Lab 2

Consume a reusable Social Network API
Overview

Now that Sheryl has designed a flow that accesses customer data from a reusable Customer API, she can proceed with consuming a Social Network API from the same flow to send the customer data to Facebook to fulfill her marketing campaign needs. Let’s once again play the role of Sheryl in completing this flow design with the Social Network API for Facebook.

Step 1: Design the Flow

In the previous lab, we consumed the Customer API by getting all the customers. Now, we would like to apply the following logic:

1. Check if the customer call has any data
2. If successful,
   a. Create an Audience
   b. Get the Audience ID
   c. Create the Facebook Campaign
3. If failure
   a. Set the response code to 400 (400 is the status code for Bad Requests)
   b. Set an error message to communicate to the user

Step 2: Check If Collection is not Empty

The steps that need to be executed are:

1. After the Customer API, press [Button]
2. Scroll down and select the choice component.
Most fields can be populated with either a static value or an expression. Switch between one modality or another by clicking the expression button.

When expressions are enabled on a field, you can write a DataWeave expression. This can consist of a simple reference to a part of the element's input (such as 'payload.user'), or may also involve functions to manipulate incoming values into something else. Smart autocomplete options help you write valid expressions that reference existing elements.

**NOTE** When using the expression editor, anything you type into the field is considered a DataWeave expression. Therefore, you shouldn't wrap your expression in #[], as this is already implied.

3. Click inside the 'Expression (required)' data entry field.
4. Enter expression `payload != null`. This is a DataWeave expression. Press the Ok button to save.

5. Enter the Display Name `If Not Empty`
You are creating an **if then else** statement.

Now let’s continue creating the statement by adding a message to the default branch.

**Step 3: Create Not Found Message**

Here we are going to send an error message when no customer is Found.

1. Click on the button that is beside the default box.

2. Add a **Transform** component.
3. Click on the **script** tab

4. Replace the value `null` with "No Customers Found", ensuring you enter the double quotes to indicate this is a string literal.
5. Close the card by pressing the X in the top right corner.

Step 4: Create an Audience

1. Click on the next to the If statement that has a red dot.

We are going to consume another API, so we need to select another REST Connector.

2. Search for Social Network on the select component panel and select Social Network API for Facebook.
3. Select **Create Audience**.
4. Create an Audience based on the Customer data by adding a new card titled **Social Network API for Facebook**.

   a. Complete the card with the following information:
      - Client Id: 11def1b704e24d87a5ea5769772c90a7
      - Client Secret: 88845E529f1F42E4a5C96Fd504c3e01
      - Content_Type: application/json
      - Accept: application/json
5. Press the Edit link

6. Complete with the following information:
   - **Configuration Name**: Social Network API for Facebook
   - **Host**: social-network-api.cloudbhub.io
   - **Port**: 80
   - **Base Path**: /api
   - **Protocol**: HTTP
As we did in the Customer API card, we need to configure a Connector Name and Type.

7. Press the **Add Connection** button.

8. Complete with a name and choose **Connection** in the **Connection Type**. We are not going to share the connection, so uncheck the option.

9. Press the **Test** button and ensure a **Success** message is returned.

10. Press the **Save** button after a successful test.

11. Close the card by pressing the **X** in the top right corner.
According to the API RAML Specification we need to adapt the customer information to what the API needs.

12. If you hover over the arrow that is between If Not Empty and Social Network API for Facebook it will convert into an plus button. Press the button.

13. Select the Transform component.
After you select the transform component, you are going to see a panel like this one:
- The first panel is the input. In this case it is the **Customer API** output.
- The second panel is the output. In this case it is what the **Social Network API for Facebook** is expecting.
- The third panel shows you how the mapping ends.

14. Click on the **script** tab

**NOTE**
You can drag and drop the fields from the input to the output and the mapping will be generated for you

15. In this case, to make it faster, we are going to go to the script tab and copy the following text:

```dw 2.0
output application/json
---
{
  customers: (payload map (value0, index0) -> {
    country: value0.billingAddress.country,
    phone: value0.phone,
    city: value0.billingAddress.city,
    postalCode: value0.billingAddress.postalCode,
    email: value0.email,
    product: []
  })
}
```

**NOTE**
We are going to add sample data to this transform so that we can see if our transform is correct in realtime.
16. Go to the **Sample data** tab.

17. Copy and paste the following example:
The third panel will show you how the transformation will work and will respond in real time as you modify the data or the transformation.
18. Close the card by pressing the X in the top right corner.

Great! We are now consuming the Social Network API for Facebook and have created an Audience.

The next step is to create a Campaign.

### Step 5: Creating the Campaign

Once we get the Audience, we need to create the campaign. First we are going to add a log to see the Create Audience response
1. Press the button that is after the Social Network component.

2. Select the logger component.

3. Complete with card with the following data:
   - Message: `payload`. This sets the entire payload to be logged.
   - Level: `Info` (which is the default)
4. Close \( \times \) logger card.

**NOTE** We will now add another call to the Social Network API for Facebook, but in this case select method **Create Campaign**

5. Press the \( + \) that is after the Logger component you just added.

6. Search for **Social Network** on the select component panel and select **Social Network API for Facebook**.
Select Create Campaign from the list.
7. Complete the configuration with the following information:
   ◦ Client Id: 11def1b704e24d87a5ea5769772c90a7
   ◦ Client Secret: 88845E529f1F42E4a5aC96Fd504c3e01
   ◦ Content_Type: application/json
   ◦ Accept: application/json
8. Press the **Edit** link.

9. Verify the following information:
   - **Configuration Name:** Social Network API for Facebook
   - **Host:** social-network-api.cloudhub.io
   - **Port:** 80
   - **Base Path:** /api
   - **Protocol:** HTTP
10. Close the card.

**NOTE** Once again we need to transform the payload before sending the post message. To accomplish this, we will add a transform between the Logger and the last **Social Network API for Facebook**

11. Press between the Logger Card and the Social Network Create Campaign card
12. Select the **Transform** component.

13. You will see the following. Note the input and output panes.

14. Open the Audience structure within the Output payload by clicking on the arrow.
15. Drag and Drop **payload id** to the **audience id** See the below image

![Diagram showing drag and drop of payload id to audience id]

**NOTE** Since we have not entered sample data, the preview shows 'null' for the id

16. Close the card.

17. Verify your final flow looks like this:
Great! You have finished the call to Create Campaign.

Congratulations! You have completed the Facebook Campaign Implementation.

In the next lab we are going to test it!

Summary

In this lab we,

- Designed a Flow
- Added a choice
- Created an Audience
- Created a Campaign
- Sent an error message when no customer is found.

This lab demonstrates how easy Design Center can be to build MuleSoft APIs.

For further reading:

- See the link Design Mule Application with Design Center doc for more information.
- See the link Create Your First Mule Application with Flow Designer doc for more information.
- See the link Fill in Cards with Values that Change at Runtime doc for more information on data.

Congratulations! You have completed Lab 2.

Please proceed to Lab 3

Take me to the TOP