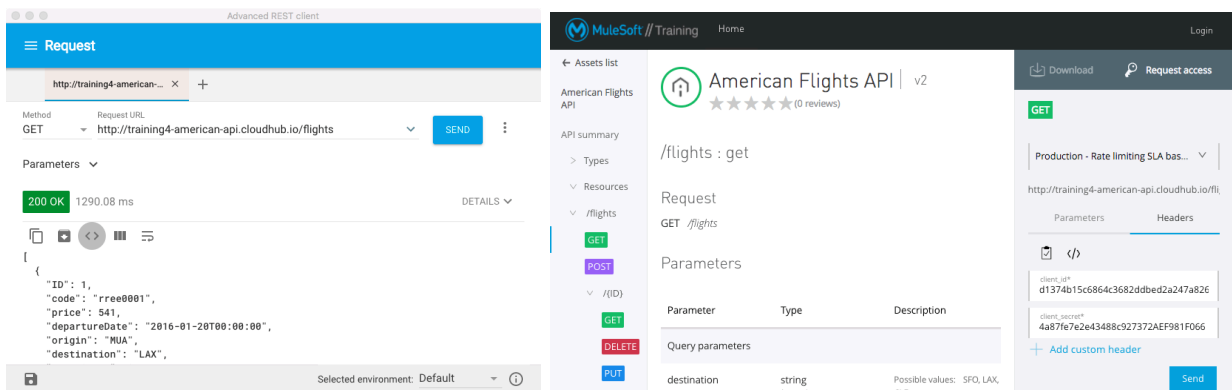


Walkthrough 1-2: Make calls to an API

In this walkthrough, you make calls to a RESTful API. You will:

- Use Advanced REST Client to make calls to an unsecured API (an implementation).
- Make GET, DELETE, POST, and PUT calls.
- Use Advanced REST Client to make calls to a secured API (an API proxy).
- Use the API console in an API portal to make calls to a managed API using a mocking service.
- Use the API console to make calls to an API proxy endpoint.



Use Advanced REST Client to make GET requests to retrieve data

1. Return to or open Advanced REST Client.
2. Make sure the method is set to GET.
3. Return to the course snippets.txt file.
4. Copy the URL for the American Flights web service:
<http://training4-american-ws.cloudhub.io/api/flights>.





Note: This is the URL for the API implementation, not the managed API proxy. The -ws stands for web service.

5. Return to Advanced REST Client and paste the URL in the text box that says Request URL.



6. Click the Send button; you should get a response.
7. Locate the return HTTP status code of 200.
8. Review the response body containing flights to SFO, LAX, and CLE.



200 OK 1283.27 ms DETAILS ▾

```
[Array[11]
  -0: {
    "ID": 1,
    "code": "rree0001",
    "price": 541,
    "departureDate": "2016-01-20T00:00:00",
    "origin": "MUA",
    "destination": "LAX",
    "emptySeats": 0,
    -"plane": {
      "type": "Boeing 787",
      "totalSeats": 200
    }
  },
  -1: {
    "ID": 2,
    "code": "eefd0123",
```

9. Click the Toggle raw response view button.

200 OK 1283.27 ms DETAILS ▾

```
{
  {
    "ID": 1,
    "code": "rree0001",
    "price": 541,
    "departureDate": "2016-01-20T00:00:00",
    "origin": "MUA",
    "destination": "LAX",
    "emptySeats": 0,
    "plane": {
      "type": "Boeing 787",
      "totalSeats": 200
    }
  },
}
```

10. Click the arrow to the right of the URL.
11. In the area that appears, set the Param name to destination and the value to CLE.

The screenshot shows a REST client interface. At the top, the Method is set to GET and the Host is http://training-american-ws.cloudhub.io. Below this, the Path is /api/flights. In the Query parameters section, a parameter named 'destination' with the value 'CLE' has been added. A blue 'SEND' button is visible on the right. Below the query parameters, there is an 'ADD' button and an 'X' icon to delete the parameter.

12. Click the Send button; you should get just flights to CLE returned.
13. Click the X next to the parameter to delete it.
14. Click the arrow to the right of the URL to collapse the parameters section.
15. Change the request URL to use a uri parameter to retrieve the flight with an ID of 3:
<http://training4-american-ws.cloudhub.io/api/flights/3>
16. Click the Send button; you should see only the flight with that ID returned.

The screenshot shows the response of the REST client. At the top, it displays '200 OK' in a green box and '1235.94 ms' in a grey box. Below this, there are icons for copying, downloading, and toggling the view. The response body is a JSON array with one object. The object contains the following fields: ID (3), code (ffee0192), price (300), departureDate (2016-01-20T00:00:00), origin (MUA), destination (LAX), emptySeats (0), and plane (Boeing 777 with 300 total seats).

```
[Array(1)]
-0: {
  "ID": 3,
  "code": "ffee0192",
  "price": 300,
  "departureDate": "2016-01-20T00:00:00",
  "origin": "MUA",
  "destination": "LAX",
  "emptySeats": 0,
  "plane": {
    "type": "Boeing 777",
    "totalSeats": 300
  }
}
```

Make DELETE requests to delete data

17. Change the method to DELETE.

18. Click the Send button; you should see a 200 response with a message that the Flight was deleted (but not really).

Note: The database is not actually modified so that its data integrity can be retained for class.

The screenshot shows a REST client interface. At the top, the 'Method' is set to 'DELETE' and the 'Request URL' is 'http://training-american-ws.cloudhub.io/api/flights/3'. A blue 'SEND' button is visible. Below this, the 'Parameters' section is collapsed. The response status is '200 OK' in a green box, with a response time of '253.51 ms'. A 'DETAILS' link is on the right. The response body is a JSON object:

```
{  "message": "Flight deleted (but not really)"}
```

19. Remove the URI parameter from the request: <http://training4-american-ws.cloudhub.io/api/flights>.
20. Click the Send button; you should get a 405 response with a message of method not allowed.

The screenshot shows the REST client interface after sending a request without a URI parameter. The response status is '405 Method Not Allowed' in an orange box, with a response time of '200.86 ms'. A 'DETAILS' link is on the right. The response body is a JSON object:

```
{  "message": "Method not allowed"}
```

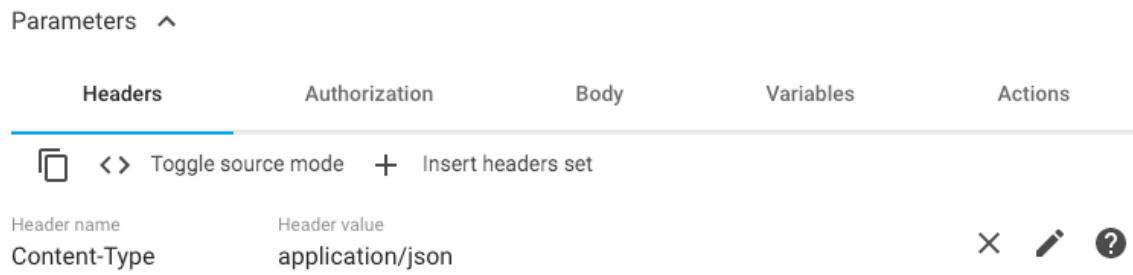
Make a POST request to add data

21. Change the method to POST.
22. Click the Send button; you should get a 415 response with a message of unsupported media type.

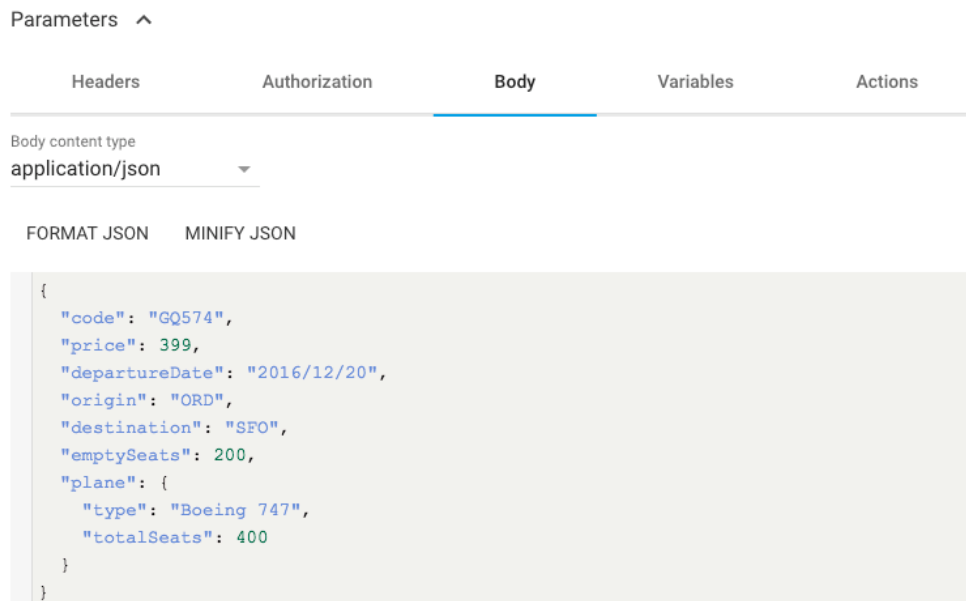
The screenshot shows the REST client interface with the 'Method' changed to 'POST'. The 'Request URL' remains 'http://training-american-ws.cloudhub.io/api/flights'. A blue 'SEND' button is visible. Below this, the 'Parameters' section is collapsed. The response status is '415 Unsupported Media Type' in an orange box, with a response time of '261.25 ms'. A 'DETAILS' link is on the right. The response body is a JSON object:

```
{  "message": "Unsupported media type"}
```

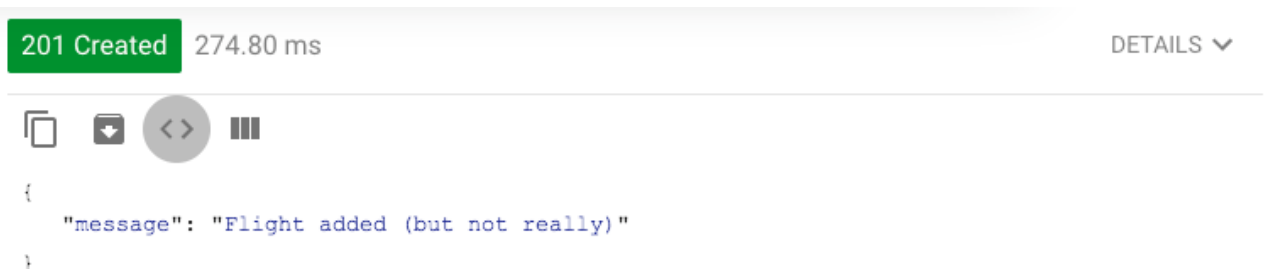
23. Click the arrow next to Parameters beneath the request URL.
24. Click in the Header name field, type C, and then select Content-Type.
25. In the Header value field, select application/json.



26. Select the Body tab.
27. Return to the course snippets.txt file and copy the value for American Flights API post body.
28. Return to Advanced REST Client and paste the code in the body text area.



29. Click the Send button; you should see a 201 Created response with the message Flight added (but not really).



30. Return to the request body and remove the plane field and value from the request body.
31. Remove the comma after the emptySeats key/value pair.

Headers

Authorization

Body

Variables

Actions

Body content type
application/json

FORMAT JSONMINIFY JSON

```
{
  "code": "GQ574",
  "price": 399,
  "departureDate": "2016/12/20",
  "origin": "ORD",
  "destination": "SFO",
  "emptySeats": 200
}
```

32. Send the request; the message should still post successfully.
33. In the request body, remove the emptySeats key/value pair.
34. Delete the comma after the destination key/value pair.

Headers

Authorization

Body

Variables

Actions

Body content type
application/json

FORMAT JSONMINIFY JSON

```
{
  "code": "GQ574",
  "price": 399,
  "departureDate": "2016/12/20",
  "origin": "ORD",
  "destination": "SFO"
}
```

35. Send the request; you should see a 400 Bad Request response with the message Bad request.

400 Bad Request291.33 msDETAILS

<>





|||

```
{
  "message": "Bad request"
}
```

Make a PUT request to update data

36. Change the method to PUT.
37. Add a flight ID of 3 to the URL.
38. Click the Send button; you should get a 400 Bad Request.





400 Bad Request 190.50 ms DETAILS ▼

```
{  
  "message": "Bad request"  
}
```

39. In the request body field, press Cmd+Z or Ctrl+Z so the emptySeats field is added back.
40. Send the request; you should get a 200 OK response with the message Flight updated (but not really).

200 OK 225.24 ms DETAILS ▼

```
{  
  "message": "Flight updated (but not really)"  
}
```


Make a request to a secured API

41. Change the method to GET.
42. Change the request URL to <http://training4-american-api.cloudhub.io/flights/3>.

Note: The -ws in the URL has been changed to -api and the /api removed.

43. Click the Send button; you should get a 401 Unauthorized response with a message about an invalid client_id or secret.

401 Unauthorized 393.32 ms DETAILS ▼







   

Unable to retrieve client_id from message

44. Return to the course snippets.txt file and copy the value for the American Flights API client_id.
45. Return to Advanced REST Client and add a header called client_id.
46. Set client_id to the value you copied from the snippets.txt file.

47. Return to the course snippets.txt file and copy the value for the American Flights API client_secret.
48. Return to Advanced REST Client and add a second header called client_secret.
49. Set client_secret to the value you copied from the snippets.txt file.

Parameters ^

Headers	Authorization	Variables	Actions
 <> Toggle source mode + Insert headers set			
Header name Content-Type	Header value application/json		  
Header name client_id	Header value d1374b15c6864c3682ddbed2a247a826		
Header name client_secret	Header value 4a87fe7e2e43488c927372AEF981F066		

50. Click the Send button; you should get data for flight 3 again.

Note: The API service level agreement (SLA) for the application with this client ID and secret has been set to allow three API calls per minute.

200 OK 1399.56 ms DETAILS ▾

  <> |||

```
[Array(1)]
-0: {
  "ID": 3,
  "code": "ffee0192",
  "price": 300,
  "departureDate": "2016-01-20T00:00:00",
  "origin": "MUA",
  "destination": "LAX",
  "emptySeats": 0,
  "-plane": {
    "type": "Boeing 777",
    "totalSeats": 300
  }
}
```

51. Click the Send button three more times; you should get a 429 Too Many Requests response with a Quota has been exceeded message.

Note: The API service level agreement (SLA) for the application with this client ID and secret has been set to allow three API calls per minute.

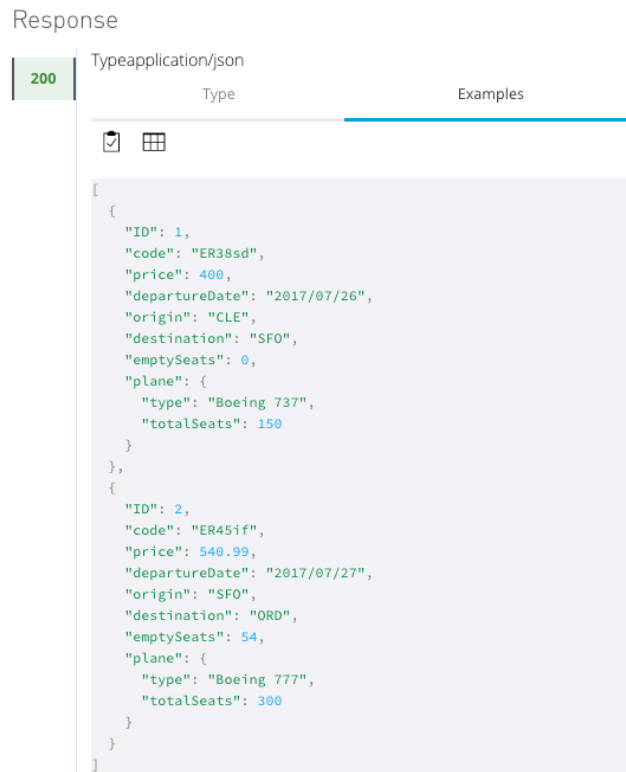
429 Too Many Requests 209.24 ms DETAILS ▾

  <> |||

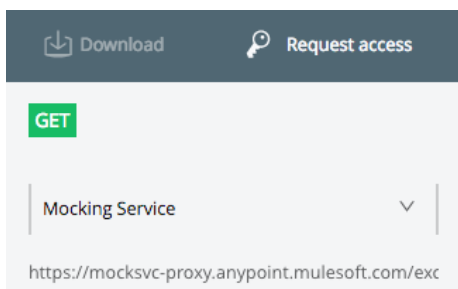
```
{
  "error": "Quota has been exceeded"
}
```


Use the API console in the API portal to make requests to the API using a mocking service

52. Return to the browser window with the American Flights API portal at <https://anypoint.mulesoft.com/exchange/portals/muletraining>.
53. In the left-side navigation click the GET method for /flights.
54. Review the Headers and Response sections.
55. In the Response section, select Examples.



56. In the API console located on the right side of the page, make sure the Mocking Service endpoint is selected.
57. Look at the endpoint URL that is displayed.



58. Click the Show optional parameters checkbox.
59. Select LAX in the destination drop-down menu.

60. Click Send.

61. Look at the response; you should get the example flights that are to SFO.

The screenshot shows an API console interface. At the top, there are tabs for 'Parameters' and 'Headers'. Under 'Parameters', there is a section for 'Query parameters' with a dropdown menu showing 'LAX'. To the right of this dropdown is a checkbox labeled 'Show optional parameters' which is checked. Below the dropdown is a blue 'Send' button. The response status is '200 OK' with a response time of '754.20 ms'. Below the status bar are icons for clipboard, download, code, and table. The response body is a JSON array with two elements, the first of which is expanded to show flight details: ID 1, code ER38sd, price 400, departure date 2017/07/26, origin CLE, destination SFO, and emotvSeats 0.

```
[Array[2]]
-0: {
  "ID": 1,
  "code": "ER38sd",
  "price": 400,
  "departureDate": "2017/07/26",
  "origin": "CLE",
  "destination": "SFO",
  "emotvSeats": 0.
```

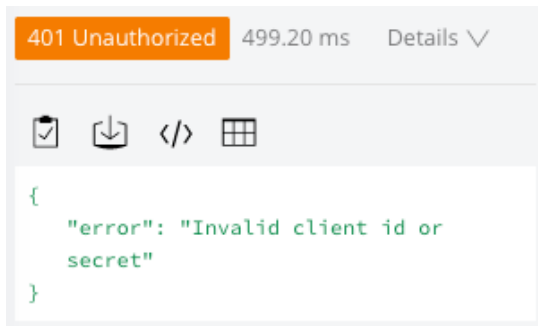
Make requests to the API using an API proxy endpoint

62. At the top of the API console, change the endpoint to Production – Rate limiting SLA based policy.

63. Look at the endpoint URL that is displayed.

The screenshot shows the top of the API console. There are two buttons: 'Download' and 'Request access'. Below these buttons, the method 'GET' is displayed. Underneath, there is a dropdown menu showing 'Production - Rate limiting SLA based policy'. Below the dropdown, the endpoint URL is displayed: 'http://training4-american-api.cloudhub.io/flights?'

64. Click Send; you should get a 401 Unauthorized response.

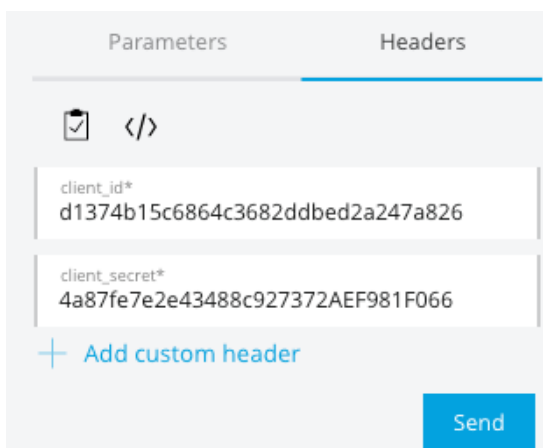


The screenshot shows a REST client interface with a status bar at the top indicating a "401 Unauthorized" response with a duration of "499.20 ms" and a "Details" link. Below the status bar is a toolbar with icons for clipboard, download, code editor, and table view. The main area displays a JSON response:

```
{
  "error": "Invalid client id or secret"
}
```

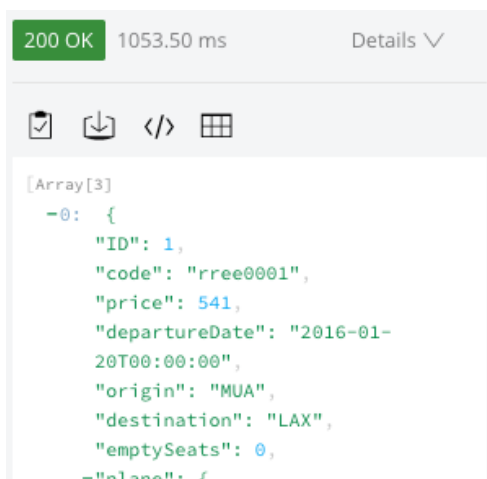
65. Select the Headers tab.

66. Copy and paste the client_id and client_secret values from the course snippets.txt file.



The screenshot shows the "Headers" tab selected in a REST client. The "Parameters" tab is also visible. The Headers section contains two entries: "client_id*" with value "d1374b15c6864c3682ddbed2a247a826" and "client_secret*" with value "4a87fe7e2e43488c927372AEF981F066". Below these is a link to "Add custom header". A "Send" button is at the bottom right.

67. Click Send; you should get a 200 OK response with only flights to LAX.



The screenshot shows a REST client interface with a status bar at the top indicating a "200 OK" response with a duration of "1053.50 ms" and a "Details" link. Below the status bar is a toolbar with icons for clipboard, download, code editor, and table view. The main area displays a JSON array response:

```
[Array[3]]
-0: {
  "ID": 1,
  "code": "rree0001",
  "price": 541,
  "departureDate": "2016-01-20T00:00:00",
  "origin": "MUA",
  "destination": "LAX",
  "emptySeats": 0,
  "plane": {
```