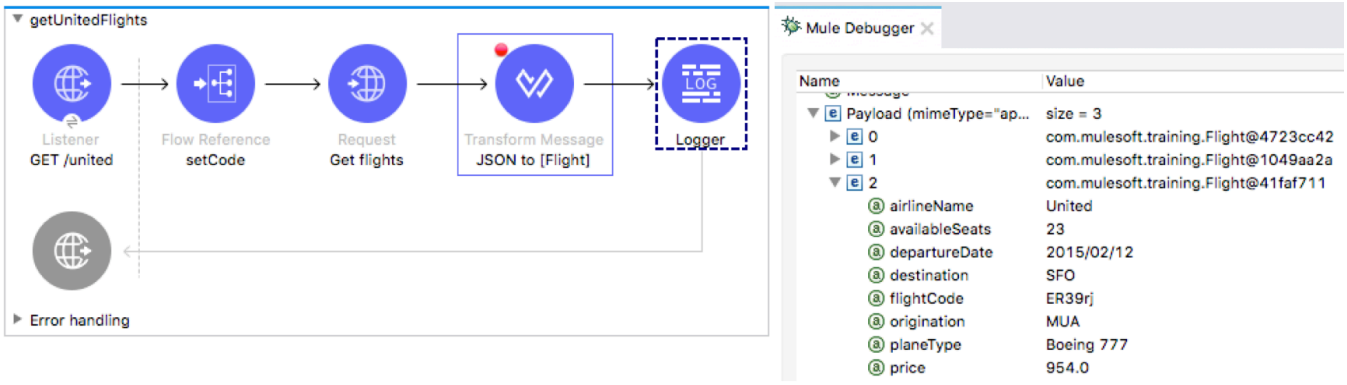
Walkthrough 8-4: Transform data from multiple services to a canonical format

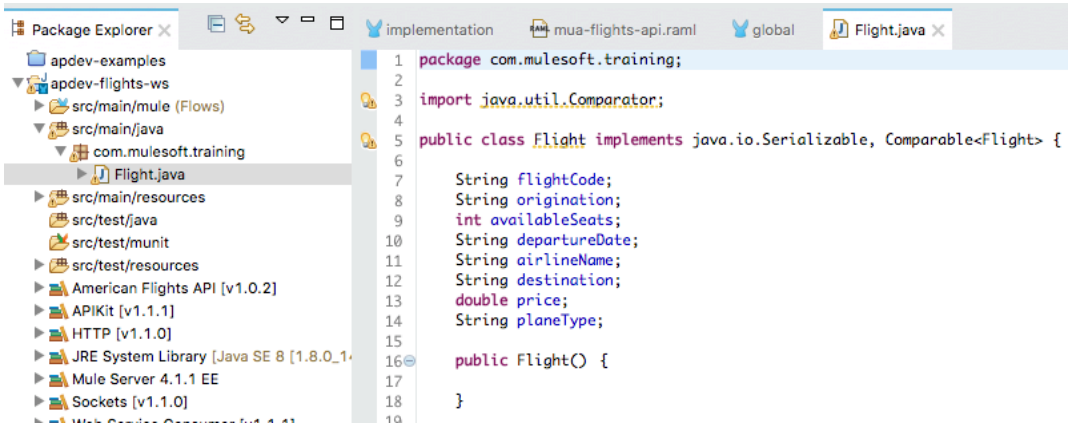
In this walkthrough, you will transform the JSON returned from the American and United web services and the SOAP returned form the Delta web service to the same format. You will:

- Define a metadata type for the Flight Java class.
- Transform the results from RESTful and SOAP web service calls to a collection of Flight objects.



Review Flight.java

1. Return to apdev-flights-ws in Anypoint Studio.
2. Open Flight.java in src/main/java and review the file.

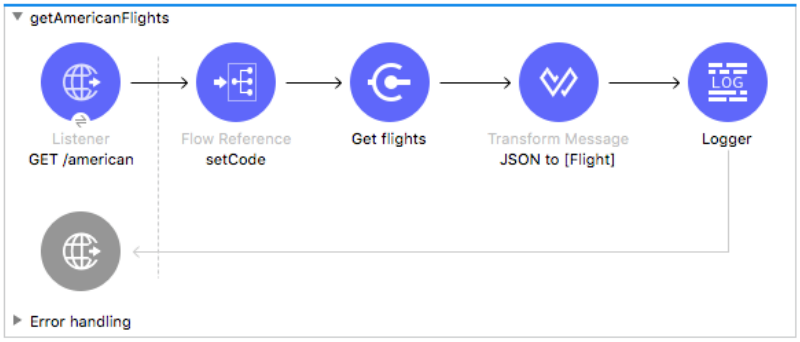


3. Close the file.

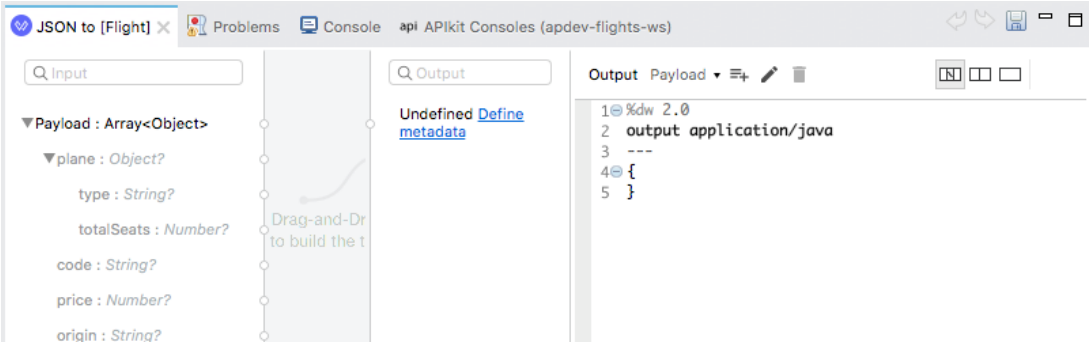
Change the American flow to return Java Flight objects instead of JSON

4. Return to getAmericanFlights in implementation.xml.
5. Add a Transform Message component to the end of the flow.
6. Add a Logger at the end of the flow.

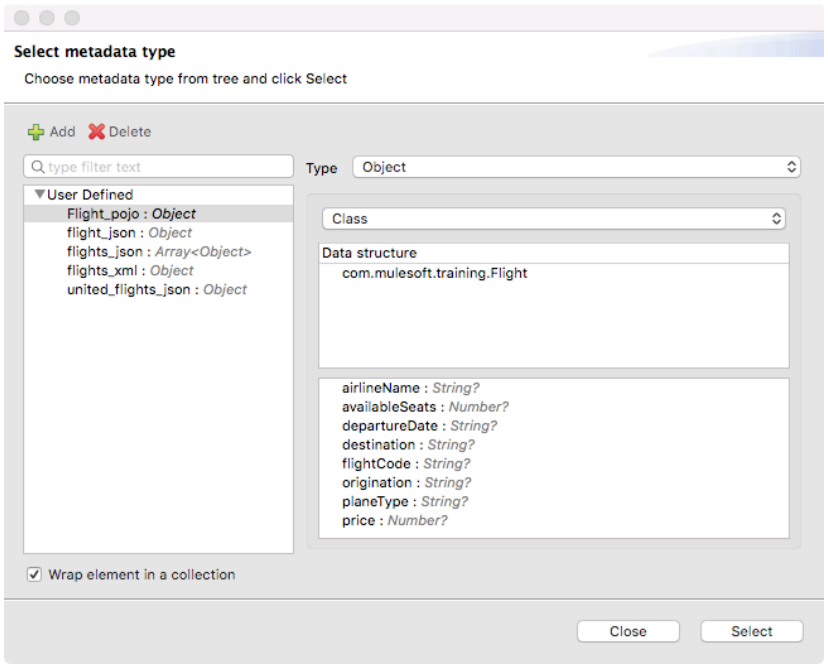
7. Change the name of the Transform Message component to JSON to [Flight].



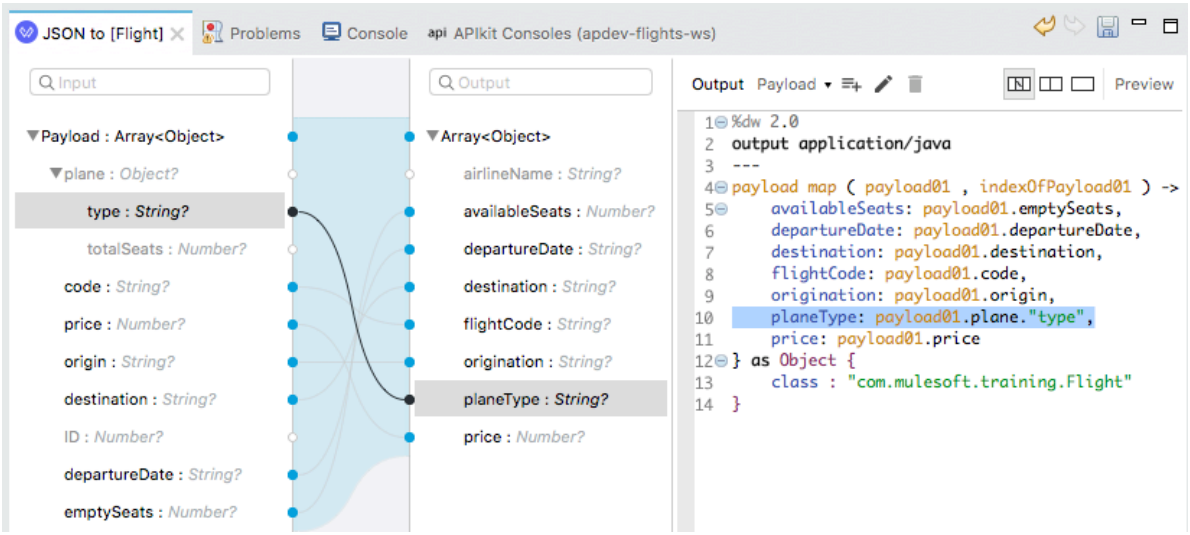
8. In the Transform Message properties view, look at the input section in the Transform Message properties view; you should see metadata already defined.



9. In the output section of the Transform Message properties view, click the Define metadata link.
10. In the Select metadata type dialog box, select the user-defined Flight_pojo.
11. Select Wrap element in a collection in the lower-left corner.



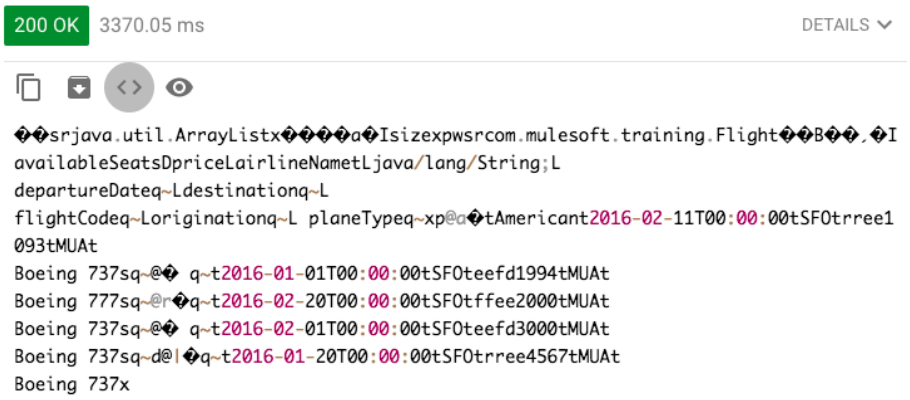
12. Click Select; you should now see output metadata in the output section of the Transform Message properties view.
13. Map fields (except ID and totalSeats) by dragging them from the input section and dropping them on the corresponding field in the output section.



14. Double-click the airlineName field in the output section.
15. In the generated DataWeave expression, change the airlineName value from null to "American".

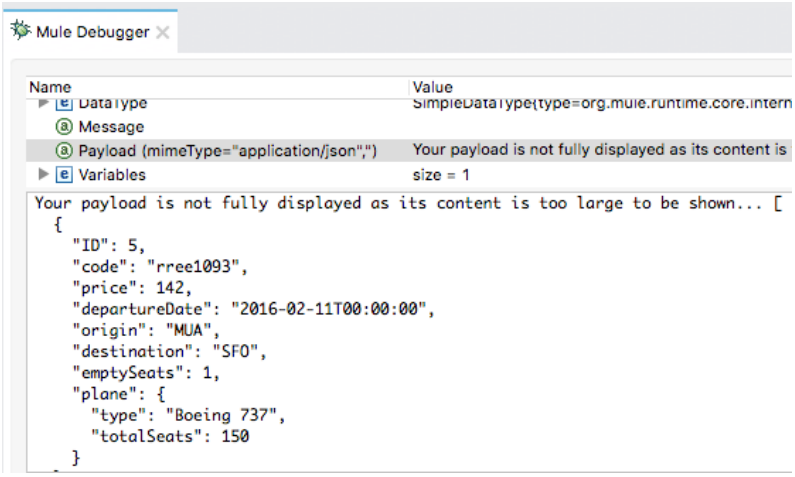
Test the application

16. Save to redeploy the project.
17. In Advanced REST Client, change the URL and make a request to http://localhost:8081/american; you should see a representation of a collection of Java objects.

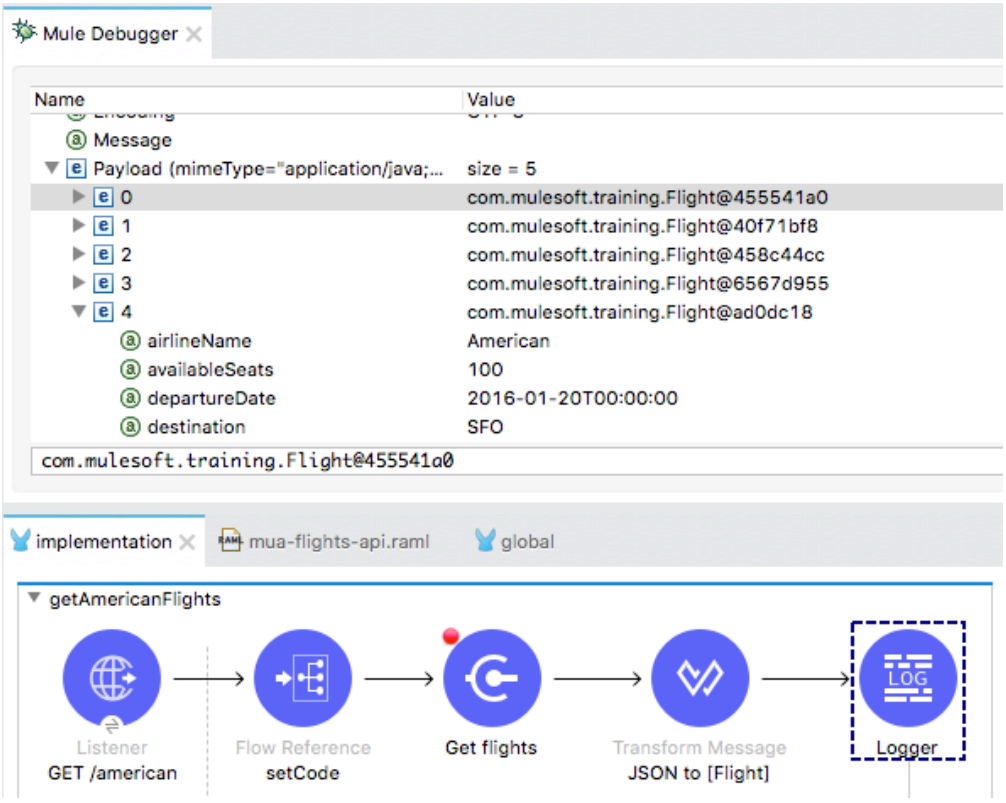


Debug the application

18. Return to Anypoint Studio.
19. Stop the project.
20. Add a breakpoint to the Get flights operation.
21. Debug the project.
22. In Advanced REST Client, make another request to http://localhost:8081/american.
23. In the Mule Debugger, step to the Transform message component and examine the payload.



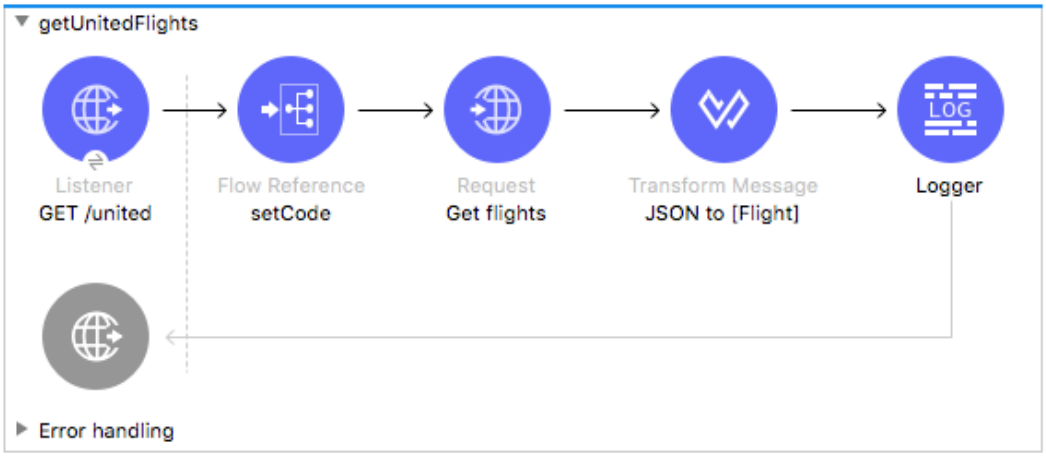
24. Step to the Logger and look at the payload it should be a collection or Flight objects.



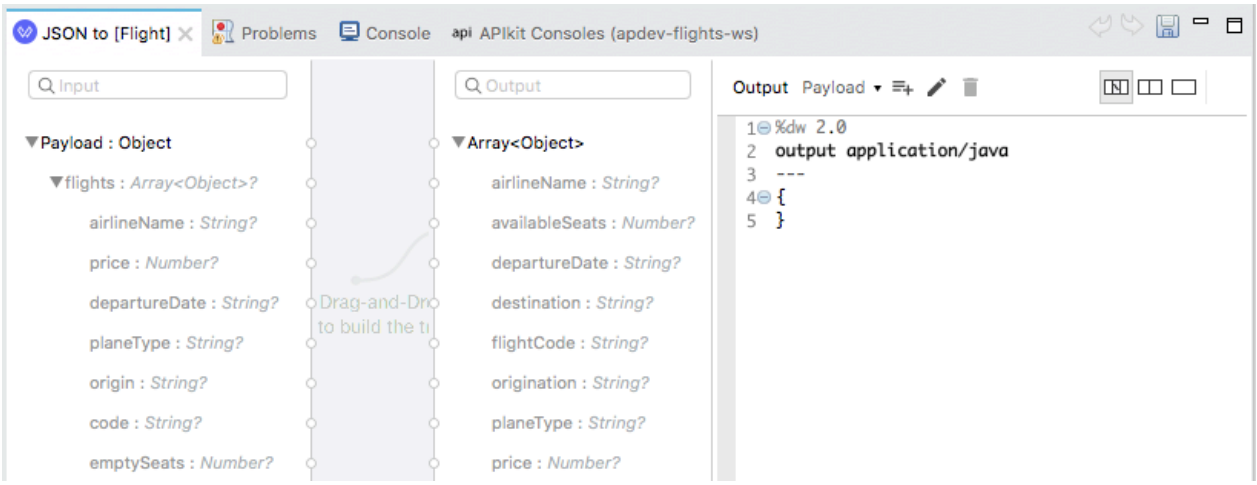
25. Step through the rest of the application and switch perspectives.

Change the United airline flows to return Java Flight objects instead of JSON

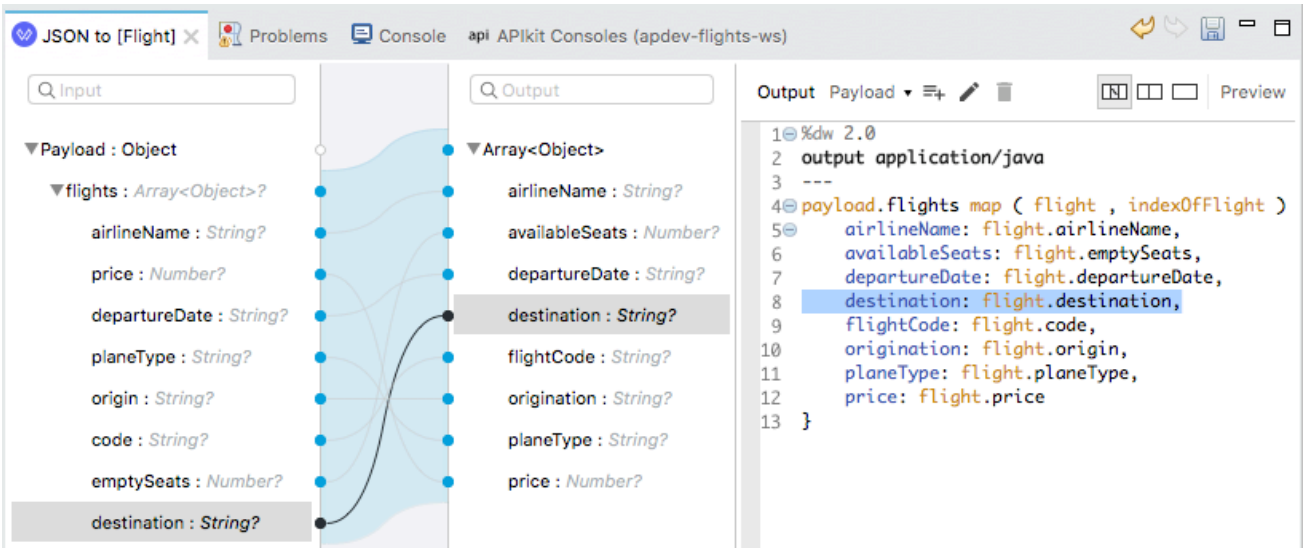
26. Return to geUnitedFlights in implementation.xml.
27. Add a Transform Message component at the end of the flow.
28. Change the name of the Transform Message component to JSON to [Flight].
29. Add a Logger to the end of the flow.



30. In the Transform Message properties view, look at the input section; you should see metadata already defined.
31. In the output section of the Transform Message properties view, click the Define metadata link.
32. In the Select metadata type dialog box, select the user-defined Flight_pojo.
33. Select Wrap element in a collection in the lower-left corner.
34. Click Select; you should now see output metadata in the output section of the Transform Message properties view.



35. Map fields by dragging them from the input section and dropping them on the corresponding field in the output section.



Note: If you do not see the object type set to com.mulesoft.training.Flight at the end of the DataWeave code, return to the course snippets.txt file and copy the DataWeave code to transform an object to a custom data type, and paste it at the end of the expression.

Note: If you get any DataWeave errors, ignore them.

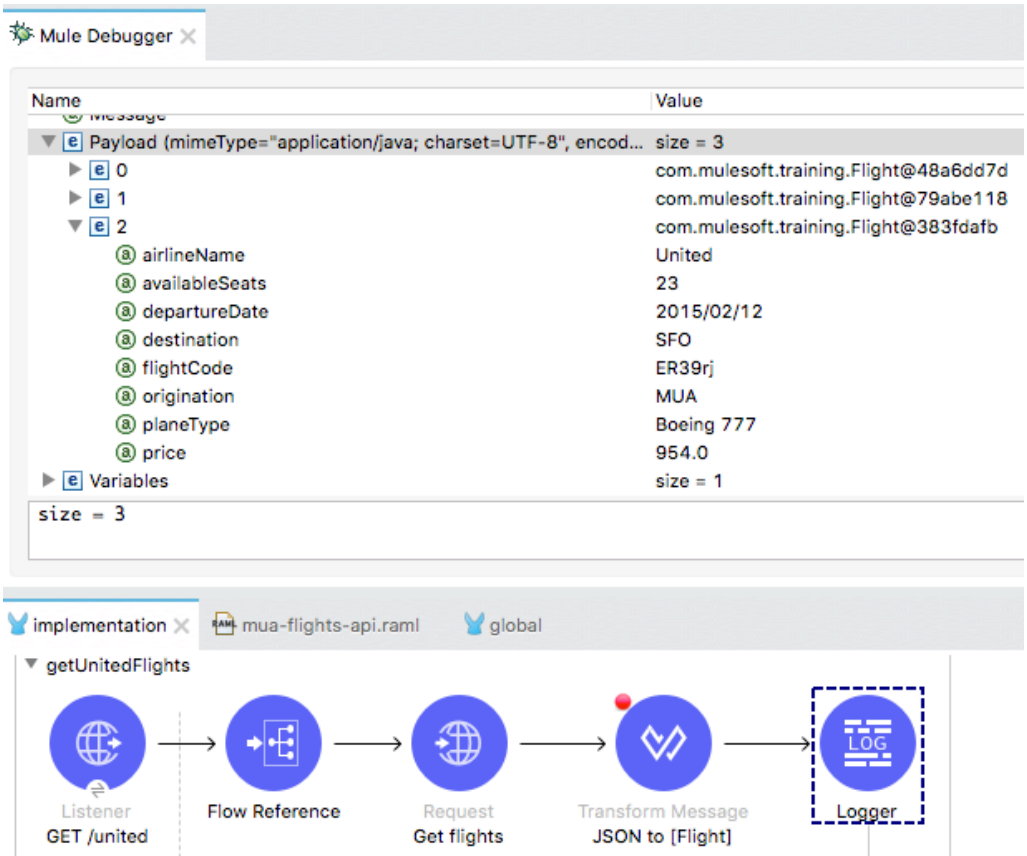
```
%dw 2.0
output application/java
---
payload.flights map ( flight , indexOfFlight ) -> {
    airlineName: flight.airlineName,
    availableSeats: flight.emptySeats,
    departureDate: flight.departureDate,
    destination: flight.destination,
    flightCode: flight.code,
    origination: flight.origin,
    planeType: flight.planeType,
    price: flight.price
} as Object {
    class : "com.mulesoft.training.Flight"
}

```

Debug the application

36. Add a breakpoint to the Transform Message component.
37. Save the files to redeploy the project.
38. In Advanced REST Client, change the URL to make a request to http://localhost:8081/united.

39. In the Mule Debugger, examine the payload.

40. Step to the Logger and look at the payload it should be a collection or Flight objects.

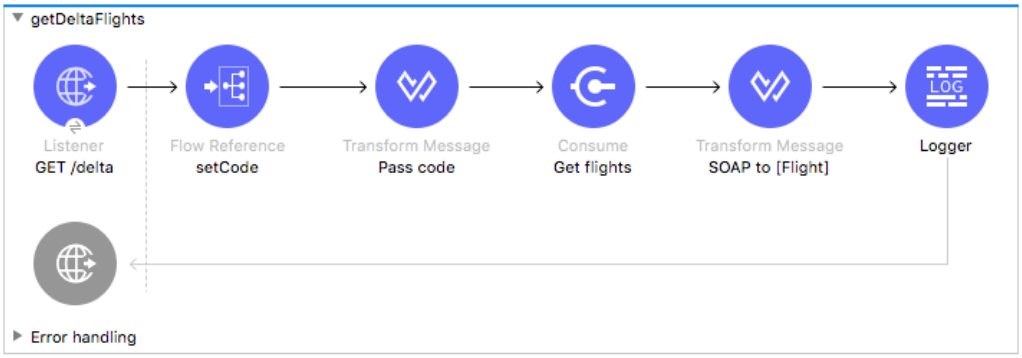


41. Step through the rest of the application and switch perspectives.

Change the Delta flow to return Java Flight objects

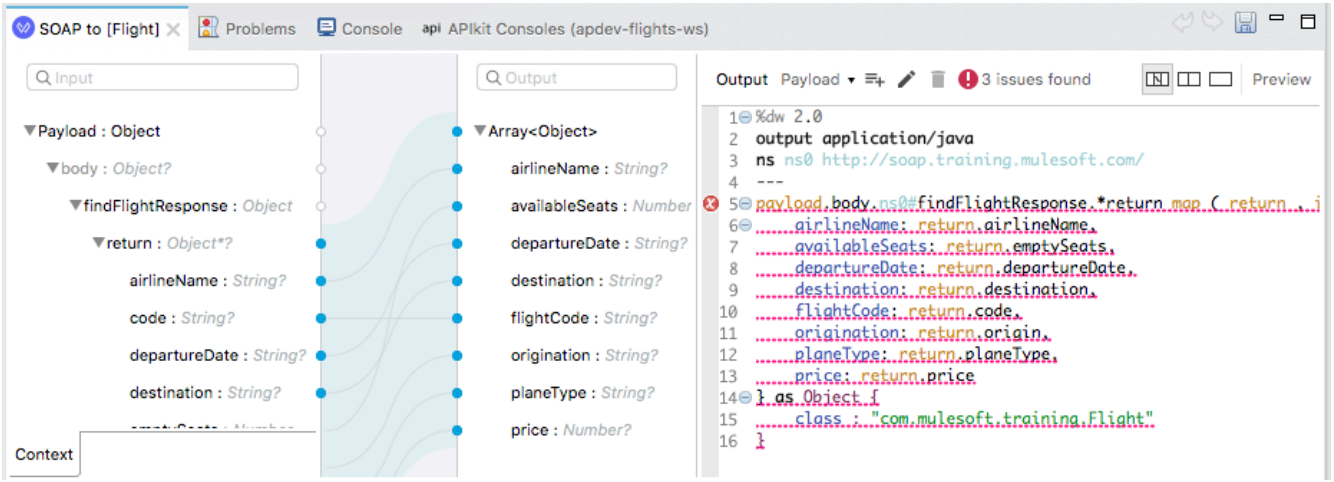
42. Return to geDeltaFlights in implementation.xml.
43. Change the name of the Transform Message component to SOAP to [Flight].

44. Add a Logger to the end of the flow.



45. Look at the input section in the Transform Message properties view; you should see metadata already defined.
46. In the output section of the Transform Message properties view, click the Define metadata link.
47. In the Select metadata type dialog box, select the user-defined Flight_pojo.
48. Select Wrap element in a collection in the lower-left corner.
49. Click Select; you should now see output metadata in the output section of the Transform Message properties view.
50. Map the fields by dragging them from the input section and dropping them on the corresponding field in the output section.

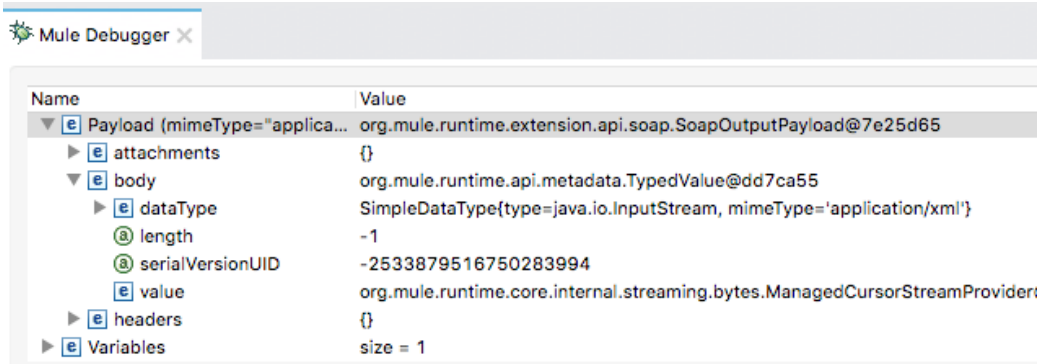
    Note: If you get a DataWeave error and cannot drag and drop the fields, delete all the existing DataWeave code, return to the course snippets.txt file and copy the DataWeave code for the Delta transformation, and then return to Anypoint Studio and paste it in the expression section.



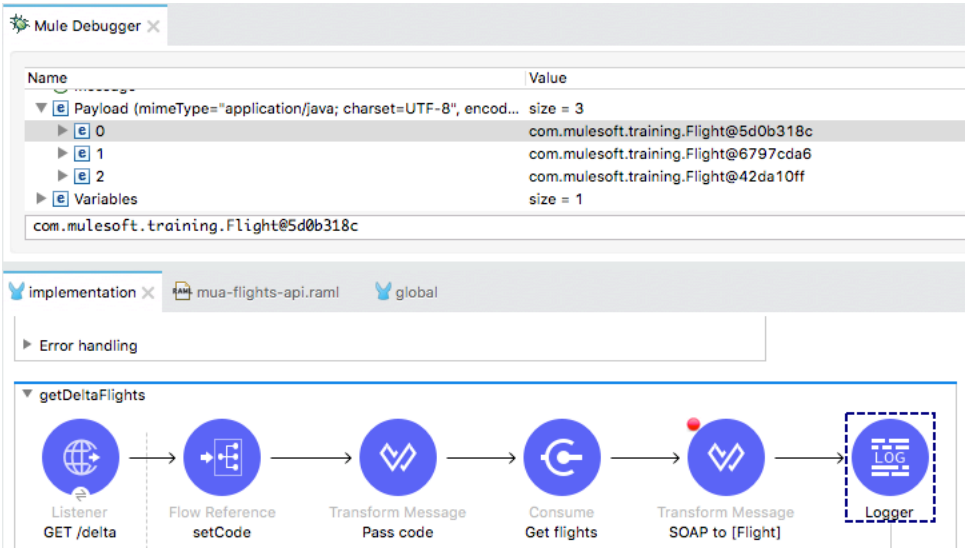
Debug the application

51. Add a breakpoint to the Transform Message component.
52. Save the file to redeploy the project.

53. In Advanced REST Client, change the URL to make a request to http://localhost:8081/delta.
54. In the Mule Debugger, examine the payload.



55. Step to the Logger and look at the payload it should be a collection or Flight objects.



56. Step through the rest of the application and switch perspectives.