ES

pple

o

er

Inputs

JavaScript

outputs = {

 Add new property

 Type property name

};

1 if (self.solace) {
2 return;
3 }
4 const { exports } = self;
5 self.exports = {};
6 await import('https://cdn.jsdelivr.net/npm/solclientjs@10.15.0/lib-browser/solclient');
7 self.solace = self.exports.solace;
8 self.exports = exports;

Number of outputs

1

Output 1

Value type

Object

result

Property type

Text

Example values

Add example value

Property is required

Add new property

Type property name

CANCEL UPDATE

The screenshot shows a configuration interface for a script component. On the left, there's a sidebar with various tabs like VA, EV, EP, AV, NSI, LAM, ES, and Apple. The main area has sections for 'Inputs' (JavaScript code defining an object with an input1 property), 'JavaScript' (the actual script body), and 'Outputs'. The 'Outputs' section shows a single output named 'Output 1' with a value type of 'Object'. A red box highlights the configuration for 'Output 1', which includes a 'result' field, a 'Property type' dropdown set to 'Text', an 'Example values' section with a 'Add example value' button and a checked 'Property is required' checkbox, and a 'Add new property' section with a 'Type property name' input. At the bottom, there are 'CANCEL' and 'UPDATE' buttons.

Your finished JavaScript should look like this.

The screenshot shows a code editor interface with a central code preview area and two side panels: 'Inputs' on the left and 'Outputs' on the right.

Inputs: A text input field containing the following code:

```
inputs = {  
};
```

A sidebar on the right of the input field includes a button labeled "Add new property" and a placeholder "Type property name".

JavaScript: The central code preview area displays the following code:

```
1 if (self.solace) {  
2   return;  
3 }  
4  
5 const { exports } = self;  
6 self.exports = {};  
7 await import('https://cdn.jsdelivr.net/npm/solclient@10.15.0/lib-browser/solclient.js');  
8  
9 self.solace = self.exports.solace;  
10 self.exports = exports;
```

Outputs: A panel on the right showing the configuration for the output:

- "Number of outputs": 1
- "Output 1": Value type is set to "Object".
- "Add new property" button and "Type property name" placeholder are present.

Now we need to convert the remaining the JavaScript widget into a flow function. If you recall from earlier, we single click the JS widget and then on the upper right, we hit the small “+” sign. Here, we will be creating the 4 parameters we need to specify the connection to the Advanced Event Mesh. YES...we know they should not be hard coded but for this purpose it will be fine 😊

The screenshot shows the EDAWithBuild application interface. The top navigation bar includes UI CANVAS, LAUNCH, THEME, NAVIGATION, DATA, AUTH, and HELP, along with REVERT and SAVE buttons. The left sidebar has sections for APP VARIABLES (PAGE VARIABLES, PAGE PARAMETERS, DATA VARIABLES, TRANSLATION VARIABLES) and LOGIC CANVAS (CORE, BY ME, INSTALLED). The LOGIC CANVAS section displays a workflow:

```

graph LR
    Start((App launched)) --> JS1[JS Load Solace Library]
    JS1 --> JS2[JS]
    JS2 --> Dialog[Toast]
    
```

The JS2 node is highlighted with a red rectangle. The Properties panel on the right shows the logic titled "untitled" with the identifier "Logic: function2". It includes tabs for PROPERTIES and OUTPUTS, and an ADVANCED section where a "Edit" icon is circled in red.

Now we will have a new Flow Function and you will double click it.

The screenshot shows the SAP Build Apps interface. At the top, there's a navigation bar with tabs like UI CANVAS, LAUNCH, THEME, NAVIGATION, DATA, AUTH, and HELP. On the right, there are buttons for REVERT, SD, and SAVE. Below the navigation bar, there's a sidebar with sections for APP VARIABLES, PAGE VARIABLES, PAGE PARAMETERS, DATA VARIABLES, and TRANSLATION VARIABLES. The main area is titled "App variables" and contains a brief description of what app variables are. A "READ MORE" button is available. On the right, the "PROPERTIES" panel is open, showing details for a flow function named "Flow function 8" with logic "Logic: Flow function 8". It also includes sections for OPTIONAL INPUTS and ADVANCED, along with some icons. At the bottom, the "LOGIC CANVAS" section shows a workflow starting with an "EVENT App launched" trigger, followed by a "JS Load Solace Library" action, then a "FLOW Flow function 8" step (which is highlighted with a red box), and finally a "DIALOG Toast" action. The canvas also has tabs for CORE, BY ME, and INSTALLED, and a MARKETPLACE search bar.

Double click on the Flow Function

Then double click on the “input”.



As before, we will start by clicking the X to remove the default parameter.

The "Flow function properties" dialog box is open, showing the "INPUTS" section. It contains a single input field named "input1" with a title placeholder "Type title for this value". A red box highlights the "X" button next to the title field. Other sections visible include "Name" (Flow function 8), "Description", "Explainer video URL", "Category", and "Value type".

Now we will add the following parameters:

url

vpnName

username

password

***Remember, add the parameters one at a time and each time hit the “+”, not the OK button until the end.

After all the parameters are entered and you Hit OK, you will be returned to the JS function. Single Click it to access the properties and name it “Connect to Solace”.

Flow function properties

Name
Flow function 8

Extra label
You can show input values with the mustache syntax
e.g. Set `[[variableName]]` to `[[inputValue]]`

Intro (short description)
Max 140 characters
Type here

Description
Type a description for this flow function. You can use Markdown syntax for formatting and paragraphing.

Explainer video URL
Type here

Category

INPUTS

ADD NEW PROPERTY
`brokerUrl`

CANCEL OK

The screenshot shows a software interface with a central workspace and various toolbars and panels.

Properties Panel:

- Connect to Solace**
Logic: function2
- ADVANCED**
- Name:** Connect to Solace
- Operations:** (+) (-)

App variables Panel:

App variables exist globally, in the context of the whole app. They are initialized when the app opens, and exist until the app is shut down. They should be used for things that are accessed from many locations, such as currently logged-in user, settings, data shared between different pages and so on.

READ MORE

No app variables added.

ADD APP VARIABLE (+)

Flow Editor:

EDITING IN ISOLATION MODE
Flow function 8

Outputs: 1
Edit Properties Advanced Exit Isolation

The flow consists of an input node connected to a JS node labeled "Connect to Solace". The JS node has one output port labeled "output".

Now we will double click on the JavaScript Widget so we can enter the connection code.

Paste the following code into Javascript area

****JAVASCRIPT BEGINS****

```
debugger;

if (self.solace === undefined) {
    return;
}

async function createSolaceSession() {
    const factoryProps = new solace.SolclientFactoryProperties();
    factoryProps.profile = solace.SolclientFactoryProfiles.version10_5;
    solace.SolclientFactory.init(factoryProps);
    const session = solace.SolclientFactory.createSession(inputs);
    const sessionUp = new Promise(rs =>
        session.once(solace.SessionEventCode.UP_NOTICE, rs));
    //session.on(solace.SessionEventCode.MESSAGE, onMessage);
    session.connect();
    self.solace.session = session;
    await sessionUp;
    debugger;
    return {};
}

await createSolaceSession();

****JAVASCRIPT CODE ENDS****
```

The screenshot shows a code editor interface with three main sections: JavaScript code, JSON inputs, and JSON outputs.

JavaScript:

```

inputs = {
  input1: {
    type: "text"
  }
};

async function createSolaceSession() {
  const factoryProps = new solace.SolclientFactoryProperties();
  factoryProps.profile = solace.SolclientFactoryProfiles.version10_5;
  solace.SolclientFactory.init(factoryProps);

  const session = solace.SolclientFactory.createSession(inputs);
  const sessionUp = new Promise(rs =>
    session.once(solace.SessionEventCode.UP_NOTICE, rs));
  //session.on(solace.SessionEventCode.MESSAGE, onMessage);
  session.connect();

  self.solace.session = session;
  await sessionUp;
  debugger;
  return {};
}

await createSolaceSession();

```

Inputs:

```
inputs = {
```

Outputs:

Number of outputs: 1

Output 1: Value type: Object

result: Property type: Text

Example values: Add example value (checkbox checked)

Property is required: checked

Add new property: Type property name

Before moving on to create the inputs for the function, you need to remove the output for “result” highlighted in red.

Now under the inputs, we will create the four properties that we created earlier for (match the case):

- url
- vpnName
- userName
- password

****DO NOT FORGET TO REMOVE THE DEFAULT INPUT, YOU SHOULD HAVE 4 WHEN YOU ARE DONE****

Your flow function should look like this.

The screenshot shows a configuration dialog for a flow function. On the left, there is a list of input properties:

- password**: A password input field.
- url**: A URL input field labeled `brokerUrl`.
- vpnName**: A VPN name input field labeled `brokerVpn`.
- userName**: A user name input field labeled `username`.
- Add new property**: A placeholder for adding new properties.

On the right, the **JavaScript** tab displays the following code:

```
1  debugger;
2
3 v if (self.solace === undefined) {
4   return;
5 }
6 v async function createSolaceSession() {
7   const factoryProps = new solace.SolclientFactoryProperties();
8   factoryProps.profile = solace.SolclientFactoryProfiles.version10_5;
9   solace.SolclientFactory.init(factoryProps);
10  const session = solace.SolclientFactory.createSession(inputs);
11  const sessionUp = new Promise(rs =>
12    session.once(solace.SessionEventCode.UP_NOTICE, rs));
13  //session.on(solace.SessionEventCode.MESSAGE, onMessage);
14  session.connect();
15  self.solace.session = session;
16  await sessionUp;
17  debugger;
18  return {};
19}
20await createSolaceSession();
21
```

At the bottom of the dialog are two buttons: **CANCEL** and **UPDATE**.

But, one thing missing....your inputs are not bound to anything. You need to bind each of those to the correspond **Flow Function Variables**....do you remember how to do it so it looks like this?

The screenshot shows a configuration dialog with two main sections: 'inputs' and 'JavaScript'.

Inputs:

```
inputs = {  
    password: "password",  
    url: "brokerUrl",  
    vpnName: "brokerVpn",  
    userName: "username",  
};
```

JavaScript:

```
1 debugger;  
2  
3 if (self.solace === undefined) {  
4     return;  
5 }  
6 async function createSolaceSession() {  
7     const factoryProps = new solace.SolclientFactoryProperties();  
8     factoryProps.profile = solace.SolclientFactoryProfiles.version10_5;  
9     solace.SolclientFactory.init(factoryProps);  
10    const session = solace.SolclientFactory.createSession(inputs);  
11    const sessionUp = new Promise(rs =>  
12        session.once(solace.SessionEventCode.UP_NOTICE, rs));  
13    //session.on(solace.SessionEventCode.MESSAGE, onMessage);  
14    session.connect();  
15    self.solace.session = session;  
16    await sessionUp;  
17    debugger;  
18    return {};  
19 }  
20 await createSolaceSession();  
21
```

At the bottom, there are 'CANCEL' and 'UPDATE' buttons.

So, now that you have it bound...where are the values specified for those 4 parameters?

Remember to hit "Update" when you are done.

Return back to the global canvas by clicking "Exit Isolation".

If you single click the Flow Function, the 4 parameters will appear and this is where you will enter the address of your broker. Once you enter each of those parameters, we just have a few things left to do.

First let's make sure that the various widgets are connected like in the flow below. You will also notice that we have changed the name on the Flow Function to "Connect to Solace".

Second, make to populate the 4 properties in the red box or you won't connect 😊

Copy these values from your "Connect" tab from your AEM broker service.

The screenshot shows the SAP AEM Broker Manager interface for a broker named "eu1". The "Connect" tab is selected. The "View by:" dropdown is set to "Protocol".

Connection Details:

- Username: solace-cloud-client
- Password: (redacted)
- Message VPN: eu1
- Secured Web Messaging Host: wss://mr-ud0mj6ykvij.messaging.solace.cloud:443
- Public Internet: wss://mr-ud0mj6ykvij.messaging.solace.cloud:443

Client Libraries:

- Solace JavaScript API (JavaScript) - Get Started
- Solace Node.js API (Node.js) - Get Started

SAP Build Apps

EDAWithBuild

Global canvas

UI CANVAS LAUNCH THEME NAVIGATION DATA AUTH HELP HISTORY

APP VARIABLES PAGE VARIABLES PAGE PARAMETERS DATA VARIABLES TRANSLATION VARIABLES

App variables

App variables exist globally, in the context of the whole app. They are initialized when the app opens, and exist until the app is shut down. They should be used for things that are accessed from many locations, such as currently logged-in user, settings, data shared between different pages and so on.

READ MORE ADD APP VARIABLE +

No app variables added.

LOGIC CANVAS

CORE BY ME INSTALLED Global: Global canvas

MARKETPLACE

NAVIGATION

Open page Navigate back

EVENT App launched → JS Load Solace Library → FLOW → Connect to Solace* → DIALOG Toast

Properties Outputs

Connect to Solace Logic: Flow function 8

INPUTS

- brokerUrl* ⓘ Required No value
- brokerVpn* ⓘ Required No value
- username* ⓘ Required No value
- password* ⓘ Required No value

ADVANCED

ADD REMOVE SEARCH

```
graph LR; Start([App launched]) --> JS[JS Load Solace Library]; JS --> Flow[FLOW]; Flow --> Connect[Connect to Solace*]; Connect --> Dialog[DIALOG]; Dialog --> Toast[Toast]
```

Last step, let's configure the Message Toast. Click on the Dialog Box.

On the right of the screen, enter your connection message "Connected"

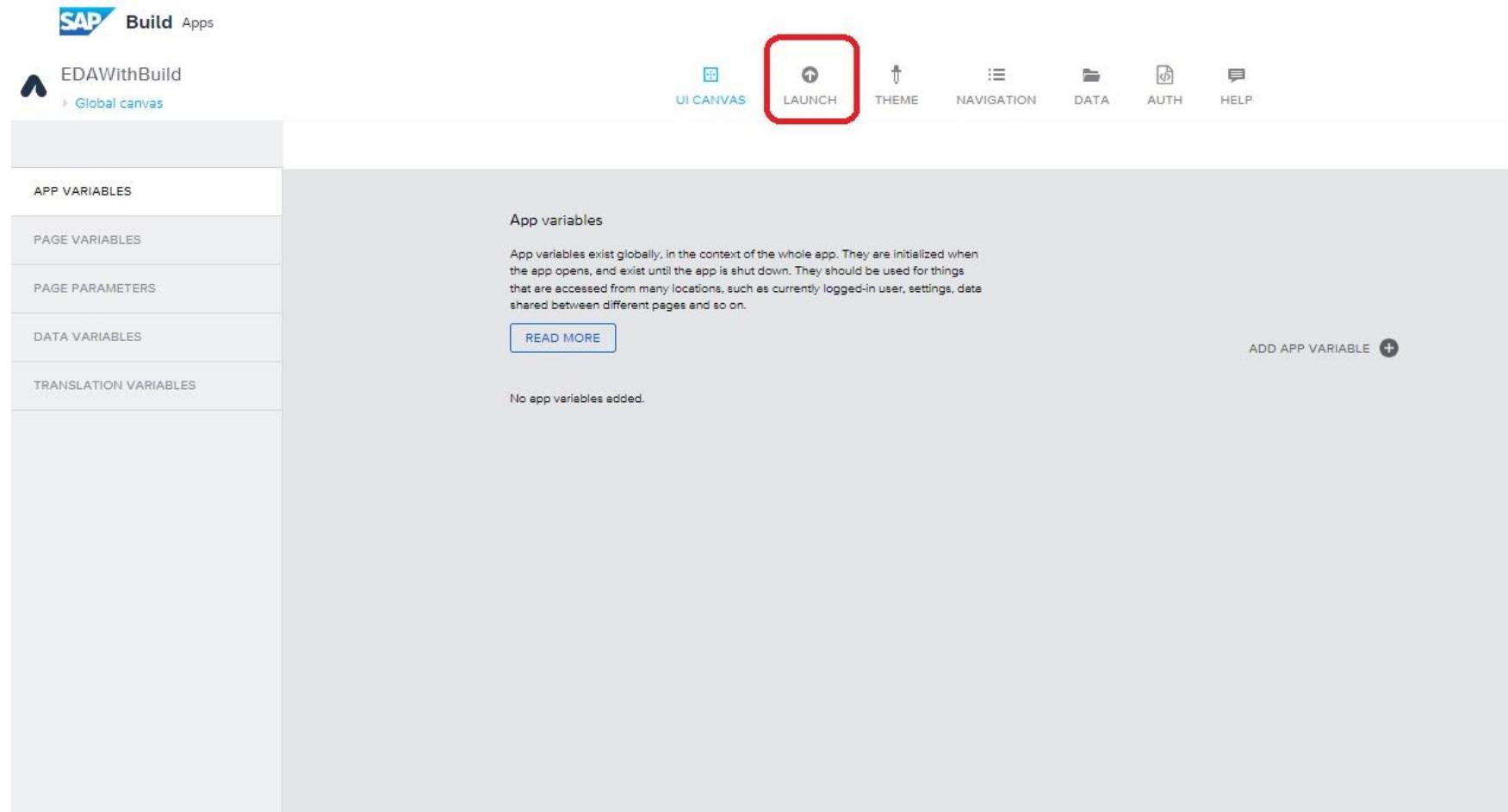
The screenshot shows the AEM UI Canvas editor interface. At the top, there are navigation tabs: UI CANVAS, LAUNCH, THEME, NAVIGATION, DATA, AUTH, and HELP. On the right side, there are 'PROPERTIES' and 'OUTPUTS' sections. The 'PROPERTIES' section shows a component named '(Toast)' with the type 'Logic.Toast'. It has a description: 'Shows a non-blocking toast dialog with the given message. The dialog will disappear automatically, unless otherwise configured.' Under 'INPUTS', there is a field 'Toast message*' containing the value 'Connected to AEM'. Below it are sections for 'OPTIONAL INPUTS' and 'ADVANCED' settings, each with a '+' button. In the bottom left, a workflow diagram titled 'Global: Global.canvas' is displayed. It starts with an 'EVENT' node labeled 'App launched', which triggers a 'JS' node labeled 'Load Solace Library'. This leads to a 'FLOW' node labeled 'Connect to Solace*', which then triggers a 'DIALOG' node labeled 'Toast Connected to AEM'. The 'UI CANVAS' tab is selected at the top.

Hit Save.

WOOHOO, you should now have an application that is ready to start sending EVENTS.

Step 5 – Previewing the Application

In order to pre-view our application, we need to click the “Launch” application icon at the top of the page.



The screenshot shows the SAP Build Apps interface. At the top, there is a navigation bar with the SAP Build Apps logo, a project name "EDAWithBuild", and a "Global canvas" link. To the right of the project name are several icons: UI CANVAS (blue square), LAUNCH (red square with a white play button icon), THEME, NAVIGATION, DATA, AUTH, and HELP. The "LAUNCH" icon is highlighted with a red box. On the left, there is a sidebar with sections: APP VARIABLES, PAGE VARIABLES, PAGE PARAMETERS, DATA VARIABLES, and TRANSLATION VARIABLES. The main content area is titled "App variables". It contains a descriptive text about app variables, a "READ MORE" button, and an "ADD APP VARIABLE" button with a plus sign. Below this, it says "No app variables added."

We won't be building the application today, only Previewing the Web Version, Click on the “Open preview portal” button.

Launch



Preview your app

Preview your app in a web browser, as well as by using native iOS and Android preview apps. Log in by entering a temporary PIN code or pasting a Login URL.

[Open preview portal →](#)

Build your app

Create mobile builds of your app for distribution, as well as build web applications to be hosted under the appgyverapp domain. Alternatively, you can download the web build as a ZIP file.

[Open build service →](#)

We will be using the Web Preview.

Preview on web

[Open web preview](#)

Preview on your device

Follow these steps to preview apps on your device

1. Open SAP Build Apps app on your device
2. Choose “Show PIN code”
3. Input the PIN code here:

PIN code

Type here

[Confirm pin](#)

Download preview apps

Android

iOS



Now you will just click on the app that we have built “EDAWithBuild” to see it running.

SAP Build Apps

</>

PriceChangeApp

Updated: 3 Hours ago

OPEN

</>

EDAWithBuild

Updated: 17 Hours ago

OPEN

If you've done all of your clicking properly 😊 , you should see a message toast pop-up that says you are connected to Solace. This is great news, the library has been loaded.

Now you can fill out the fields and publish your first message from SAP Build Apps....nice work 😊

Home page

EDA with Build Apps

Topic

Region

Product ID

Product Description

Price

Currency

Origin

Specification

Publish Price Change

Connected to Solace

Step 6 – Testing the Application

Now that you have the application up and running, we want to make sure that when you Publish the Price Change, it's being received on the Advanced Event Mesh.

From the console of the Advanced Event Mesh, you will be navigating to the “Try Me” tab.

Once you open the console, click the “Cluster Manager” on the left side of the screen and then select the Broker you have created during the session. I will be selecting the Montreal broker.

The screenshot shows the SAP Cluster Manager Services interface. On the left, a sidebar lists various management tools: Mission Control (Cluster Manager, highlighted with a red box), Mesh Manager, Event Portal, Designer, Catalog, Runtime Manager, KPI, Event Insights, and SAP DEMO. The main area displays a grid of services under 'All services'. The services listed are:

- aws ap1 eks-ap-southeast-1a
- cn1 aks-eastasia
- eu1 sapdemo-eks-eu-central-1
- MontrealBroker-10.1 eks-ca-central-1a (highlighted with a red box)
- Broker 250 Christian Holtfurth (Running)
- Broker 250 Christian Holtfurth (Running)
- Broker 250 Christian Holtfurth (Running)
- Broker 100 Scott Dillon (Running)
- MyFirstService aks-canadacentral
- sa1 eks-af-south-1b
- SAPWS eks-eu-central-1a
- us1 gke-gcp-us-central1-a
- Broker 100 Justin Le (Running)
- Broker 250 Christian Holtfurth (Running)
- Broker 100 Daniel Brunold (Running)
- Broker 250 Christian Holtfurth (Running)

At the bottom, there is a large empty box with a plus sign, and a footer indicating page 1 of 8 results.

Now you will select the far right option for “Try Me” and press the “Connect” button which has been pre-populated with the credentials for your broker. We will also use the default topic, just hit the “Subscribe” button.

SAP | MontrealBroker-10.1

Status Connect Manage Monitoring Configuration Try Me! Try Me!

Open Broker Manager ...

Send and receive messages using the Solace Web Messaging API

AMQP: 0
MQTT: 0
SMF: 44
REST: 0
Web: 11

Queues: 67
Topic Endpoints: 3

Direct received: 0
Direct sent: 0
Guaranteed received: 223597
Guaranteed sent: 1212299

Publisher

Endpoint Connectivity: Public Endpoint (Public Internet)

HTML SCSS Babel Result EDIT ON CODEPEN

1 Establish connection ✗

Connect Disconnect show advanced settings

2 Publish

Topic: try-me

Message: Hello world!

binary text

Publish

Resources 1x 0.5x 0.25x Rerun

Subscriber

Endpoint Connectivity: Public Endpoint (Public Internet)

HTML SCSS Babel Result EDIT ON CODEPEN

1 Establish connection ✗

Connect Disconnect show advanced settings

2 Subscribe

Add Topic: try-me

Subscribe

Subscribed Topics: try-me

Add a topic subscription to begin receiving messages.

Messages:

Resources 1x 0.5x 0.25x Rerun

Once you have successfully connected, you will see that both the Connect Button and the Subscribe Button are greyed out and we have 1 Topic listed for the “Subscribed Topics”. Now this web based client is listening for Events that have the “try-me” topic.

Subscriber

Endpoint Connectivity

Public Endpoint (Public Internet)

HTML SCSS Babel

Result

EDIT ON
CODEPEN

- 1 Establish connection ⚙

Connect

Disconnect

show advanced settings

- 2 Subscribe

Add Topic

Subscribe

Subscribed Topics

try-me



Messages

So, let's try it with the app. You can see we have defaulted the Topic to "try-me". For this first test, just leave the default topic in place and use the drop down fields to populate where necessary like below and hit the "Publish Price Change".

The screenshot shows a web browser window with a form titled "Home page". The browser's address bar displays "useastbpa-p8xih...". The form contains the following fields:

- Topic:** try-me
- Region:** North America
- Product ID:** 62009
- Product Description:** Apple
- Price:** 5.0
- Currency:** USD
- Origin:** Ontario
- Specification:** Great Apples

A large blue button at the bottom of the form is labeled "Publish Price Change".

If your application is configured properly, you have just successfully published to the AEM service and also subscribed from it. Congratulations.

Subscriber

Endpoint Connectivity

Public Endpoint (Public Internet)

HTML SCSS Babel Result EDIT ON CODEPEN

Subscribe

Subscribed Topics

try-me X

Messages

4:34:55 PM [Topic try-me] X

```
7b 22 70 72 69 63 69 6e 67 22 3a 22 35 2e 30 22  
2c 22 72 65 67 69 6f 6e 22 3a 22 4e 6f 72 74 68 20  
41 6d 65 72 69 63 61 22 2c 22 70 72 6f 64 75 63  
74 5f 69 64 22 3a 22 36 32 30 30 39 22 2c 22 70  
72 6f 64 75 63 74 5f 64 65 73 63 72 69 70 74 69 6f  
6e 22 3a 22 41 70 70 6c 65 22 2c 22 6f 72 69 67  
69 6e 22 3a 22 4f 6e 74 61 72 69 6f 22 2c 22 73 70  
65 63 69 66 69 63 61 74 69 6f 6e 22 3a 22 47 72  
65 61 74 20 41 70 70 6c 65 73 22 2c 22 63 75 72  
72 65 6e 63 79 22 3a 22 55 53 44 22 7d
```

```
{"pricing": "5.0", "region": "North America", "product_id": "62009", "product_description": "Apples", "currency": "USD"}
```

Resources 1x 0.5x 0.25x Rerun

