

Using Subflows to Manage Complexity

Table of Contents

- 1 Objective
- 2 Do the following task items
 - 2.1 Design base requirements of the subflow
 - 2.2 Implement the subflow
 - 2.2.1 Create subflow for API call
 - 2.2.2 Define Input interface for subflow
 - 2.2.3 Define success and error response for subflow
 - 2.3 Implement a test flow for subflow
 - 2.3.1 Test the flow calling the subflow
 - 2.4 Replace API call in registration flow with subflow
 - 2.4.1 Change the display pages to use new display values
 - 2.4.2 Test your registration flow that invokes the subflow

1 Objective

As your flow evolves and grows, you may find it beneficial to externalize some functionality that's often reused or complex to its own flow. For example, if your flow needed to connect to an API that isn't available as a native connector, you could build your CRUD operations in a new flow that could be leveraged by many.

Another use case would be a very large flow with many paths that are not always utilized. Breaking it up into a set of logical components that make it easier to manage and understand would be beneficial. Essentially apply the concepts of object and functional development to your flows to make them easier for you and others to support.

Tip

All the number instruction steps in each section must be followed to successfully complete this exercise. Any bullet points or images are for example only unless called out in a numbered step.

2 Do the following task items

As an example of creating a subflow in this exercise you will take the existing API call to the Deck Of Cards API into a subflow. This could be potential functionality that could be expanded for other flows and could even lead to expanded use of the API for other use cases.

2.1 Design base requirements of the subflow

In this section you will review the base requirements for the subflow. What functionality being moved into the flow and the interface for calling and what needs to be returned to continue the process in the calling flow.

You may want to refer to the best practices for [subflows](#) document. If not at this time remember for future work in DaVinci.

- 1 The HTTP component that calls the API will need to be moved
- 2 The subflow will need to return the data required by the calling flow
 - Card URL
 - Card suit
 - Card value
- 3 Pass in the number of cards required.
 - For our use case only 1 is needed, but understanding how to pass data to a subflow is an important concept.
 - This also allows for potential use in future cases that could leverage this API service.
- 4 Continue with the next section.

2.2 Implement the subflow

Now that you have covered our goals for building this subflow it is time to implement the solution.

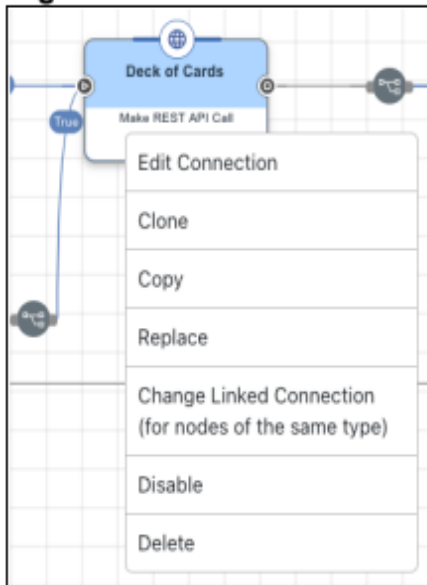
In this section you will need to copy an existing node from one flow to another flow. This functionality only works with browsers that support JavaScript copy and paste. If you are using Firefox you will need to switch to Chrome at least to get the node copied. Of course you can simply just open it in two tabs and copy and paste configuration, but lets take the easy path.

- 1 If you are not using the **Chrome** browser, open it for this section.
 - 1.1 Login into your PingOne account
 - 1.2 Select the **Intro DaVinci Training** environment.
 - 1.3 Click on the **PingOne DaVinci** service to open the DaVinci console.
 - 1.4 Select **Flows** from the left menu.
- 2 Click **Progressive Registration** flow to open it, you may want to create a backup before you start.
- 3 Continue with the next section.

2.2.1 Create subflow for API call

In this section you will define the subflow that will handle the API call to the deck of cards services. This will be done by using the node from the existing flow as your starting point.

- 1 **Right-mouse click** on the **Deck of Cards** node in your flow.



- 2 Select **Copy** from the menu to copy this node configuration.
- 3 Click **Flows** on the left menu to see your list of flows. If prompted that you have unsaved changes just click **Yes** button. You have not made any changes to this flow at this time that you want to save.
- 4 Click **Add Flow** button to create a new flow.
- 5 Click **Blank Flow** option at the pop-up.
- 6 **Type** the following for **Name**; it is a best practice to start your subflows with Subflow to easily find them in the list
 - **Subflow Deck of Cards API**
- 7 **Type** the following for **Description**
 - **Calls the deck of cards API returning card details from the API**

 A screenshot of a dialog box titled 'Add Flow' with a close button (X) in the top right corner. The dialog has two text input fields. The first field contains the text 'Subflow Deck of Cards API'. The second field contains the text 'Calls the deck of cards API returning card details from the API'. At the bottom right of the dialog is a blue button labeled 'Create'.

- 8 Click **Create** button to create the flow.
- 9 On your blank canvas **right-mouse click** to paste the node from the other flow.

10 Click **Paste Nodes** menu item.



Note

If you get a read permissions error in chrome or a chromium based browser you may need to go into settings in the browser and under site permissions allow *console.pingone.com* access to the clipboard.

11 Click **Save** button to save your flow at this stage.

12 Continue with the next section.

2.2.2 Define Input interface for subflow

The flow is going to accept a value as input, how many cards are needed. For now it will not return more than one card, so the number of cards input is to give us future uses for this subflow in other main flows.

The output from the flow will depend on if the flow is successful or fails and you will deal with that in the next section.

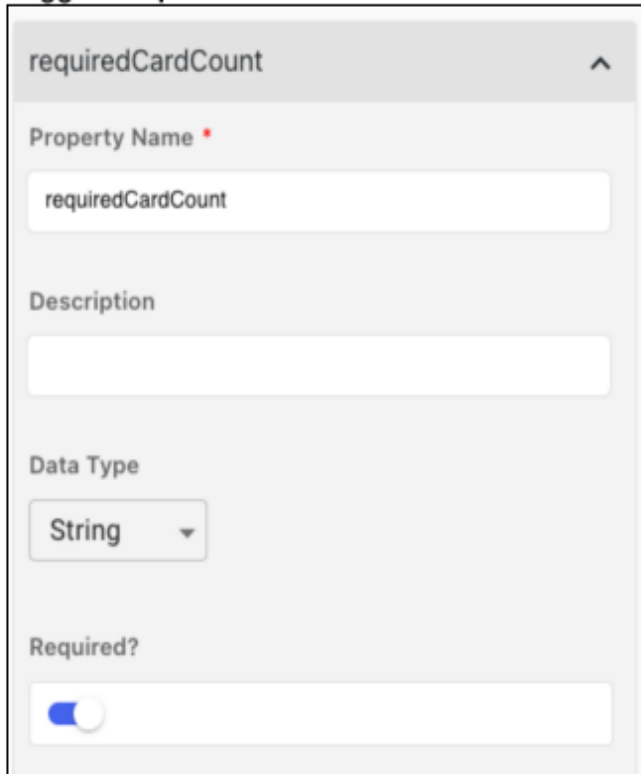
At this point you can switch back to your regular browser if you were no using Chrome, the node has been copied into your new flow.

1 Click **Input Schema** in the top right had corner to define the inputs into this flow.

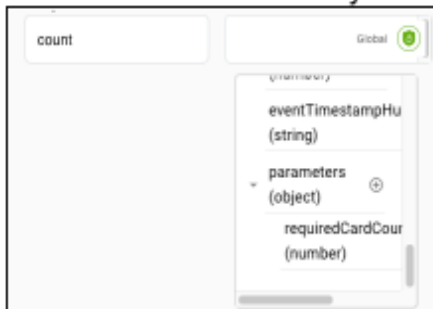


2 Click **Add** button to an input object.

- 3 Enter the following information for the field:
 - 3.1 **Property Name** as
 - **requiredCardCount**
 - 3.2 **Data Type** as **String** from the dropdown list.
 - 3.3 Toggle **Required** on.



- 3.4 Click **Save** button then **Cancel** button to close the dialog.
- 4 Click the **Deck of Cards** node to open its configuration.
- 5 For the query parameter **count** remove the **1** from the field.
- 6 Click the **{}** to add a variable reference, select **Global** from the list.
 - Input schema parameters for the flow becomes a global variable.
- 7 **Scroll** down the list until you see parameters, see screenshot



- 8 Select **requiredCardCount** as the variable.
- 9 Click **Apply** button to save your changes.
- 10 Continue with the next section.

2.2.3 Define success and error response for subflow

The flow will return a JSON success or failure depending on how the API responds to your request.

The success will return the card image URL, card suit, and card value. These are the three values the calling flow require at this time.

The failure will return the details of the error.

- 1 From the **Deck of Cards** node drag a line to add a new node to the right of the existing node.
- 2 Select **Http** connector as the new node with a decision of **All Triggers True**
- 3 From the **Action Decision** node drag a new line to add a new connector.
- 4 Select **Http** connector as the new node with a decision of **Any Trigger False**
- 5 On the **All Triggers True** path click the **Http** node to configure it.
- 6 Click **Send Success JSON Response** capability.
- 7 Under **Additional Fields in the Response** click **+ Field** button to add a new field.

Additional Fields in the Response

Title	Key (Optional)	Value
	Select or Type	Enter value

- 7.1 Enter the following value for the field **Key**, when you press enter the variable dialog will open.

- **doc_card_url**

- 7.2 In the **Data Type** field select **String** from the dropdown list.
- 7.3 Accept the remaining defaults in the **Add Variable** dialog, click **Create** button.
- 7.4 In the **Value** field select **{}** to add variable reference to **png** from the **Deck of Cards** node.

Key (Optional) doc_card_url ...

Value {

Enter value

image (string)

images (object)

svg (string)

png (string)

value (string)

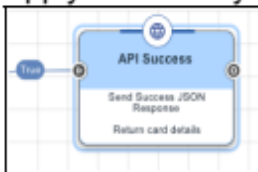
suit (string)

- 8 Under **Additional Fields in the Response** click **+ Field** button to add a new field.
- 8.1 Enter the following value for the field **Key**, when you press enter the variable dialog will open.
- **doc_card_value**
- 8.2 In the **Data Type** field select **String** from the dropdown list.
- 8.3 Accept the remaining defaults in the **Add Variable** dialog, click **Create** button.

- 8.4 In the **Value** field select **{}** to add variable reference to **value** from the **Deck of Cards** node.
- 9 Under **Additional Fields in the Response** click **+ Field** button to add a new field.
 - 9.1 Enter the following value for the field **Key**, when you press enter the variable dialog will open.
 - **doc_card_suit**
 - 9.2 In the **Data Type** field select **String** from the dropdown list.
 - 9.3 Accept the remaining defaults in the **Add Variable** dialog, click **Create** button.
 - 9.4 In the **Value** field select **{}** to add variable reference to **suit** from the **Deck of Cards** node.

The screenshot shows a dialog titled "Additional Fields in the Response". It has a header with "Title", "Edit", and "+ Field" buttons. Below the header, there are three field entries. Each entry has a "Key (Optional)" field and a "Value" field. The first entry has a key of "doc_card_url ..." and a value of "doc". The second entry has a key of "doc_card_val..." and a value of "value". The third entry has a key of "doc_card_sui..." and a value of "suit". Each value field has a variable reference icon (a circle with two curly braces) on the right.

- 10 Click **Apply** button to save your changes.
- 11 Select **Settings** and configure it as follows:
 - 11.1 **Node Title** as
 - **API Success**
 - 11.2 **Node Description** as
 - **Return card details**
- 12 Apply and close your dialog.



- 13 On the **Any Trigger False** path click the **Http** node to configure it.
- 14 Click **Send Error JSON Response** capability.
- 15 Under **Additional Fields in the Response** click **+ Field** button to add a new field.

15.1 Enter the following value for the field **Key**, when you press enter the variable dialog will open.

- **errorResponse**

Additional Fields in the Response

Title Edit + Field

Key (Optional) ⓘ errorResponse ▼

Value ⓘ

Enter value Create "errorResponse"

15.2 In the **Data Type** field select **String** from the dropdown list.

15.3 Accept the remaining defaults in the **Add Variable** dialog, click **Create** button.

15.4 In the **Value** field select **{}** to add variable reference to **error** (will need to click on the +) from the **Deck of Cards** node.

- This will return the entire error response from the call. You could refine to return more specific data but for now you will return the entire response.

16 Select **Settings** and configure it as follows:

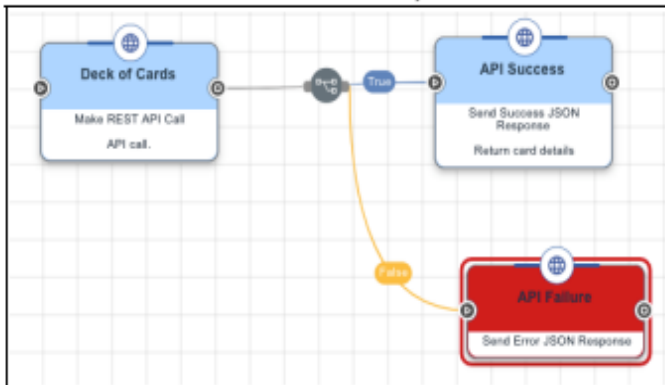
16.1 **Node Title** as

- **API Failure**

16.2 **Node Background Color** as **red**.

17 Apply and close your dialog.

18 If needed click save and then **Deploy** buttons. No point in testing the flow as you will get an error that it could not render it, since there is no UI.



19 Continue with the next section.

2.3 Implement a test flow for subflow

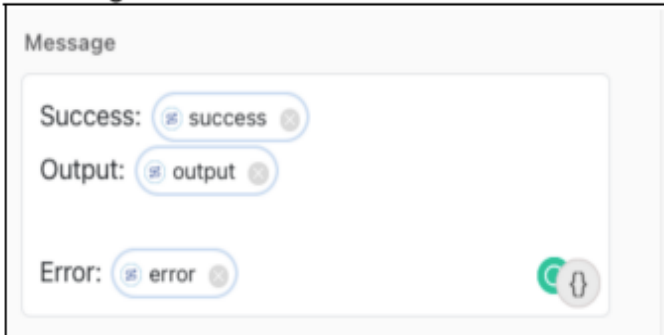
It is a good practice to develop a simple flow that wraps your subflow with a UI response on the results to test a subflow. This makes it easier to test it, especially when you make changes to the subflow. Testing in the main flow that uses the subflow could get more involved depending on the complexity of that main flow. In the case of your main flow it is the number of user interactions you go through before the subflow will be invoked.

1 Click **Flows** in the left-menu to go to your list of flows.

- 2 Click **Add Flow** button.
- 3 Click **Blank Flow** to create a new flow.
- 4 **Type** the following as the flow name:
 - **Test Deck of Cards Subflow**
- 5 Click **Create** button.
- 6 Click the **+** icon in the bottom left to add a connector.
- 7 Search for **flo** then select **Flow Conductor** this is the connector that calls a subflow.
 - You may need to create it if it is the first time you are using this connector.
- 8 Click **Flow Conductor** node in your flow to configure it.
- 9 Click **Invoke Subflow** capability.
 - The subflow has no user interaction so this is the option you want to use.
- 10 In **Flow Id** field use the dropdown to select **Subflow Deck of Cards API** from the list.
- 11 For **Flow Version Id** enter **-1**
 - This will execute the most current version of the subflow, which is what you always want to test.
- 12 Note the **requiredCardCount** entry, this comes from your subflow input schema. And it is marked as required.
- 13 Enter **1** for the **requiredCardCount** field.

- 14 Click **Settings** then set the **Node Title** to **Call Subflow**
- 15 Apply and close your dialog.
- 16 From the **Call Subflow** node drag a line set the decision to **All Triggers Complete** then **Http** connector.
 - This node will display both the success and error responses from calling the subflow.
- 17 Click **Custom Html Message** as the capability.

- 18 In the **Message** field you want to use the variables from **Call Subflow** node as shown in the following screenshot:



- For output and error, just click on the plus next to the variable.

- 19 Apply and close your dialog.



- 20 Continue with the next section.

2.3.1 Test the flow calling the subflow

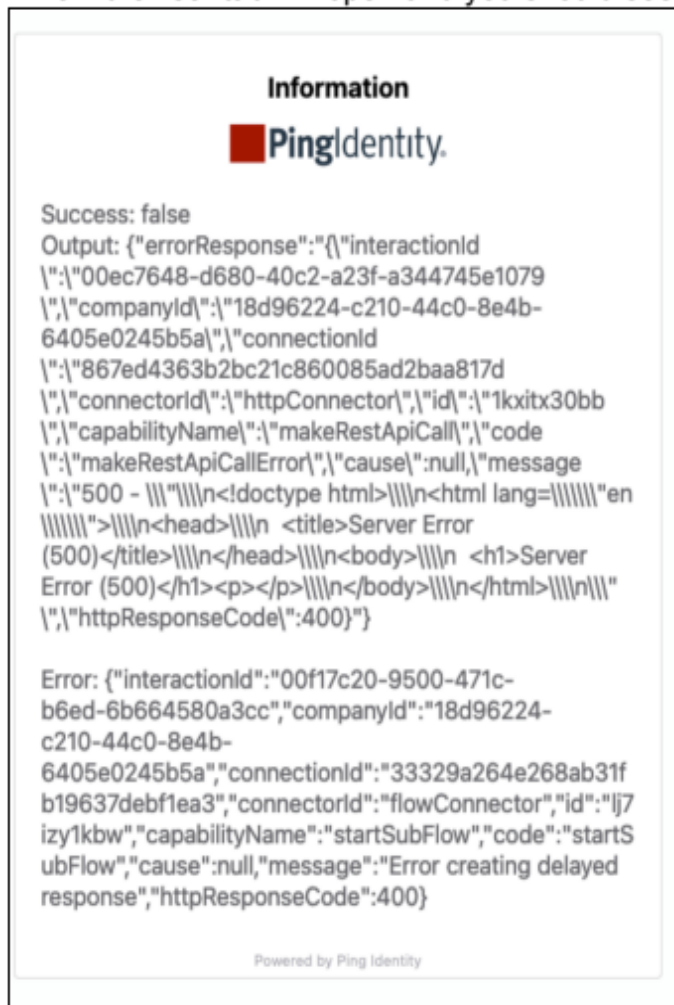
Now it is time to test your subflow by using the new testing flow you created.

- 1 Deploy and try your flow.
- 2 A new browser tab will open and you should see the following screenshot, which is a success:



- 3 You can close the browser tab.
- 4 In order to generate a fail test, in the **Call Subflow** node change the **requiredCardCount** to **all**, the parameter must be numeric.
- 5 Apply the changes and the save, deploy, and try your flow.

6 A new browser tab will open and you should see the following screenshot, which is a failure:



- 7 Change the required value back to 1 for future tests.
- 8 Save and deploy then exit out of our flow to the list by selecting **Flows** from left menu.
- 9 Continue with the next section.

2.4 Replace API call in registration flow with subflow

Your subflow is complete and you tested that it works to call the API and return a result. Now it is time to change your main flow to use the new subflow, replacing the existing API call.

- 1 Click **Progressive Registration** flow in the list of flows to edit the flow.
- 2 Scroll to the right until you see the **Deck of Cards** node.
- 3 **Right-mouse click** the **Deck of Cards** node.
- 4 Click **Replace** menu item to replace the existing node in the flow.
 - Saves you having to delete and move things around in your flow.
- 5 Search for **flo** then select **Flow Conductor** connector to replace the existing node.
- 6 Click **Flow Conductor** node you just placed on your canvas, to edit it.
- 7 Click **Invoke Subflow** capability.
- 8 In **Flow Id** field use the dropdown to select **Subflow Deck of Cards API** from the list.
- 9 For **Flow Version Id** enter **-1**
- 10 Enter **1** for the **requiredCardCount** field.

11 Click **Settings** tab and enter the following:

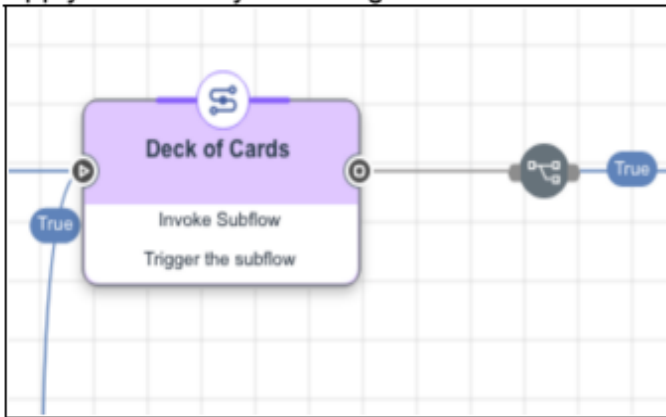
11.1 **Node Title** as

- **Deck of Cards**

11.2 **Node Description** as

- **Trigger the subflow**

12 Apply and close your dialog.



13 For completeness you should add a failure path in the event the subflow fails, this optional at this point. But if you have time feel free to try it out.

14 Don't forget to save your flow.

15 Continue with the next section.

2.4.1 Change the display pages to use new display values

Now you need to change the nodes that use the values returned by the API call to use the new values that you defined as output from success on your subflow. There are two nodes that need to be changed.

1 Click **Robot Test** node to edit its configuration.

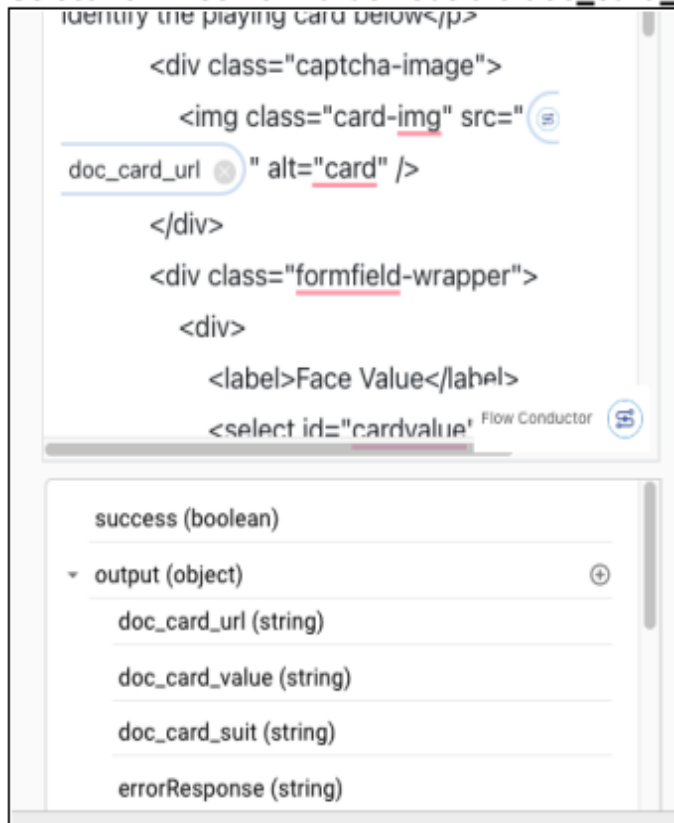
2 In the **HTML Template** field scroll down until you see the `<img class="card-img"`

3 For the **src** attribute you will see something like the following:

- `{{local.vhvvo42cwm.payload.output.rawResponse.body.cards[0].images.png}}`
- The ID after local will look different in your flow, but everything else should be the same.

4 Highlight the text

`{{local.vhvvo42cwm.payload.output.rawResponse.body.cards[0].images.png}}` then use `{}` to select a new variable.

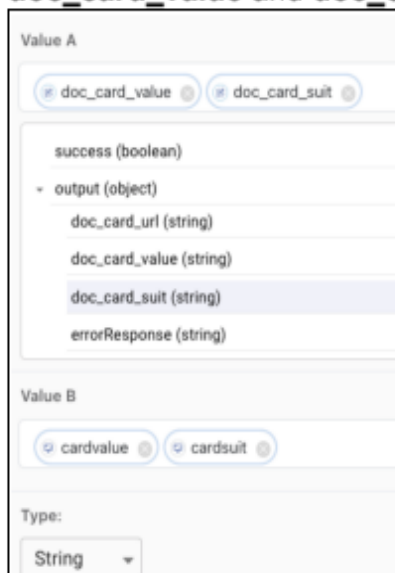
5 Select from **Deck of Cards** node the **doc_card_url**

6 Apply and close the dialog.

7 Save your changes to the flow.

8 Click **Card Validation** node.9 For the **Value A** field you need to replace9.1 **value** and **suit** variables

9.2 with

9.3 **doc_card_value** and **doc_card_suit** from **Deck of Cards** node.

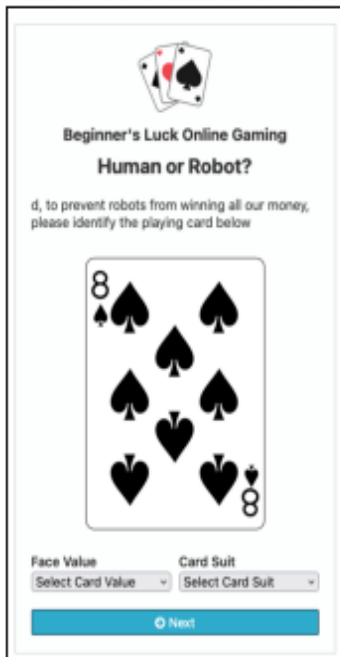
10 Apply and close the dialog.

11 Continue with the next section.

2.4.2 Test your registration flow that invokes the subflow

Now it is time to test your main flow to make sure that the same functionality is provided with the robot test using the subflow.


- 1 Deploy and try your flow to test it.
- 2 Enter values for first and last name.
- 3 Enter a valid age over 21 or greater.
- 4 Enter values for email and password.
- 5 Confirm the entries.
- 6 You should see the robot test card



Beginner's Luck Online Gaming

Human or Robot?

d, to prevent robots from winning all our money,
please identify the playing card below



Face Value Card Suit

Select Card Value Select Card Suit

Next

- 7 Enter first an incorrect selection and then retry with correct values.
- 8 You have completed this exercise.