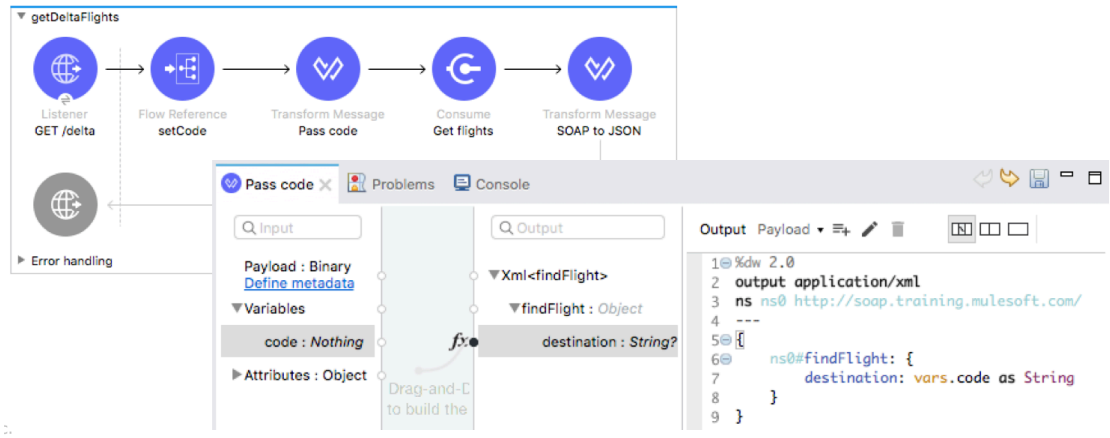
Walkthrough 8-3: Consume a SOAP web service

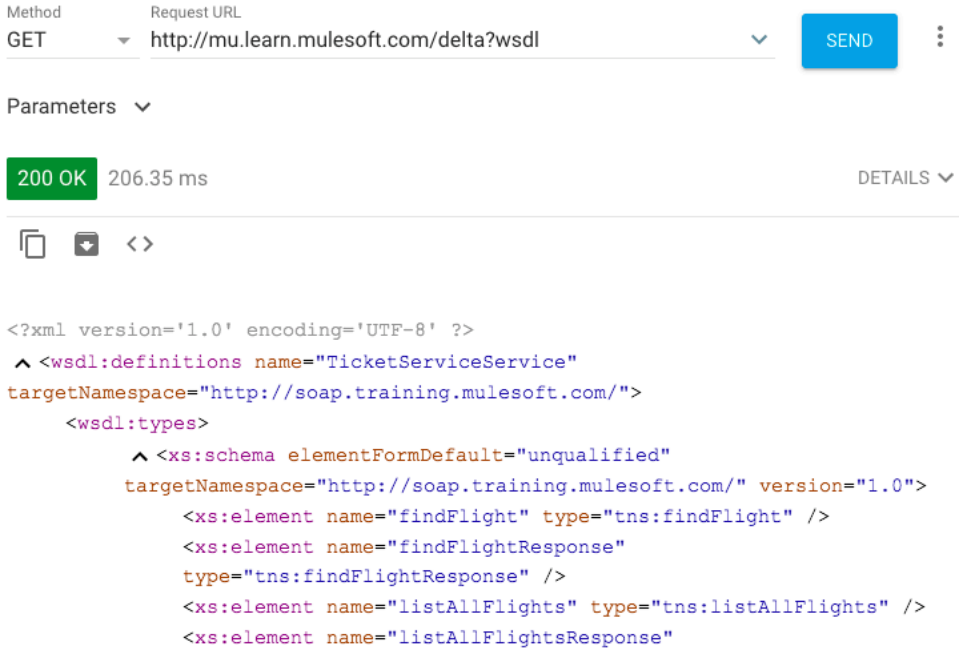
In this walkthrough, you consume a Delta SOAP web service. You will:

- Create a new flow to call the Delta SOAP web service.
- Use a Web Service Consumer connector to consume a SOAP web service.
- Use the Transform Message component to pass arguments to a SOAP web service.



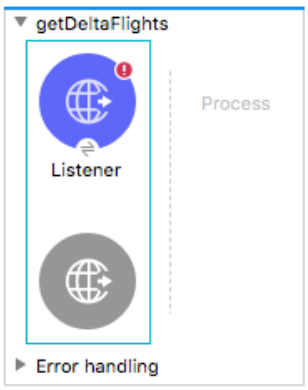
Browse the WSDL

1. Return to the course snippets.txt file and copy the Delta SOAP web service WSDL.
2. In Advanced REST Client, return to the third tab, the one with the learn.mulesoft 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.

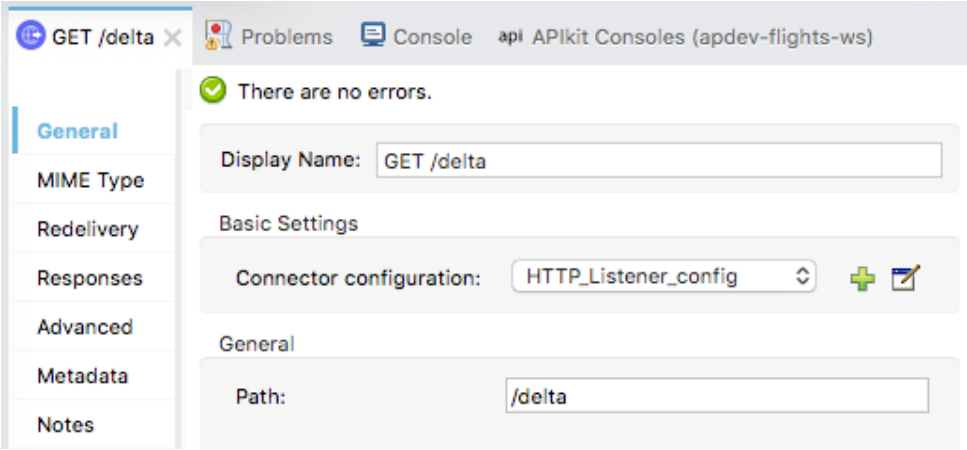


Create a new flow with an HTTP Listener connector endpoint

5. Return to Anypoint Studio.
6. Drag out another HTTP Listener from the Mule Palette and drop it in the canvas after the existing flows.
7. Rename the flow to getDeltaFlights.



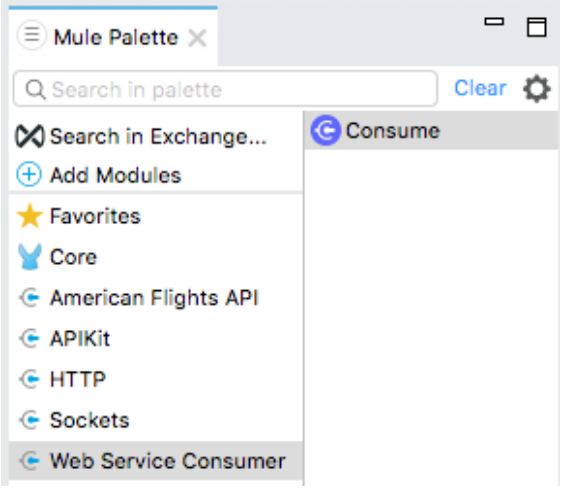
8. In the Listener properties view, set the display name to GET /delta.
9. Set the connector configuration to the existing HTTP_Listener_config.
10. Set the path to /delta and the allowed methods to GET.



Add the Web Service Consumer module to the project

11. In the Mule Palette, select Add Modules.

12. Select the Web Service Consumer connector in the right side of the Mule Palette and drag and drop it into the left side.
13. If you get a Select module version dialog box, select the latest version and click Add.



Configure the Web Service Consumer connector

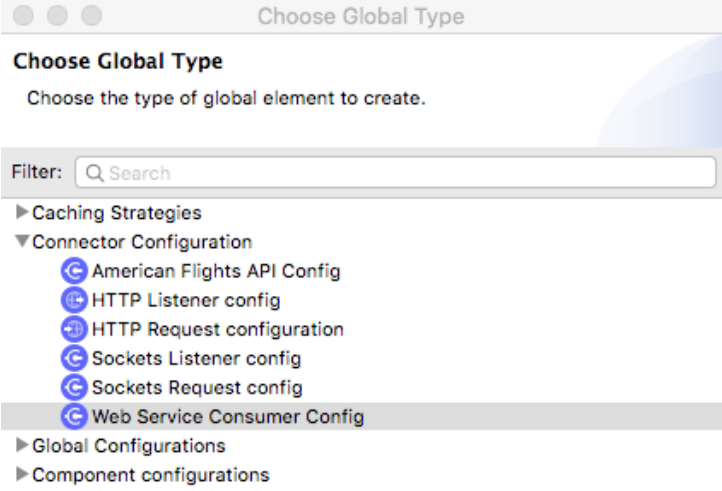
14. Return to the course snippets.txt file and copy the text for the Delta web service properties.
15. Return to config.yaml in src/main/resources and paste the code at the end of the file.

*implementation | *config.yaml | global

```yaml
http:
  port: "8081"

american:
  host: "training4-american-api-{lastname}.cloudhub.io"
  port: "80"
  basepath: "/"
  protocol: "HTTP"
  client_id: "your_client_id"
  client_secret: "your_client_secret"

training:
  host: "mu.learn.mulesoft.com"
  port: "80"
  basepath: "/"
  protocol: "HTTP"

delta:
  wsdl: "http://mu.learn.mulesoft.com/delta?wsdl"
  service: "TicketServiceService"
  port: "TicketServicePort"
```

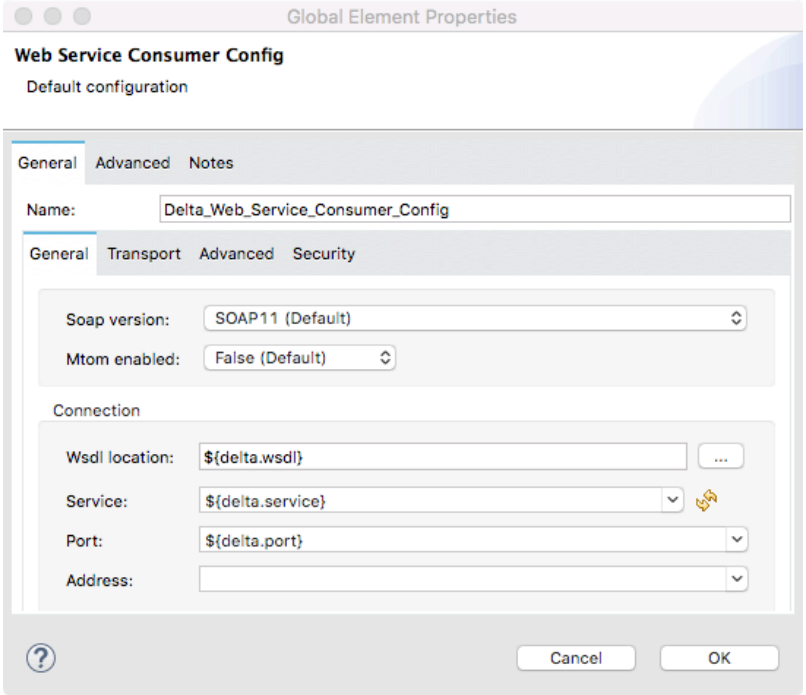
16. Save the file.
17. Return to global.xml.
18. Click Create.

19. In the Choose Global Type dialog box, select Connector Configuration > Web Service Consumer Config and click OK.



20. In the Global Element Properties dialog box, change the name to Delta_Web_Service_Consumer_Config.
21. Set the

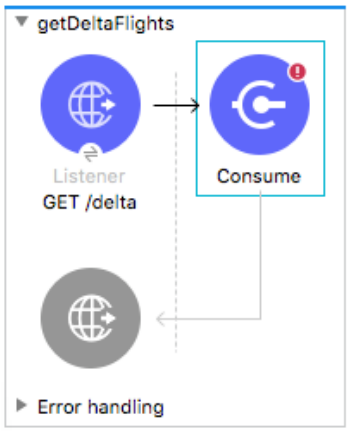
- Wsdl location: ${delta.wsdl}
- Service: ${delta.service}
- Port: ${delta.port}



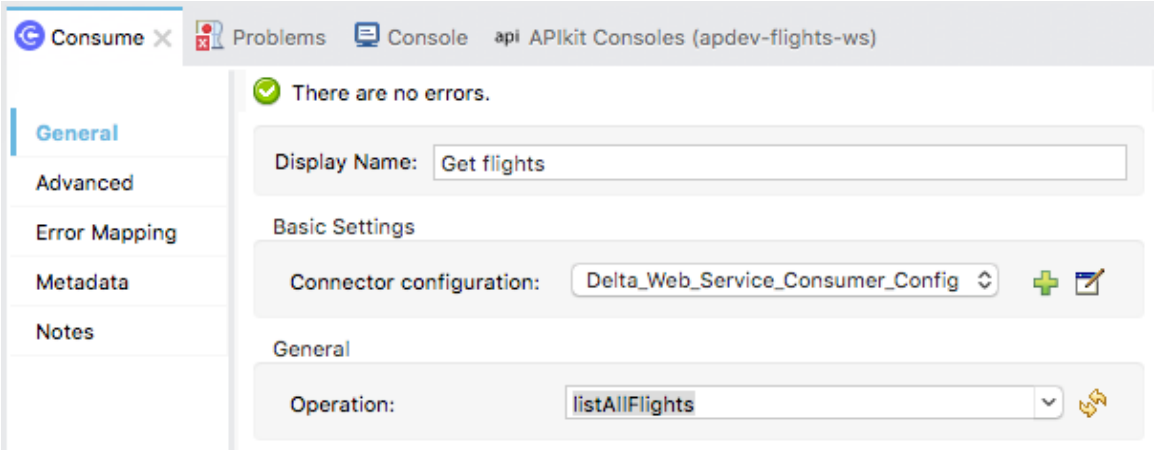
22. Click OK.

Add and configure a Consume operation

23. Return to implementation.xml.
24. Locate the Consume operation for the Web Service Consumer connector in the Mule Palette and drag and drop it in the process section of getDeltaFilghts.

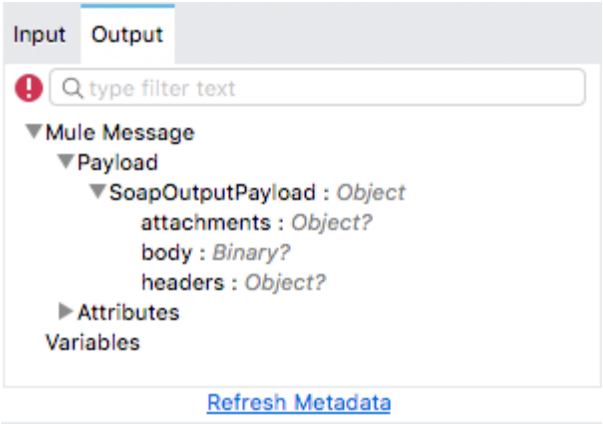


25. In the Consume properties view, change its display name to Get flights.
26. Set the connector configuration to the existing Delta_Web_Service_Consumer_Config.
27. Click the operation drop-down menu button; you should see all of the web service operations listed.
28. Select the listAllFlights operation.

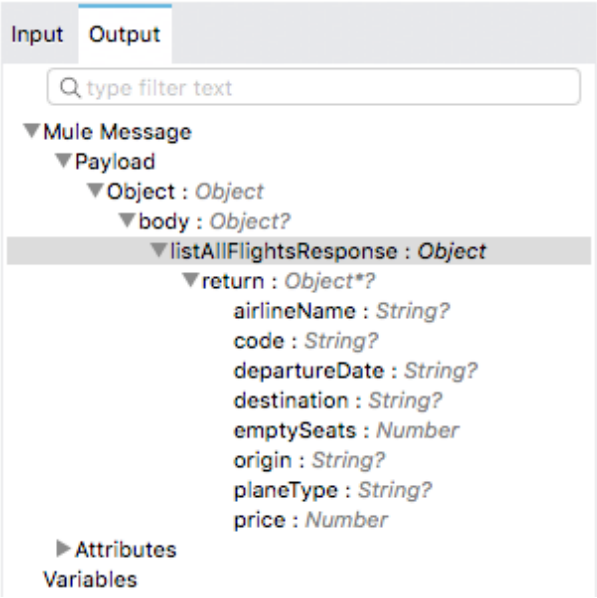


Review metadata associated with the United Get flights operation response

29. Select the Output tab in the DataSense Explorer and expand Payload; you should see payload metadata but with no detailed structure.



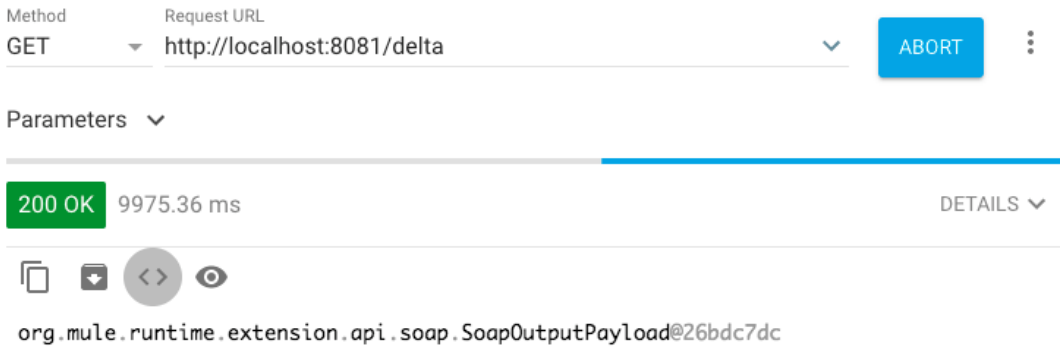
30. Save the file.
31. Select the Output tab in the DataSense Explorer and expand Payload again; you should now see the payload structure.



Test the application

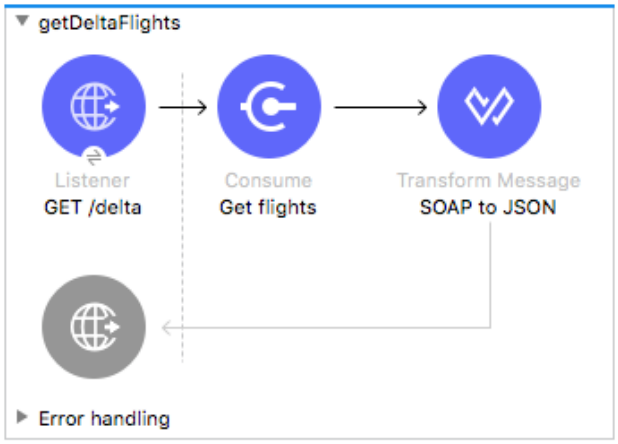
32. Save the files to redeploy the project.
33. In Advanced REST Client, return to the middle tab – the one with the localhost requests.

34. Make a request to http://localhost:8081/delta; you should get a response that is object of type SoapOutputPayload.

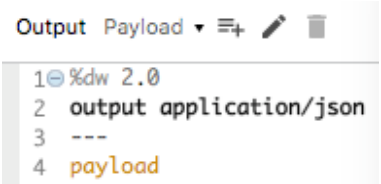


Transform the response to JSON

35. Return to Anypoint Studio.
36. Add a Transform Message component to the end of the flow.
37. Set the display name to SOAP to JSON.



38. In the expression section of the Transform Message properties view, change the output type to json and the output to payload.



Test the application

39. Save the file to redeploy the project.

40. In Advanced REST Client, make another request to http://localhost:8081/delta; you should get JSON returned.

200 OK 591.36 ms


```
{
  -"body": {
     -"listAllFlightsResponse": {
        -"return": {
            "airlineName": "Delta",
            "code": "A1B3D4",
            "departureDate": "2015/02/12",
            "destination": "PDX",
            "emptySeats": "10",
            "origin": "MUA",
            "planeType": "Boing 777",
            "price": "385.0"
         }
      }
    },
    "attachments": {},
    "headers": {}
}
```


41. Click the Toggle raw response view button; you should see all the flights.

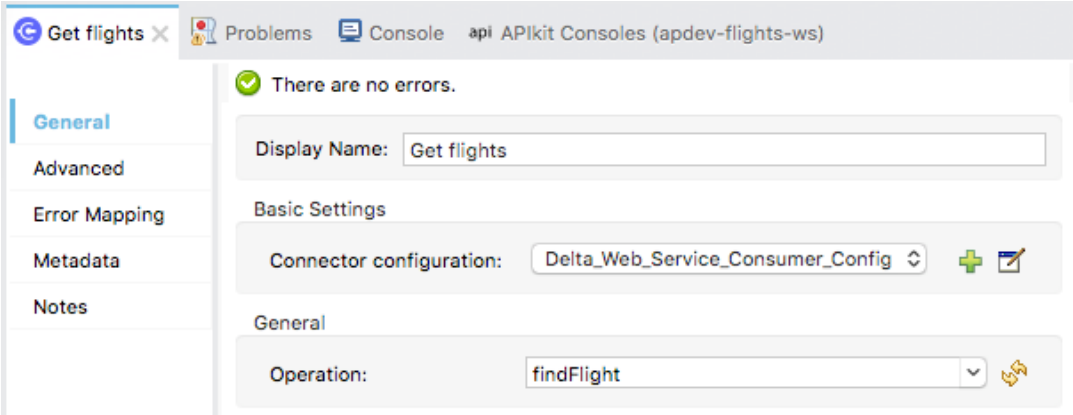
200 OK 3936.91 ms


```
{
  "body": {
    "listAllFlightsResponse": {
      "return": {
        "airlineName": "Delta",
        "code": "A1B2C3",
        "departureDate": "2015/03/20",
        "destination": "SFO",
        "emptySeats": "40",
        "origin": "MUA",
        "planeType": "Boing 737",
        "price": "400.0"
      },
      "return": {
        "airlineName": "Delta",
        "code": "A1B2C4",
        "departureDate": "2015/02/11",
        "destination": "LAX",
```


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

Call a different web service operation

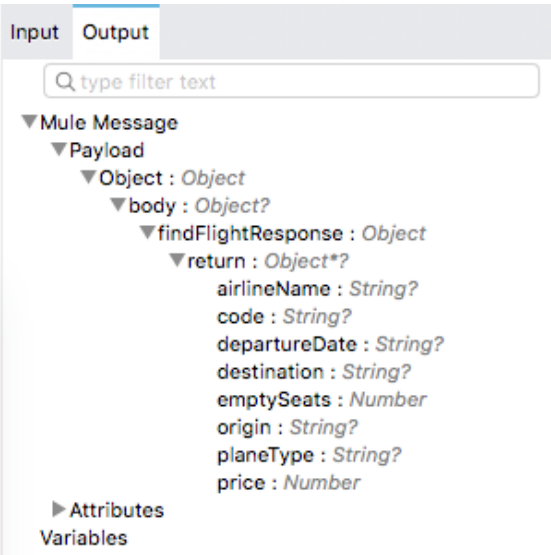
44. Return to getDeltaFlights in Anypoint Studio.
45. In the properties view for Get flights Consume operation, change the operation to findFlight.



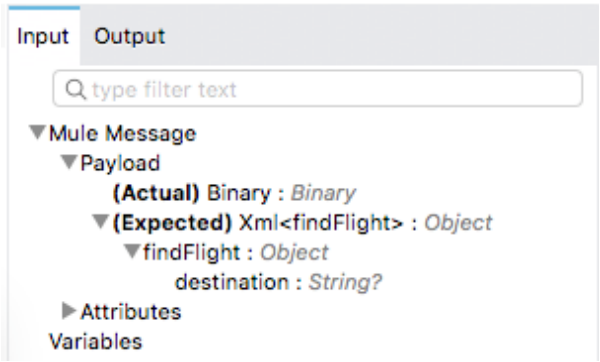
46. Save all files to redeploy the application.

Review metadata associated with the United Get flights operation response

47. Return to the Get flights properties view.
48. Select the Output tab in the DataSense Explorer and expand Payload; you should now see different payload metadata.

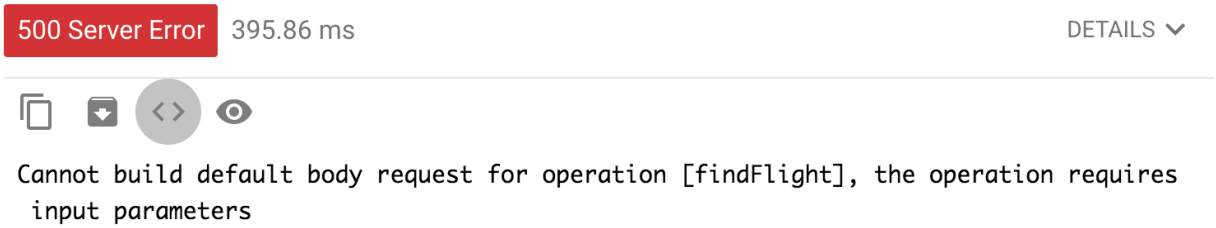


49. Select the Input tab and expand Payload; you should see this operation now expects a destination.



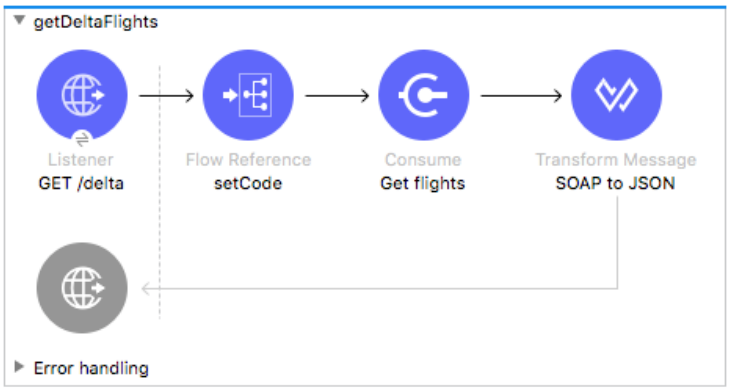
Test the application

50. In Advanced REST Client, send the same request with the query parameter; you should get a 500 Server Error with a message that the operation requires input parameters.



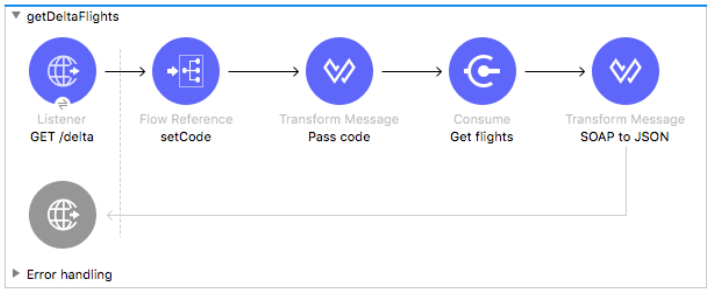
Use the set airport code subflow

51. Return to getDeltaFlights in Anypoint Studio.
52. Add a Flow Reference component after the GET /delta Listener.
53. In the Flow Reference properties view, set the flow name to setCode.

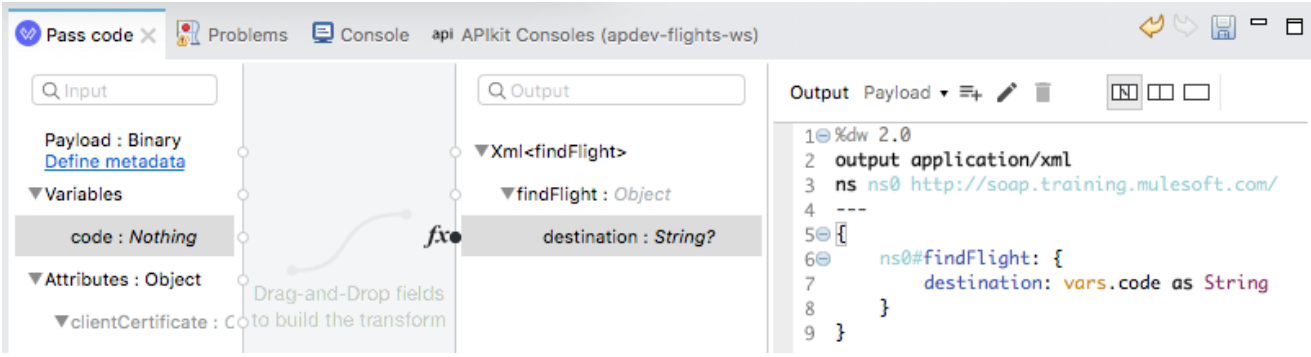


Use the Transform Message component to pass a parameter to the web service

54. Add a Transform Message component after the Flow Reference component.
55. Change its display name to Pass Code.

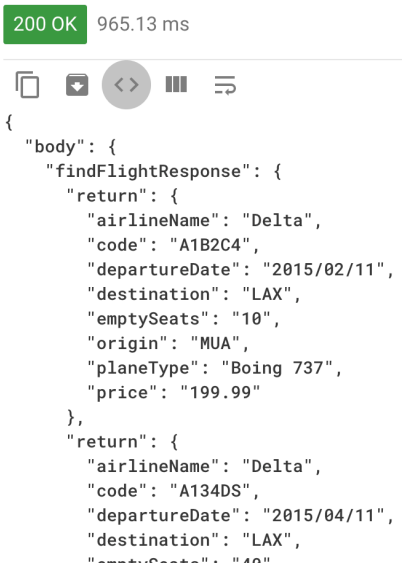


56. In the Pass code properties view, look at the input and output sections.
57. Drag the code variable in the input section to the destination element in the output section.



Test the application

58. Save the file to redeploy the application.
59. In Advanced REST Client, make another request.; you should now only see flights to LAX.




```
{
  "body": {
    "findFlightResponse": {
      "return": {
        "airlineName": "Delta",
        "code": "A1B2C4",
        "departureDate": "2015/02/11",
        "destination": "LAX",
        "emptySeats": "10",
        "origin": "MUA",
        "planeType": "Boing 737",
        "price": "199.99"
      },
      "return": {
        "airlineName": "Delta",
        "code": "A134DS",
        "departureDate": "2015/04/11",
        "destination": "LAX",
```