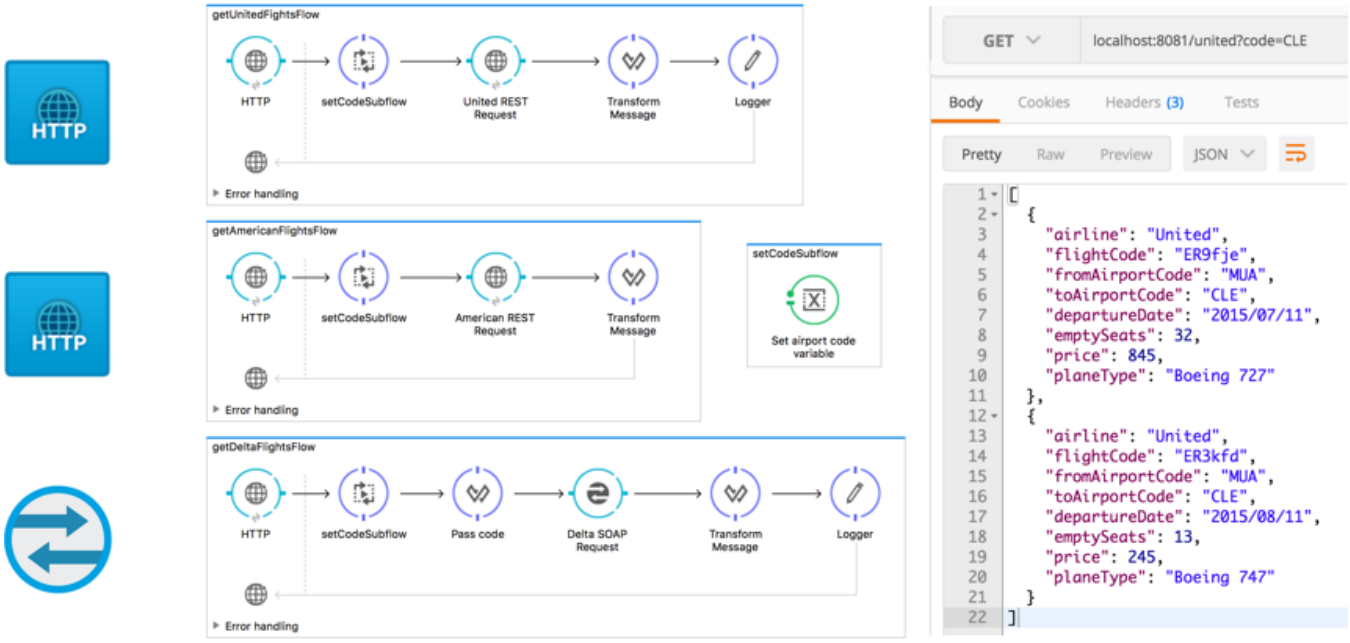
Module 7: Consuming Web Services



2

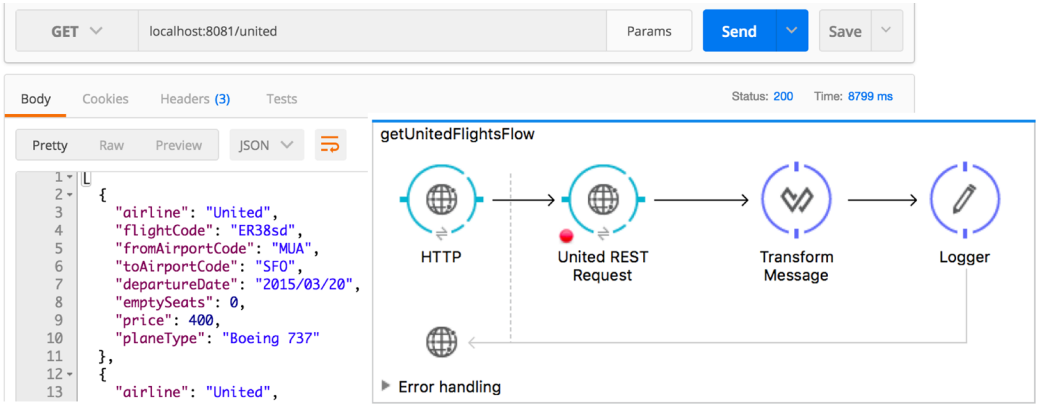
Objectives:

- Consume RESTful web services with and without parameters.
- Consume RESTful web services that have RAML definitions.
- Consume SOAP web services.
- Use DataWeave to pass parameters to SOAP web services.

Walkthrough 7-1: Consume a RESTful web service

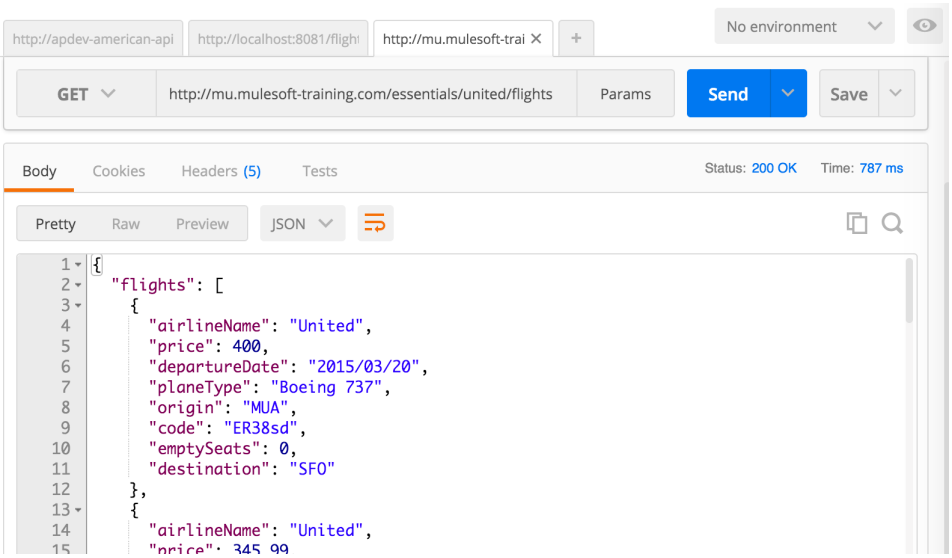
In this walkthrough, you consume a RESTful web service that returns a list of all United flights as JSON. You will:

- Create a new flow to call a RESTful web service.
- Use an HTTP Request endpoint to consume a RESTful web service.
- Use DataWeave to transform the JSON response into JSON specified by an API.

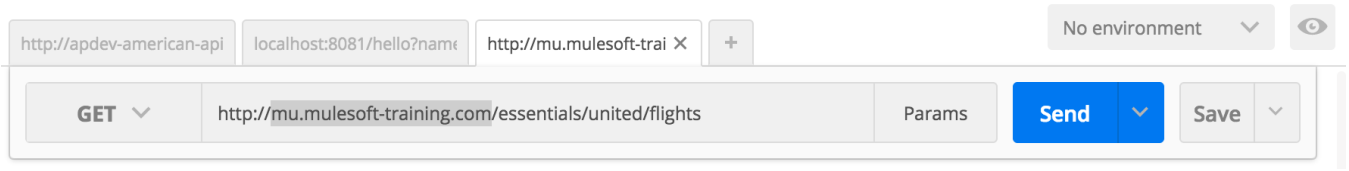


Make a request to the web service

1. Return to the course snippets.txt file.
2. Locate and copy the United RESTful web service URL.
3. In Postman, make a new tab.
4. Make a GET request to this URL; you should see JSON data for the United flights as a response.



5. Look at the destination values; you should see SFO, LAX, CLE, PDX, and PDF.
6. Copy the host name from the URL; this should be something like mu.mulesoft-training.com.

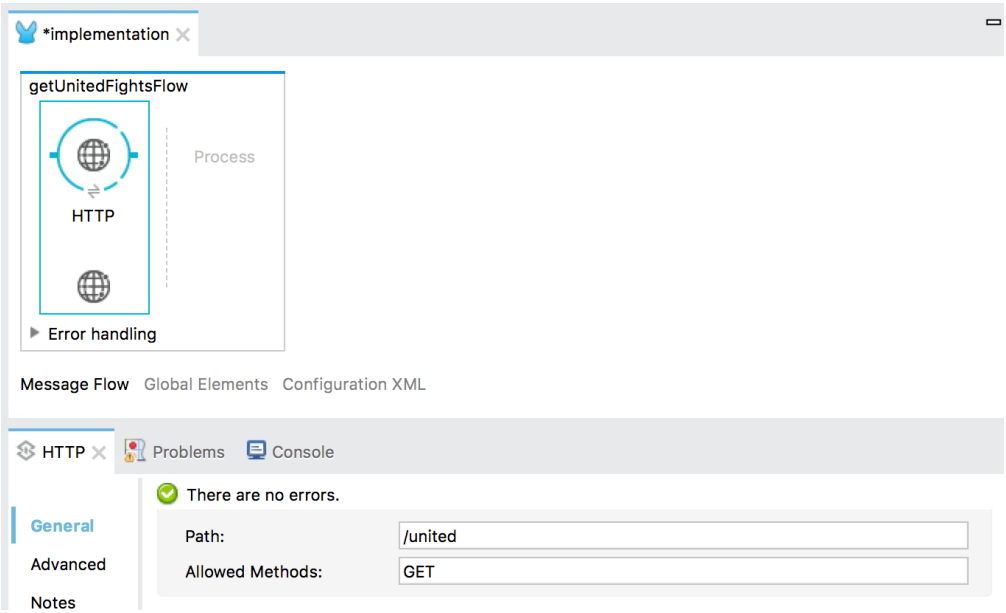


Add a new flow with an HTTP Listener endpoint

7. Return to implementation.xml in Anypoint Studio.
8. Drag out an HTTP connector and drop it in the canvas.
9. Double-click the name of the flow in the canvas and give it a new name of getUnitedFlightsFlow.

Configure the HTTP Listener endpoint

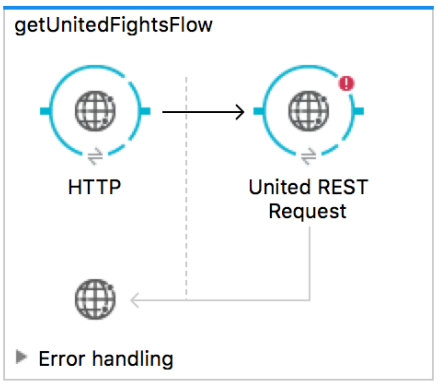
10. In the HTTP properties view, set the connector configuration to the existing HTTP_Listener_Configuration.
11. Set the path to /united.
12. Set the allowed methods to GET.



Add an HTTP Request endpoint

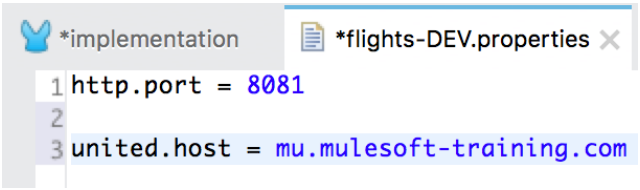
13. Drag out another HTTP connector and drop it into the process section of the flow.

14. Change the HTTP Request endpoint display name to United REST Request.



Configure the HTTP Request connector

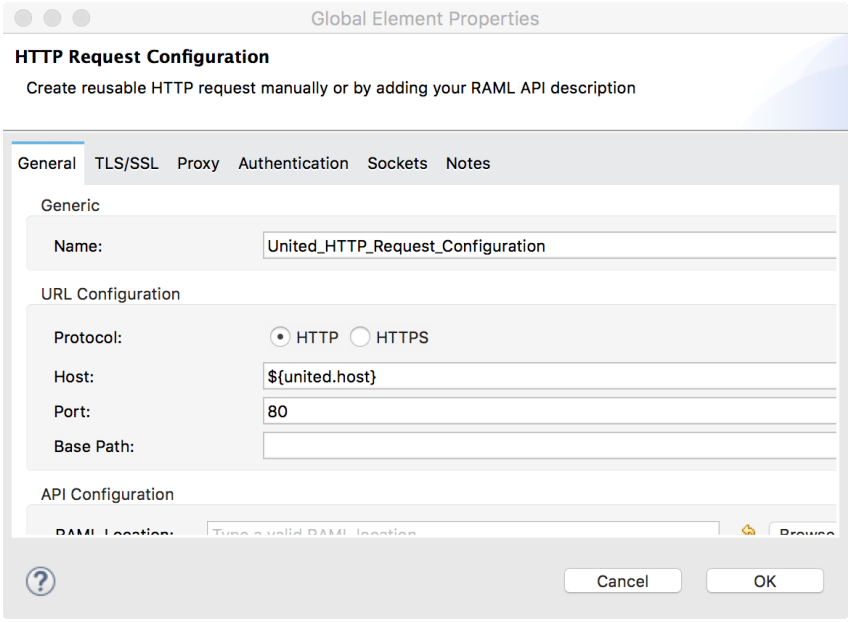
15. Return to flights-DEV.properties in src/main/resources.
16. Create a property called united.host and set it equal to the value you copied from the URL.



17. Save the file.
18. Return to global.xml.
19. In the Global Elements view, click Create.
20. In the Choose Global Type dialog box, select Connector Configuration > HTTP Request Configuration and click OK.

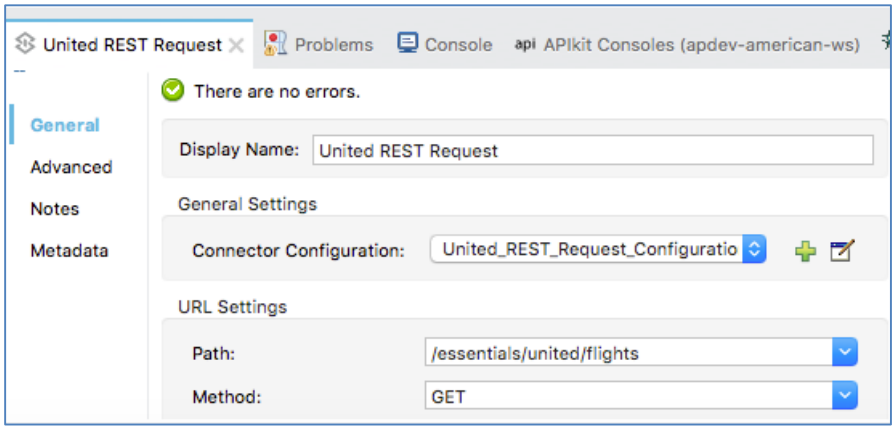
21. In the Global Element Properties dialog box, set the following values and click OK.

    - Name: United_REST_Request_Configuration
    - Host: ${united.host}
    - Port: *80*
    - Base Path: *Leave blank*



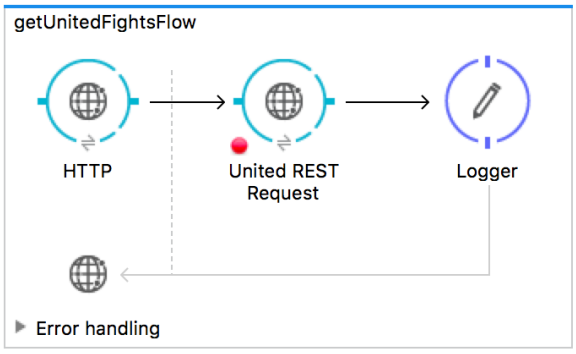
Configure the HTTP Request endpoint

22. Return to implementation.xml.
23. In the United REST Request properties view, set the connector configuration to the existing United_REST_Request_Configuration.
24. Set the path to /essentials/united/flights.
25. Set the method to GET.

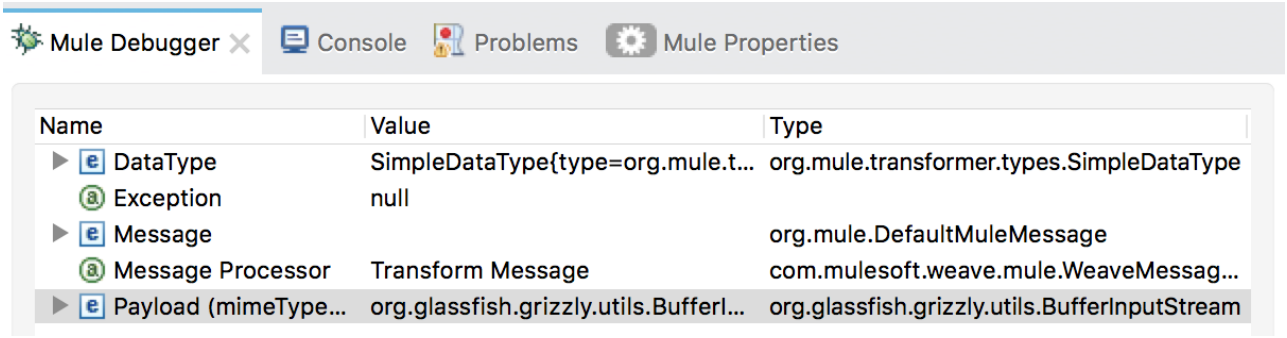


Test the application

26. Make sure the United REST Request endpoint has a breakpoint.
27. Add a Logger after the United REST Request endpoint.

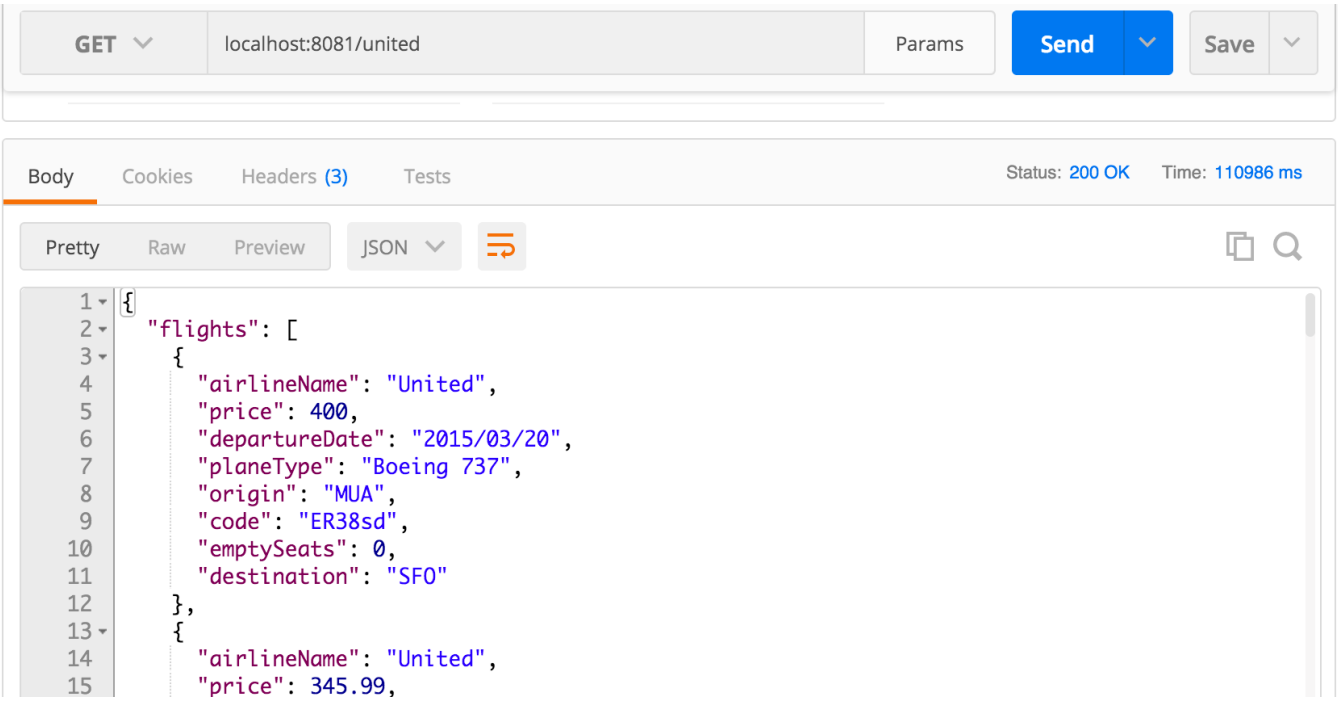


28. Debug the project.
29. In Postman, return to the tab with the localhost request.
30. Make a request to http://localhost:8081/united.
31. In the Mule Debugger, step to the Logger and look at the payload; it should be of type BufferInputStream.



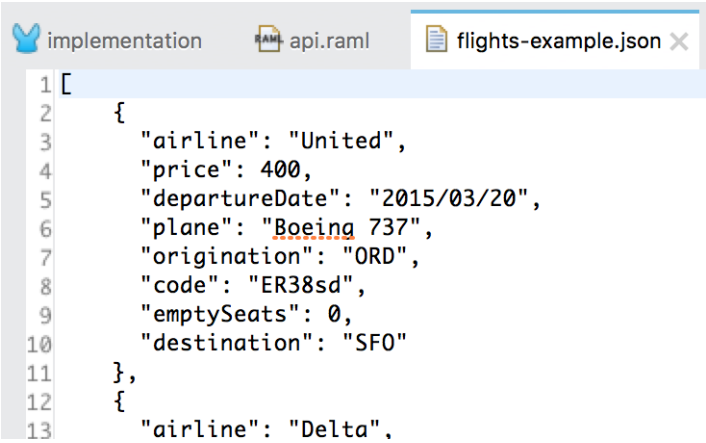
32. Step through the application.
33. Return to Postman; you should see the JSON flight data returned.

34. Examine the data structure of the JSON response.



Review the structure for the desired JSON response

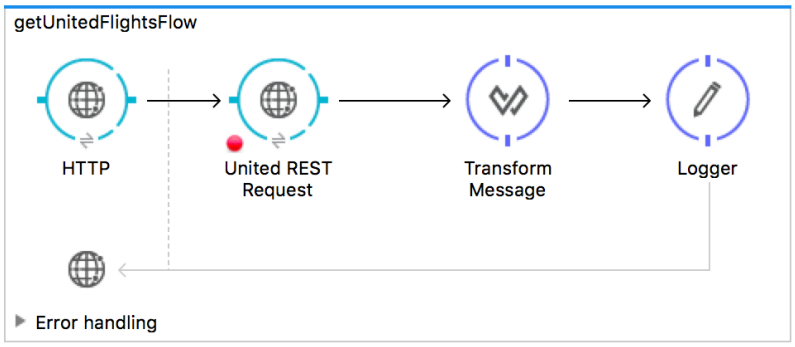
35. In Anypoint Studio, open api.raml in src/main/api and review the response example.
36. Open flights-example.json in src/main/resources and review the sample.



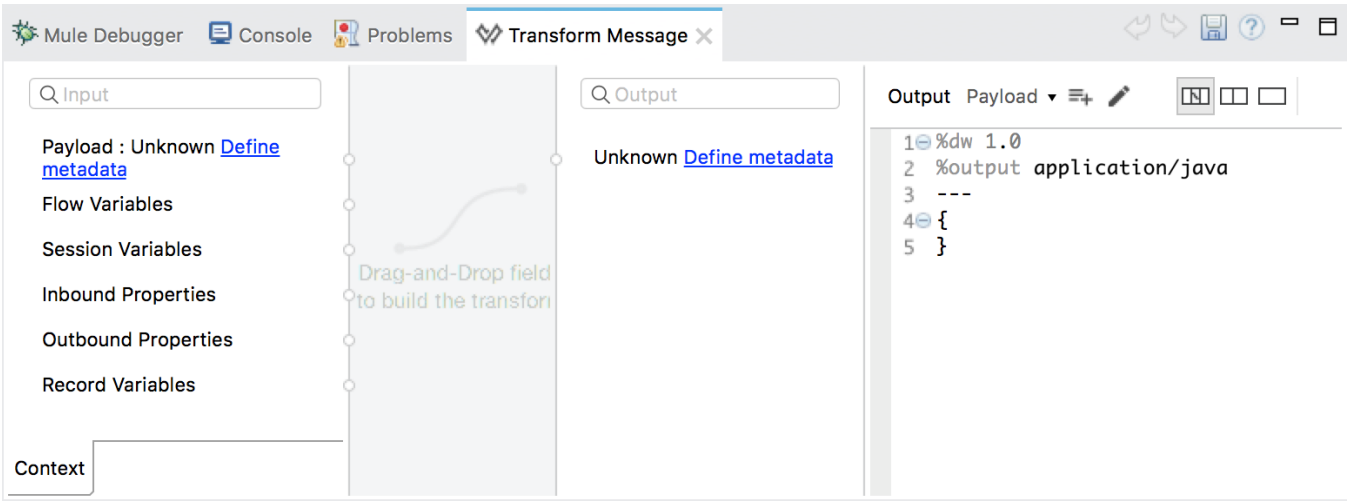
Add input metadata for the transformation

37. Return to implementation.xml.

38. Add a Transform Message component after the United REST Request endpoint.

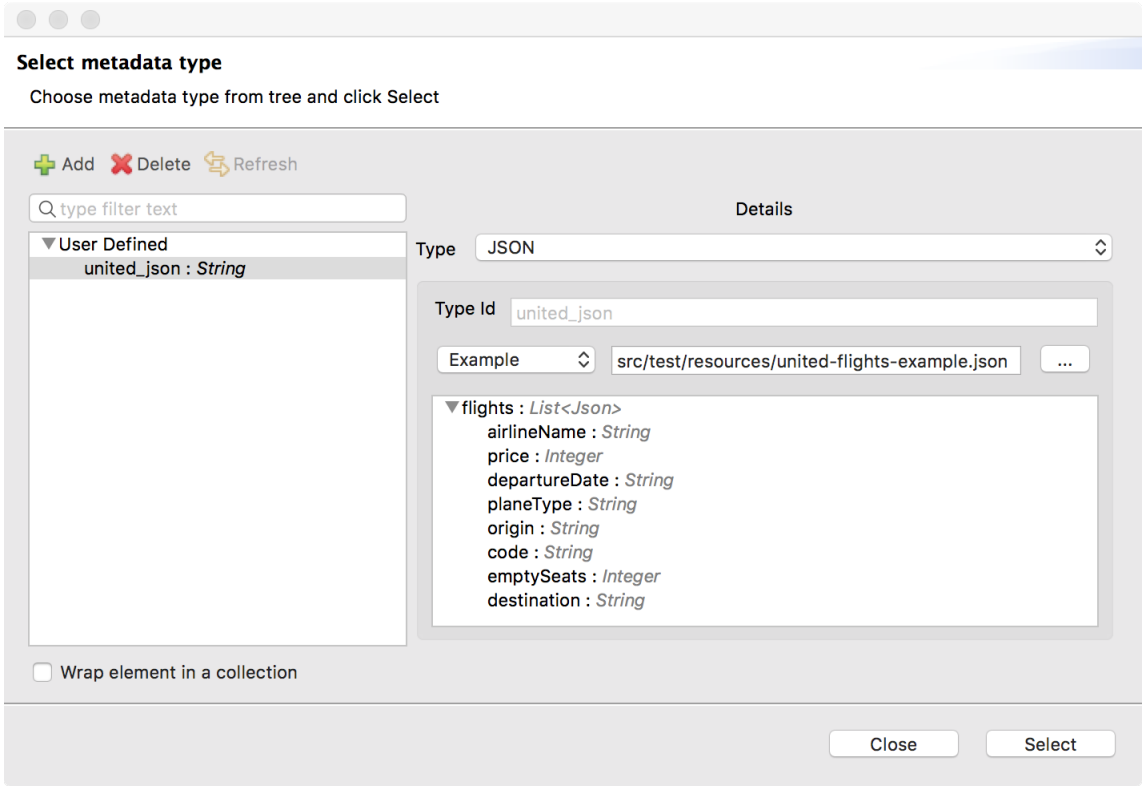


39. In the input section of the Transform Message properties view, click the Define metadata link for the payload.

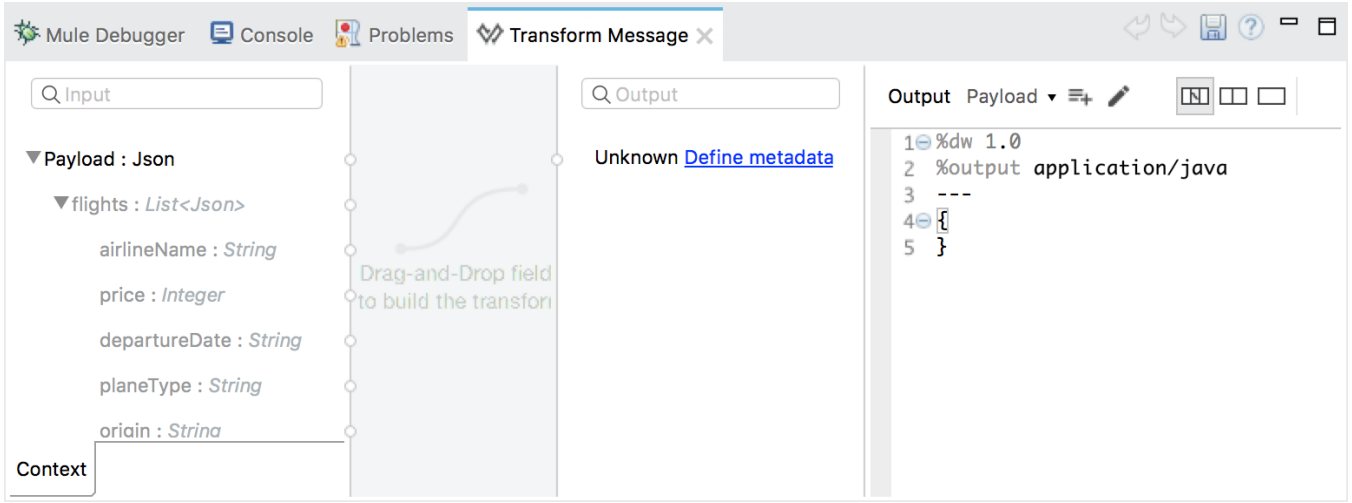


40. In the Select metadata type dialog box, click the Add button.
41. In the Create new type button dialog box, set the type id to united_json.
42. Click Create type.
43. In the Select metadata type dialog box, set the type to JSON.
44. Change the Schema selection to Example.
45. Click the browse button.
46. Click the browse button and navigate to the projects's src/test/resources folder.

47. Select united-flights-example.json and click Open; you should see the example data for the metadata type.



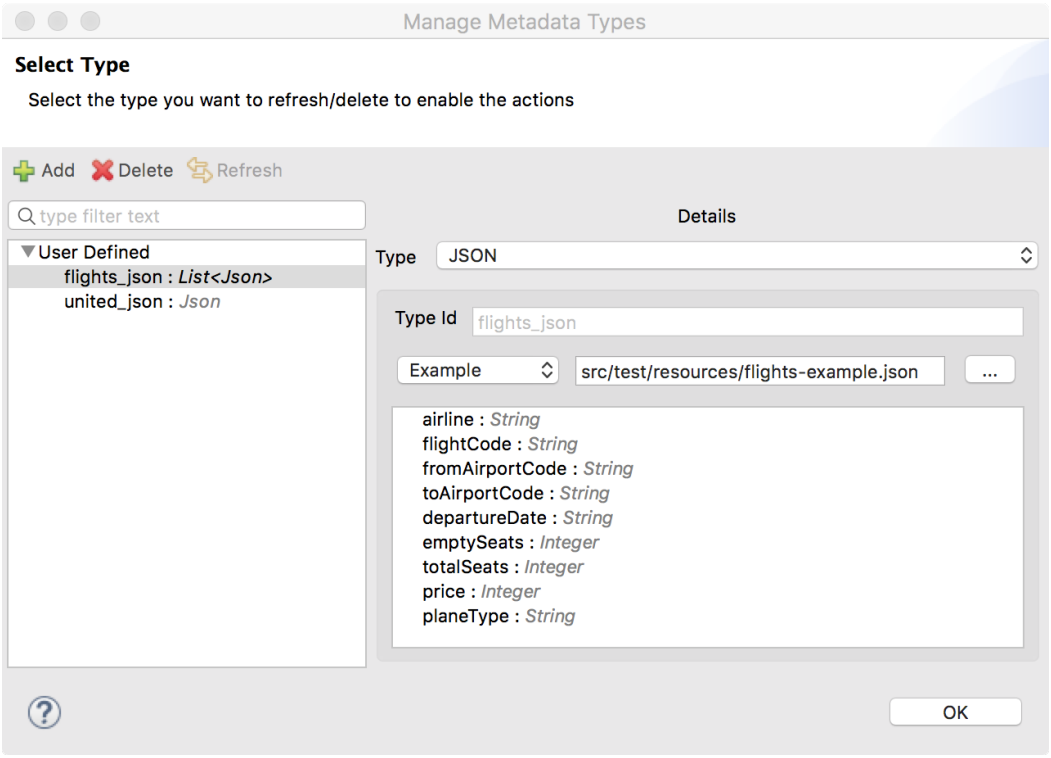
48. Click Select; you should now see output metadata in the input section of the Transform Message properties view.



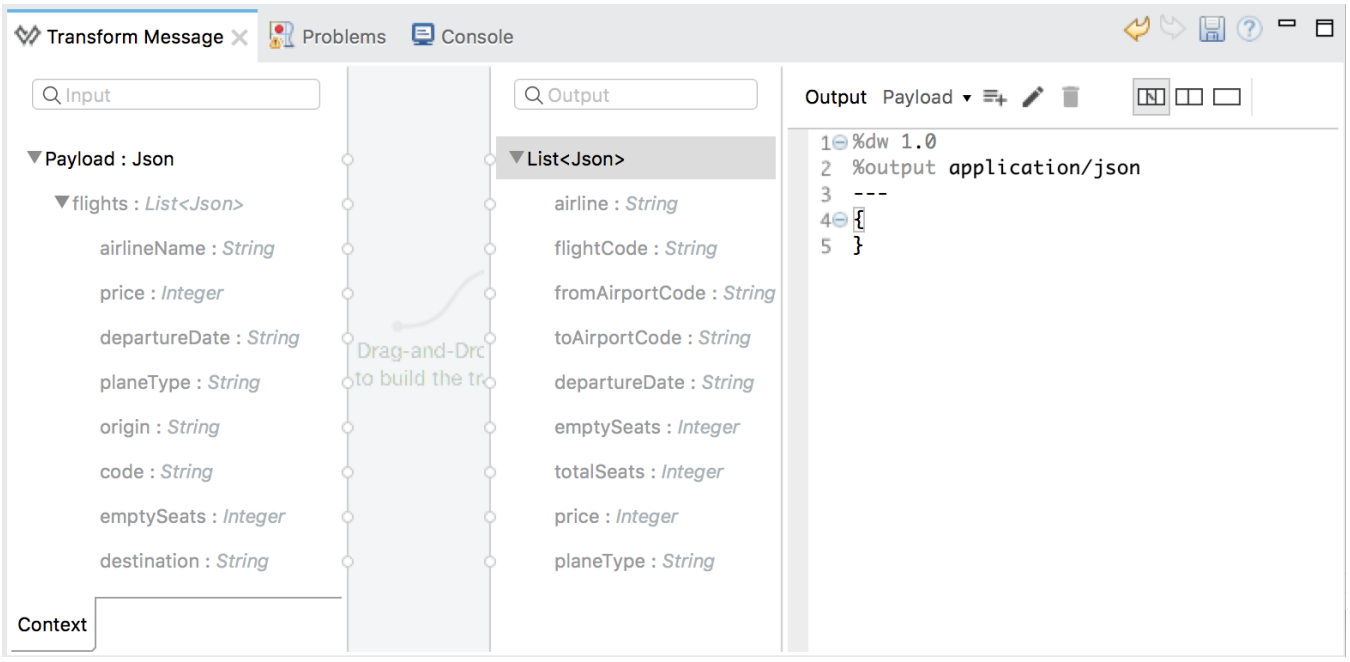
Add output metadata for the transformation

49. In the output section of the Transform Message properties view, find where the output is set to application/java.
50. Click the Define metadata link.
51. In the Select metadata type dialog box, click the Add button.
52. In the Create new type dialog box, set the type id to flights_json.
53. Click Create type.
54. In the Select metadata type dialog box, set the type to JSON.
55. Change the Schema selection to Example.
56. Click the browse button and navigate to the projects's src/test/resources folder.
57. Select flights-example.json and click Open; you should see the example data for the metadata type.

    Note: Be sure to select the JSON file with flights plural, not singular.



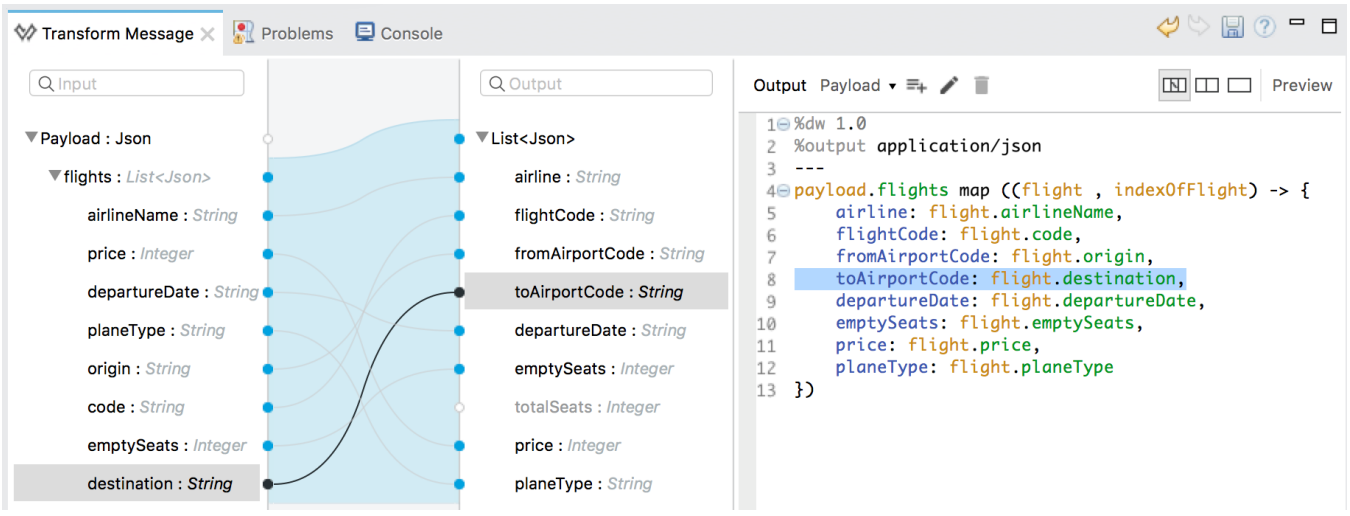
58. Click Select; you should now see output metadata in the output section of the Transform Message properties view.



Create the transformation

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

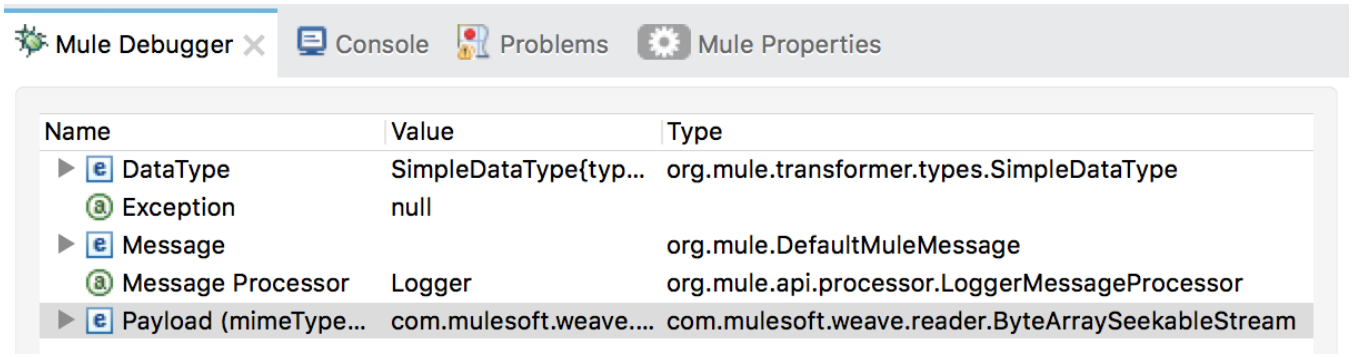
    Note: There is no input field to map to totalSeats.



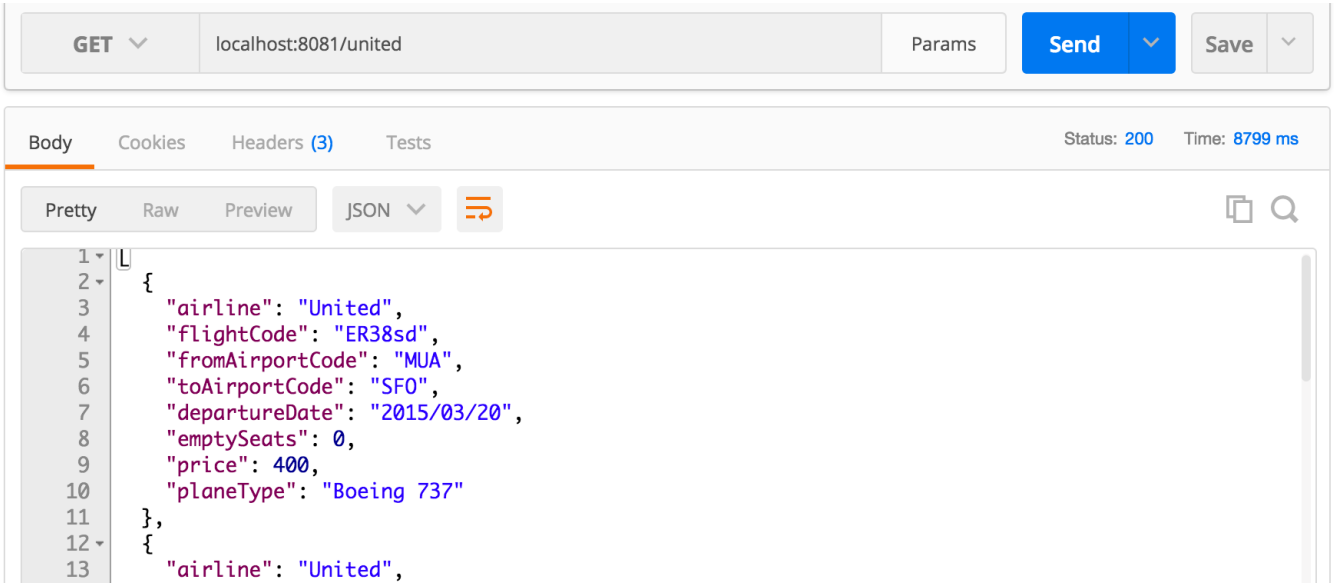
Test the application

60. Debug the project.

61. In Postman, make another request to http://localhost:8081/united.
62. In the Mule Debugger, step to the Logger; you should see the payload is now a DataWeave ByteArraySeekableStream.



63. Step through the application.
64. Return to Postman; you should see the flight data with the different JSON structure.

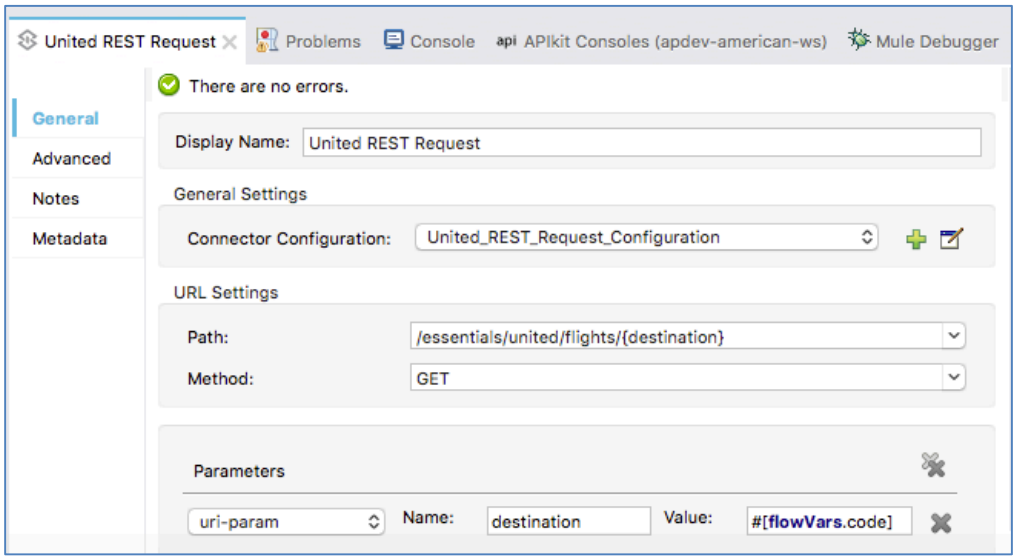


65. Return to Anypoint Studio and stop the project.

Walkthrough 7-2: Pass arguments to a RESTful web service

In this walkthrough, you retrieve United flights for a specific destination by setting the destination as a URI parameter. You will:

- Modify the HTTP Request endpoint to use a URI parameter for the destination.
- Set the destination to a static value.
- Create a flow variable to store the value of a query parameter with an airport code value.
- Set the destination to the dynamic value of this flow variable.

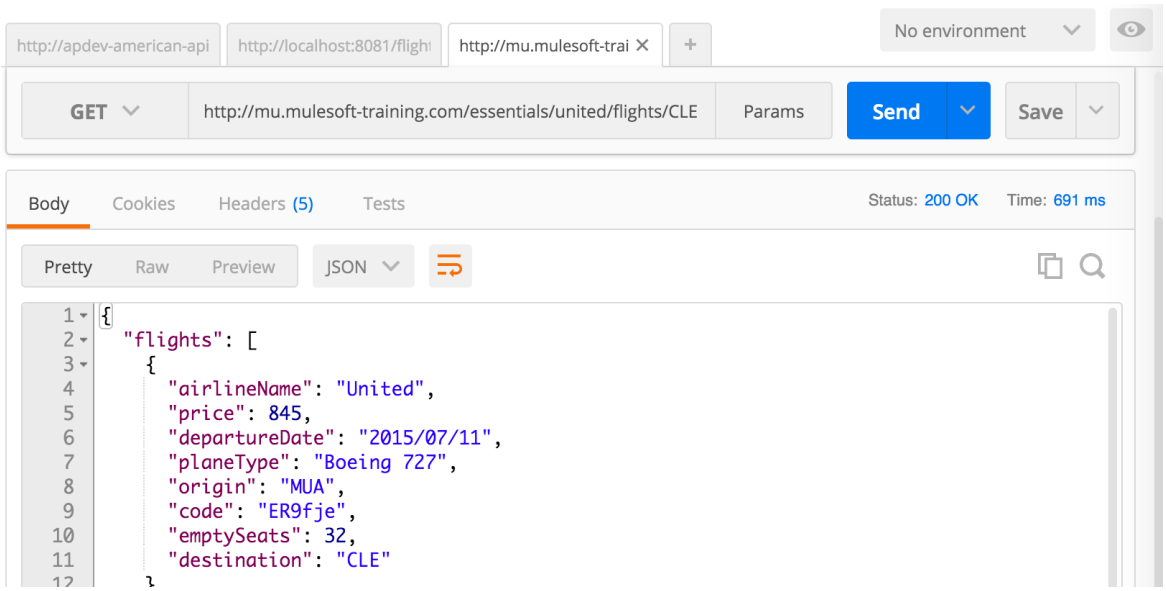


Make a request to the web service specifying a destination

1. Return to Postman and click the third tab, the one with the request to the United web service.
2. In the URL field, add the destination CLE as a URI parameter: http://mu.mulesoft-training.com/essentials/united/flights/CLE.

   Note: The API you are building for flights.raml has a query parameter called code for the destination; this existing United web service uses a URI parameter for the destination.

3. Send the request; you should see JSON data for only the flights to CLE.

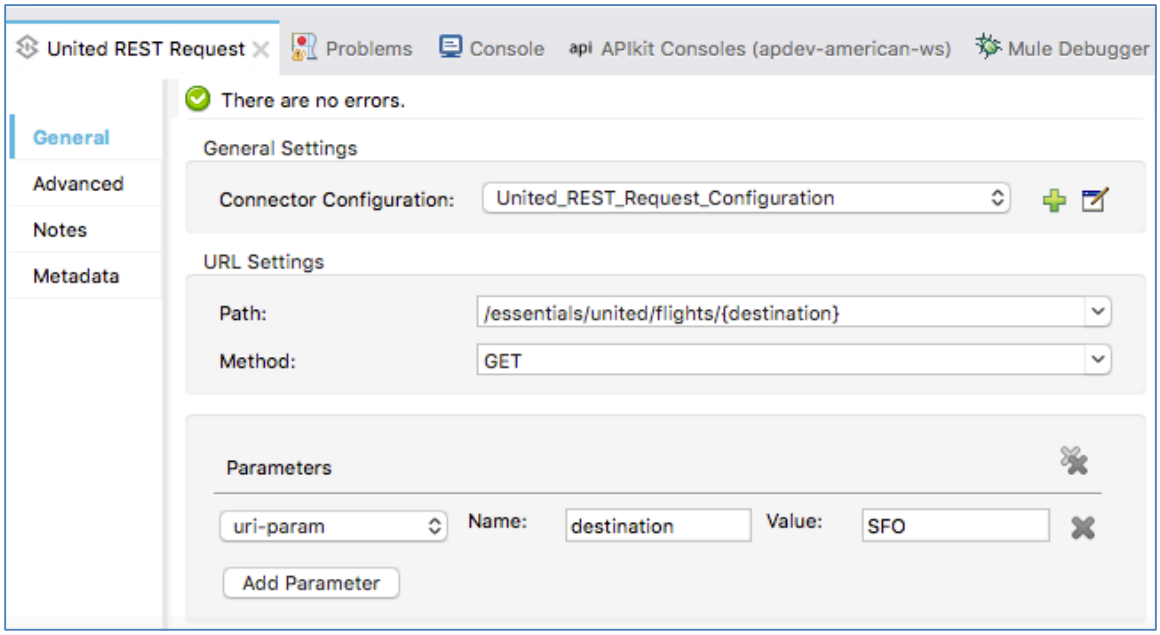


4. Make additional requests for destinations of LAX, SFO, PDX, or PDF.

Add a URI parameter with a static value

5. Return to implementation.xml in Anypoint Studio.
6. Double-click the United REST Request endpoint.
7. In the Properties view, locate the parameters section; there should be no parameters listed.
8. Change the United REST Request path to /essentials/united/flights/{destination}.
9. Click the Add Parameter button.
10. Select a parameter type of uri-param.
11. Set the name to destination.

12. Set the parameter value to SFO.



Test the application

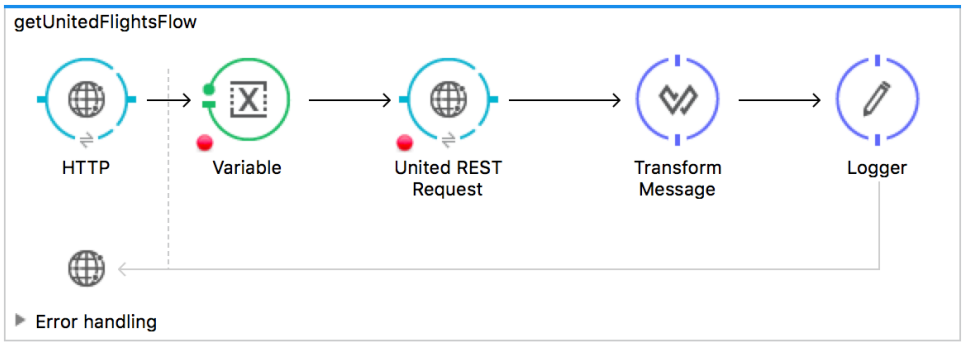
13. Run the project.
14. In Postman, return to the tab with the localhost requests.
15. Make a request to http://localhost:8081/united/; you should only get the flights to SFO.
16. Add a query parameter called code and set it to LAX.

17. Send the request; of course, you should still get only flights to SFO.

Create a variable to set the destination airport code

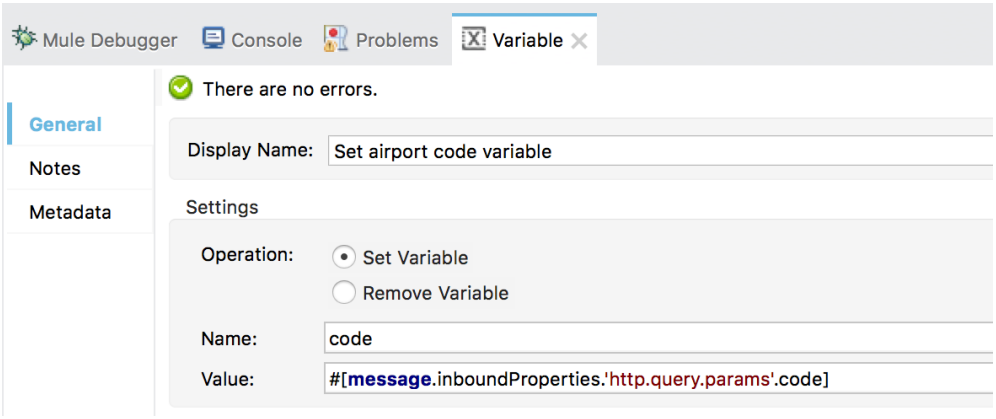
18. Return to Anypoint Studio.

19. Add a Variable transformer before the United REST Request endpoint.



20. In the Variable Properties view, change the display name to Set airport code variable.
21. Set the operation to Set Variable and the name to code.
22. Set the value to a query parameter called code.

```
#[message.inboundProperties.'http.query.params'.code]
```



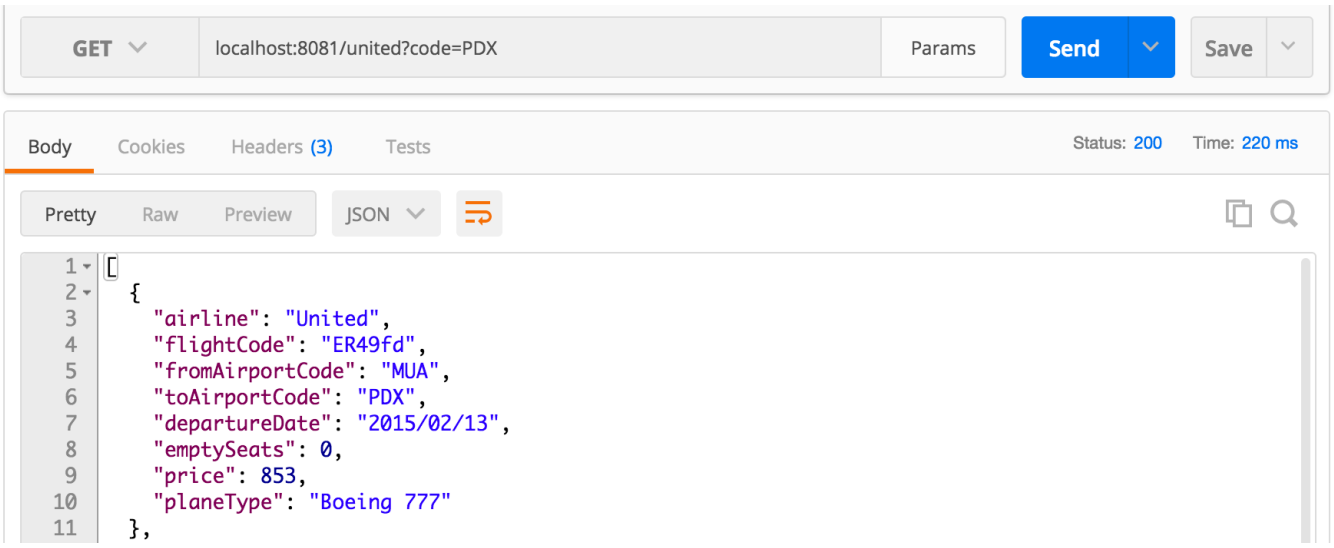
Change the REST request URI parameter to a dynamic value

23. In the United REST Request properties view, change the value of the uri-param from SFO to the value of the flow variable containing the airport code.

```
#[flowVars.code]
```

Test the application

24. Save and redeploy the application.
25. In Postman, send the same request again; this time you should only get flights to LAX.
26. Change the code to PDX and make the request; you should now see flights to PDX.



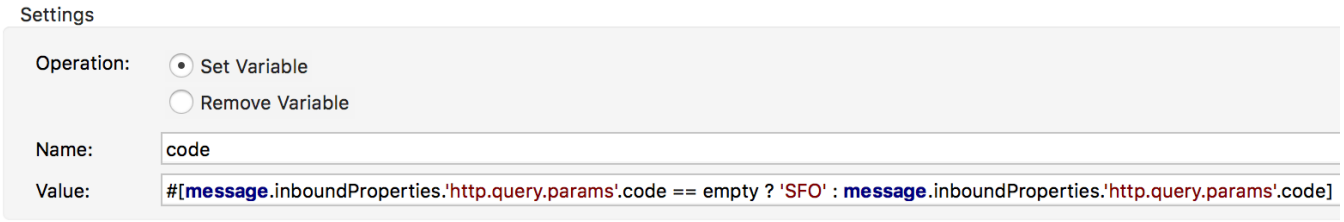
27. Remove the code parameter and make the request; you should get a 500 responses and an exception.

Modify the set airport code flow variable to assign a default value

28. Return to Anypoint Studio.

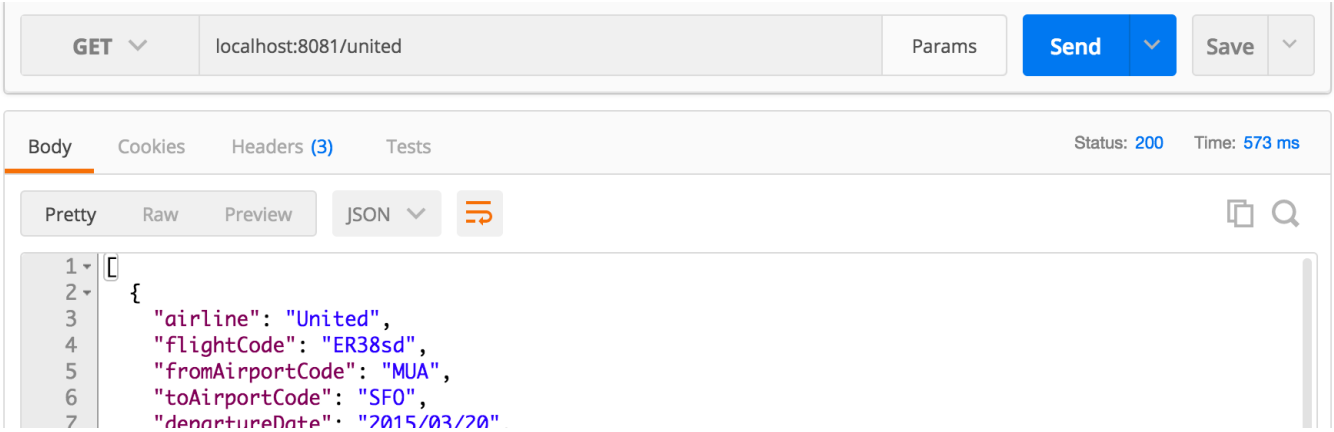
29. Modify the Set variable transformer to use a ternary expression to assign a default value of SFO if no query parameter is passed to the flow.

```
#[(message.inboundProperties.'http.query.params'.code == empty) ?
'SFO' : message.inboundProperties.'http.query.params'.code]
```



Test the application

30. Save and redeploy the application.
31. In Postman, send the same request again with no query parameter; this time you should get flights to SFO instead of an exception.

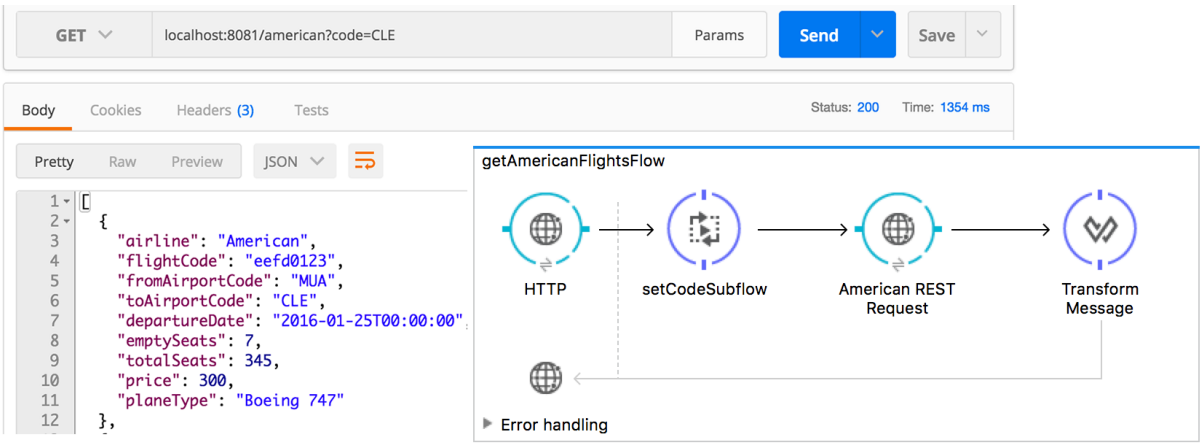


32. Return to Anypoint Studio and stop the project.

Walkthrough 7-3: Consume a RESTful web service that has a RAML definition

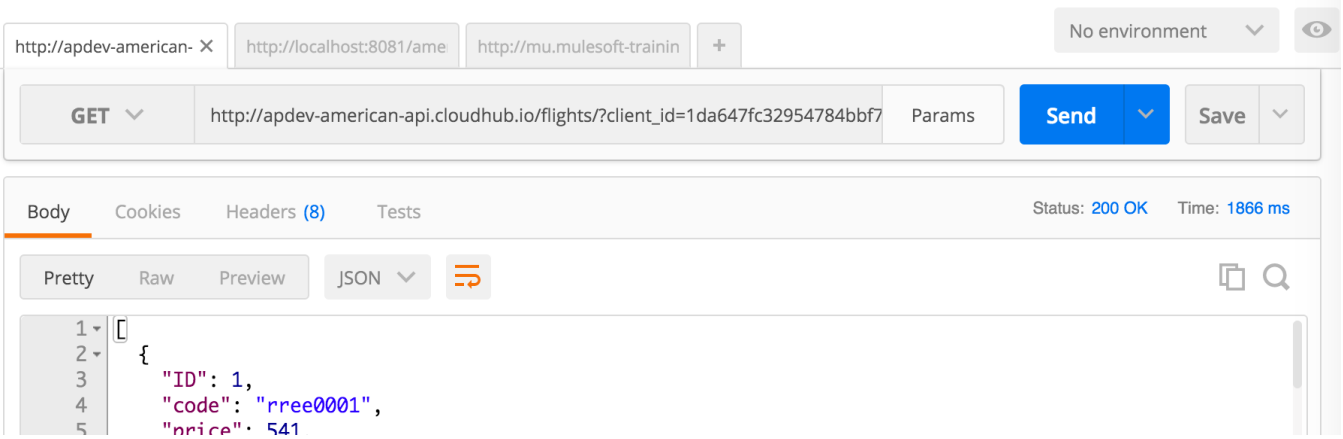
In this walkthrough, you consume the American flights RESTful web service that you built and deployed to the cloud. You will:

- Create a new flow to call a RESTful web service that has a RAML definition.
- Select the web service resource from the list provided by Anypoint Studio from the RAML file.
- Use DataWeave to transform the JSON response into JSON specified by an API.



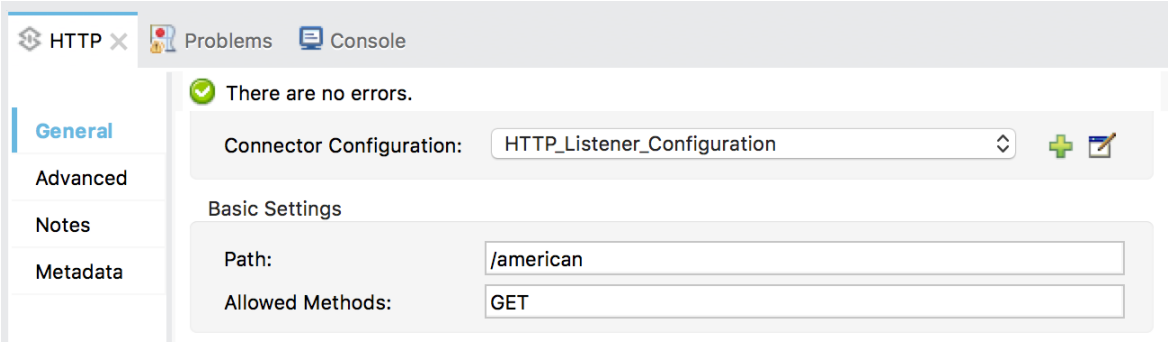
Make a request to the web service

1. In Postman, return to the first tab – the one with the request to your American Flights API http://apdev-american-api-{lastname#}.cloudhub.io/flights and that passes a client_id and client_secret.
2. Send the request; you should still see JSON data for the American flights as a response.



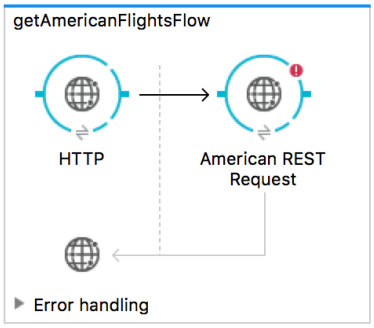
Add a new flow with an HTTP Listener endpoint

3. Return to implementation.xml.
4. Drag out another HTTP connector and drop it in the canvas.
5. Rename the flow to getAmericanFlightsFlow.
6. In the Properties view, set the connector configuration to the existing HTTP_Listener_Configuration.
7. Set the path to /american.
8. Set the allowed methods to GET.



Add an HTTP Request endpoint for a web service with a RAML definition

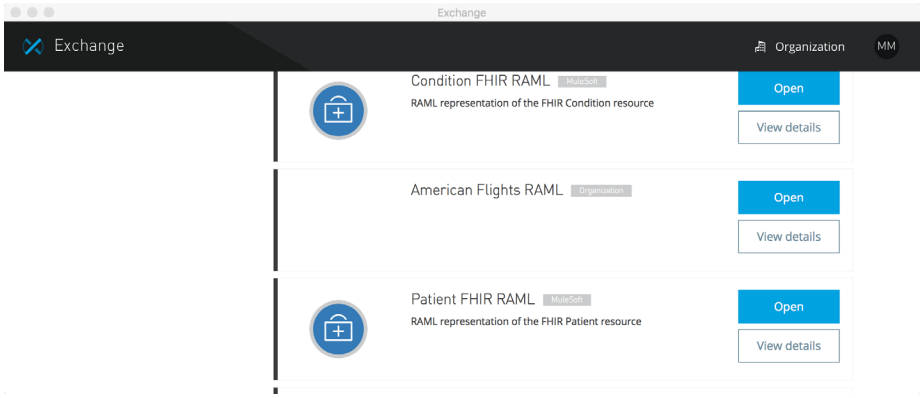
9. Drag out another HTTP connector and drop it in the process section of the new flow.
10. Change the endpoint display name to American REST Request.



Configure the HTTP Request connector

11. Return to global.xml.
12. In the Global Elements view, click Create.
13. In the Choose Global Type dialog box, select Connector Configuration > HTTP Request Configuration and click OK.

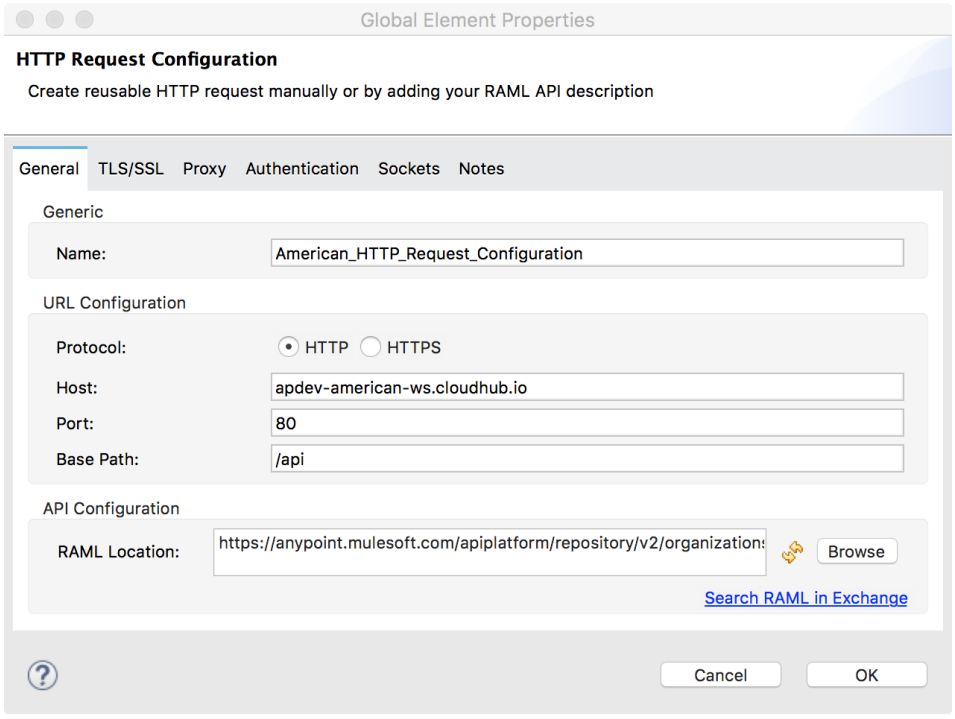
14. In the Global Element Properties dialog box, change the name to American_HTTP_Request_Configuration.
15. Click the Search RAML in Exchange link next to RAML location.
16. In the Exchange window that opens, scroll down and locate your American Flights RAML.



17. Click Open; the Exchange window should close.
18. Back in the Global Element Properties dialog box, wait for the RAML to be parsed and the host, port, and base path fields to be populated and then click OK.

    Note: If the fields did not populate, click the Reload RAML button next to the RAML location.

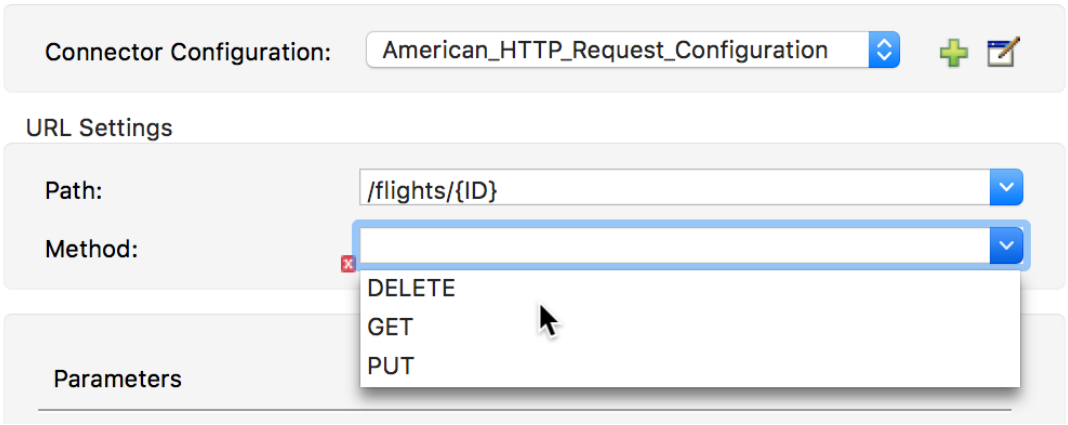
    Note: If you did not successfully create the RAML in the Exchange or did not successfully deploy the web service and API to the cloud, use the Solution - American Flights RAML URL located in the course snippets.txt file.



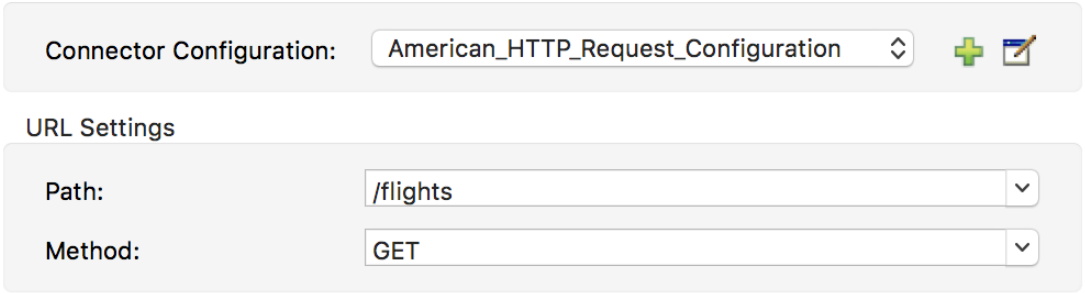
19. Click OK.

Configure the HTTP Request endpoint

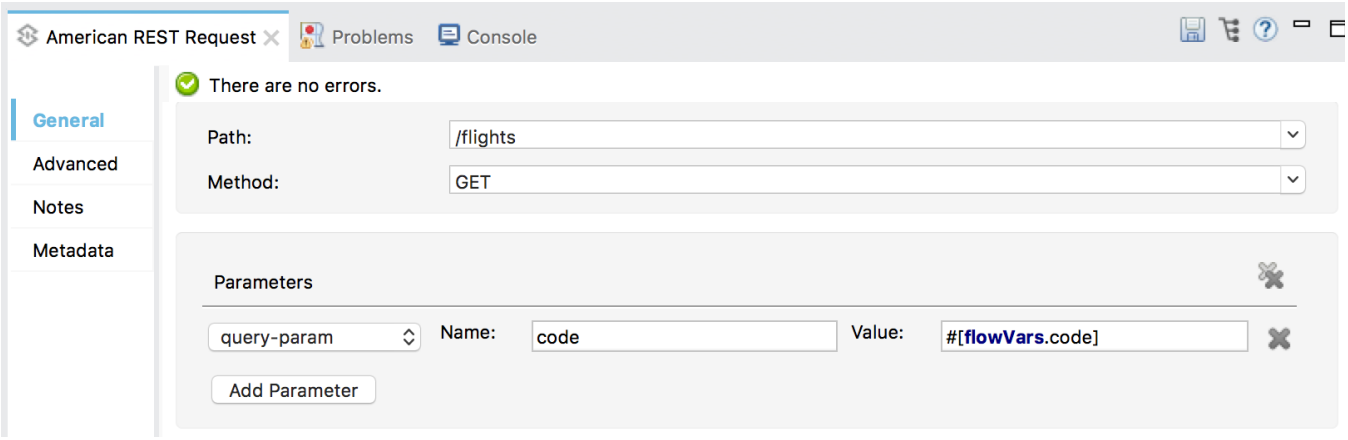
20. Return to implementation.xml.
21. In the American REST Request properties view, set the connector configuration to the existing American_HTTP_Request_Configuration.
22. Click the expand button for the path field; you should see all the available resources for the RESTful web service defined by the RAML file listed – in this case, there are two.
23. Select /flights/{ID}.
24. Look at the method drop-down menu; you should see DELETE, GET, and PUT.



25. Set the path to /flights.
26. Look at the method drop-down menu; you should see GET and POST.
27. Set the method to GET.



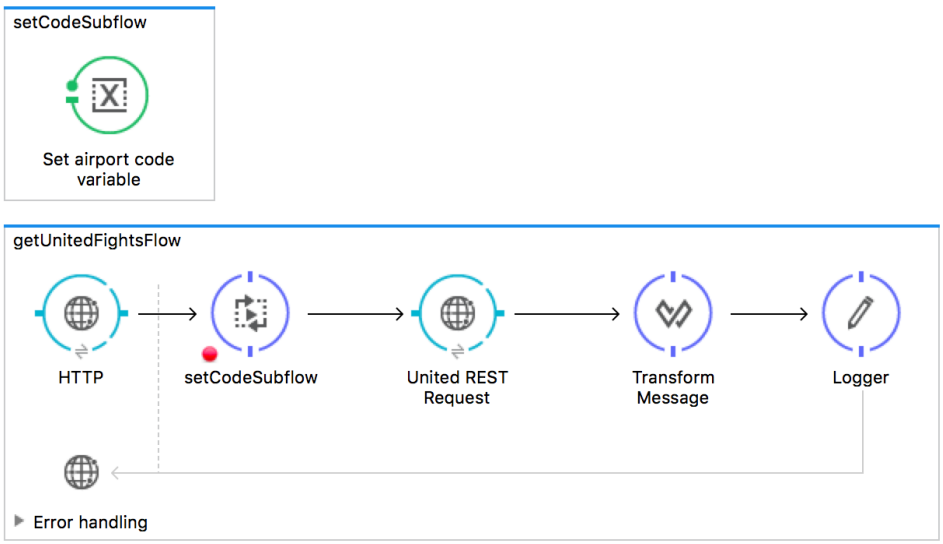
28. Scroll down and check to see if a query parameter called code has been added.
29. If the destination parameter was not automatically created, create it.
30. Set the value to a flow variable called code; you will add this to the flow next.



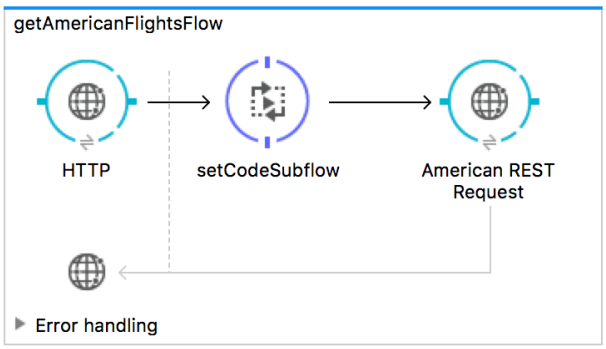
Extract the Set airport code variable processor into a subflow and use it

31. Right-click the Set airport code variable processor in getUnitedFlightsFlow and select Extract to > Sub Flow.
32. In the Extract Flow dialog box, set the flow name to setCodeSubflow and click OK.
33. Double-click the new Flow Reference in getUnitedFlightsFlow; the display name should update.
34. Locate the new subflow.

    Note: If you do not see the subflow in the canvas, switch to the Configuration XML view and then back to the Message Flow view.

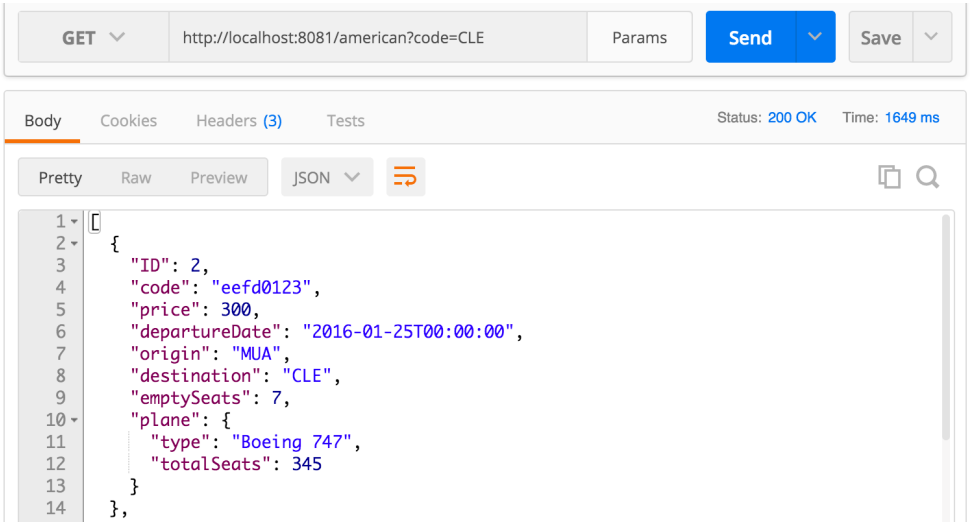


35. Drag a Flow Reference component from the Mule Palette before the American REST Request in getAmericanFlightsFlow.
36. In the Flow Reference properties view, set the flow name to setCodeSubflow.



Test the application

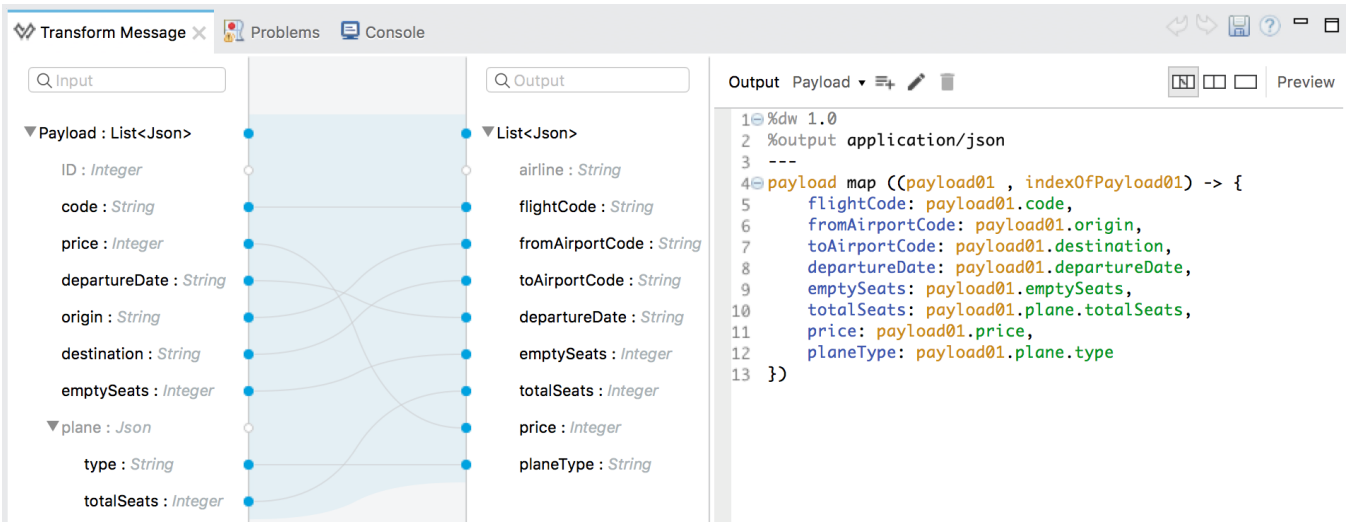
37. Save all the files to redeploy the application.
38. In Postman, return to the tab with the local requests.
39. Make a request to http://localhost:8081/american; you should get the American flights to SFO.
40. Add a query parameter called code with a value of CLE.
41. Send the request; you should get just the flights to CLE.



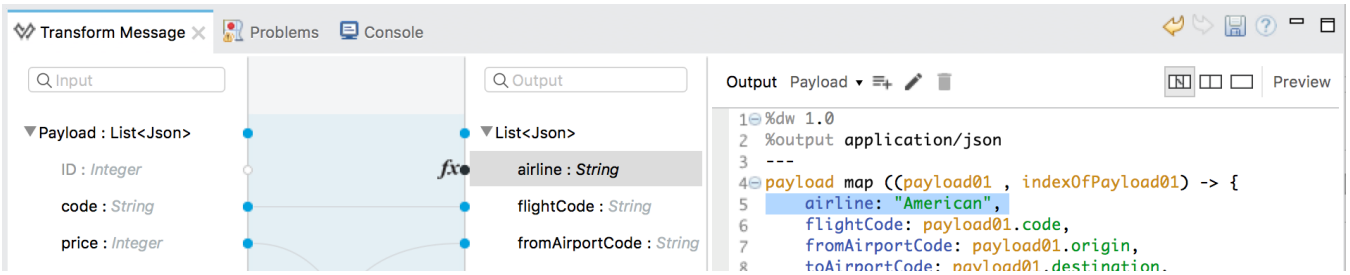
Transform the data

42. Return to getAmericanFlightsFlow.
43. Add a Transform Message component after the American REST Request endpoint.
44. Double-click the Transform Message component.
45. Look at the input section; 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 Define metadata type dialog box, select flights_json.
48. Click Select; you should now see output metadata in the output section of the Transform Message properties view.
49. Map fields (except ID and airline) by dragging them from the input section and dropping them on the corresponding field in the output section.



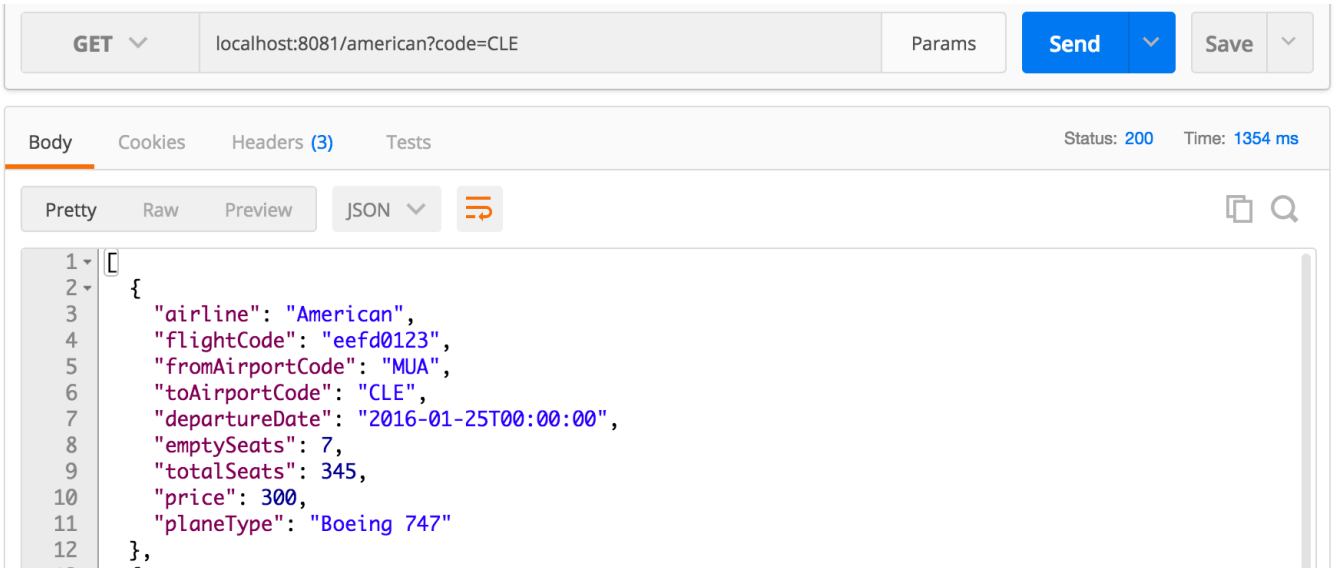
50. Double-click the airline field in the output section.
51. In the generated DataWeave expression, set airline to "American".



Test the application

52. Save to redeploy the project.

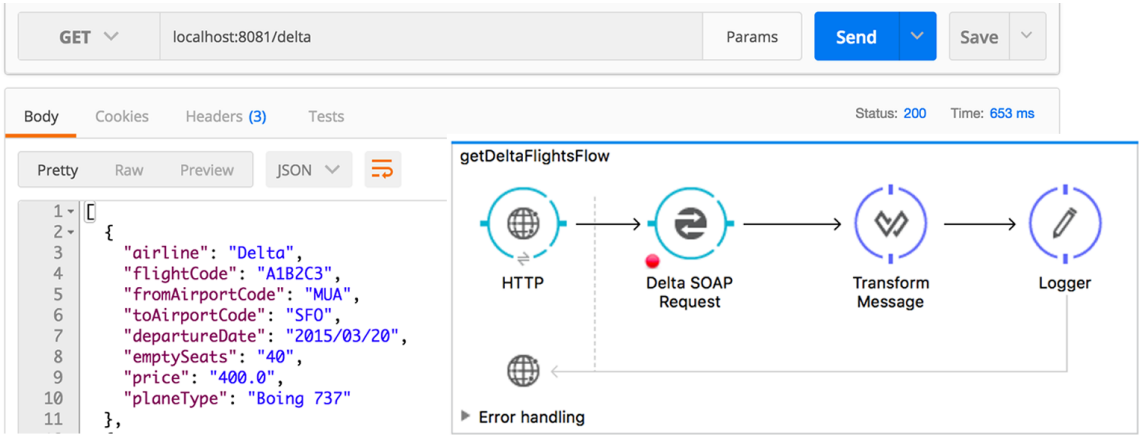
53. In Postman, make another request to http://localhost:8081/american; you should see all the flight data as JSON again but now with a different structure.



Walkthrough 7-4: Consume a SOAP web service

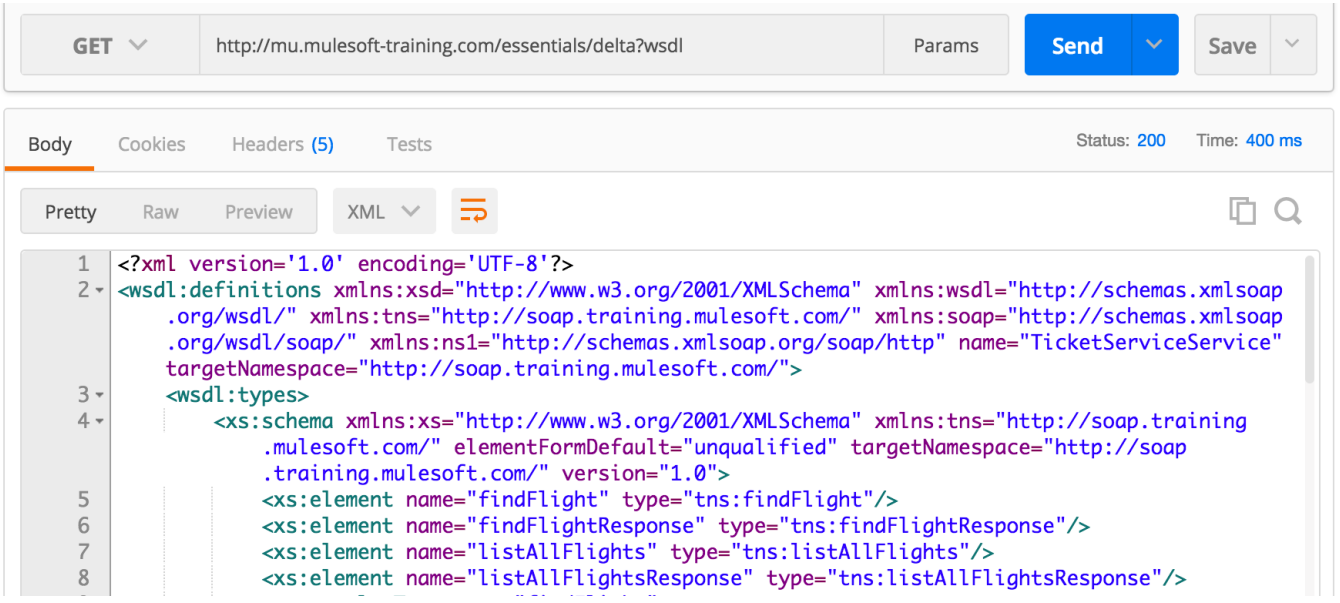
In this walkthrough, you consume a SOAP web service that returns a list of all Delta flights as XML. You will:

- Create a new flow to call a SOAP web service.
- Use a Web Service Consumer endpoint to consume a SOAP web service for Delta flight data.
- Use DataWeave to transform the XML response into JSON specified by the MUA Flights API.



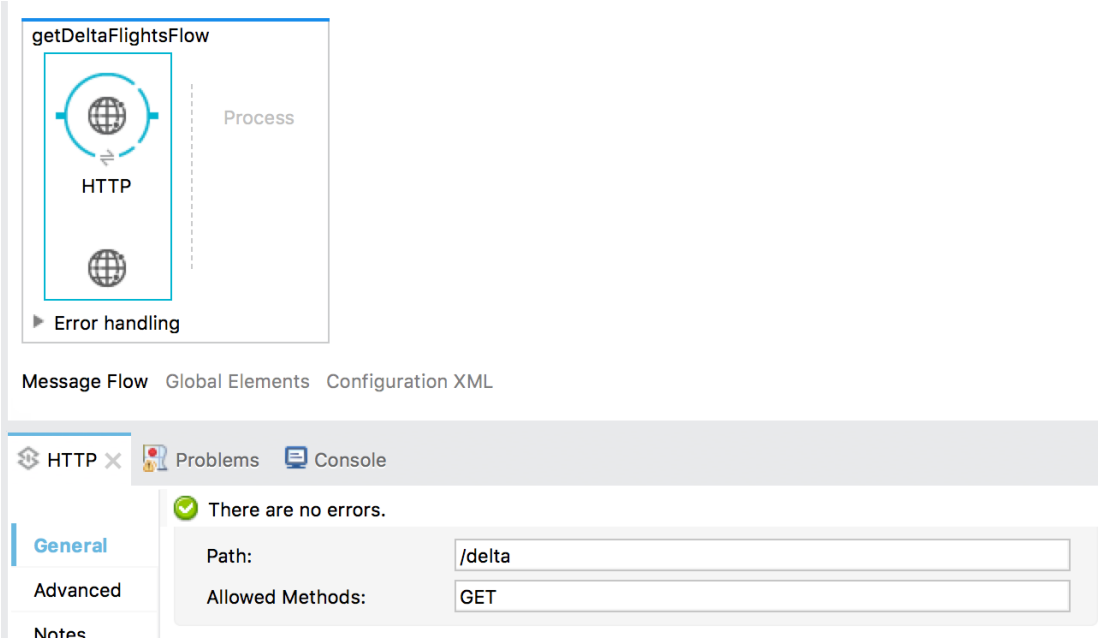
Browse the WSDL

1. Return to the course snippets.txt file and copy the WSDL URL for the Delta SOAP web service.
2. In Postman, return to the third tab, the one with the mulesoft-training request.
3. Paste the URL and send the request; you should see the web service WSDL returned.
4. Browse the WSDL; you should find references to operations listAllFlights and findFlight.



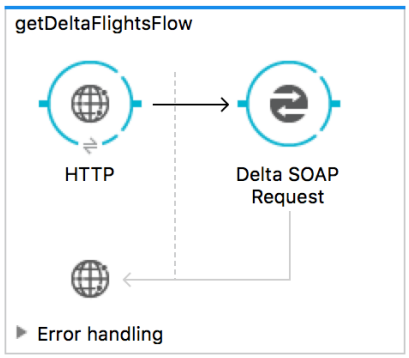
Create a new flow with an HTTP Listener connector endpoint

5. Return to Anypoint Studio.
6. Drag out another HTTP connector and drop it in the canvas after the existing flows.
7. Rename the flow to getDeltaFlightsFlow.
8. In the Properties view for the endpoint, set the connector configuration to the existing HTTP_Listener_Configuration.
9. Set the path to /delta.
10. Set the allowed methods to GET.



Add a Web Service Consumer connector endpoint

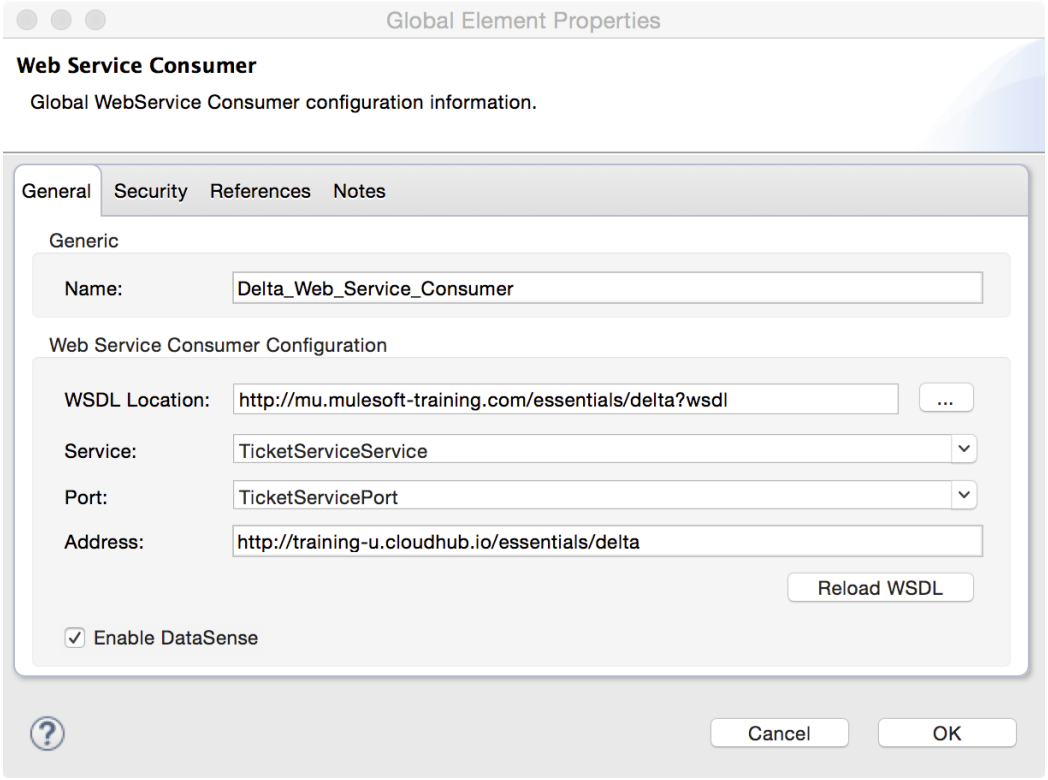
11. Drag out a Web Service Consumer connector and drop it in the process section of the flow.
12. Change its display name to Delta SOAP Request.



Configure the Web Service Consumer connector

13. Return to global.xml.
14. Click Create.
15. In the Choose Global Type dialog box, select Connector Configuration > Web Service Consumer and click OK.
16. In the Global Element Properties dialog box, change the name to Delta_Web_Service_Consumer.
17. Set the WSDL location to the value you copied from the course snippets.txt file.
18. Wait for the service, port, and address fields to populate.

    Note: If the fields do not populate, click the Reload WSDL button. If the fields still do not auto-populate, select TicketServiceService from the service drop-down menu and select TicketServicePort from the port drop-down menu.

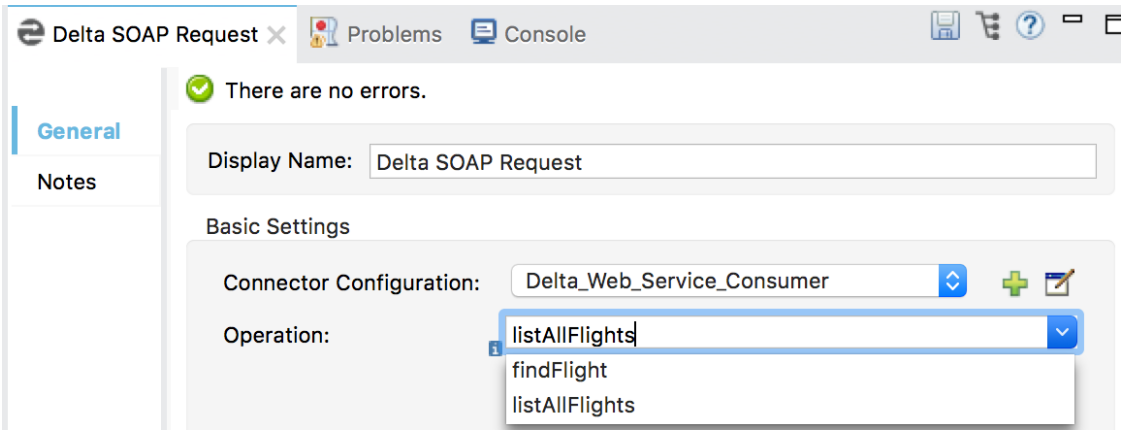


19. Click OK.

Configure the Web Service Consumer endpoint

20. Return to implementation.xml.

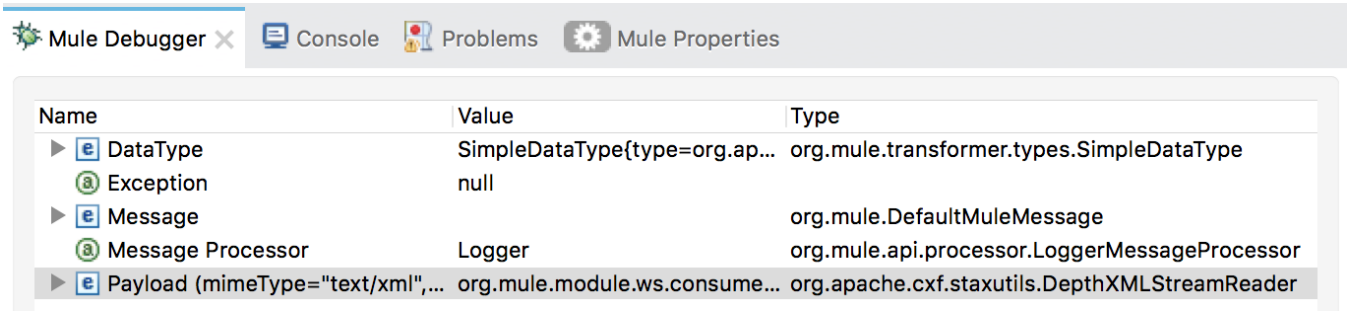
21. In the Delta SOAP Request properties view, set the connector configuration to the existing Delta_Web_Service_Consumer.
22. Click the operation drop-down menu button; you should see all of the web service operations listed.



23. Select the listAllFlights operation.

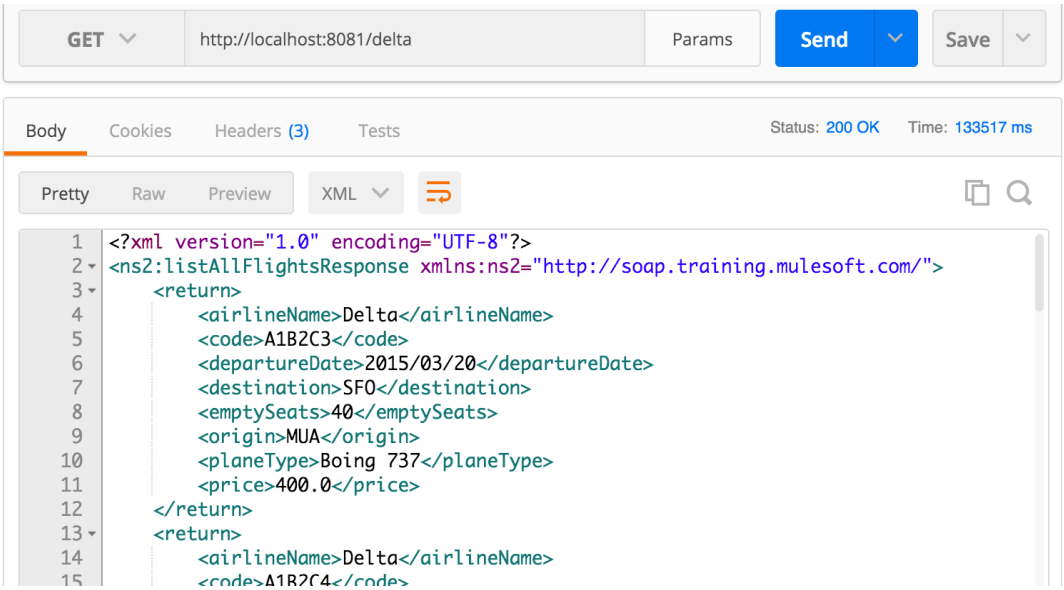
Test the application

24. Add a breakpoint to the Delta SOAP Request endpoint.
25. Add a Logger to the end of the flow.
26. Debug the project.
27. In Postman, return to the middle tab – the one with the localhost requests.
28. Make a request to http://localhost:8081/delta.
29. In the Mule Debugger, step to the Logger and look at the payload; it should be of type DepthXMLStreamReader.



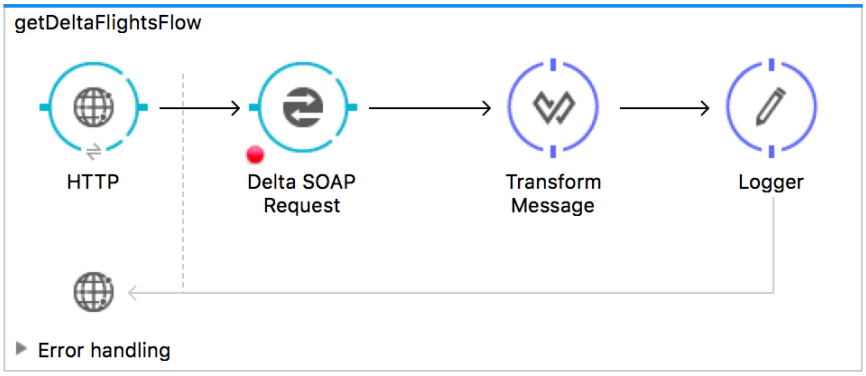
30. Step through the application.

31. Return to Postman; you should see the XML flight data returned.



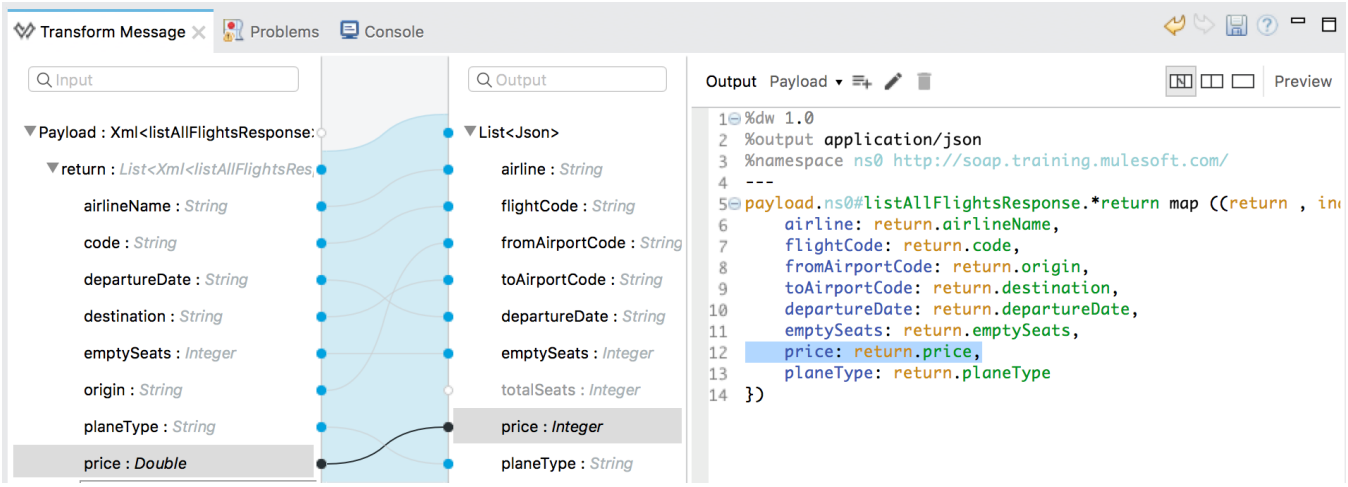
Transform the data

32. Return to Anypoint Studio and stop the project.
33. In getDeltaFlightsFlow, add a Transform Message component after the Delta SOAP Request endpoint.



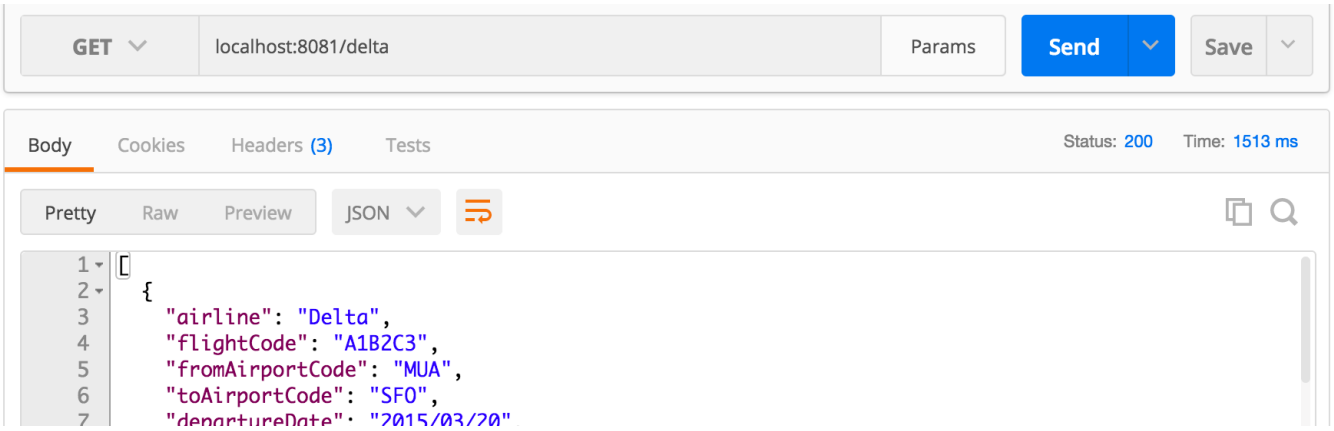
34. Look at the input section of the Transform Message properties view; you should see metadata already defined.
35. In the output section of the Transform Message properties view, click the Define metadata link.
36. In the Define metadata type dialog box, select flights_json.
37. Click Select; you should now see output metadata in the output section of the Transform Message properties view.

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



Test the application

39. Run the project.
40. In Postman, make another request to http://localhost:8081/delta; you should see all the flight data but now as JSON.

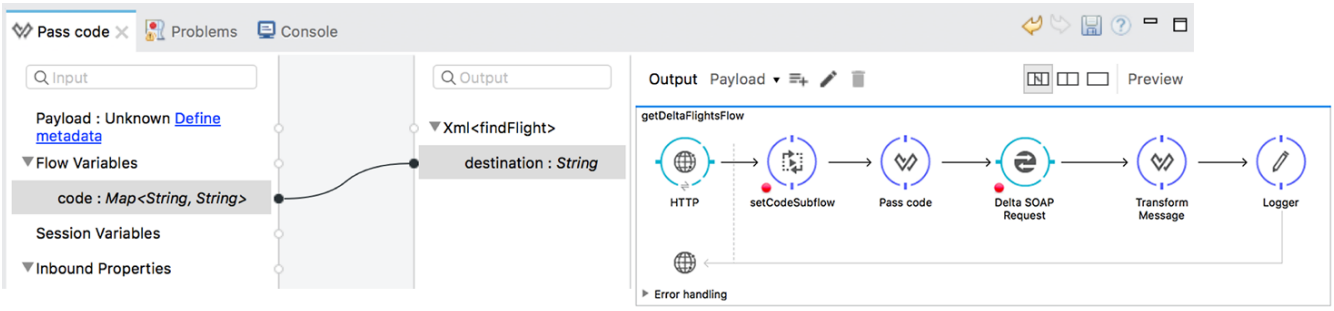


41. Add a query parameter called code and set it equal to LAX.
42. Send the request; you should still get all flights.

Walkthrough 7-5: Pass arguments to a SOAP web service using DataWeave

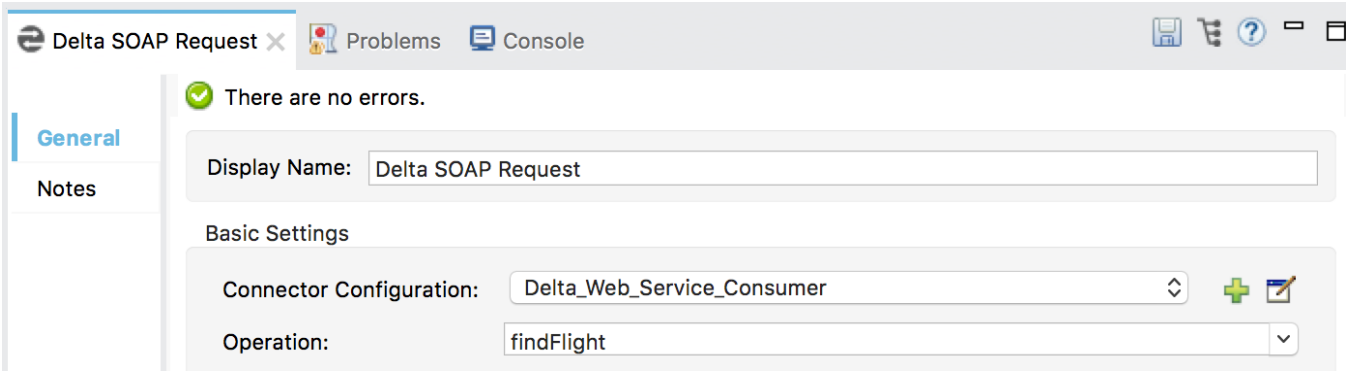
In this walkthrough, you modify the Delta flow to return the flights for a specific destination instead of all the flights. You will:

- Change the web service operation invoked to one that requires a destination as an input argument.
- Set a flow variable to the desired destination.
- Use DataWeave to pass the flow variable to the web service operation.



Call a different web service operation

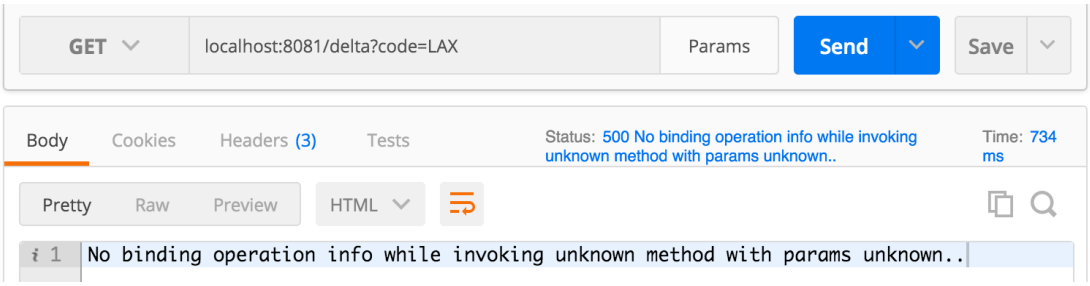
1. Return to getDeltaFlightsFlow.
2. In the Properties view for the Delta SOAP Request endpoint, change the operation to findFlight.



Test the application

3. Apply the changes to redeploy the application.

4. In Postman, send the same request with the query parameter; you should get a 500 response with a message about unknown parameters.

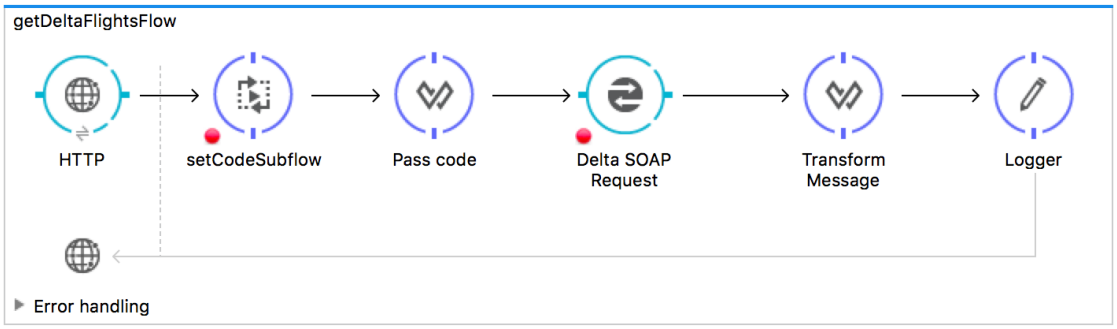


Use the set airport code subflow

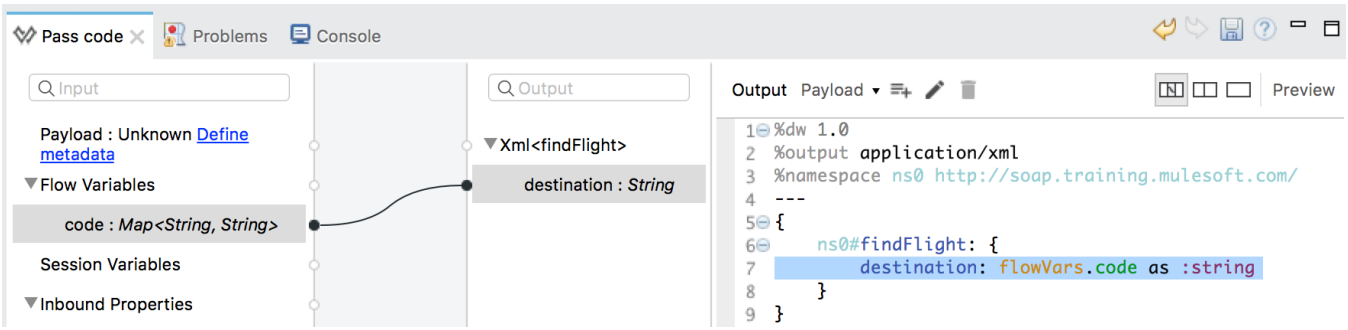
5. Return to getDeltaFlightsFlow.
6. Add a Flow Reference component before the Delta REST Request endpoint.
7. In the Flow Reference properties view, set the flow name to setCodeSubflow.

Use DataWeave to pass parameters to the web service

8. Add a Transform Message component to the left of the Delta SOAP Request endpoint.
9. Change its display name to Pass code.

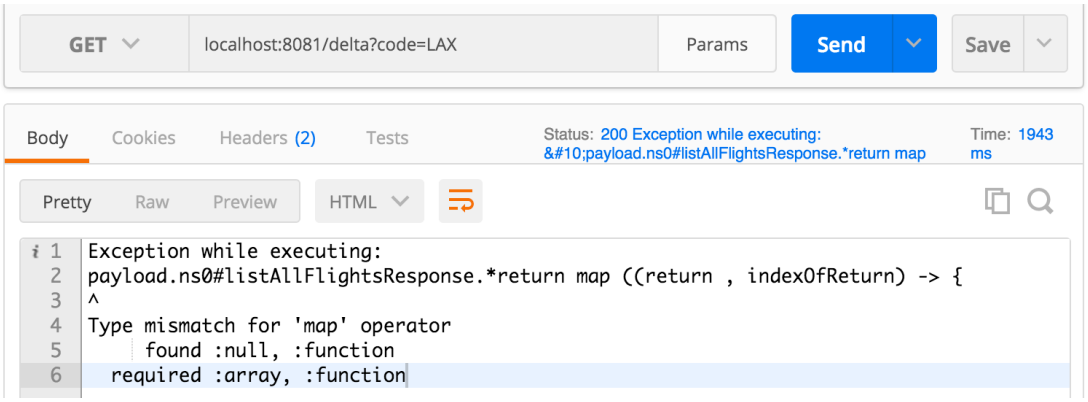


10. In the Pass code properties view, look at the input and output sections.
11. Drag the code flow variable in the input section to the destination element in the output section.



Test the application

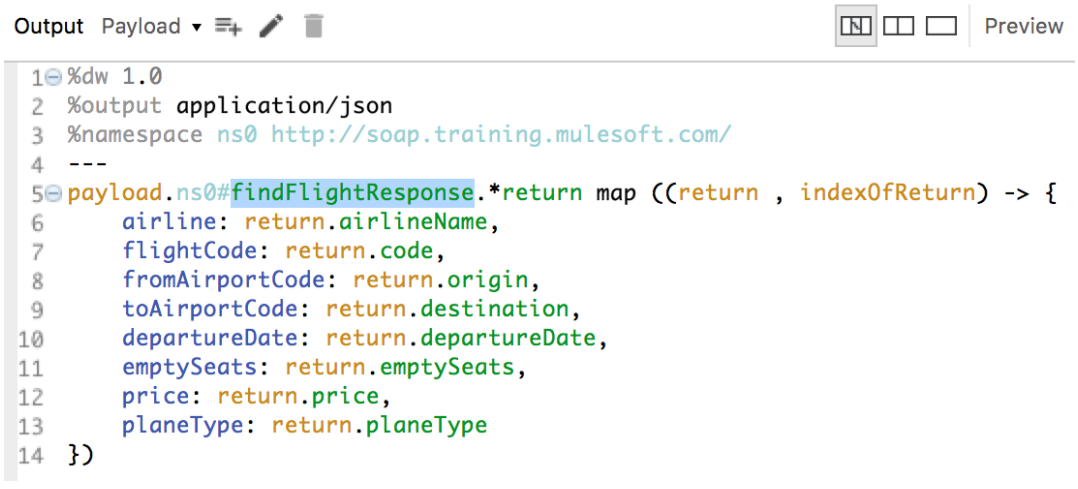
12. Redeploy the application.
13. In Postman, make the same request; you should get a 200 status code with an exception message.



14. Examine the exception message and figure out what is wrong with the transformation.

Modify the DataWeave expression

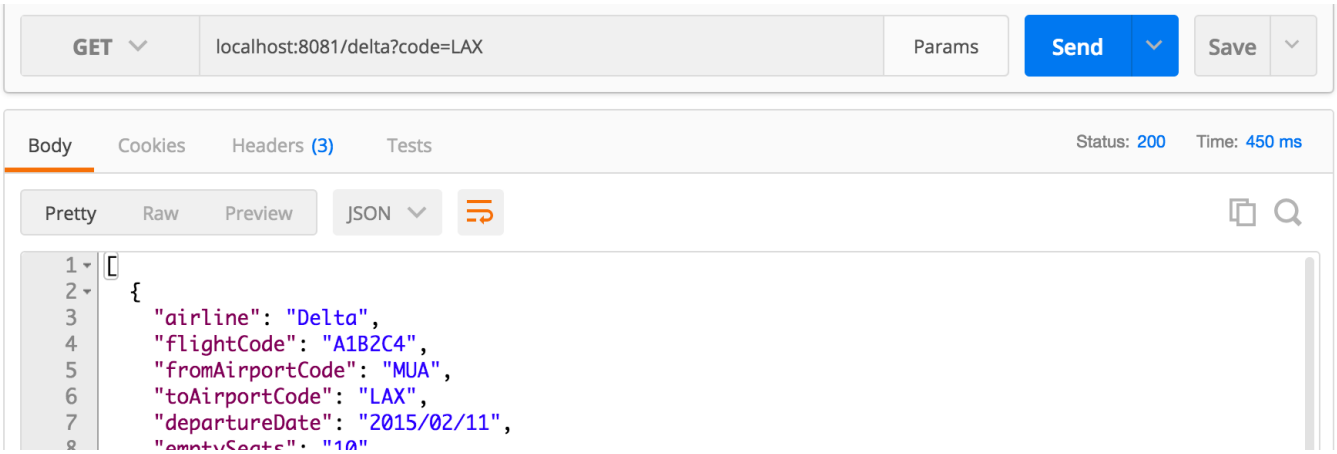
15. Return to getDeltaFlightsFlow.
16. Go to the Properties view for the Transform Message component after Delta SOAP Request.
17. Look at the transformation expression.
18. Change ns0#listAllFlightsResponse in the transformation code to ns0#findFlightResponse.



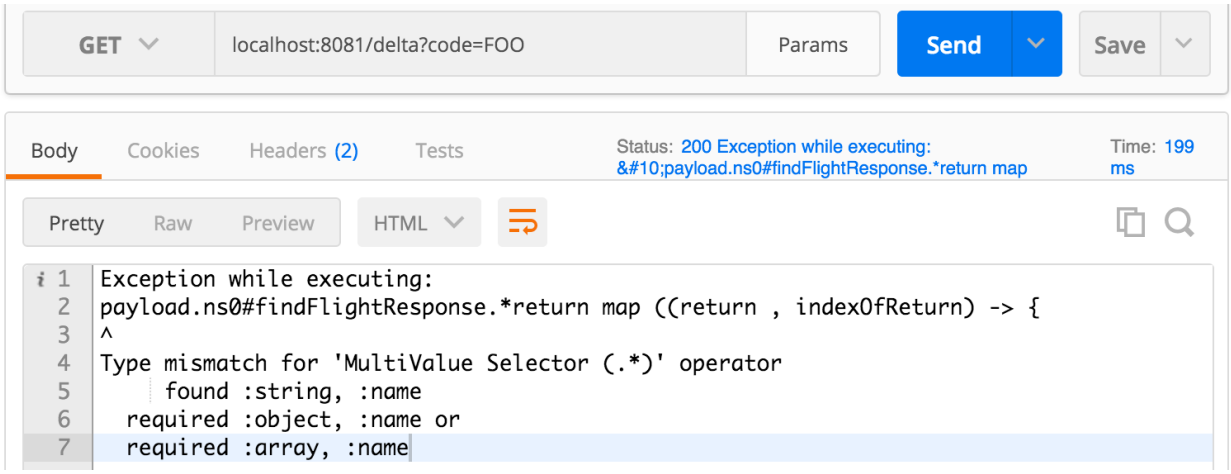
Test the application

19. Redeploy the application.

20. In Postman, make the same request; you should now get all the flights to LAX.



21. Change the code query parameter to have a value of CLE.
22. Send the request; you should now get only flights to CLE.
23. Change the code query parameter to have a value of FOO.
24. Send the request; you should get a 200 status code and a message with an exception.



25. Return to Anypoint Studio and stop the project.