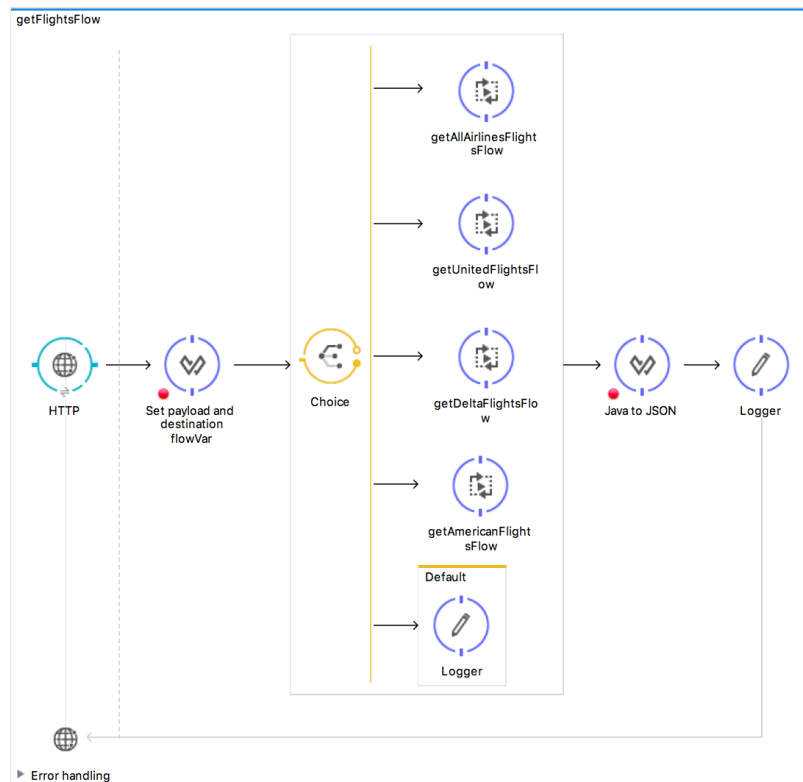


Walkthrough 8-2: Route messages based on conditions

In this walkthrough, you will create a flow that routes messages based on conditions. You will:

- Use a Choice router to get flight results for all three airlines or only a specific airline.
- Set the router paths based on the airline value sent from the flight form.



Look at selected airline values in HTML form

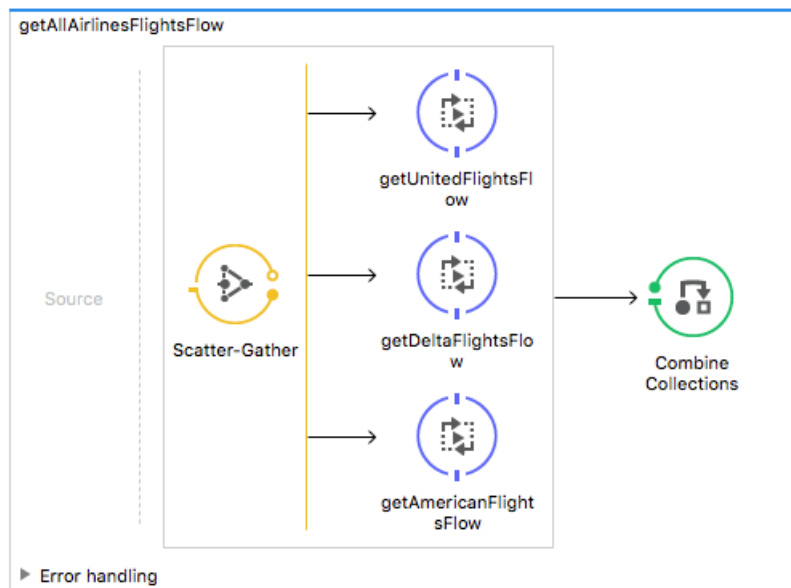
1. In Anypoint Studio, return to FlightFinder.html.
2. Locate the select box that sets the airline and see what values are set and returned.

```
190 <select id="airline" name="airline" class="select2">
191   <option value="all">All Airlines</option>
192   <option value="united">United</option>
193   <option value="delta">Delta</option>
194   <option value="american">American</option>
195 </select>
```

Move the Scatter-Gather to a separate flow

3. Return to getFlights.xml.
4. Shift+click to select the Scatter-Gather router and the Combine Collections transformer.
5. Right-click and select Extract to > Flow.

6. In the Extract Flow dialog box, set the flow name to getAllAirlinesFlightsFlow.
7. Leave the target Mule configuration set to current and click OK.

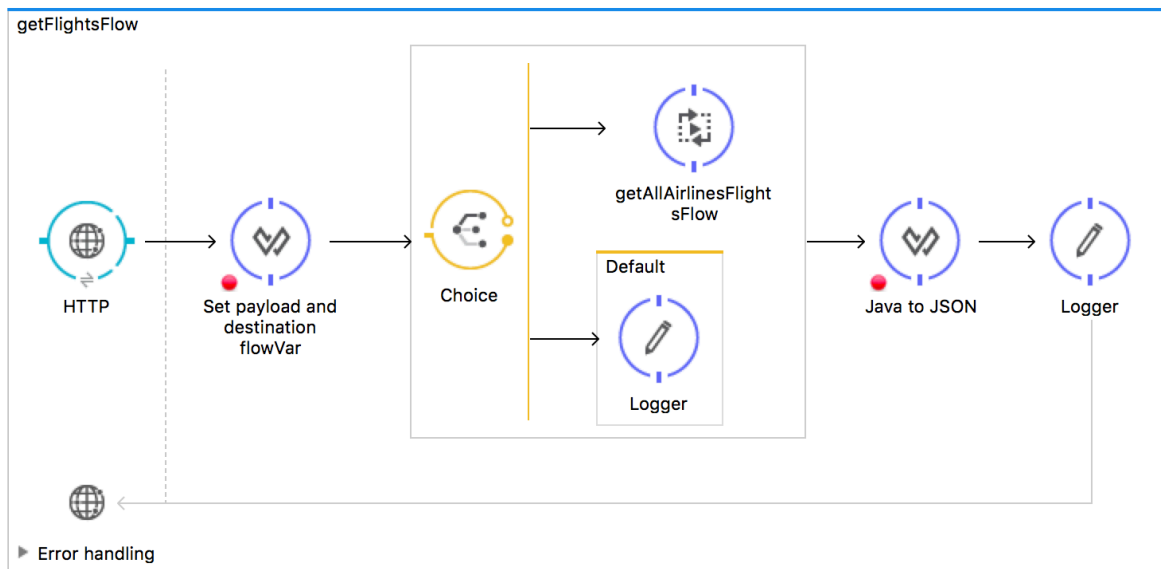


8. In `getFlightsFlow`, double-click the new Flow Reference.
9. Change its display name to `getAllAirlinesFlightsFlow`.
10. Move `getAllAirlinesFlightsFlow` beneath `getFlightsFlow`.
11. Add a breakpoint to the `Combine Collections` transformer.

Add a Choice router

12. In `getFlightsFlow`, add a Choice flow control element between the `Set payload` and `destination flowVar` component and the `getAllAirlinesFlightsFlow` flow reference.
13. Drag the `getAllAirlinesFlightsFlow` flow reference into the router.

14. Add a new Logger component to the default branch.

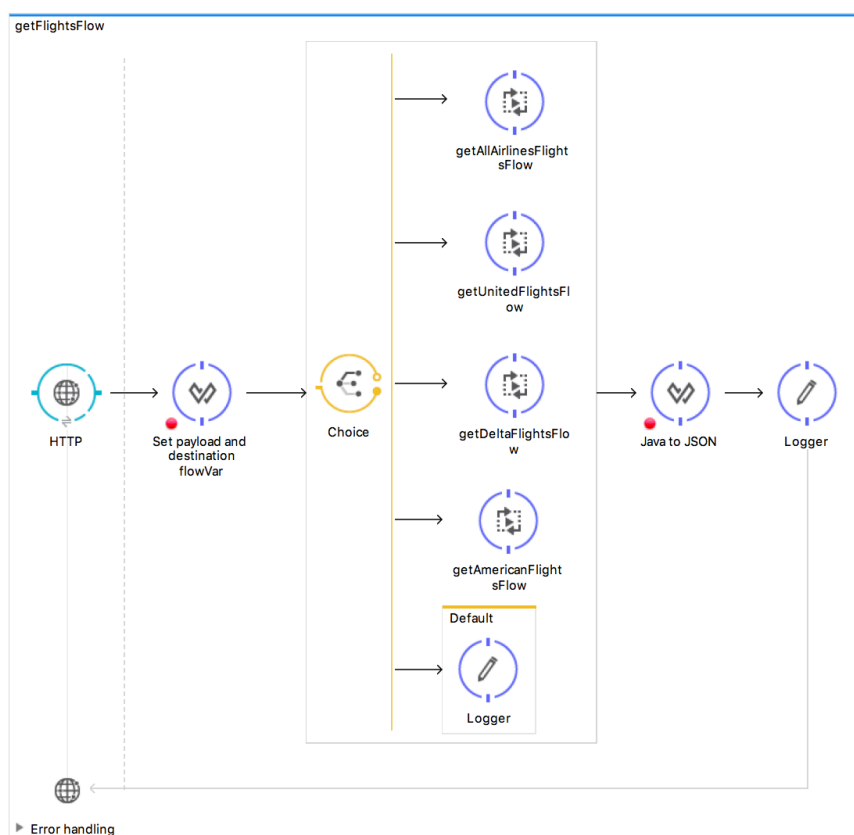


15. Add three additional Flow Reference components to the Choice router to create a total of five branches.

16. In the Properties view for the first new flow reference, set the flow name to `getUnitedFlightsFlow`.

17. In the Properties view for the second flow reference, set the flow name to `getDeltaFlightsFlow`.

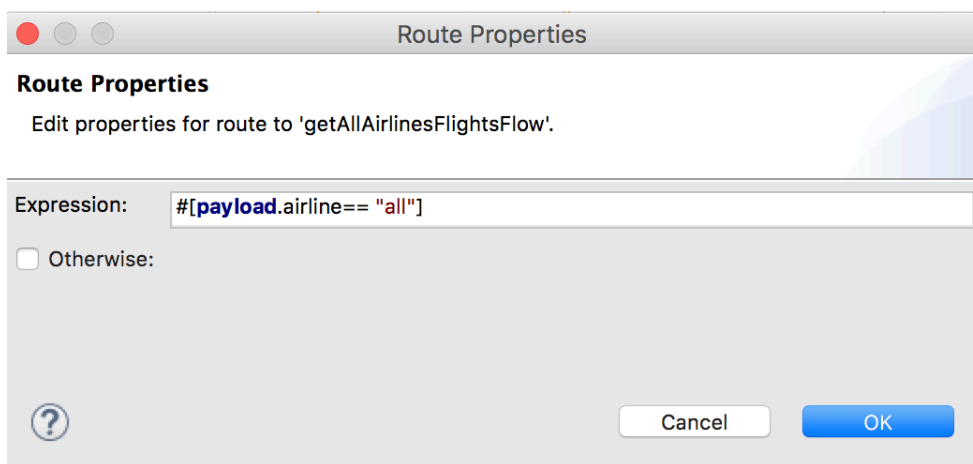
18. In the Properties view for the third flow reference, set the flow name to getAmericanFlightsFlow.



Configure the Choice router

19. In the Properties view for the Choice router, double-click the getAllAirlinesFlightsFlow route.

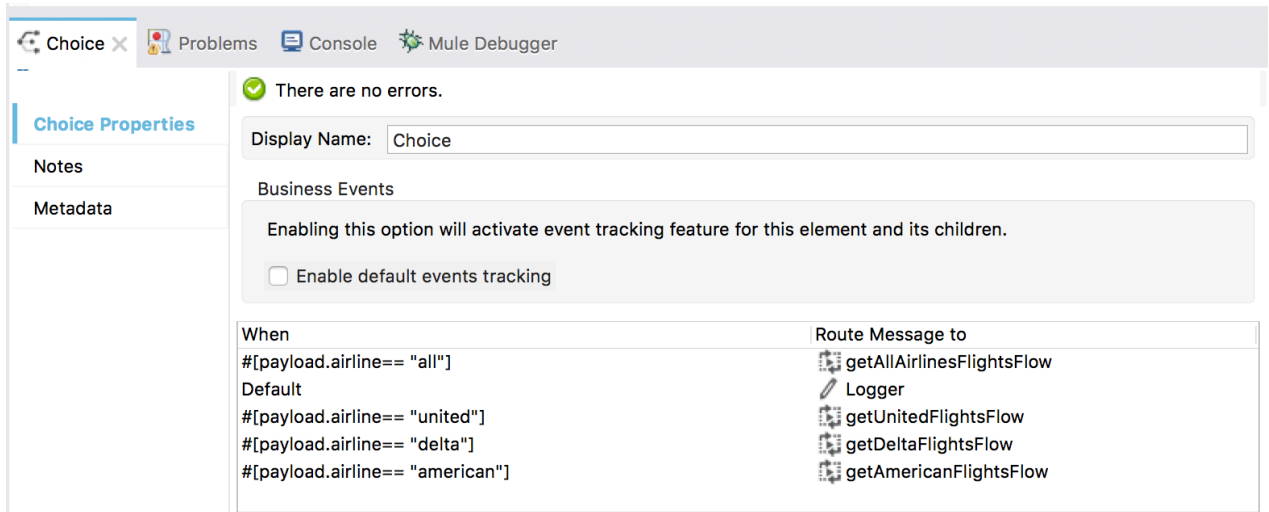
20. In the Route Properties dialog box, set the expression to `#[payload.airline=="all"]` and click OK.



21. Set a similar expression for the United route, routing to it when payload.airline is equal to united.

22. Set a similar expression for the Delta route, routing to it when payload.airline is equal to delta.

23. Set a similar expression for the American route, routing to it when payload.airline is equal to american.



Choice x Problems Console Mule Debugger

Choice Properties

Notes

Metadata

There are no errors.

Display Name: Choice

Business Events

Enabling this option will activate event tracking feature for this element and its children.

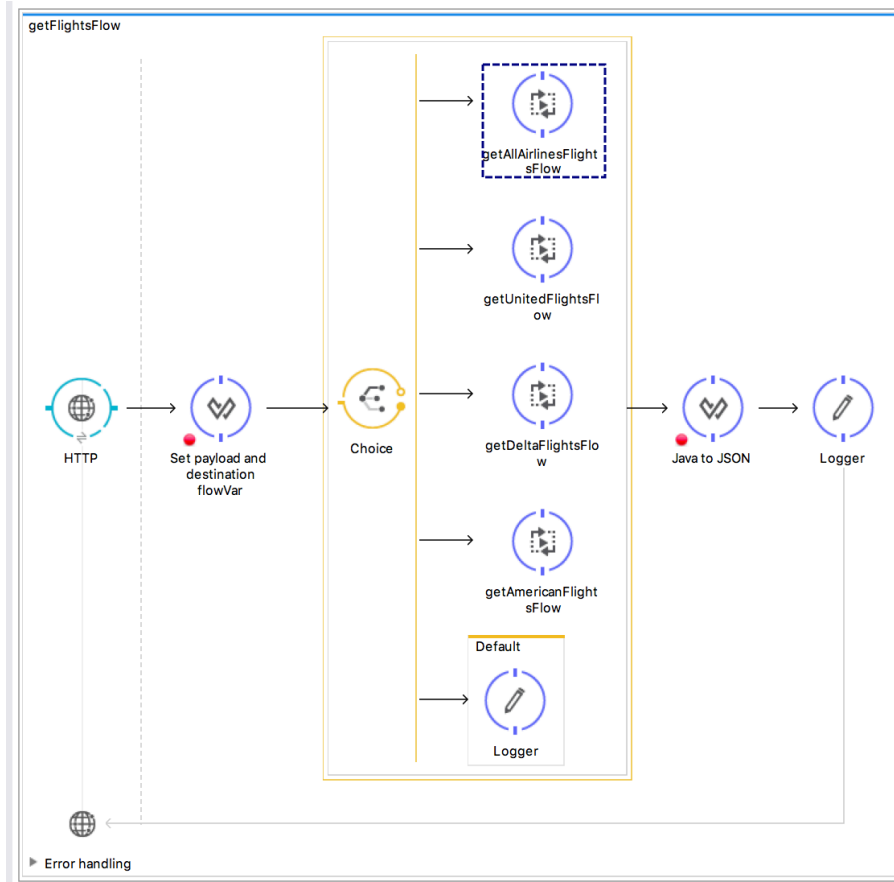
☐ Enable default events tracking

When	Route Message to
<code>#[payload.airline == "all"]</code>	getAllAirlinesFlightsFlow
Default	Logger
<code>#[payload.airline == "united"]</code>	getUnitedFlightsFlow
<code>#[payload.airline == "delta"]</code>	getDeltaFlightsFlow
<code>#[payload.airline == "american"]</code>	getAmericanFlightsFlow

Debug the application

24. Save the file and debug the application.
25. Make a request to <http://localhost:8081/flights>.
26. On the form page, leave SFO and All Airlines selected and click Find Flights.

27. Return to the Mule Debugger view and step through the application; you should see the Choice router pass the message to the getAllAirlinesFlightsFlow branch.



28. Click the Resume button until flight data for all airlines is returned back to the form page.

29. On the form page, select SFO and United and click Find Flights.

30. Return to the Mule Debugger view and step through the application; you should see the Choice router pass the message to the United branch.

31. Click the Resume button until flight data is returned back to the form page; you should see only United flights.

The screenshot shows a web browser window with the address bar at `localhost:8081/flights`. The page title is "Mule United Airport Flight". The main content area has a header "Mule United Airport" and a search bar with "SFO - San Francisco" and "United" selected, and a "Find Flights" button. Below the search bar, there is a section titled "Available Flights" with three flight entries separated by dashed lines:

Flight Code	Airline Name	Destination	Plane Type	Price	Departure Date	Available Seats
ER38sd	United	SFO	Boeing 737	\$400	2015/03/20	0
ER39rk	United	SFO	Boeing 757	\$945	2015/09/11	54
ER39rj	United	SFO	Boeing 777	\$954	2015/02/12	23

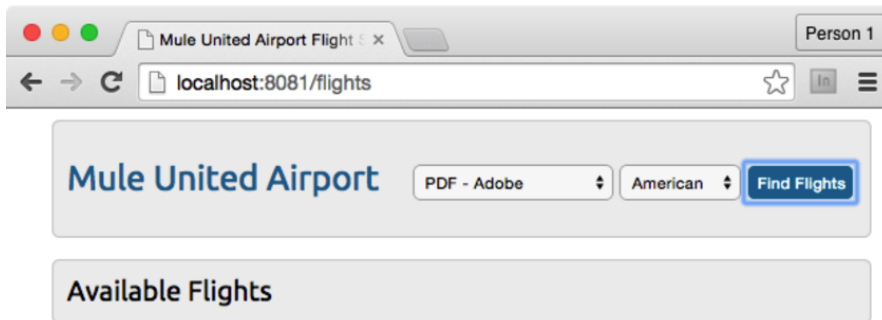
Test the application

32. Stop the debugger and run the application.
33. Make a request to <http://localhost:8081/flights>.
34. Test the Delta and American choices with SFO, LAX, or CLE; you should see the appropriate flights returned.
35. Test the PDF location with United; you should see one flight.

The screenshot shows the same web browser window, but the search bar now has "PDF - Adobe" selected instead of "SFO - San Francisco". The "Available Flights" section now displays a single flight entry:

Flight Code	Airline Name	Destination	Plane Type	Price	Departure Date	Available Seats
ER95jf	United	PDF	Boeing 787	\$234	2015/02/12	23

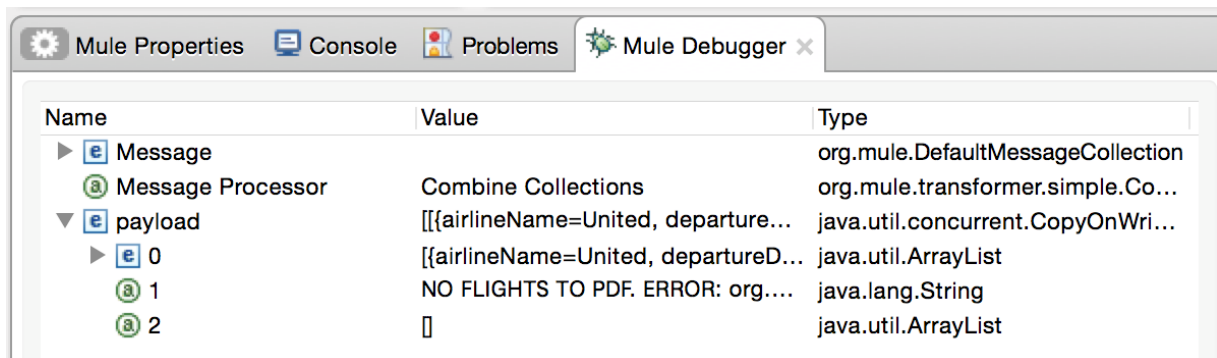
36. Test the PDF location with Delta; you should get an error in the console.
37. Test the PDF location with American; you should get no error, but no results.



38. Test the PDF location with All Airlines; you should get no results and an error in the console.

Debug the application

39. Debug the application.
40. Make a request to <http://localhost:8081/flights> and submit the form with PDF and All Airlines.
41. Step through the application to the Combine Collections transformer; you should see different types of objects returned.



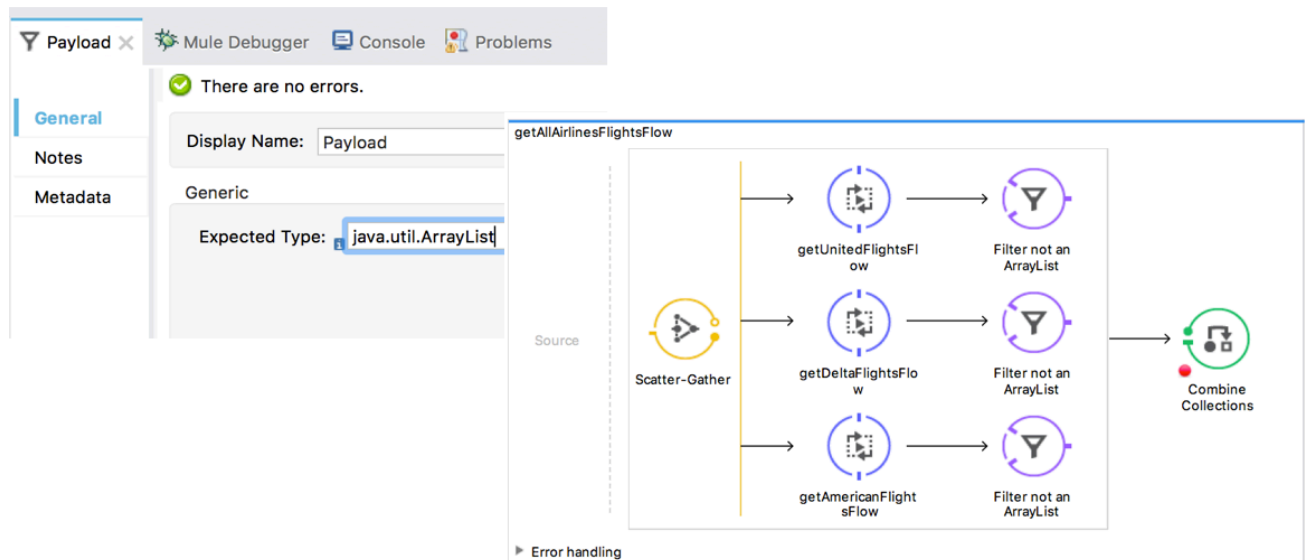
42. Step to the Java to JSON component; you should see the payload still contains the error message in addition to the valid flight results to PDF.
43. Step to the end of the application; you should not get the United results to PDF.

Note: You could write a custom aggregation strategy for the Scatter-Gather component with Java to handle this situation, but instead you will use the simpler approach of filtering out the exception messages.

Walkthrough 8-3: Filter messages

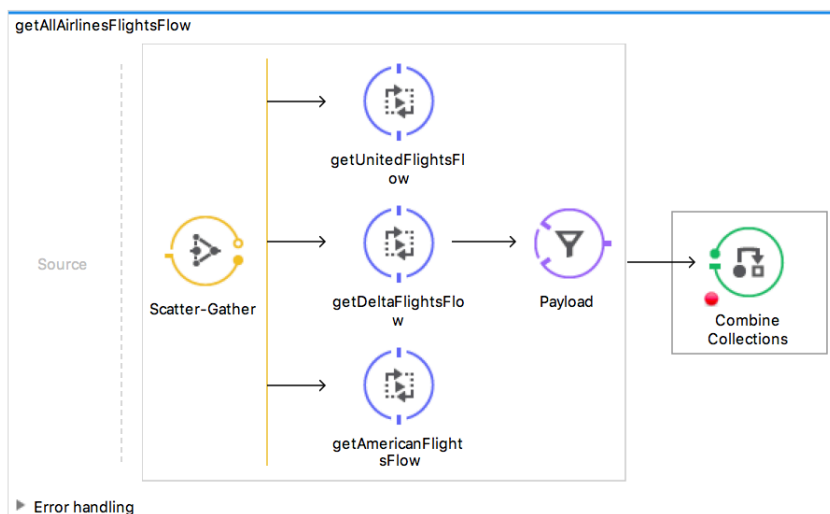
In this walkthrough, you will continue to work with the airline flight results. You will:

- Filter the results in the multicast to ensure they are ArrayLists and not exception strings.
- Use the Payload and Expression filters.
- Create and use a global filter.

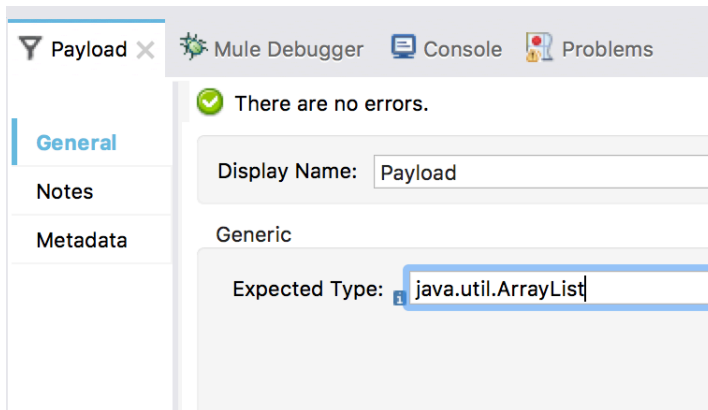


Add a filter

1. Return to getFlights.xml.
2. Drag out a Payload Flow Control element from the palette and drop it after the getDeltaFlightsFlow reference in the Scatter-Gather.

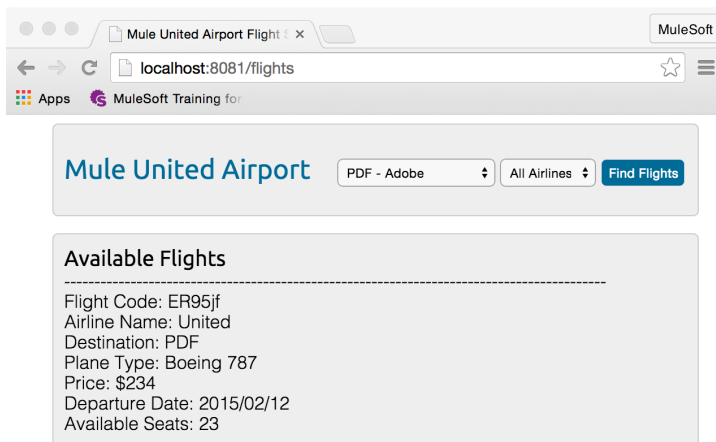


3. In the Payload Properties view, set the expected type to `java.util.ArrayList`.



Test the application

4. Save the file and debug the application.
5. Make a request to <http://localhost:8081/flights> and submit the form with PDF and All Airlines.
6. Step through the Java to JSON component; this time you should see the payload only contains the ArrayList and not the error string.
7. Step to the end of the application; you should see the United results to PDF returned to the form.

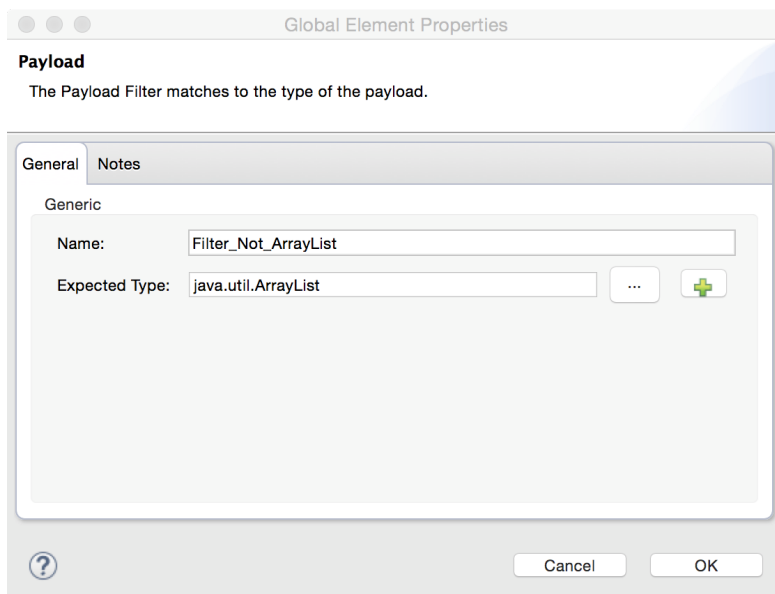


Create a global filter

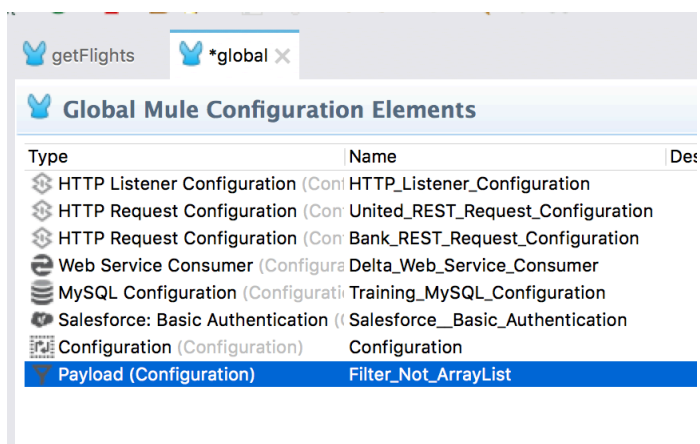
8. Delete the Payload filter.
9. Return to global.xml.
10. Switch to the Global Elements view and click Create.
11. In the Choose Global Type dialog box, select Filters > Payload and click OK.

12. In the Global Elements dialog box, set the name to Filter_Not_ArrayList.

13. Set the expected type to java.util.ArrayList and click OK.



14. See your new global filter in the list.

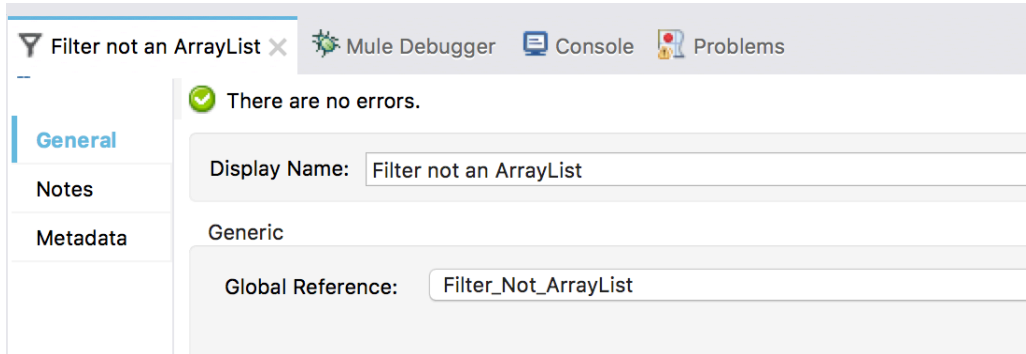


Use the global filter

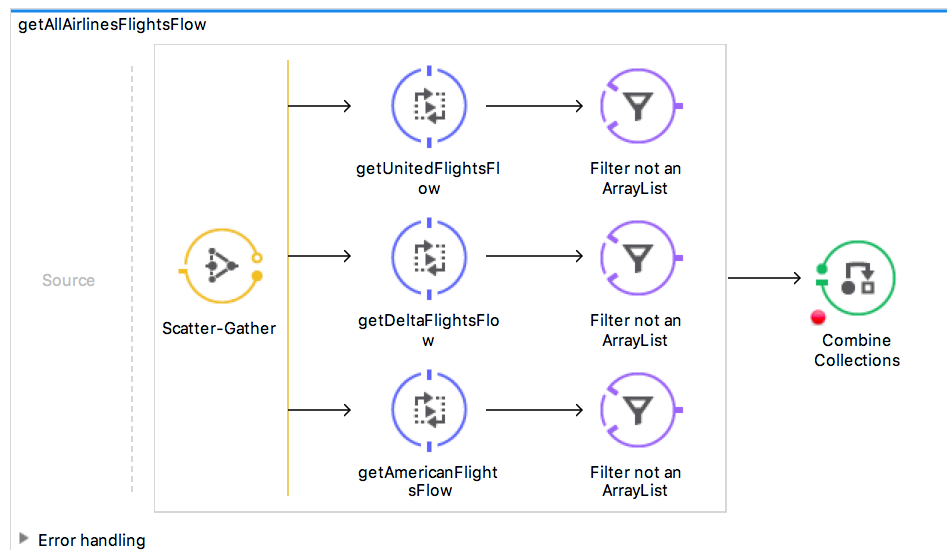
15. Return to getFlights.xml.

16. Drag a Filter Reference from the palette and drop it after getDeltaFlightsFlow in the Scatter-Gather.

17. In the Filter Reference Properties view, set the display name to Filter not an ArrayList and set the global reference to Filter_Not_ArrayList.



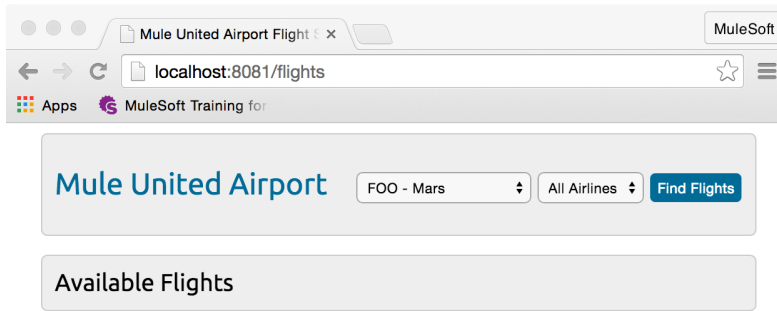
18. Add a Filter Reference after getAmericanFlightsFlow.
19. In the Filter Reference Properties view, set the display name to Filter not an ArrayList and set the global reference to Filter_Not_ArrayList.
20. Copy and paste the Filter Reference in the Scatter-Gather.
21. Drag it after getUnitedFlightsFlow and change its name to Filter not an ArrayList.



Test the application

22. Save all the files and run the application.
23. Make a request to <http://localhost:8081/flights> and submit the form with PDF and All Airlines; you should still see the United results.

24. Submit the form for FOO and All Airlines; you should get no results and no message.



Look at the HTML form and find the display if no flights are returned

25. In Anypoint Studio, return to or open FlightFinder.html.

26. Locate the code that sets the text if there is no flights are returned.

```
158         catch(e) {
159             if (response) {
160                 document.getElementById("myDiv").innerHTML = response;
161             }
162             else{
163                 document.getElementById("myDiv").innerHTML = "There are no available flights";
164             }
165         },
166     }
```

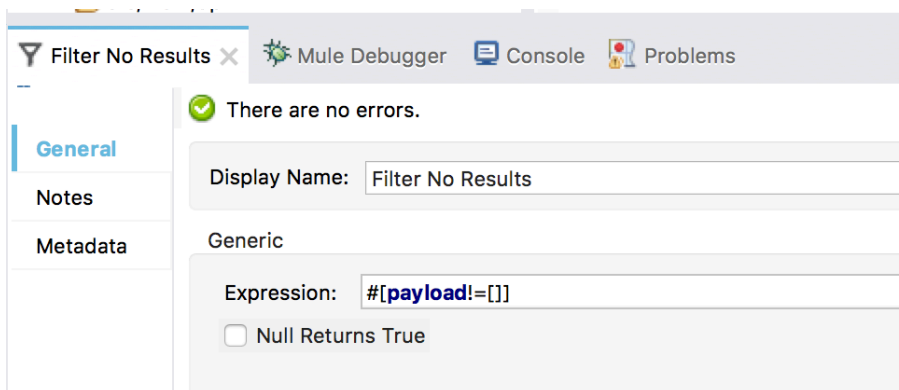
Add an Expression filter to filter no results

27. In the getFlightsFlow, add an Expression filter after the Choice router.

28. In the Expression Properties view, set the display name to Filter No Results.

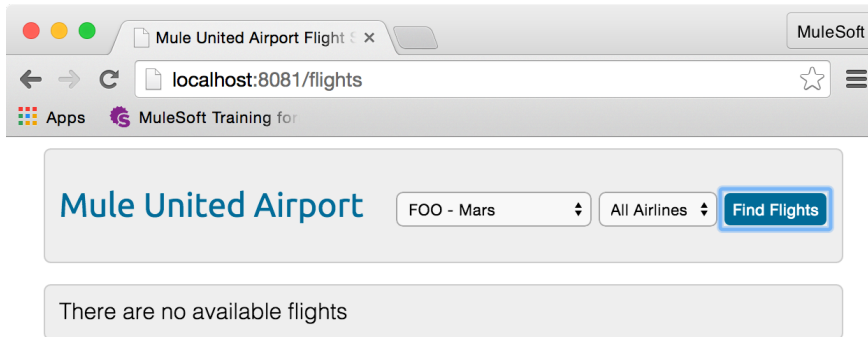
29. Set the expression to see if the payload is equal to an empty Array.

```
#[payload!=[]]
```



Test the application

30. Save the file to redeploy the application.
31. Submit the form for FOO and All Airlines; you should see the message there are no available flights in the browser window.



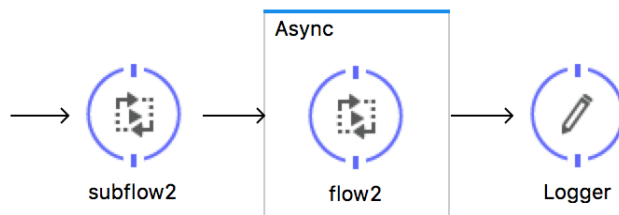
32. Test the PDF location with American; you should get the no available flights message.

Note: If you test the PDF location with United or Delta, you should get no results and an error in the console: Execution of the expression "exception.causeMatches('com.mulesoft.weave.')" failed. This is a bug in 3.7.1. As a workaround, you can modify the global Catch Exception Strategy for DataWeave expressions to `#[message.?exception.causeMatches('com.mulesoft.weave.*')]`, but you will also need to add some more logic to return the desired results.*

Walkthrough 8-4: Pass messages to an asynchronous flow

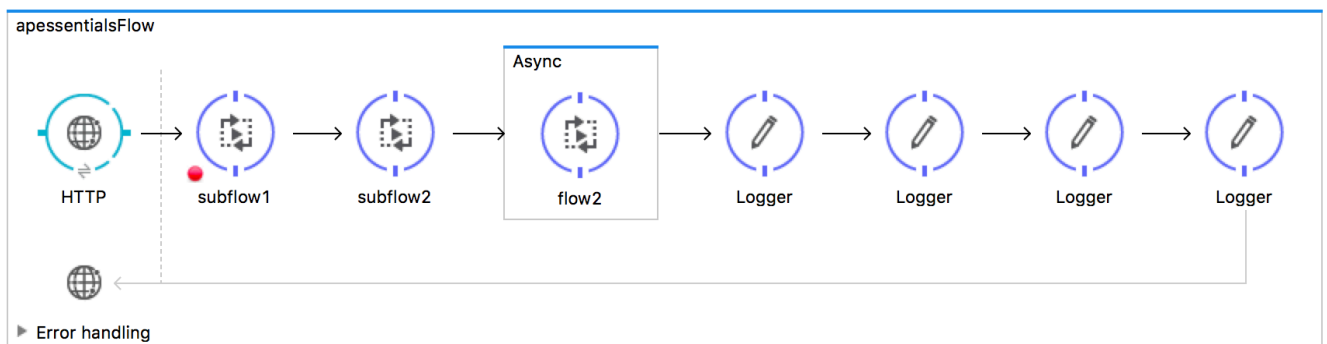
In this walkthrough, you will work in the `apessentials2` project and create and pass messages to an asynchronous flow. You will:

- Use the Async scope element to create an asynchronous flow.
- Use the Mule Debugger to watch messages flow through both flows.



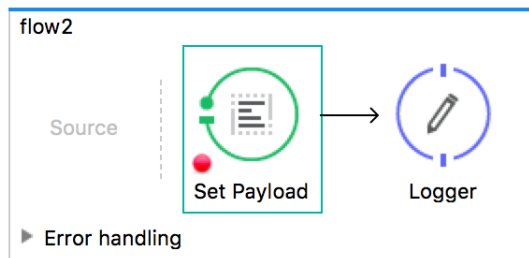
Create an asynchronous flow

1. Return to `examples.xml` in `apessentials2`.
2. Drag an Async scope element from the palette and drop it into the `apessentialsFlow` between the HTTP Request endpoint and the Logger.
3. Delete the HTTP Request endpoint in `apessentialsFlow`.
4. Drag a Flow Reference into the Async scope.
5. In the Flow Reference Properties view, set the flow name to `flow2`.
6. Add three more Logger components after the Async scope.



7. Delete the HTTP Listener endpoint in `flow2`.
8. Add a Logger component after the Set Payload in `flow2`.

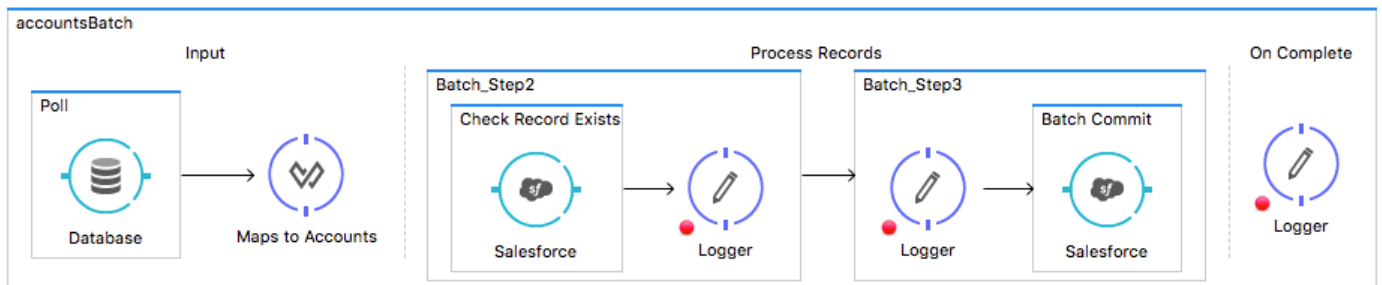
9. Make sure there is a breakpoint on the Set Payload in flow2.



Debug the application

10. Save the file and debug the application.
11. Make another request to <http://localhost:8082?name=pliny&type=cat> using your own query parameter values.
12. Step through the application again; you should see a copy of the message passed to the asynchronous flow2.
13. Watch the values of the payload change in both flows.

Module 9: Processing Records



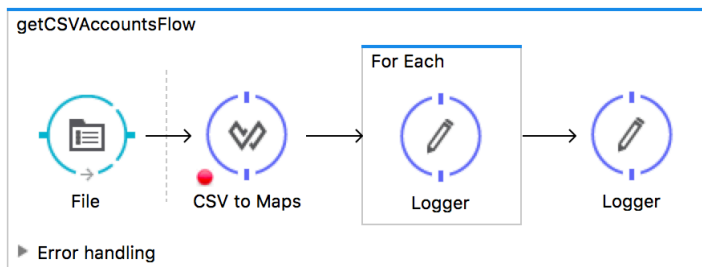
In this module, you will learn:

- Process items in a collection individually.
- Use DataWeave with CSV files.
- Use the Batch Job element (EE) to process individual records.
- Synchronize data from a CSV file to a SaaS application.
- Synchronize data from a legacy database to a SaaS application.

Walkthrough 9-1: Process items in a collection individually

In this walkthrough, you will split a collection and process each item in it individually. You will:

- Add metadata to a File endpoint.
- Read a CSV file and use DataWeave to convert it to a collection of objects.
- Use the For Each scope element to process each item in a collection individually.



Modify the File endpoint to not rename the files

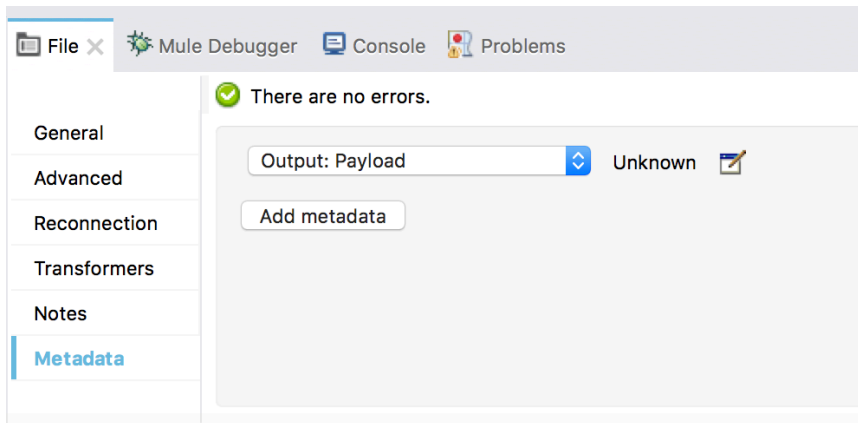
1. Open `accounts.xml`.
2. In the Package Explorer, expand the `src/main/resources/input` folder.
3. Right-click `accounts.csv.backup` and select **Refactor > Rename**.
4. In the Rename Resource dialog box, set the new name to `accounts.csv` and click **OK**.
5. In `getCSVAccountsFlow`, delete the File-to-String transformer.
6. In the File Properties view, delete the move to pattern.

Note: This will make it easier to test the application because you won't have to keep renaming the file as you move it back to the input directory.

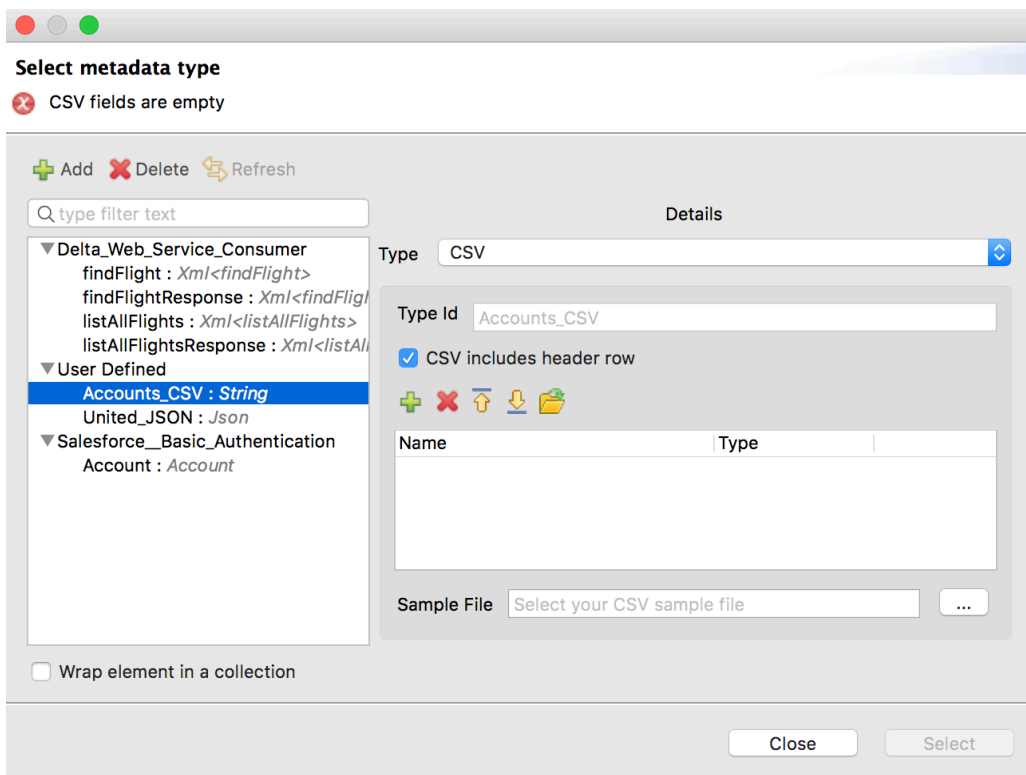
Add File endpoint metadata

7. In the File Properties view, click **Metadata** in the left-side navigation.
8. Click the **Add metadata** button.

9. In the drop-down menu that appears, make sure Output: Payload is selected.



10. Click the Edit button next to the drop-down menu.
11. In the Define Type dialog box, select Create new type.
12. Set the Type Id to Accounts_CSV.
13. Click Create type.
14. Change the Type to CSV.
15. Make sure CSV includes header row is checked.
16. Click the Load from example button (the folder icon).



17. In the Load CSV fields from file dialog box, click the Browse button next to Example.

18. In the Select CSV example dialog box, browse to the project's src/main/resources/input folder, select accounts.csv, and click Open.
19. In the Load CSV fields from file dialog box, click OK.

Load CSV fields from file

Select your CSV file and complete your configuration

Example ...

Ignored Rows

Delimiter

Quoted Fields ☐

Quote char

Configuration preview

Billing Street	Billing City	Billing Country	Billing State	Name
111 Boulevard...	Paris	France		Dog Park Indust...
400 South St	San Francisco	USA	CA	Iguana Park Ind...
777 North St	San Francisco	USA	CA	Cat Park Industries

Cancel OK

20. In the Select metadata type dialog box, click the Select button.

Select metadata type

Choose metadata type from tree and click Select

+ Add - Delete Refresh

Q type filter text

▼ Delta_Web_Service_Consumer
findFlight : Xml<findFlight>
findFlightResponse : Xml<findFlightResponse>
listAllFlights : Xml<listAllFlights>
listAllFlightsResponse : Xml<listAllFlightsResponse>

▼ User Defined
Accounts_CSV : String
United_JSON : Json

▼ Salesforce__Basic_Authentication
Account : Account

Type CSV

Type Id Accounts_CSV

☒ CSV includes header row

Name Type

Billing Street	String
Billing City	String
Billing Country	String
Billing State	String
Name	String
BillingPostalCode	String

Sample File ...

☐ Wrap element in a collection

Close Select