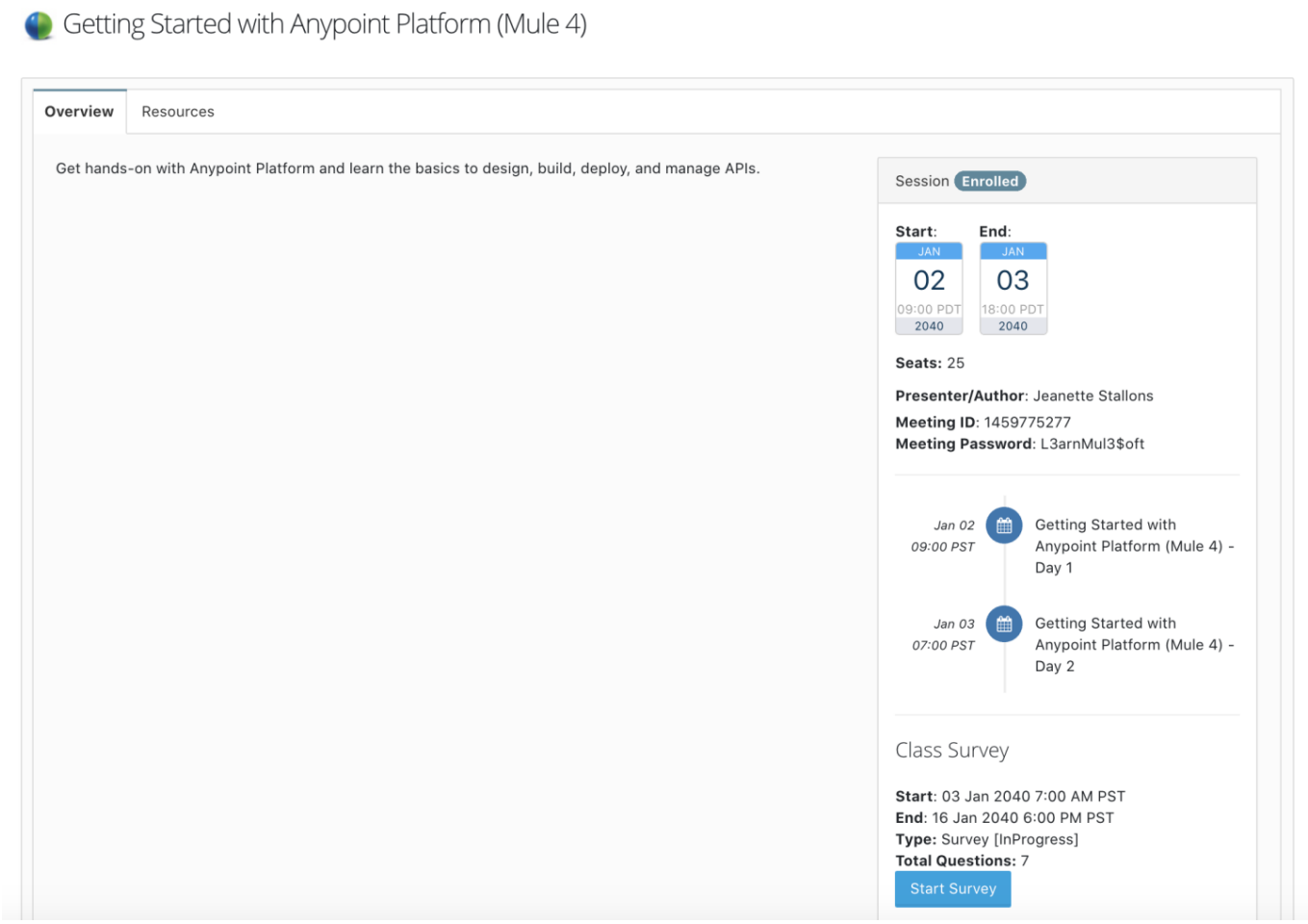
Getting Started with Anypoint Platform

Student Manual

Mule runtime 4.4
October 27, 2022

Table of Contents

Introducing the course



In this module, you will:

- Learn about the course format.
- Download the course files.
- Make sure your computer is set up for class.
- Review the course outline.

Walkthrough: Set up your computer for class

In this walkthrough, you make sure your computer is set up correctly, so you can complete the class exercises. You will:

- Download the course files from the MuleSoft Training Learning Management System.
- Make sure Anypoint Studio starts successfully.
- Install Advanced REST client (if you did not already).
- Make sure you have an active Anypoint Platform account.

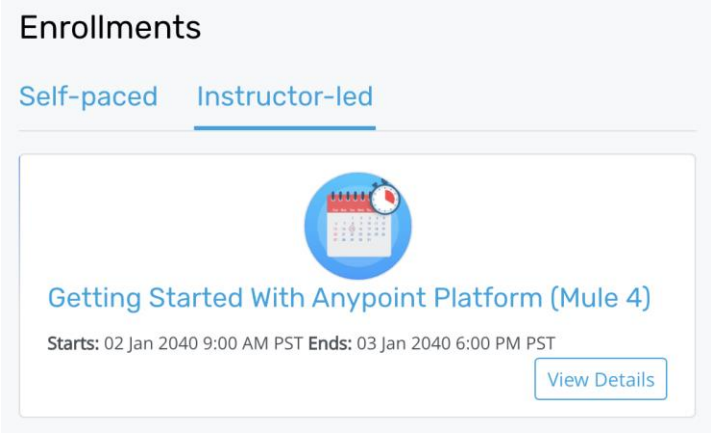
Download student files

1. In a web browser, navigate to http://training.mulesoft.com.
2. Click Login and select Training.

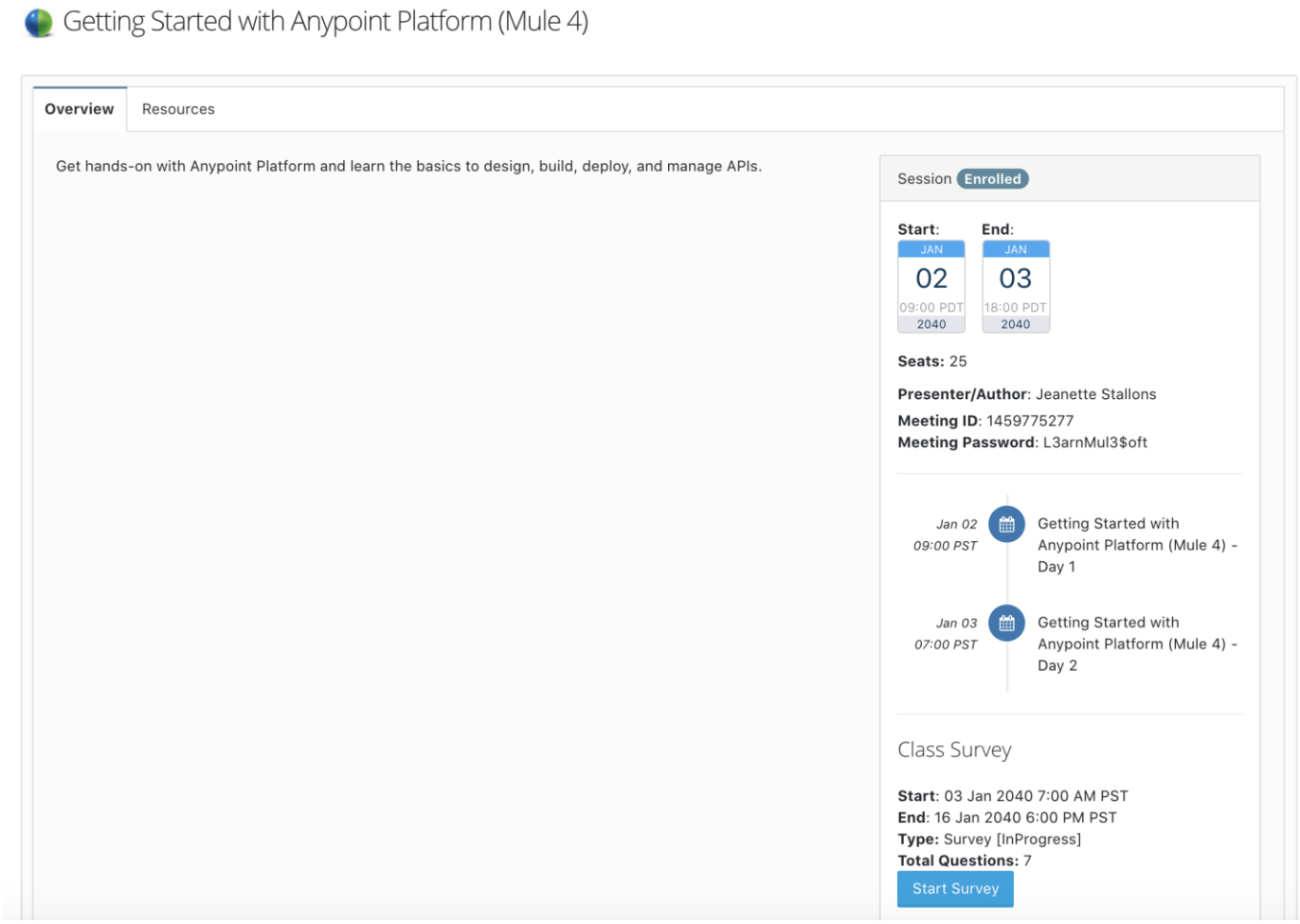
3. Log in to your MuleSoft training account using the email that was used to register you for class.

Note: If you have never logged in before and do not have a password, click the Forgot your password link, follow the instructions to obtain a password, and then log in.

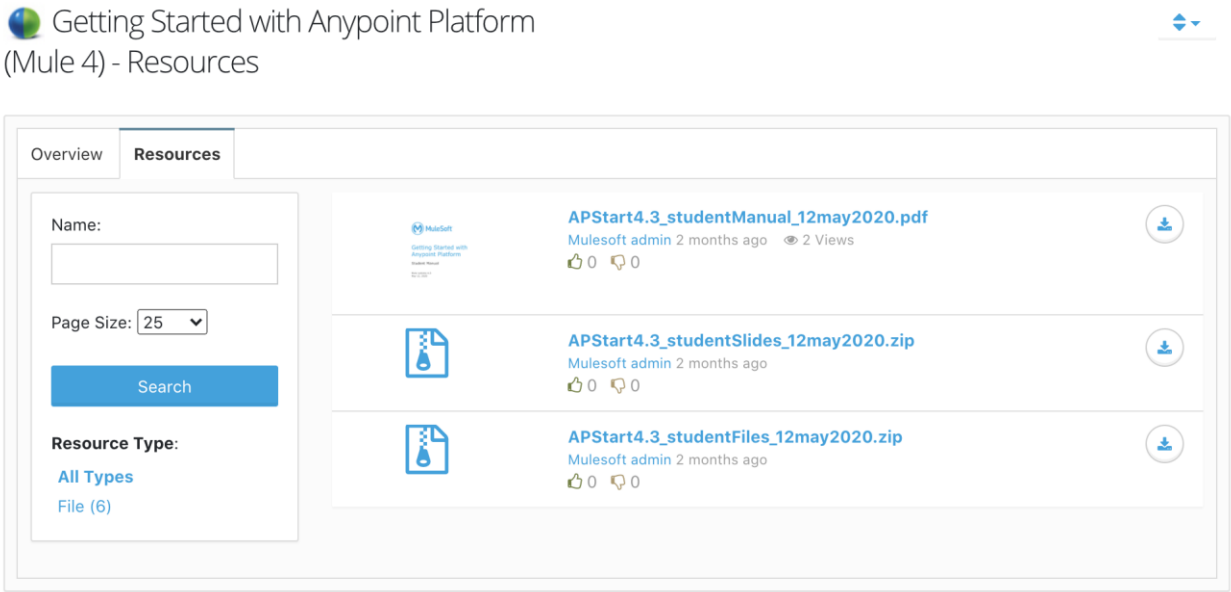
4. On the Dashboard page, select Instructor-led under Enrollments then locate the card for your class.



5. Click the course name to see the course overview page then locate the class details on the right side of the page.



6. Click the Resources tab then locate the list of course materials on the right side of the page.



7. Click the student files link to download the files.
8. Click the student manual link to download the manual.
9. Click the student slides link to download the slides.
10. On your computer, locate the student files ZIP and expand it.
11. Open the course snippets.txt file.

Note: Keep this file open. You will copy and paste text from it during class.

Start Anypoint Studio

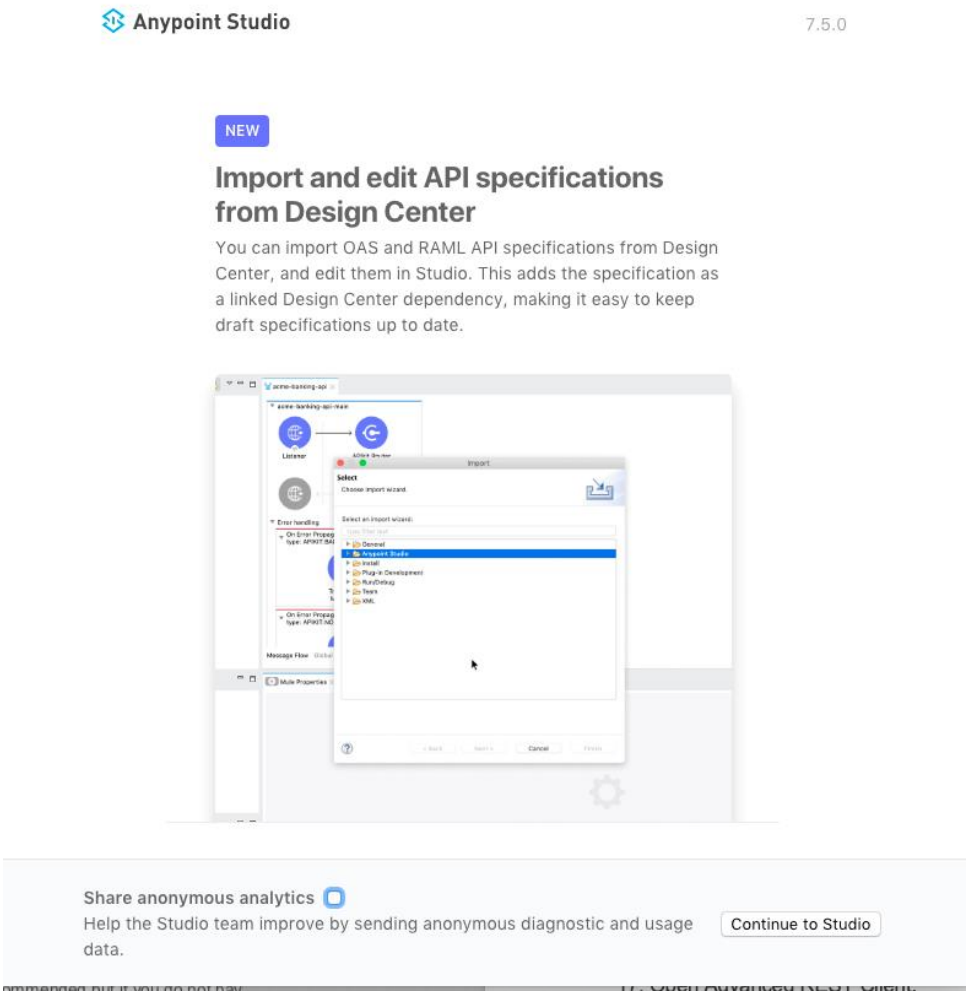
12. In your computer's file browser, navigate to where you installed Anypoint Studio and open it.

Note: If you do not have Anypoint Studio, you can download it from https://www.mulesoft.com/lp/dl/studio. Upon starting Anypoint Studio, users on Windows may get a popup asking to allow Windows Defender Firewall access for OpenJDK; access should be allowed.

13. In the Workspace Launcher dialog box, look at the location of the default workspace; change the workspace location if you want.
14. Click OK to select the workspace; Anypoint Studio should open.

Note: If you cannot successfully start Anypoint Studio, make sure that you have enough available memory (at least 8GB available) to run Anypoint Studio.

15. If you get a new features page, click the Continue to Studio button to close it.



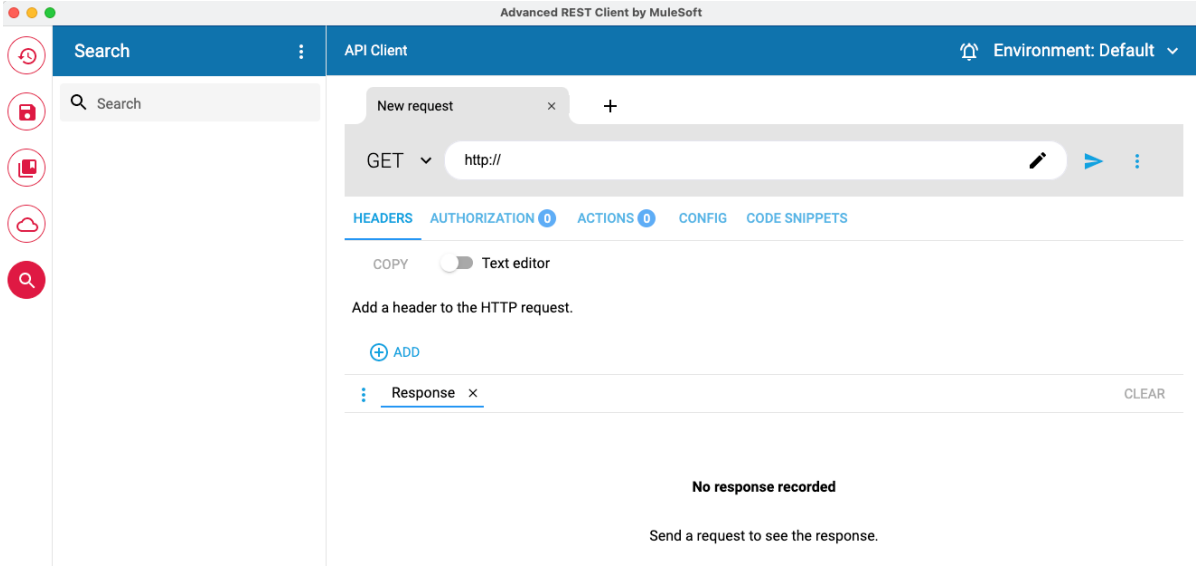
16. If you get an Updates Available popup in the lower-right corner of the application, click it and install the available updates.

Open Advanced REST Client

17. Open Advanced REST Client.

Note: If you do not have Advanced REST Client (or another REST API client) installed, download it now from https://install.advancedrestclient.com/ and install it.

18. Leave Advanced REST client open; you will use it throughout class.



Make sure you have an active Anypoint Platform account

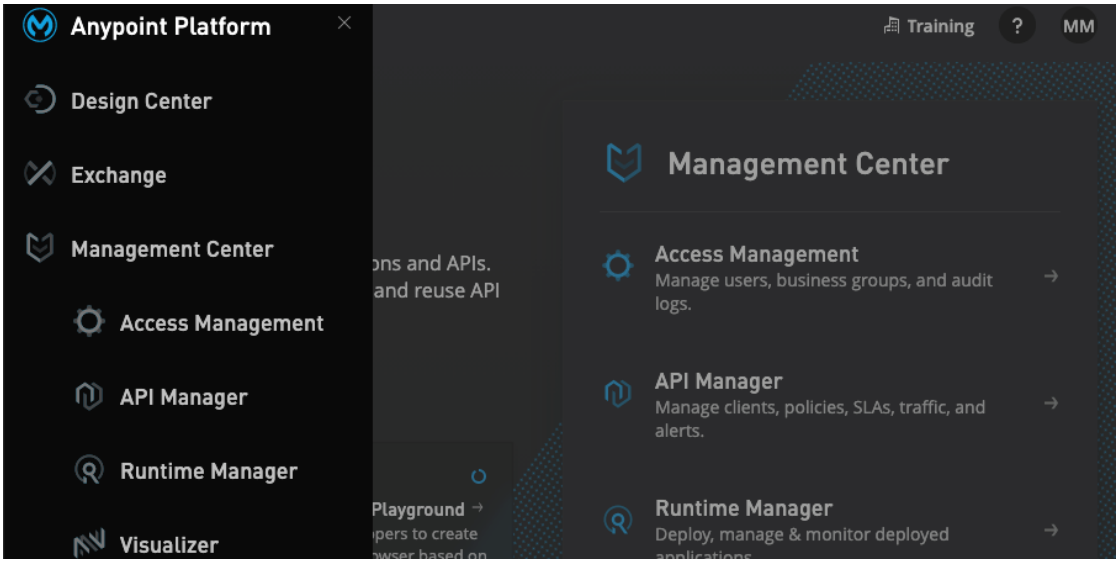
19. In a web browser, navigate to http://anypoint.mulesoft.com/ and log in.

Note: If you do not have an account, sign up for a free, 30-day trial account now. Also, if you get prompted here or in other parts of the course to enable multi-factor authentication, select Not Now.

20. Click the menu button located in the upper-left in the main menu bar.

21. In the menu that appears, select Access Management.

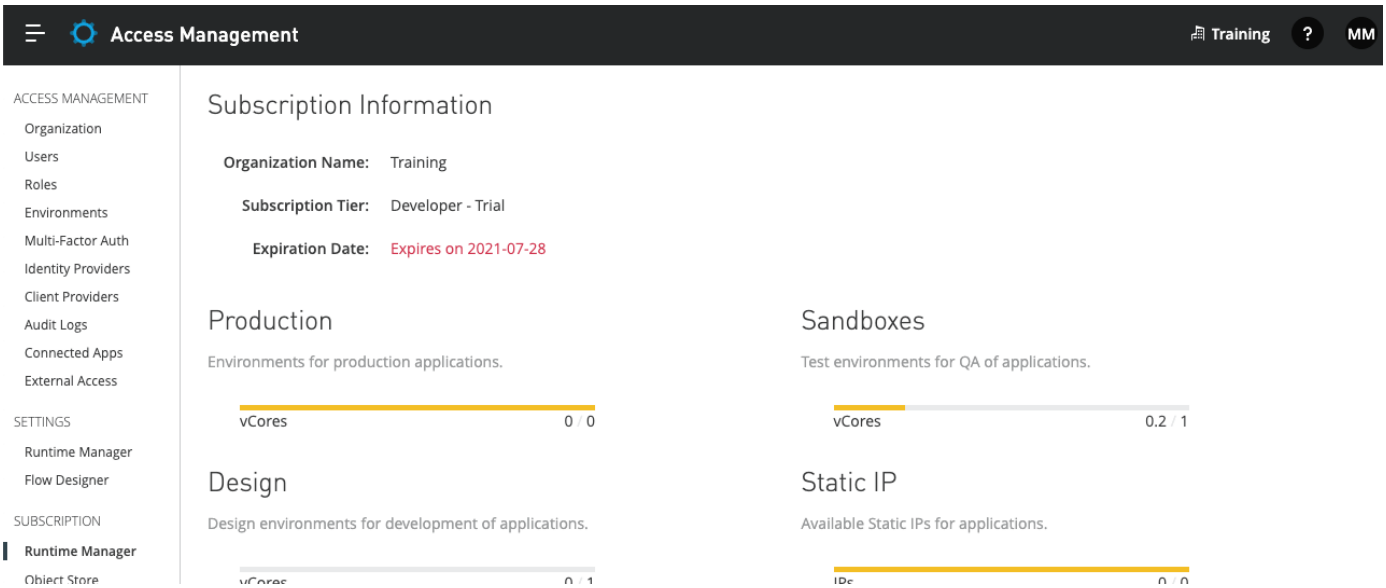
Note: This will be called the main menu from now on.



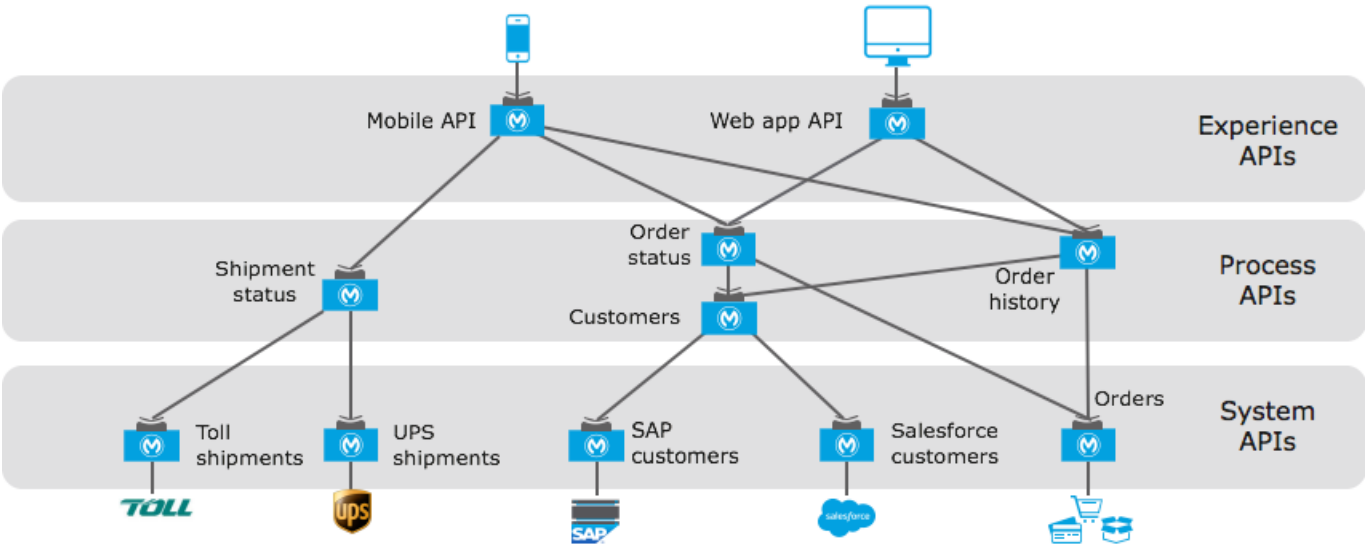
22. In the left-side navigation, click the Runtime Manager link under Subscription.

23. Check your subscription level and if it is a trial account, make sure it is not expired.

Note: If your trial is expired or will expire during class, sign out and then sign up for a new trial account now.



Module 1: Introducing application networks and API-led connectivity



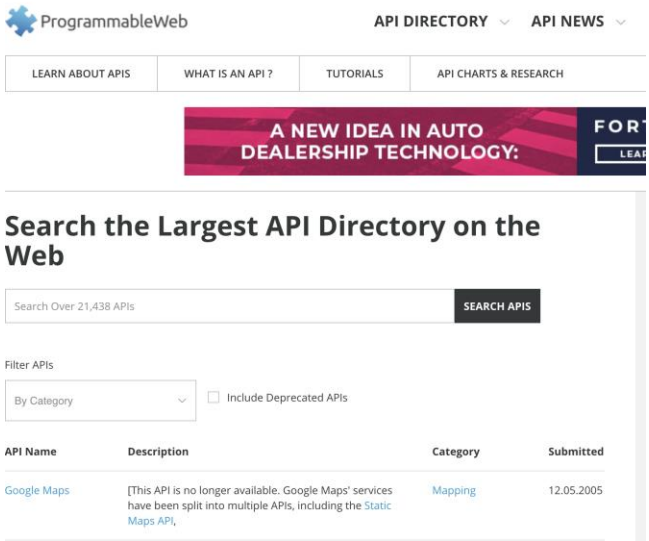
At the end of this module, you should be able to:

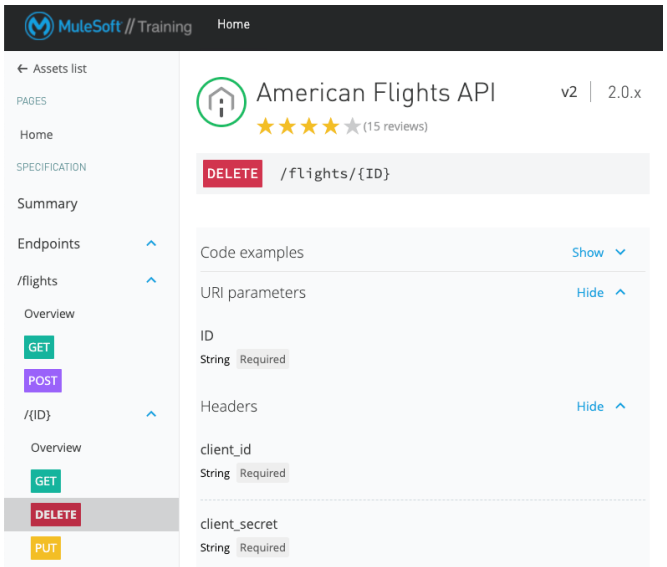
- Explain what an application network is and its benefits.
- Describe how to build an application network using API-led connectivity.
- Explain what web services and APIs are.
- Make calls to secure and unsecured APIs.

Walkthrough 1-1: Explore an API directory and an API portal

In this walkthrough, you locate and explore documentation about APIs. You will:

- Browse the ProgrammableWeb API directory.
- Explore the API reference for an API (Vimeo).
- Explore the API portal for an API to be used in the course.





Browse the ProgrammableWeb API directory

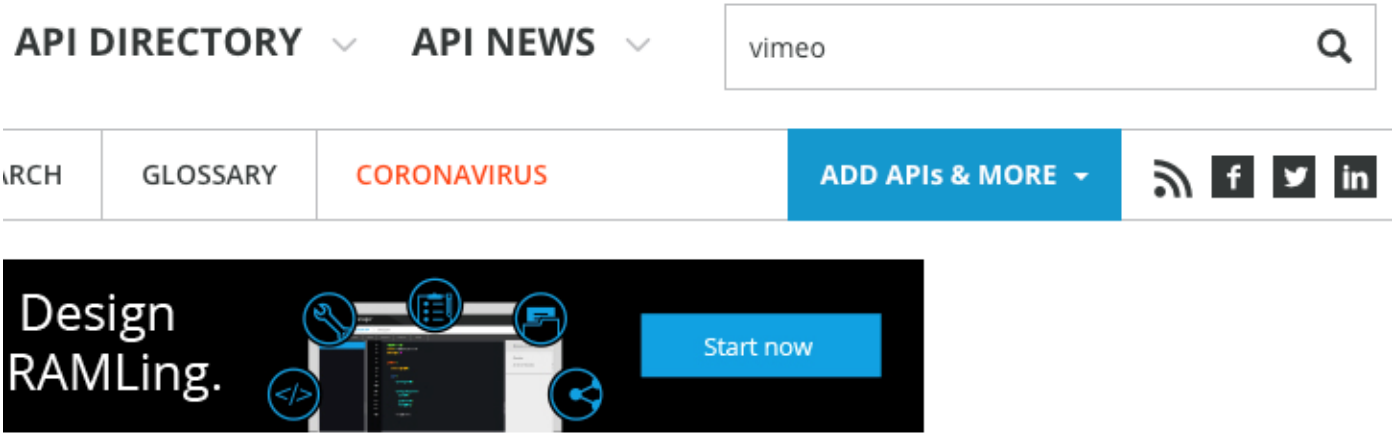
1. In a web browser, navigate to http://www.programmableweb.com/.

Note: If the ProgrammableWeb site does not open, navigate to https://developer.vimeo.com/api/guides/start instead and proceed to step 7 below.

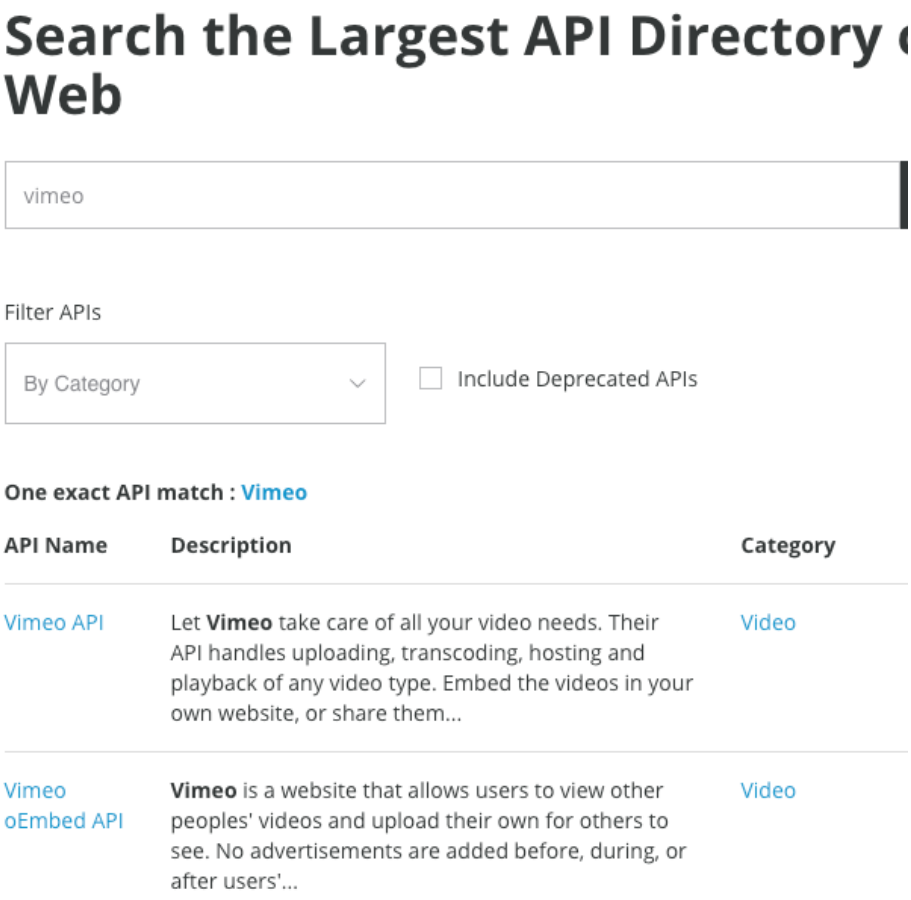
2. Click the API directory link.

Explore the API reference for the Vimeo API

3. Enter vimeo in the search text field then press Enter/Return.

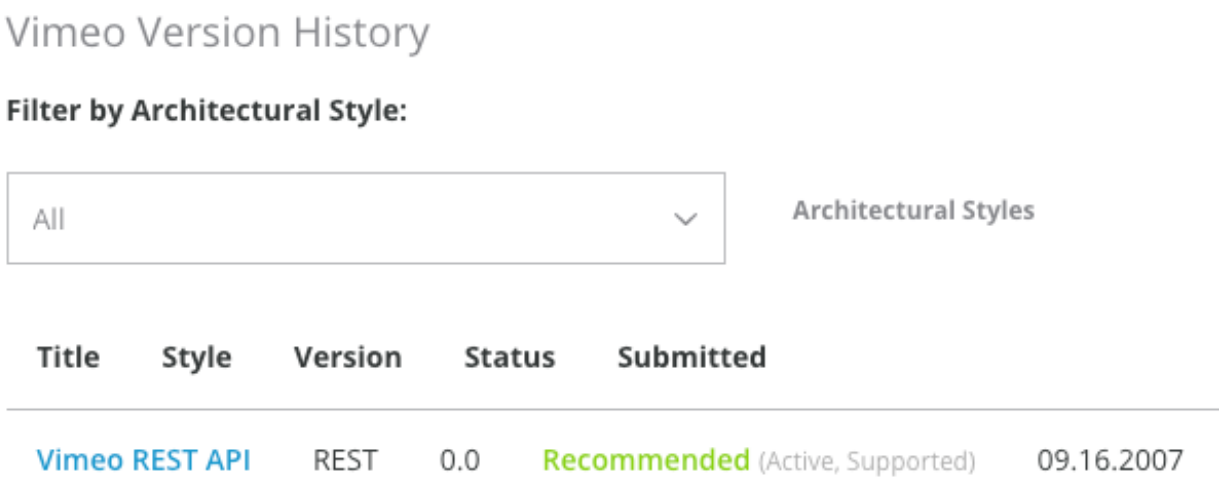


4. In the search results, select the Vimeo API.

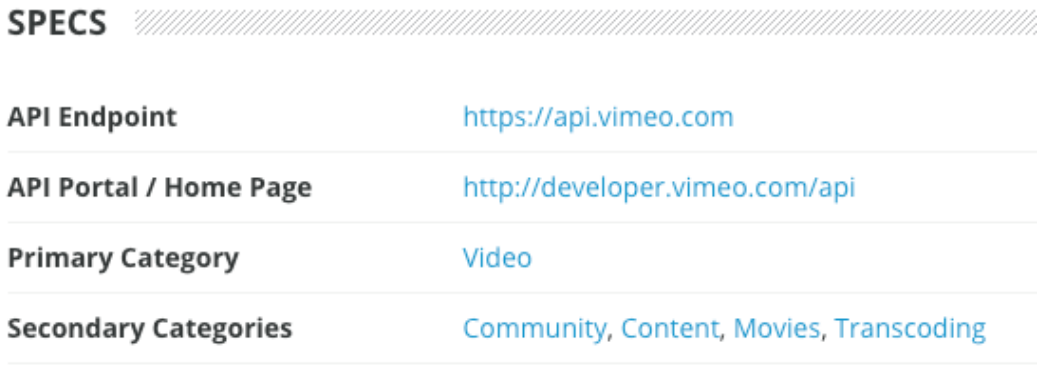


Note: If the search doesn't work, navigate to the URL http://www.programmableweb.com/api/vimeo manually.

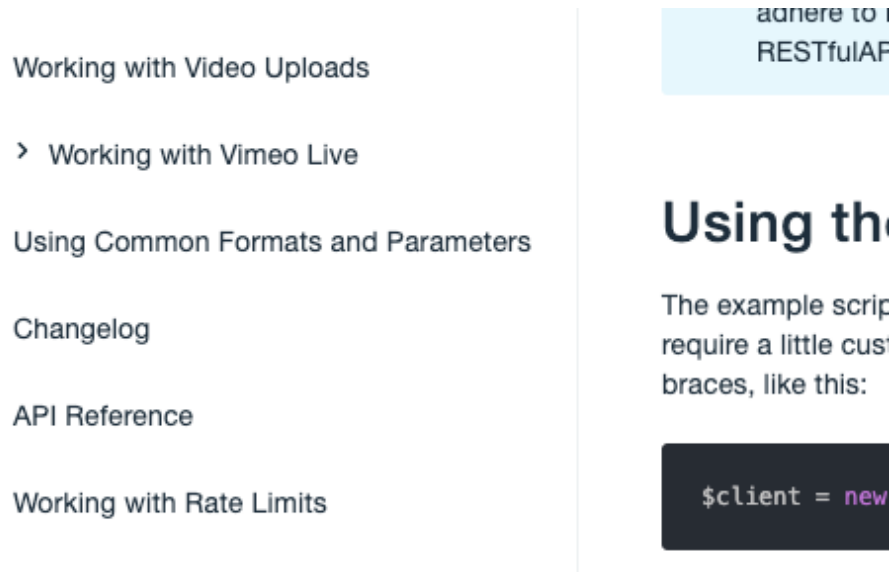
5. In the Vimeo Version History section, select the Vimeo REST API.



6. In the Specs section, click the API Portal / Home Page link.



7. In the new browser tab that opens, click API Reference in the left-side navigation.



8. Review the API Reference for information about endpoints, HTTP methods and parameters.

Using the API Reference

Congratulations! You've hit the motherlode. This is the Vimeo API reference. Here you'll find complete (some might say exhaustive) information about all the methods, endpoints, fields, and values that go into the Vimeo API. If it's listed on these reference pages, it's available for your development project.

This guide explains how to use the reference.

Before you begin

You communicate to the Vimeo API through HTTP messages called requests. Especially if you're coming to us new, you might want to familiarize yourself with three essential components of an API request — endpoints, methods, and parameters — just so we're speaking the same language going forward. Your summary (or refresher course) awaits.

> **NOTE:** Headers are an equally important component of API requests. We touch on headers briefly here, but you can find more detailed information in Using Common Formats and Parameters.

About endpoints

An API *endpoint* is a path that uniquely identifies a Vimeo resource: anything from a video to a user account. The exact path of the endpoint differs depending on the type of resource, but you always append it to `https://api.vimeo.com`.

Table 1 shows a few of the most common endpoints from around these parts. Keep in mind that there are many others. You'll find them all in the pages of the reference.

Table 1. Common endpoints by resource type

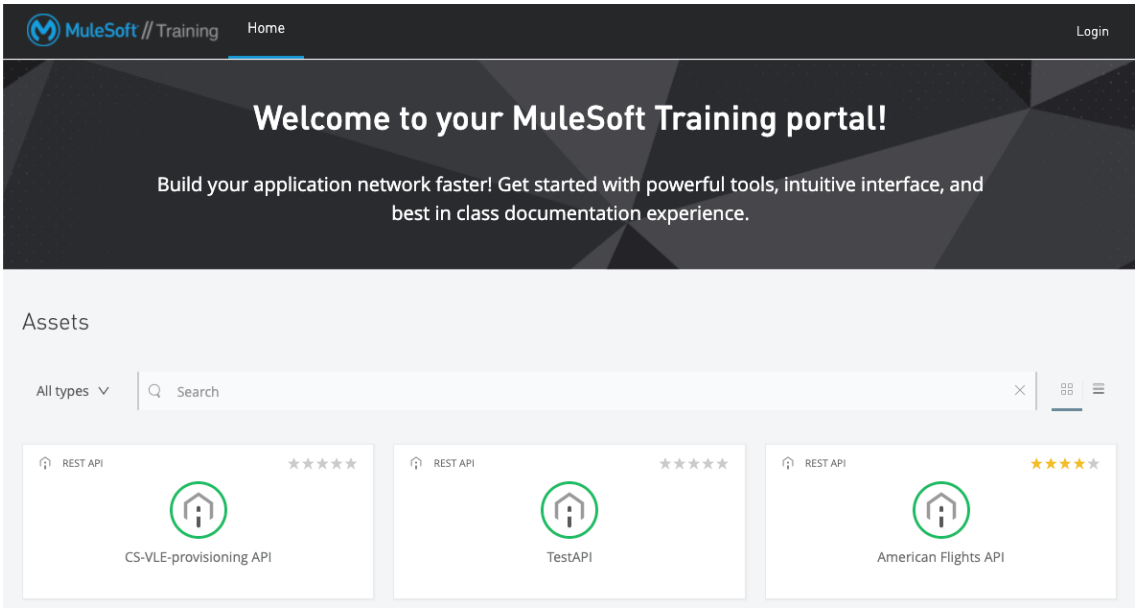
Resource	Endpoint*
Video	`https://api.vimeo.com/videos/{video_id}`
User	`https://api.vimeo.com/users/{user_id}`

9. Close the browser tab.

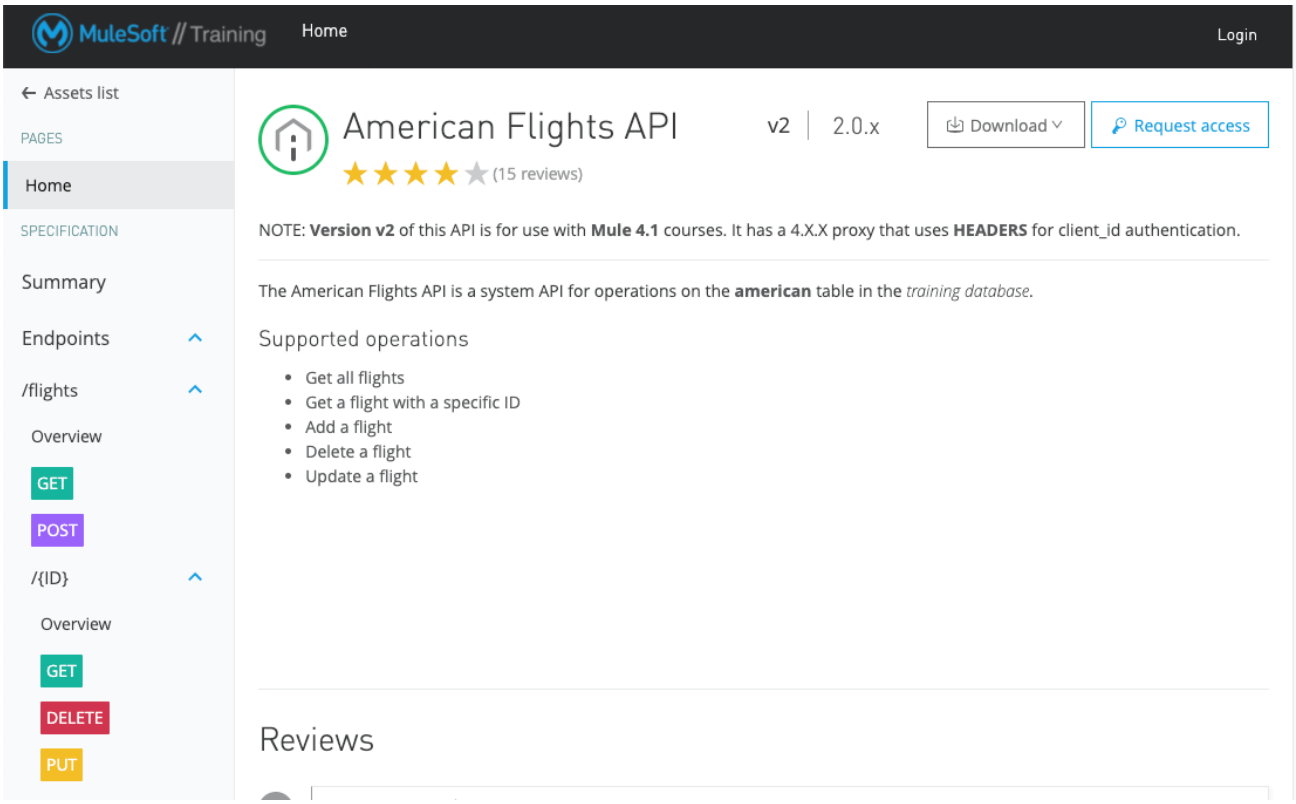
Explore an API portal for an API to be used in the course

10. Return to or open the course snippets.txt file.
11. Copy the URL for the MuleSoft Training API portal.

12. Return to a browser window and navigate to that URL: https://anypoint.mulesoft.com/exchange/portals/muletraining/.

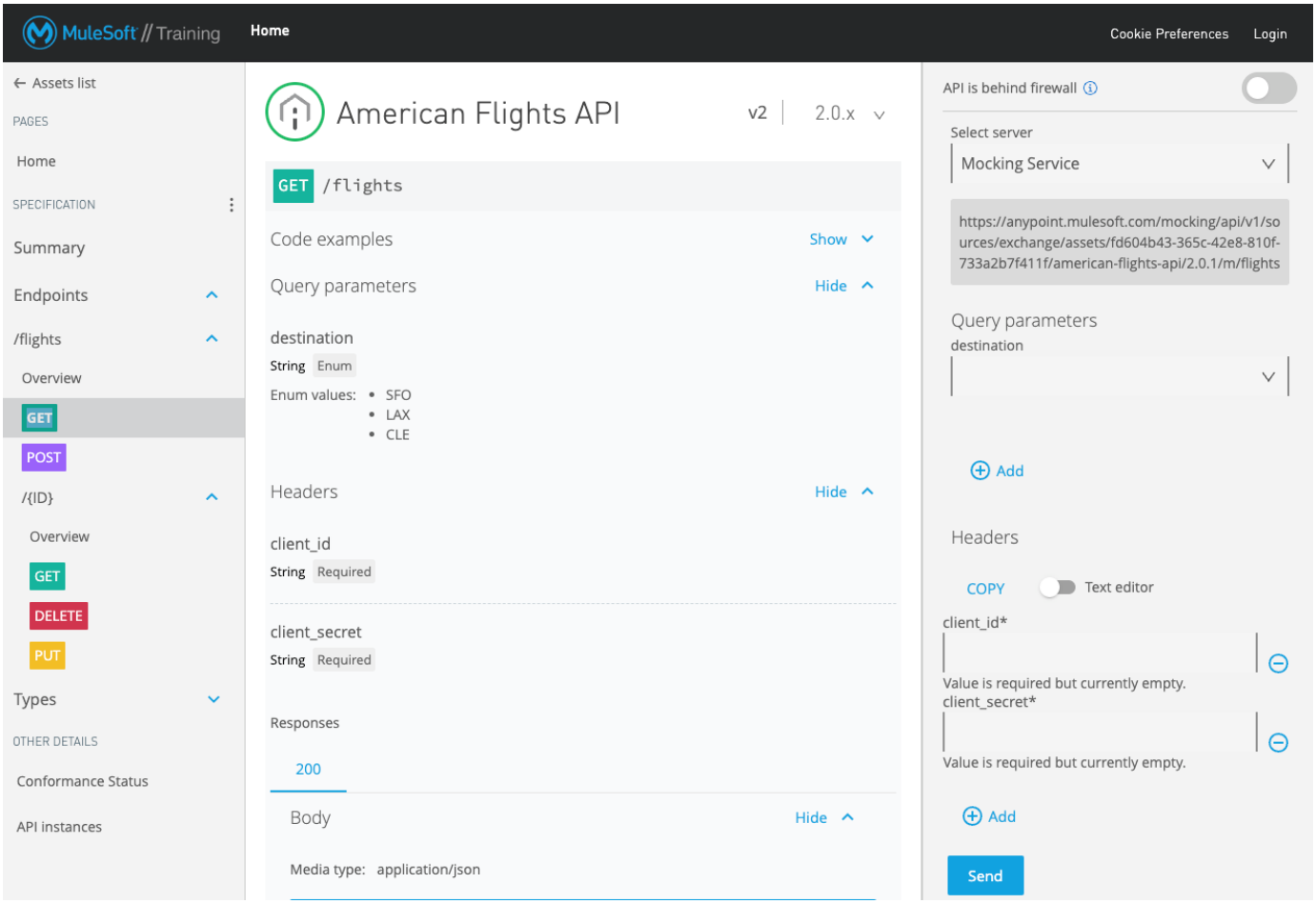


13. Click the American Flights API.
14. Click /flights in the API Summary on the left side.
15. Click /{ID} in the API Summary.
16. Browse the resources that are available for the API.

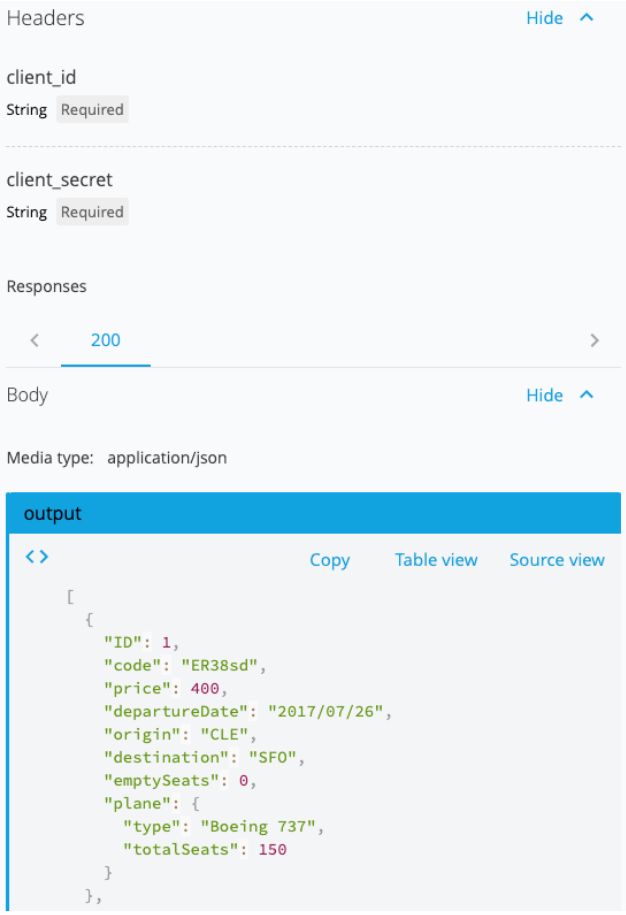


17. Select the GET method for /flights.

18. Review the information about the GET method; you should see there is an optional query parameter called destination.

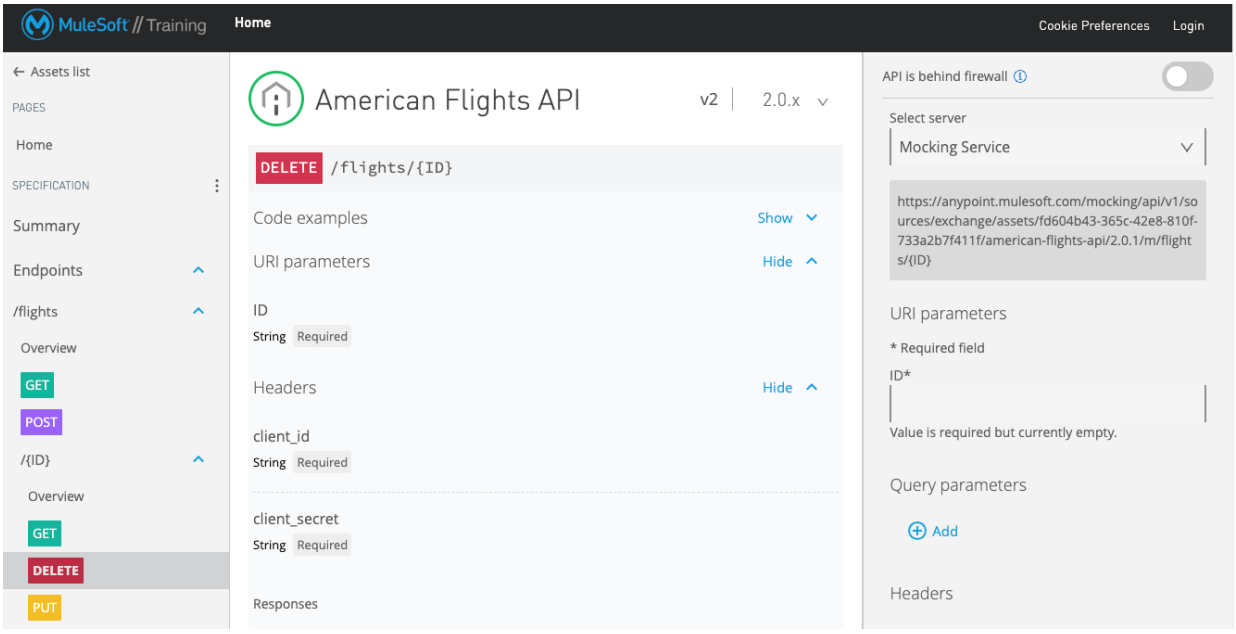


19. Scroll down and review the Headers and Responses sections.



20. In the left-side navigation, select the DELETE method.

21. Locate information about the required ID URI parameter.

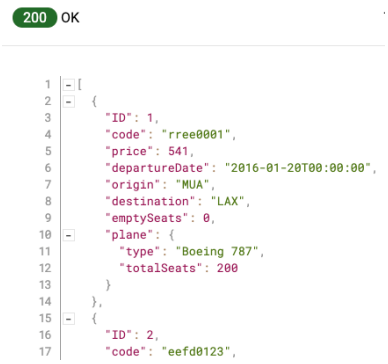


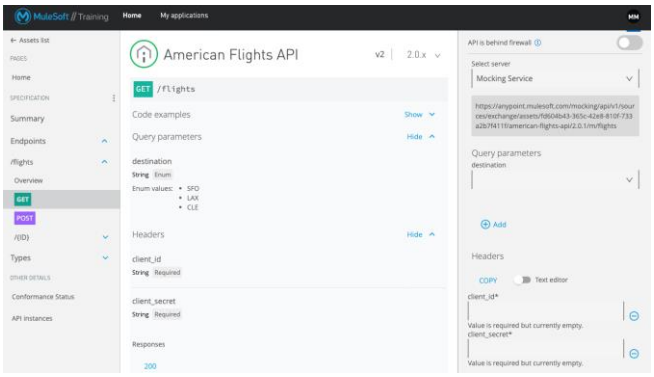
22. Review the information for the other resources.

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.





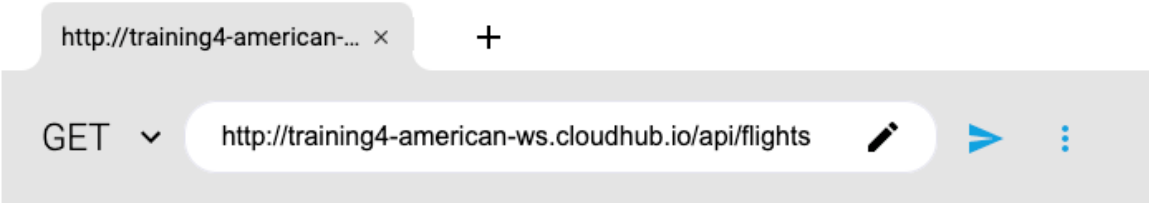
Use Advanced REST Client to make GET requests to retrieve data

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, replacing any existing content.



6. Click the Send the request 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.

Response ×

200 OK


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


9. Select Raw from the options menu in the response area.

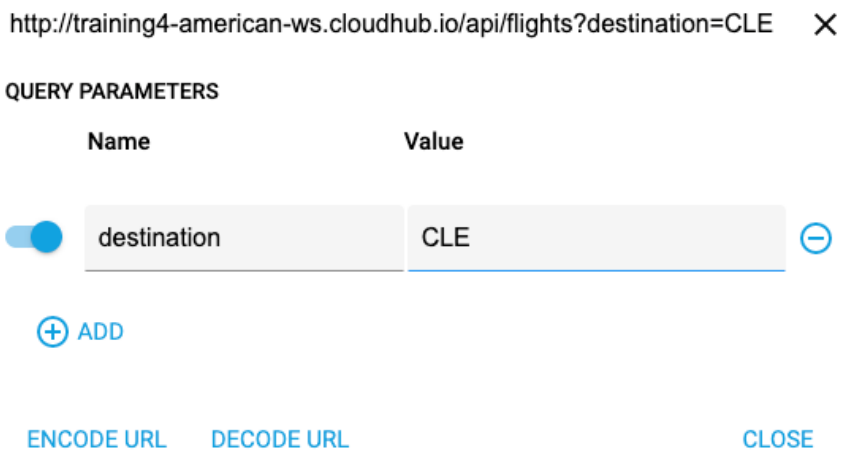
Response Raw ×

200 OK

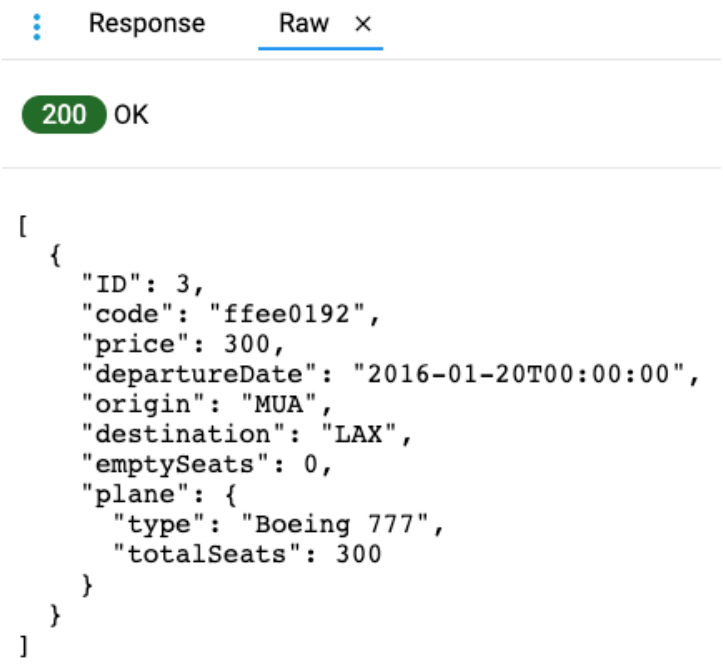

```
[
  {
    "ID": 1,
    "code": "rree0001",
    "price": 541,
    "departureDate": "2016-01-20T00:00:00",
    "origin": "MUA",
    "destination": "LAX",
    "emptySeats": 0,
    "plane": {
      "type": "Boeing 787",
      "totalSeats": 200
    }
  },
  {
    "ID": 2,
    "code": "eefd0123",
    "price": 300,
    "departureDate": "2016-01-25T00:00:00",
```


10. Click the Open parameters editor icon (the pencil) to the right of the URL.
11. In the query parameters editor area that appears, click the Add button.

12. Set the parameter name to destination and the parameter value to CLE then click Close.



13. Click the Send the request button; you should get just flights to CLE returned.
14. Edit the query parameters again, click the Remove this parameter button next to the parameter to delete it, then click Close.
15. Change the request URL to add 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.

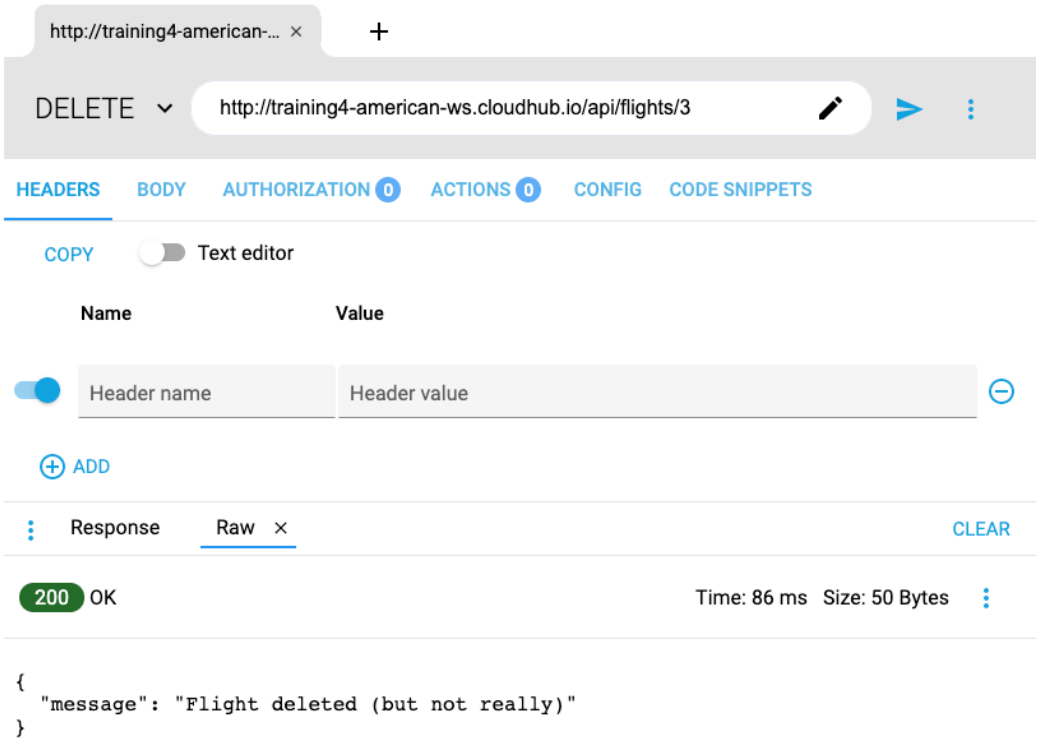


Make DELETE requests to delete data

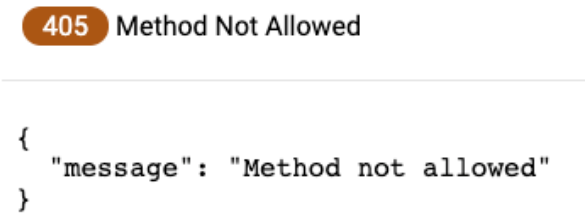
17. Change the method to DELETE.

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

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

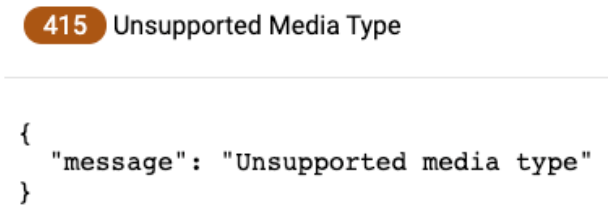


19. Remove the URI parameter from the request: http://training4-american-ws.cloudhub.io/api/flights.
20. Click the send button; you should get a 405 response with the message Method not allowed.

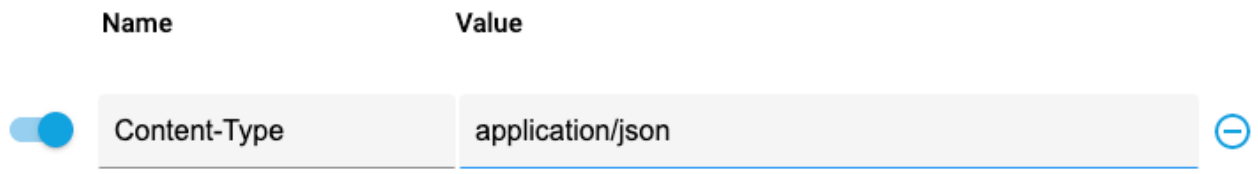


Make a POST request to add data

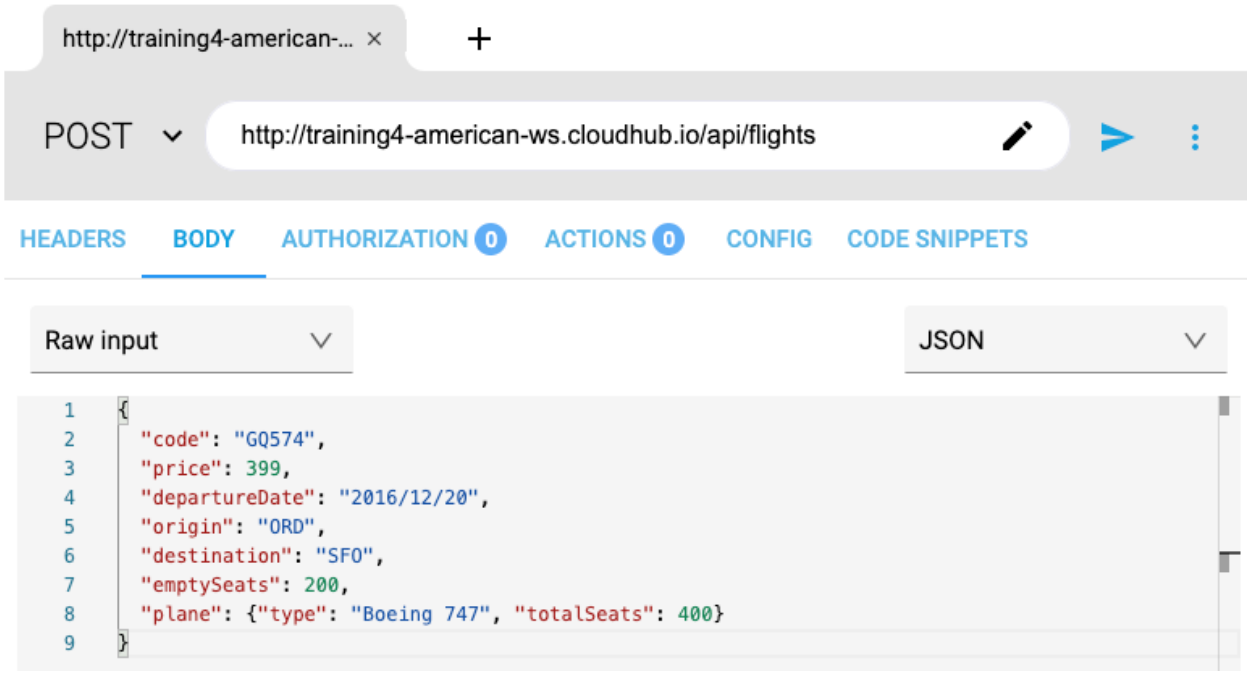
21. Change the method to POST.
22. Click the send button; you should get a 415 response with the message Unsupported media type.



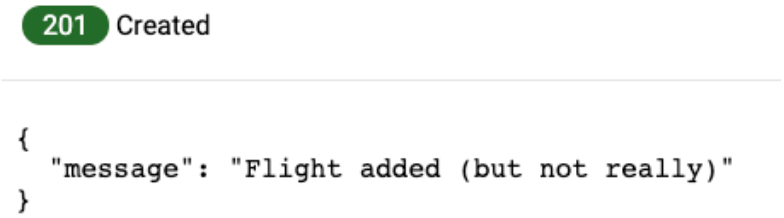
23. Click the Add button in the header area.
24. Select Content-Type in the resultant Header name drop-down menu.
25. Type app in the Header value field then select application/json.



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 BODY AUTHORIZATION 0 ACTIONS 0 CONFIG CODE SNIPPETS

Raw input JSON

```
1 {
2   "code": "GQ574",
3   "price": 399,
4   "departureDate": "2016/12/20",
5   "origin": "ORD",
6   "destination": "SFO",
7   "emptySeats": 200
8 }
```

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 BODY AUTHORIZATION 0 ACTIONS 0 CONFIG CODE SNIPPETS

Raw input JSON

```
1 {
2   "code": "GQ574",
3   "price": 399,
4   "departureDate": "2016/12/20",
5   "origin": "ORD",
6   "destination": "SFO"
7 }
```

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

400 Bad Request

```
{
  "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

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

39. In the request body field, press Cmd+Z or Ctrl+Z until 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

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

Make a request to a secured API

41. Remove the Content-Type header by selecting the Headers tab then clicking the Remove this parameter button next to the header.
42. Change the method to GET.
43. 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.

44. Click the send button; you should get a 401 Unauthorized response with the message Invalid client id or secret.

401 Unauthorized

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

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

50. Set client_secret to the value you copied from the snippets.txt file.

Name	Value
client_id	d1374b15c6864c3682ddbed2a247a826
client_secret	4a87fe7e2e43488c927372AEF981F066

Note: The client credentials in the snippets file may be different than what is shown here; the values in the snippets file differ for instructor-led and self-study training classes.

51. Click the send button; you should get data for flight 3 again.

200 OK


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


52. Click the send button several more times; you should get a 429 Too Many Requests response with the message Quota has been exceeded.

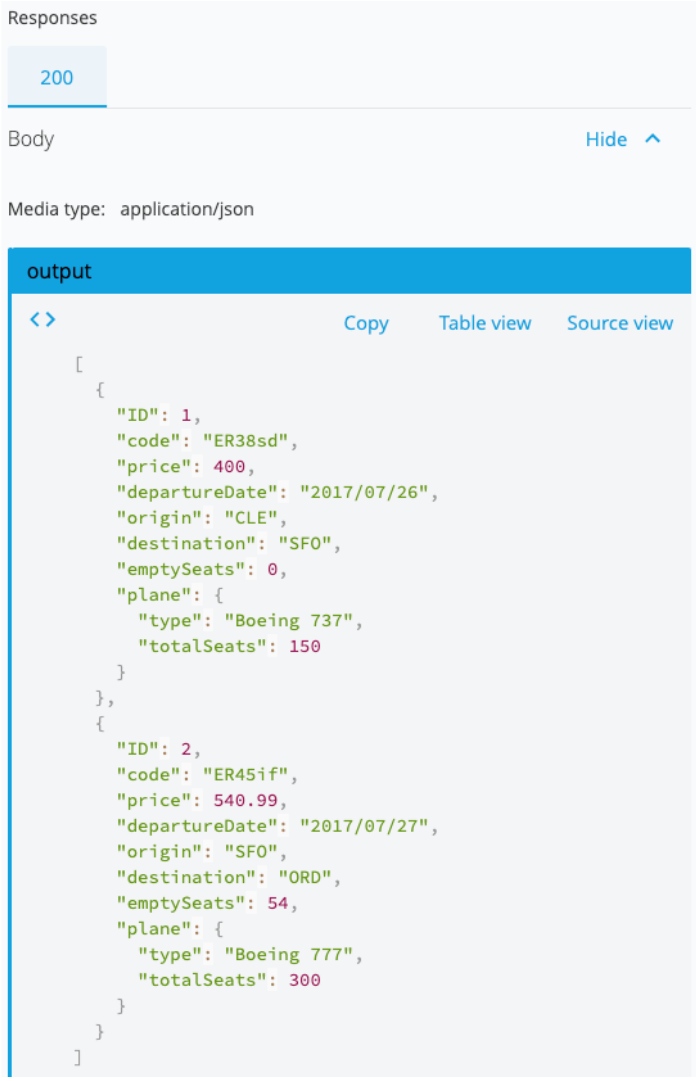
Note: For the self-study training class, the API service level agreement (SLA) for the application with your client ID and secret has been set to allow three API calls per minute while, for the instructor-led class, the SLA allows for a higher number to accommodate shared use.

429 Too Many Requests


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

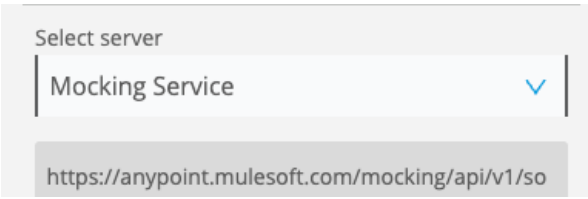
Use the API console in the API portal to make requests to the API using a mocking service

53. Return to the browser window with the American Flights API portal at https://anypoint.mulesoft.com/exchange/portals/muletraining.
54. In the left-side navigation click the GET method for /flights.
55. Review the Headers and Responses sections.
56. In the Responses section, look at the output example.

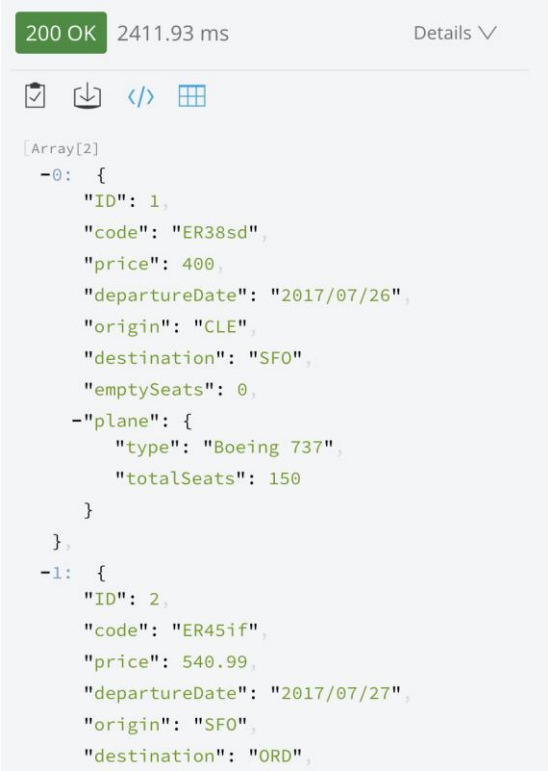


57. In the API console located on the right side of the page, make sure the Mocking Service endpoint is selected.

58. Look at the endpoint URL that is displayed.

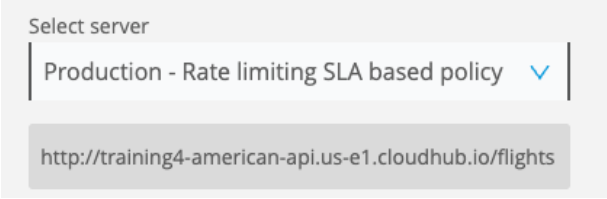


59. Select LAX in the destination drop-down menu.
60. Enter any values for client_id and client_secret.
61. Click Send; you should get the example flights that are to SFO and ORD.

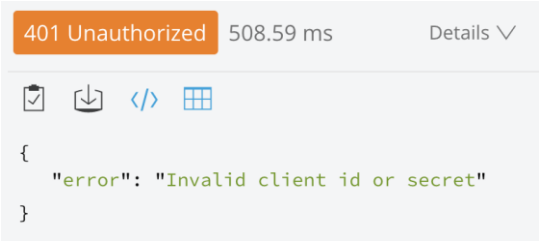


Make requests to the API using an API proxy endpoint

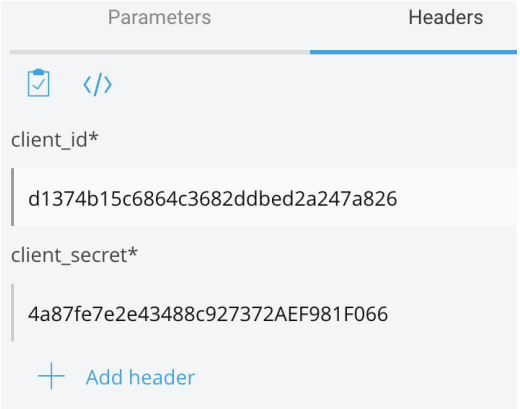
62. At the top of the API console, change the endpoint to Production – Rate limiting SLA based policy.
63. Look at the endpoint URL that is displayed.



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

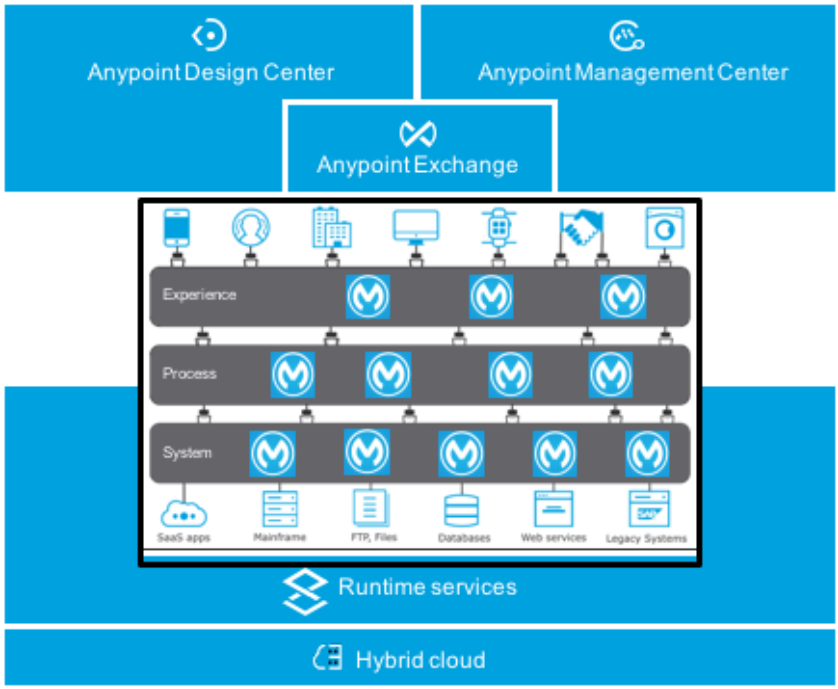


65. Copy and paste the client_id and client_secret values from the course snippets.txt file.



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

Module 2: Introducing Anypoint Platform



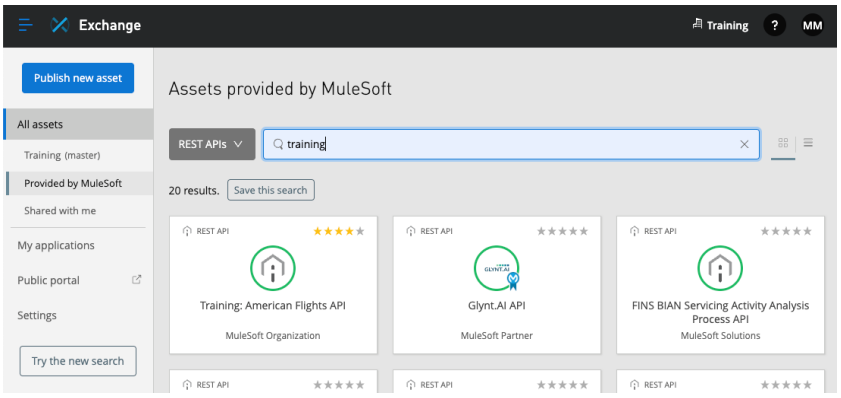
At the end of this module, you should be able to:

- Describe the benefits of Anypoint Platform and MuleSoft's approach to be successful with it.
- Describe the role of each component in building application networks.
- Navigate Anypoint Platform.
- Locate APIs and other assets needed to build integrations and APIs in Anypoint Exchange.

Walkthrough 2-1: Explore Anypoint Platform and Anypoint Exchange

In this walkthrough, you get familiar Anypoint Platform. You will:

- Explore Anypoint Platform.
- Browse Anypoint Exchange.
- Review an API portal for a REST API in Exchange.
- Discover and make calls to the Training: American Flights API in the public Exchange.



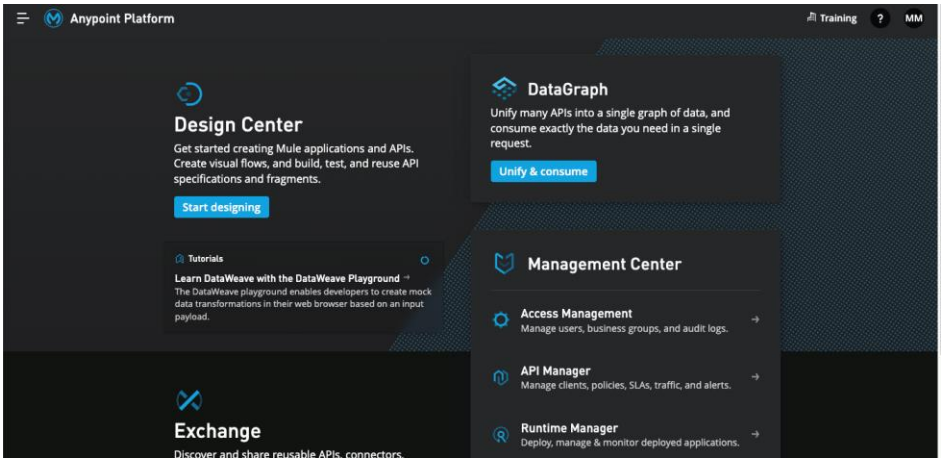
Return to Anypoint Platform

1. Return to Anypoint Platform at https://anypoint.mulesoft.com (not the public API portal you used last module!) in a web browser.

Note: If you closed the browser window or logged out, return to https://anypoint.mulesoft.com and log in.

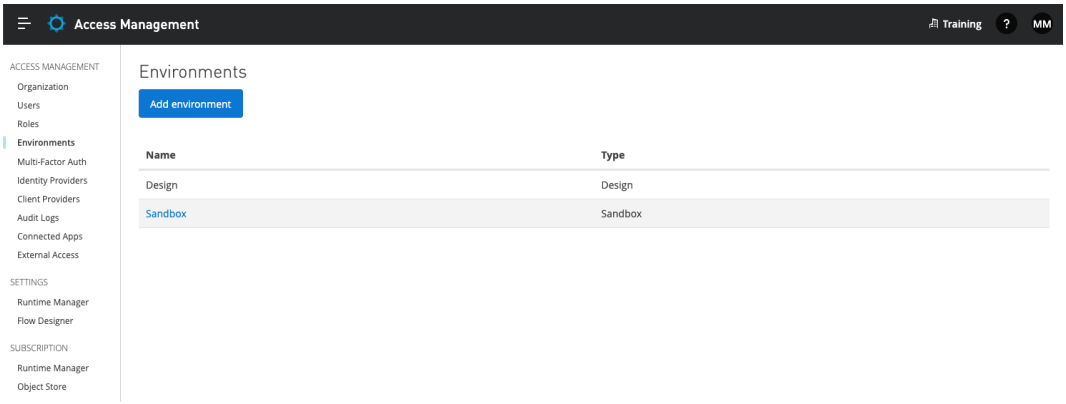
2. Click the menu button located in the upper-left in the main menu bar.
3. In the menu that appears, select Anypoint Platform; this will return you to the home page.

Note: This will be called the main menu from now on.

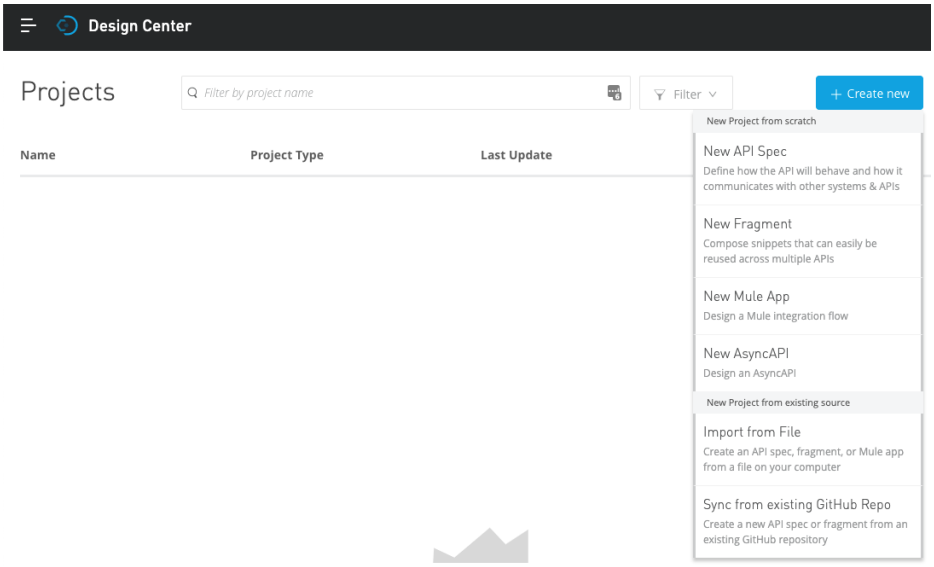


Explore Anypoint Platform

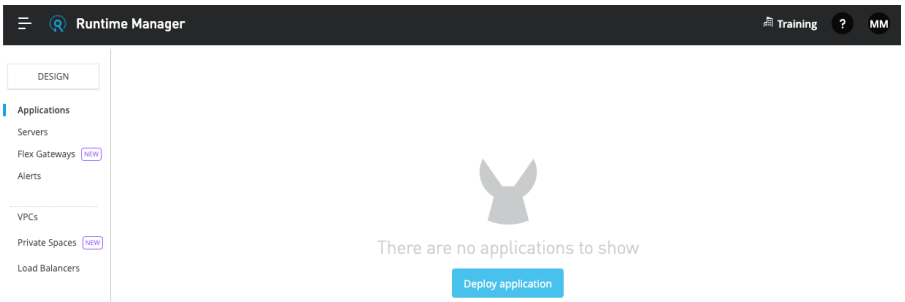
4. In the main menu, select Access Management.
5. In the left-side navigation, select Users.
6. In the left-side navigation, select Environments.



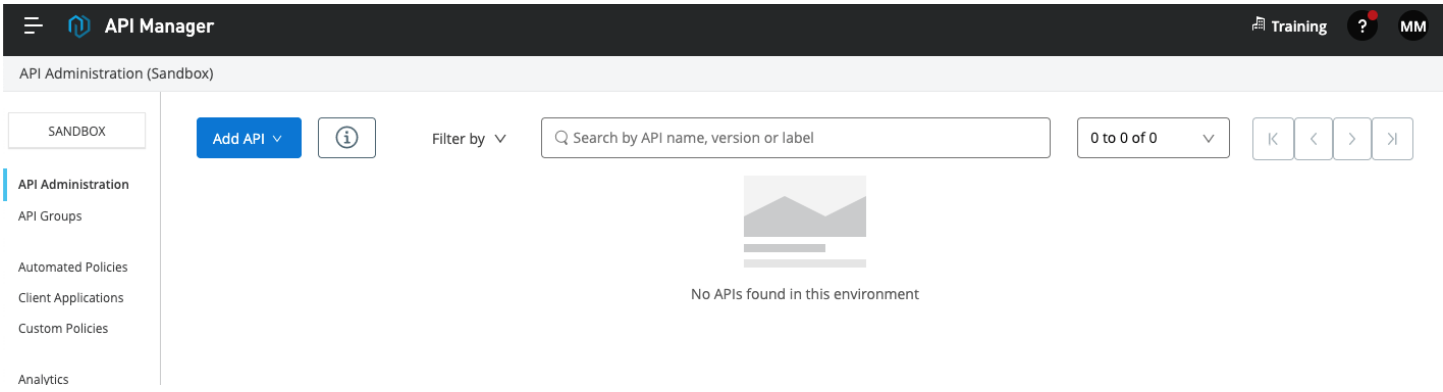
7. In the main menu, select Design Center.
8. Click the Create new button and look at the options in the popup.



9. Close the popup.
10. In the main menu, select Runtime Manager.
11. If you get a Choose Environment page, select Design.

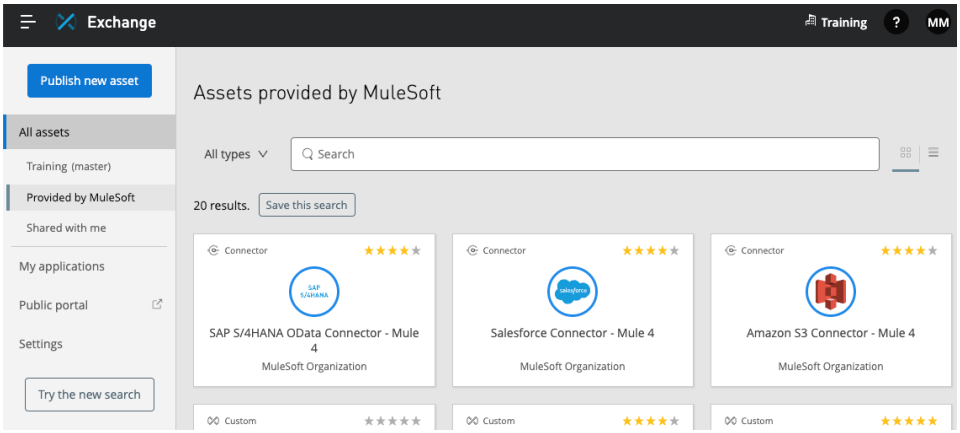


12. In the main menu, select API Manager.

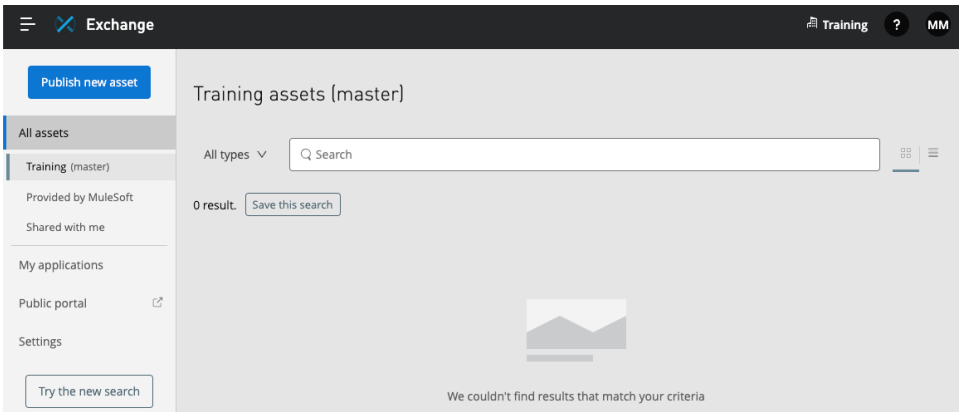


Explore Anypoint Exchange

13. In the main menu, select Exchange.
14. In the left-side navigation, select Provided by MuleSoft; you should see all the content in the public Exchange.

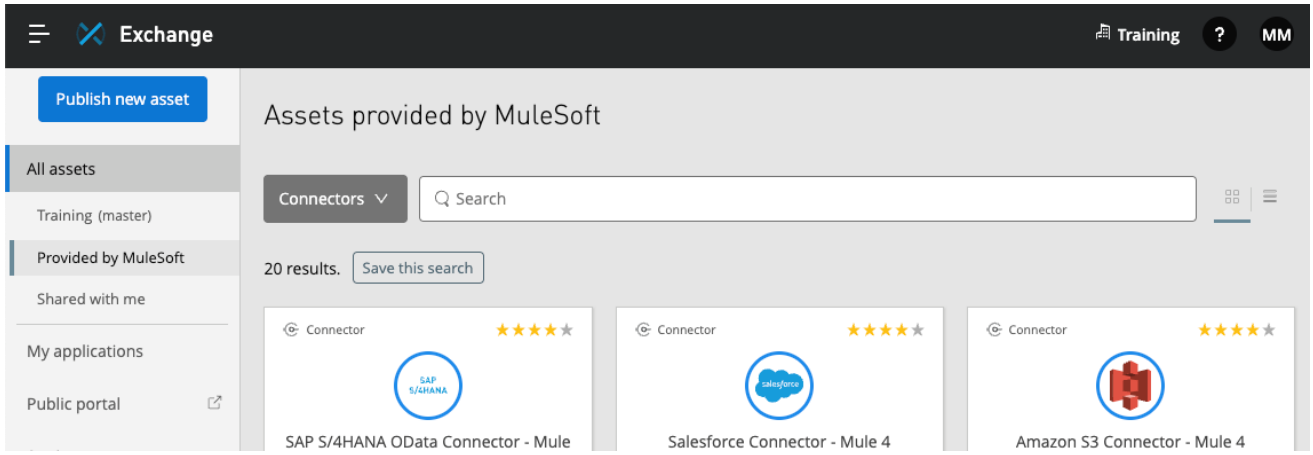


15. In the left-side navigation, select the name of your organization above Provided by MuleSoft (Training in the screenshots); you should now see only the content in your private Exchange, which is currently empty.



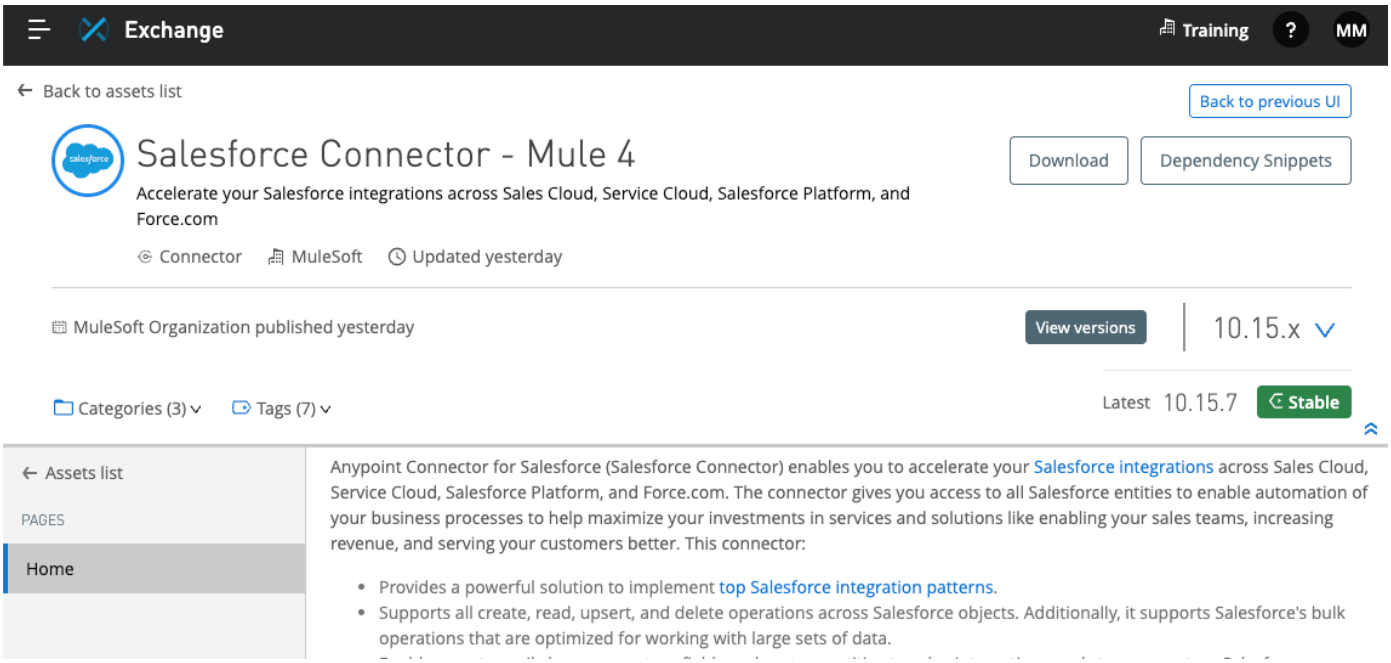
16. In the left-side navigation, select Provided by MuleSoft.

17. In the types menu, select Connectors.



18. Select one of the connectors and review its information.
19. In the left-side navigation, click the Assets list link.
20. Enter salesforce into the search field and press Enter/Return.
21. Locate the Salesforce Connector - Mule 4 connector and review its details.

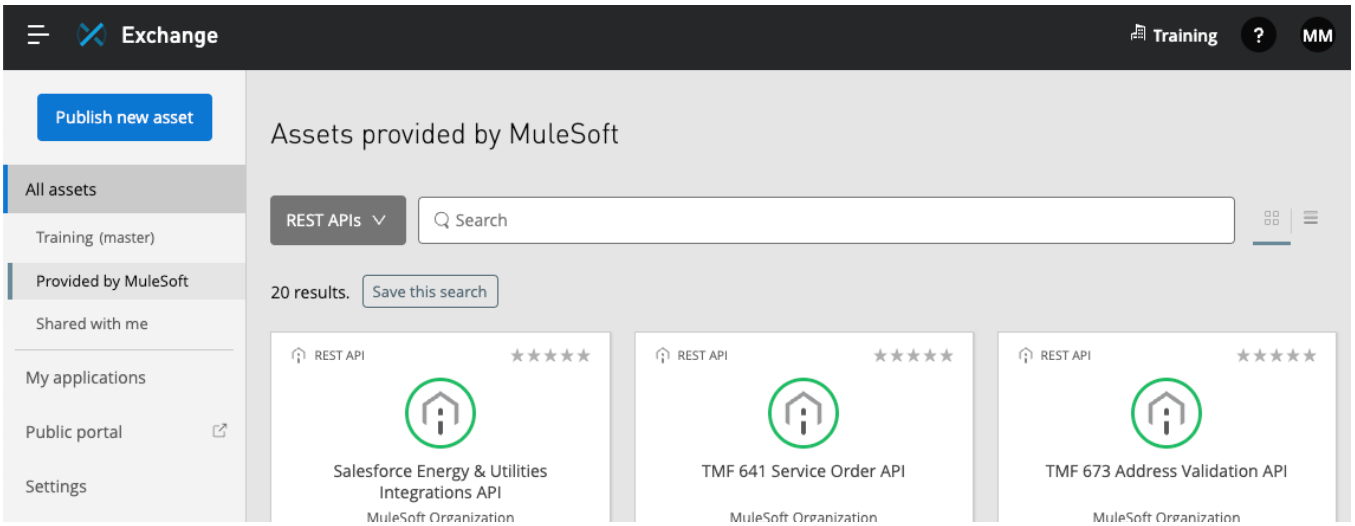
Note: This Salesforce connector is used in the Development Fundamentals course.



22. In the left-side navigation, click Assets list.
23. In the types menu, select Templates.
24. Remove salesforce from the search field and press Enter/Return.

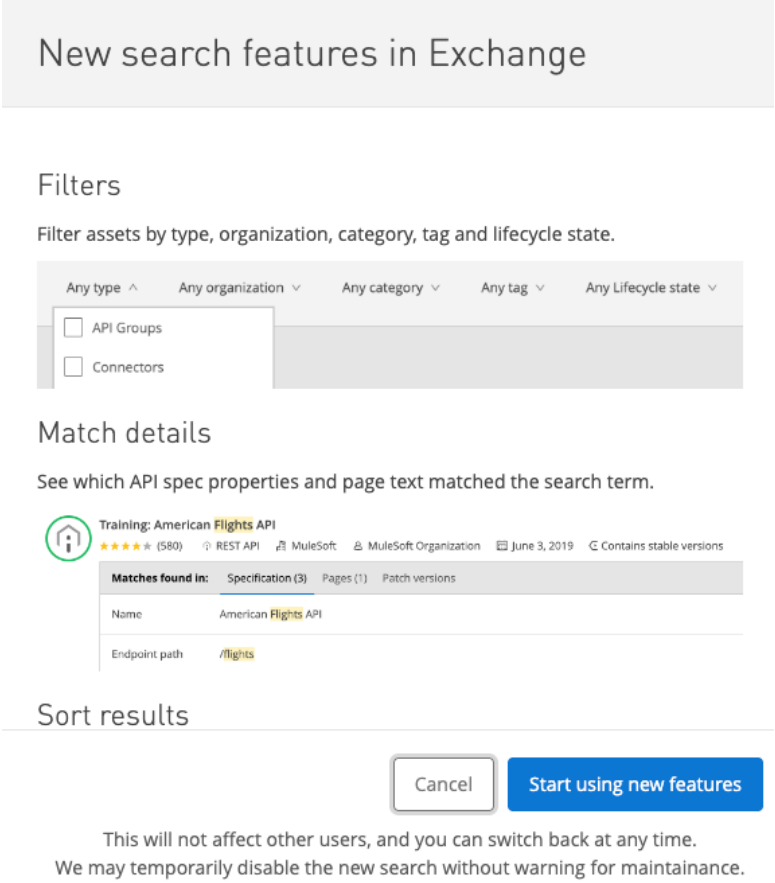
Browse REST APIs in Anypoint Exchange

25. In the types menu, select REST APIs.
26. Browse the APIs.



Discover and review the API portal for the Training: American Flights API

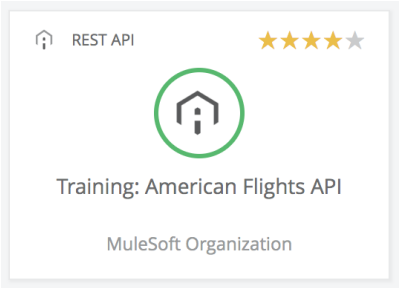
27. Click Try the new search and examine the new search features available in Exchange.



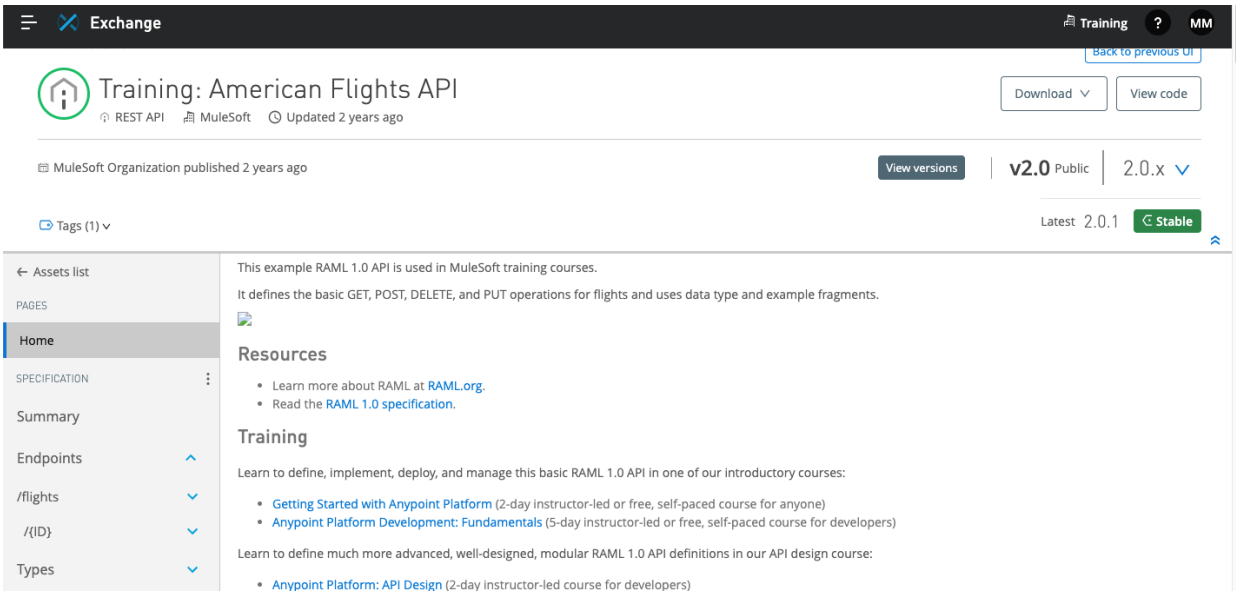
28. Click Cancel.

Note: For this course we will use the original Exchange search experience.

29. Locate and click the Training: American Flights API.



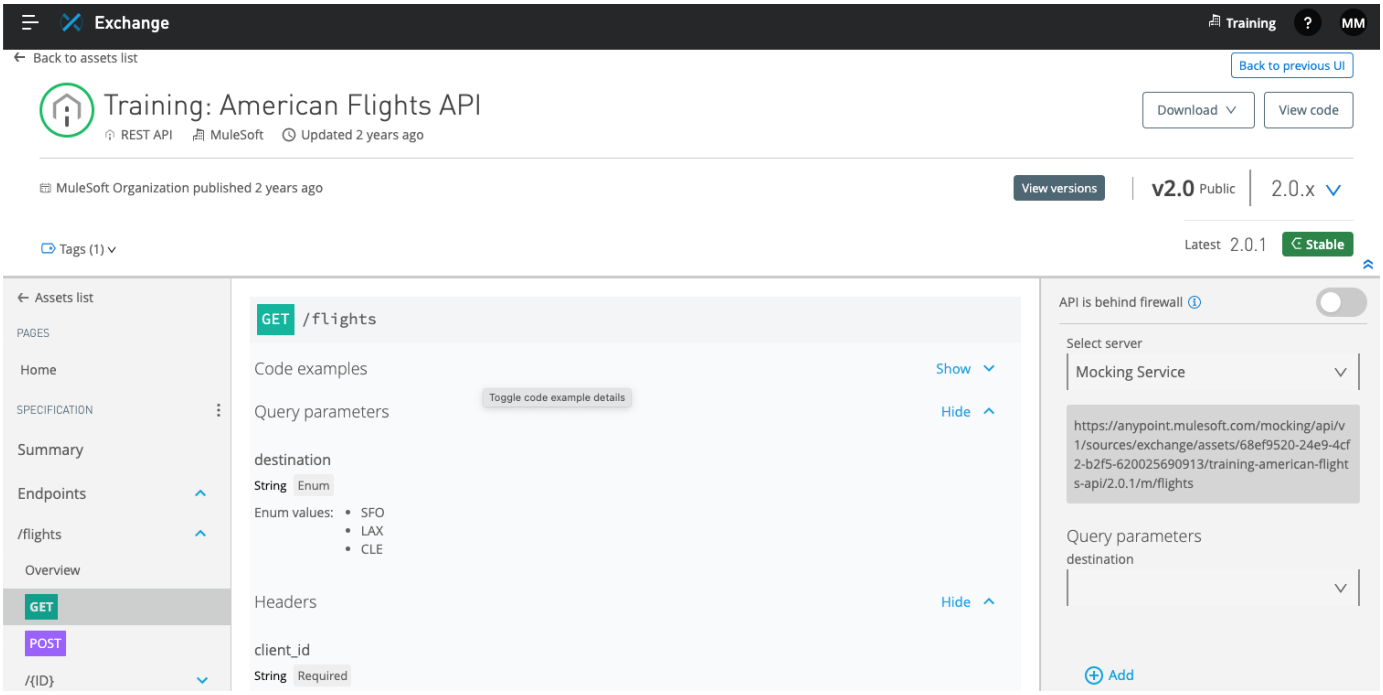
30. Review the API portal.



31. In the left-side navigation, expand and review the list of available resources.

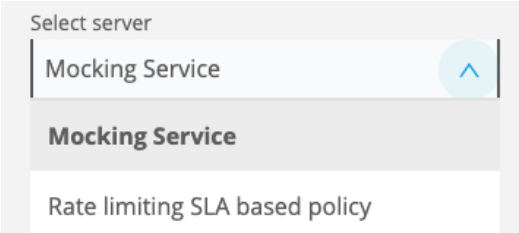
32. Click the GET link for the /flights resource.

33. On the GET /flights page, review the information for the optional destination query parameter; you should see the API is similar to the one you explored in the public MuleSoft Training portal.

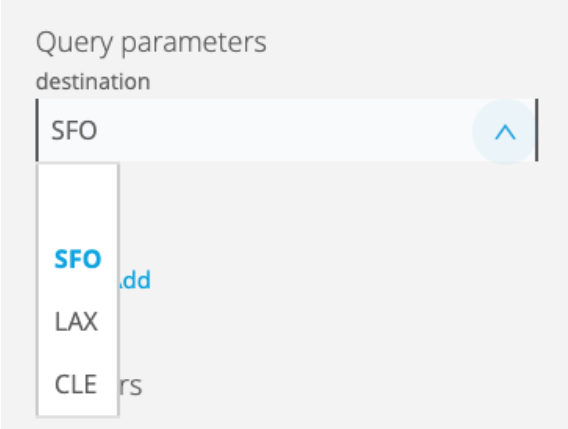


Use the API console to make calls to the Training: American Flights API

34. In the API console, review the options for the instances you can test.



35. Select Mocking Service.
36. Select a destination in the drop-down menu.



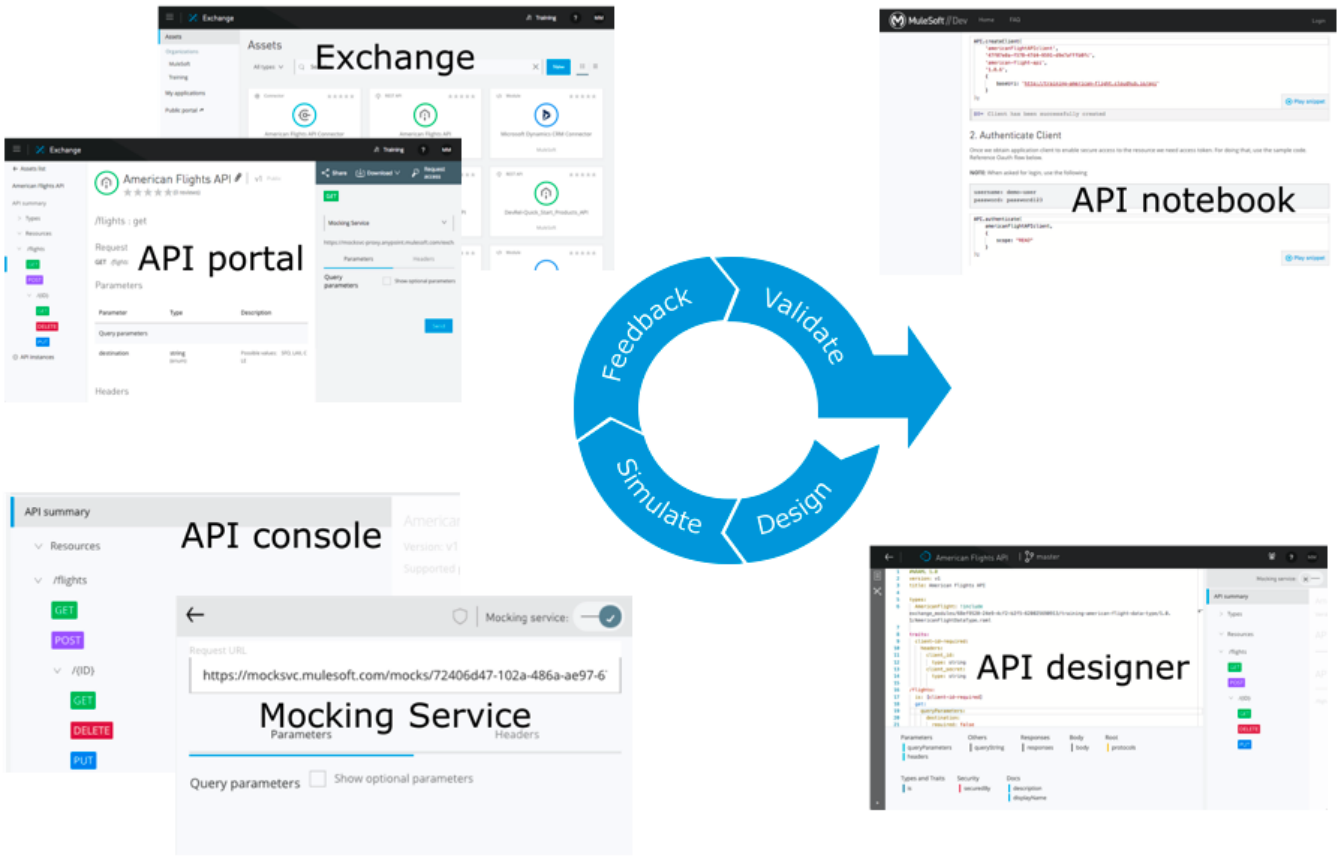
37. In the Headers section, enter any values for client_id and client_secret.
38. Click Send; you should get the two example flights.


```
[
  {
    "ID": 1,
    "code": "ER38sd",
    "price": 400,
    "departureDate": "2017/07/26",
    "origin": "CLE",
    "destination": "SFO",
    "emptySeats": 0,
    "plane": {
      "type": "Boeing 737",
      "totalSeats": 150
    }
  },
  {
    "ID": 2,
    "code": "ER45if",
    "price": 540.99,
```


39. Change the API instance from Mocking Service to Rate limiting SLA based policy.
40. Select a destination in the drop-down menu.
41. In the Headers section, copy and paste the client_id and client_secret values from the course snippets.txt file
42. Click Send again; you should get results from the actual API implementation for the destination you selected.


```
[
  {
    "ID": 1,
    "code": "rree0001",
    "price": 541,
    "departureDate": "2016-01-20T00:00:00",
    "origin": "MUA",
    "destination": "LAX",
    "emptySeats": 0,
    "plane": {
      "type": "Boeing 787",
      "totalSeats": 200
    }
  },
  {
    "ID": 3,
    "code": "ffee0192",
    "price": 300,
    "departureDate": "2016-01-20T00:00:00",
    "origin": "MUA",
    "destination": "LAX",
    "emptySeats": 0,
```

Module 3: Designing APIs



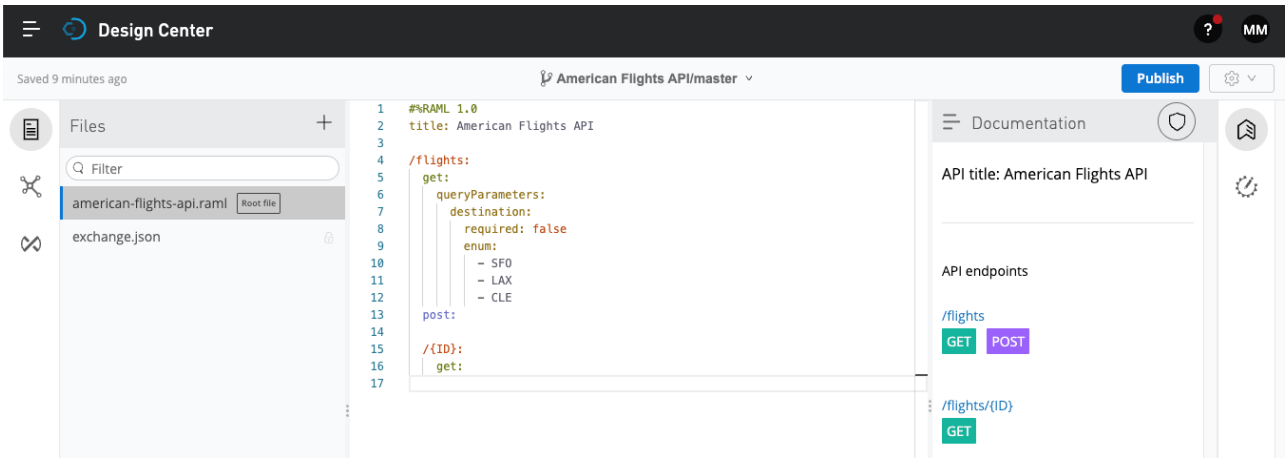
At the end of this module, you should be able to:

- Define APIs with RAML, the Restful API Modeling Language.
- Mock APIs to test their design before they are built.
- Make APIs discoverable by adding them to the private Anypoint Exchange.
- Create public API portals for external developers.

Walkthrough 3-1: Use API Designer to define an API with RAML

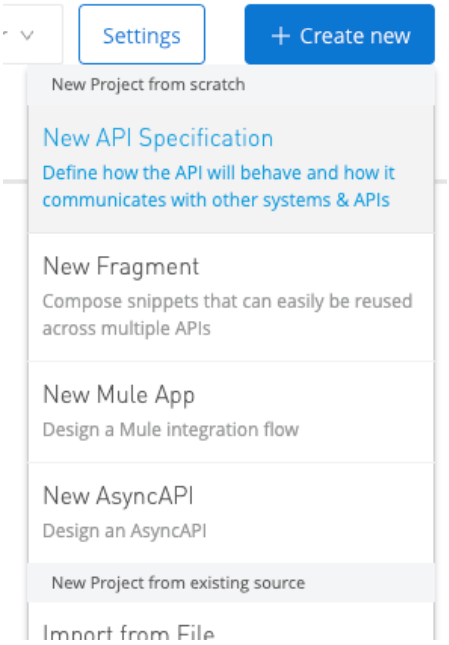
In this walkthrough, you create an API definition with RAML using API Designer. You will:

- Define resources and nested resources.
- Define get and post methods.
- Specify query parameters.



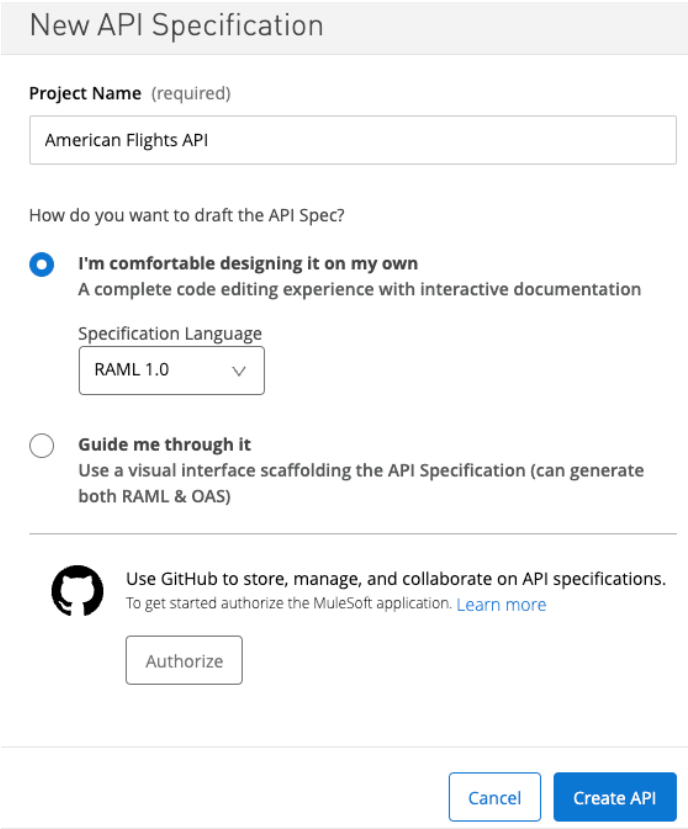
Create a new Design Center project

1. Return to Design Center.
2. Click the Create new button and select New API Specification.

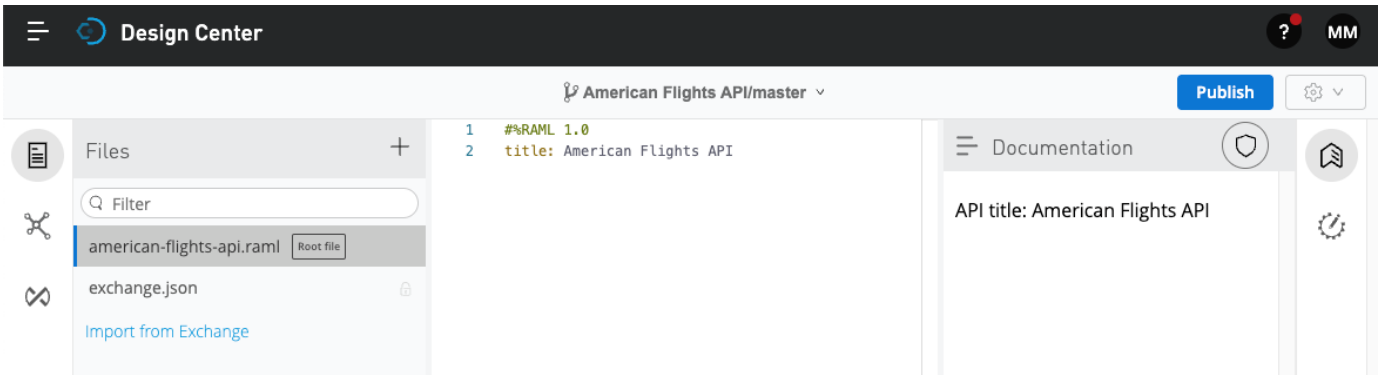


3. In the New API Specification dialog box, set the project name to American Flights API.

4. Ensure I'm comfortable designing it on my own is selected and click Create API; API Designer should open.



5. Review the three sections of API Designer: the file browser, the editor, and the API console.



Add a RAML resource

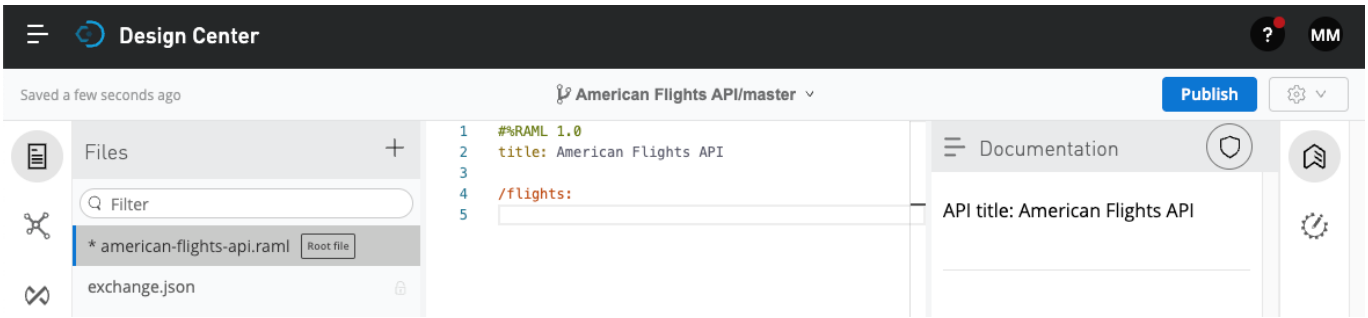
6. In the editor, place the cursor on a new line of code at the end of the file.
7. Add a resource called flights and an additional new line.

```
1  #%RAML 1.0
2  title: American Flights API
3
4  /flights:
5
```

View the API console

8. Look at the API console on the right side of the window; you should see summary information for the API.

Note: If you do not see the API console, click the Documentation icon located in the right column.

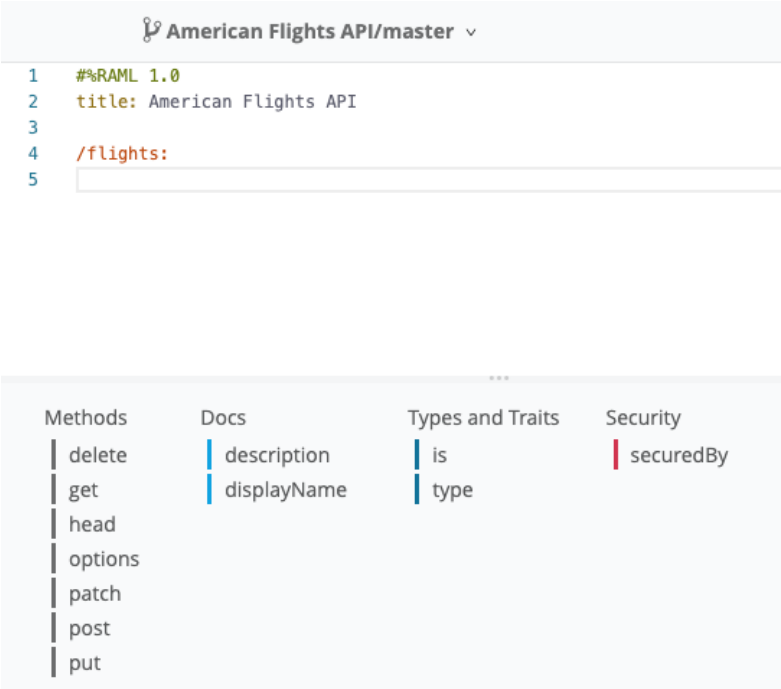


Add RAML methods

9. In the editor, on the last new line of code backspace so you are indented the same amount as the flights resource; look at the contents of the API Designer shelf.

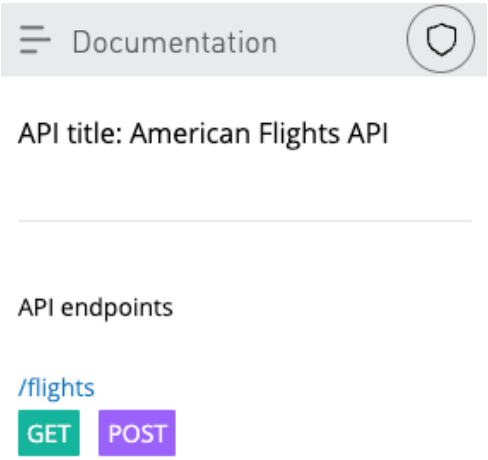
Note: If you don't see the API Designer shelf, it is either minimized or there is an error in your code. To check if it is minimized, go to the bottom of the web browser window and look for an arrow. If you see the arrow, click it to display the shelf.

10. Indent by pressing the Tab key; the contents in the API Designer shelf should change.



11. Click the get method in the shelf.
12. Look at the API console; you should see a GET method for the flights resource.

13. In the editor, backspace so you are indented the same amount as the get method.
14. Click the post method in the shelf.
15. Look at the API console; you should see GET and POST methods for the flights resource.

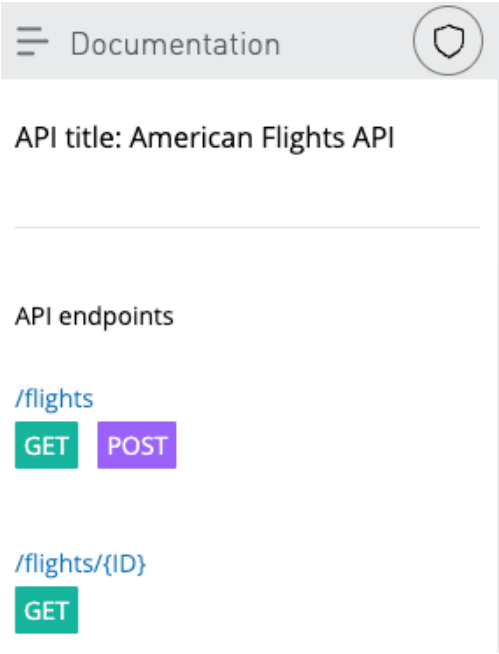


Add a nested RAML resource

16. In the editor, backspace and then go to a new line.
17. Make sure you are still under the flights resource (at the same indentation as the methods).
18. Add a nested resource for a flight with a particular ID.

```
/{ID}:
```

19. Add a get method to this resource.
20. Look at the API console; you should see the nested resource with a GET method.



Add an optional query parameter

21. In the editor, indent under the /flights get method (not the /flights/{ID} get method).
22. In the shelf, click the queryParameters parameter.
23. Add a key named destination.

```
1   #%RAML 1.0
2   title: American Flights API
3
4   /flights:
5     get:
6       queryParameters:
7         destination:
8     post:
9
10    /{ID}:
11      get:
```

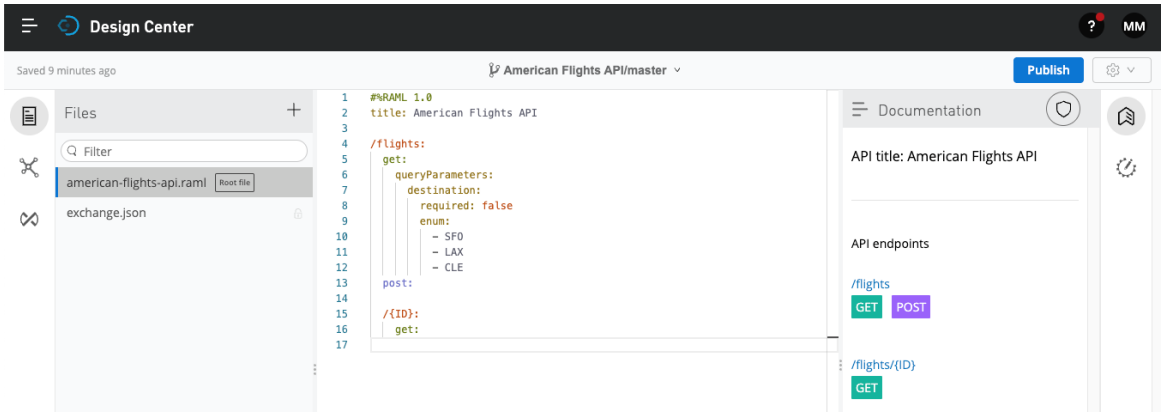
24. Indent under the destination query parameter and look at the possible parameters in the shelf.
25. In the shelf, click the required parameter.
26. In the shelf, click false.
27. Go to a new line of code; you should be at the same indent level as required.
28. In the shelf, click the enum parameter.
29. Set enum to a set of values including SFO, LAX, and CLE.

```
4   /flights:
5     get:
6       queryParameters:
7         destination:
8           required: false
9           enum:
10            - SFO
11            - LAX
12            - CLE
```

Walkthrough 3-2: Use the mocking service to test an API

In this walkthrough, you test the API using the Anypoint Platform mocking service. You will:

- Use the API console to make calls to a mocked API .
- Use a shareable public link to make a call to a mocked API.

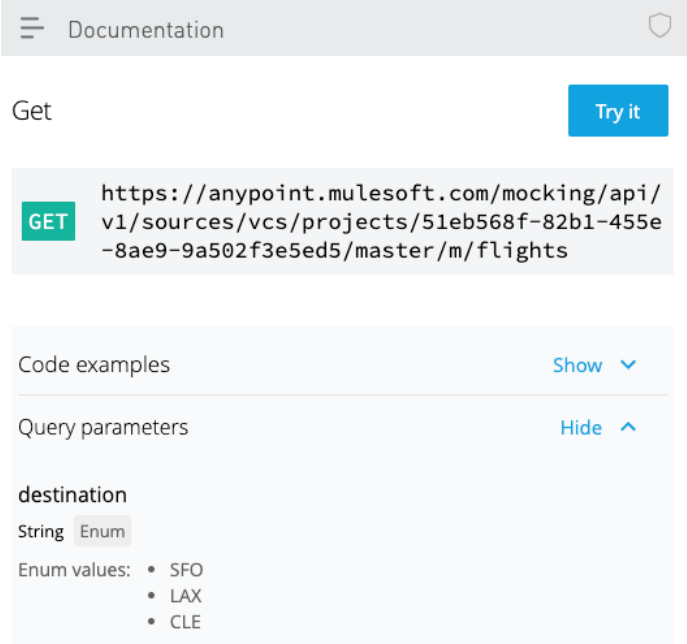


Starting file

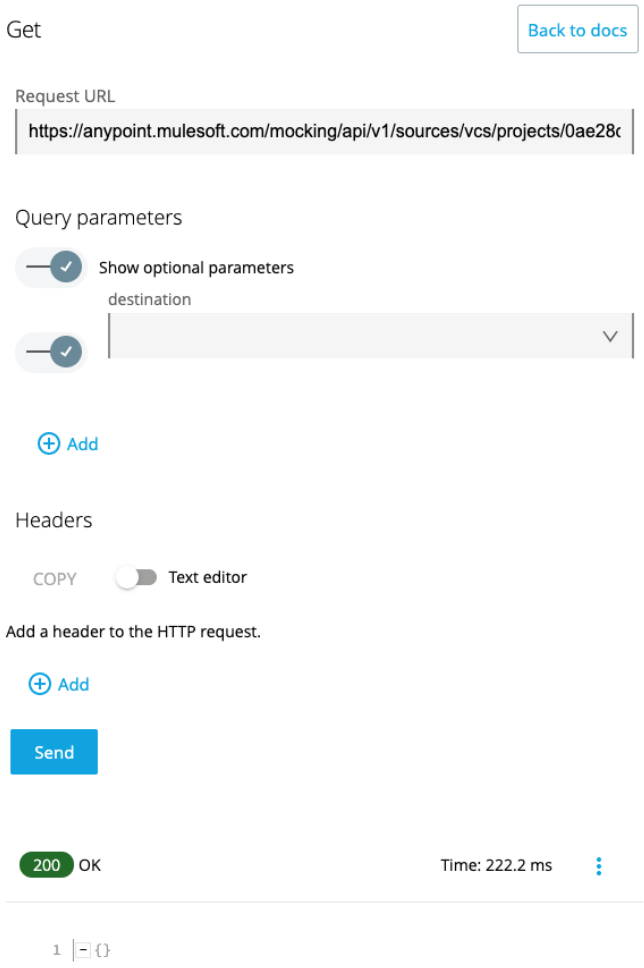
If you did not complete the previous walkthrough, you can get a starting file here. This file is also located in the solutions folder of the student files ZIP located in the Course Resources.

Test the /flights:get resource

1. Return to API Designer.
2. In the API console, click the GET method for the /flights resource.
3. Review the information; you should see a mocking service URL.



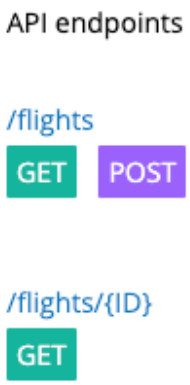
4. Click the Try it button for the flights GET method then click the Send button; you should get a 200 status code and an empty response.



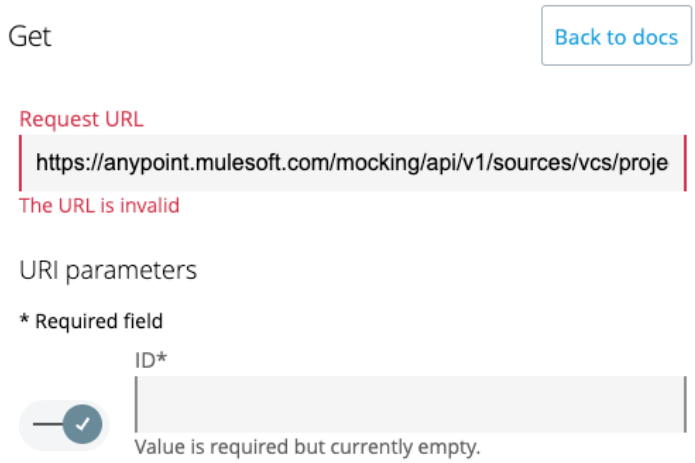
5. In the destination drop-down menu, select SFO and click Send; you should get the same response.

Test the /flights/{ID} resource

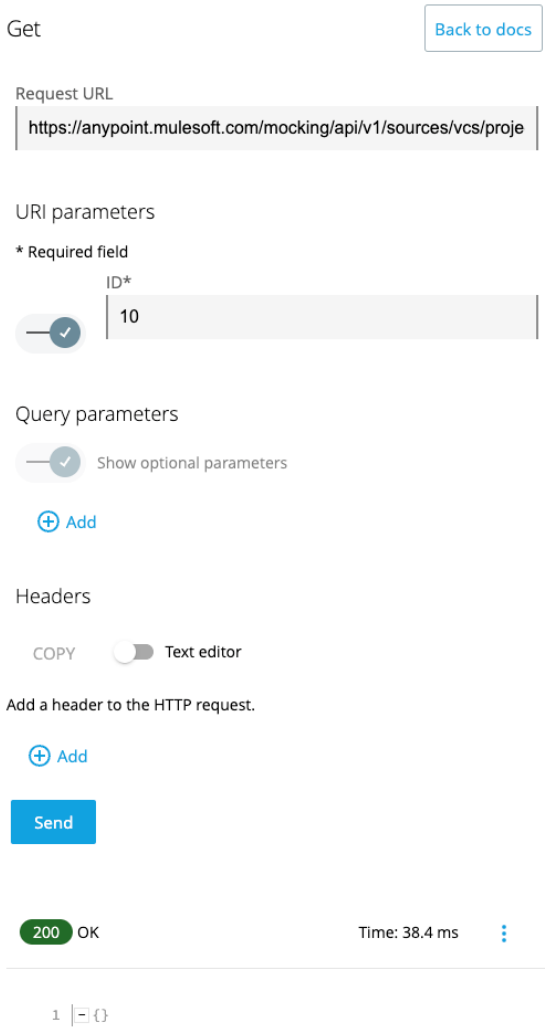
6. Click the menu button located in the upper-left of the API console and select Summary to return to the resource list.



7. Click the GET method for the /{ID} nested resource.
8. Click Try it; you should see a message that the Request URL is invalid.

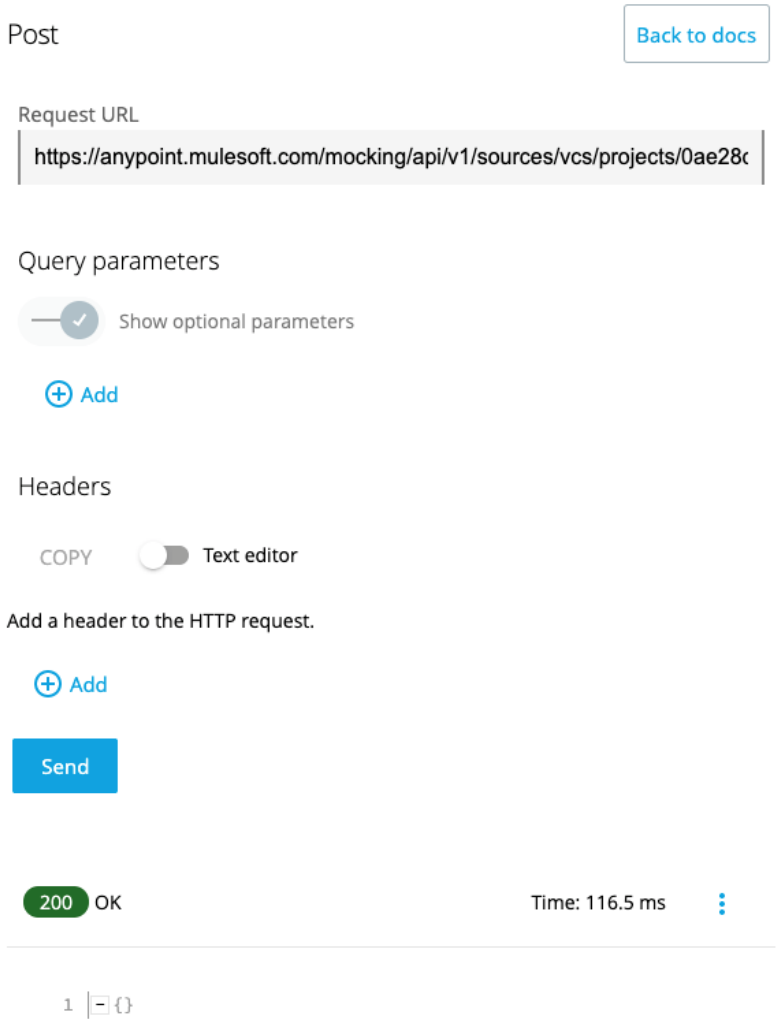


9. In the ID text box, enter a value of 10.
10. Click the Send button.
11. Look at the response; you should get the same 200 status code and an empty response.



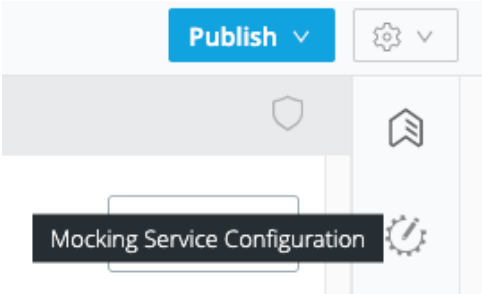
Test the /flights:post resource

12. Use the menu button located in the upper-left of the API console to return to the resource list.
13. Click the POST method.
14. Click the Try it button then click the Send button.
15. Look at the response; you should get the same generic 200 status code response.

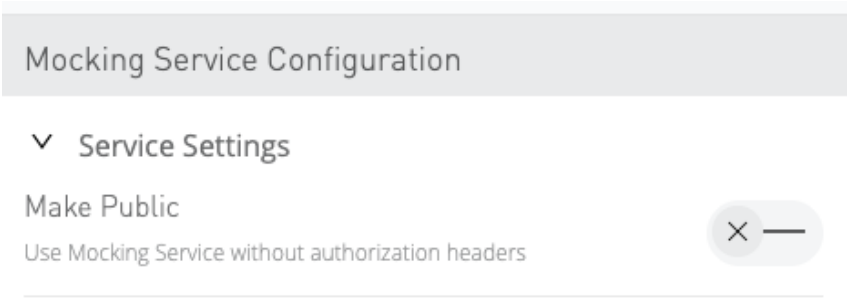


Test the /flights:get resource through a shareable link

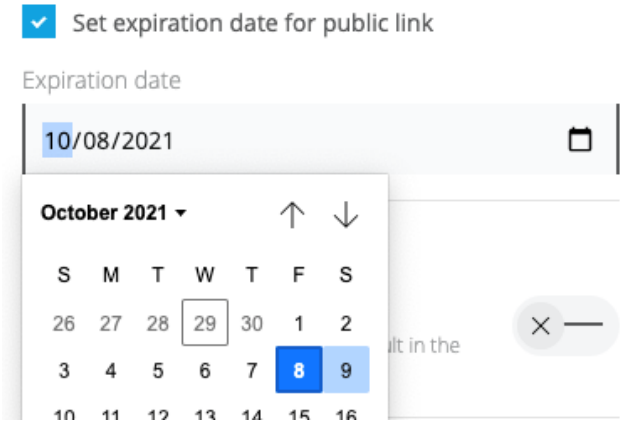
16. Click the Mocking Service Configuration icon in the right column.



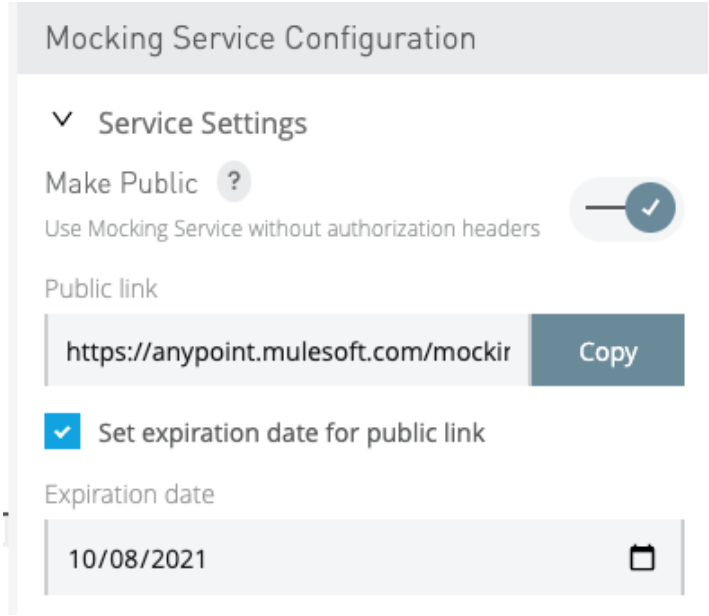
17. Locate the Make Public slider under Service Settings.



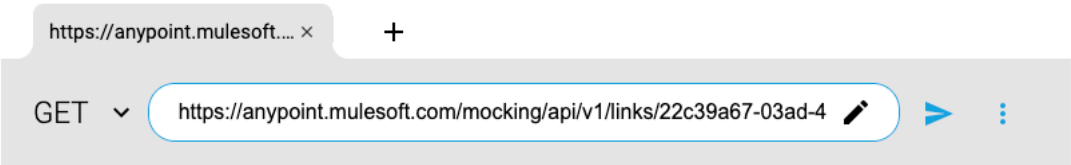
18. Click the slider to turn it on.
19. Select the Set expiration date for public link checkbox.
20. Click the calendar icon in the Expiration date field and select a future day.



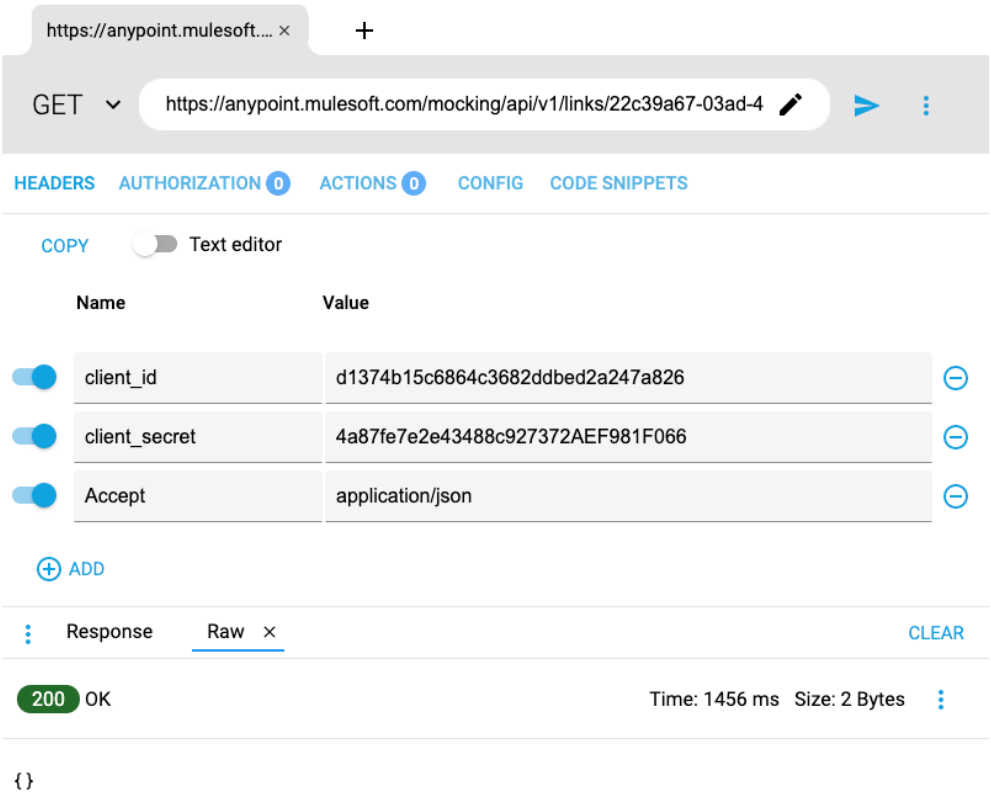
21. Click the Copy button to copy the public link.



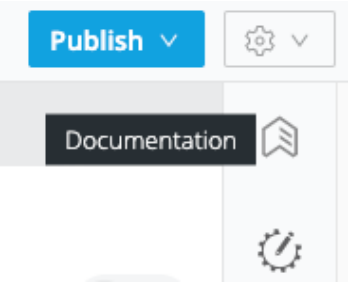
22. Return to Advanced REST Client, paste the copied link and append it with /flights.



23. Add a header called Accept with a value of application/json using the header drop-down menus.
24. Make sure the method is set to GET and click Send; you should get a 200 status code and an empty response.



25. Return to your American Flights API in API Designer.
26. Click the Make Public slider to turn it off.
27. Click the Documentation icon in the right column.

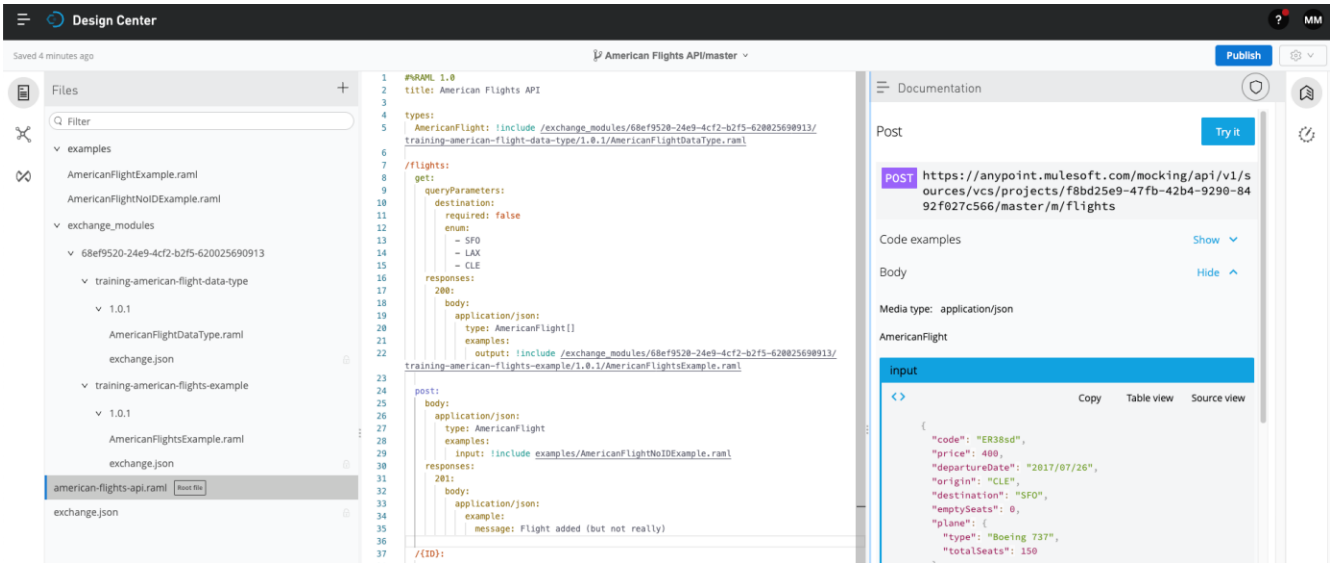


28. Use the menu button to return to the resource list.

Walkthrough 3-3: Add request and response details

In this walkthrough, you add information about each of the methods to the API specification. You will:

- Use API fragments from Exchange.
- Add a data type and use it to define method requests and responses.
- Add example JSON requests and responses.
- Test an API and get example responses.

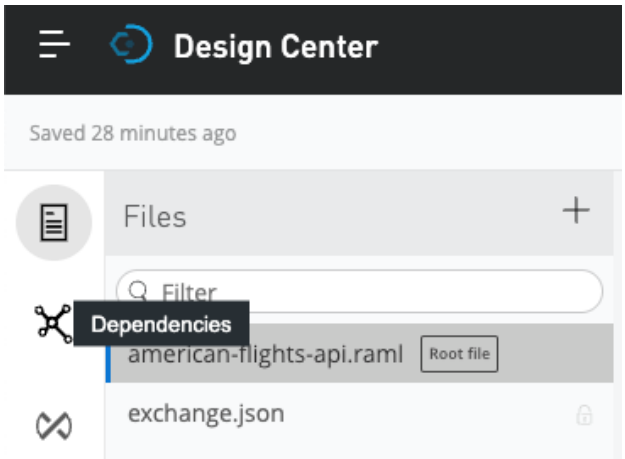


Starting file

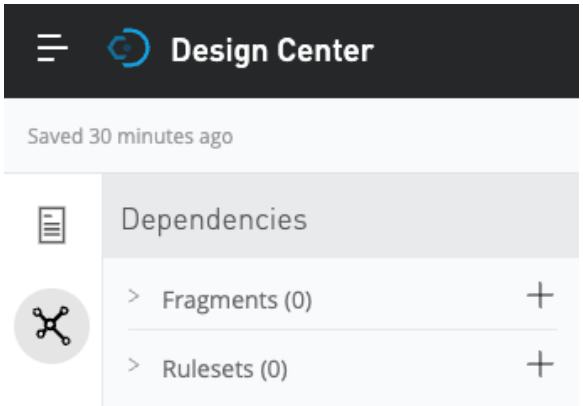
If you did not complete walkthrough 3.1, you can get a starting file here. This file is also located in the solutions folder of the student files ZIP located in the Course Resources.

Add data type and example fragments from Exchange

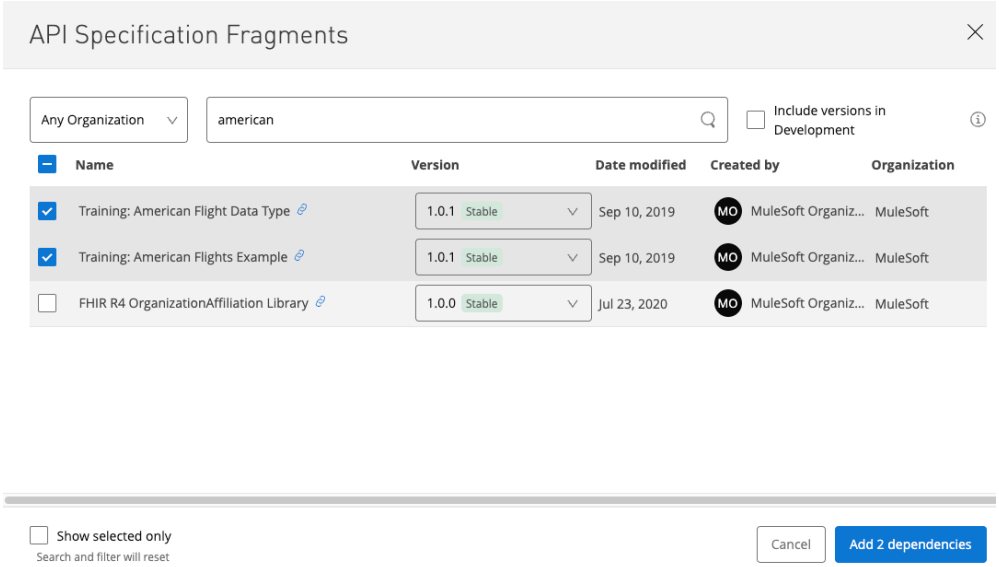
1. Return to API Designer.
2. In the file browser, click the Dependencies button.



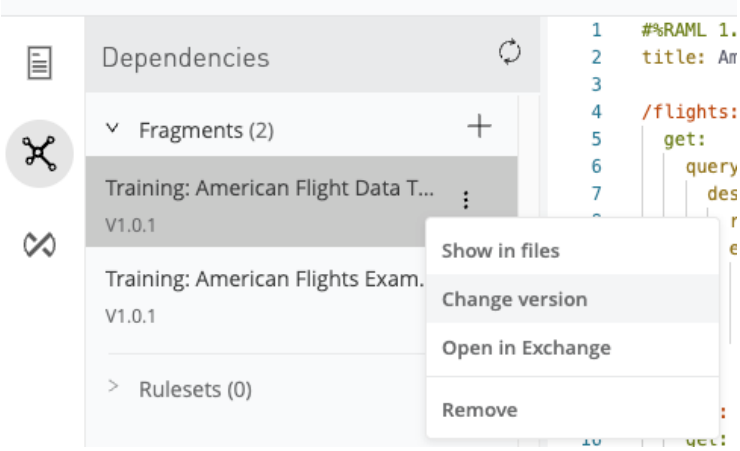
3. Click the Fragments Add button.



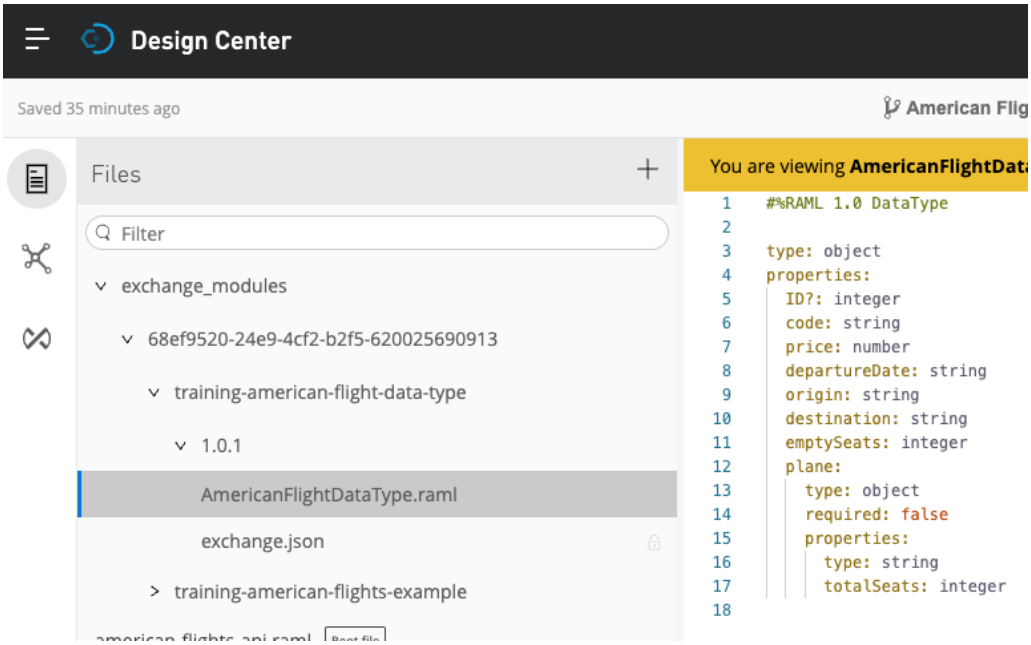
4. In the API Specification Fragments dialog box, select the Training: American Flight Data Type and the Training: American Flights Example.



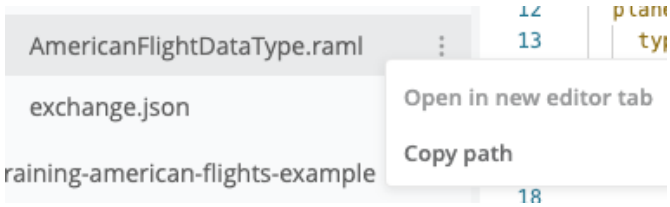
5. Click the Add 2 dependencies button.
6. In the dependencies list, click the Training: American Flight Data Type options menu and review the menu options.



7. Click the Files button (above the Dependencies button).
8. Expand the exchange_modules section until you see AmericanFlightDataType.raml.
9. Click AmericanFlightDataType.raml and review the code.



10. In the file browser, click the options menu button next to AmericanFlightDataType.raml and select Copy path.



Define an AmericanFlight data type for the API

11. Return to american-flights-api.raml.
12. Near the top of the code above the /flights resource, add a types element.
13. Indent under types and add a type called AmericanFlight.
14. Add the !include keyword and then paste the path you copied.

Note: You can also add the path by navigating through the exchange_modules folder in the shelf.

```
#%RAML 1.0
title: American Flights API

types:
  AmericanFlight: !include /exchange_modules/c1b0acdc-127d-4277-be03-2261ff15b14a/training-a

/flights:
  get:
```

Specify the /flights:get method to return an array of AmericanFlight objects

15. Go to a new line of code at the end of the /flights get method and indent to the same level as queryParameters.
16. In the shelf, click responses > 200 > body > application/json > type > AmericanFlight.

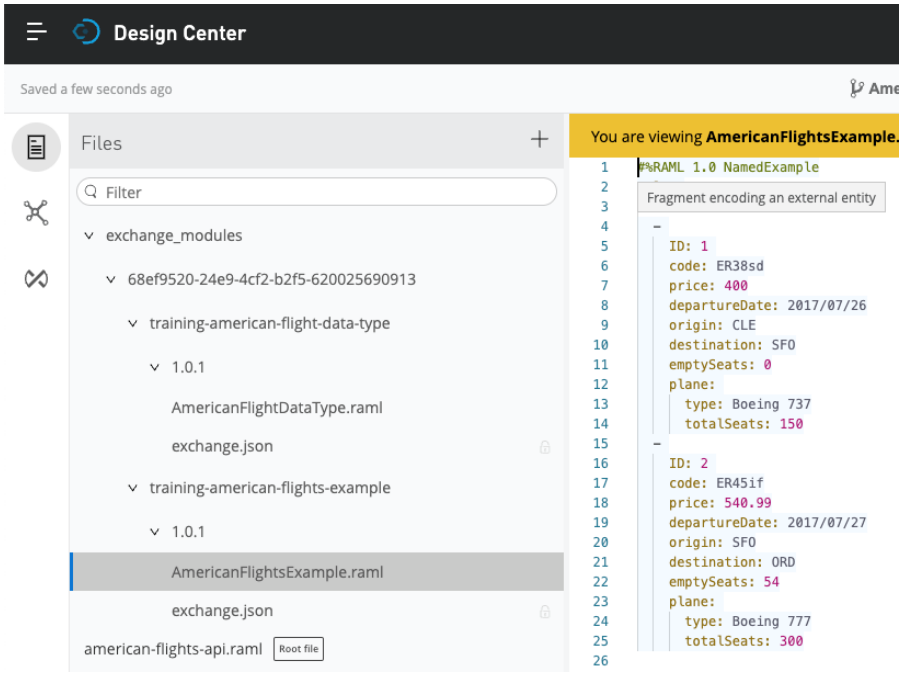
```
7  /flights:
8    get:
9      queryParameters:
10       destination:
11         required: false
12         enum:
13           - SFO
14           - LAX
15           - CLE
16     responses:
17       200:
18         body:
19           application/json:
20             type: AmericanFlight
```

17. Set the type to be an array of AmericanFlight objects: AmericanFlight[].

```
16     responses:
17       200:
18         body:
19           application/json:
20             type: AmericanFlight[]
```

Add an example response for the /flights:get method

18. In the file browser, locate AmericanFlightsExample.raml in exchange_modules and review the code.



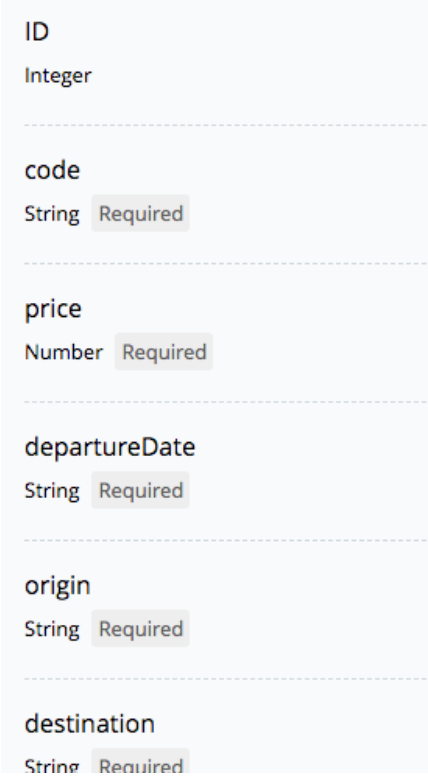
19. In the file browser, click the options menu next to AmericanFlightsExample.raml and select Copy path.
20. Return to american-flights-api.raml.
21. In the editor, go to a new line after the type declaration in the /flights:get 200 response (at the same indentation as type).
22. In the shelf, click examples.
23. Add a key called output.
24. Add the !include keyword and then paste the path you copied.

Note: You can also add the path by navigating through the exchange_modules folder in the shelf.

```
16      responses:
17        200:
18          body:
19            application/json:
20              type: AmericanFlight[]
21              examples:
22                output: !include /exchange_modules/c1b0acdc-127d-42
23
24    post:
```

Review and test the /flights:get resource in the API console

25. In the API console, click the /flights:get method.
26. In the response information, look at the type information.



27. In the response information, ensure you see the example array of AmericanFlight objects.

output

< > Copy Table view Source view


```
[
  {
    "ID": 1,
    "code": "ER38sd",
    "price": 400,
    "departureDate": "2017/07/26",
    "origin": "CLE",
    "destination": "SFO",
    "emptySeats": 0,
    "plane": {
      "type": "Boeing 737",
      "totalSeats": 150
    }
  },
  {
    "ID": 2,
    "code": "ER45if",
    "price": 540.99,
    "departureDate": "2017/07/27",
    "origin": "SFO",
    "destination": "ORD",
    "emptySeats": 54,
    "plane": {
      "type": "Boeing 777",
      "totalSeats": 300
    }
  }
]
```


28. Click the Try it button and click Send; you should now see the example response with two flights.

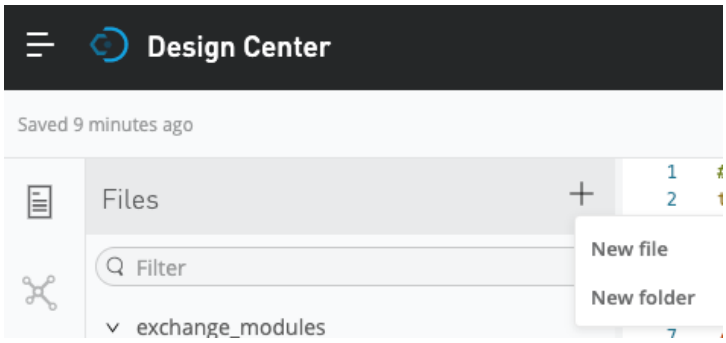
Specify the /{ID}:get method to return an AmericanFlight object

29. In the editor, indent under the /{ID} resource get method.
30. In the shelf, click responses > 200 > body > application/json > type > AmericanFlight.

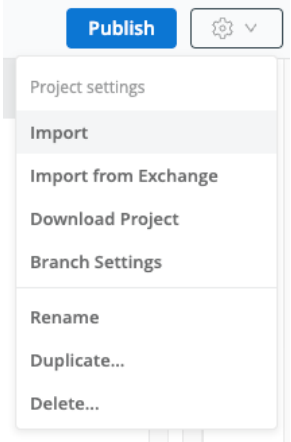
```
26    /{ID}:
27      get:
28        responses:
29          200:
30            body:
31              application/json:
32                type: AmericanFlight
```

Define an example response for the /{ID}:get method in a new folder

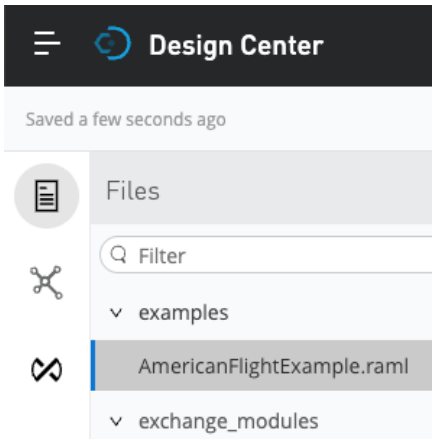
31. In the file browser, click the add button and select New folder.



32. In the Add new folder dialog box, set the name to examples and click Create.
33. In the settings drop-down menu in the upper-right corner, select Import.



34. In the Import file(s) dialog box, leave File or ZIP selected and click the Choose file button.
35. Navigate to your student files and select the AmericanFlightExample.raml file in the resources/examples folder and click Open.
36. In the Import file(s) dialog box, click Import.
37. In the file browser, click AmericanFlightExample.raml and review the code.
38. Drag AmericanFlightExample.raml into the examples folder.



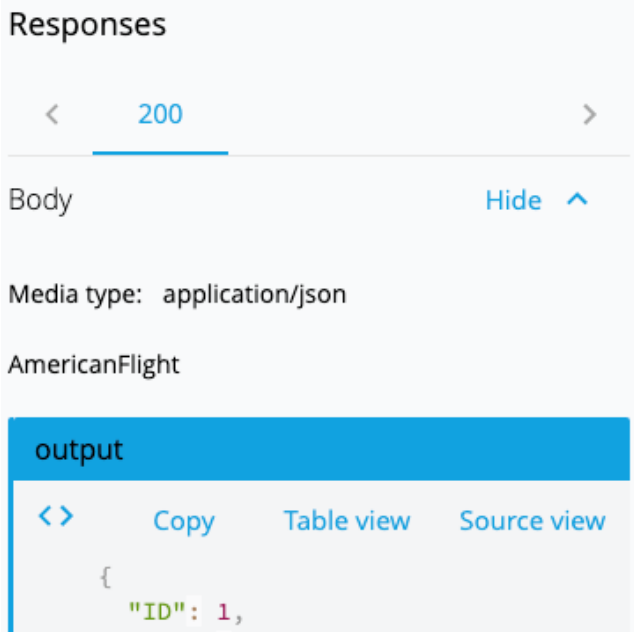
Add an example response for the /{ID}:get method

39. Return to american-flights-api.raml.
40. In the editor, go to a new line after the type declaration in {ID}:/get (at the same indentation).
41. In the shelf, click examples.
42. Add a key called output.
43. Add an include statement and include the example in examples/AmericanFlightExample.raml.

```
26  /{ID}:
27    get:
28      responses:
29        200:
30          body:
31            application/json:
32              type: AmericanFlight
33              examples:
34                output: !include examples/AmericanFlightExample.raml
```

Review and test the /{ID}:get resource in the API console

44. In the API console, return to the /{ID}:get method; you should now see the response will be of type AmericanFlight.



45. In the response information, ensure you see the example AmericanFlightExample data.

46. Click the Try it button, enter an ID, and click Send; you should now see the example flight data returned.

200 OK

```
1  {
2    "ID": 1,
3    "code": "ER38sd",
4    "price": 400,
5    "departureDate": "2017/07/26",
6    "origin": "CLE",
7    "destination": "SFO",
8    "emptySeats": 0,
9    "plane": {
10     "type": "Boeing 737",
11     "totalSeats": 150
12   }
13 }
```

Specify the /flights:post method request to require an AmericanFlight object

47. In the editor, indent under the /flights post method.
48. In the shelf, click body > application/json > type > AmericanFlight.

```
24  post:
25    body:
26      application/json:
27        type: AmericanFlight
```

Define an example request body for the /flights:post method

49. Return to AmericanFlightExample.raml and copy all the code.
50. In the file browser, click the add button next to the examples folder and select New file.
51. In the Add new file dialog box, set the following values:

- Version: RAML 1.0
- Type: Example
- File name: AmericanFlightNoIDExample.raml

52. Click Create.
53. Delete any code in the new file and then paste the code you copied.

54. Delete the line of code containing the ID.

```
#%RAML 1.0 NamedExample
value:
  code: ER38sd
  price: 400
  departureDate: 2017/07/26
  origin: CLE
  destination: SFO
  emptySeats: 0
  plane:
    type: Boeing 737
    totalSeats: 150
```

55. Return to american-flights-api.raml.
56. In the post method, go to a new line under type and add an examples element.
57. Add a key called input.
58. Add an include statement and include the example in examples/AmericanFlightNoIDExample.raml.

```
post:
  body:
    application/json:
      type: AmericanFlight
      examples:
        input: !include examples/AmericanFlightNoIDExample.raml
```

Specify an example response for the /flights:post method

59. Go to a new line of code at the end of the /flights:post method and indent to the same level as body.
60. Add a 201 response body of type application/json.

```
post:
  body:
    application/json:
      type: AmericanFlight
      examples:
        input: !include examples/AmericanFlightNoIDExample.raml
  responses:
    201:
      body:
        application/json:
```

61. In the shelf, click example.

62. Indented under example, add a message property equal to the string: Flight added (but not really).

```
24  post:
25    body:
26      application/json:
27        type: AmericanFlight
28        examples:
29          input: !include examples/AmericanFlightNoIDExample.raml
30    responses:
31      201:
32        body:
33          application/json:
34            example:
35              message: Flight added (but not really)
```

Review and test the /flights:post resource in the API console

63. In the API console, return to the /flights:post method.
64. Look at the request information; you should now see information about the body - that it is type AmericanFlight and it has an example.

Body Hide

Media type: application/json

AmericanFlight

input

Copy Table view Source view

```
{
  "code": "ER38sd",
  "price": 400,
  "departureDate": "2017/07/26",
  "origin": "CLE",
  "destination": "SFO",
  "emptySeats": 0,
  "plane": {
    "type": "Boeing 737",
    "totalSeats": 150
  }
}
```

65. Click the Try it button; in the Body section you should see the example request body.

Body

Format JSON | Minify JSON | Copy

```
{
  "code": "ER38sd",
  "price": 400,
  "departureDate": "2017/07/26",
  "origin": "CLE",
  "destination": "SFO",
  "emptySeats": 0,
  "plane": {
    "type": "Boeing 737",
    "totalSeats": 150
  }
}
```

Send

66. Click the Send button; you should now get a 201 response with the example message.

201 Created Time: 751.7 ms

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

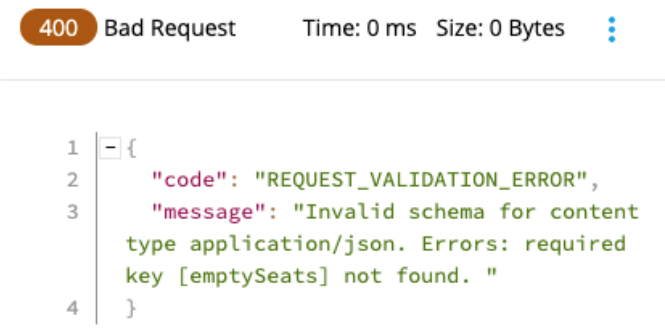
67. In the body, remove the emptySeats property.

Body

Format JSON | Minify JSON | Copy

```
{
  "code": "ER38sd",
  "price": 400,
  "departureDate": "2017/07/26",
  "origin": "CLE",
  "destination": "SFO",
  "plane": {
    "type": "Boeing 737",
    "totalSeats": 150
  }
}
```

68. Click Send again; you should get a 400 Bad Request response and a message that the emptySeats key is required.

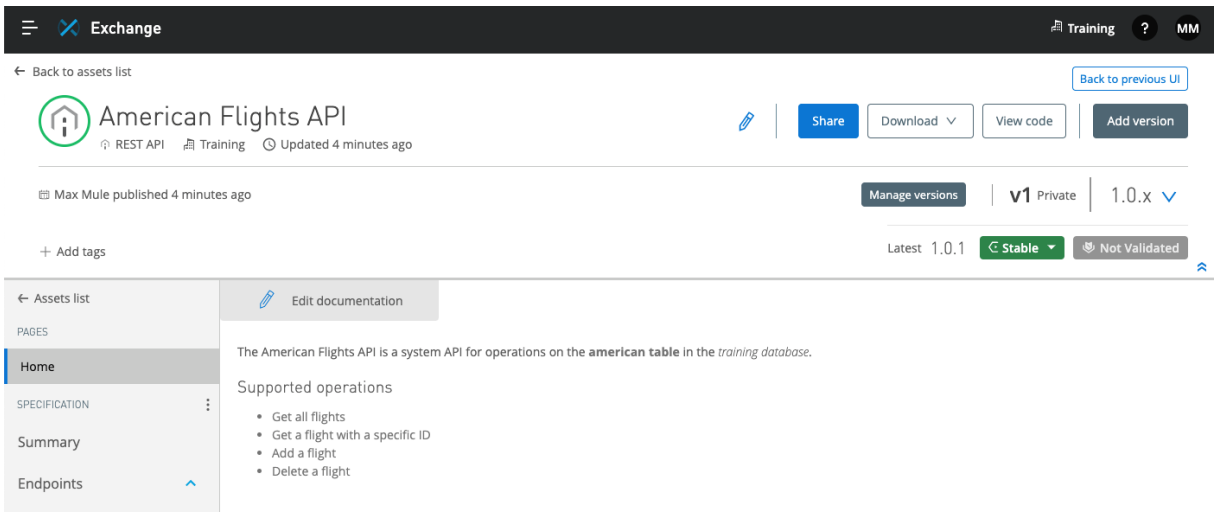


69. Add the emptySeats property back to the body.

Walkthrough 3-4: Add an API to Anypoint Exchange

In this walkthrough, you make an API discoverable by adding it to Anypoint Exchange. You will:

- Publish an API to Exchange from API Designer.
- Review an auto-generated API portal in Exchange and test the API.
- Add information about an API to its API portal.
- Create and publish a new API version to Exchange.

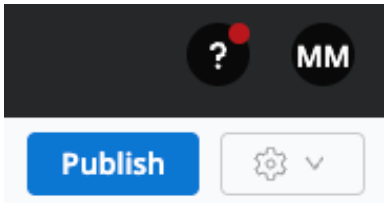


Starting file

If you did not complete the previous walkthrough, you can get a starting file here. This file is also located in the solutions folder of the student files ZIP located in the Course Resources.

Publish the API to Anypoint Exchange from API Designer

1. In API Designer, click the Publish button.



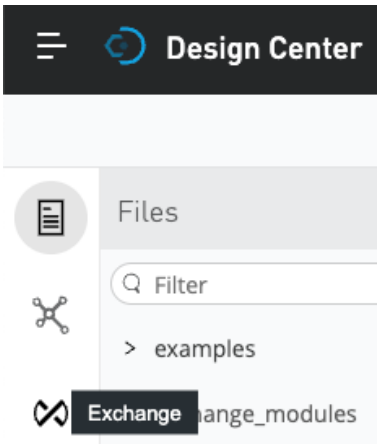
2. In the Publishing to Exchange dialog box, ensure the following values are set:

- Asset version: 1.0.0
- API version: v1
- LifeCycle State: Stable

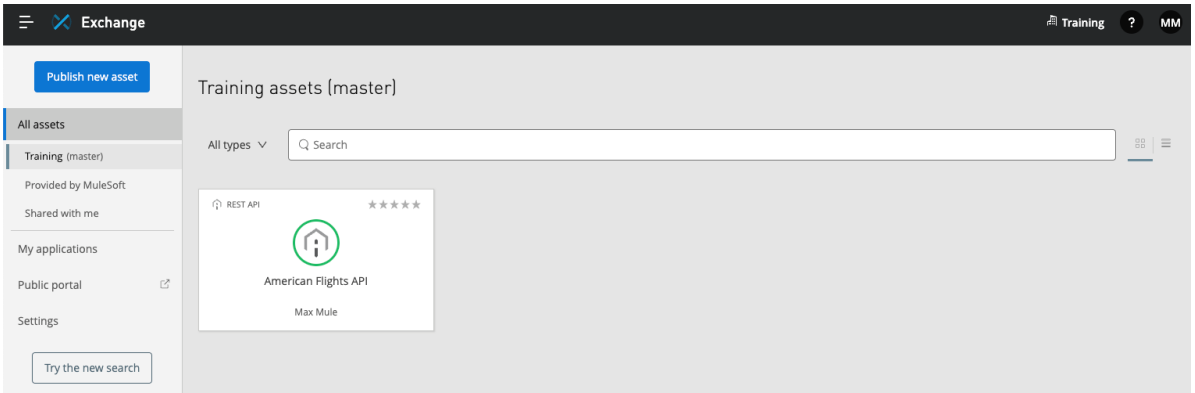
3. Click the Publish to Exchange button.
4. Wait for the API to publish then in the resultant dialog box, click Close.

Locate your API in Anypoint Exchange

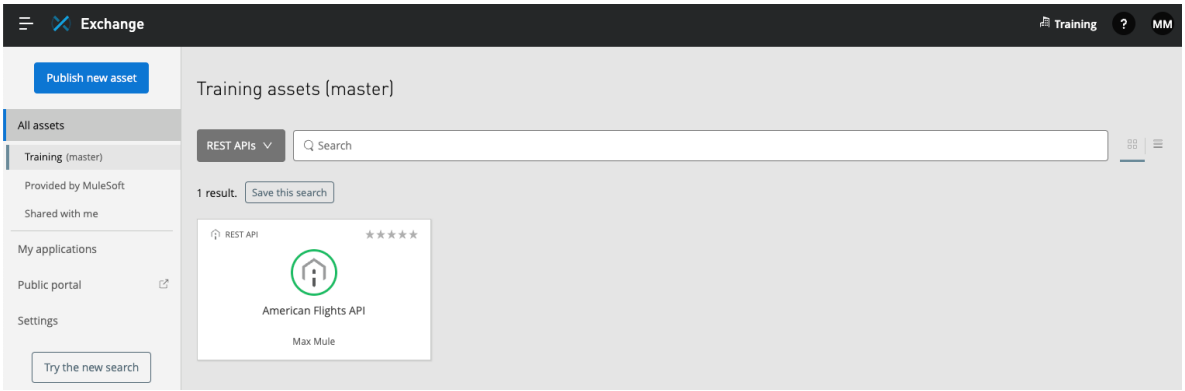
5. In the file browser, click the Exchange button.



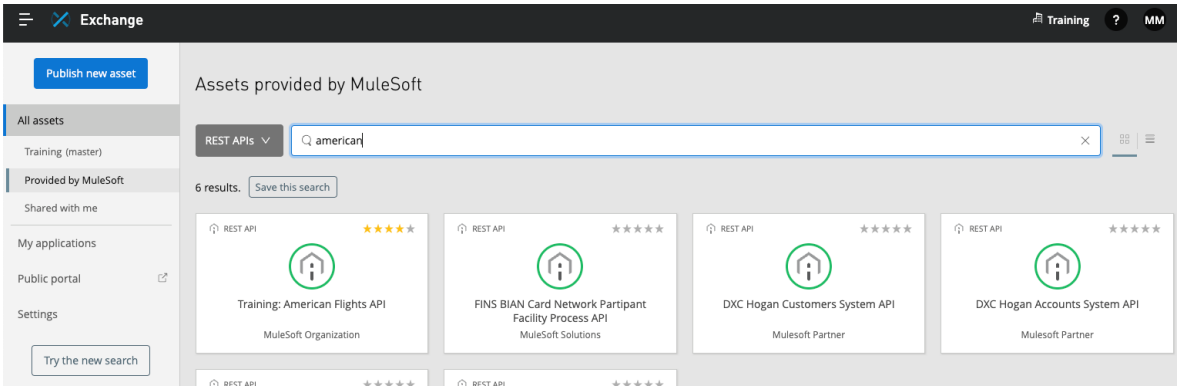
6. Click the Open in Exchange link; the auto-generated portal for your American Flights API opens in a new browser tab.
7. Select the Exchange link near the upper-left corner of the page; you should see the asset listing for your API.



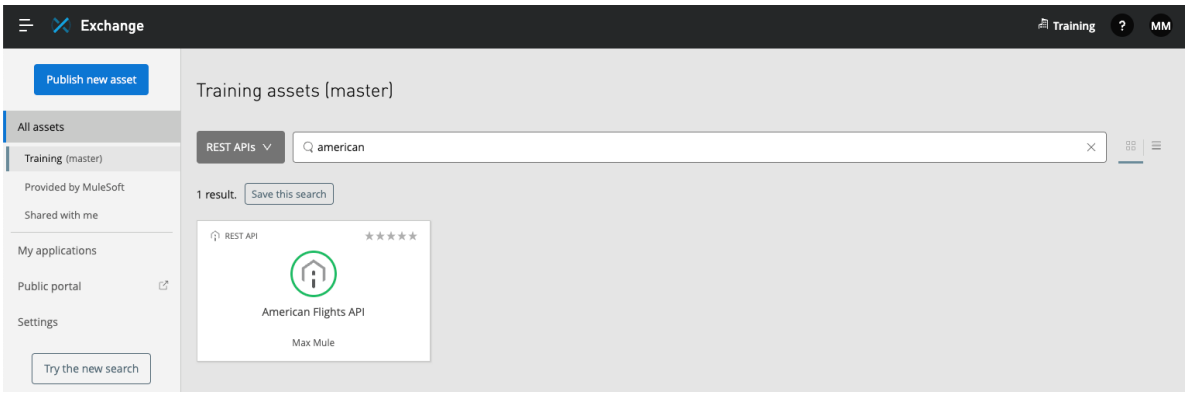
8. In the types drop-down menu, select REST APIs; you should still see your API.



9. In the left-side navigation, select Provided by MuleSoft; you should not find your American Flights API in the public Exchange (just the Training: American Flights API).

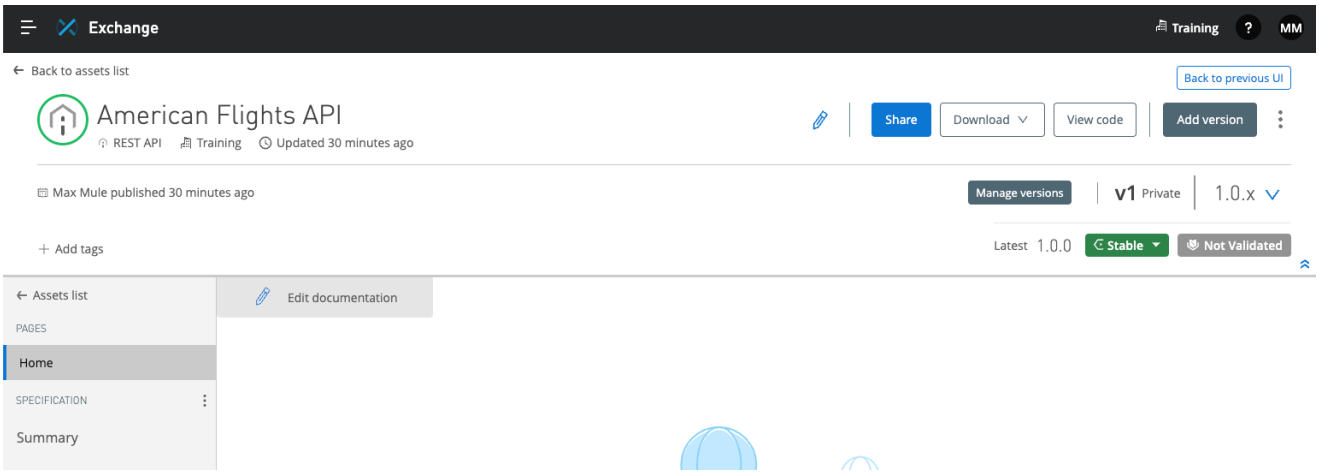


10. In the left-side navigation, select the name of your organization (Training in the screenshots); you should see your American Flights API in your private Exchange.



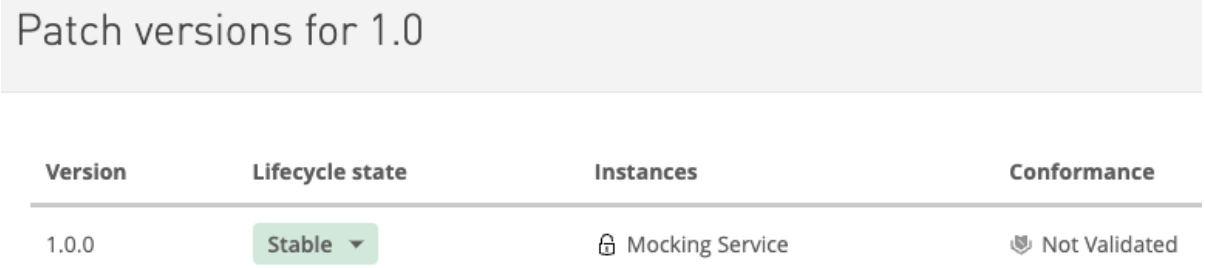
Review the auto-generated API portal

11. Click the American Flights API.
12. Review the page; you should see that as the creator of this API, you can perform management operations such as editing, sharing, and downloading this version.

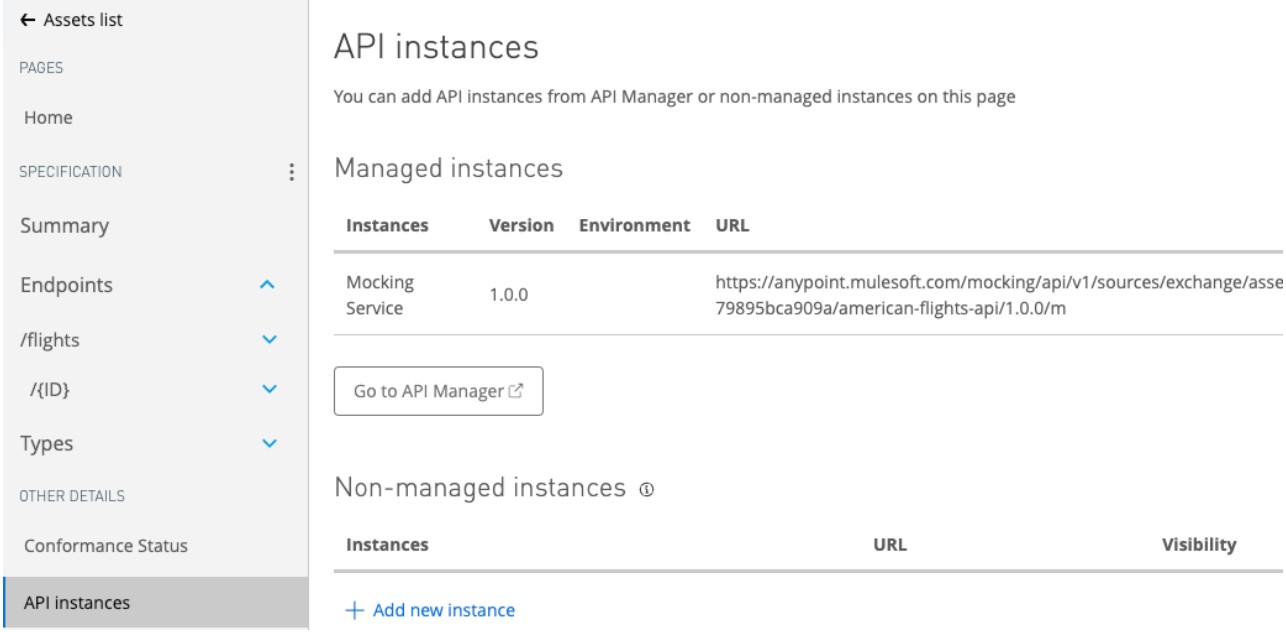


13. Locate the API version (v1) and minor version (1.0.x) near the upper-right corner of the page.

14. Click the Manage versions button.
15. In the Patch versions for 1.0 dialog box, you should see one version (1.0.0) of this API specification has been published for this minor version and there is one API instance for it that uses the mocking service and has a stable lifecycle state.

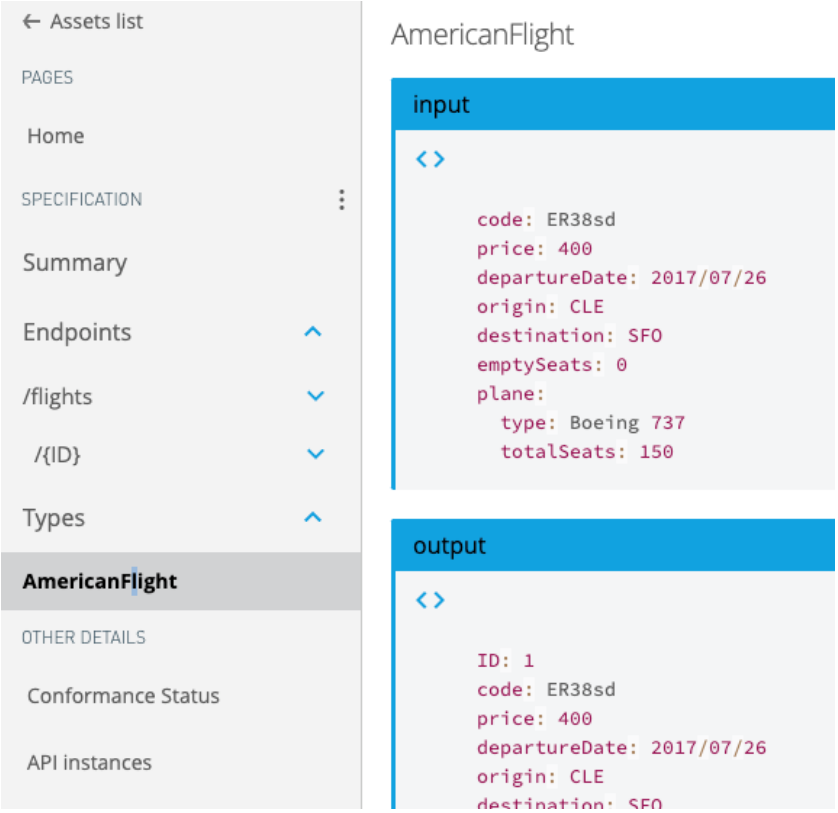


16. Click Close.
17. In the left-side navigation, select API instances; you should see information for the API instance generated from the API specification using the mocking service.



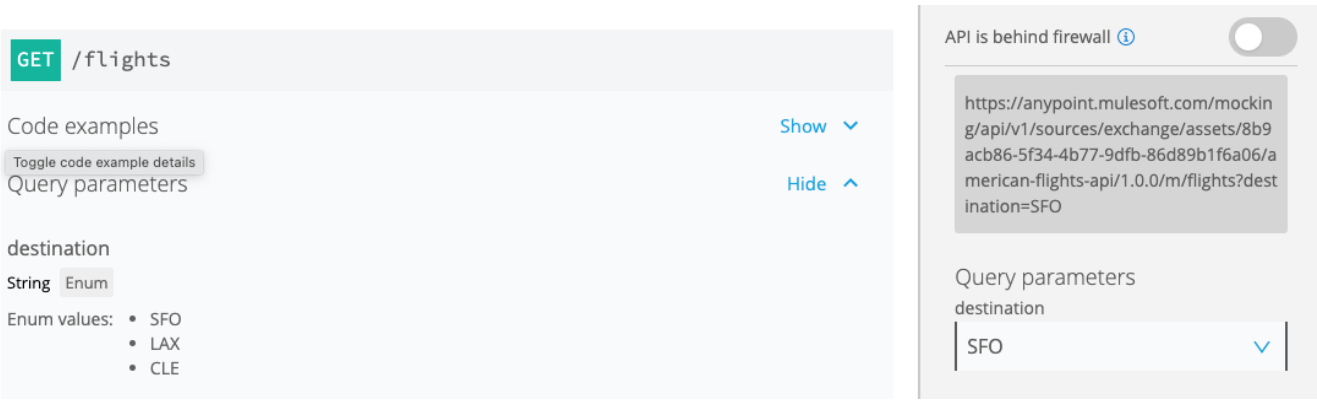
18. In the left-side navigation, expand Types.

19. Select AmericanFlight and review the information.

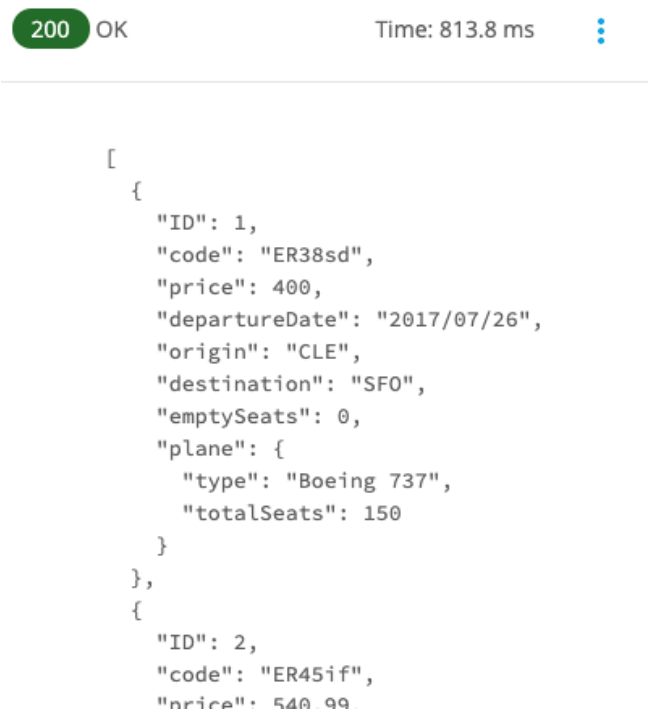


Test the API in its API portal in Exchange

20. In the left-side navigation, select the /flights GET method; you should now see the API console on the right side of the page.
21. Click the destination drop-down and select a value.

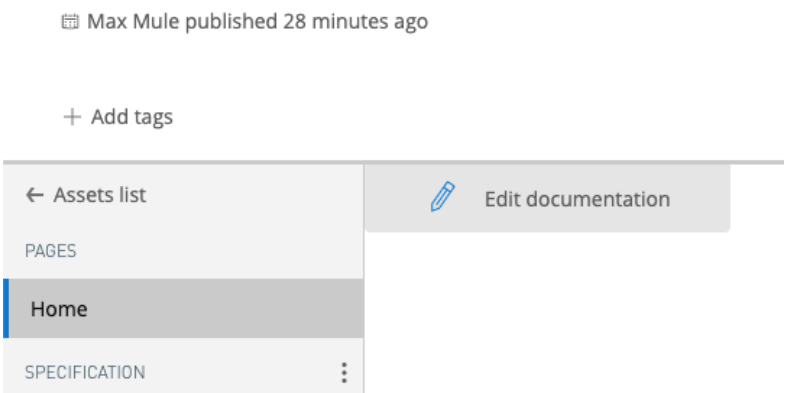


22. Click Send; you should get a 200 response and the example data displayed – just as you did in the API console in API Designer.



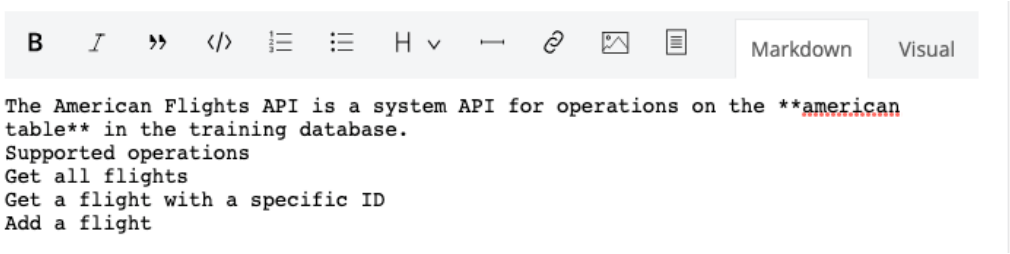
Add information about the API

23. In the left-side navigation, select Home.
24. Click the Edit documentation button for the API.

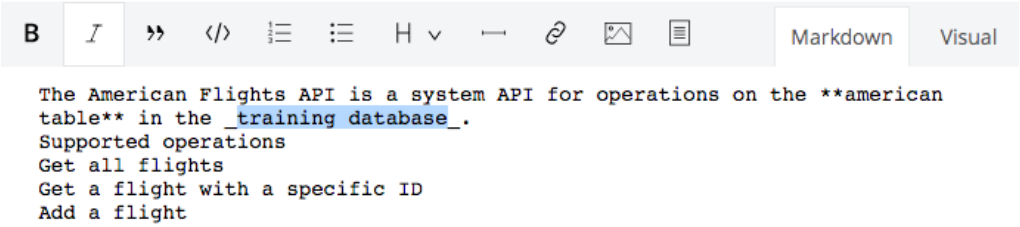


25. Return to the course snippets.txt file and copy the text for the American Flights API description text.
26. Return to the editor in Anypoint Exchange and paste the content.

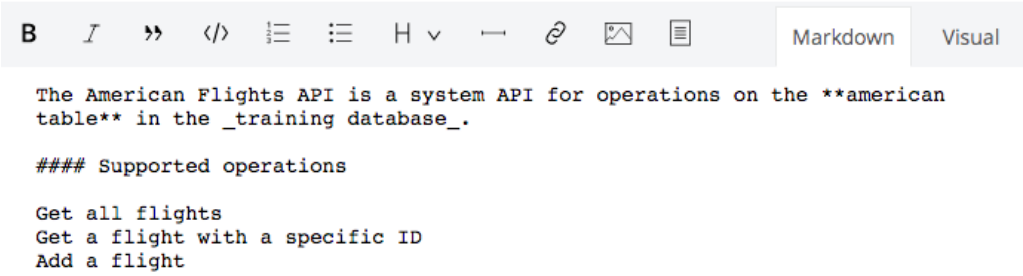
27. Select the words american table and click the strong button (the B).



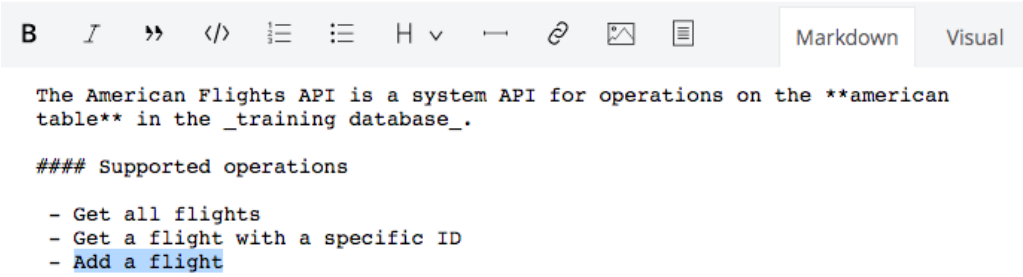
28. Select the words training database and click the emphasis button (the I).



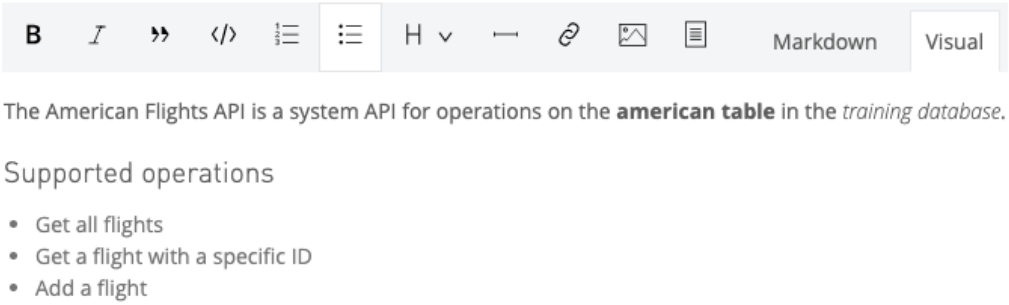
29. Select the words Supported operations and select Heading 4 from the heading drop-down menu (the H).



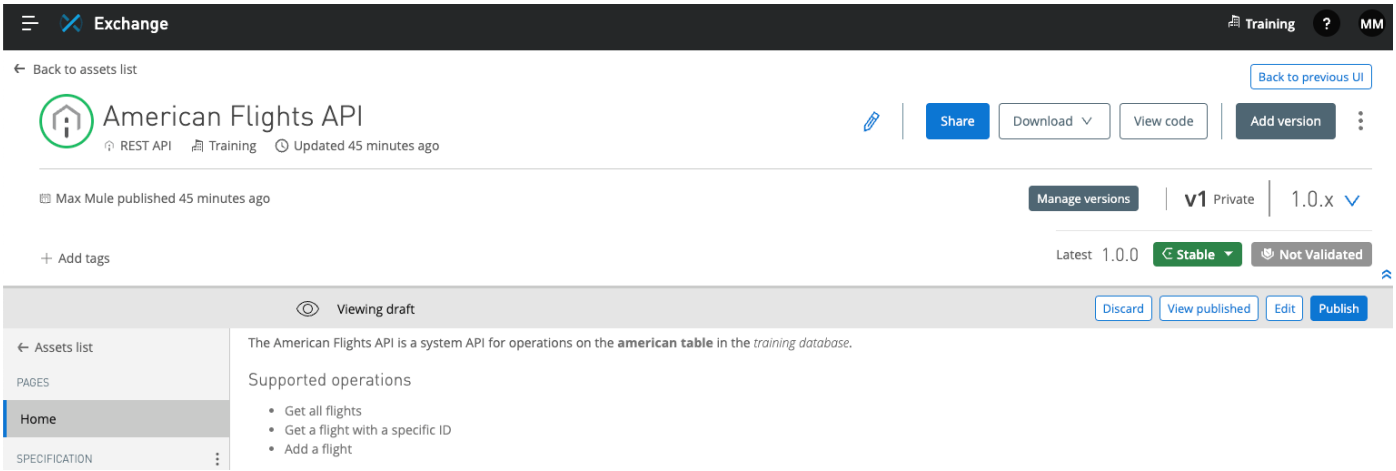
30. Select Get all flights and click the bulleted list button.
31. Repeat for the other operations.



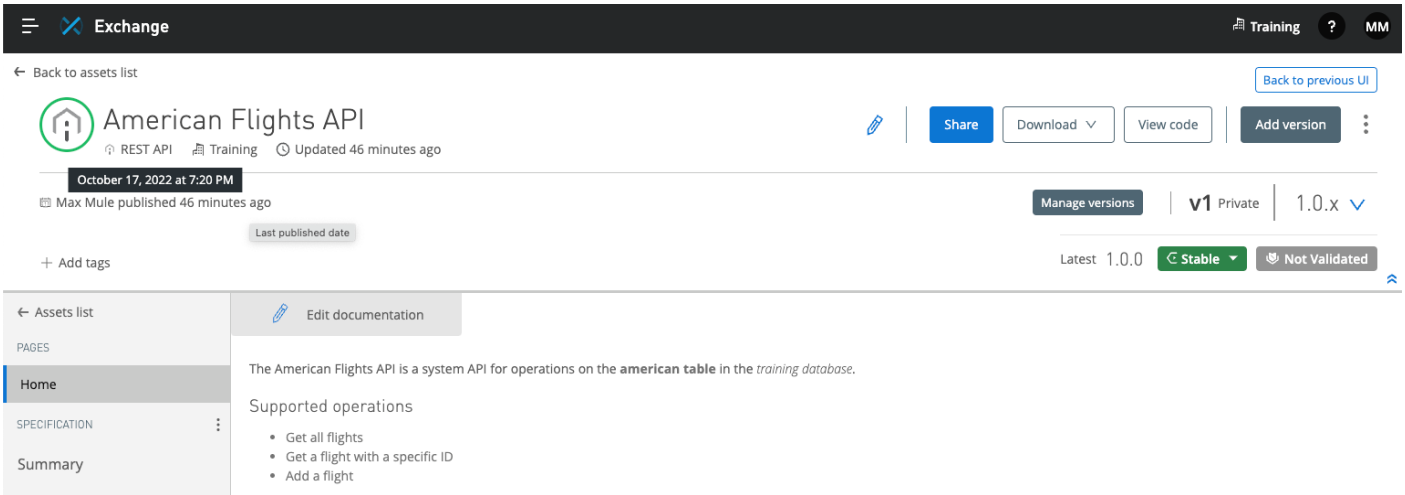
32. Select the Visual tab in the editor toolbar and view the rendered markdown.



33. Click the Save button; you should now see buttons to discard, view, edit, or publish the draft.



34. Click the Publish button; you should now see the new information about the API in the API portal.



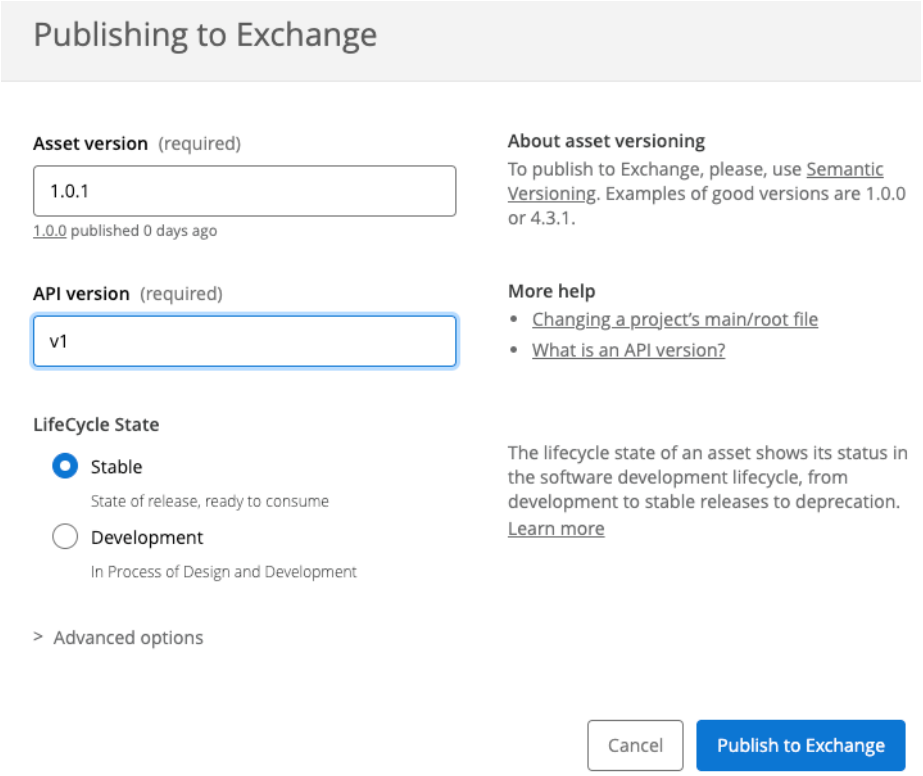
35. Close the browser tab with Exchange.

Modify the API and publish the new version to Exchange

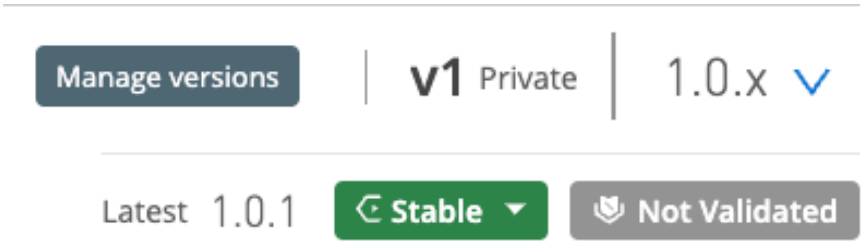
36. Return to the browswer tab with your American Flights API in API Designer.
37. Return to the course snippets.txt file and copy the American Flights API - /{ID} DELETE method.
38. Return to american-flights-api.raml and paste the code after the {ID}/get method.
39. Fix the indentation if necessary.
40. Review the code.

```
/{ID}:
  get:
    responses:
      200:
        body:
          application/json:
            type: AmericanFlight
            examples:
              output: !include examples/AmericanFlightExample.raml

  delete:
    responses:
      200:
        body:
          application/json:
            example:
              message: Flight deleted (but not really)
```

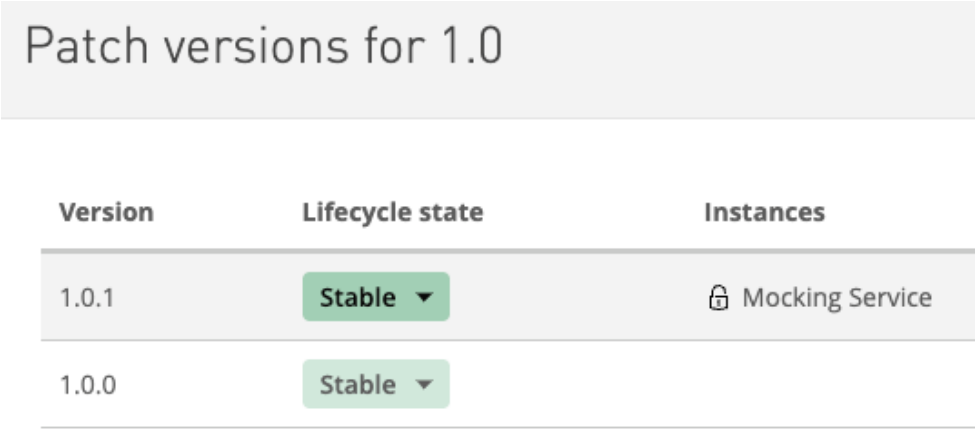
41. Click the Publish button.
42. In the Publishing to Exchange dialog box, examine the asset version then click the Publish to Exchange button.



43. Wait for the API to publish then in the resultant dialog box, click Close.
44. Navigate to the portal for your American Flights API using the Exchange button in the file browser.
45. Click the Manage versions button.

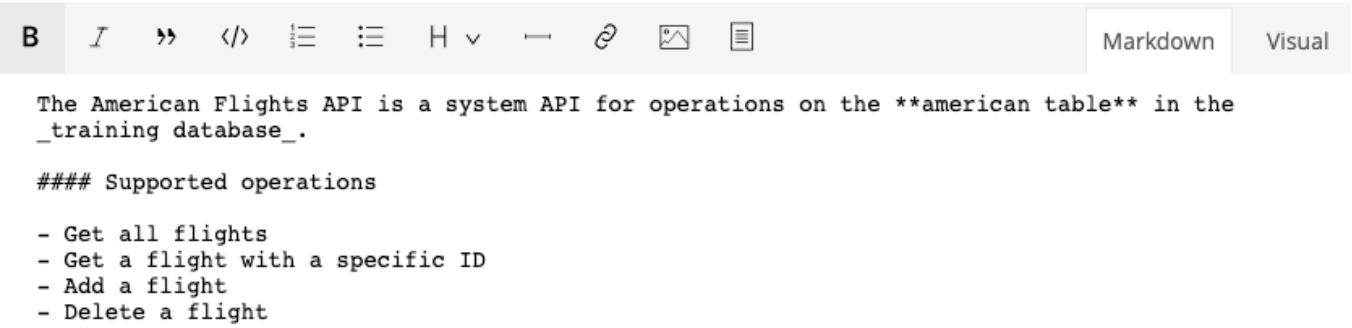


46. In the Patch versions for 1.0 dialog box, you should see both asset versions of the API listed with an associated API instance using the mocking service for the latest version.

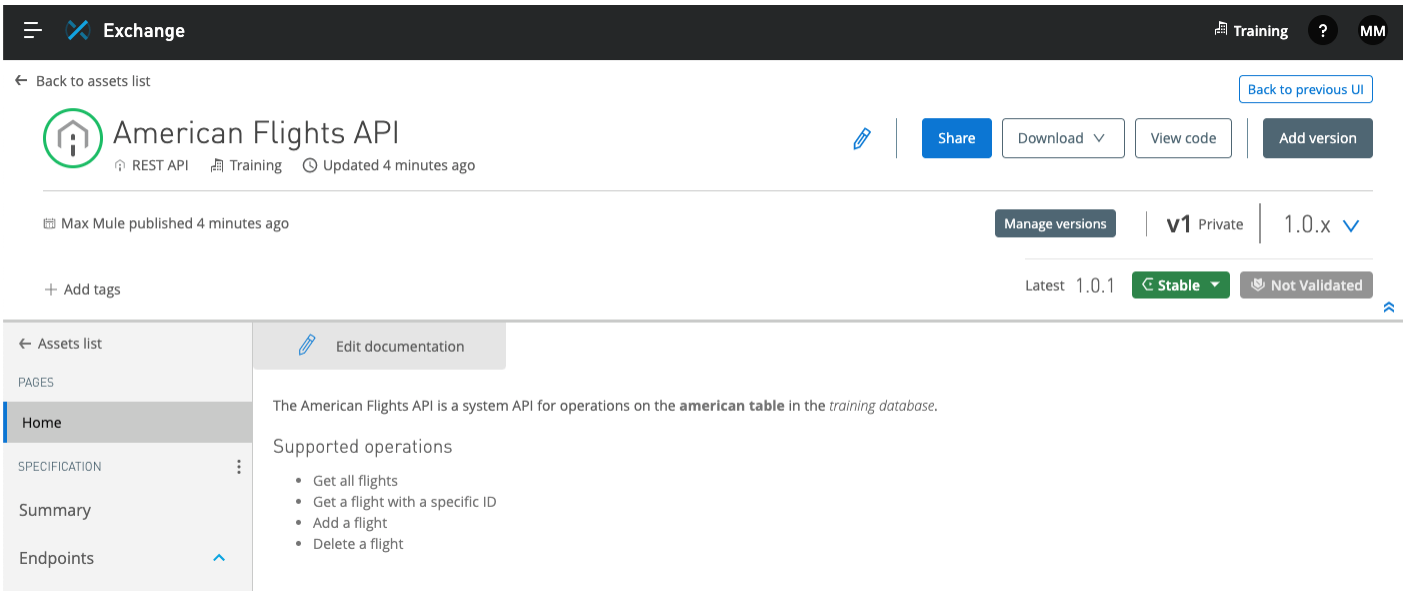


47. Click Close.
48. Click the Edit documentation button.

49. Add the new delete a flight operation.



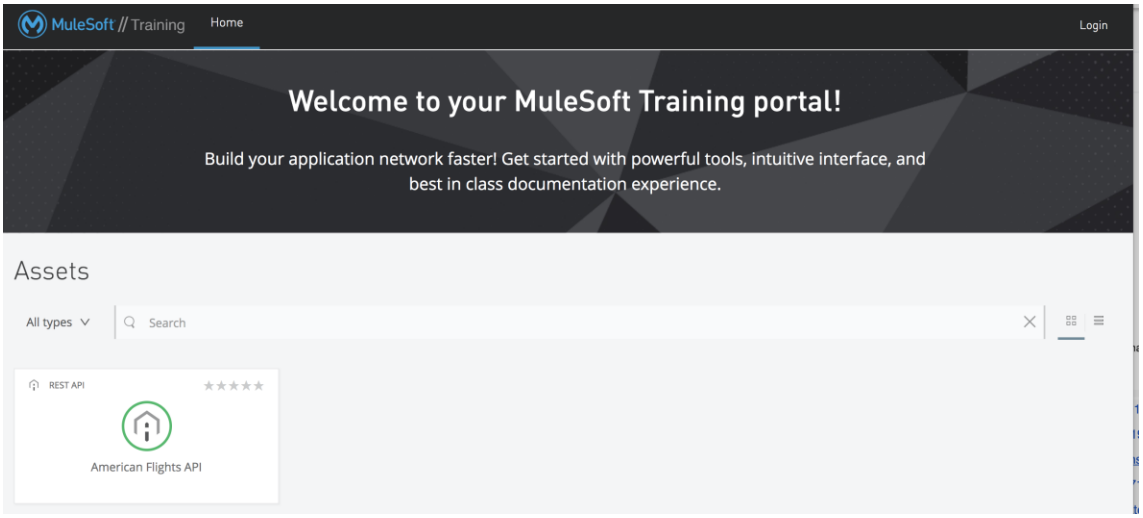
50. Click Save and then Publish.



Walkthrough 3-5: Share an API

In this walkthrough, you share an API with both internal and external developers to locate, learn about, and try out the API. You will:

- Share an API within an organization using the private Exchange.
- Create a public API portal.
- Customize a public portal.
- Explore a public portal as an external developer.

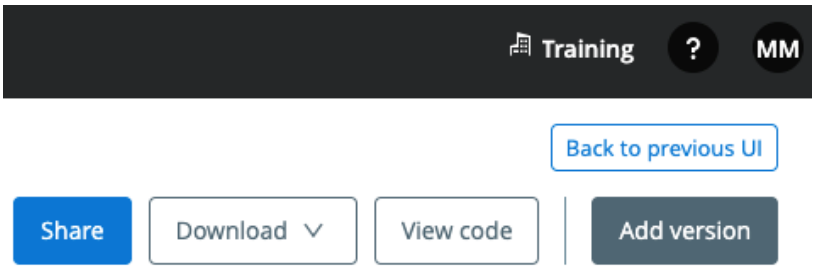


Starting file

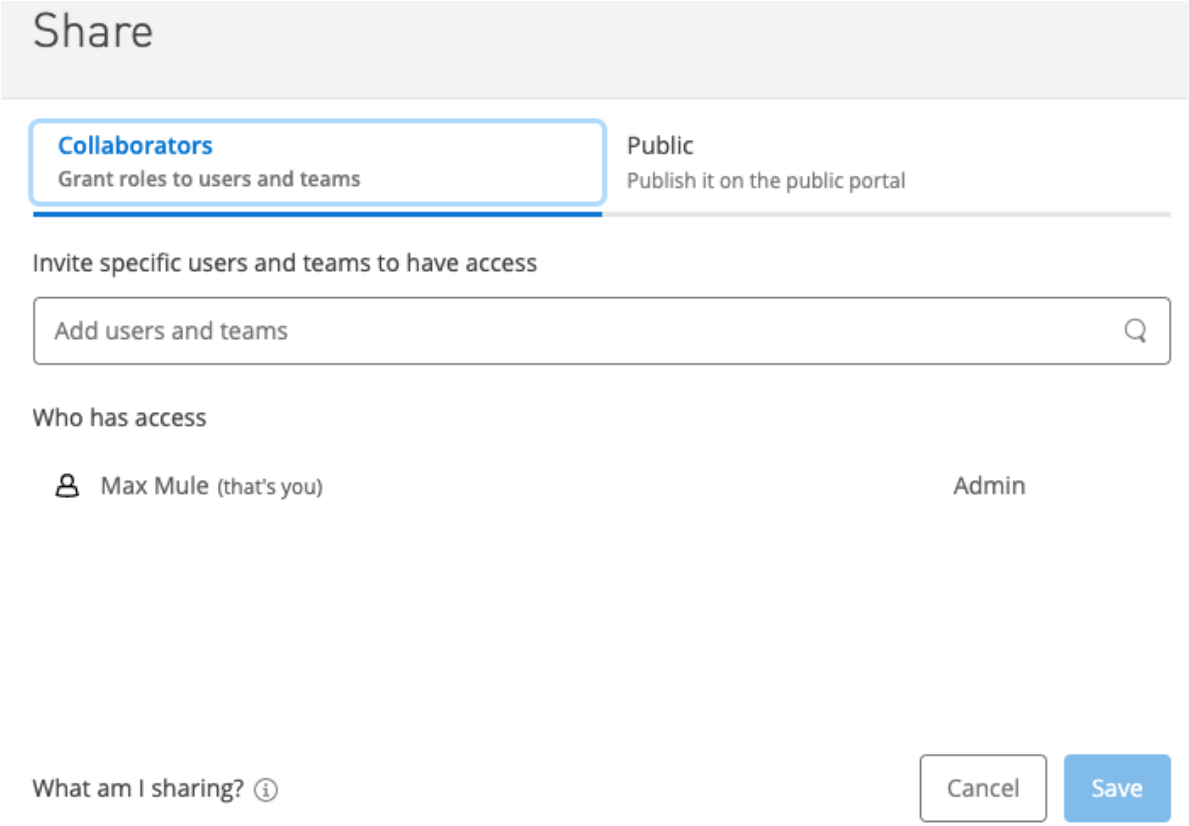
If you did not complete the previous walkthrough, you can get a starting file here. This file is also located in the solutions folder of the student files ZIP located in the Course Resources.

Share the API in the private Exchange with others

1. Return to your American Flights API in Exchange.
2. Click Share.

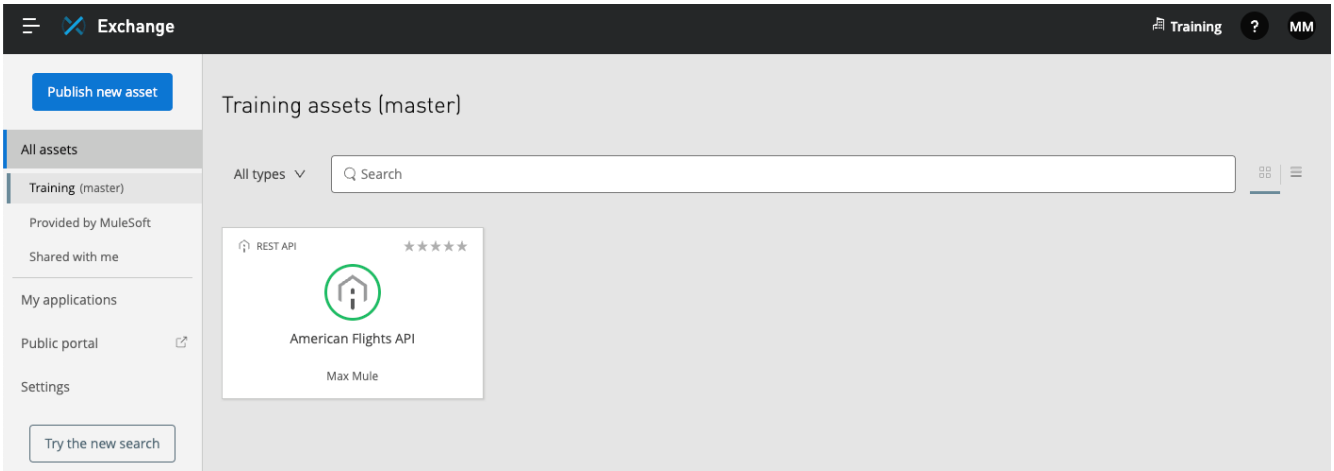


3. In the Share dialog box, you should see that you are an Admin for this API.



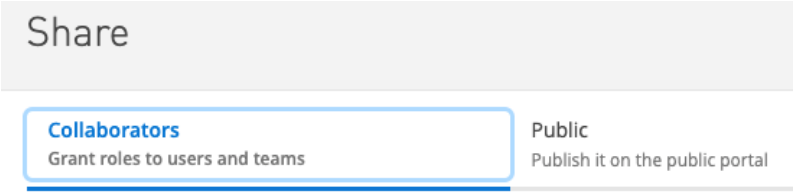
4. Click Cancel.
5. In the left-side navigation, click Assets list.
6. In the left-side navigation, select the name of your organization if necessary.
7. Click your American Flights API.

Note: This is how users you share the API with will find the API.

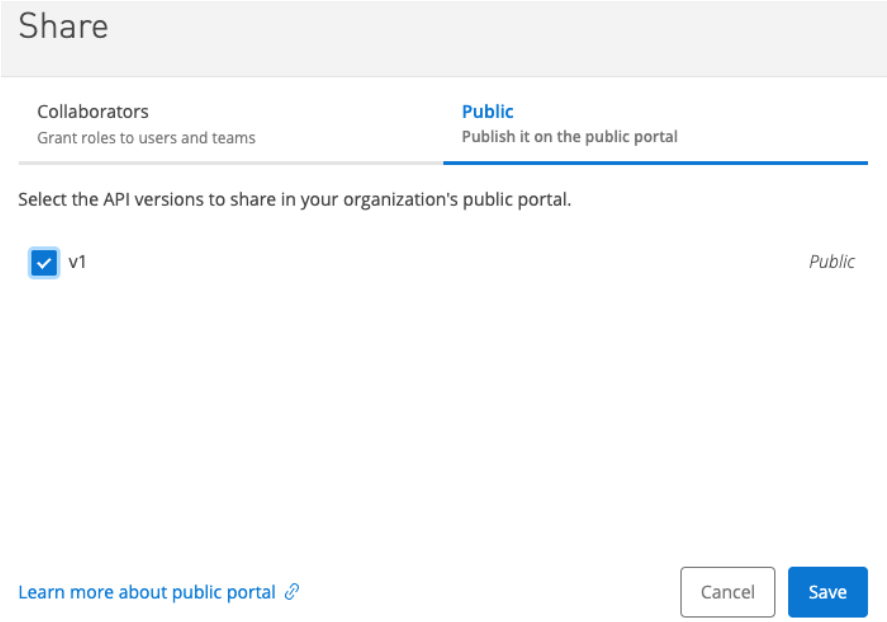


Create a public API portal

8. Click the Share button for the API again.
9. In the Share dialog box, click the Public tab.

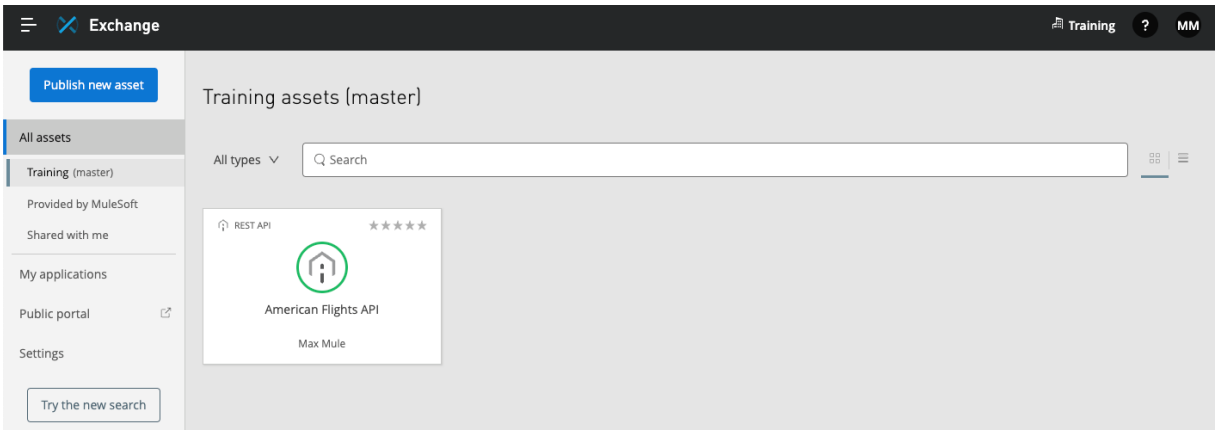


10. Select to share v1 of the API.
11. Click Save.



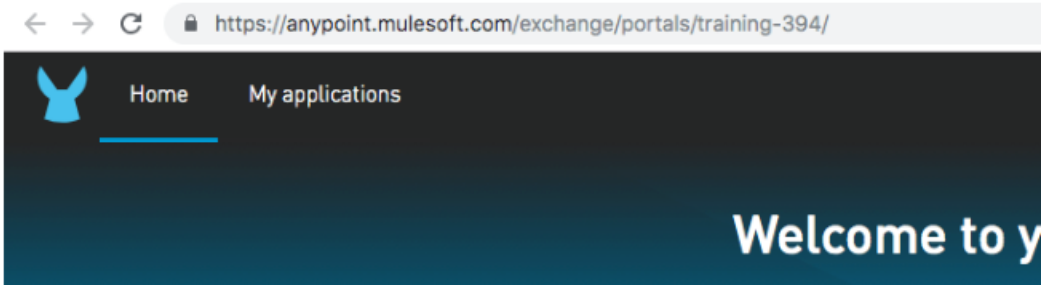
Explore the API in the public portal

12. In the left-side navigation, click Assets list.
13. In the left-side navigation, select Public Portal.

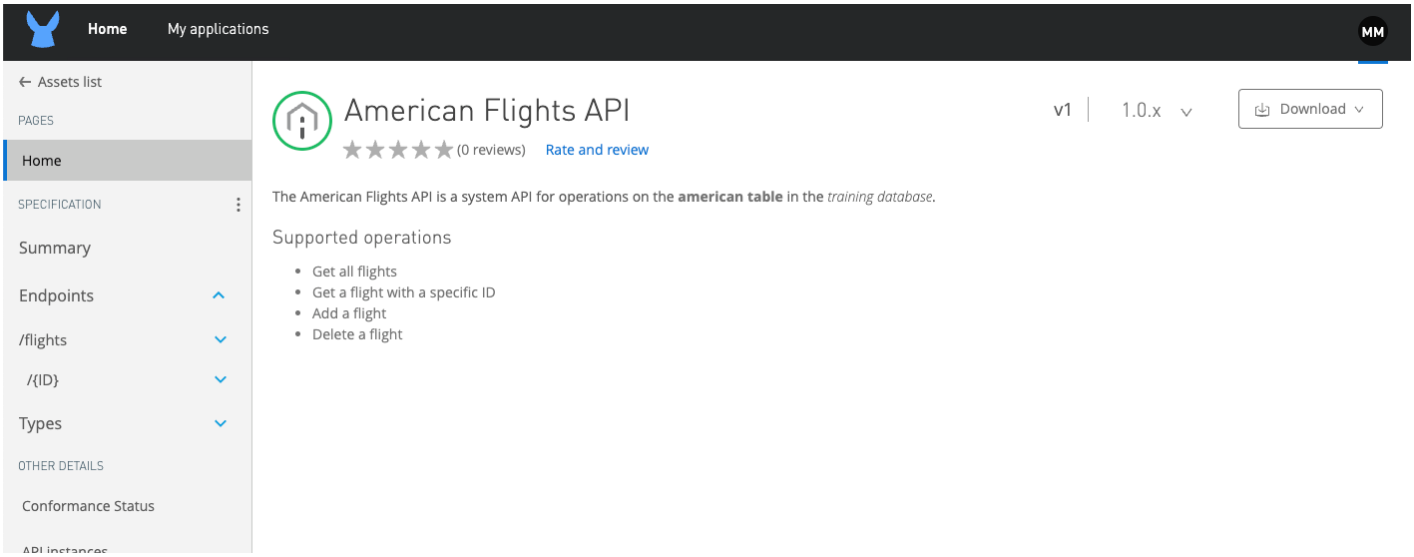


14. Review the public portal that opens in a new browser tab.

15. Look at the URL for the public portal.



16. Click your American Flights API and review the auto-generated public API portal.



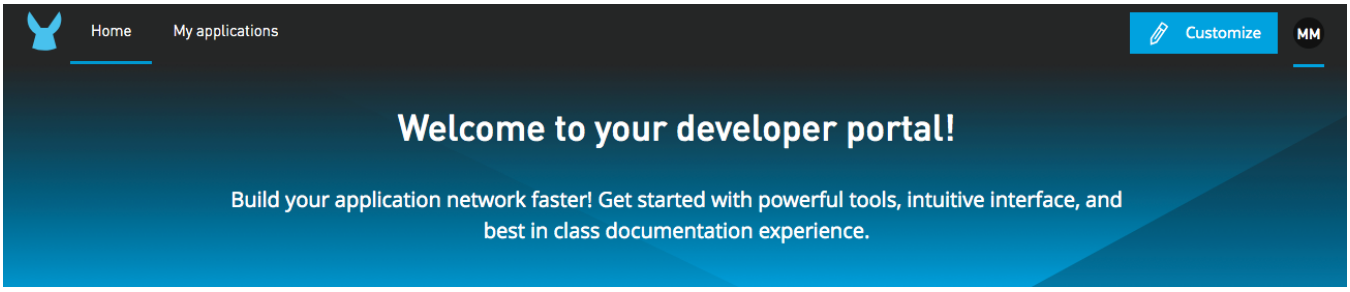
17. In the left-side navigation, click the GET method for the flights resource.

18. Review the response information.

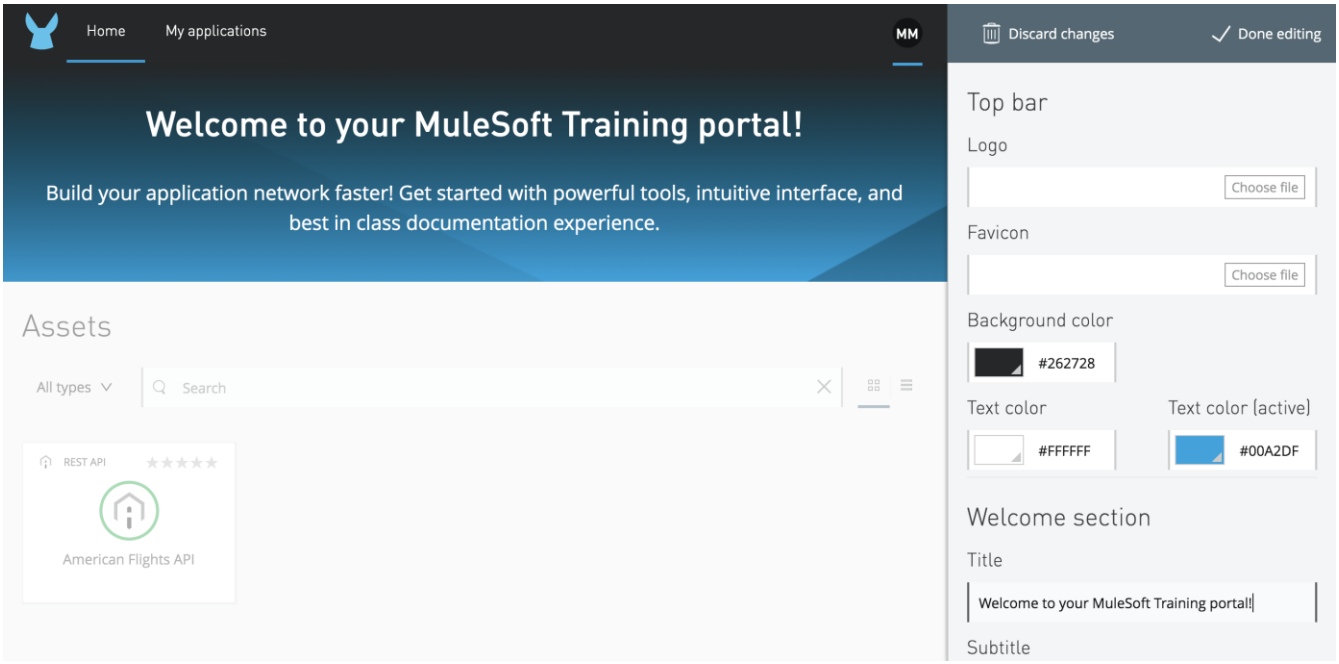
19. In the API console, click Send; you should get a 200 response with example flights returned from the mocking service.

Customize the public portal

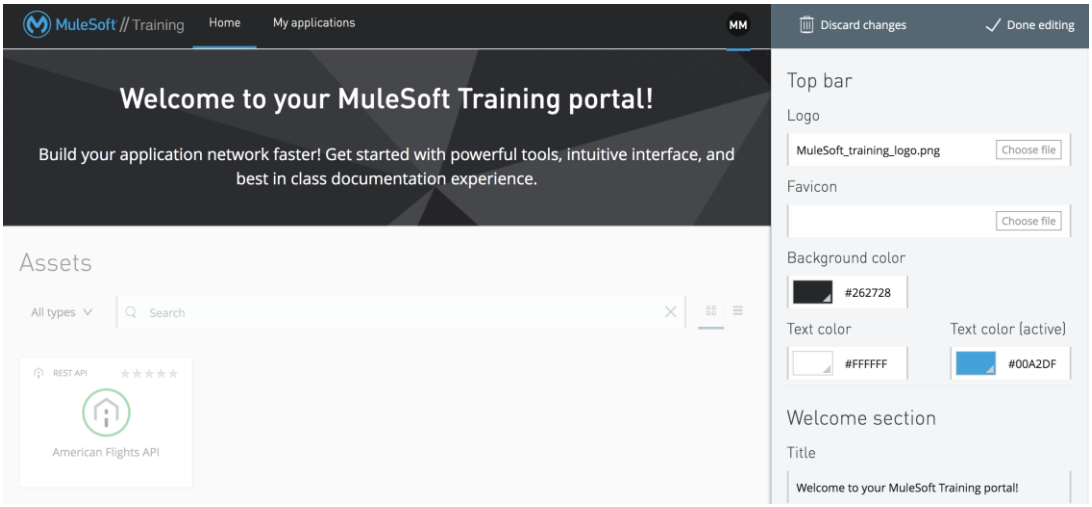
20. In the left-side navigation, click Assets list.
21. Click the Customize button.



22. In the Title field, change the text to Welcome to your MuleSoft Training portal!



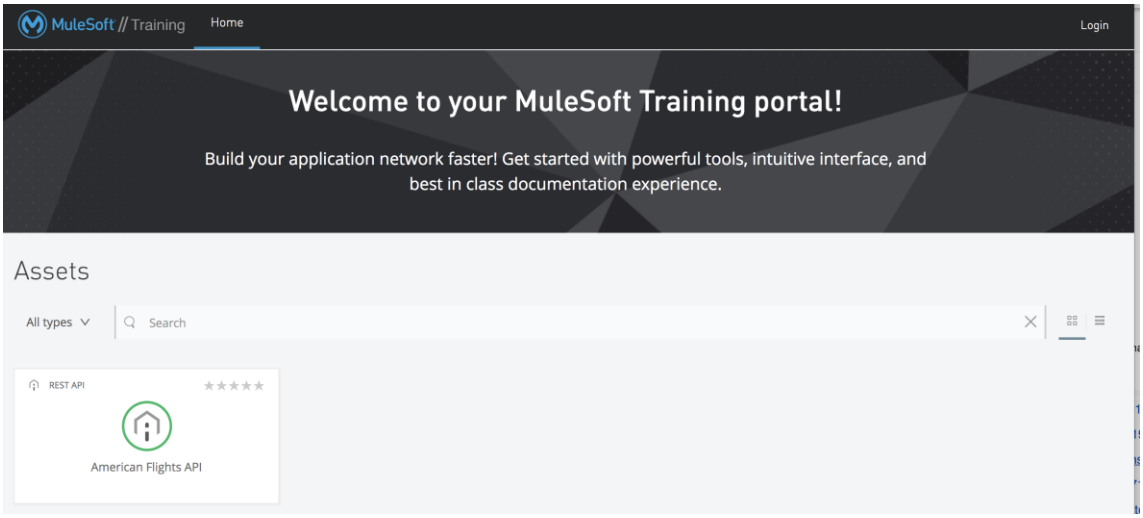
23. In the logo field, click Choose file.
24. In the file browser dialog box, navigate to the student files and locate the MuleSoft_training_logo.png file in the resources folder and click Open.
25. Locate the new logo in the preview.
26. In the hero image field, click Choose file.
27. In the file browser dialog box, navigate to the student files and locate the banner.jpg file in the resources folder and click Open.
28. Review the new logo and banner in the preview.



29. Change any colors that you want.
30. Click the Done editing button.
31. In the Publish changes dialog box, click Yes, publish; you should see your customized public portal.

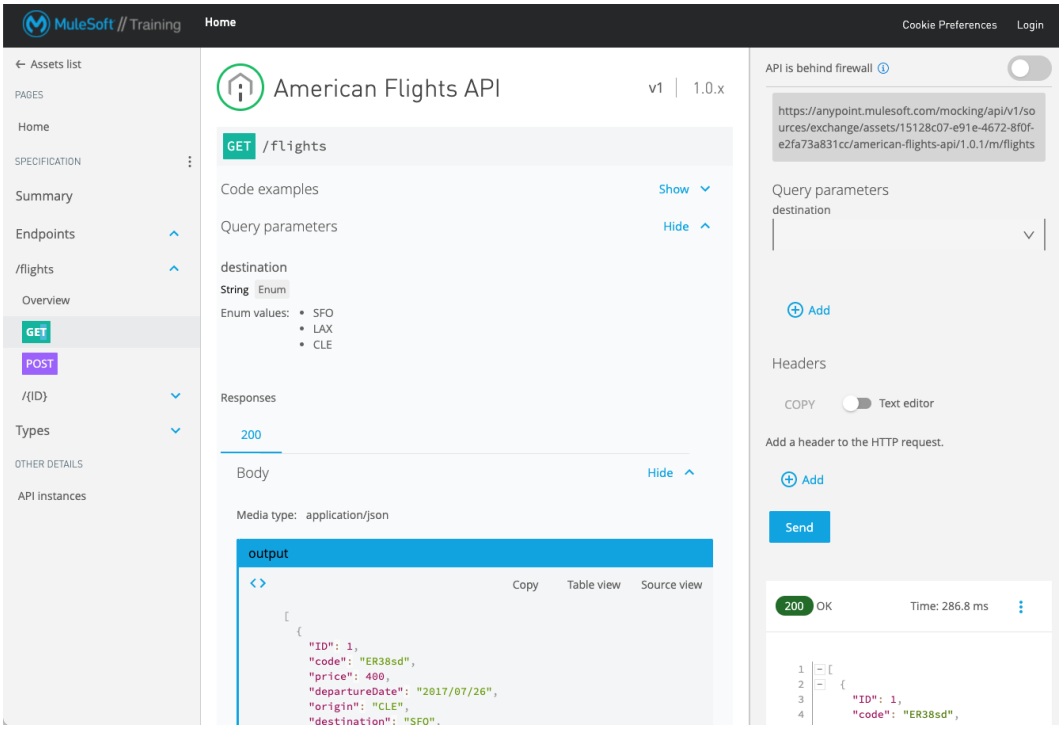
Explore the public portal as an external developer

32. In the browser, copy the URL for the public portal.
33. Open a new private or incognito window in your browser.
34. Navigate to the portal URL you copied; you should see the public portal (without the customize button).

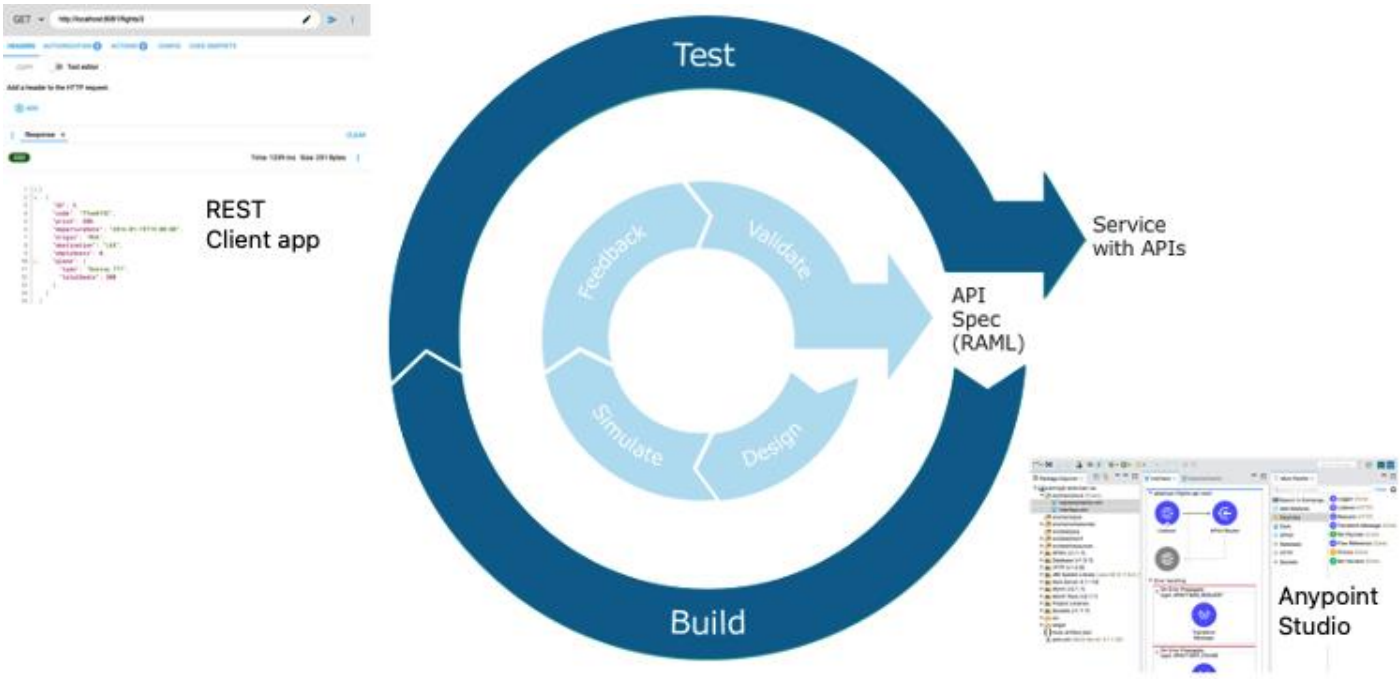


35. Click the American Flights API.
36. Explore the API portal.
37. Make a call to one of the resource methods.

Note: As an anonymous user, you can make calls to an API instance that uses the mocking service but not managed APIs.



Module 4: Building APIs



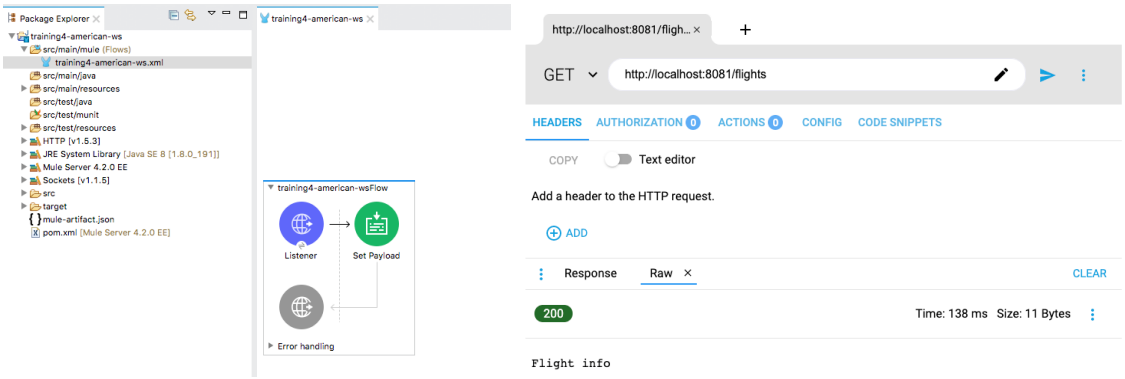
At the end of this module, you should be able to:

- Use Anypoint Studio to build, run, and test Mule applications.
- Use a connector to connect to databases.
- Use the graphical DataWeave editor to transform data.
- Create RESTful interfaces for applications from RAML files.
- Connect API interfaces to API implementations.
- Synchronize changes to API specifications between Anypoint Studio and Anypoint Platform.

Walkthrough 4-1: Create a Mule application with Anypoint Studio

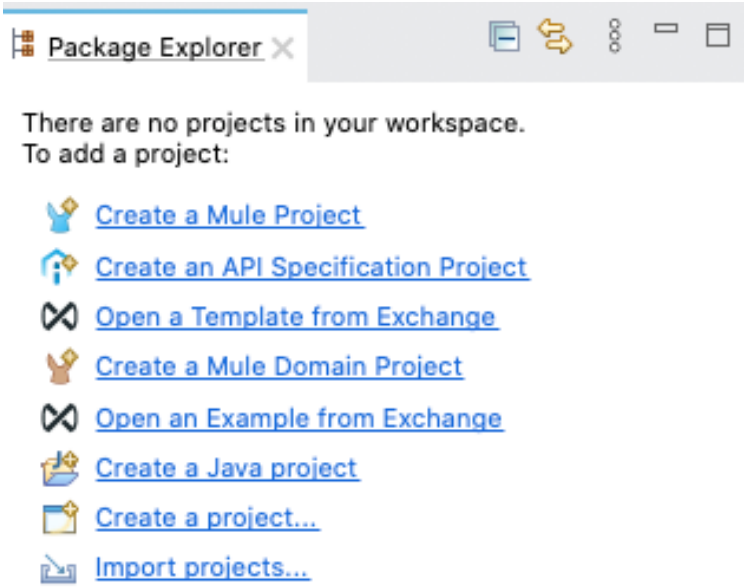
In this walkthrough, you build a Mule application. You will:

- Create a new Mule project with Anypoint Studio.
- Add a connector to receive requests at an endpoint.
- Set the event payload.
- Comment a component.
- Run a Mule application using the embedded Mule runtime.
- Make an HTTP request to the endpoint using Advanced REST Client.

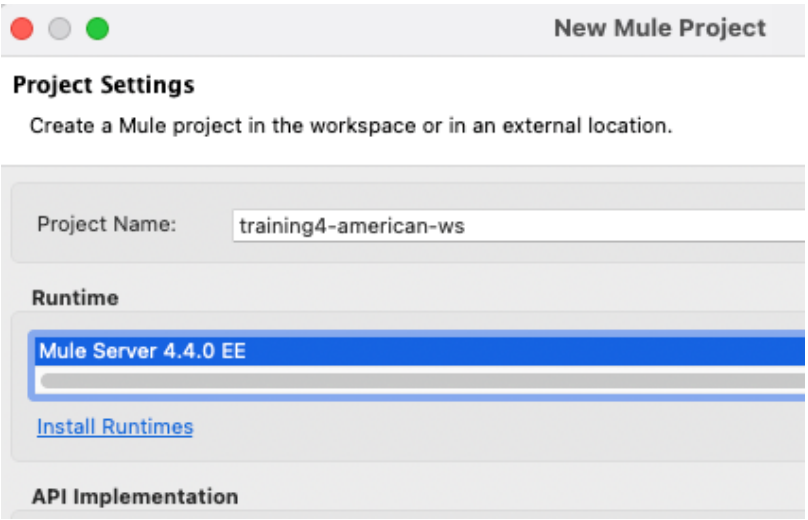


Create a Mule project

1. Open Anypoint Studio.
2. In the Package Explorer, select Create a Mule Project or select File > New > Mule Project if you already have a Mule project in your Studio workspace.



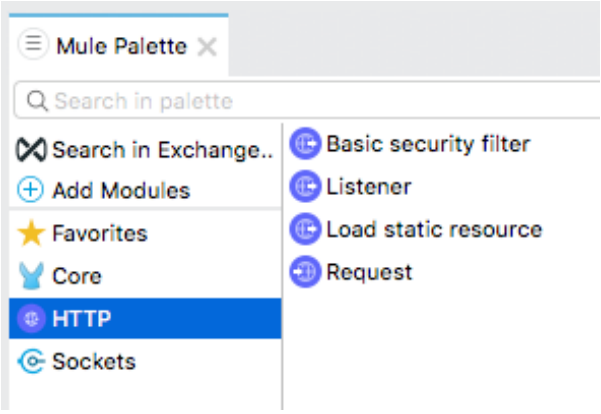
3. In the New Mule Project dialog box, set the Project Name to training4-american-ws.
4. Ensure the Runtime is set to the latest version of Mule.



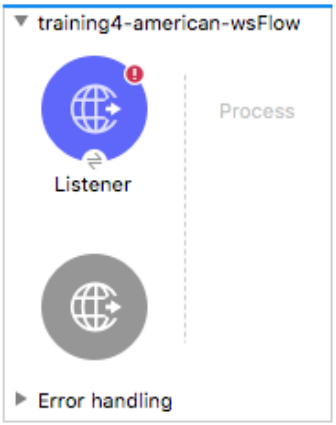
5. Click Finish.

Create an HTTP connector endpoint to receive requests

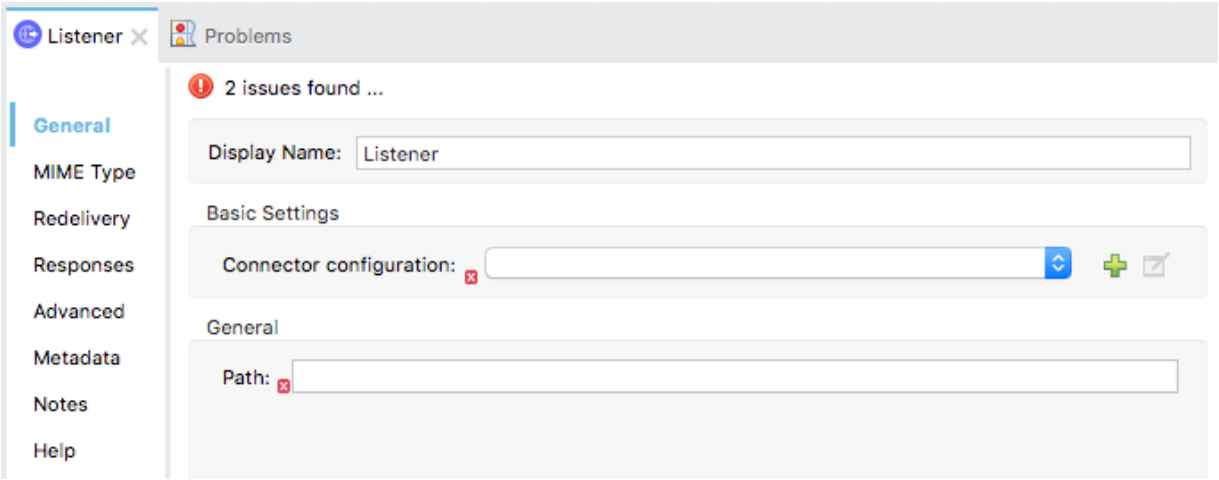
6. In the Mule Palette, select the HTTP module.



7. Drag the Listener operation from the Mule Palette to the canvas.

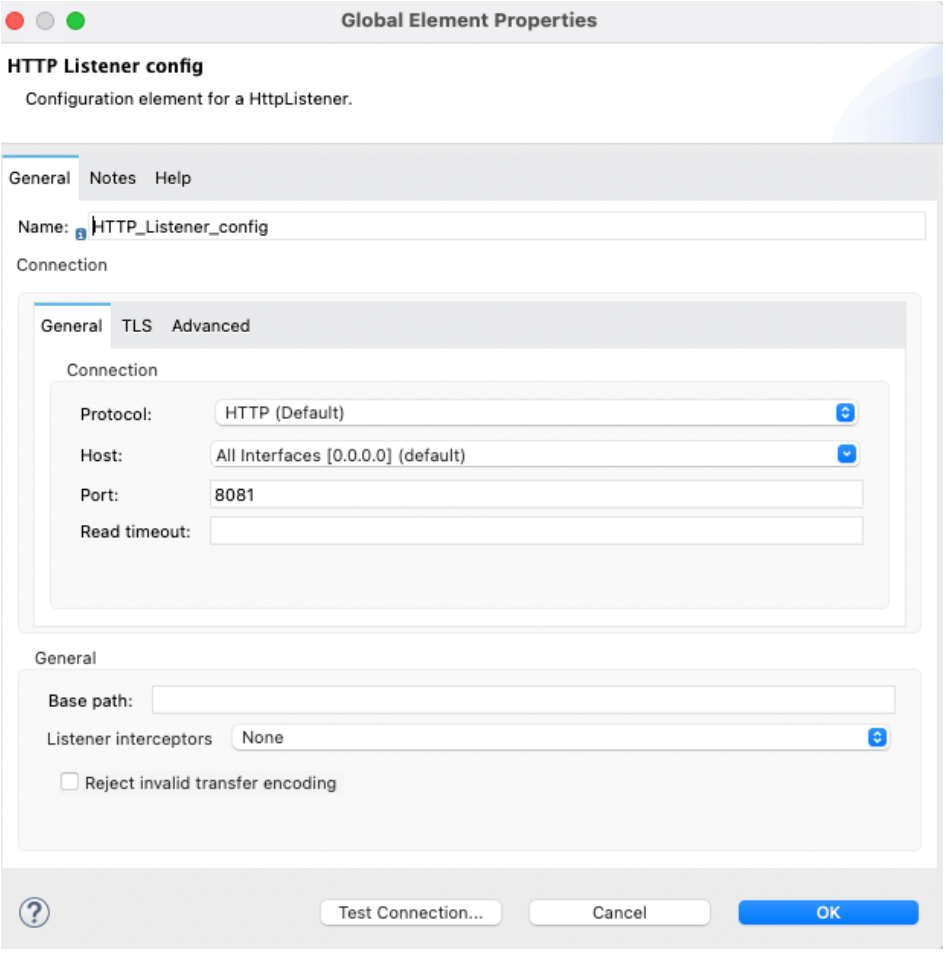


8. In the Listener properties view that opens at the bottom of the window, click the Add button next to connector configuration.



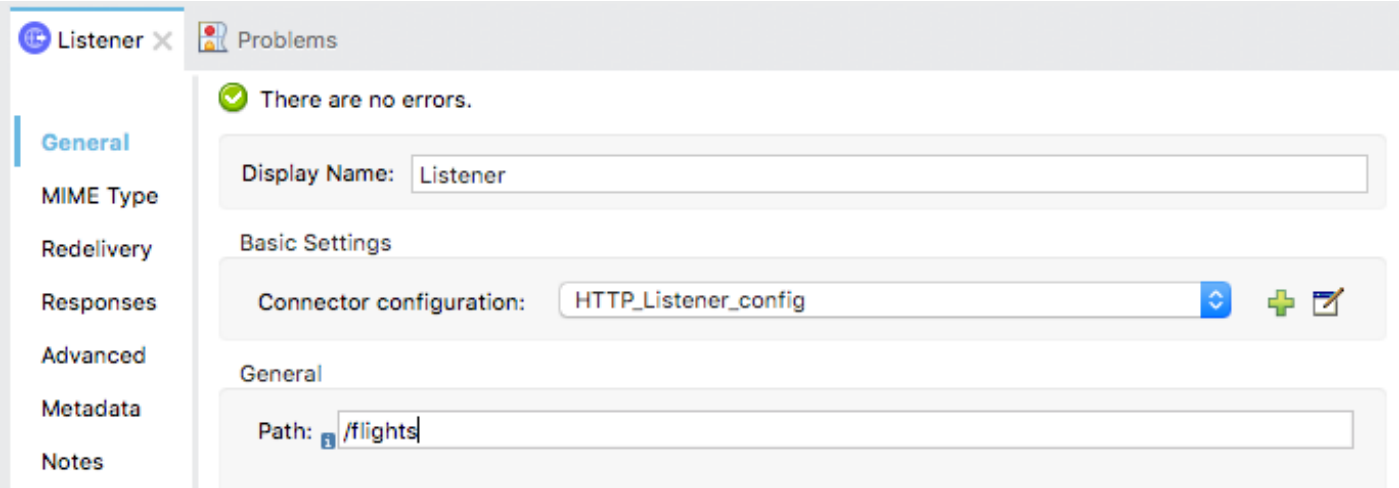
9. In the Global Element Properties dialog box, verify the following default values are present.

- Host: 0.0.0.0
- Port: 8081

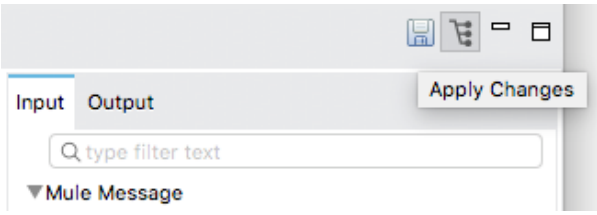


10. Click OK.

11. In the Listener properties view, set the path to /flights.

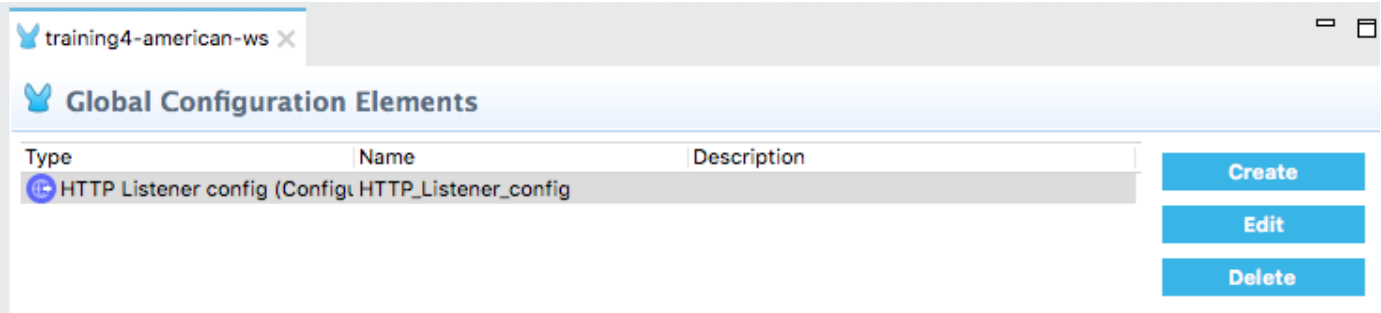


12. Click the Apply Changes button to save the file.



Review the HTTP Listener global element

13. Select the Global Elements tab at the bottom of the canvas.
14. Double-click the HTTP Listener config.

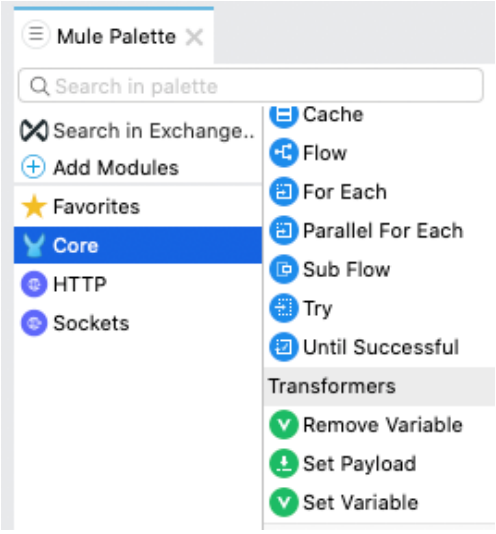


15. Review the information in the Global Element Properties dialog box and click Cancel.
16. Select the Message Flow tab to return to the canvas.

Display data

17. In the Mule Palette, select Core.

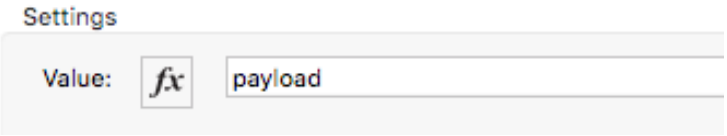
18. Scroll down in the right side of the Mule Palette and locate the Transformers section.



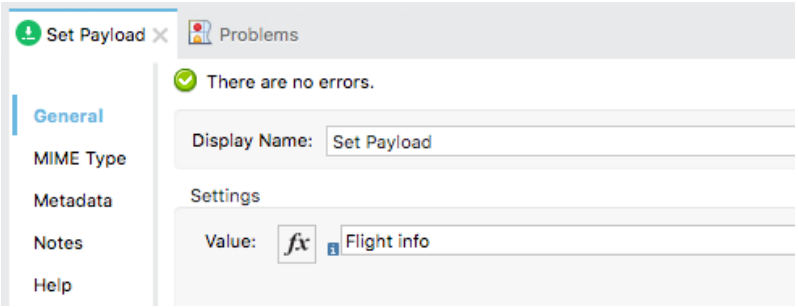
19. Drag the Set Payload transformer from the Mule Palette into the process section of the flow**.**

Configure the Set Payload transformer

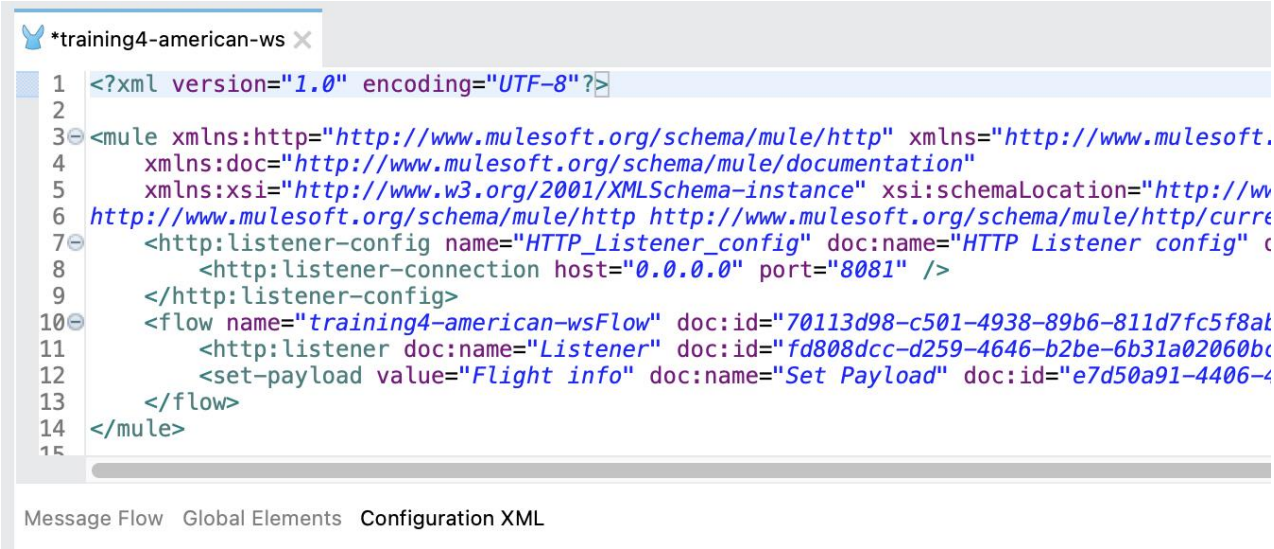
20. In the Set Payload properties view, click the Switch to literal mode button for the value field.



21. Set the value field to Flight info.



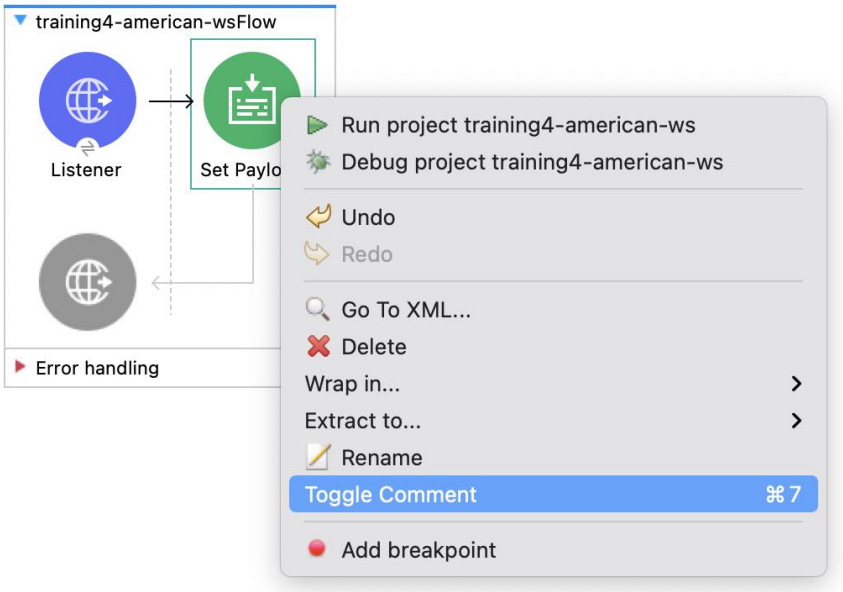
22. Select the Configuration XML tab at the bottom of the canvas and examine the corresponding XML.



23. Select the Message Flow tab to return to the canvas.

Comment a component

24. Right-click the Set Payload transformer and select Toggle Comment.



25. Select the Configuration XML tab at the bottom of the canvas and examine the changes to the XML.

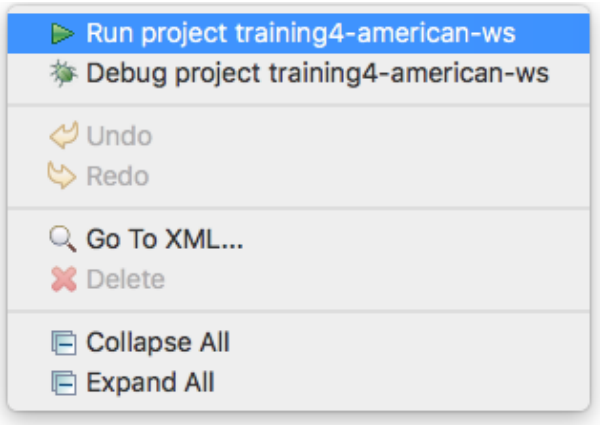
```
*training4-american-ws
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <mule xmlns:http="http://www.mulesoft.org/schema/mule/http" xmlns="http://www.mulesoft.
4      xmlns:doc="http://www.mulesoft.org/schema/mule/documentation"
5      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://ww
6  http://www.mulesoft.org/schema/mule/http http://www.mulesoft.org/schema/mule/http/curre
7      <http:listener-config name="HTTP_Listener_config" doc:name="HTTP Listener config" d
8          <http:listener-connection host="0.0.0.0" port="8081" />
9      </http:listener-config>
10     <flow name="training4-american-wsFlow" doc:id="70113d98-c501-4938-89b6-811d7fc5f8ab
11         <http:listener doc:name="Listener" doc:id="fd808dcc-d259-4646-b2be-6b31a02060bc
12 <!-- [STUDIO:"Set Payload"]     <set-payload value="Flight info" doc:name="Set Payload"
13     </flow>
14 </mule>
15
```

Message Flow Global Elements Configuration XML

26. Select the Message Flow tab to return to the canvas.
27. Right-click the Set Payload transformer and select Toggle Comment.
28. Click the Save button or press Cmd+S or Ctrl+S.

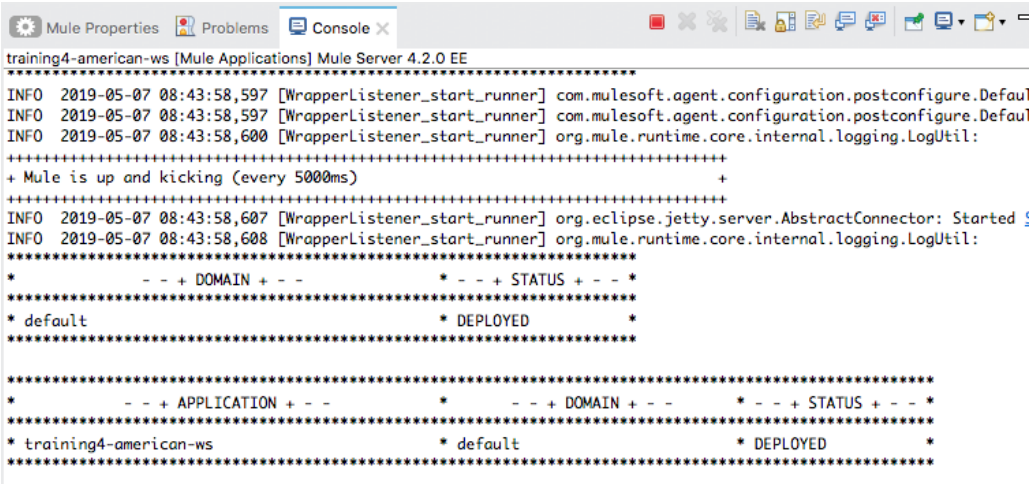
Run the application

29. Right-click in the canvas and select Run project training4-american-ws.



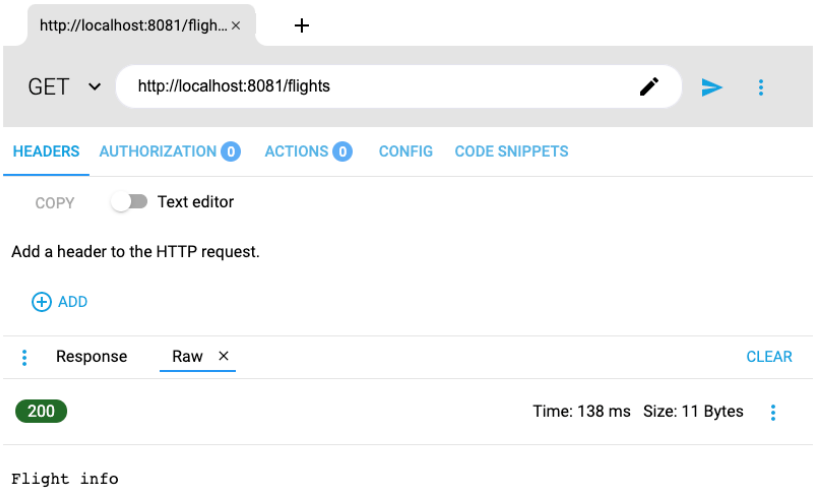
Note: If you get a dialog asking to accept incoming network connections, click Allow.

30. Watch the Console view; it should display information letting you know that both the Mule runtime and the training4-american-ws application started.

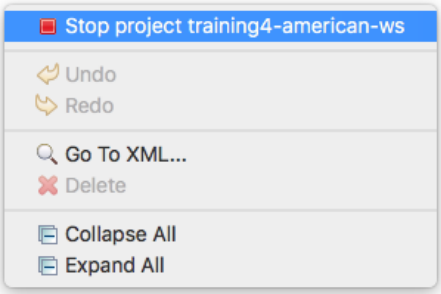


Test the application

31. Return to Advanced REST Client.
32. Make sure the method is set to GET and that no headers or body are set for the request.
33. Make a GET request to http://localhost:8081/flights; you should see Flight info displayed.



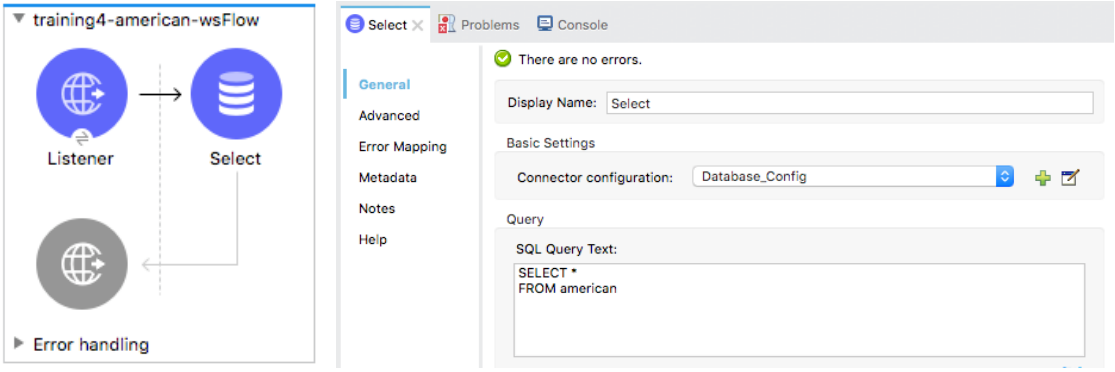
34. Return to Anypoint Studio.
35. Right-click in the canvas and select Stop project training4-american-ws.



Walkthrough 4-2: Connect to data (MySQL database)

In this walkthrough, you connect to a database and retrieve data from a table that contains flight information. You will:

- Add a Database Select operation.
- Configure a Database connector that connects to a MySQL database (or optionally an in-memory Derby database if you do not have access to port 3306).
- Configure the Database Select operation to use that Database connector.
- Write a query to select data from a table in the database.



Starting file

If you did not complete the previous walkthrough, you can get a starting file here. This file is also located in the solutions folder of the student files ZIP located in the Course Resources. See here for steps on importing a starting file deployable jar archive into Studio.

Locate database information

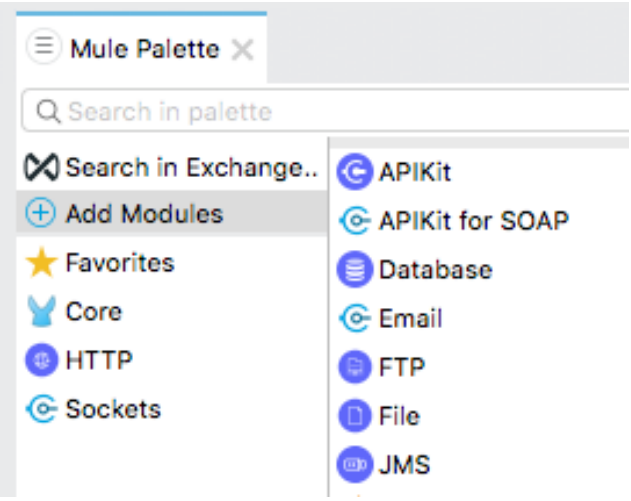
1. Return to the course snippets.txt file and locate the MySQL and Derby database information.

```
* MySQL database
db:
        host: "mudb.learn.mulesoft.com"
        port: "3306"
        user: "mule"
        password: "mule"
        database: "training"

American table: american
Account table: accounts
Account list URL:  http://mu.learn.mulesoft.com/accounts/show
or if using mulesoft-training-services.jar application:
http://localhost:9090/accounts/show.html

* MySQL database as URL and driver name
URL: jdbc:mysql://mudb.learn.mulesoft.com:3306/training?user=mule&password=mule
Driver class name: com.mysql.jdbc.Driver

* Derby database
URL: jdbc:derby://localhost:1527/memory:training
Driver class name: org.apache.derby.jdbc.ClientDriver
```

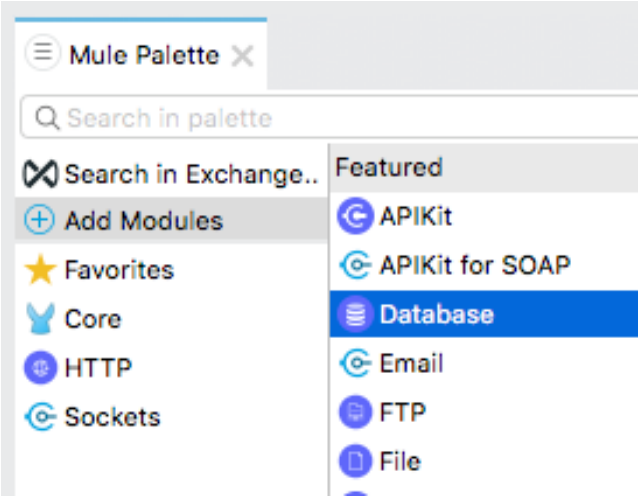
Note: The database information you see may be different than what is shown here; the values in the snippets file differ for instructor-led and self-study training classes.

Add a Database connector endpoint

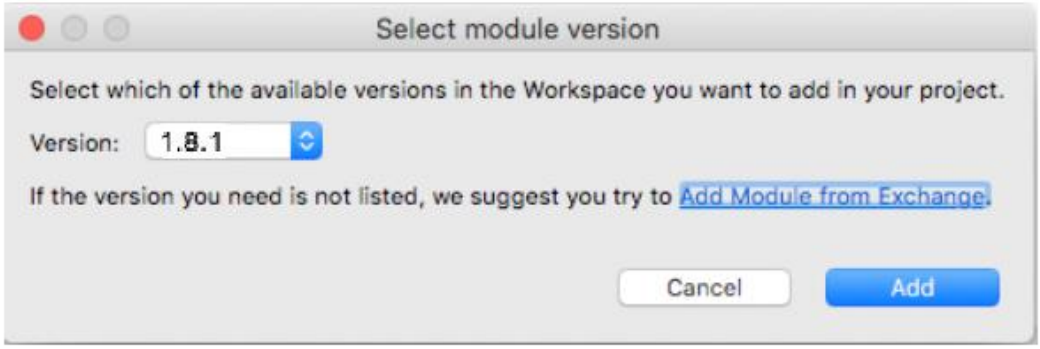
2. Return to Anypoint Studio.
3. Right-click the Set Payload message processor and select Delete.
4. In the Mule Palette, select Add Modules.



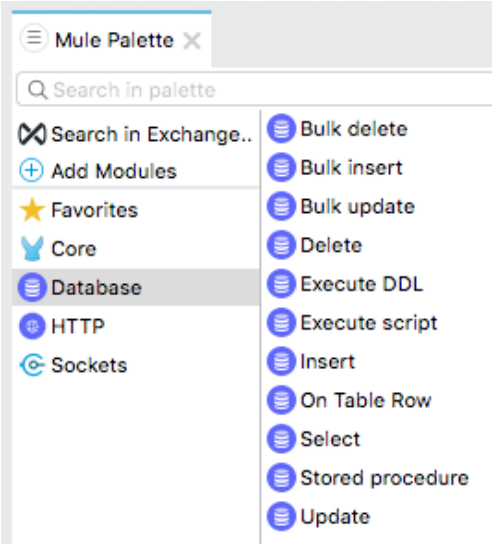
5. Select the Database connector in the right side of the Mule Palette and drag and drop it into the left side.



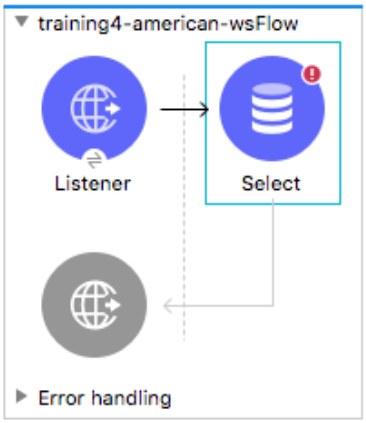
6. If you get a Select module version dialog box, select the latest version and click Add.



7. Locate the new Database connector in the Mule Palette.



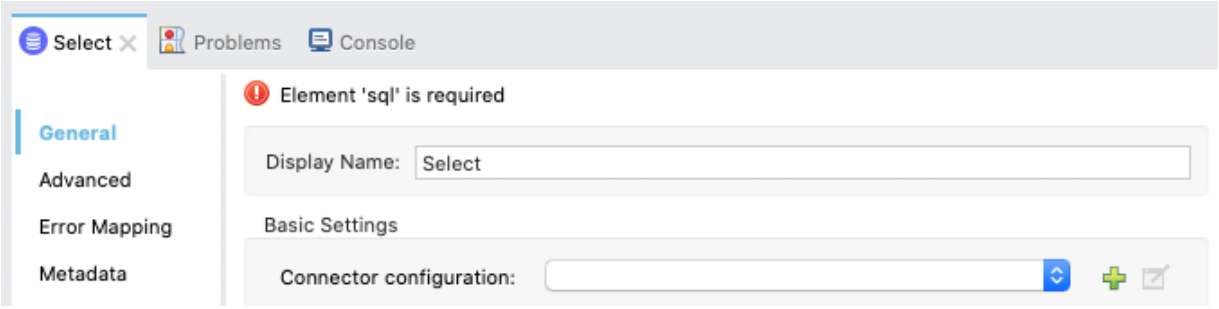
8. Drag and drop the Select operation in the process section of the flow.



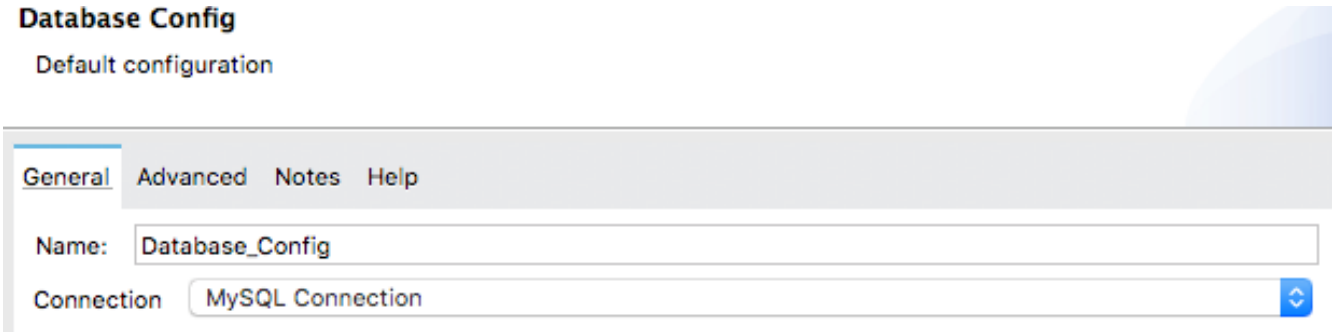
Option 1: Configure a MySQL Database connector (if you have access to port 3306)

This section attempts to connect to a MySQL database and tests whether you are successful. A successful connection requires access to port 3306. If you find that you cannot successfully connect, a second option is described in the sections following this one that will allow you to locally host the database along with other services that may be needed in the course.

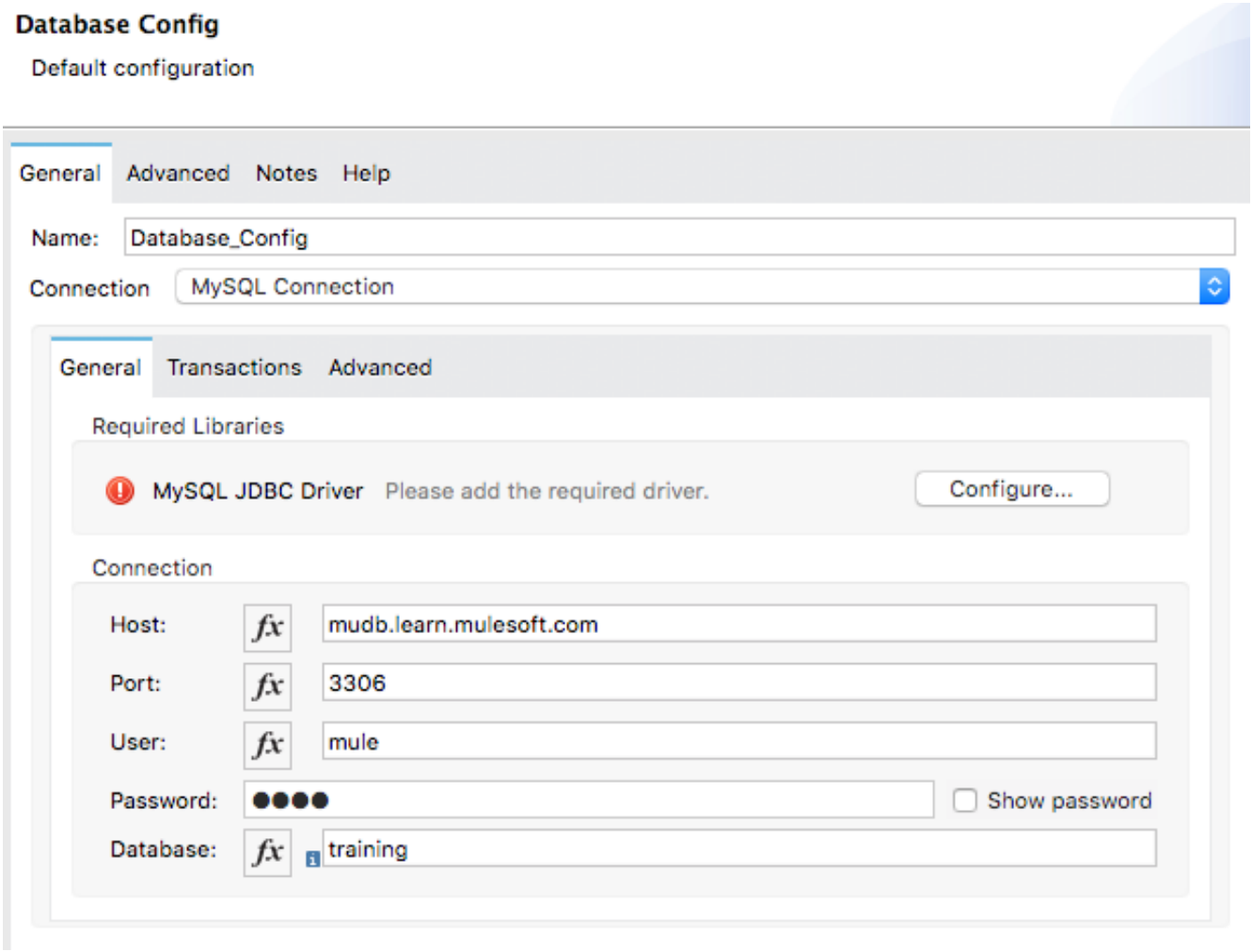
9. In the Select properties view, click the Add button next to connector configuration.



10. In the Global Element Properties dialog box, set the Connection to MySQL Connection.

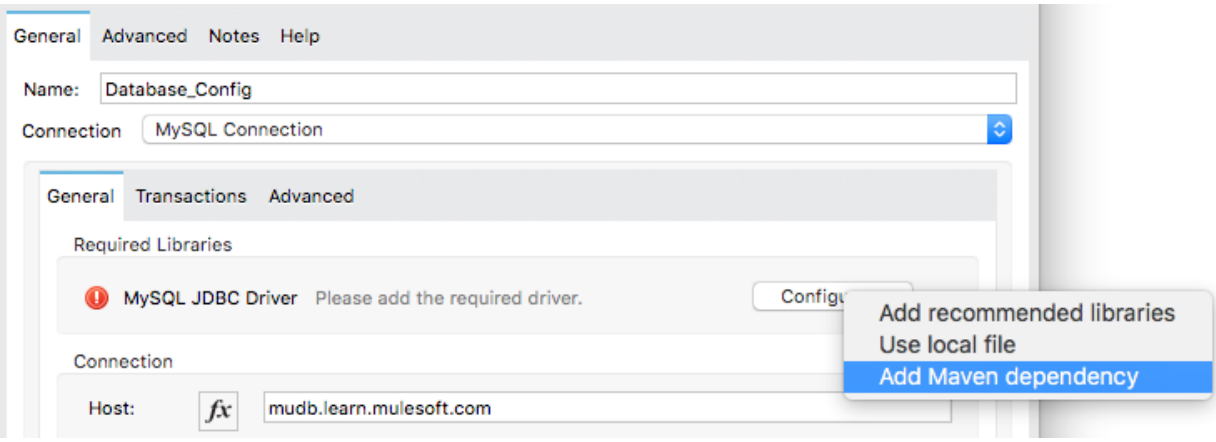


11. Set the host, port, user, password, and database values to the values listed in the course snippets.txt file.

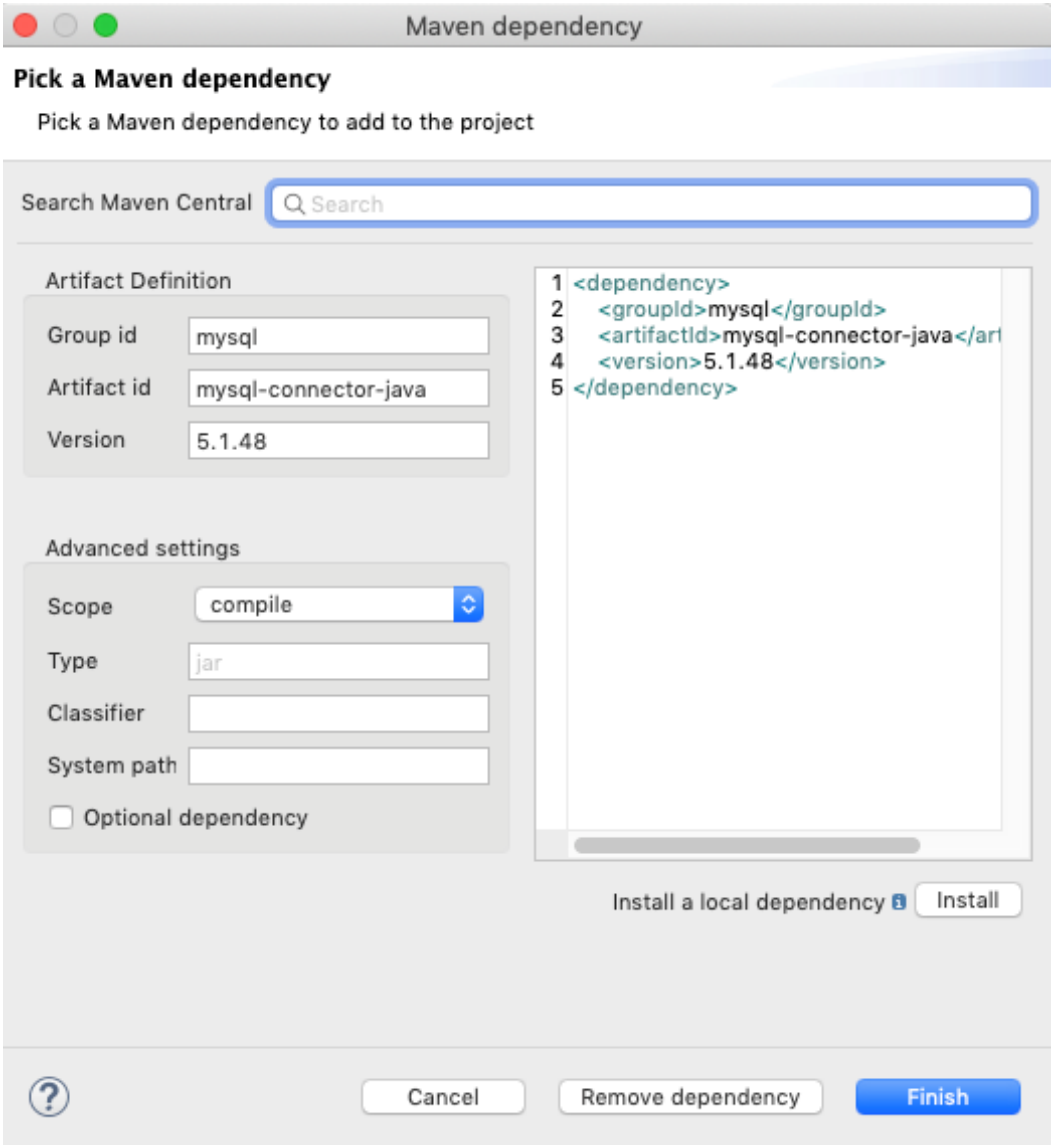


12. Click the Configure button next to MySQL JDBC Driver.

13. In the configure drop-down menu, select Add Maven dependency.

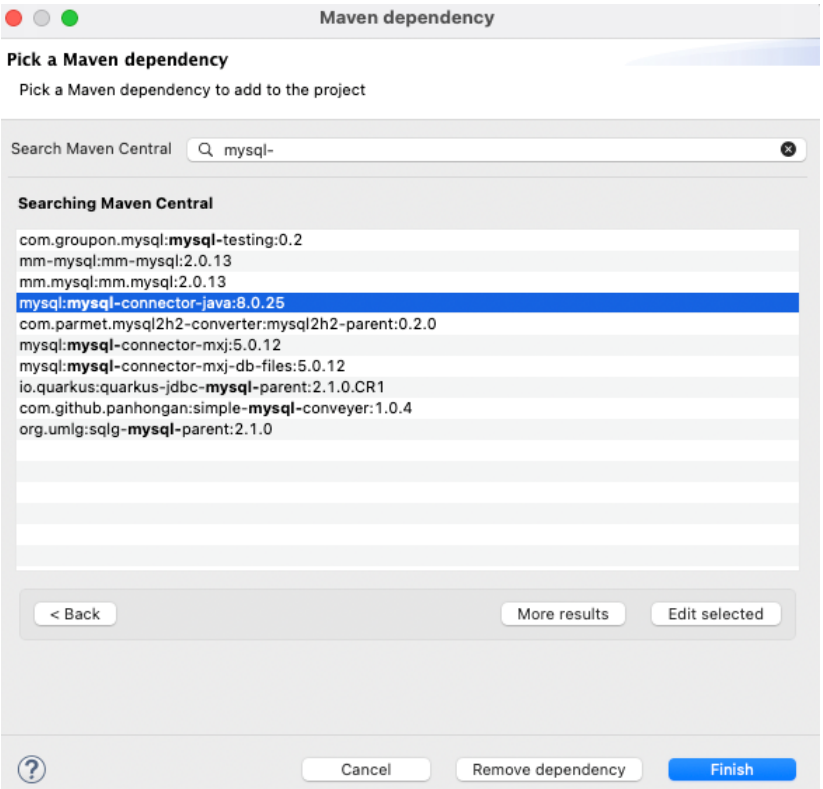


14. In the Maven dependency dialog box, locate the Search Maven Central text field.

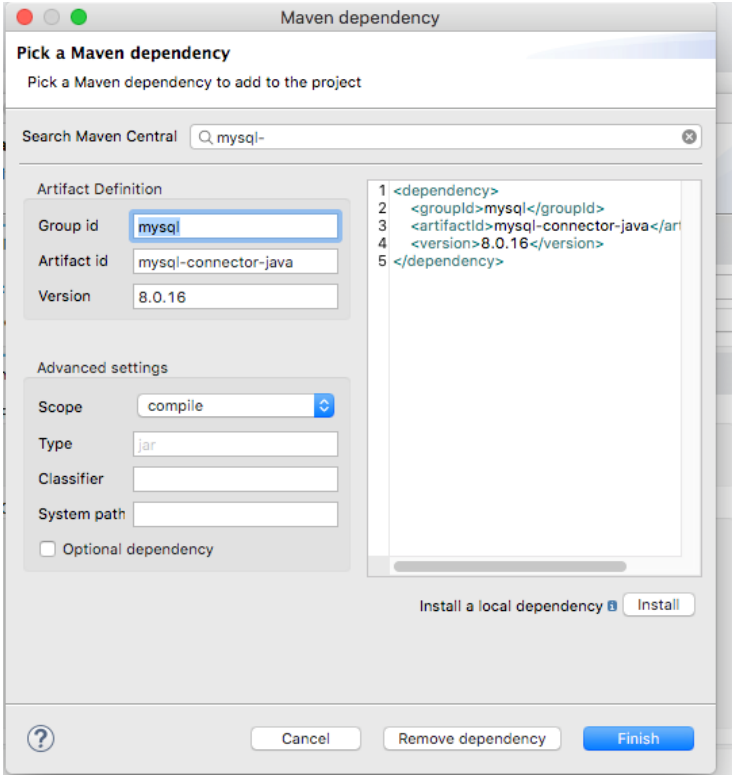


15. Enter mysql- in the Search Maven Central text field.

16. Select mysql:mysql-connector-java in the results that are displayed.

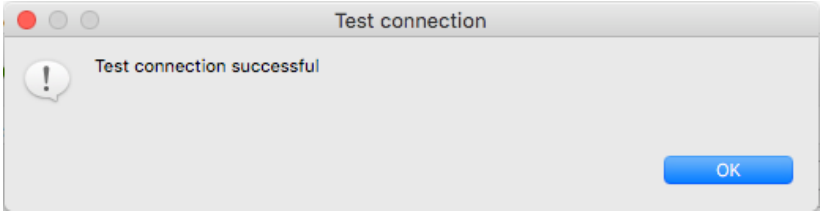


17. Click Edit selected.
18. Enter 8.0.16 in the Version text field.
19. Click Finish.



20. Back in the Global Element Properties dialog box, click the Test Connection button; you should get a successful test dialog box.

Note: Make sure the connection succeeds before proceeding.



Note: If the connectivity test fails, make sure you are not behind a firewall restricting access to port 3306. If you cannot access port 3306, use the instructions in the next section for option 2.

21. Click OK to close the Test connection dialog box.
22. Click OK to close the Global Element Properties dialog box and skip to the Write a query to return all flights section below.

Option 2: Run the MuleSoft training services application (if no access to port 3306)

If you were unable to connect to the MySQL database, the second option described here presents an alternative solution that will allow you to locally host the database along with other services that may be needed in the course.

23. In a command-line interface, use the cd command to navigate to the folder containing the jars folder of the student files.
24. Run the mulesoft-training-services.jar file.

```
java –jar mulesoft-training-services-X.X.X.jar
```

Note: Replace X.X.X with the version of the JAR file, for example 1.8.8.

Note: The application uses ports 1527, 9090, 9091, and 61616. If any of these ports are already in use, you can change them when you start the application as shown in the following code.

```
java -jar mulesoft-training-services-X.X.X.jar --database.port=1530 --
ws.port=9092 --activemq.broker.url=tcp://localhost:61617 --server.port=9193
```

25. Look at the output and determine whether all the services started successfully or if there was an error.

```
(\_/)          M U L E S O F T   T R A I N I N G   &   C E R T I F I C A T I O N
/   \          Anypoint Platform Development Fundamentals - Services & APIs

Starting resources:
- Starting embedded database on port 1527. Please wait...
- Database started
- Creating and populating database tables. Please wait...
- Database ready
- Message Broker started
- Order web service started
- Starting Delta flights web service
- Delta flights web service started
- United flights web service started
- JMS API published
- American flights web service with RAML API spec started
- Banking REST API published
- Accounts REST API published

Published resources:
- Landing page           : http://localhost:9090
- American database URL  : jdbc:derby://localhost:1527/memory:training
- American REST API      : http://localhost:9090/american/flights
- American REST API RAML : http://localhost:9090/american/flights-api.raml
- United REST service    : http://localhost:9090/united/flights
- Delta SOAP service     : http://localhost:9191/delta
- Delta SOAP WSDL        : http://localhost:9191/delta?wsdl
- Accounts API           : http://localhost:9090/accounts/api
- Accounts web form      : http://localhost:9090/accounts/show.html
- JMS broker URL         : tcp://localhost:61616
- JMS topic name         : apessentials
- JMS web form           : http://localhost:9090/jmsform.html
- Banking API base URL   : http://localhost:9090/api/...
- Banking API RAML       : http://localhost:9090/api/banking-api.raml

Press CTRL-C to terminate this application...
```

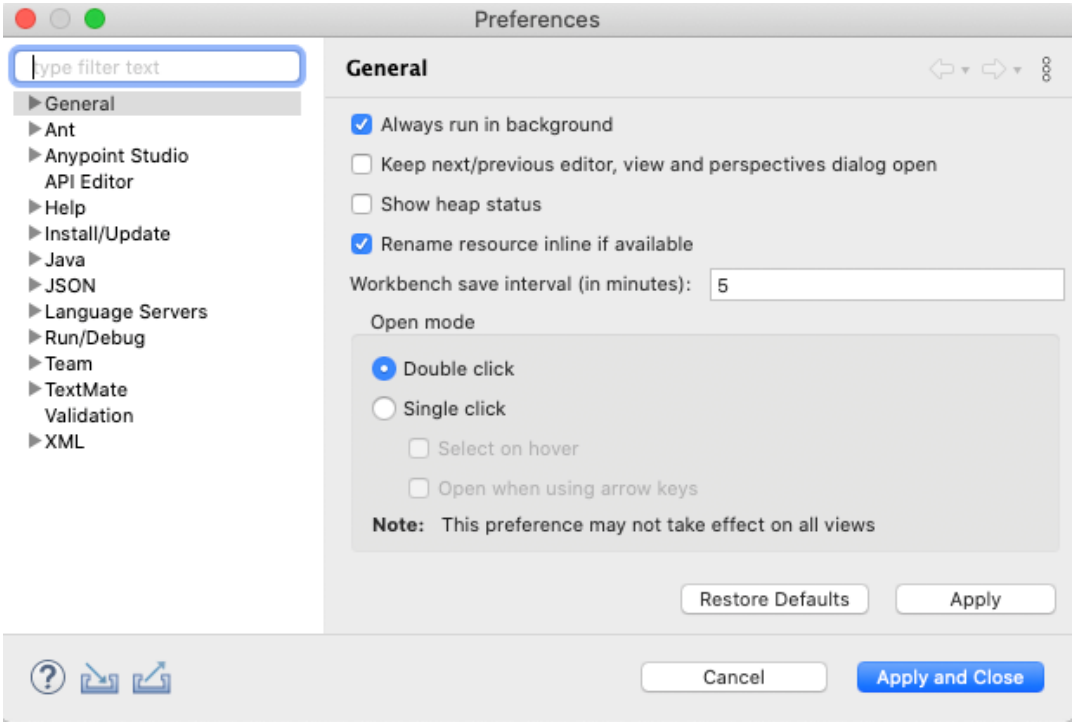
Note: When you want to stop the application, return to this window and press Ctrl+C.

26. If the services started successfully, skip to the Option 2 (continued): Configure a Derby Database connector section below.

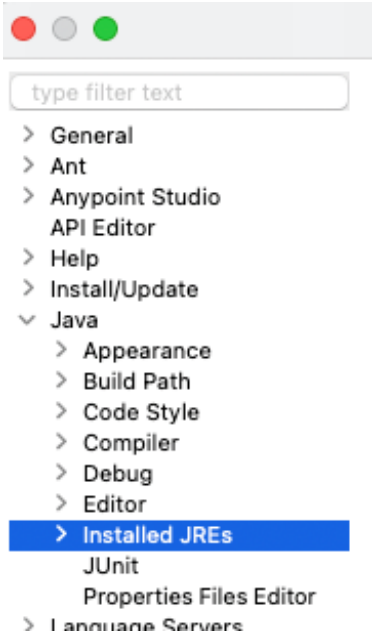
Option 2 (continued): Set the Java environment to the Studio-embedded Java

27. Return to Anypoint Studio.
28. Open Anypoint Studio's Preference dialog.

- Windows: Window > Preferences from top menu
- Mac: Anypoint Studio > Preferences from top menu



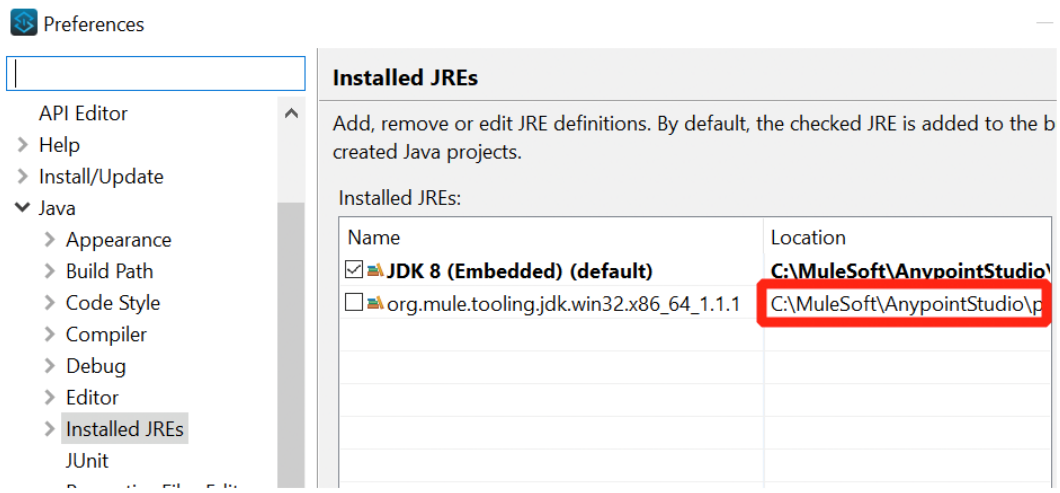
29. Select Java > Installed JREs.



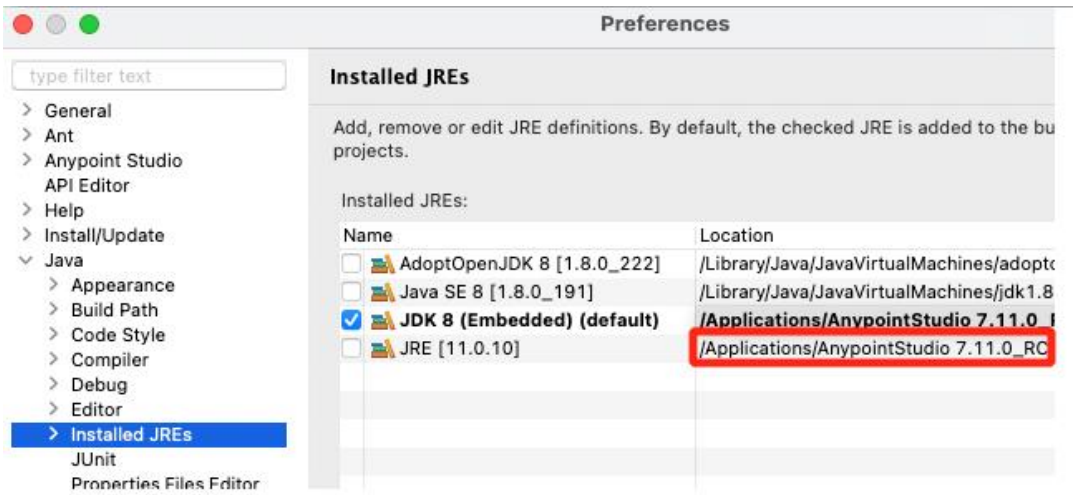
30. Use the edit functionality to copy the value in the Location column for the Java 11 JRE then close the Preference dialog.

Note: Be sure not to copy the location for the selected embedded JDK.

- Windows



- Mac:



31. Return to the open terminal window and set JAVA_HOME and Path.

- Windows:

```
set JAVA_HOME="<Copied JRE Location>"
set Path=.;%JAVA_HOME%\bin;%Path%
```

- Mac:

```
export JAVA_HOME="<Copied JRE Location>"
export PATH=.:$JAVA_HOME/bin:$PATH
```

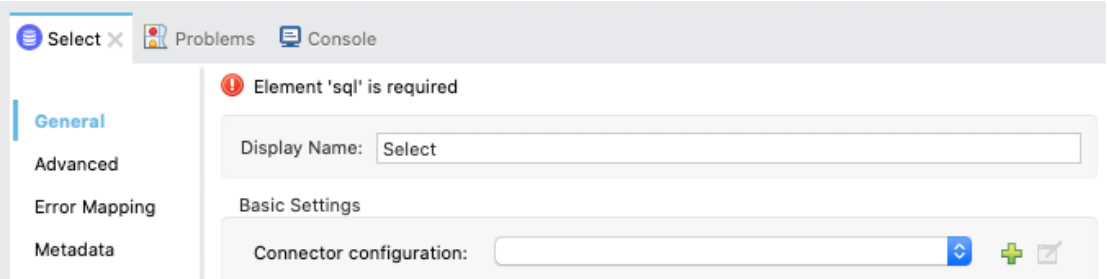
32. From this same terminal, rerun the mulesoft-training-services.jar file.

```
java –jar mulesoft-training-services-X.X.X.jar
```

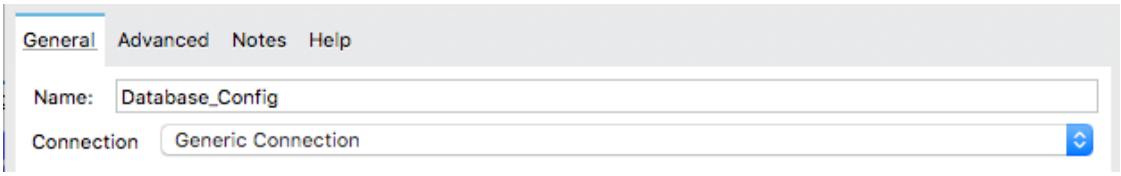
33. Look at the output and make sure all the services started.

Option 2 (continued): Configure a Derby Database connector

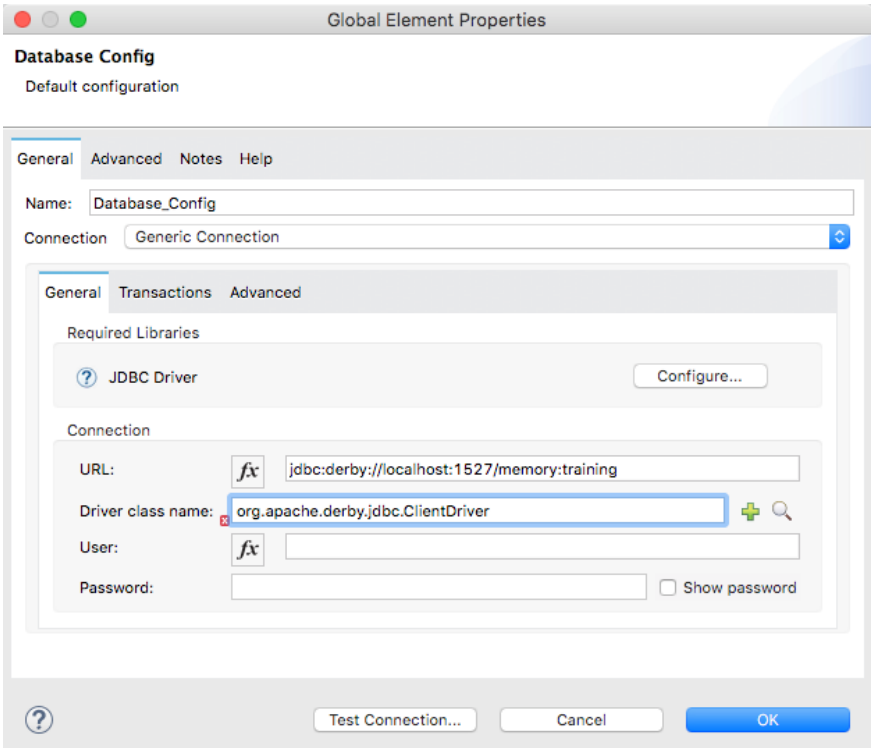
34. Return to Anypoint Studio.
35. In the Select properties view, click the Add button next to connector configuration.



36. In the Global Element Properties dialog box, set the Connection to Generic Connection.



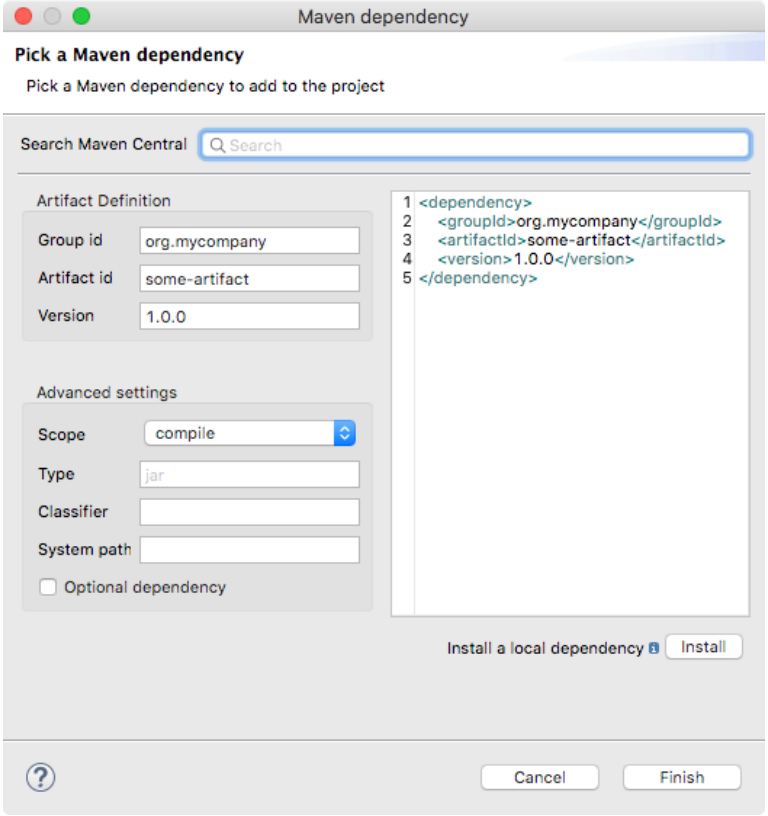
37. Set the URL and driver class name values to the Derby database values listed in the course snippets.txt file.



38. Click the Configure button next to JDBC Driver.

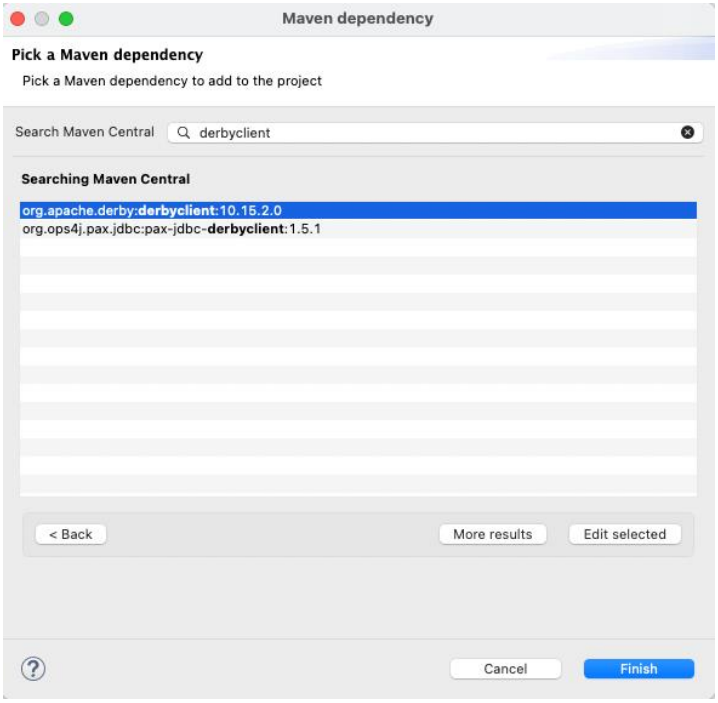
39. In the configure drop-down menu, select Add Maven dependency.

40. In the Maven dependency dialog box, locate the Search Maven Central text field.



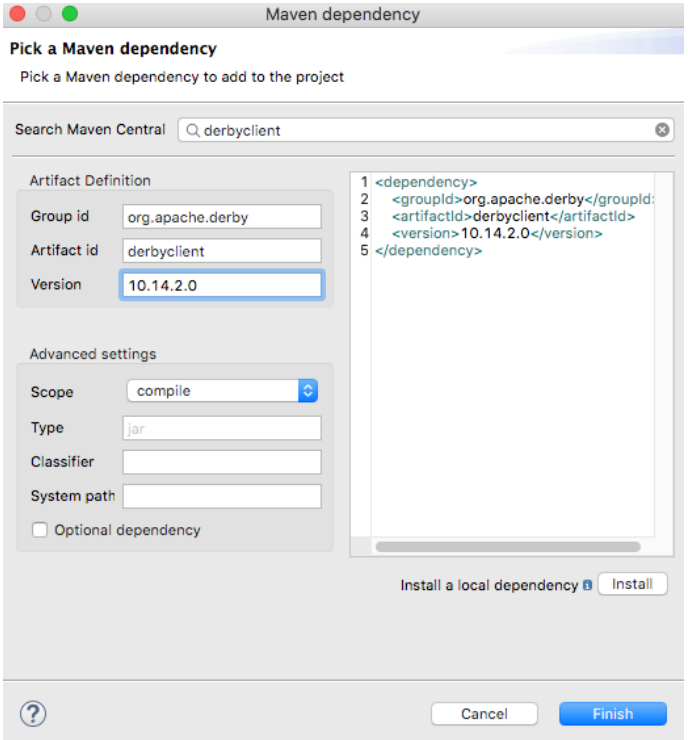
41. Enter derbyclient in the Search Maven Central text field.

42. Select org.apache.derby:derbyclient in the results that are displayed.



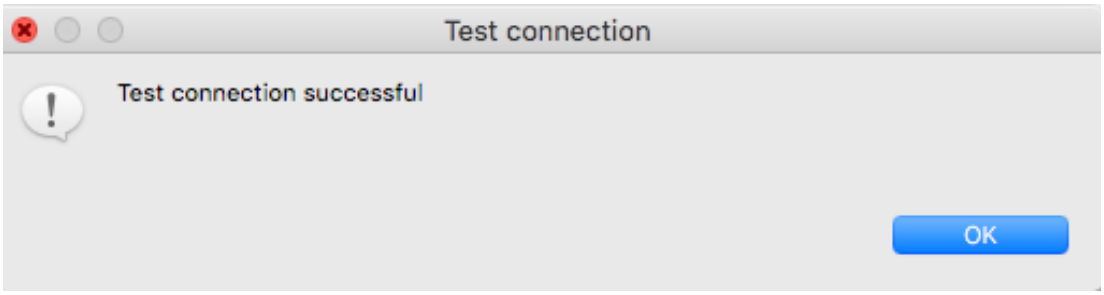
43. Click Edit selected.
44. Enter 10.14.2.0 in the Version text field.

Note: Version 10.14.2.0 is the latest Derby client compatible with Java 8 which is used by the Mule runtime.



45. Click Finish.
46. Back in the Global Element Properties dialog box, click the Test Connection button; you should get a successful test dialog box.

Note: Make sure the connection succeeds before proceeding.

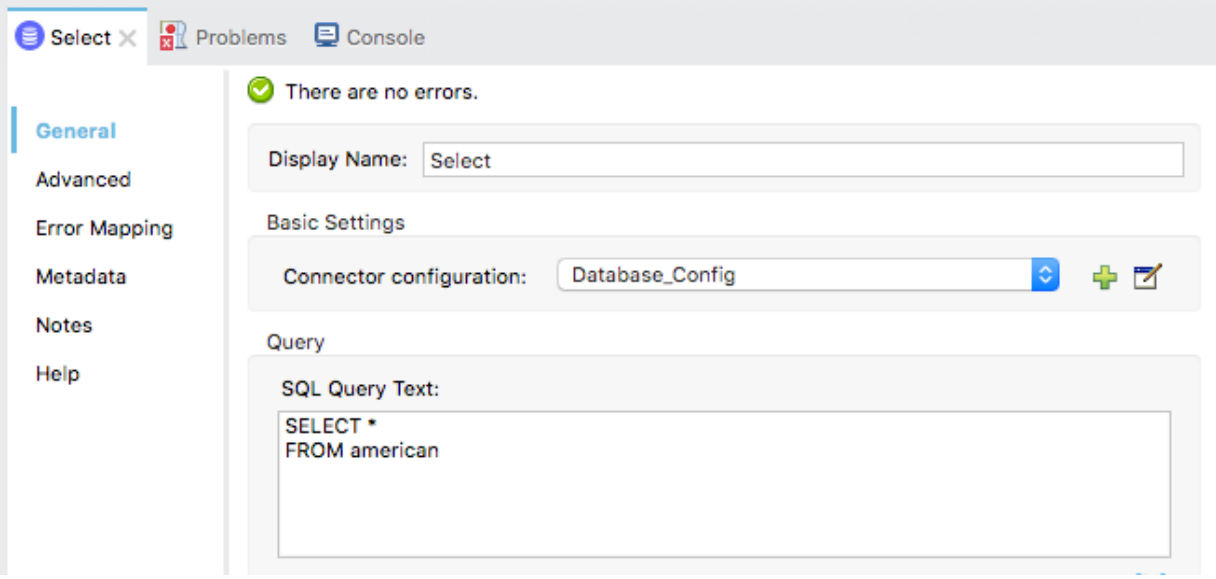


47. Click OK to close the Test connection dialog box.
48. Click OK to close the Global Element Properties dialog box.

Write a query to return all flights

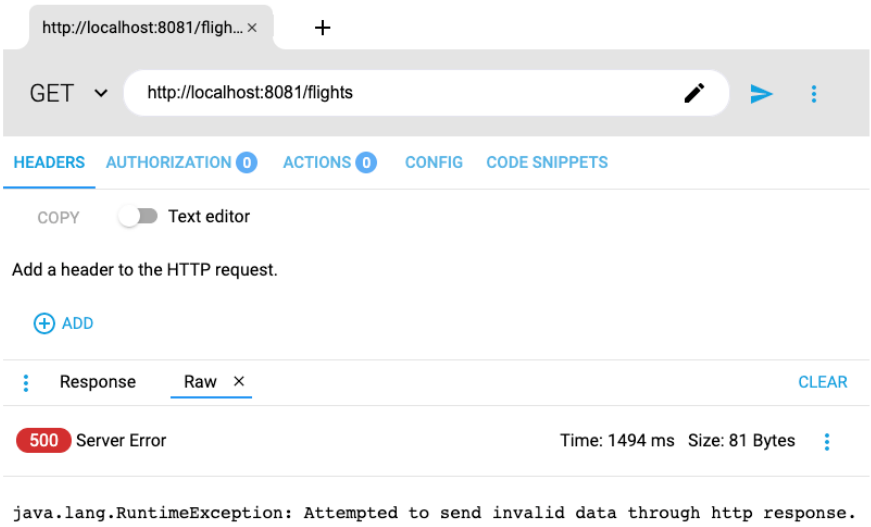
49. In the Select properties view, add a query to select all records from the american table.

```
SELECT *
FROM american
```



Test the application

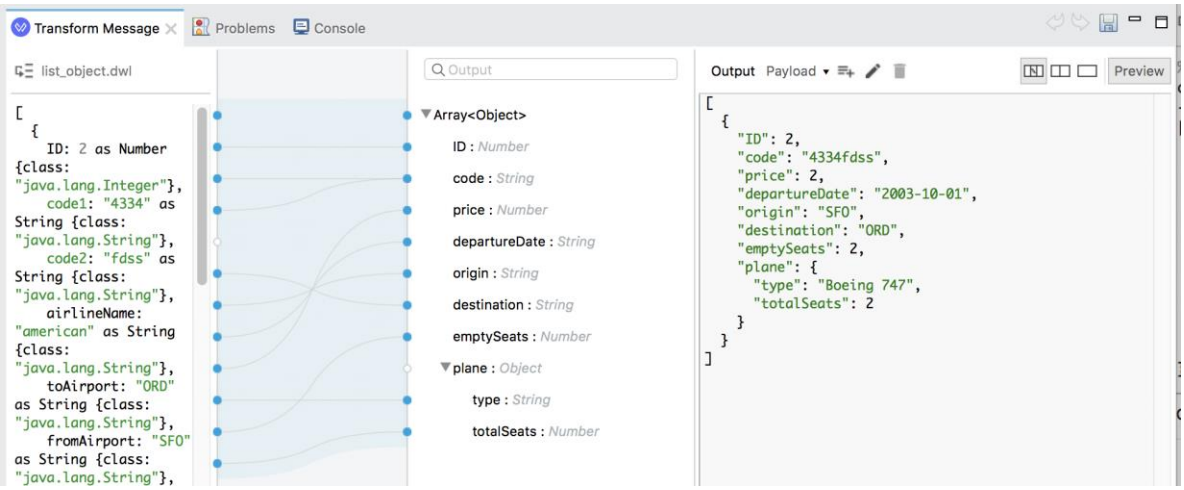
50. Run the project.
51. In the save changes dialog box, click Save.
52. Watch the console and wait for the application to start.
53. Once the application has started, return to Advanced REST Client.
54. In Advanced REST Client, make another request to http://localhost:8081/flights; you should get a 500 Server Error with an invalid data message.



Walkthrough 4-3: Transform data

In this walkthrough, you transform and display the flight data into JSON. You will:

- Use the Transform Message component.
- Use the DataWeave visual mapper to change the response to a different JSON structure.

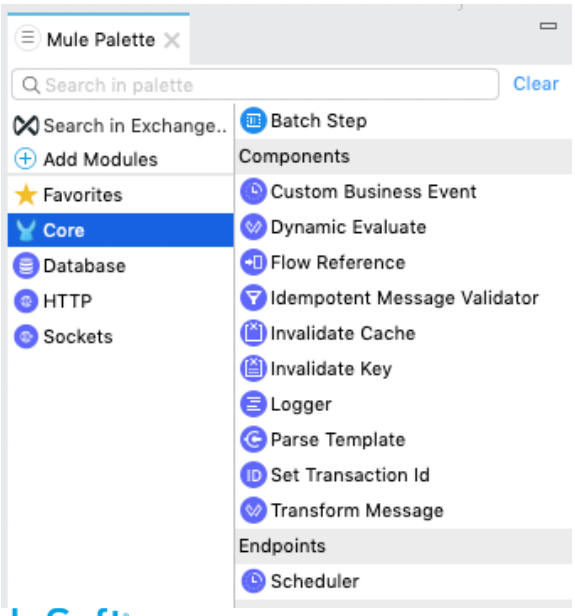


Starting file

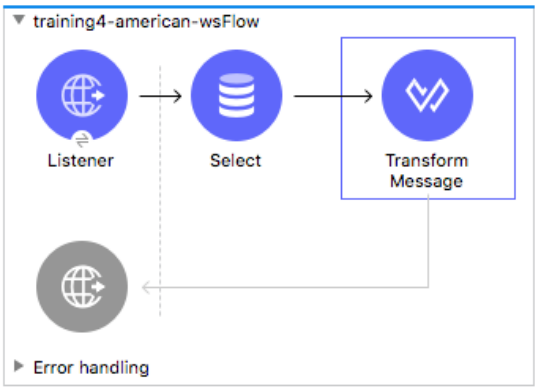
If you did not complete the previous walkthrough, you can get a starting file here. This file is also located in the solutions folder of the student files ZIP located in the Course Resources.

Add a Transform Message component

1. Return to Anypoint Studio.
2. In the Mule Palette, select Core and locate the Transform Message component in the Components section.



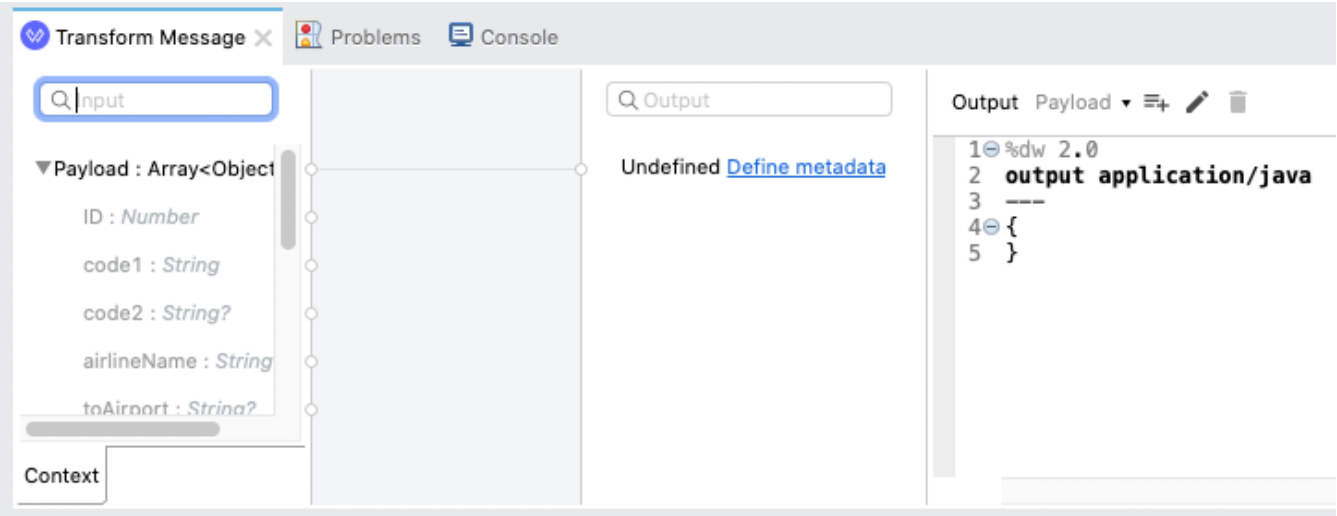
3. Drag the Transform Message component and drop it after the Select processor.



Review metadata for the transformation input

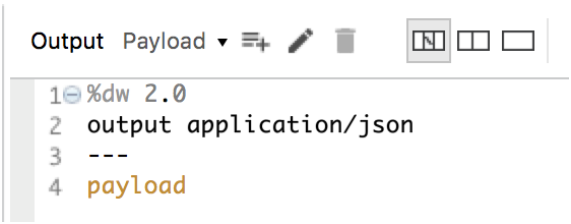
4. In the Transform Message properties view, look at the input section and review the payload metadata.

Note: If you are using the local Derby database, the properties will be uppercase instead.



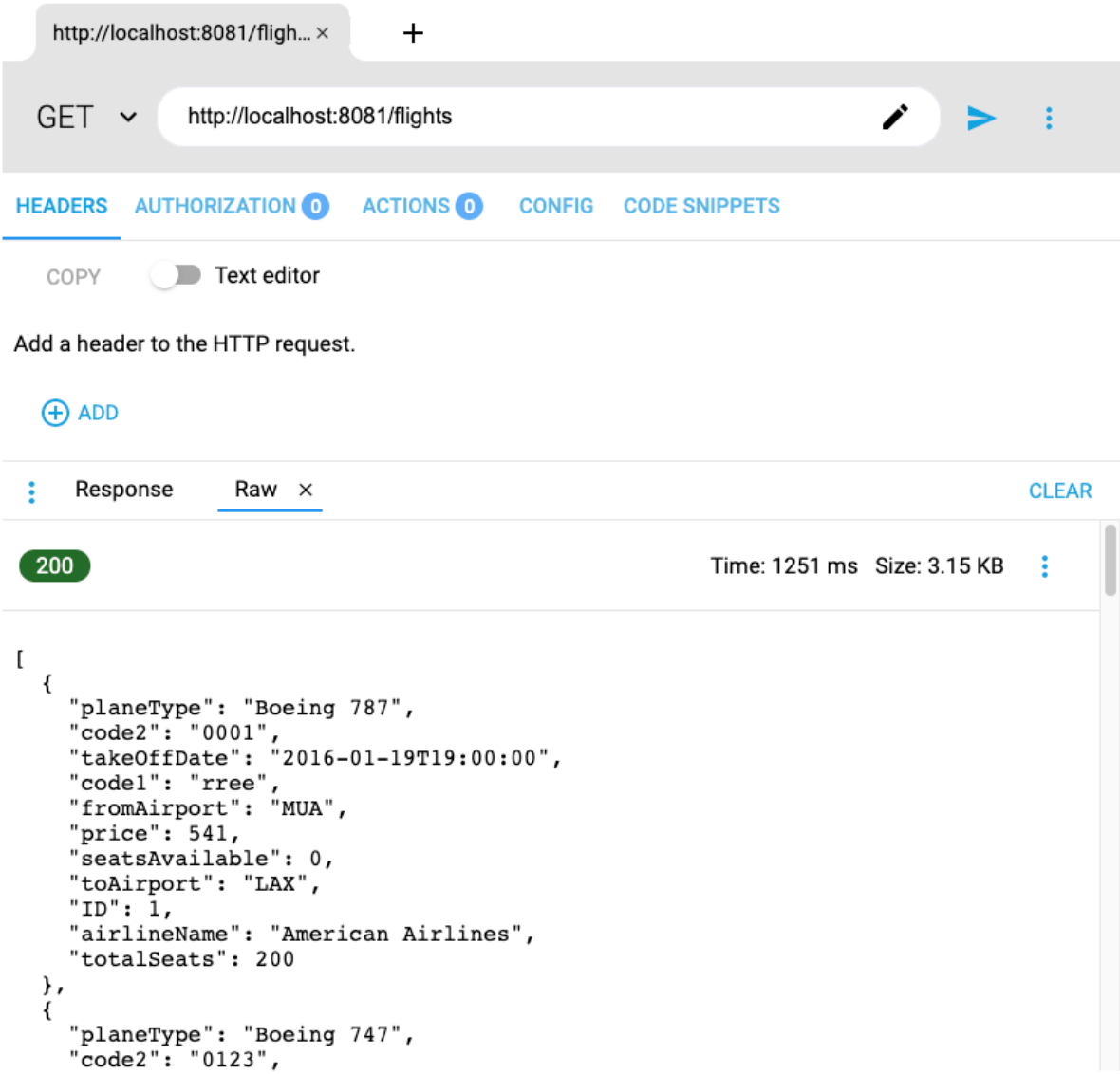
Return the payload as JSON

5. In the Transform Message properties view, change the output type from application/java to application/json and change the {} to payload.



Test the application

6. Save the file to redeploy the project.
7. In Advanced REST Client, send the same request; you should get a 200 response and the American flight data represented as JSON.



Note: If you are using the local Derby database, the properties will be uppercase instead.

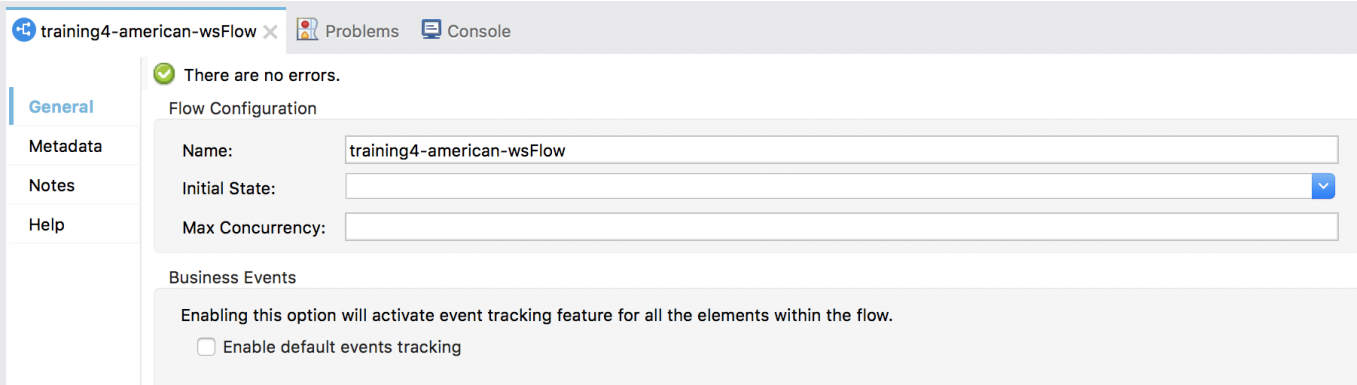
Review the data structure to be returned by the American flights API

8. Return to your American Flights API in Exchange.
9. Look at the example data returned for the /flights GET method.

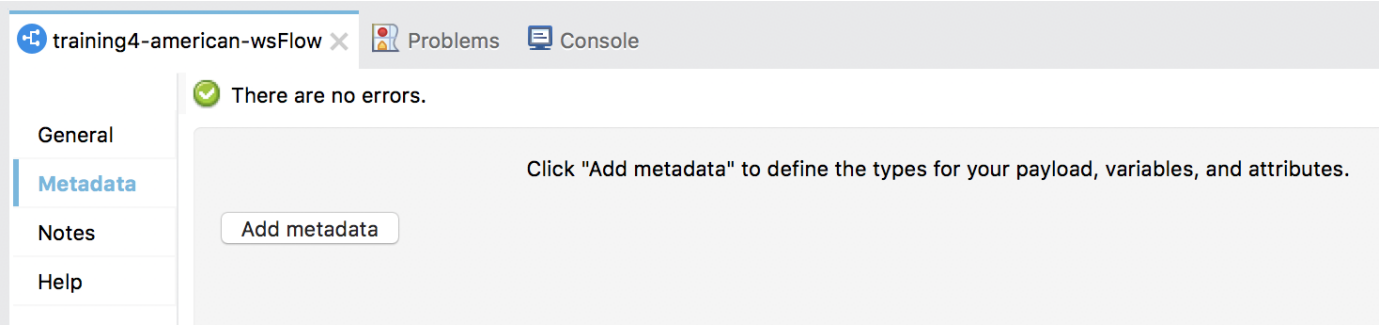
10. Notice that the structure of the JSON being returned by the Mule application does not match this example JSON format.

Define metadata for the data structure to be returned by the American flights API

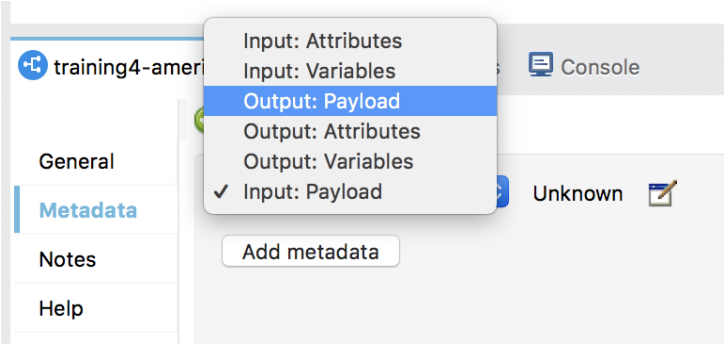
11. Return to Anypoint Studio.
12. In the canvas, click the training4-american-wsFlow name.
13. In the training4-american-wsFlow properties view, select the Metadata tab.



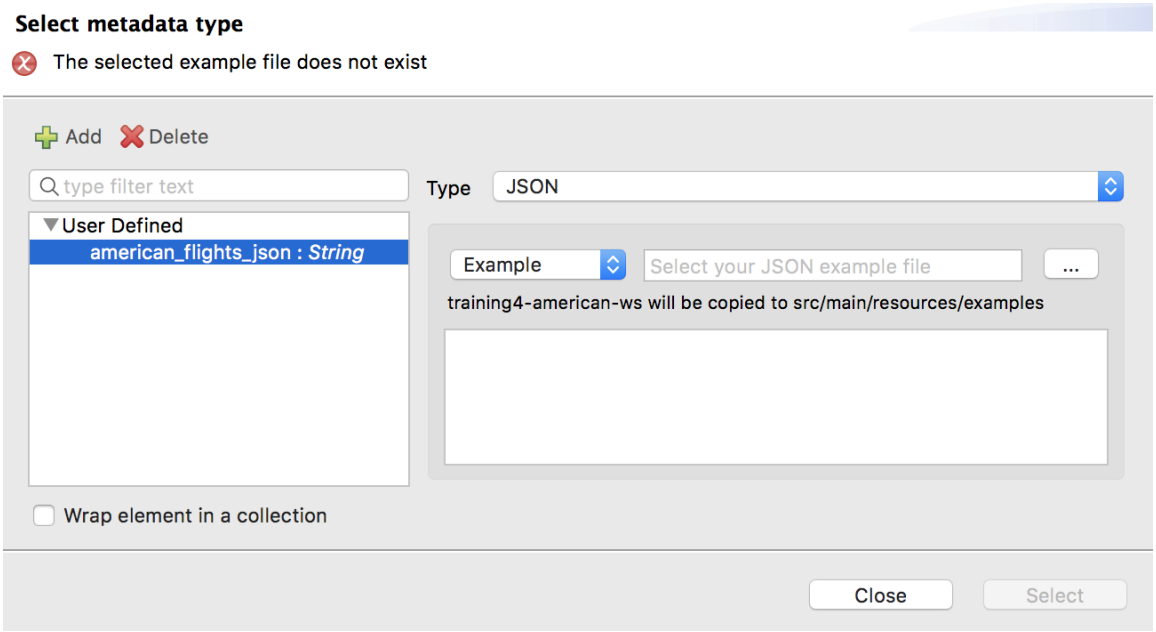
14. Click the Add metadata button.



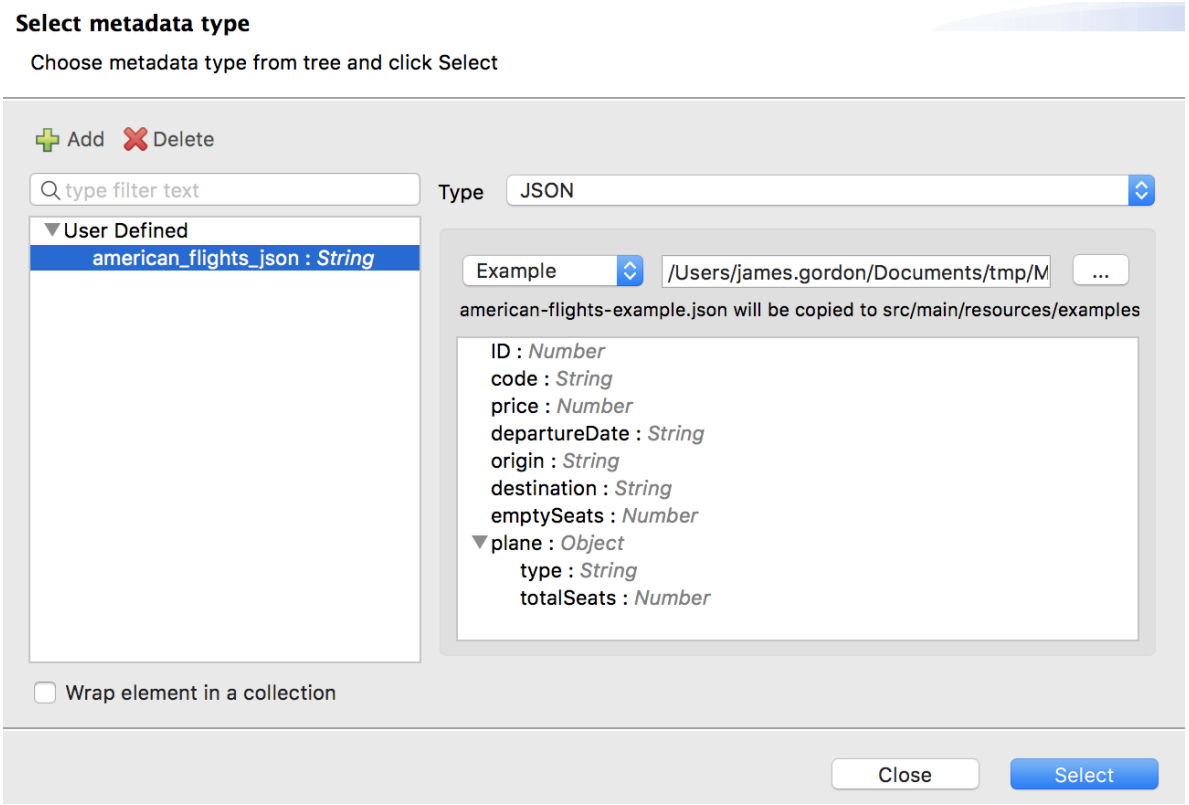
15. In the drop-down menu, select Output: Payload.



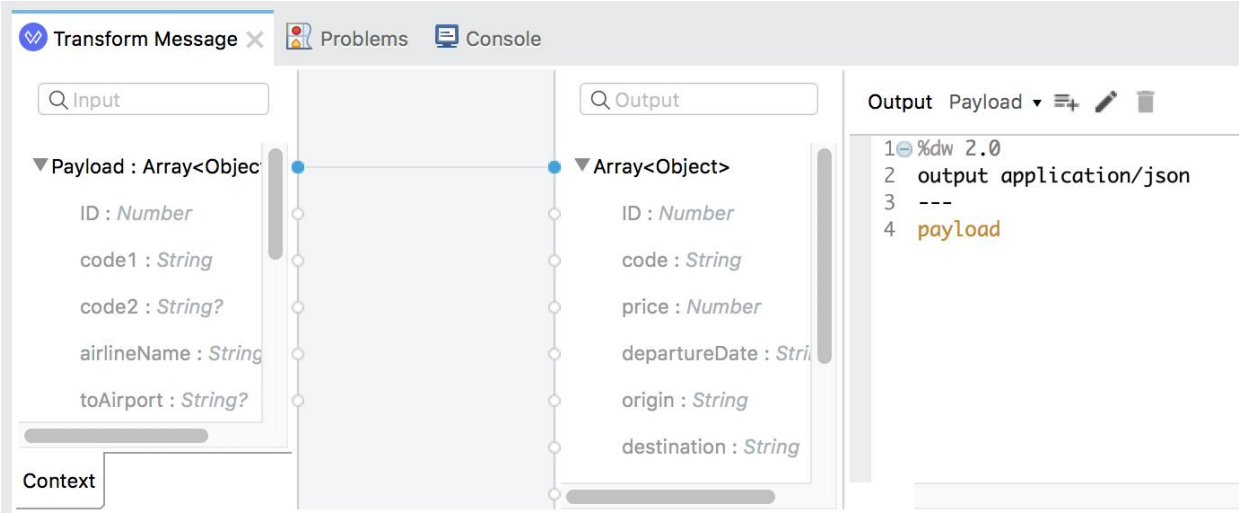
16. Click the edit button.
17. In the Select metadata type dialog box, click the Add button.
18. In the Create new type dialog box, set the type id to american_flights_json.
19. Click Create type.
20. Back in the Select metadata type dialog box, set the first type to JSON.
21. Change the Schema selection to Example.



22. Click the browse button and navigate to the student files.
23. Select american-flights-example.json in the resources/examples folder and click Open; you should see the example data for the metadata type.



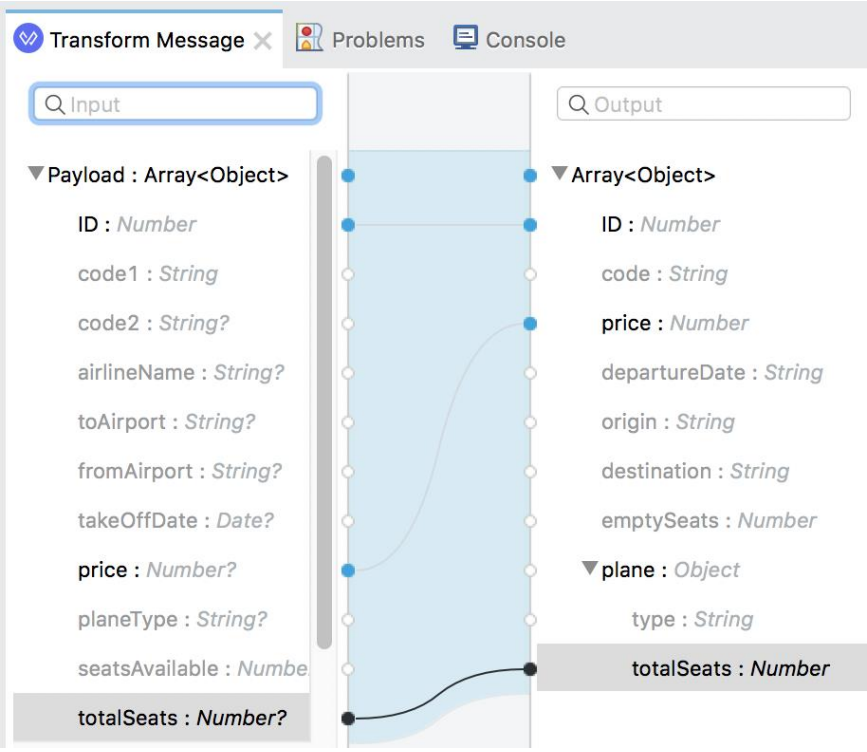
24. Click Select.
25. In training4-american-wsFlow, click the Transform Message component; you should now see output metadata in the output section of the Transform Message properties view.



Create the transformation

26. Map fields with the same names by dragging them from the input section and dropping them on the corresponding field in the output section.

- ID to ID
- price to price
- totalSeats to plane > totalSeats

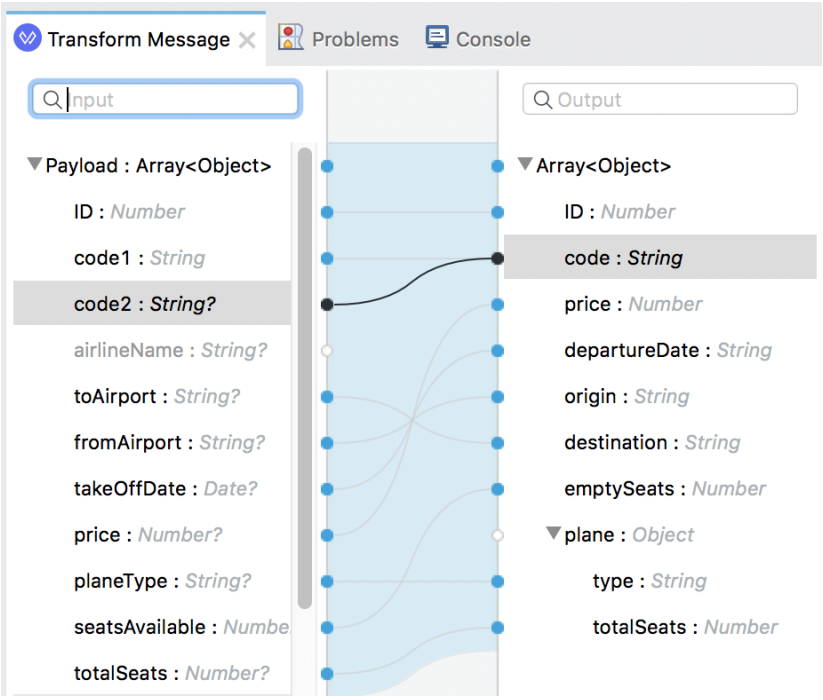


27. Map fields with different names by dragging them from the input section and dropping them on the corresponding field in the output section.

- toAirport to destination
- takeOffDate to departureDate
- fromAirport to origin
- seatsAvailable to emptySeats
- planeType to plane > type

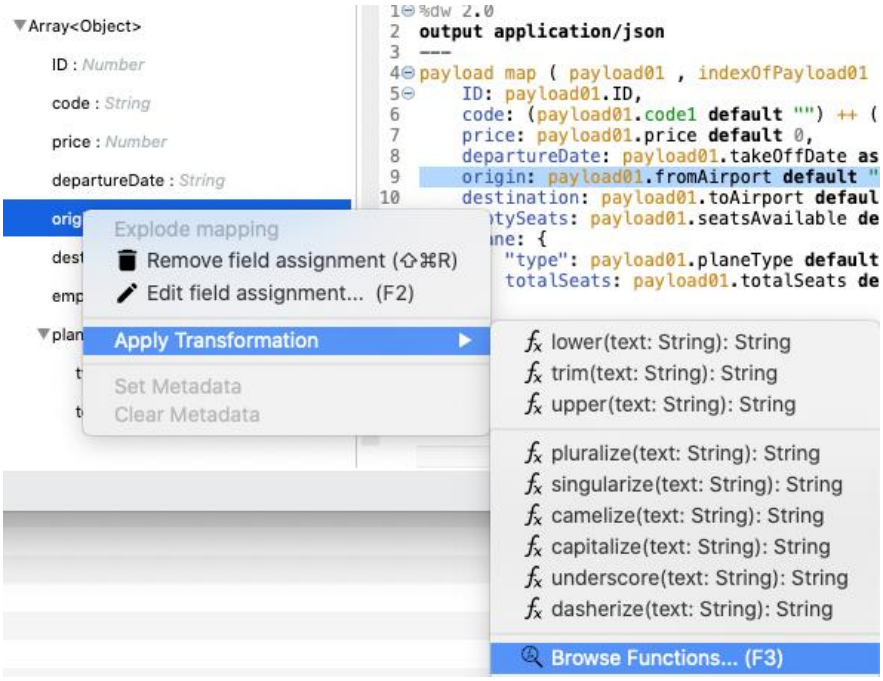
28. Concatenate two fields by dragging them from the input section and dropping them on the same field in the output section.

- code1 to code
- code2 to code

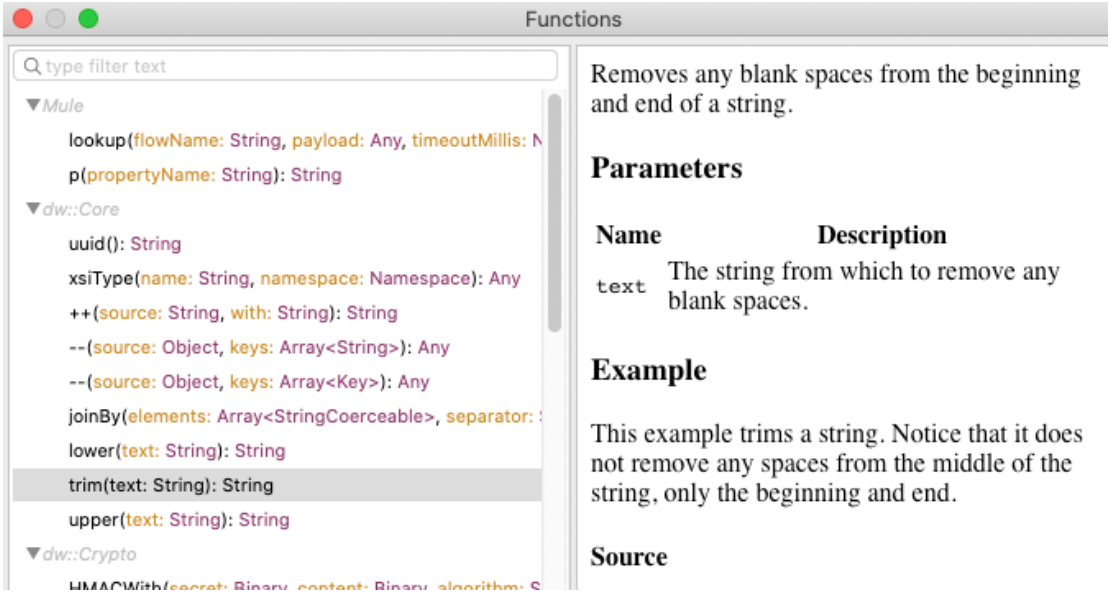


Browse functions and operators

29. Right-click origin in the output section, select Apply Transformation then select Browse Functions.

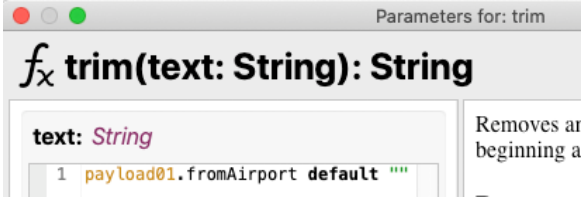


30. In the Functions dialog box, select the trim function under dw:Core and examine the documentation for it on the right side.



31. Click Insert.

32. In the Parameters for: trim dialog box, locate the editable contents of the text: String section.

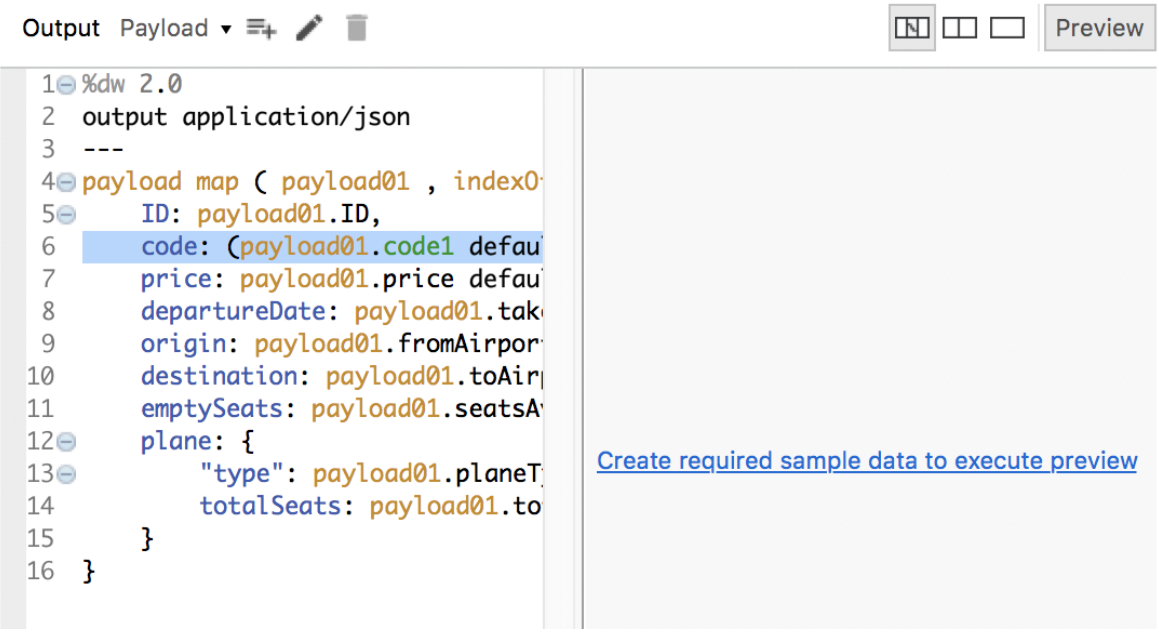


33. Click Cancel.

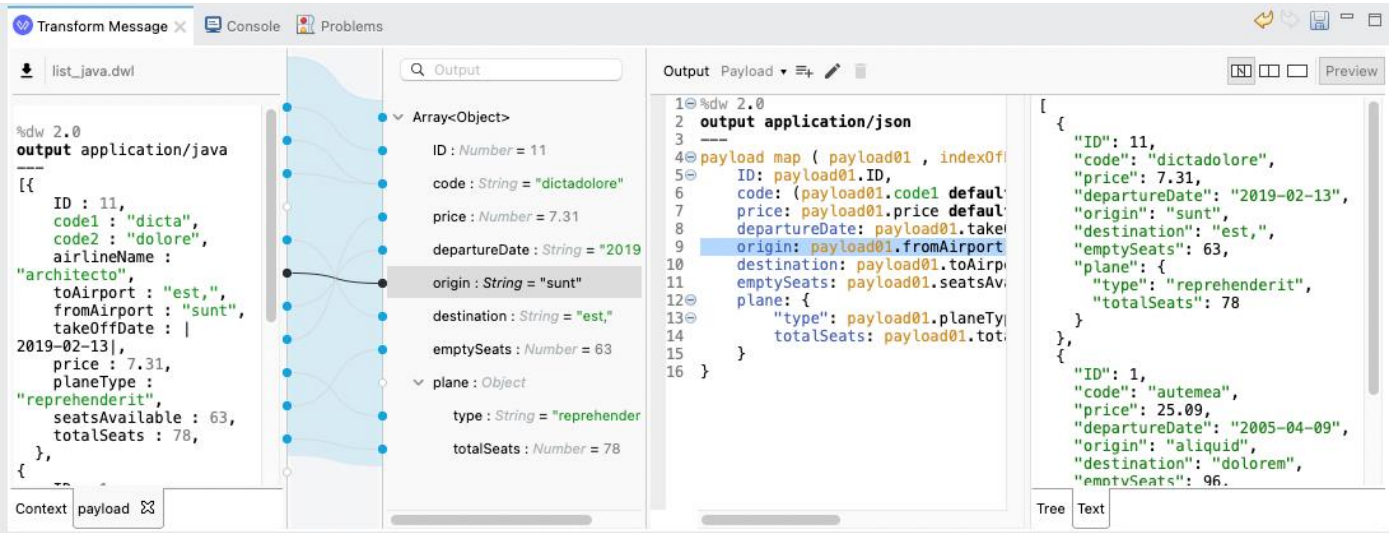
Note: You would normally select Finish to incorporate the selected function or operator into your transformation.

Add sample data (optional)

34. Click the Preview button in the output section.
35. In the preview section, click the Create required sample data to execute preview link.



36. Look at the input section, you should see a new tab called payload with sample data generated from the input metadata.
37. Look at the output section, you should see a sample response for the transformation.



Test the application

38. Save the file to redeploy the project.
39. In Advanced REST Client, make another request to http://localhost:8081/flights; you should see all the flight data as JSON again but now with the example JSON format.

200


```
[
  {
    "ID": 1,
    "code": "rree0001",
    "price": 541,
    "departureDate": "2016-01-19T19:00:00",
    "origin": "MUA",
    "destination": "LAX",
    "emptySeats": 0,
    "plane": {
      "type": "Boeing 787",
      "totalSeats": 200
    }
  },
  {
    "ID": 2,
    "code": "eefd0123",
```


Try to retrieve information about a specific flight

40. Add a URI parameter to the URL to make a request to http://localhost:8081/flights/3; you should get a 404 Not Found response with a no listener message.

404 Not Found

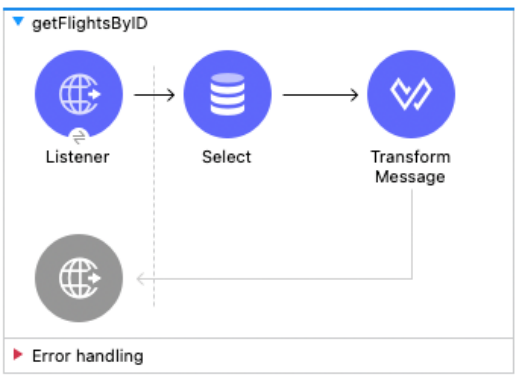
```
No listener for endpoint: /flights/3
```

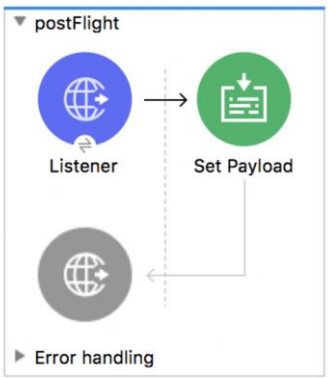
41. Return to Anypoint Studio.
42. Look at the console; you should get a no listener found for request (GET)/flights/3.

Walkthrough 4-4: Create a RESTful interface for a Mule application

In this walkthrough, you continue to create a RESTful interface for the application. You will:

- Route based on path.
- Use a URI parameter in the path of a new HTTP Listener.
- Route based on HTTP method.



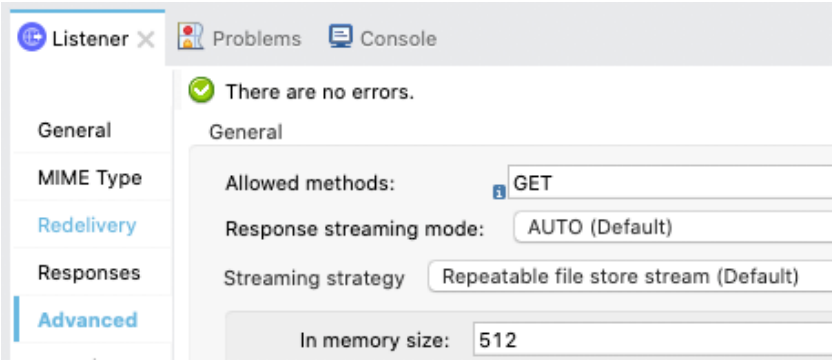


Starting file

If you did not complete the previous walkthrough, you can get a starting file here. This file is also located in the solutions folder of the student files ZIP located in the Course Resources.

Restrict method calls to GET

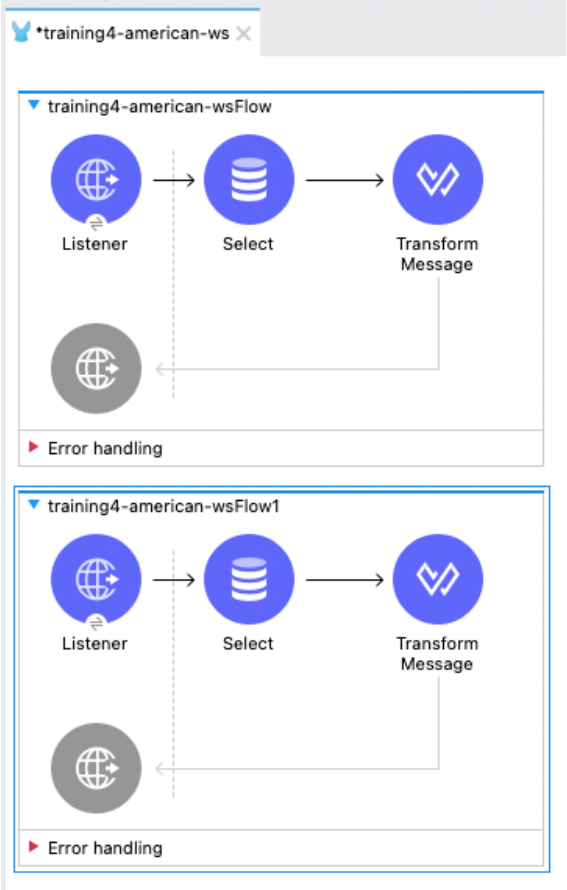
1. Return to Anypoint Studio.
2. Double-click the HTTP Listener in the flow.
3. In the left-side navigation of the Listener properties view, select Advanced.
4. Set the allowed methods to GET.



Make a copy of the existing flow

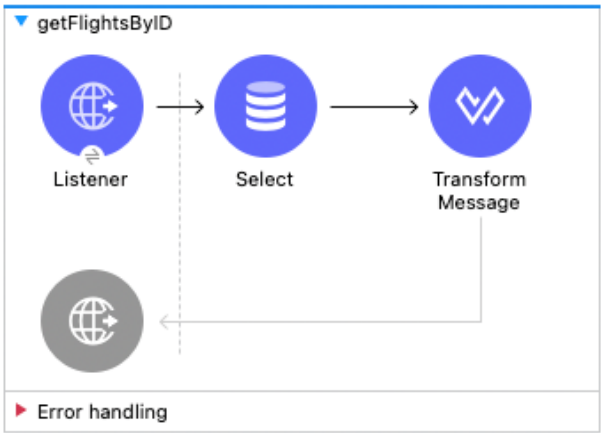
5. Click the flow in the canvas to select it.

6. From the main menu bar, select Edit > Copy.
7. Click in the canvas beneath the flow and select Edit > Paste.



Rename the flows

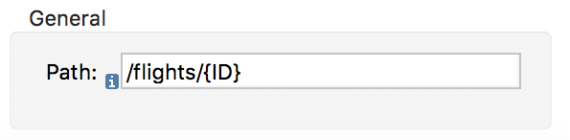
8. Double-click the first flow.
9. In the properties view, change its name to getFlights.
10. Change the name of the second flow to getFlightsByID.



Note: If you want, change the name of the event source and event processors.

Specify a URI parameter for the new HTTP Listener endpoint

11. Double-click the HTTP Listener in getFlightsByID.
12. Modify the path to have a URI parameter called ID.



Modify the Database endpoint

13. Double-click the Select operation in getFlightsByID.
14. Modify the query WHERE clause, to select flights with the ID equal to 1.

```
SELECT *
FROM american
WHERE ID = 1
```

Test the application

15. Save the file to redeploy the project.
16. In Advanced REST Client, make another request to http://localhost:8081/flights/3; you should see details for the flight with an ID of 1.

200


```
[
  {
    "ID": 1,
    "code": "rree0001",
    "price": 541,
    "departureDate": "2016-01-19T19:00:00",
    "origin": "MUA",
    "destination": "LAX",
    "emptySeats": 0,
    "plane": {
      "type": "Boeing 787",
      "totalSeats": 200
    }
  }
]
```


Note: You did not add logic to the application to search for a flight with a particular ID. You will deploy an application with this additional functionality implemented next.

Modify the database query to use the URI parameter

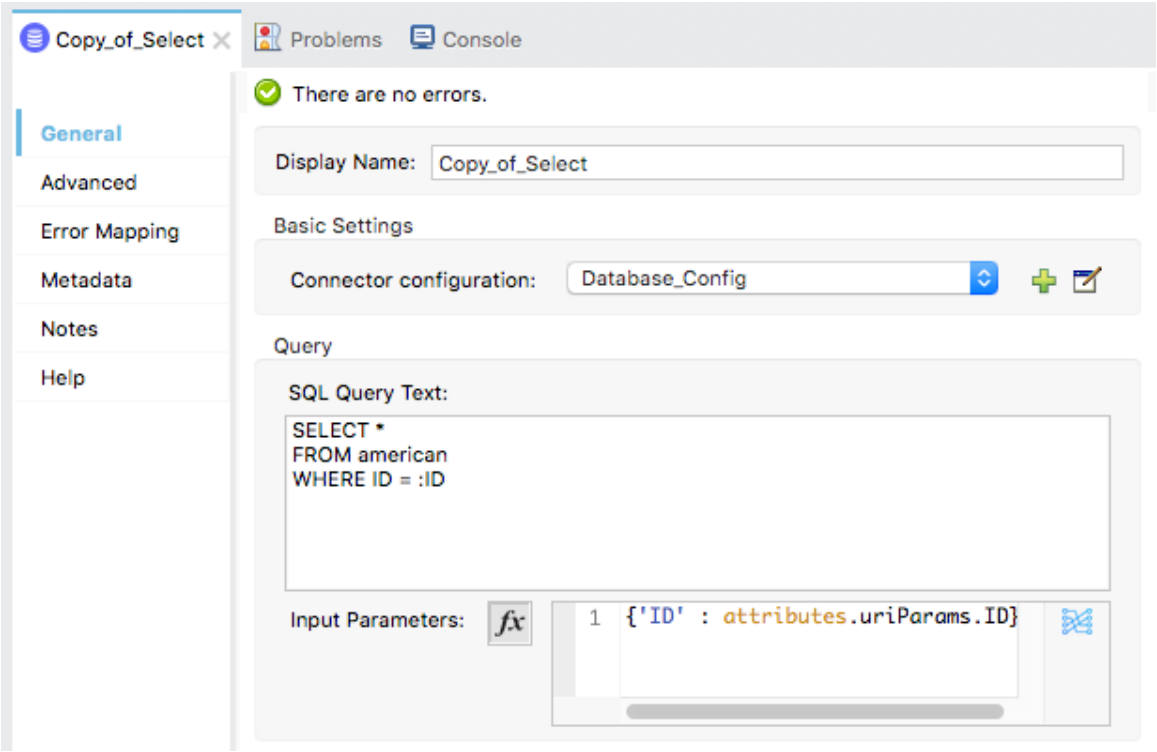
17. Return to the course snippets.txt file and copy the SQL input parameter expression.
18. Return to the getFlightsByID flow in Anypoint Studio.
19. In the Select properties view, locate the Query Input Parameters section, click the Switch to expression mode button, then paste the expression you copied.

```
{'ID' : attributes.uriParams.ID}
```

Note: You learn to write expressions in the Development Fundamentals course.

20. Change the WHERE clause in the SQL Query Text to use this input parameter.

```
SELECT *
FROM american
WHERE ID = :ID
```



Test the application

21. Save the file to redeploy the project.
22. In Advanced REST Client, make another request to http://localhost:8081/flights/3; you should now see the info for the flight with an ID of 3.

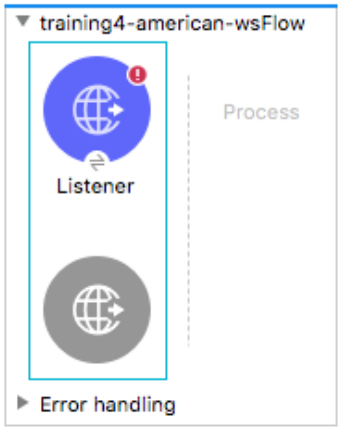
200


```
[
  {
    "ID": 3,
    "code": "ffee0192",
    "price": 300,
    "departureDate": "2016-01-19T19:00:00",
    "origin": "MUA",
    "destination": "LAX",
    "emptySeats": 0,
    "plane": {
      "type": "Boeing 777",
      "totalSeats": 300
    }
  }
]
```


23. Return to Anypoint Studio.

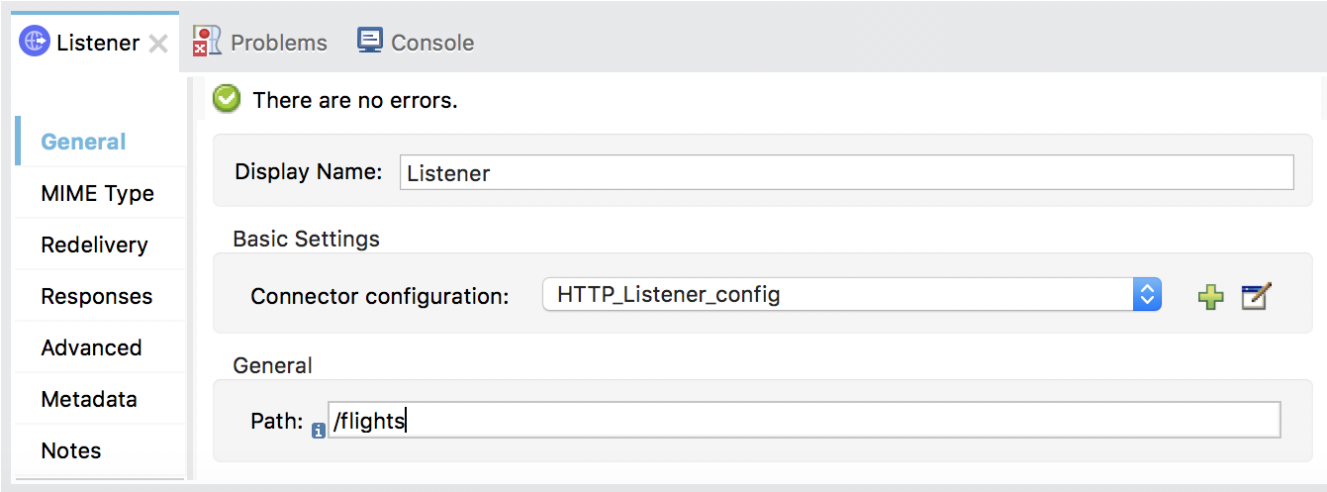
Make a new flow to handle post requests

24. In the Mule Palette, select HTTP.
25. Drag Listener from the Mule Palette and drop it in the canvas below the two existing flows.

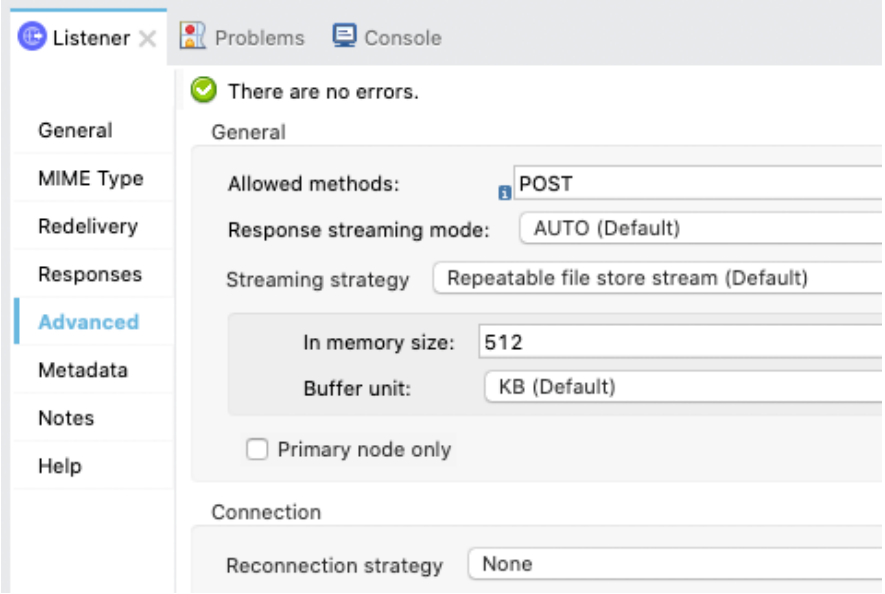


26. Change the name of the flow to postFlight.
27. In the Listener properties view, ensure the connector configuration is set to the existing HTTP_Listener_config.

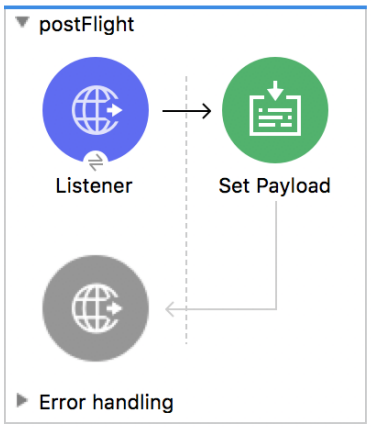
28. Set the path to /flights.



29. In the left-side navigation of the Listener properties view, select Advanced.
30. Set the allowed methods to POST.



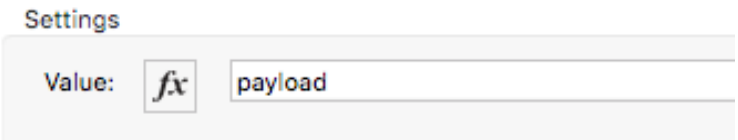
31. Drag the Set Payload transformer from the Mule Palette and drop it in the process section of the flow.



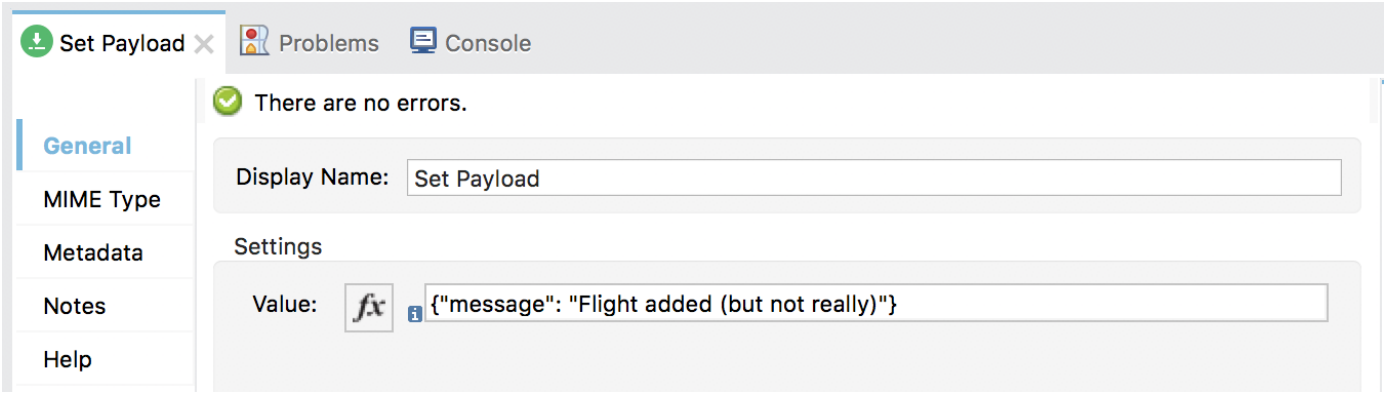
32. Return to the course snippets.txt file and copy the American Flights API - /flights POST response example.

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

33. Return to Anypoint Studio and in the Set Payload properties view, click the Switch to literal mode button for the value field.



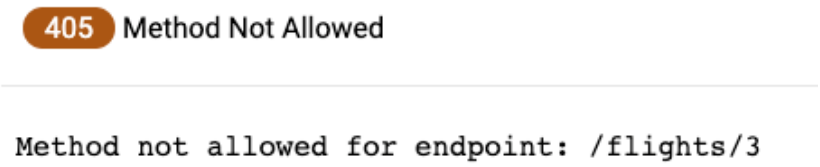
34. Set the value field to the value you copied.



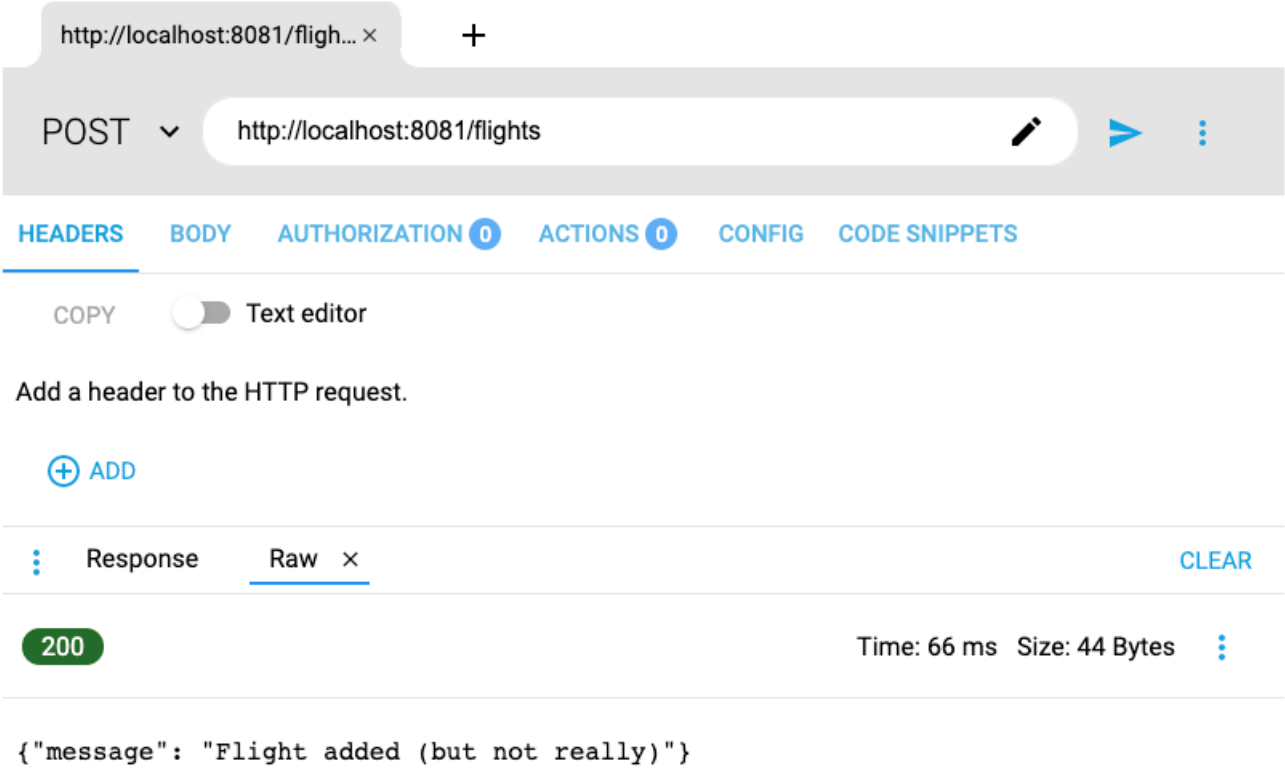
Note: This flow is just a stub. For it to really work and add data to the database, you would need to add logic to insert the request data to the database.

Test the application

35. Save the file to redeploy the project.
36. In Advanced REST Client, change the request type from GET to POST.
37. Click Send; you should get a 405 Method Not Allowed response.



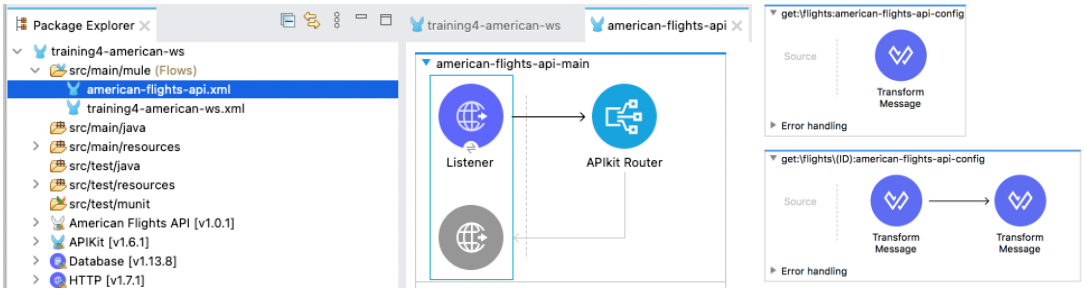
38. Remove the URI parameter from the request URL: http://localhost:8081/flights.
39. Send the request; you should now see the message the flight was added – even though you did not send any flight data to add.



Walkthrough 4-5: Use Anypoint Studio to create a RESTful API interface from a RAML file

In this walkthrough, you generate a RESTful interface from the RAML file. You will:

- Add Anypoint Platform credentials to Anypoint Studio.
- Import an API from Exchange into an Anypoint Studio project.
- Use APIkit to generate a RESTful web service interface from an API.
- Test a web service using APIkit console and Advanced REST Client.



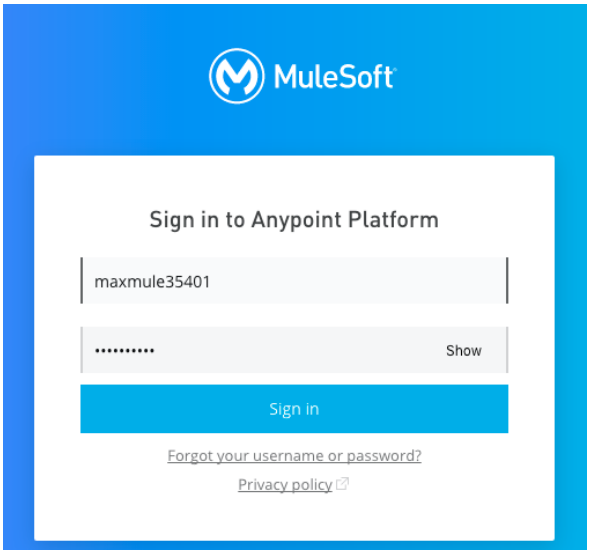
Starting file

If you did not complete the previous walkthrough, you can get a starting file here. This file is also located in the solutions folder of the student files ZIP located in the Course Resources.

Add Anypoint Platform credentials to Anypoint Studio

1. Return to Anypoint Studio.
2. In the Package Explorer, right-click the training4-american-ws project and select Anypoint Platform > Configure Credentials.
3. In the Authentication page of the Preferences dialog box, click the Add button.

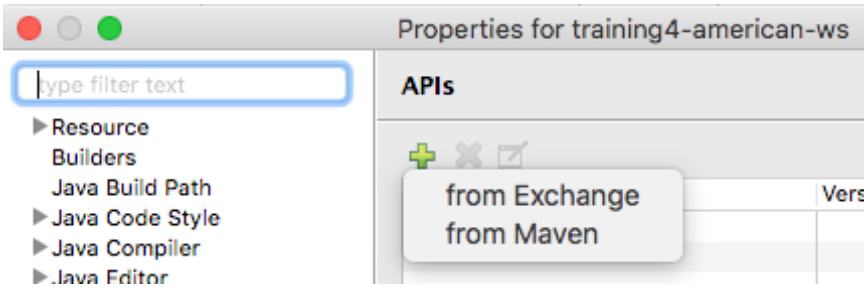
4. In the User login dialog box, enter your username and password and click Sign In.



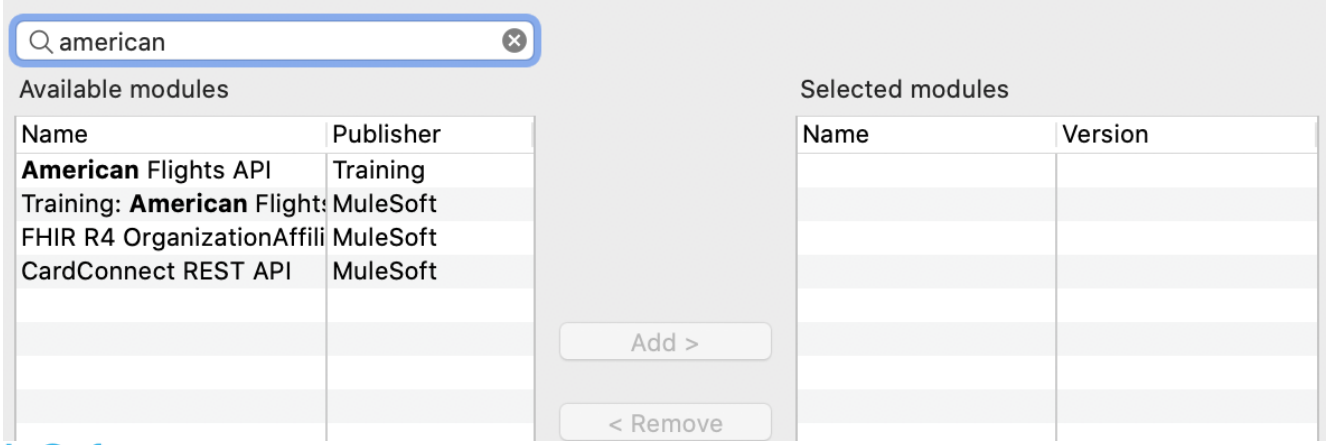
5. On the Authentication page, make sure your username is listed and selected.
6. Click Apply and Close.

Add an API from Exchange to the Anypoint Studio project

7. Right-click the training4-american-ws project and select Manage Dependencies > Manage APIs.
8. In the Properties for training4-american-ws dialog box, click the Add button and select from Exchange.



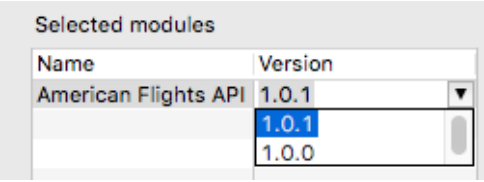
9. In the Add Dependencies to Project dialog box, enter american in the search field.



10. Select your American Flights API (**not** the Training: American Flights API) and click Add.

Note: If you did not successfully create the American Flights API in the first part of the course, you can use the pre-built Training: American Flights API connector.

11. In the selected modules, click American Flights API, then the 1.0.1 version then the version's down-arrow.



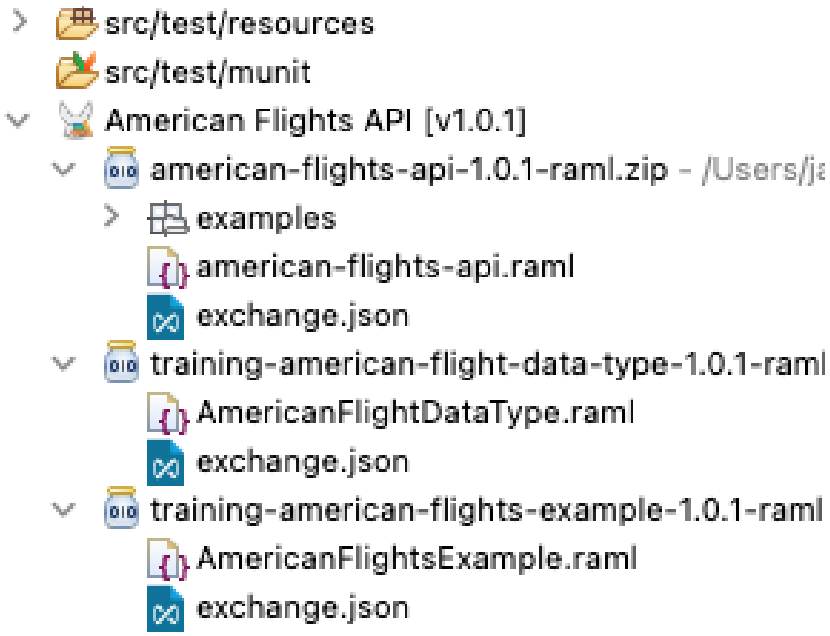
12. Note the list of available versions then select 1.0.1.

Note: If you use the pre-built Training: American Flights API connector, select the latest version.

13. Click Finish.
14. In the Properties for training4-american-ws dialog box, click Apply and Close.
15. Click Yes when prompted to scaffold the American Flights API specification.

Locate the API files added to the project

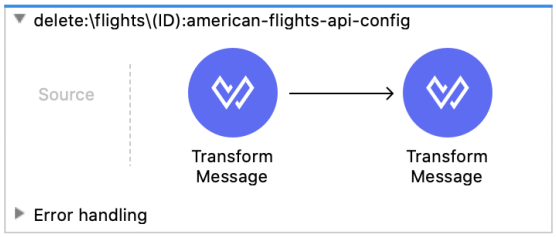
16. In the Package Explorer, locate American Flights API [v1.0.1].
17. Expand the API; you should see the RAML files imported from Exchange.

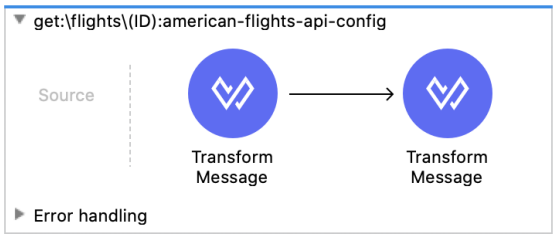


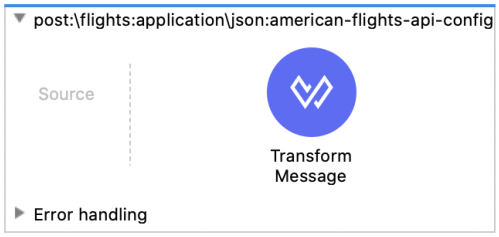
Examine the XML file created

18. Examine the generated american-flights-api.xml file and locate the following four flows:

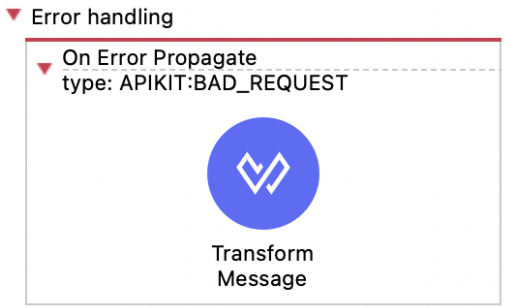
- delete:\flights\{ID}
- get:\flights
- get:\flights\{ID}
- post:\flights



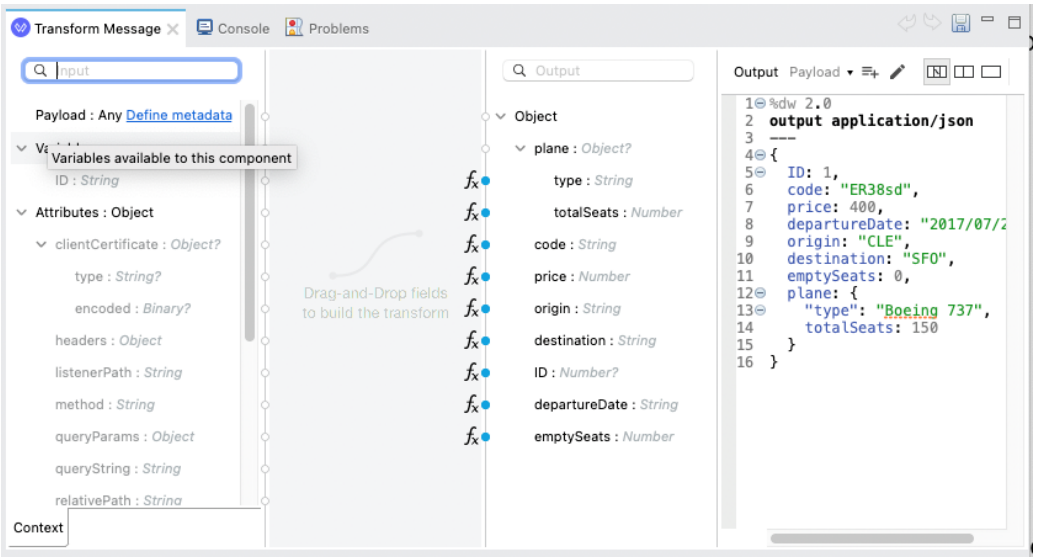




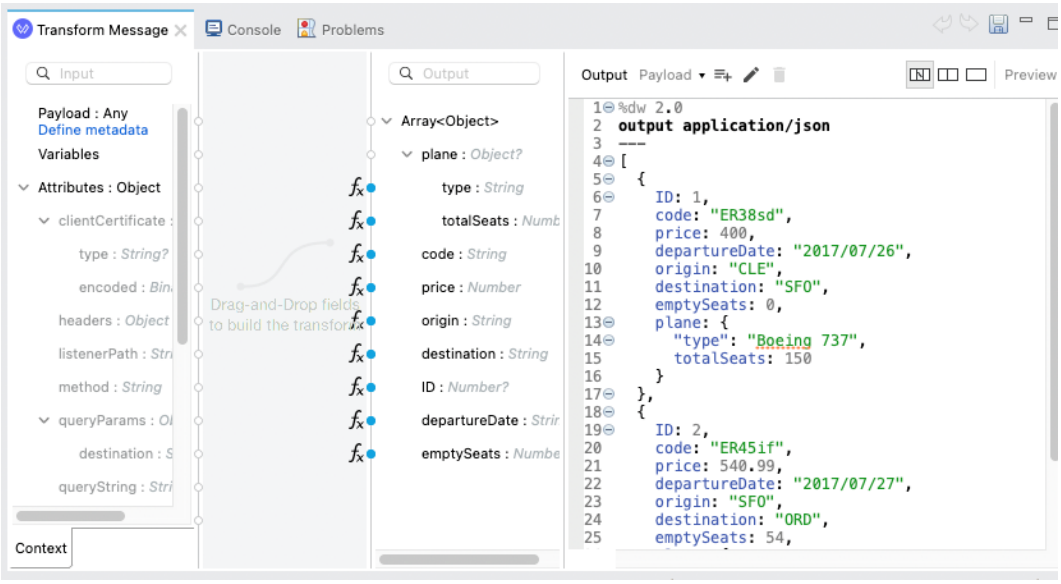
Note: You will also see a series of automatically generated components like the one shown here. These are related to error handling.



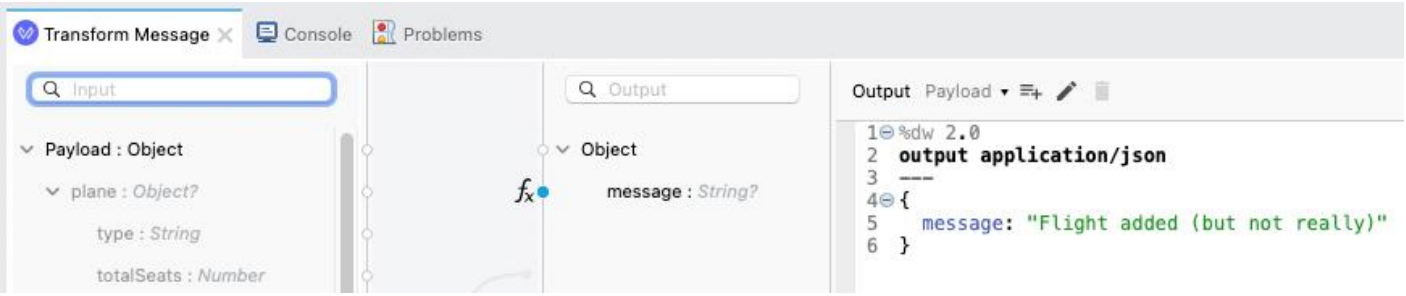
19. In the get:\flights/{ID} flow, double-click the second Transform Message component and look at the output JSON in the properties view.



20. In the get:\flights flow, double-click the Transform Message component and look at the output JSON in the properties view.

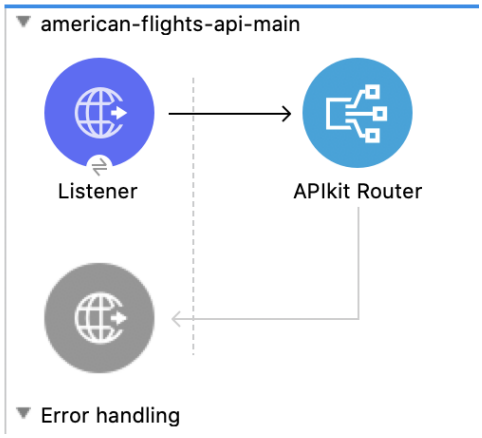


21. In the post:\flights flow, double-click the Transform Message component and look at the output JSON in the properties view.



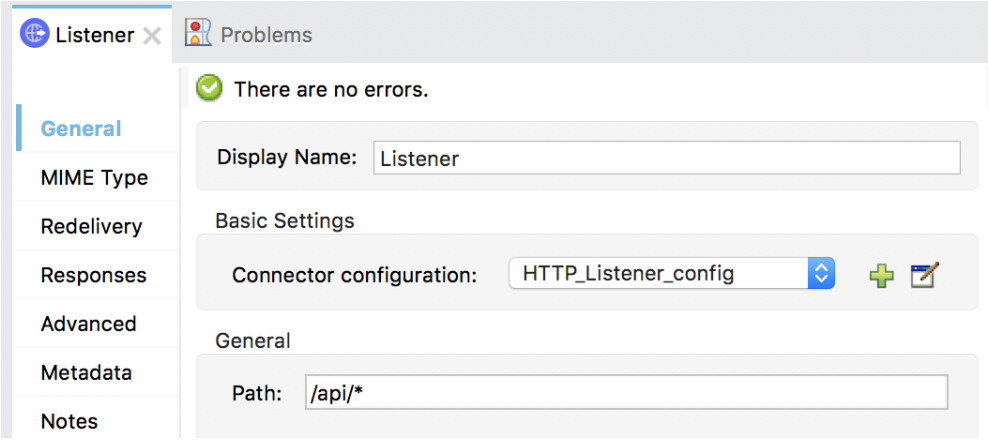
Examine the main flow and its HTTP Listener

22. Locate the american-flights-api-main flow.
23. Double-click its HTTP Listener.



24. In the Listener properties view, notice that the connector configuration is the existing HTTP_Listener_config and that path is set to /api/*.

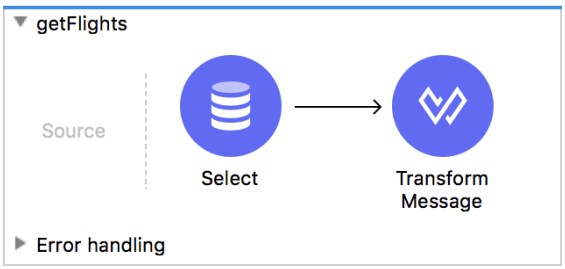
*Note: The * is a wildcard allowing any characters to be entered after /api/.*



25. Click the Edit button for the connector configuration; you should see that the same port 8081 is used as the HTTP Listener you created previously.
26. Click Cancel.

Remove the other listeners

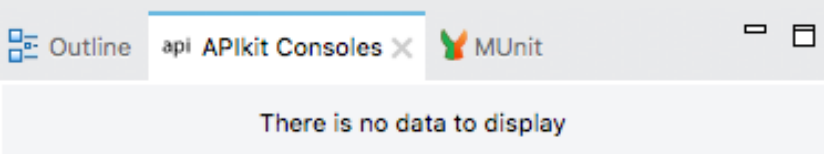
27. Select the Message Flow tab to return to the canvas for training4-american-ws.xml.
28. Right-click the HTTP Listener in getFlights and select Delete.



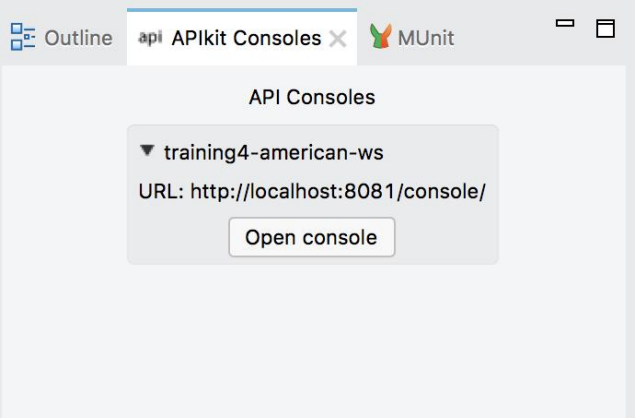
29. Delete the other two HTTP Listeners.

Test the web service using APIkit console

30. Save the files.
31. Locate the new APIkit Consoles view that is created and opened in Anypoint Studio and note that there is no data to display.

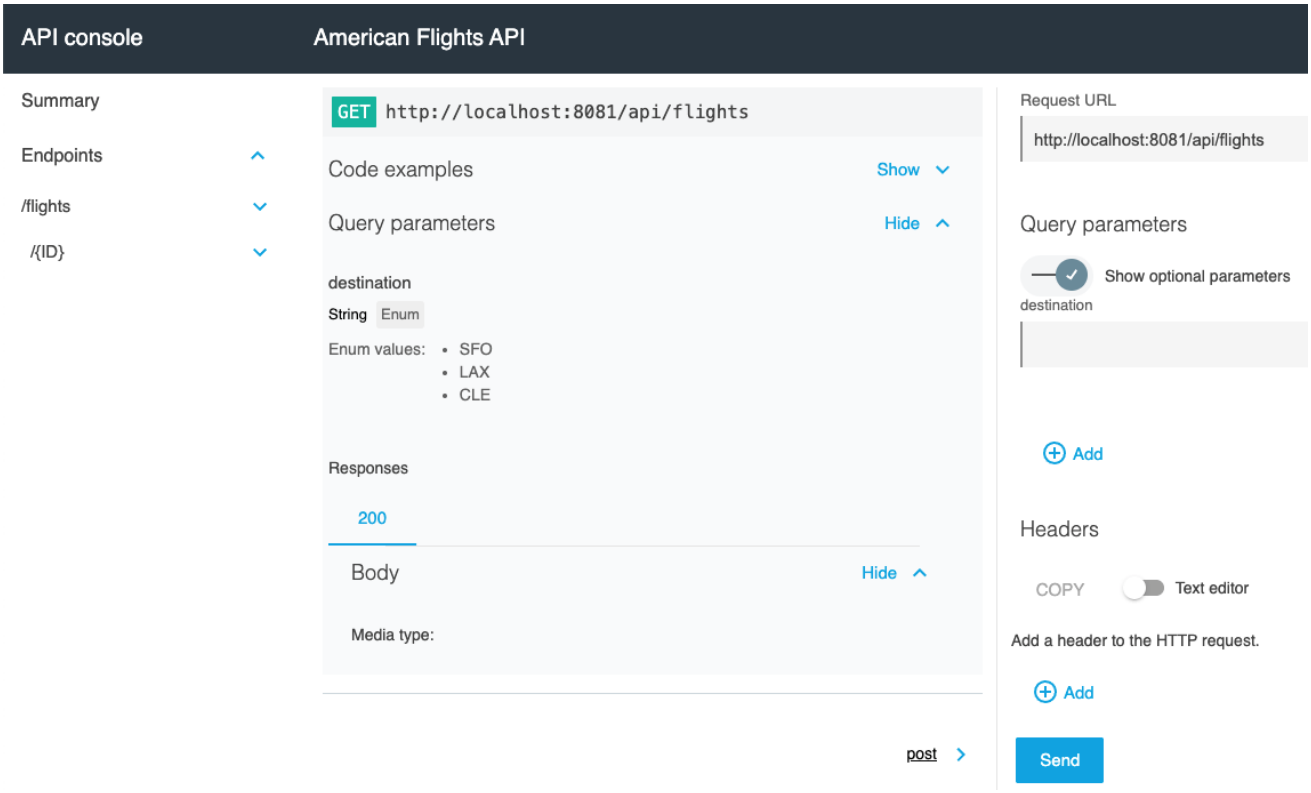


32. Stop the project.
33. Run the project and wait until Mule and the application restart.
34. Verify that the APIkit Consoles view is now populated.



35. Click the Open console button; a browser tab should open with an API console.

36. Select the GET method for /flights and then click the Try it button if it's present.



37. Click Send; you should get a 200 response with the example flight data – not all the flights.

Note: If you use the pre-built Training: American Flights API connector, you must enter any values for client_id and client_secret in the Headers section.

38. Close the browser tab .

Test the web service using Advanced REST Client

39. Return to Advanced REST Client.
40. Change the method to GET and click Send to make a request to http://localhost:8081/flights; you should get a 404 Not Found response.

404 Not Found

```
No listener for endpoint: /flights
```

41. Change the URL to http://localhost:8081/api/flights and send the request; you should get a 200 response with the example flight data.

200 OK


```
[
  {
    "ID": 1,
    "code": "ER38sd",
    "price": 400,
    "departureDate": "2017/07/26",
    "origin": "CLE",
    "destination": "SFO",
    "emptySeats": 0,
    "plane": {
      "type": "Boeing 737",
      "totalSeats": 150
    }
  },
  {
    "ID": 2,
    "code": "ER45if",
    "price": 540.99
```


Note: If you use the pre-built Training: American Flights API connector, you must add client_id and client_secret headers with any values.

42. Make a request to http://localhost:8081/api/flights/3; you should see the example data returned for a flight with an ID of 1.

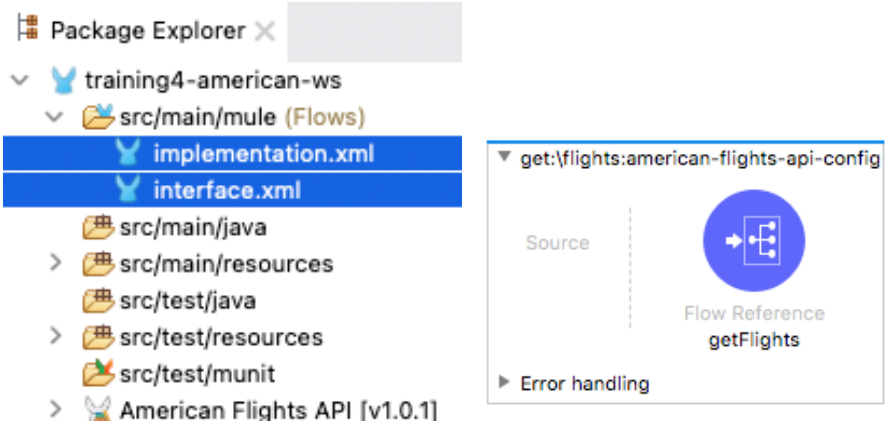
200 OK


```
{
  "ID": 1,
  "code": "ER38sd",
  "price": 400,
  "departureDate": "2017/07/26",
  "origin": "CLE",
  "destination": "SFO",
  "emptySeats": 0,
  "plane": {
    "type": "Boeing 737",
    "totalSeats": 150
  }
}
```

Walkthrough 4-6: Implement a RESTful web service

In this walkthrough, you wire the RESTful web service interface up to your back-end logic. You will:

- Pass an event from one flow to another.
- Call the backend flows.
- Create new logic for the nested resource call.
- Test the web service using Advanced REST Client.

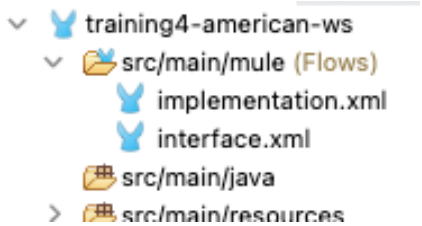


Starting file

If you did not complete the previous walkthrough, you can get a starting file here. This file is also located in the solutions folder of the student files ZIP located in the Course Resources.

Rename the configuration files

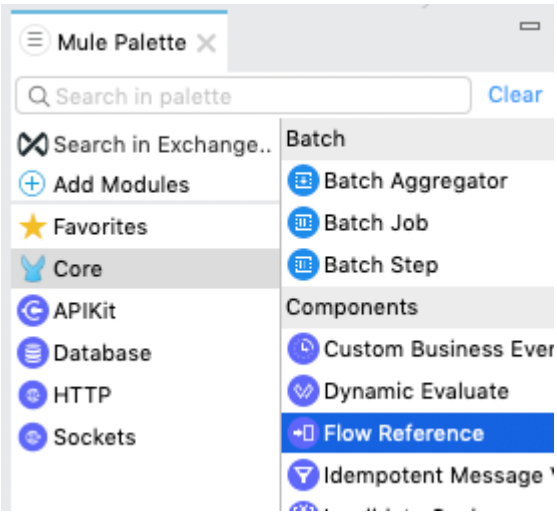
1. Return to Anypoint Studio.
2. Right-click american-flights-api.xml in the Package Explorer and select Refactor > Rename.
3. In the Rename Resource dialog box, set the new name to interface.xml and click OK.
4. Right-click training4-american-ws.xml and select Refactor > Rename.
5. In the Rename Resource dialog box, set the new name to implementation.xml and click OK.



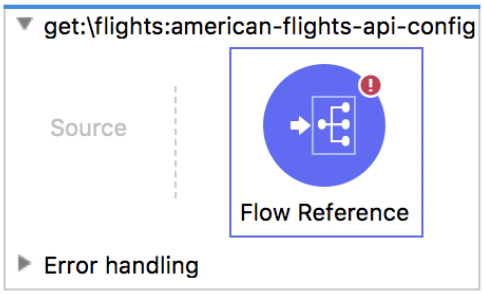
Use a Flow Reference in the /flights resource

6. Open interface.xml.
7. Delete the Transform Message component in the get:\flights flow.

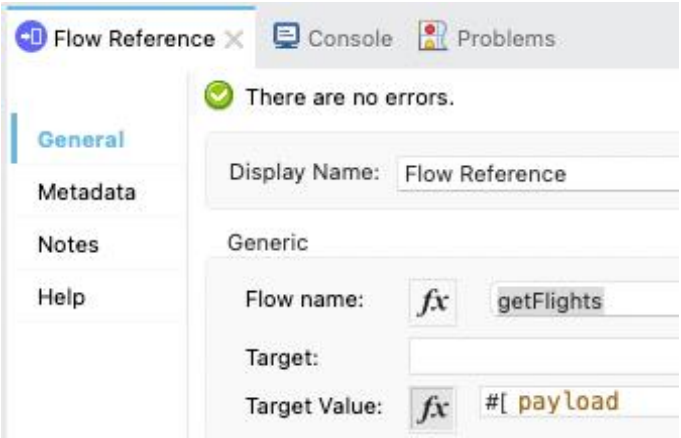
8. In the Mule Palette, select Core and locate the Components section in the right-side.



9. Drag a Flow Reference component from the Mule Palette and drop it into the process section of the flow.



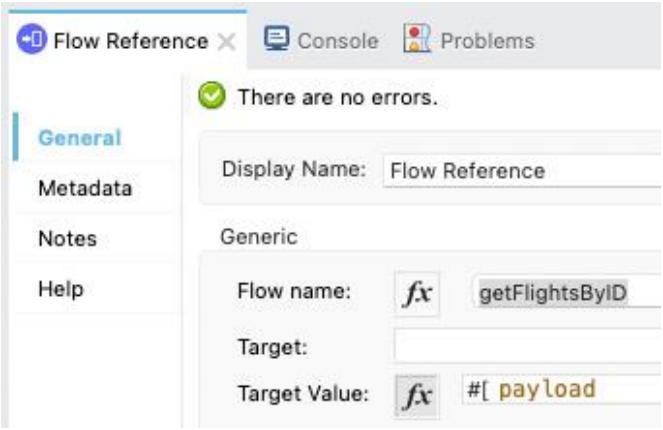
10. In the Flow Reference properties view, select getFlights for the flow name.



11. Change the display name to getFlights.

Use a Flow Reference in the /flights/{ID} resource

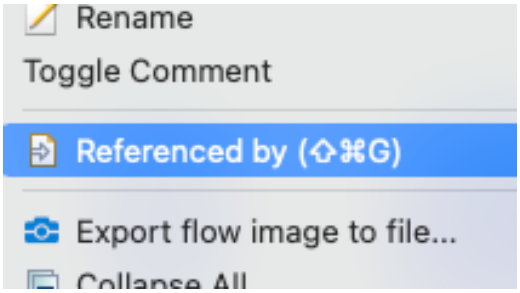
12. Delete both the Transform Message components in the get:\flights\{ID} flow.
13. Drag a Flow Reference component from the Mule Palette and drop it into the flow.
14. In the Flow Reference properties view, select getFlightsByID for the flow name.



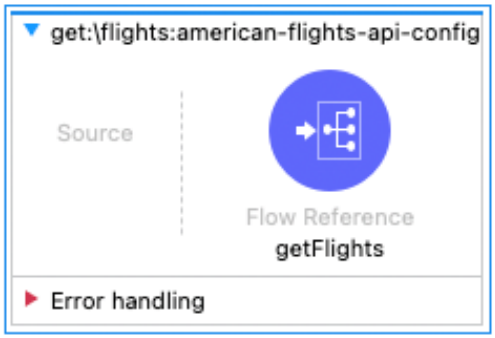
15. Change the display name to getFlightsByID.

Examine the referenced flows

16. Open implementation.xml.
17. Right-click the getFlights flow and select Referenced by.



18. In the resultant dialog, double-click get:\flights to navigate to the referencing flow.



19. Return to implementation.xml.
20. Repeat for the getFlightsByID flow.

Test the web service using Advanced REST Client

21. Save interface.xml to redeploy the project.
22. In Advanced REST Client, make a request to http://localhost:8081/api/flights; you should now get the data for all the flights from the database instead of the sample data.

200 OK


```
[
  {
    "ID": 1,
    "code": "rree0001",
    "price": 541,
    "departureDate": "2016-01-19T19:00:00",
    "origin": "MUA",
    "destination": "LAX",
    "emptySeats": 0,
    "plane": {
      "type": "Boeing 787",
      "totalSeats": 200
    }
  },
  {
    "ID": 2,
    "code": "eefd0123",
    "price": 300,
    "departureDate": "2016-01-24T19:00:00",
    "origin": "MUA",
    "destination": "CLE",
    "emptySeats": 7,
    "plane": {
      "type": "Boeing 747",
      "totalSeats": 345
    }
  },
  {
    "ID": 3,
    "code": "ffee0192"
```


Note: If you use the pre-built Training: American Flights API connector, you must add client_id and client_secret headers with any values.

23. Make a request to http://localhost:8081/api/flights/3; you should now get the data for that flight from the database instead of the sample data.

200 OK

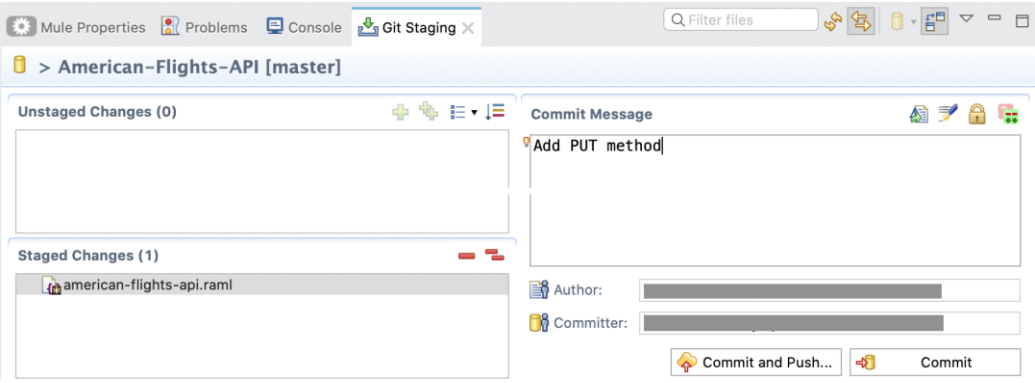

```
[
  {
    "ID": 3,
    "code": "ffee0192",
    "price": 300,
    "departureDate": "2016-01-19T19:00:00",
    "origin": "MUA",
    "destination": "LAX",
    "emptySeats": 0,
    "plane": {
      "type": "Boeing 777",
      "totalSeats": 300
    }
  }
]
```


24. Return to Anypoint Studio.
25. Stop the project.

Walkthrough 4-7: Synchronize changes to an API specification between Studio and Anypoint Platform

In this walkthrough, you synchronize changes to an API specification between Anypoint Studio, Design Center, and Anypoint Exchange. You will:

- Create an editable version of an API specification in Anypoint Studio.
- Make changes to an API specification in Anypoint Studio.
- Push the changes from Anypoint Studio to Design Center.
- Publish the modified API specification from Anypoint Studio to Exchange.
- Update the version of an API specification used in a Mule project.
- Rescaffold an API interface from an updated API specification.



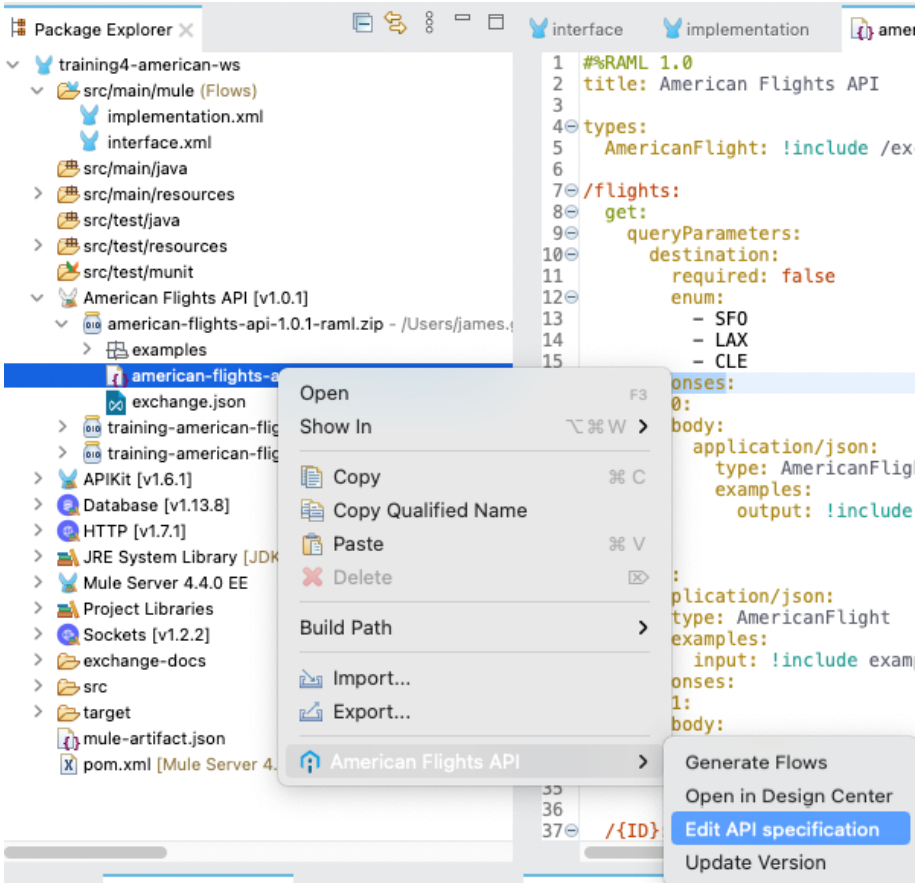
Starting file

If you did not complete the previous walkthrough, you can get an Anypoint Studio starting file here. If you have not developed an *American Flights API* to add to your project, you can also get an API Designer starting file here. These files are also located in the solutions folder of the student files ZIP located in the Course Resources. Use these two starting files while following the steps in the addendum following this walkthrough.

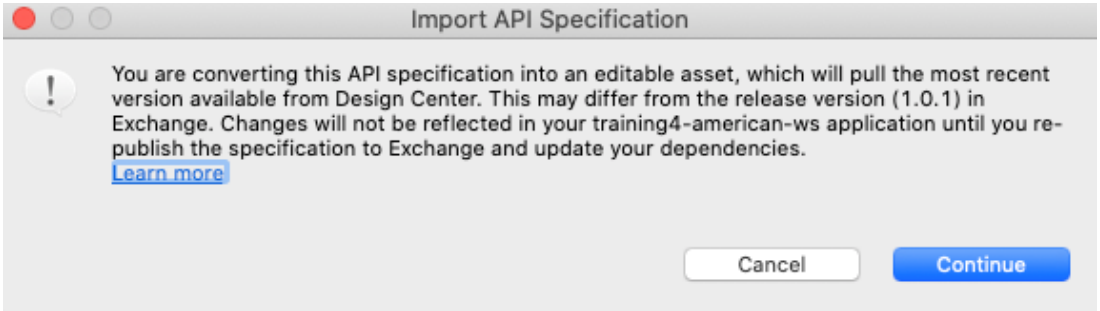
Make the imported API specification editable in Anypoint Studio in a new API project

1. Return to the training4-american-ws project in Anypoint Studio.
2. Open american-flights-api.raml and try to make changes to the code; you should NOT be able to make any changes.

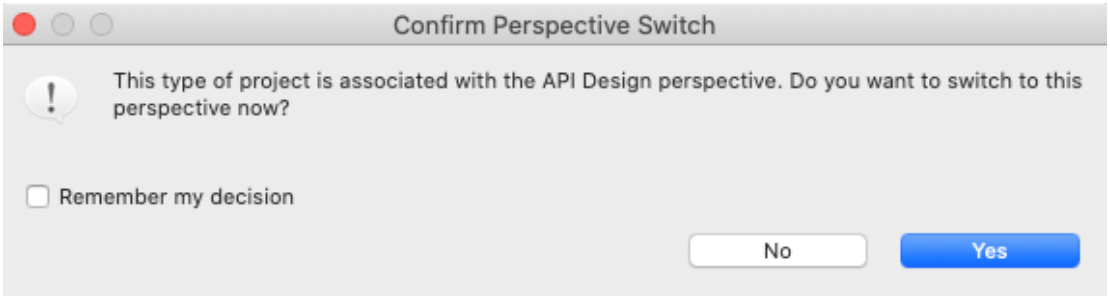
3. Right-click american-flights-api.raml in the Package Explorer and select American Flights API > Edit API specification.



4. In the Import API Specification dialog box, click Continue.

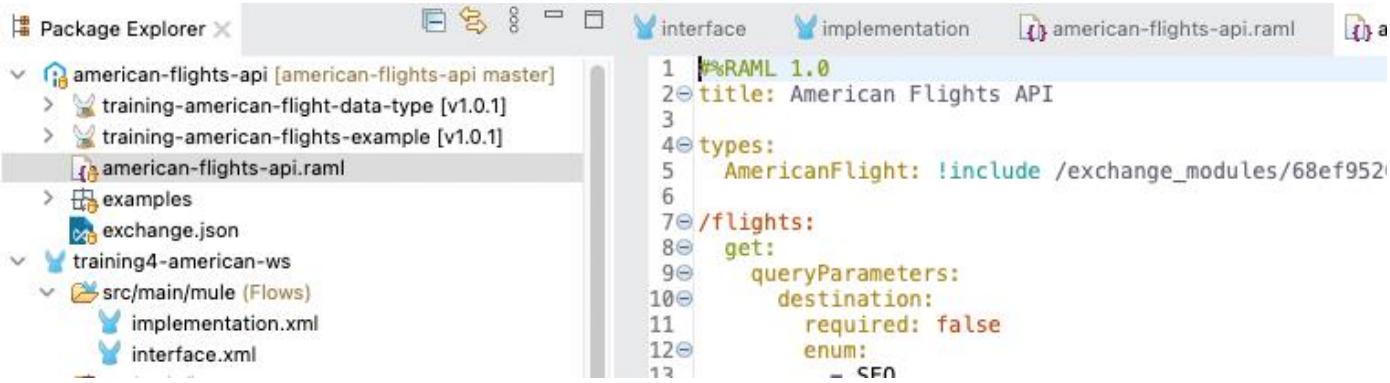


5. In the Confirm Perspective Switch dialog box, click Yes.



Locate the new API project created in Anypoint Studio

6. Locate the new american-flights-api project that was created in the Package Explorer; its american-flights-api.raml file should have automatically opened.



Make changes to the API specification in Anypoint Studio

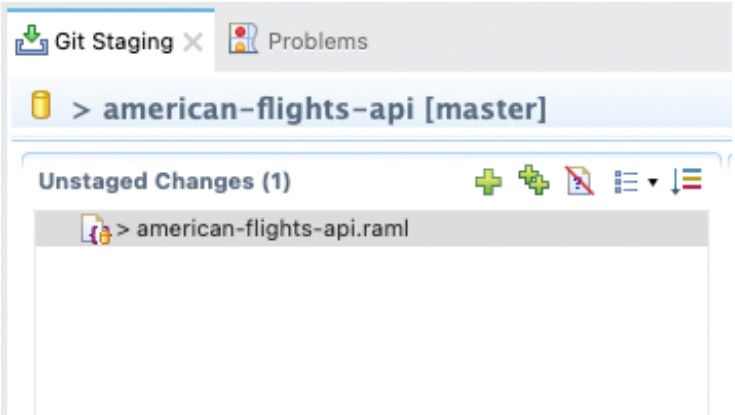
7. Return to the course snippets.txt file and copy the American Flights API - /{ID} PUT method.
8. Return to american-flights-api.raml in the new american-flights-api project in Anypoint Studio.
9. Paste the put method that you copied after the {ID}/delete method.
10. Fix the indentation if necessary.
11. Review the code.

```
  delete:
    responses:
      200:
        body:
          application/json:
            example:
              message: Flight deleted (but not really)

  put:
    body:
      application/json:
        type: AmericanFlight
        examples:
          input: !include examples/AmericanFlightNoIDExample.raml
    responses:
      200:
        body:
          application/json:
            example:
              message: Flight updated (but not really)
```

12. Save the file.

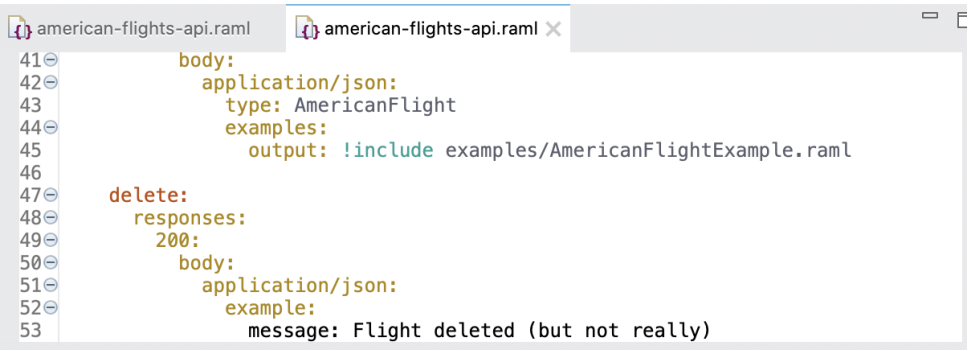
Review the Git Staging view for the API project

13. Locate the new Git Staging view that opened in Anypoint Studio.
14. Examine the Unstaged Changes section; you should see the american-flights-api.raml file that is not yet staged for synchronization.



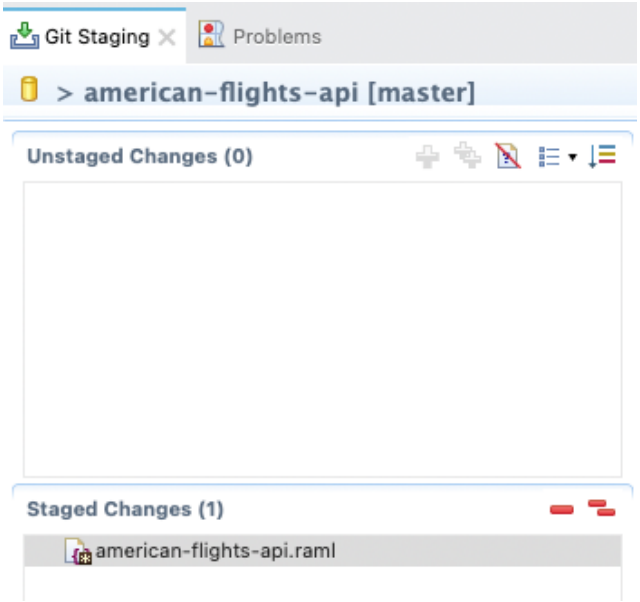
Review the API specification that was imported into the Mule project

15. Return to the training4-american-ws project.
16. Review the american-flights-api.raml file in that project; you should NOT see the new put method.

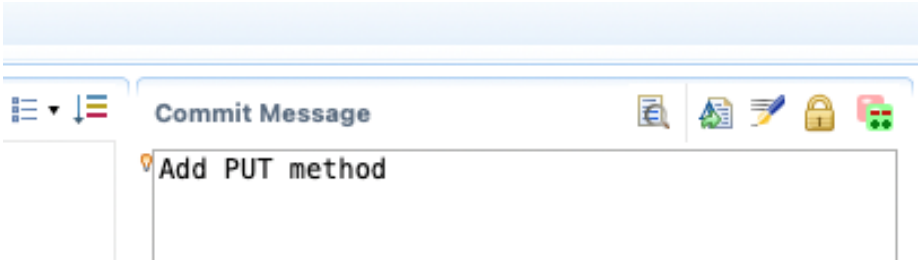


Push the changes for the modified API from Anypoint Studio to Design Center

17. Return to the Git Staging view.
18. Click the Add selected files to the index button in the upper-right corner of the Unstaged Changes section; the american-flights-api.raml file should move from the Unstaged Changes section to the Staged Changes section.

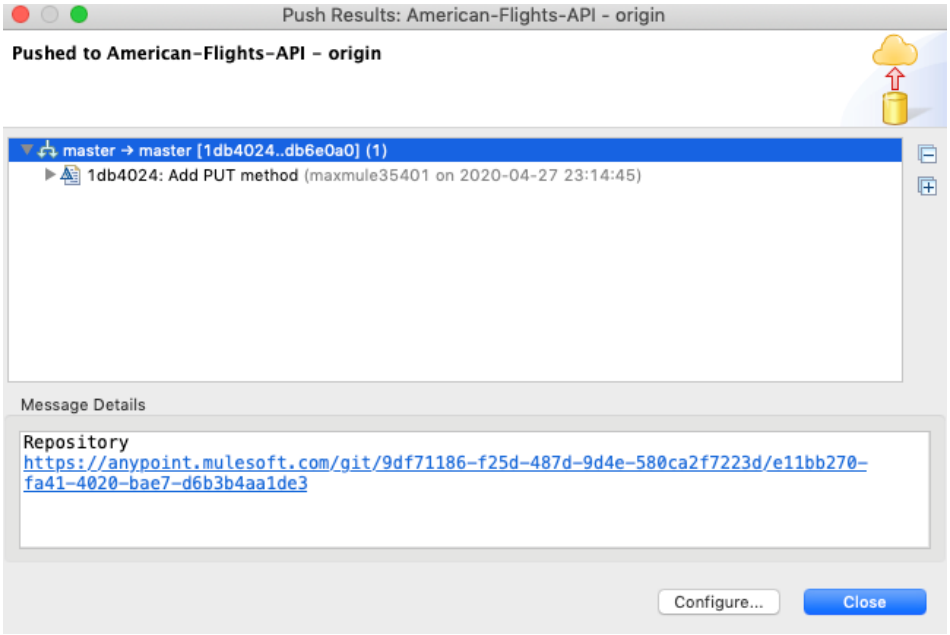


19. In the Commit Message section, enter the text Add PUT method.



20. Click the Commit and Push button; the Staged Changes section should now be empty.

21. In the Push Results dialog box, examine the commit log messages.



22. Click Close.

Examine the synchronized changes in API Designer

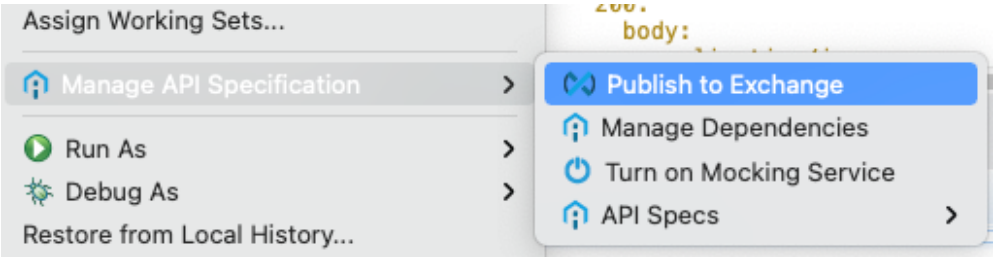
23. Return to your American Flights API in API Designer.
24. Notice the addition of the PUT method after the {ID}/delete method.

```
47      delete:
48        responses:
49          200:
50            body:
51              application/json:
52                example:
53                  message: Flight deleted (but not really)
54
55      put:
56        body:
57          application/json:
58            type: AmericanFlight
59            examples:
60              input: !include examples/AmericanFlightNoIDExample.raml
61        responses:
62          200:
63            body:
64              application/json:
65                example:
66                  message: Flight updated (but not really)
```

Note: If you do not see the PUT method, refresh the page.

Publish the modified API to Exchange from Anypoint Studio

25. Return to Anypoint Studio.
26. In the Package Explorer, right-click the american-flights-api project and select Manage API Specification > Publish to Exchange.



27. In the API configuration dialog box, notice that Last published version is 1.0.1.
28. Set the Asset version to 1.0.2.

29. Click Finish.
30. Wait until the API publishes to Exchange and then in the Publish your API to Exchange dialog box that appears, click OK.
31. In the Package Explorer, right-click the american-flights-api project and select Close Project.
32. Close the Git Staging view.

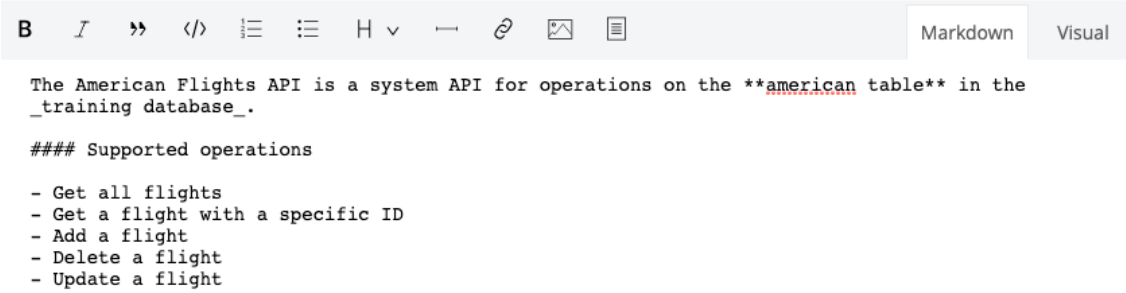
Examine the new published API asset version in Exchange

33. Return to your American Flights API in Exchange.
34. In the left-side navigation, click Home to return to the API portal for American Flights API.
35. On the right side of the page, locate the latest asset versions listed for the API; you should see the new 1.0.2 asset version.

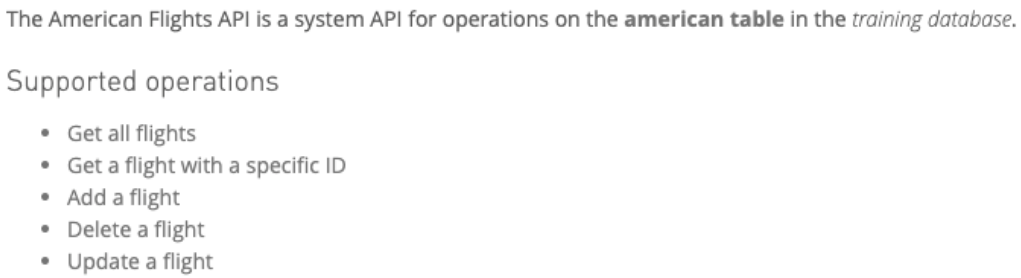
Note: If you do not see the 1.0.2 asset version, refresh the page.

Update the API portal information

36. Click the Edit documentation button.
37. Add the new update a flight operation.

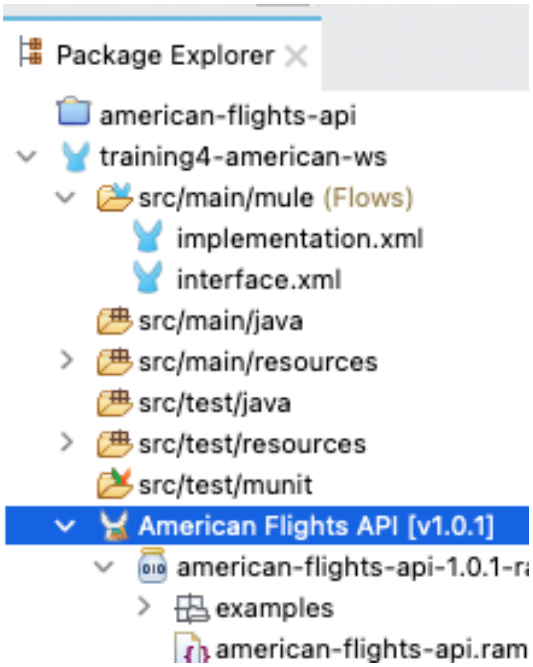


38. Click Save and then Publish.



Review the API specification that was imported into the Mule project again

39. Return to the training4-american-ws project in Anypoint Studio.
40. Locate the American Flights API in the training4-american-ws project and notice that it is still v1.0.1.

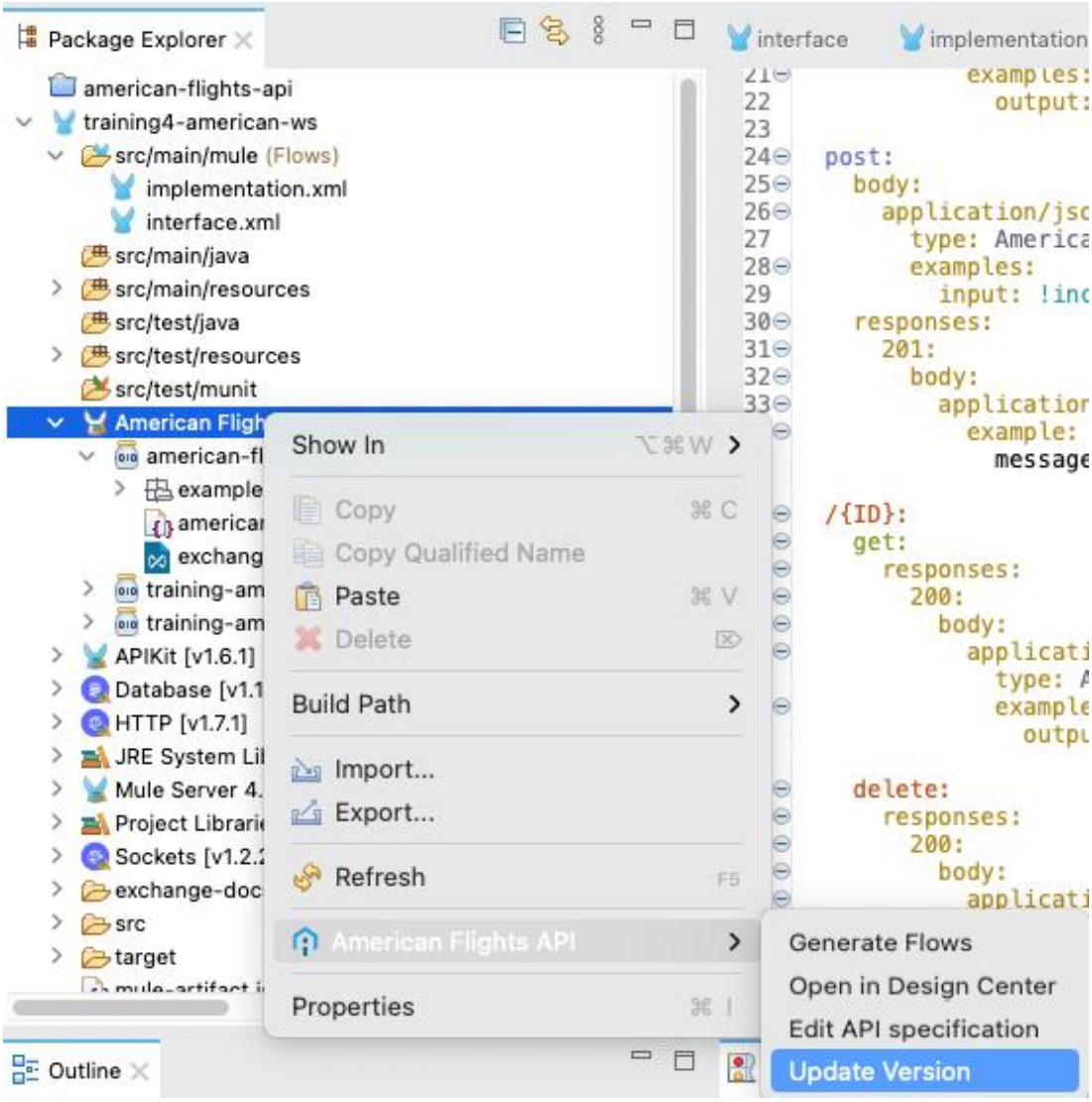


41. Review the american-flights-api.raml file; you should still NOT see the new put method.

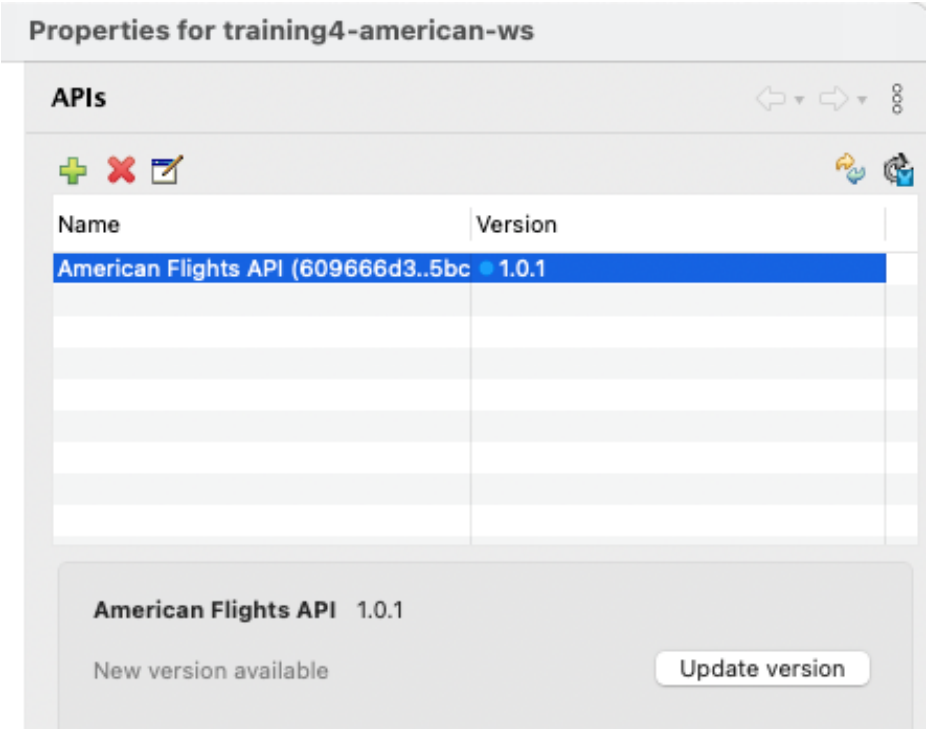
```
american-flights-api.raml   american-flights-api.raml
41          body:
42            application/json:
43              type: AmericanFlight
44              examples:
45                output: !include examples/AmericanFlightExample.raml
46
47    delete:
48      responses:
49        200:
50          body:
51            application/json:
52              example:
53                message: Flight deleted (but not really)
```

Update the version of the API used in the Mule project and rescaffold the flows

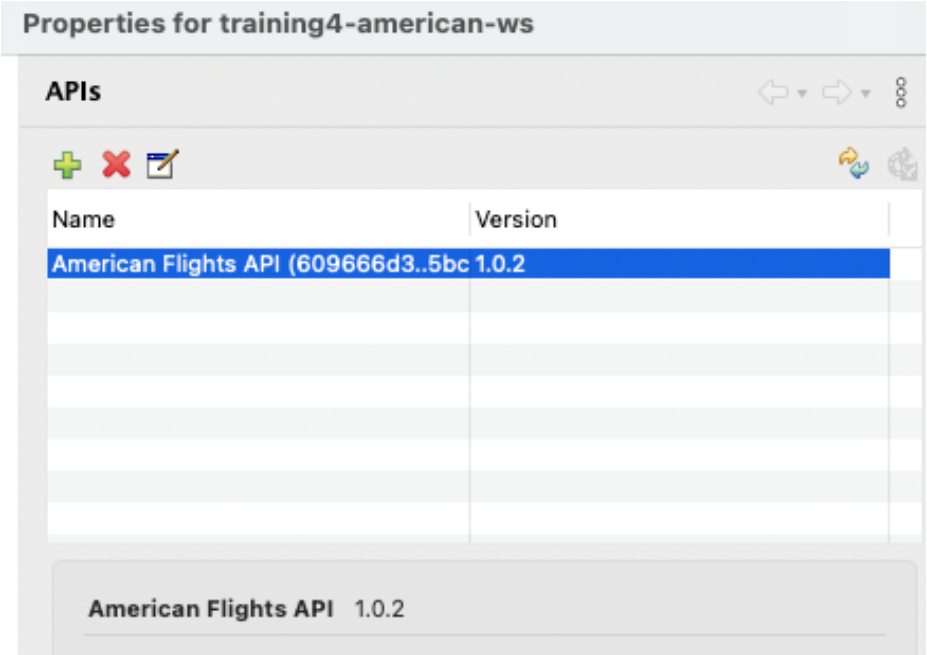
42. Right-click American Flights API in the training4-american-ws project and select American Flights API > Update Version.



43. In the Properties for training4-american-ws dialog box, select the American Flights API; an Update version button should appear.

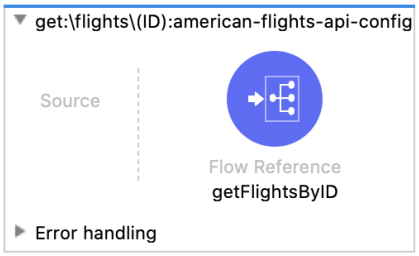


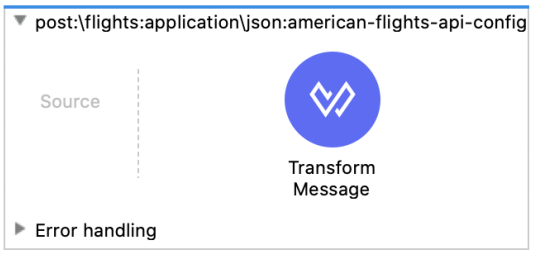
44. Click Update version; the version of the API should change to 1.0.2.

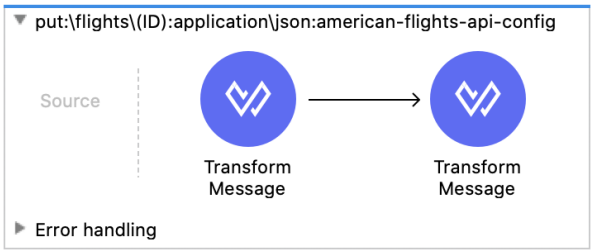


45. Click Apply and Close.
46. In the dialog box that appears, click Yes to scaffold American Flights API specification, and click Yes if you get a Confirm Perspective Switch dialog box.

47. When the scaffolding is complete, review the flows in interface.xml; you should now see five method flows, including the new put:\flights\{ID} method.
48. Look at the get:\flights\{ID} method; it was not overwritten and still has your change to use a Flow Reference component.







Starting File Addendum (only needed if you did not complete Walkthrough 4-6)

Create your editable API in Exchange with the correct asset number

1. In API Designer, create American Flights API then import wt3-4_American-Flights-API_solution.zip electing to replace american-flights-api.raml.
2. Publish the API to Exchange with an asset/version of 1.0.1/v1 and a stable lifecycle state.

Create your Studio application and replace the static API with your editable API

3. In Studio, import wt4-6_training4-american-ws_solution.jar.
4. Right-click the project and select Manage Dependencies > Manage APIs to delete Training: American Flights API and to add your American Flights API from Exchange.

Note: Be sure to scaffold your American Flights API when prompted.

Transfer the flow references from the old scaffold to the new

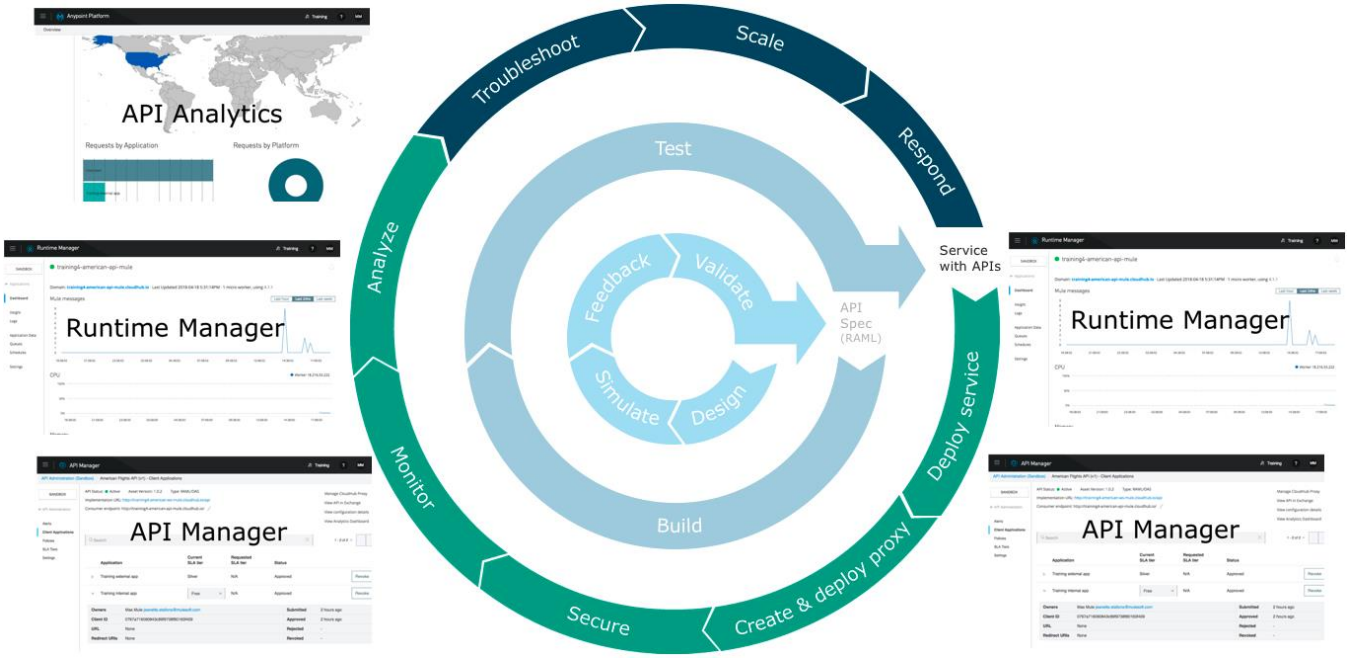
5. Copy/paste the flow reference from interface.xml get:\flights to its corresponding location in american-flights-api-2.xml replacing any transforms.
6. Repeat for the get:\flights\(ID) flow reference.

Remove the old scaffolding and configure the new scaffolding

7. Delete interface.xml and refactor/rename american-flights-api-2.xml to interface.xml.
8. Set the main Listener in interface.xml to use the configuration HTTP_Listener_config.
9. Save your changes; you should now be ready to perform this walkthrough.

Note: When using these starting files, walkthrough steps such as updating the API portal information in Exchange may not match the screenshots.

Module 5: Deploying and managing APIs



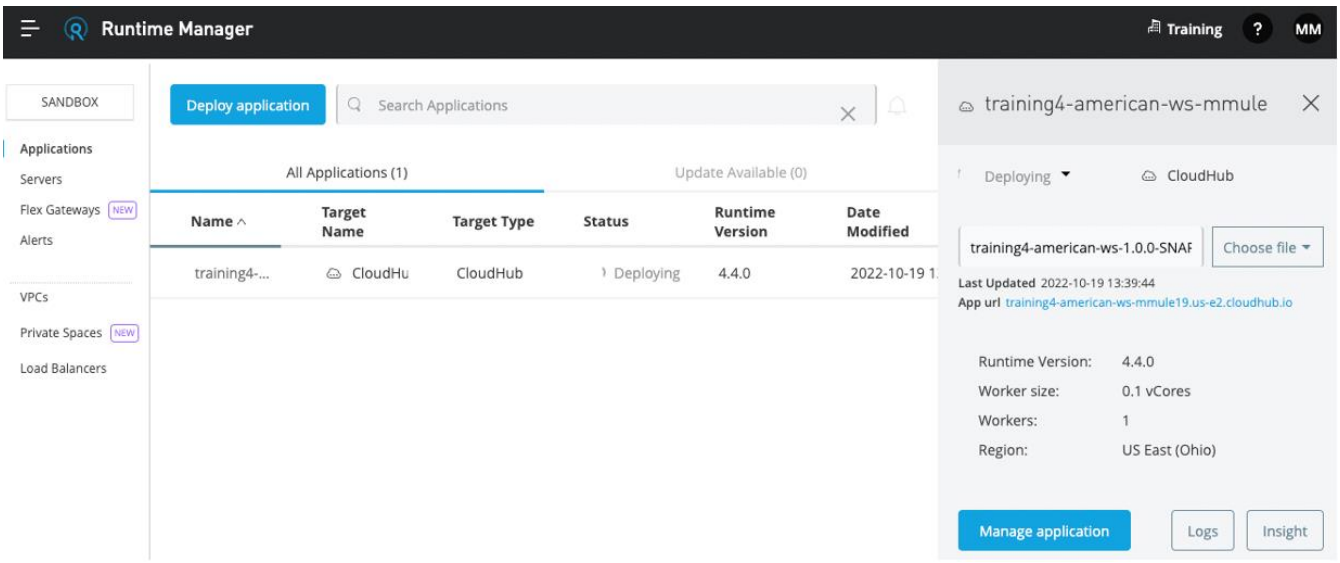
At the end of this module, you should be able to:

- Describe the options for deploying Mule applications.
- Deploy Mule applications to CloudHub.
- Use API Manager to create and deploy API proxies.
- Use API Manager to restrict access to API proxies.

Walkthrough 5-1: Deploy an application to CloudHub

In this walkthrough, you deploy and run your application on CloudHub. You will:

- Deploy an application from Anypoint Studio to CloudHub.
- Run the application on its new, hosted domain.
- Make calls to the web service.
- Update an API implementation deployed to CloudHub.



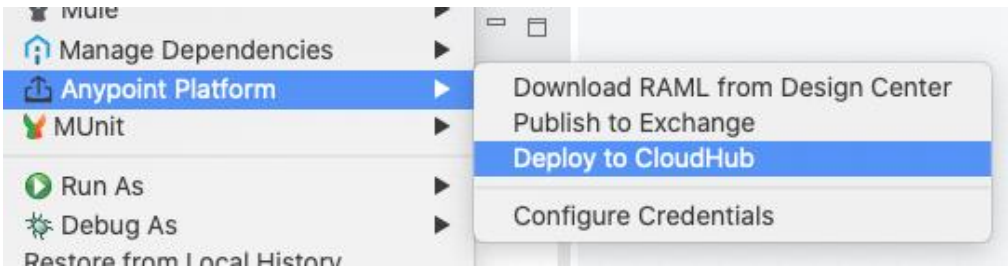
Note: If you do not have a working application at this point, import the wt4-7_training4-american-ws_solution.jar solution into Anypoint Studio and work with that project.

Starting file

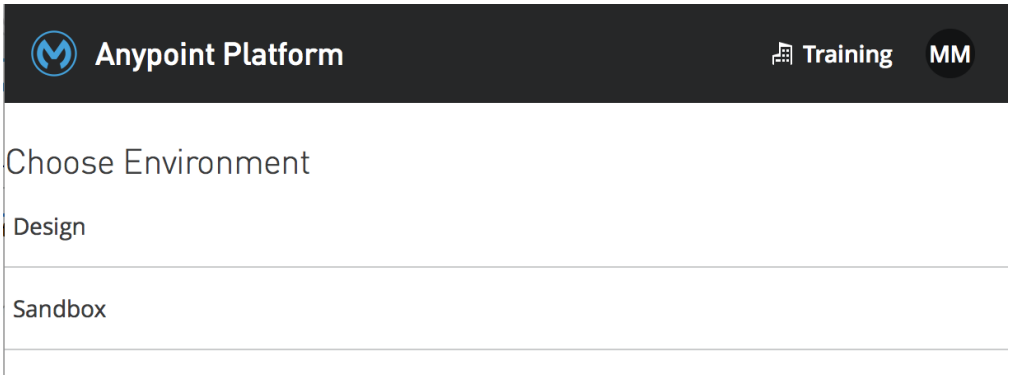
If you did not complete the previous walkthrough, you can get a starting file here. This file is also located in the solutions folder of the student files ZIP located in the Course Resources.

Deploy the application to CloudHub

1. Return to the training4-american-ws project in Anypoint Studio.
2. In the Package Explorer, right-click the project and select Anypoint Platform > Deploy to CloudHub.



3. In the Choose Environment dialog box, select Sandbox.

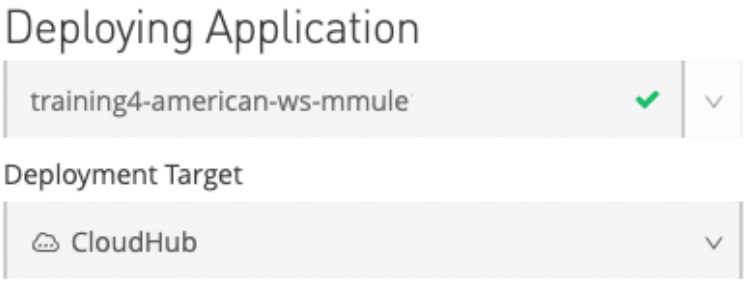


Note: If you get a dialog asking to enable multi-factor authentication, click Not Now.

4. At the top of the Anypoint Platform dialog box, set the application name to training4-american-ws-*{your-lastname}* so it is a unique value.

Note: This name will be part of the URL used to access the application on CloudHub. It must be unique across all applications on CloudHub. The availability of the domain is instantly checked and you will get a green check mark if it is available.

5. Make sure Deployment Target is set to CloudHub.

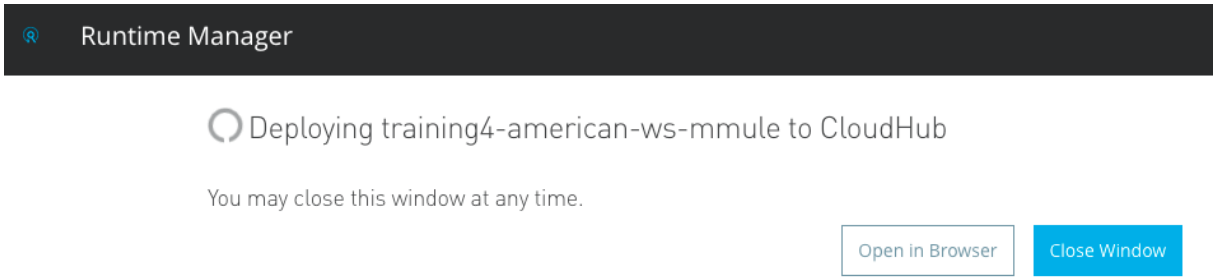


6. Make sure the runtime version is set to the version your project is using and that the worker size is set to 0.1 vCores.

Note: If you don't know what version it is using, look at the Package Explorer and find a library folder with the name of the server being used, like Mule Server 4.4.0 EE.

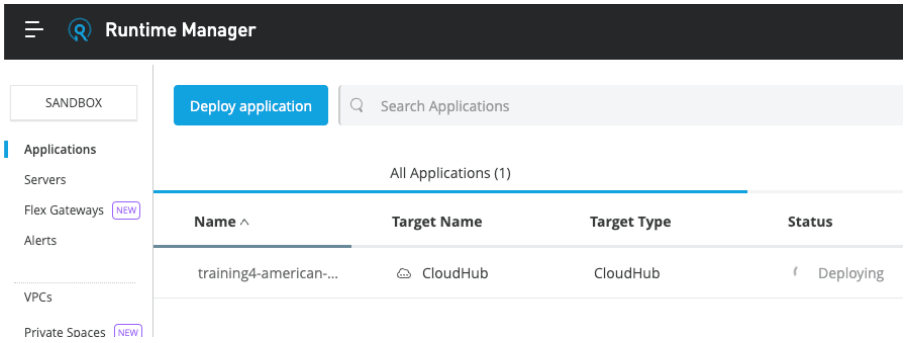
7. Click the Deploy Application button.

8. Click the Open in Browser button (you don't have to wait for your application to fully deploy).



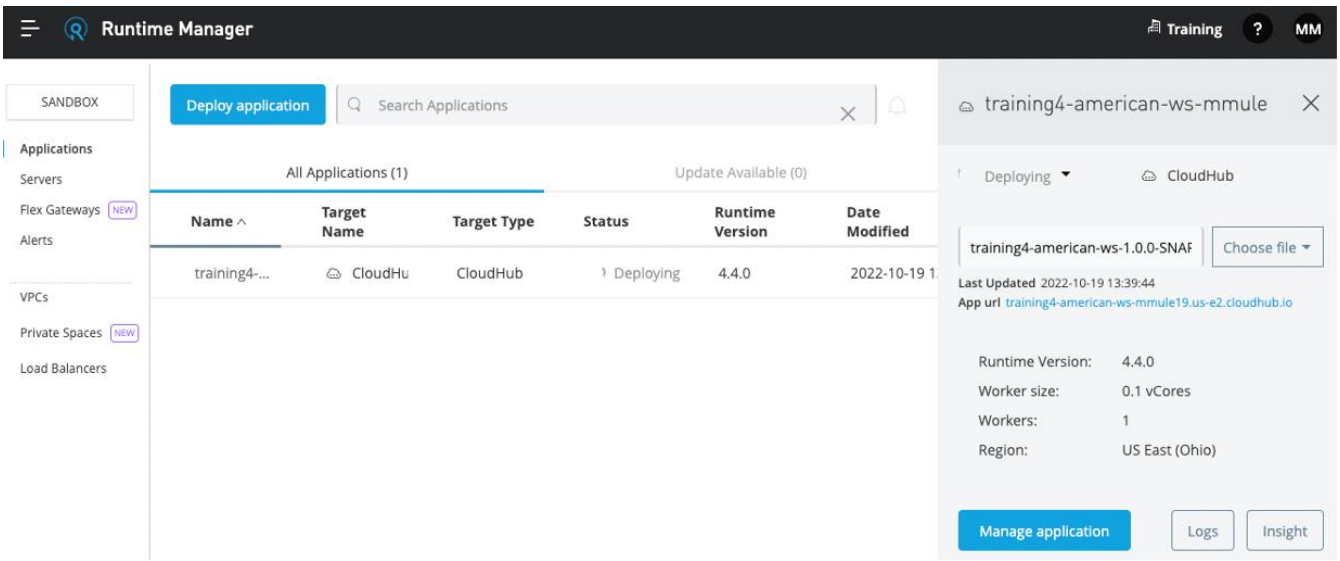
Note: The Runtime Manager dialog box which contains this button may take some time to appear.

9. In the Anypoint Platform browser window that opens, choose Sandbox for an environment then locate the status of your deployment in the Runtime Manager.



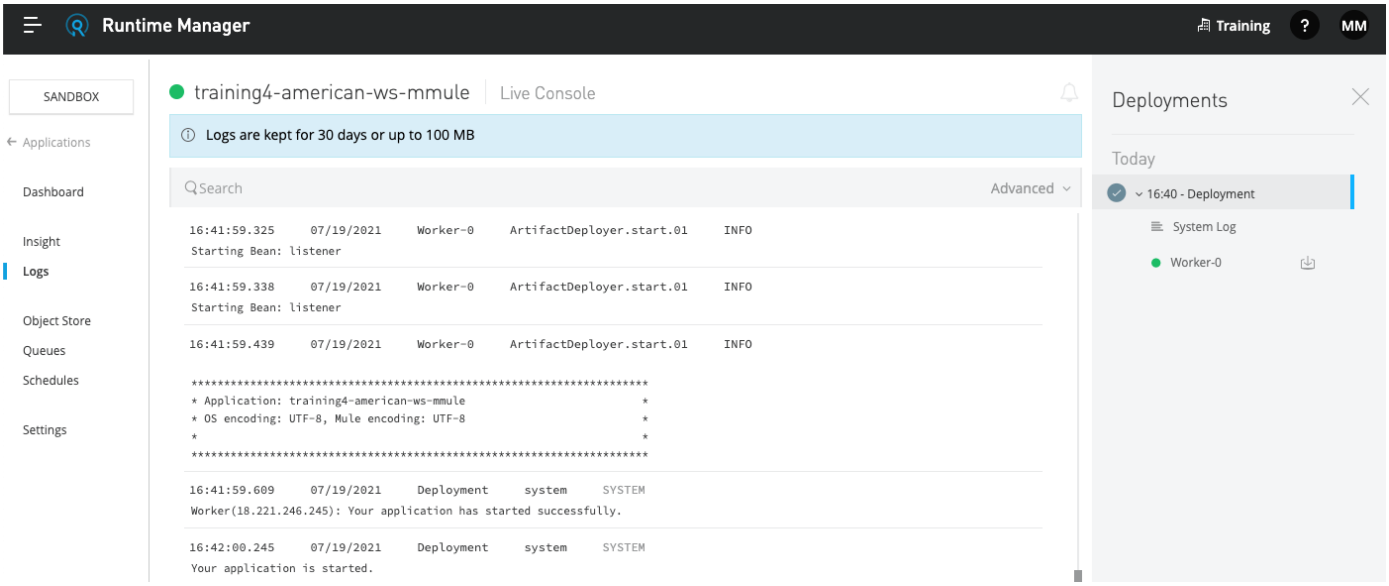
Watch the logs and wait for the application to start

10. Click in the row of the application (not on its name); you should see information about the application appear on the right side of the window.



11. Click the Logs button.
12. Watch the logs as the application is deployed.

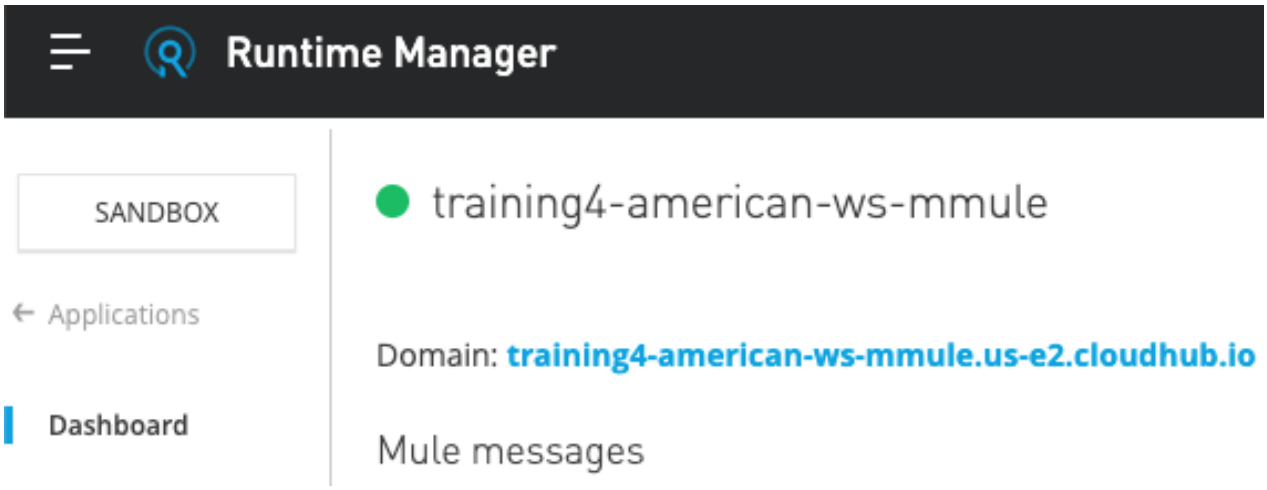
13. Wait until the application starts (or fails to start).



Note: If your application did not successfully deploy, read the logs to help figure out why the application did not deploy. If you had errors when deploying, troubleshoot them, fix them, and then redeploy.

Test the application

14. In the left-side navigation, select Dashboard.
15. Locate the link for the application on its new domain:
    training4-american-ws-*{lastname}*.{region}.cloudhub.io.



Note: {region} represents the worker region to which the Mule application is deployed. In North America, the default region is US East and is denoted by us-e2 in the application URL.

16. Click the link; a GET request will be made to that URL in a new browser tab and you should get a message that there is no listener for that endpoint.

17. Modify the path to http://training4-american-ws-{lastname}.{region}.cloudhub.io/api/flights; you should see the flights data.


```
[
  {
    "ID": 1,
    "code": "rree0001",
    "price": 541,
    "departureDate": "2016-01-20T00:00:00",
    "origin": "MUA",
    "destination": "LAX",
    "emptySeats": 0,
    "plane": {
      "type": "Boeing 787",
      "totalSeats": 200
    }
  },
  {
    "ID": 2,
    "code": "eefd0123",
    "price": 300,
    "departureDate": "2016-01-25T00:00:00",
    "origin": "MUA",
    "destination": "CLE",
```


Note: If you are using the local Derby database, your application will not return results when deployed to CloudHub. You will update the application with a version using the MySQL database in the next section, so it works.

18. Add a query parameter called destination to the URL and set it equal to SFO.
19. Send the request; you should still get all the flights.

Note: You did not add logic to the application to search for a particular destination. You will deploy an application with this additional functionality implemented next.


```
[
  {
    "ID": 1,
    "code": "rree0001",
    "price": 541,
    "departureDate": "2016-01-20T00:00:00",
    "origin": "MUA",
    "destination": "LAX",
    "emptySeats": 0,
    "plane": {
      "type": "Boeing 787",
      "totalSeats": 200
    }
  },
  {
    "ID": 2,
    "code": "eefd0123",
    "price": 300,
    "departureDate": "2016-01-25T00:00:00",
    "origin": "MUA",
    "destination": "CLE",
    "emptySeats": 7
```


20. Leave this browser tab open.

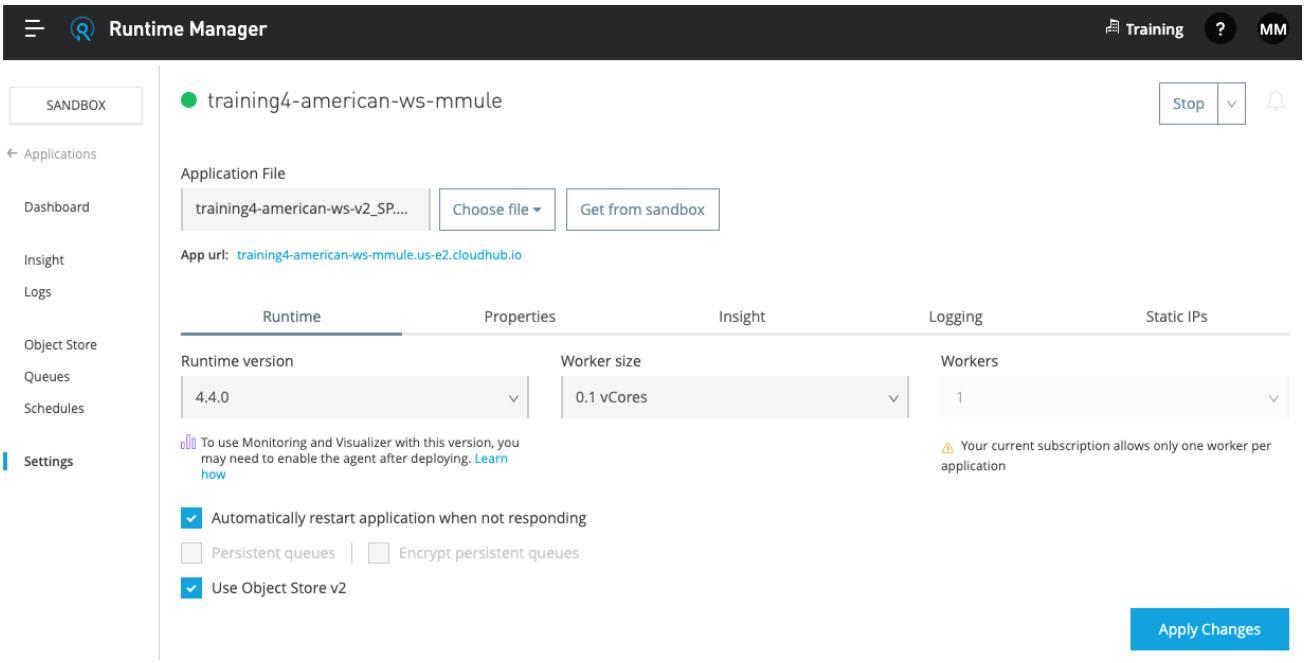
Update the API implementation deployed to CloudHub

21. Return to the browser tab with Runtime Manager.
22. In the left-side navigation, select Settings.
23. Click the Choose file button and select Upload file.
24. Browse to the jars folder in the student files.
25. Select training4-american-ws-v2_SP.jar and click Open.

Note: The filename differs slightly for instructor-led and self-study training classes.

Note: This updated version of the application adds functionality to return results for a particular destination. You will learn to do this later in the Development Fundamentals course.

26. Click the Apply Changes button.



27. Wait until the application is uploaded and then redeploys successfully.

Note: Because this can take some time for trial accounts, your instructor may move on with the next topic and then come back to test this later.

28. Close the browser tab with Runtime Manager.

Test the updated application

29. Return to the browser tab with a request to the API implementation on CloudHub with a destination of SFO and refresh it; you should now get only flights to SFO.

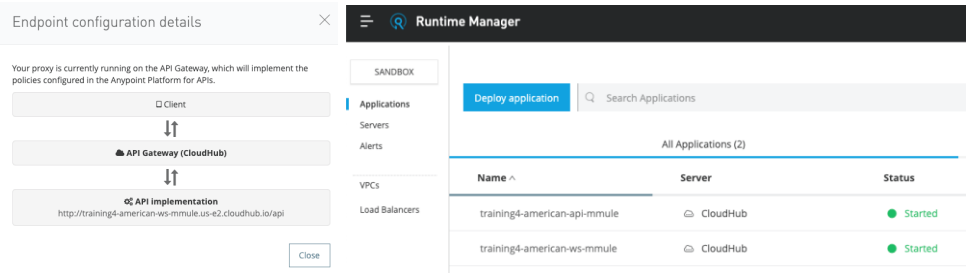

```
[
  {
    "ID": 5,
    "code": "rree1093",
    "price": 142,
    "departureDate": "2016-02-11T00:00:00",
    "origin": "MUA",
    "destination": "SFO",
    "emptySeats": 1,
    "plane": {
      "type": "Boeing 737",
      "totalSeats": 150
    }
  },
  {
    "ID": 7,
    "code": "eefd1994",
    "price": 676,
    "departureDate": "2016-01-01T00:00:00",
    "origin": "MUA",
    "destination": "SFO",
    "emptySeats": 0,
    "plane": {
      "type": "Boeing 777",
      "totalSeats": 300
    }
  },
  {
    "ID": 8,
    "code": "ffee2000",
    "price": 300,
    "departureDate": "2016-02-20T00:00:00",
    "origin": "MUA",
    "destination": "SFO",
    "emptySeats": 30,
```


30. Close this browser tab.

Walkthrough 5-2: Create and deploy an API proxy

In this walkthrough, you create and deploy an API proxy for your API implementation on CloudHub. You will:

- Add an API to API Manager.
- Use API Manager to create and deploy an API proxy application.
- Set a proxy consumer endpoint so requests can be made to it from Exchange.
- Make calls to an API proxy from API portals for both internal and external developers.
- View API request data in API Manager.

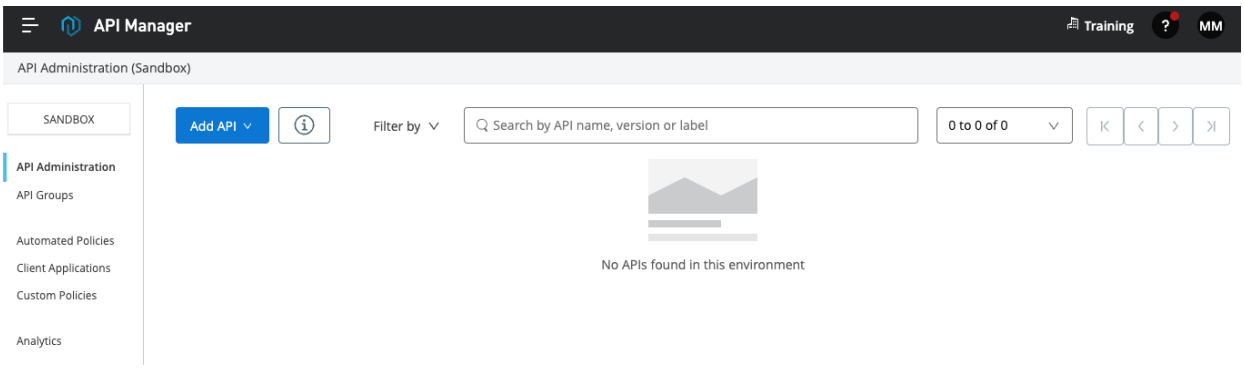


Starting file

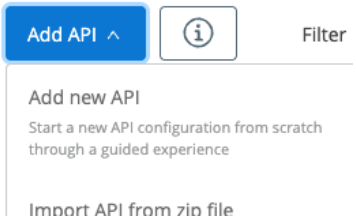
This walkthrough uses Anypoint Platform. There is no starting file. To complete the walkthrough, you must have completed the preceding walkthrough.

Create and deploy a proxy application

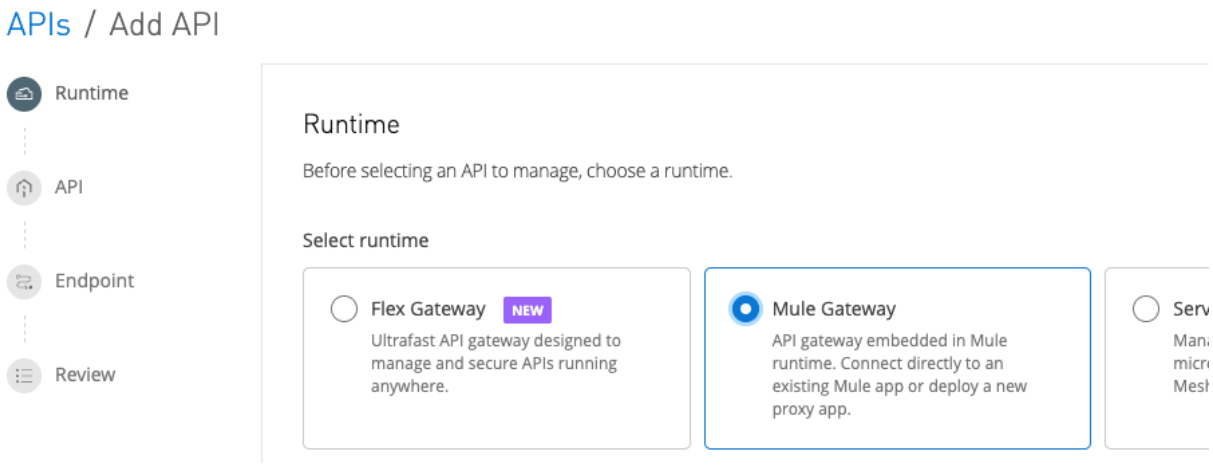
1. Return to Anypoint Platform.
2. In the main menu, select API Manager; you should see no APIs listed for the Sandbox environment.



3. Click the Add API button and select Add new API.

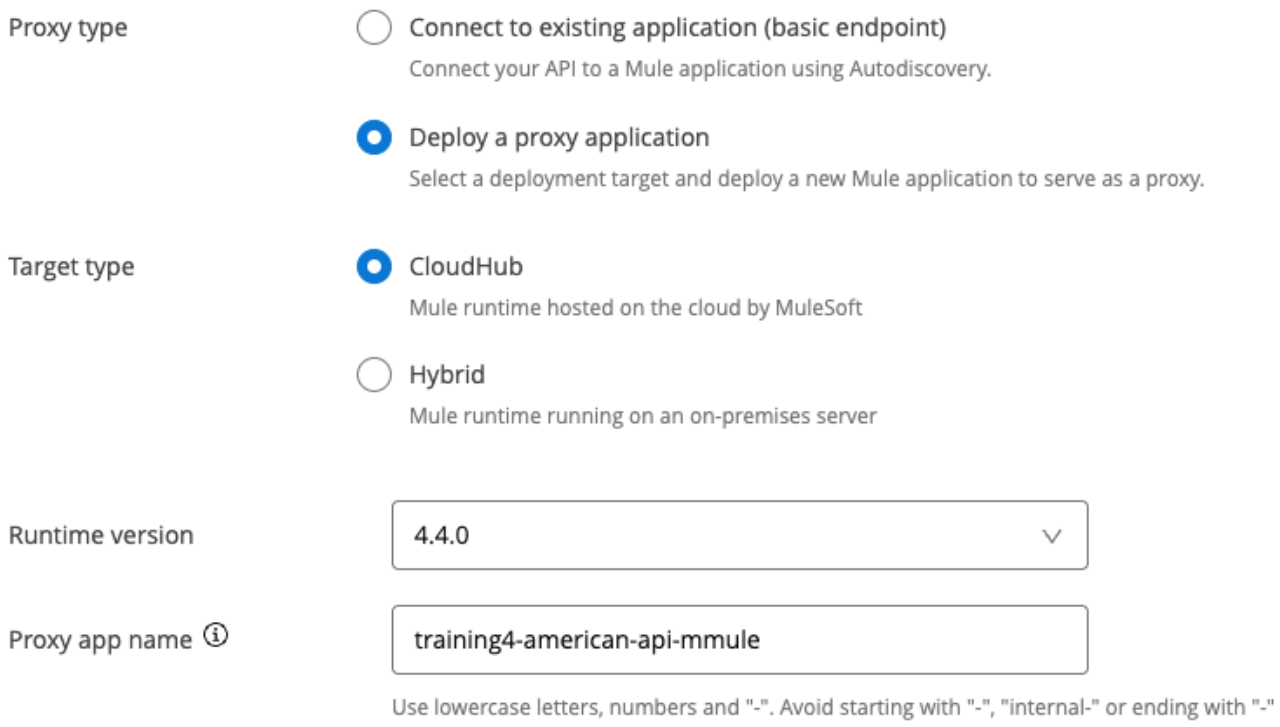


4. On the APIs / Add API page, select the Mule Gateway button for the Runtime settings.



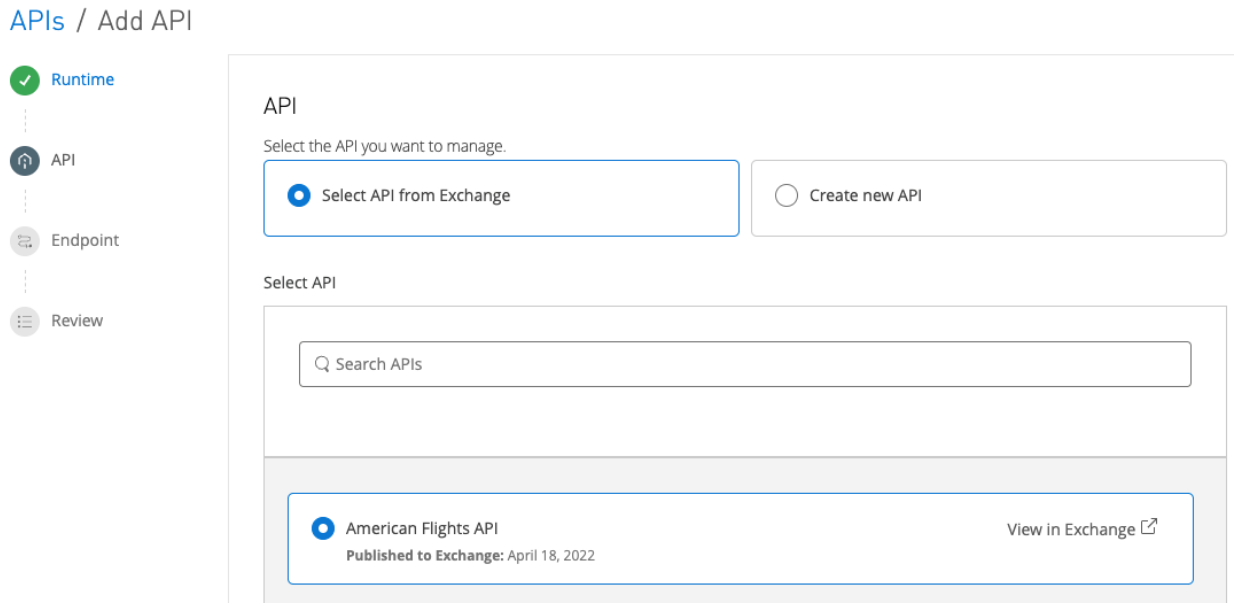
5. Select Deploy a proxy application for the proxy type.
6. Set the rest of the Runtime settings to the following values:

- Target type: CloudHub
- Runtime version: 4.4.0
- Proxy app name: training4-american-api-{*lastname*}



7. Click Next.

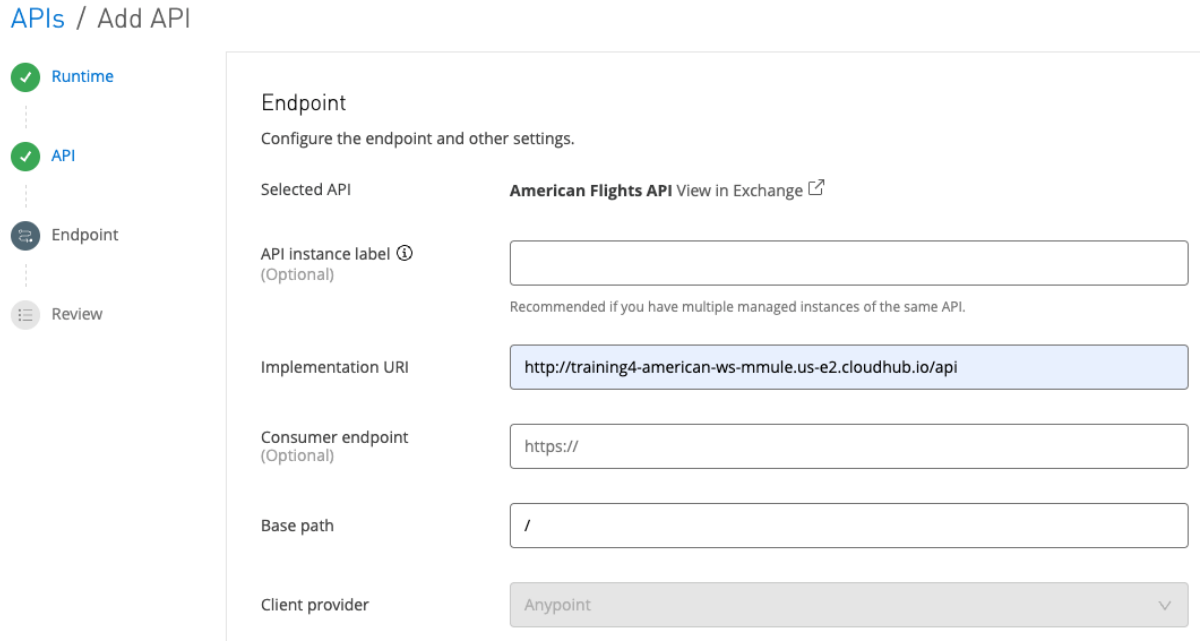
8. For the API settings, ensure Select API from Exchange is selected then select your American Flights API in the Select API section.



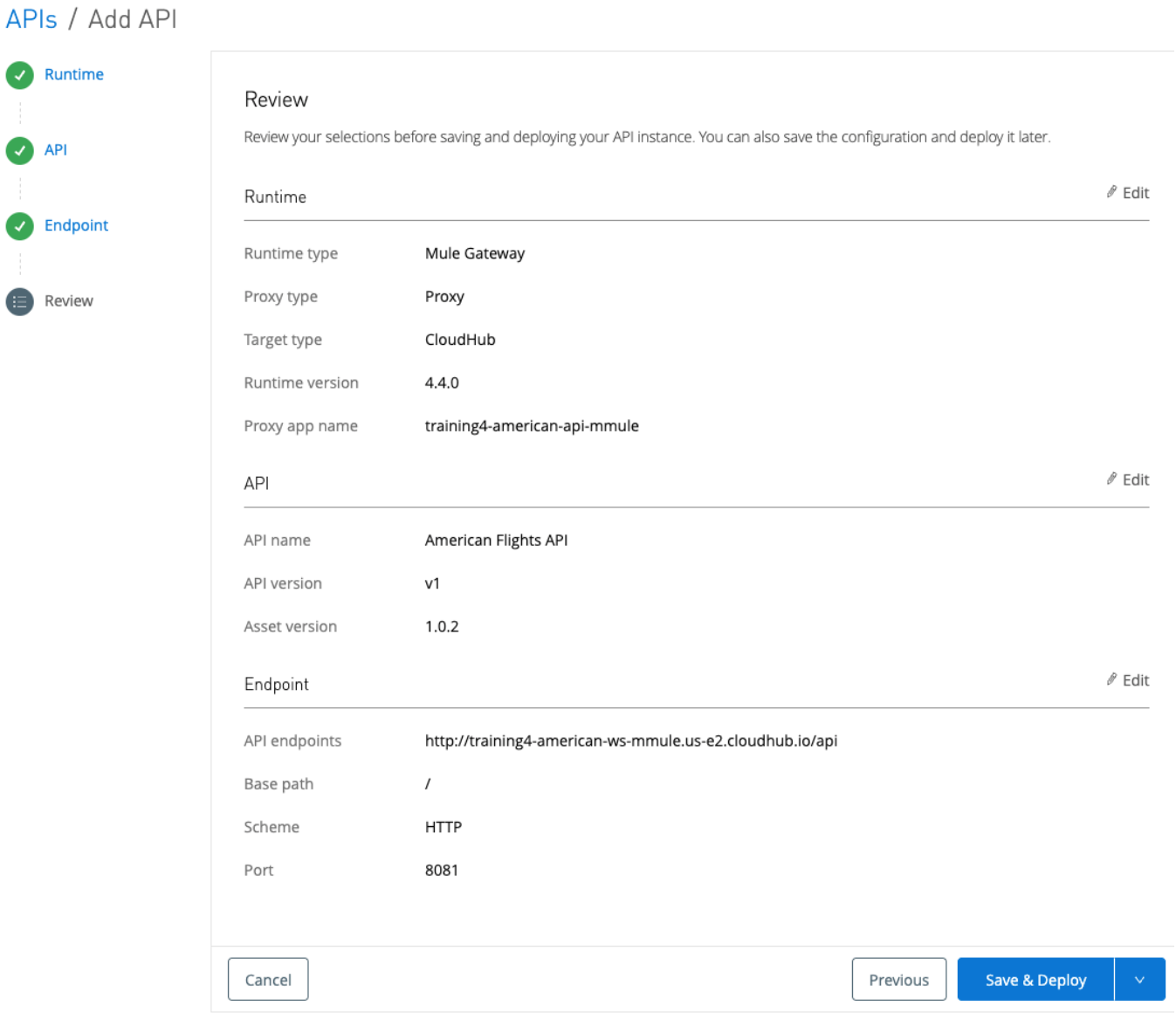
9. Ensure the rest of the API settings fields are set to the following values:

- Asset type: RAML/OAS
- API version: v1 (Latest)
- Asset version: 1.0.2 (Latest)

10. Click Next.
11. Set the Endpoint Implementation URI setting to http://training4-american-ws-{lastname}.{region}.cloudhub.io/api.



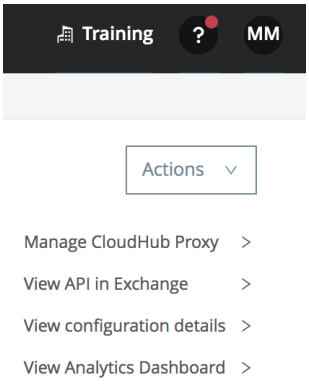
12. Expand Advanced options and examine the additional fields and values.
13. Click Next.
14. Review the Add API settings and click Save & Deploy.



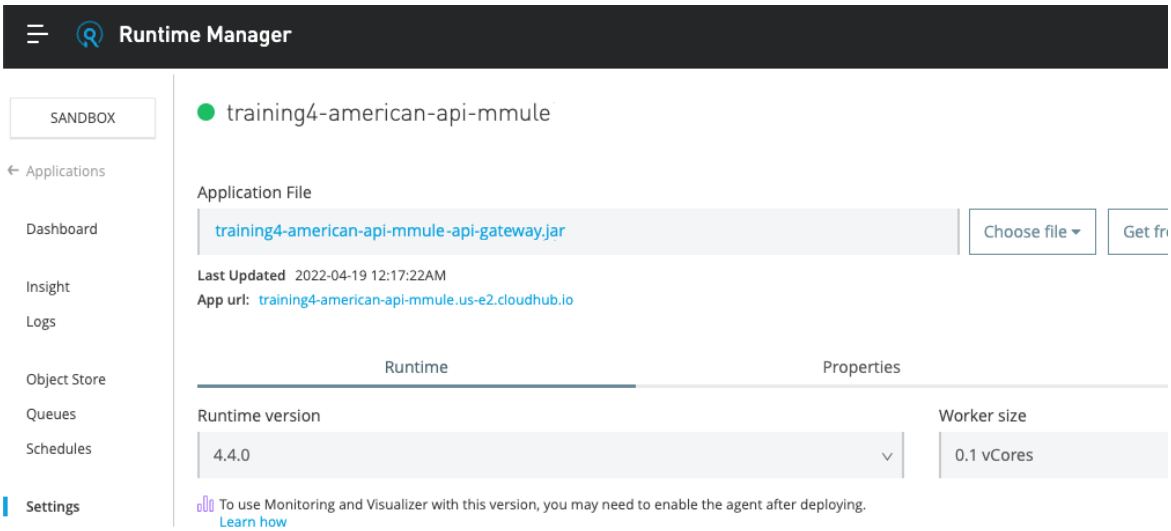
15. Wait until your API is deployed to CloudHub.

Note: If it does not successfully deploy, examine the logs for your proxy application in Runtime Manager to help figure out why the application did not deploy. If you had errors when deploying, troubleshoot them, fix them, and then redeploy.

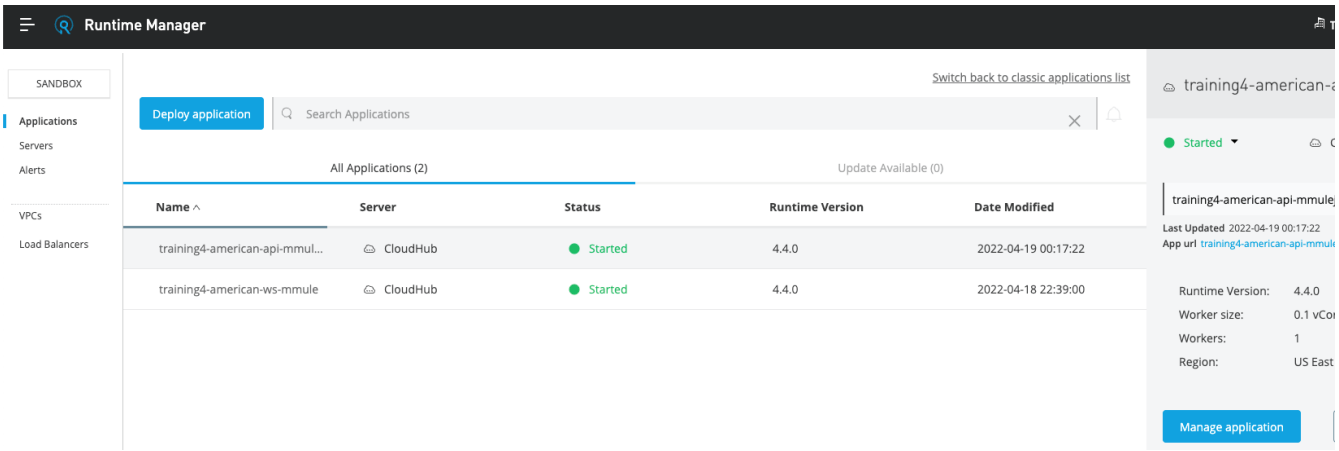
16. Locate and review the links in the upper-right corner.



17. Select Manage CloudHub Proxy: a new browser tab should open with your new proxy application in Runtime Manager.



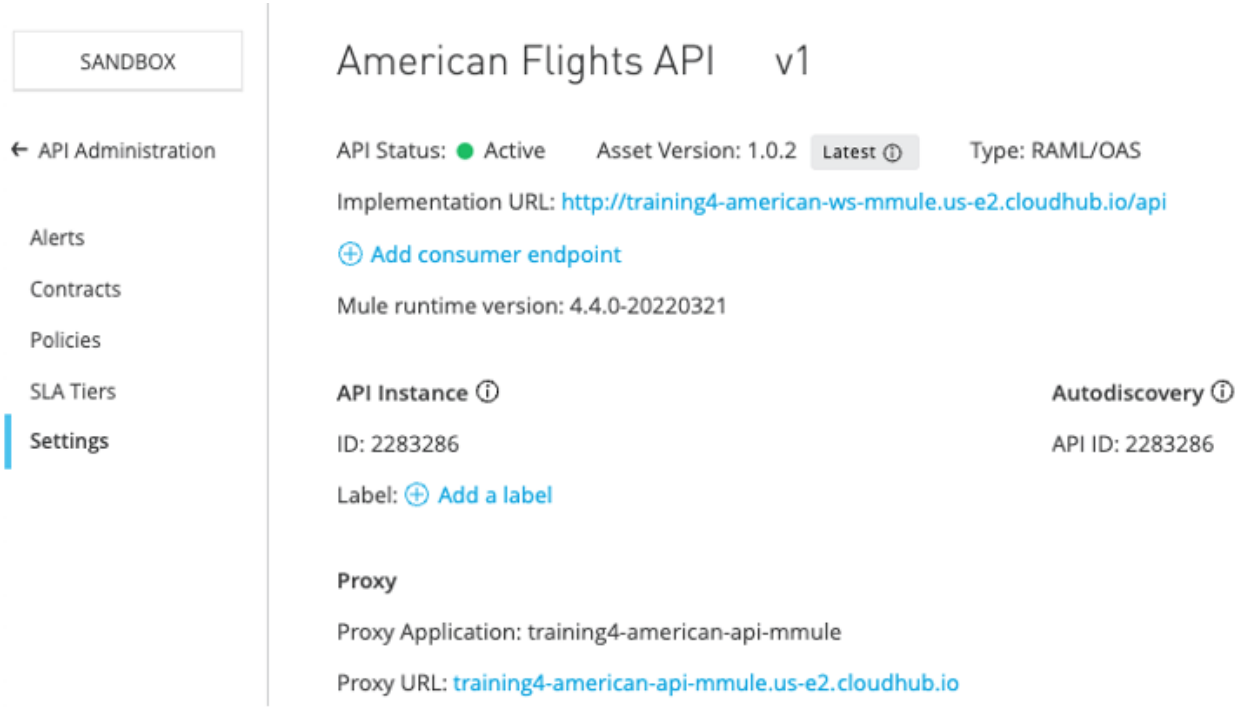
18. In the left-side navigation, select Applications; you should see the proxy application listed along with your original application deployed from Anypoint Studio.
19. Click the row for the proxy and review its information in the right section of the window.



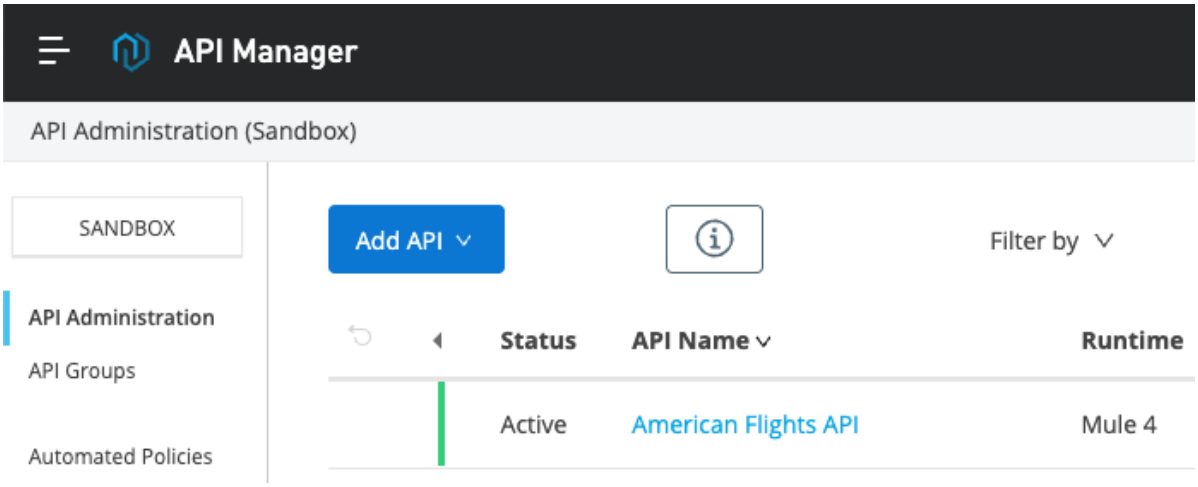
20. Close the browser tab.

View API details in API Manager

21. Return to the tab with API Manager and review the API proxy information at the top of the page.

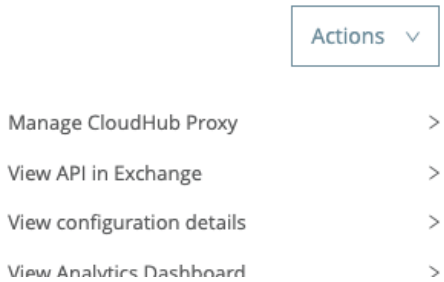


22. In the left-side navigation, click the API Administration link; you should now see your American Flights API listed.

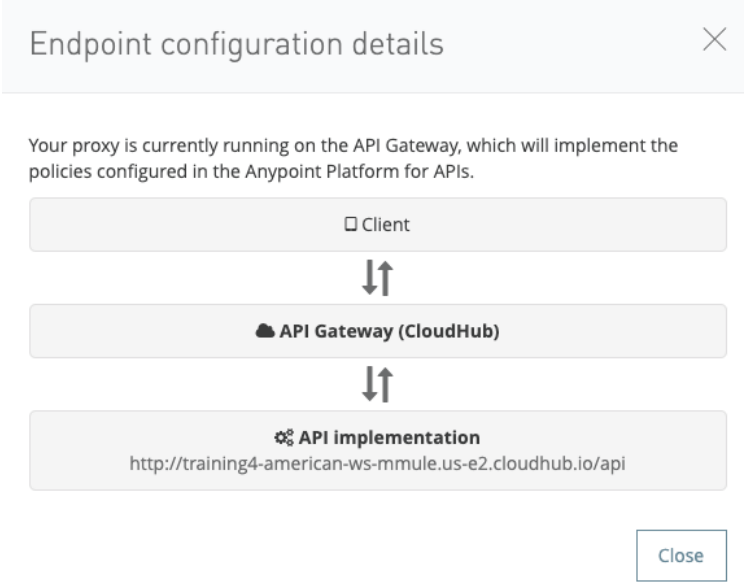


23. In the API list, click the name of the API; you should be returned to the Settings page for the API.

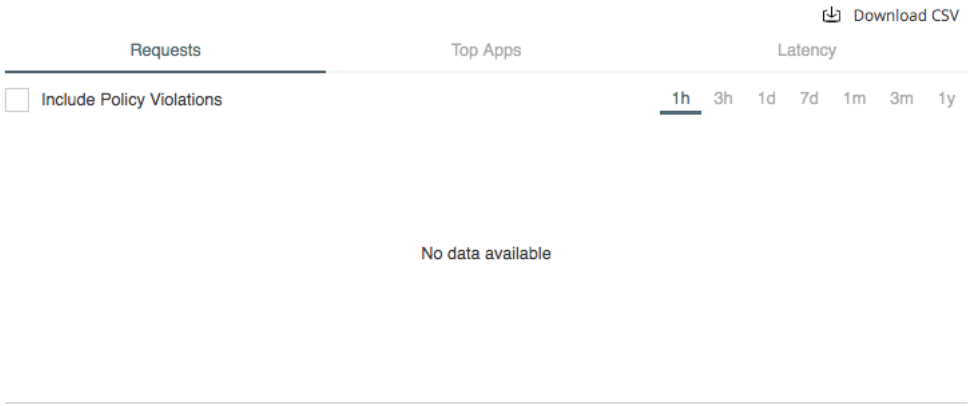
24. Click the View configuration details link in the upper-right corner.



25. In the Endpoint configuration details dialog box, click Close.

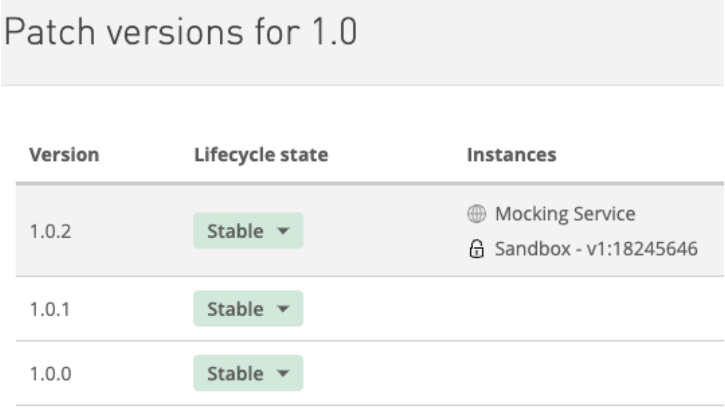


26. On the Settings page, look at the requests graph in the Key Metrics section; you should not see any API requests yet.



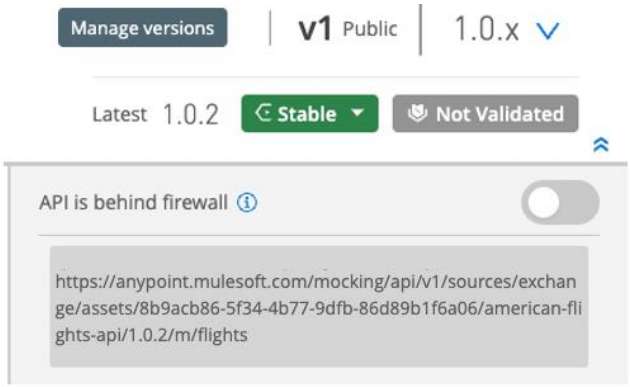
View the new API proxy instance in Exchange

27. Return to the browser tab with Exchange.
28. Return to the home page for your American Flights API.
29. Click the Manage versions button to locate the API instances now associated with asset version 1.0.2; you should see the Mocking Service instance and now the new proxy.

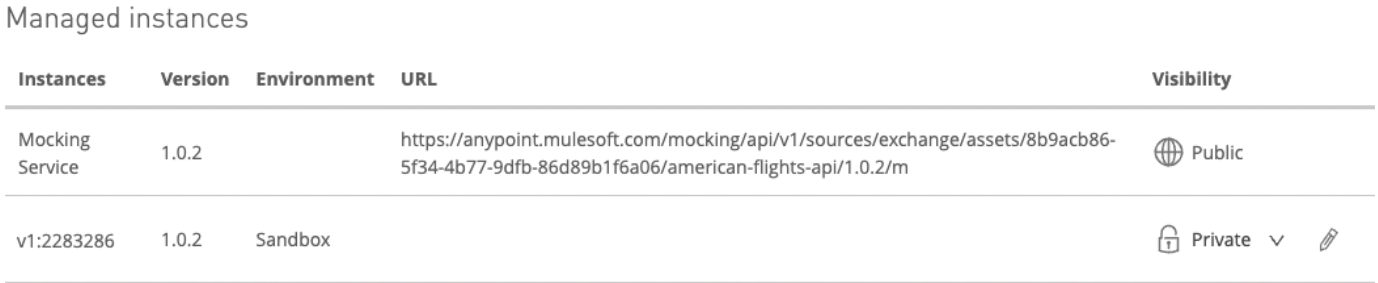


Note: You may need to refresh your browser page before clicking Manage versions.

30. Click Close.
31. Click the GET method for the flights resource.
32. In the API console, notice that there is no drop-down menu to select an instance and the URL indicates that only the mocking service is available.

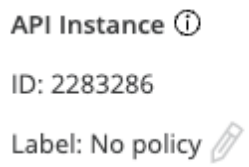


33. In the left-side navigation, select API instances; you should see that the new proxy instance does not have a URL.



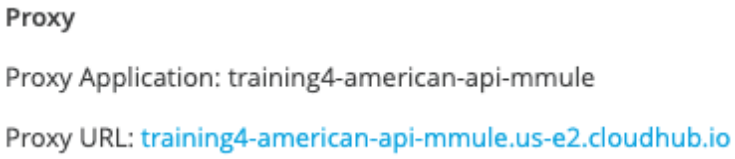
Set a friendly label for the API instance in API Manager

34. Return to the browser tab with API Manager.
35. On the Settings page for your American Flights API, click the Add a label link.
36. Set the label to No policy and press Enter/Return.



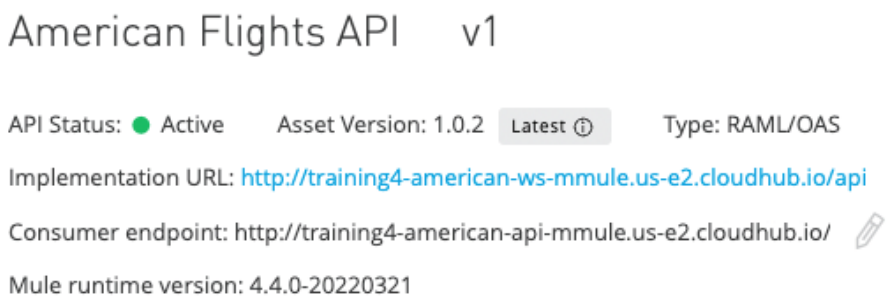
Set a consumer endpoint for the proxy in API Manager

37. Locate the proxy URL.



38. Right-click it and copy the link address.
39. Click the Add consumer endpoint link.

40. Paste the value of the proxy URL.
41. Press Enter/Return.



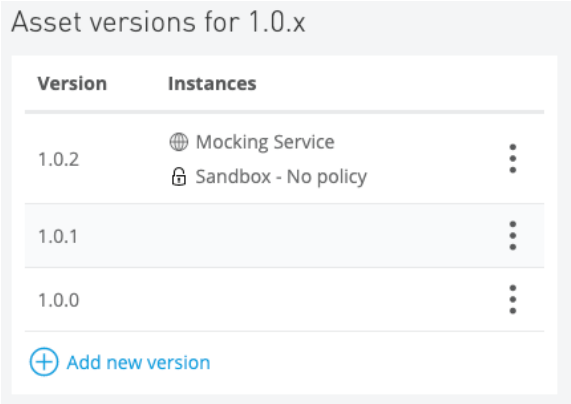
Make requests to the API proxy from Exchange

42. Return to the browser tab with Exchange.
43. Refresh the API instances page for your American Flights API; you should see the new label and the URL.

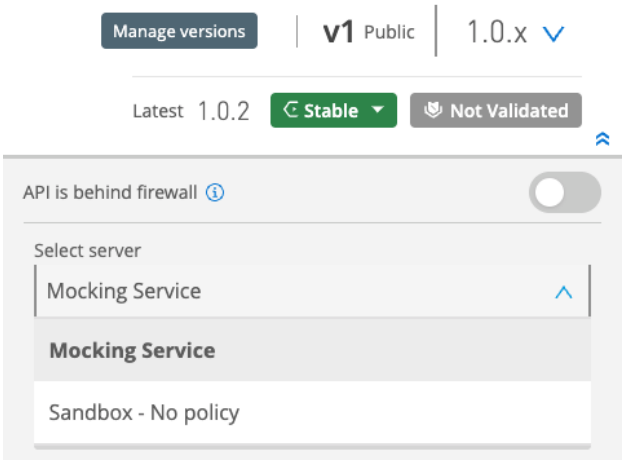
Managed instances

Instances	Version	Environment	URL
Mocking Service	1.0.2		https://anypoint.mulesoft.com/mocking/api/v1/sources/exchange/assets/135f2a50-c700-4a79-934b-8715e9c432b1/american-flights-api/1.0.2/m
No policy	1.0.2	Sandbox	http://training4-american-api-mmule.us-e2.cloudhub.io/

44. In the left-side navigation, select Home to return to the API's main page.
45. Click the Manage versions button to locate the API instances now associated asset version 1.0.2; you should see the new label for the API instance.



46. Click Close.
47. Click the GET method for the flights resource.
48. In the API console, you should now see a drop-down menu to select a server; click the drop-down arrow and notice your API proxy instance is now available as a choice.



49. Select the Sandbox - No policy instance.
50. Click the Send button; you should now see the real data from the database, which contains multiple flights.

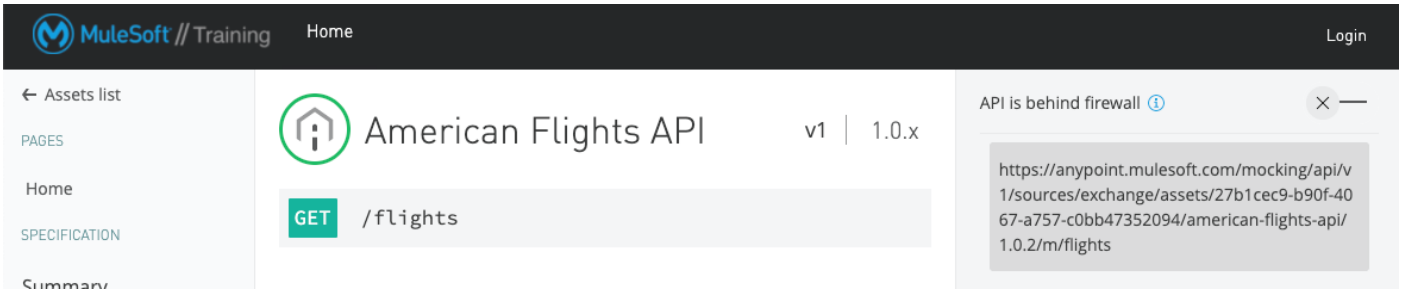
200 OK Time: 2068.1 ms


```
[
  {
    "ID": 1,
    "code": "rree0001",
    "price": 541,
    "departureDate": "2016-01-20T00:00:00",
    "origin": "MUA",
    "destination": "LAX",
    "emptySeats": 0,
    "plane": {
      "type": "Boeing 787",
      "totalSeats": 200
    }
  },
  {
    "ID": 2,
    "code": "eefd0123",
    "price": 300
```


51. Make several more calls to this endpoint.
52. Make calls to different methods.

Make requests to the API proxy from the public portal

53. Return to the public portal in the private/incognito window.
54. Click the GET method for the flights resource.
55. In the API console, notice that there is no drop-down menu to select an instance and the URL indicates that only the mocking service is available.



Make an API instance visible in the public portal

56. Return to the browser with Exchange.
57. In the left-side navigation for American Flights API, select API instances.
58. Change the visibility of the No policy instance from private to public.

API instances

You can add API instances from API Manager or non-managed instances on this page

Managed instances

Instances	Version	Environment	URL	Visibility
Mocking Service	1.0.2		https://anypoint.mulesoft.com/mocking/api/v1/sources/exchange/assets/135f2a50-c700-4a79-934b-8715e9c432b1/american-flights-api/1.0.2/m	Public
No policy	1.0.2	Sandbox	http://training4-american-api-mmule.us-e2.cloudhub.io/	Public

59. Return to the public portal in the private/incognito window.
60. Refresh the page.
61. In the API console, change the API instance from Mocking Service to Sandbox - No policy.

API is behind firewall

Select server

Sandbox - No policy

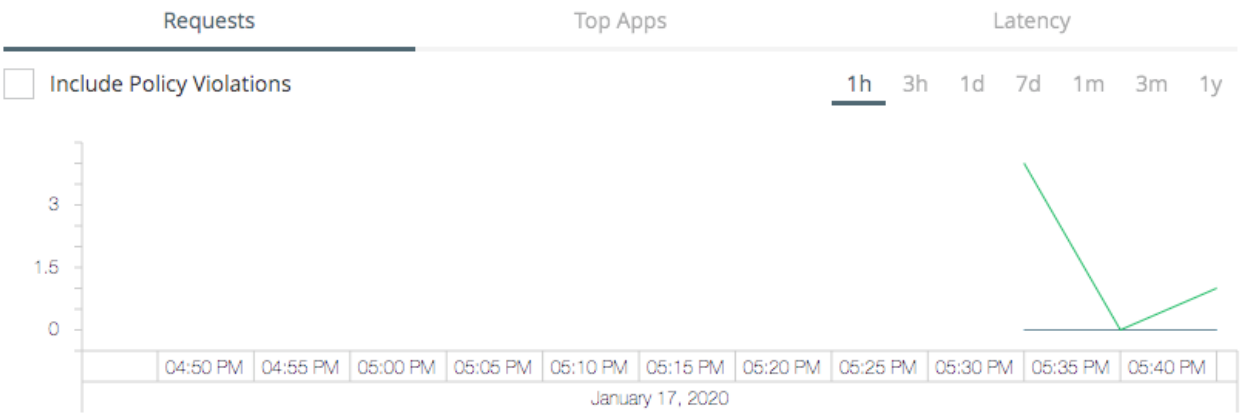
http://training4-american-api-m

62. Click Send; you should a 200 response and flight data.

Look at the API request data

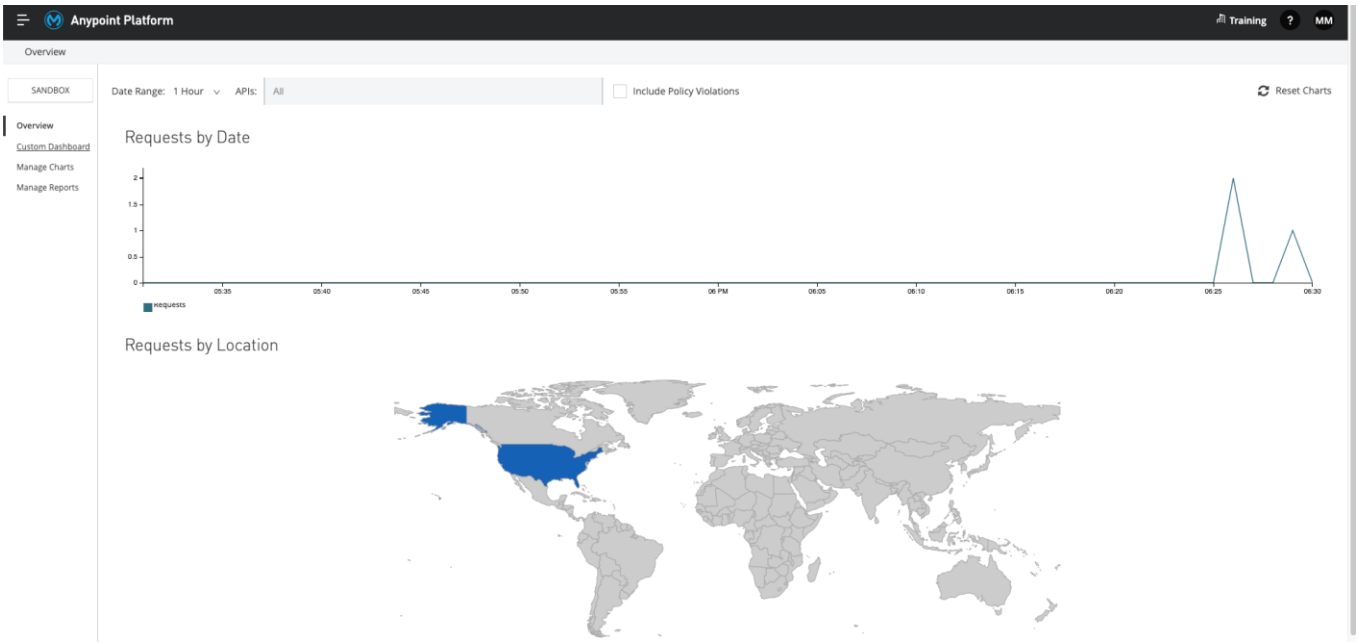
63. Return to the browser tab with API Manager.
64. Refresh the Settings page for your American Flights API.

65. Look at the Request chart again; you should now see data for some API calls.



Note: You may have to wait for the data to populate.

66. Click the View Analytics Dashboard link located in the upper-right corner.
67. In the Date Range drop-down menu near the upper-left corner, select 1 Hour.
68. Review the data in the dashboard.

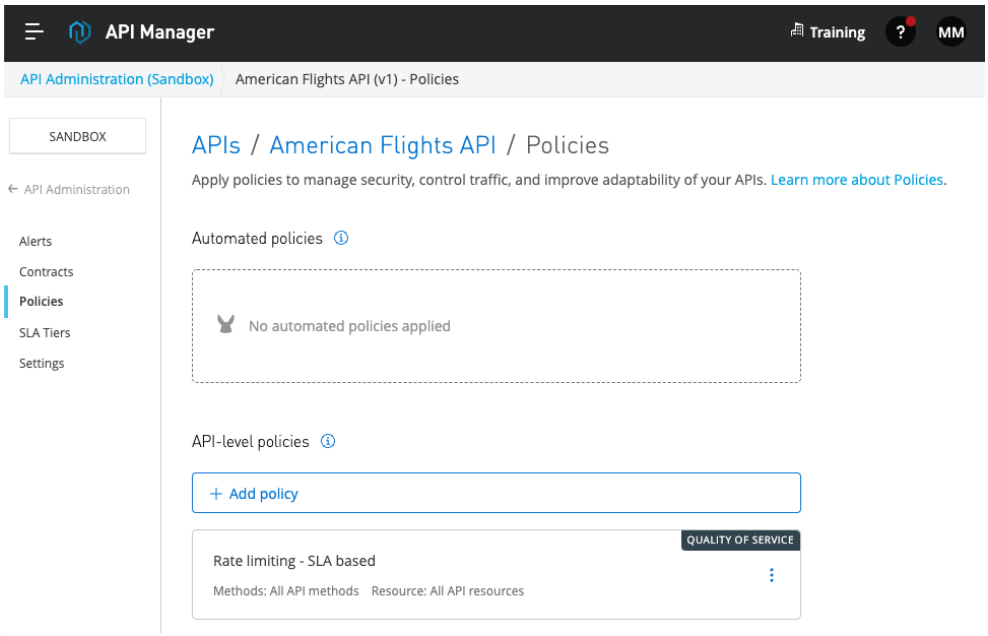


69. Close the browser tab.

Walkthrough 5-3: Restrict API access with policies and SLAs

In this walkthrough, you govern access to the API proxy. You will:

- Add and test a rate limiting policy.
- Add SLA tiers, one with manual approval required.
- Add and test a rate limiting SLA based policy.



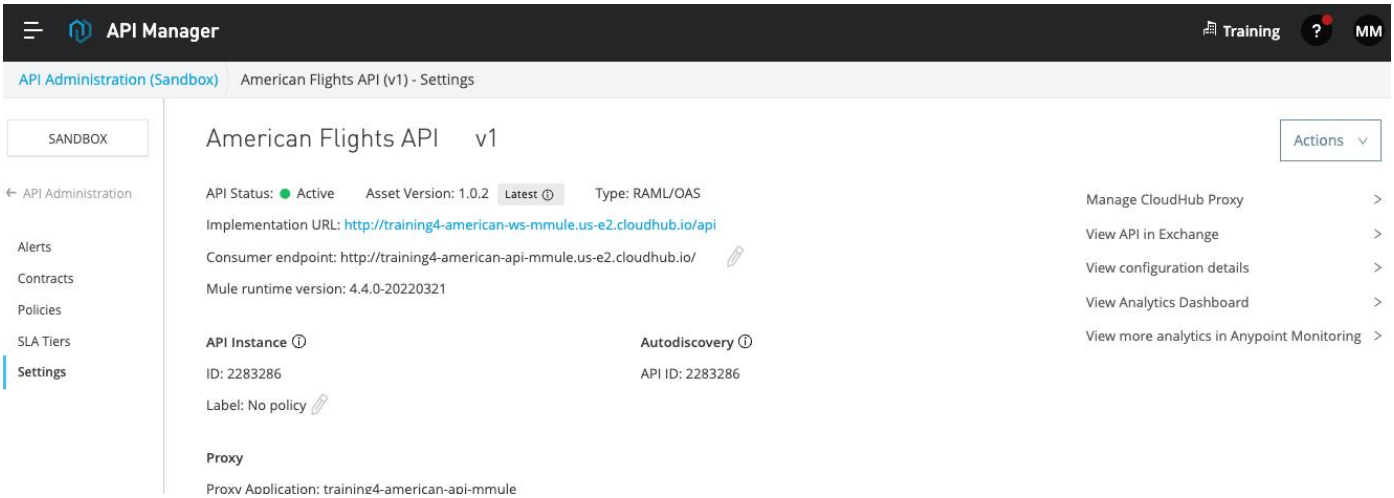
Starting file

This walkthrough uses Anypoint Platform. There is no starting file. To complete the walkthrough, you must have completed the preceding walkthrough.

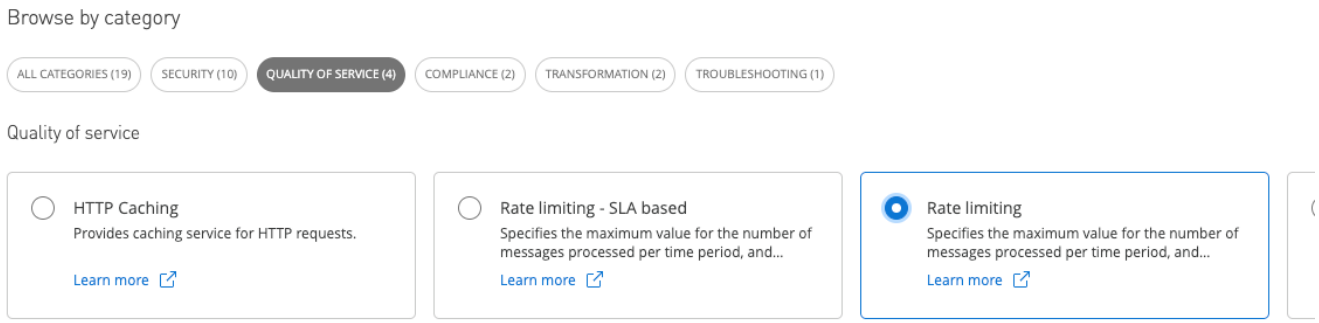
Create a rate limiting policy

1. Return to the Settings page for your American Flights API in API Manager.

2. In the left-side navigation, select Policies.



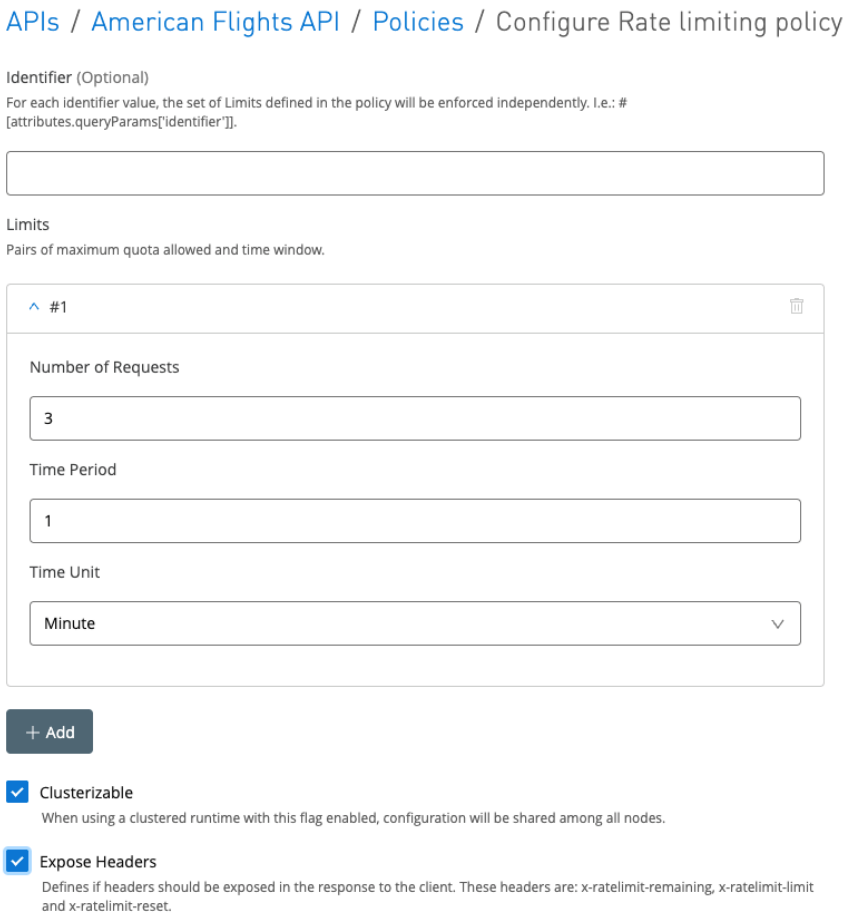
3. Under API-level policies, click the Add Policy button.
4. Under Browse by category, select quality of service.
5. Choose the Rate limiting policy.



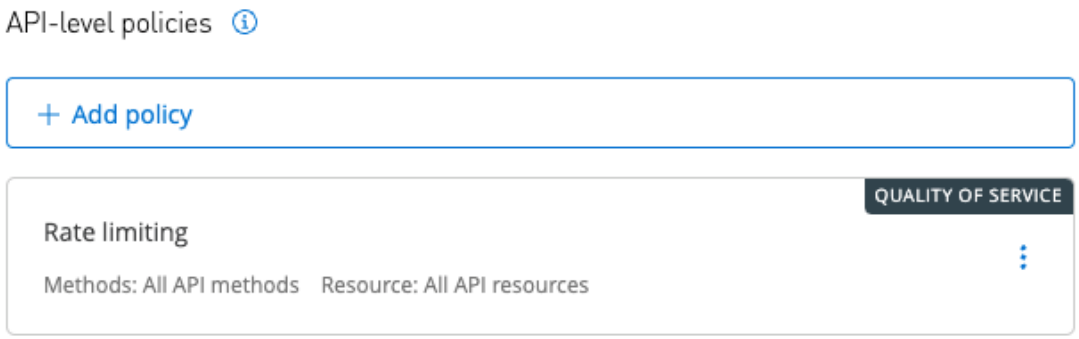
6. Click Next.
7. On the Configure Rate limiting policy page, set the following values and click Apply:

- Number of Requests: 3
- Time Period: 1
- Time Unit: Minute

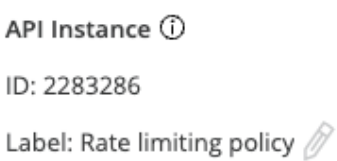
8. Select Expose Headers.



9. Expand Advanced options and examine the additional fields and values.
10. Click Apply; you should see the policy listed under API-level policies.



11. In the left-side navigation, select Settings.
12. Change the API instance label to Rate limiting policy.

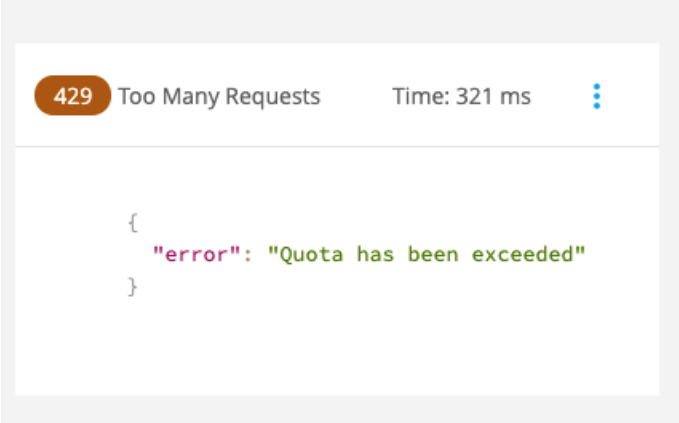


Test the new rate limiting policy

13. Return to the browser tab with your American Flights API in Exchange.
14. In the left-side navigation, select the /flights GET method.
15. Select the Sandbox – Rate limiting policy API instance.

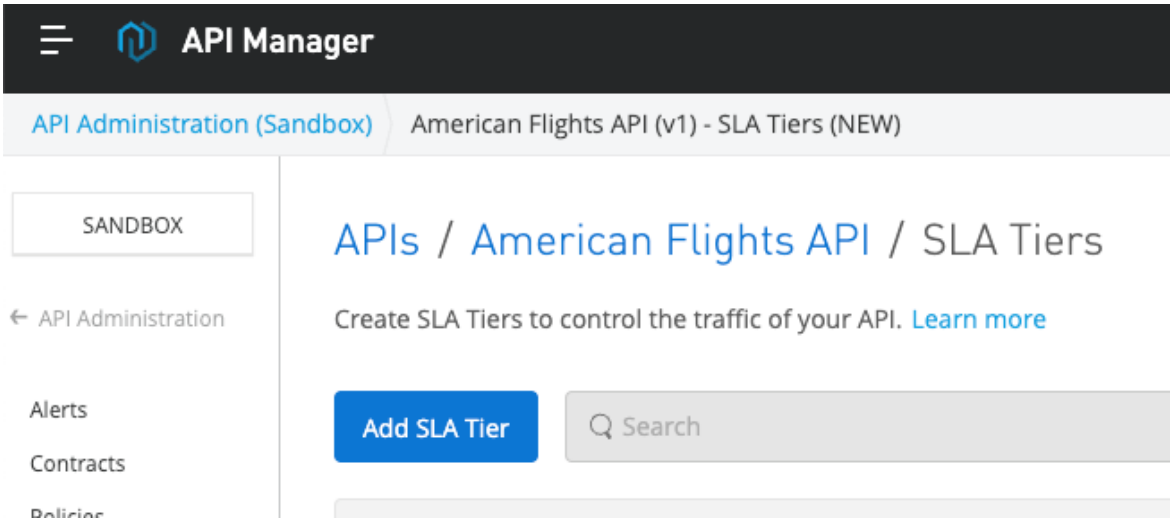
Note: You may need to refresh the page to see the new label for the API instance.

16. Press Send until you get a 429 Too Many Requests response.



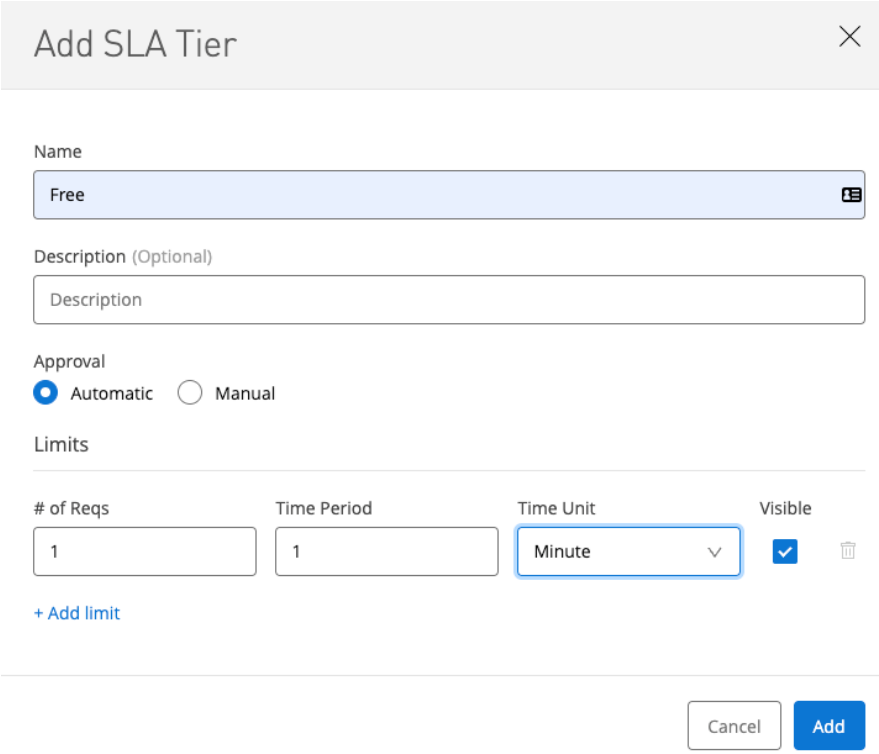
Create SLA tiers

17. Return to the browser tab with your American Flights API in API Manager.
18. In the left-side navigation, select SLA Tiers.
19. Click the Add SLA tier button.



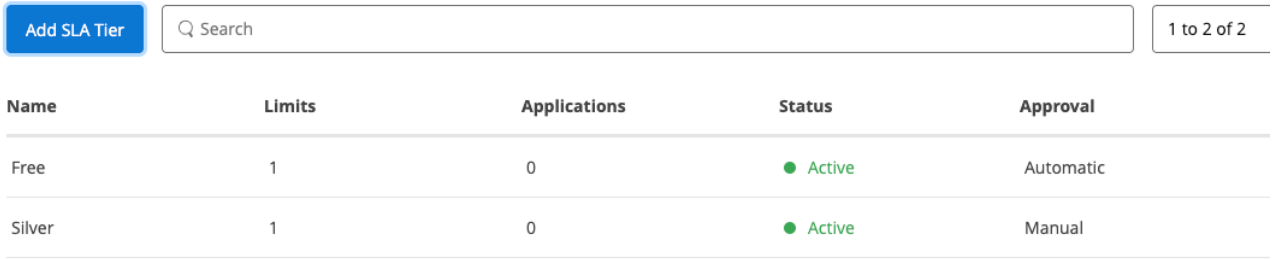
20. In the Add SLA tier dialog box, set the following values:

- Name: Free
- Approval: Automatic
- # of Reqs: 1
- Time Period: 1
- Time Unit: Minute



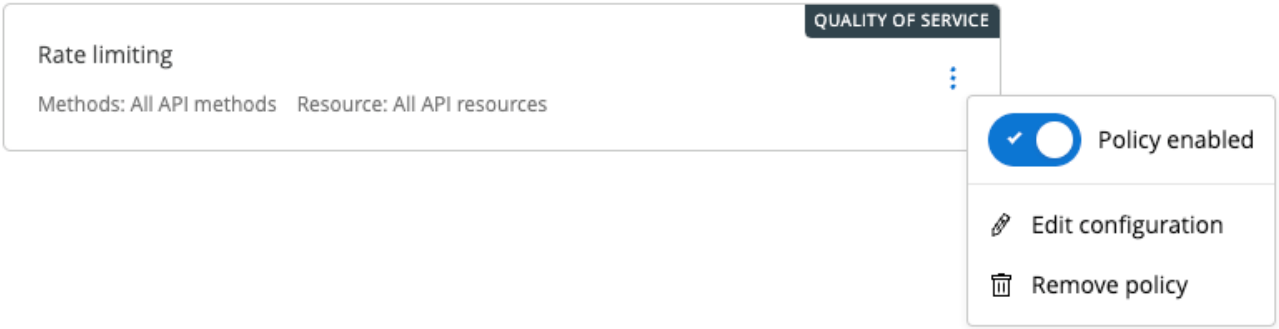
21. Click the Add button.
22. Create a second SLA tier with the following values:

- Name: Silver
- Approval: Manual
- # of Reqs: 1
- Time Period: 1
- Time Unit: Second

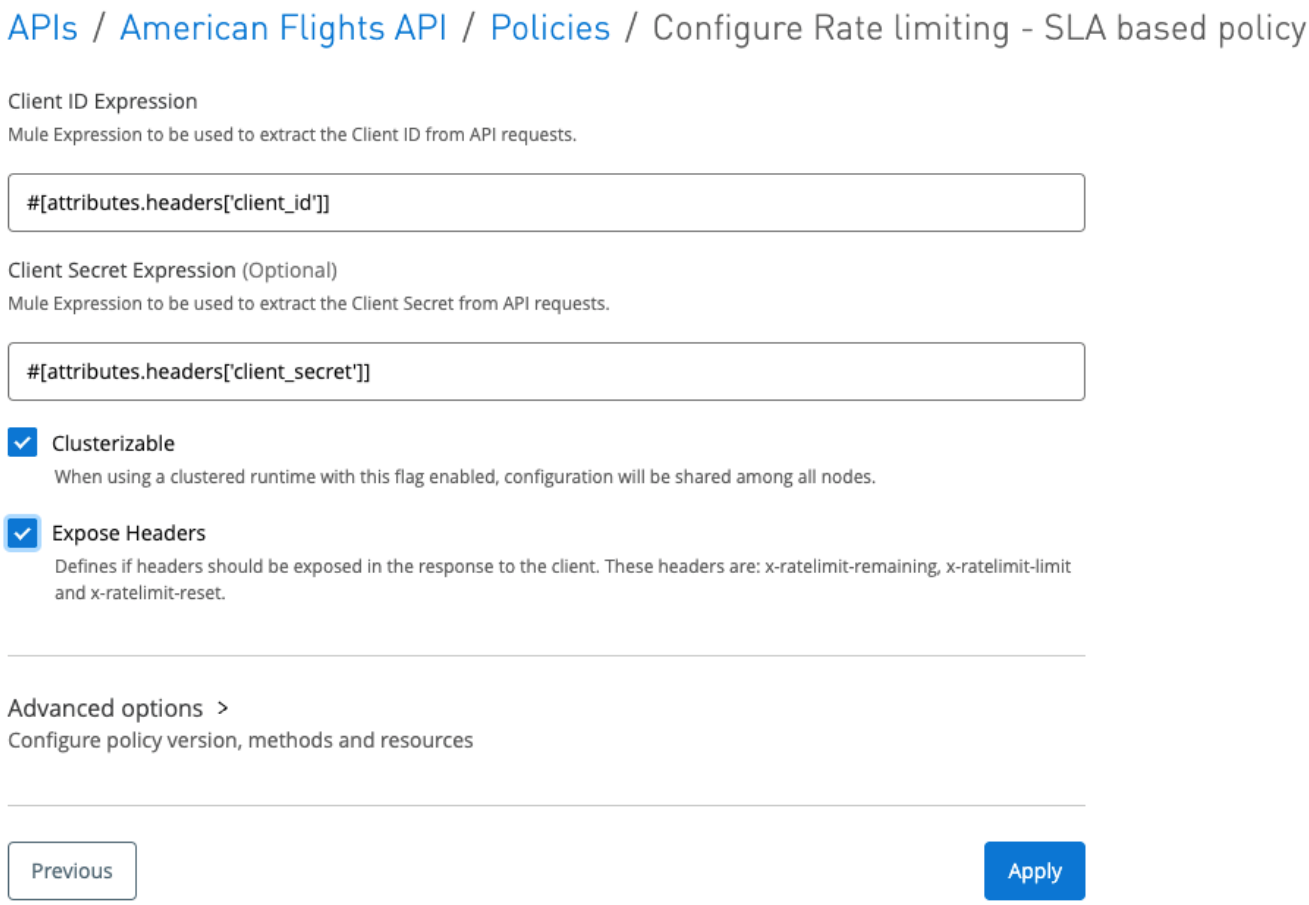


Change the policy to rate limiting – SLA based

23. In the left-side navigation, select Policies.
24. In the Rate limiting policy options menu, select Remove policy.



25. In the Remove policy dialog box, click Remove.
26. Click the Add Policy button.
27. In the quality of service category, choose the Rate limiting - SLA based policy and click Next.
28. On the Configure Rate limiting – SLA based policy page, look at the expressions and see that a client ID and secret need to be sent with API requests as headers.
29. Select Expose Headers.



30. Click Apply; you should see the policy listed under API-level policies.
31. In the left-side navigation, select Settings.
32. Change the API instance label to Rate limiting – SLA based policy.

API Instance ⓘ

ID: 2283286

Label: Rate limiting - SLA based policy

Test the rate limiting – SLA based policy in Exchange

33. Return to the browser tab with your API in Exchange.
34. Refresh the page and select the /flights GET method to make a call to the Sandbox – Rate limiting – SLA based policy.
35. Click Send; you should get a 401 Unauthorized response with the error Invalid client id or secret.

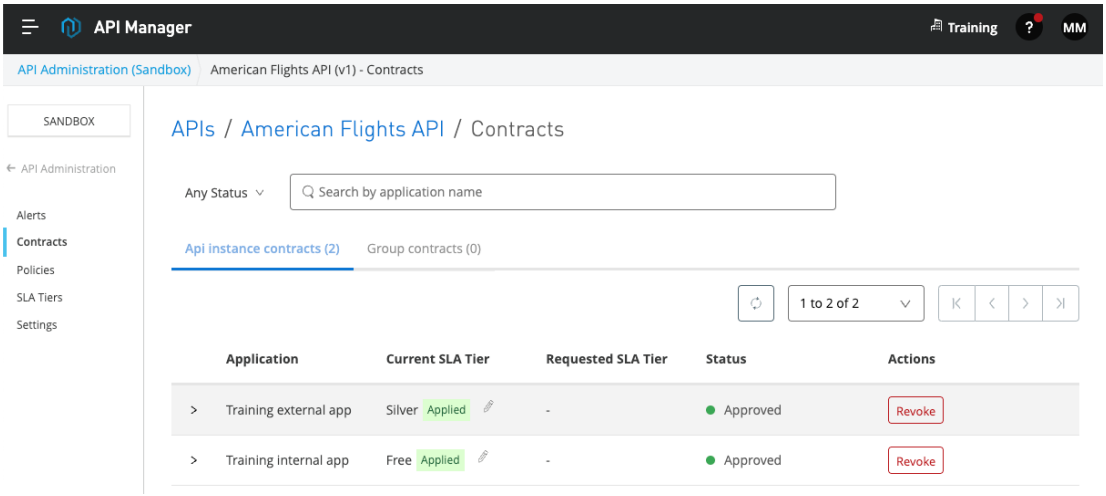
401 Unauthorized Time: 422.4 ms

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

Walkthrough 5-4: Request and grant access to a managed API

In this walkthrough, clients request access to an API proxy and administrators grant access. You will:

- Request application access to SLA tiers from private and public API portals.
- Approve application requests to SLA tiers in API Manager.



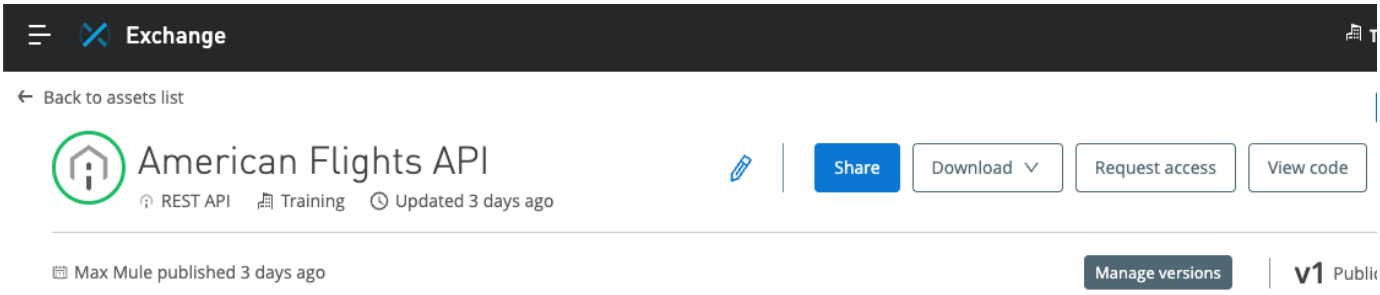
Starting file

This walkthrough uses Anypoint Platform. There is no starting file. To complete the walkthrough, you must have completed the preceding walkthrough.

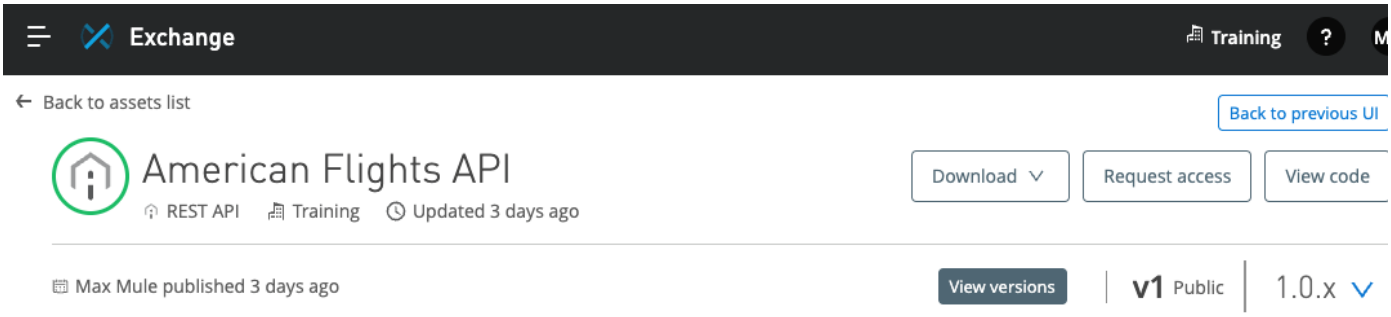
Request access to the API as an internal consumer

1. Return to the browser tab with Anypoint Exchange.
2. In the left-side navigation, select Home to return to the API's home page.

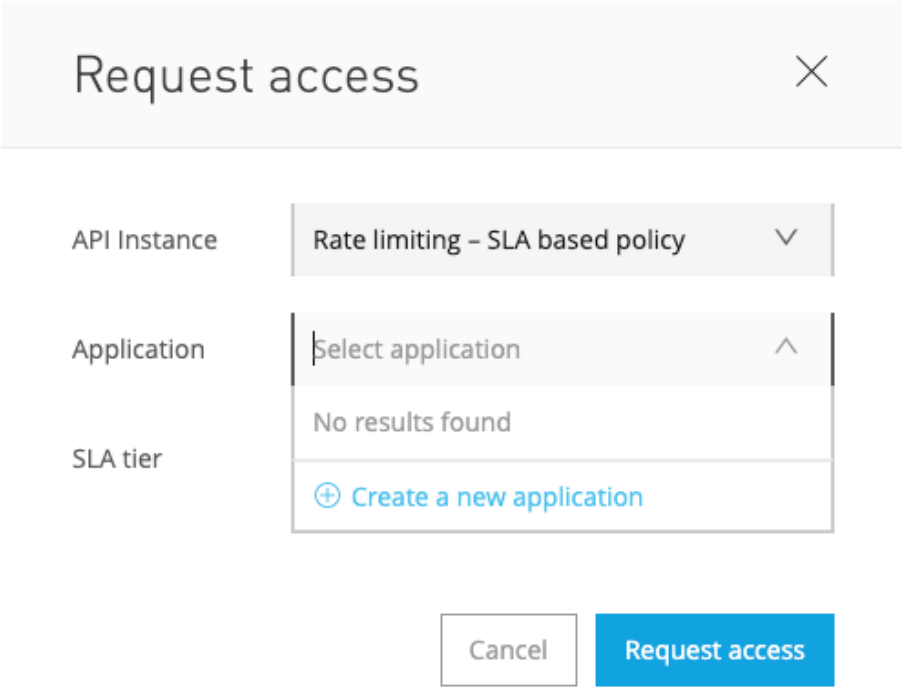
3. Click the Request access button near in the upper-right corner.



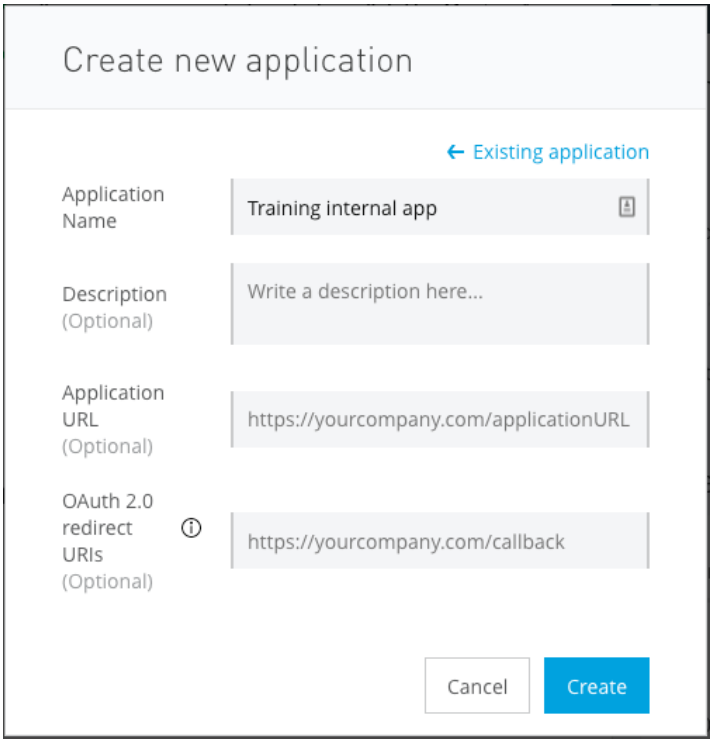
Note: Other internal users that you shared the API with that do not have Edit permissions will see a different menu.



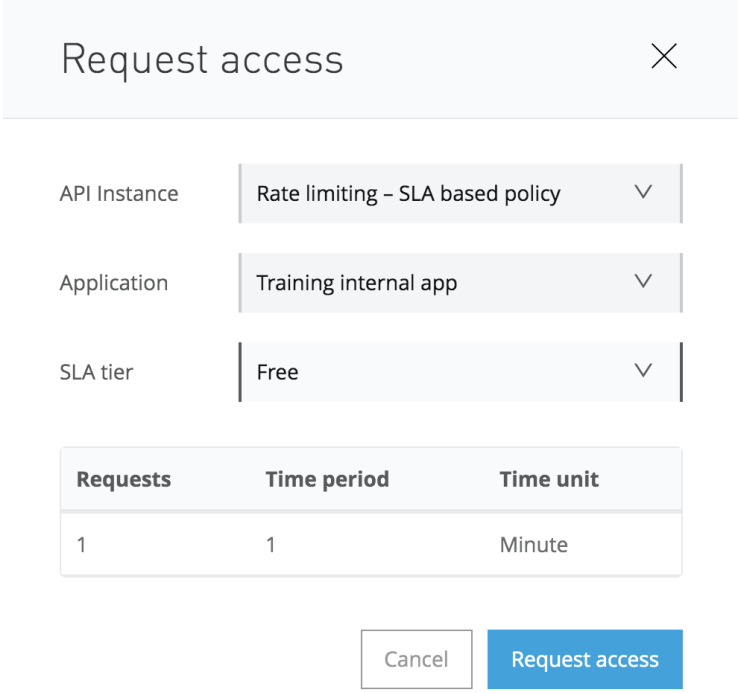
4. In the Request access dialog box, select Rate limiting - SLA based policy in the API Instance drop-down menu.
5. Click the Create a new application link in the Application drop-down menu.



6. In the Create new application dialog box, set the name to Training internal app and click Create.

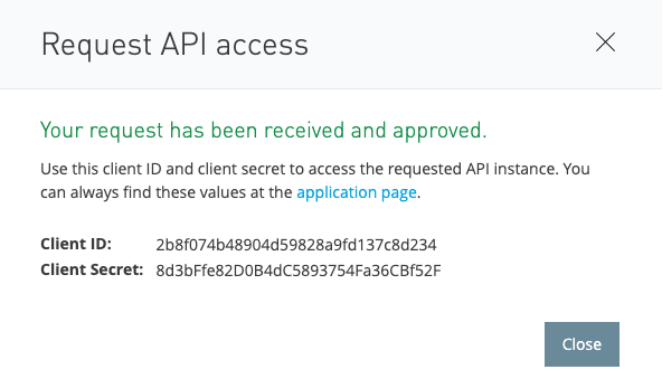


7. In the Request access dialog box, set the SLA tier to Free.



8. Click Request access.

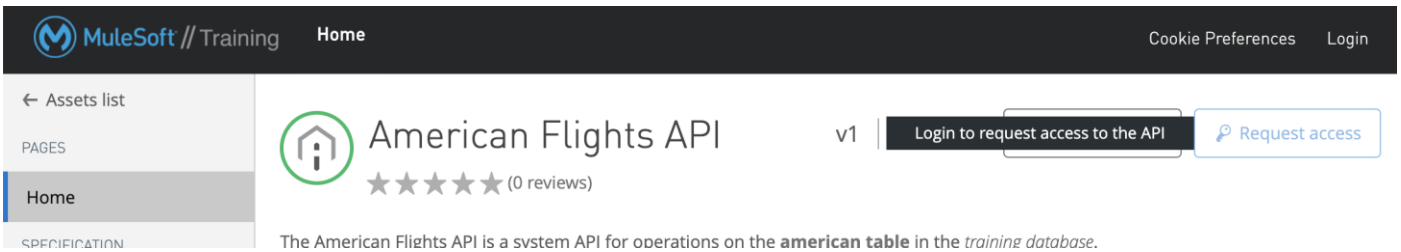
9. In the Request API access dialog box, you should see that your request has been approved; view the assigned values for the client ID and client secret.



10. Click Close.

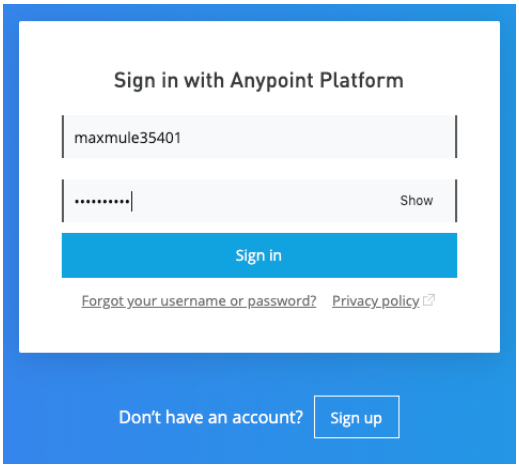
Request access to the API as an external consumer

11. Return to the public portal in the private/incognito window.
12. On the main page for the American Flights API, you should now see a Request access button (refresh if necessary); hover over it and notice the login tooltip.

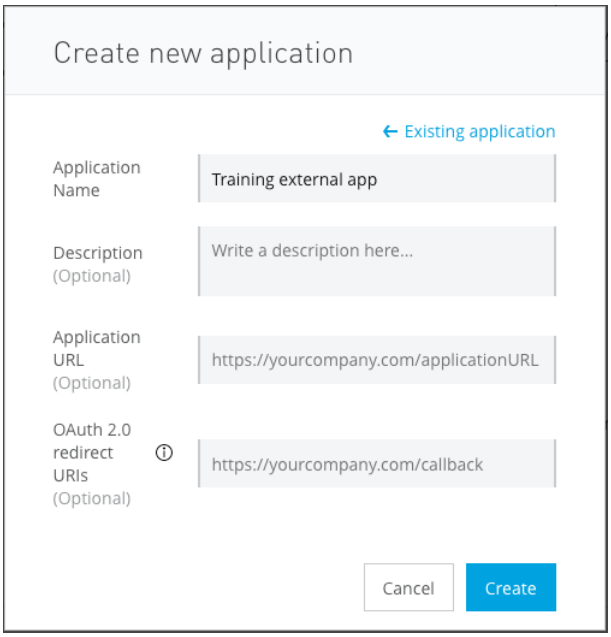


13. Click Login in the upper-right corner; you should get a page to sign in or create an Anypoint Platform account.
14. Enter your existing credentials and click Sign in.

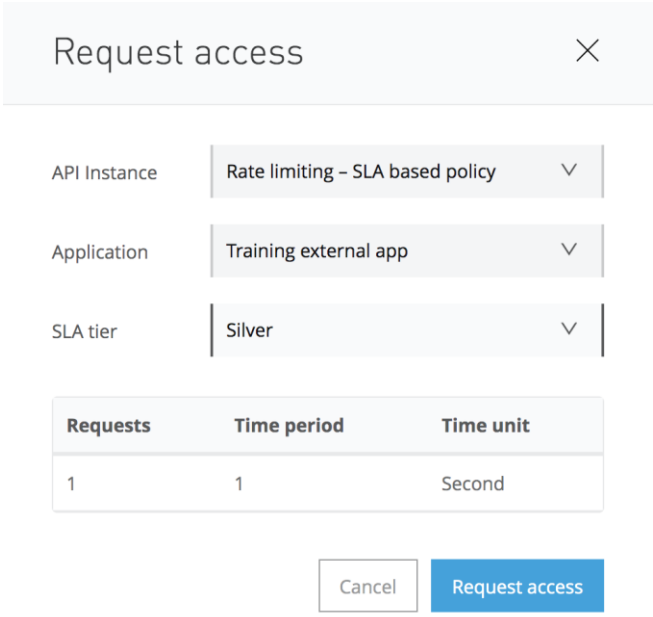
Note: Instead of creating an external user, you will just use your existing account.



15. Back in the public portal, click the Request access button.
16. In the Request access dialog box, select Rate limiting - SLA based policy in the API Instance drop-down menu.
17. Click the Create a new application link in the Application drop-down menu.
18. In the Create new application dialog box, set the name to Training external app and click Create.

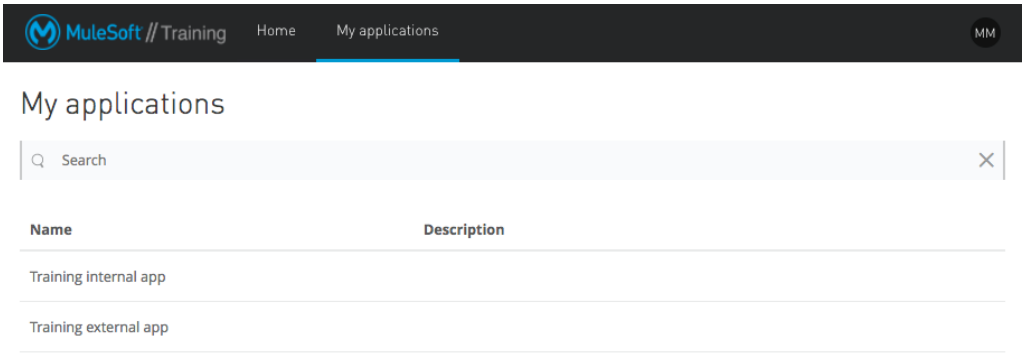


19. In the Request access dialog box, set the SLA tier to Silver.

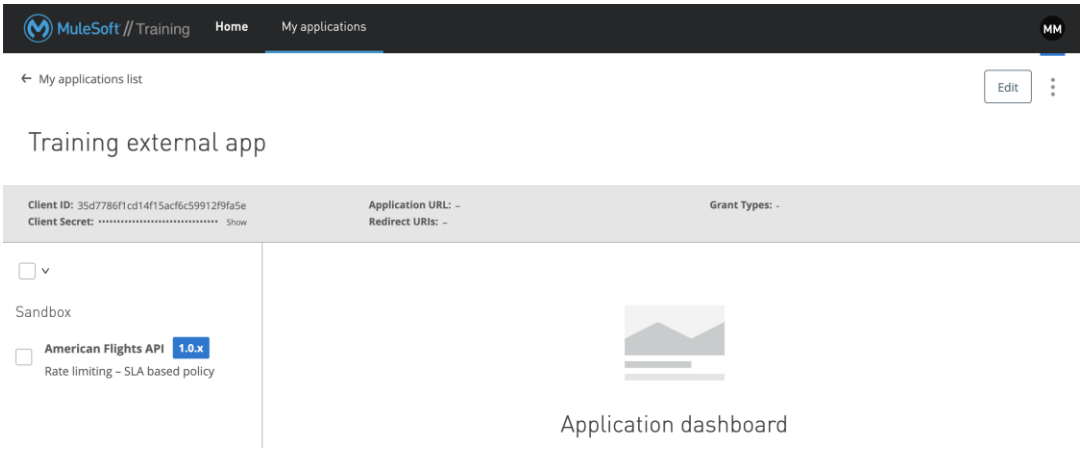


20. Click Request access.
21. In the Request API access dialog box, click Close.

22. In the portal main menu bar, right-click My applications and select to open the link in a new tab; on the new tab, you should see the two applications you created.



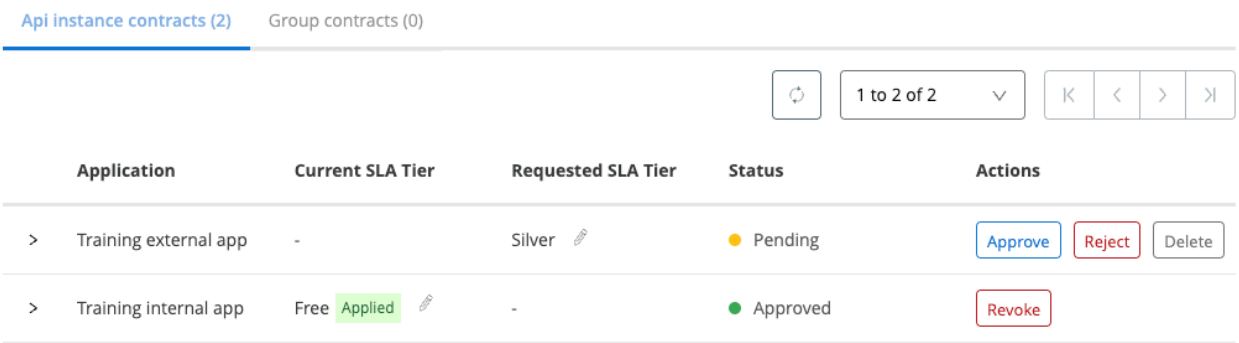
23. Click the link for Training external app; you should see what APIs the application has access to, values for the client ID and secret to access them, and request data.



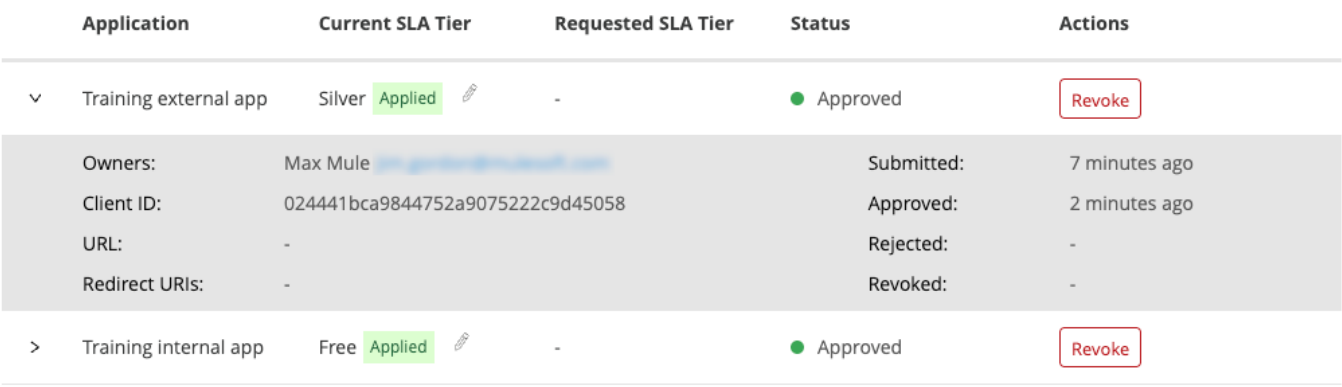
24. Leave this page open in a browser so you can return to it and copy these values.

Grant an application access

25. Return to the browser window and tab with the Settings page for American Flights API v1 in API Manager.
26. In the left-side navigation, select Contracts; you should see the two applications that requested access to the API.



27. Click the Approve button for the application requesting Silver tier access.
28. Expand the Training external app row and review its information.
29. Copy the value of the Client ID.



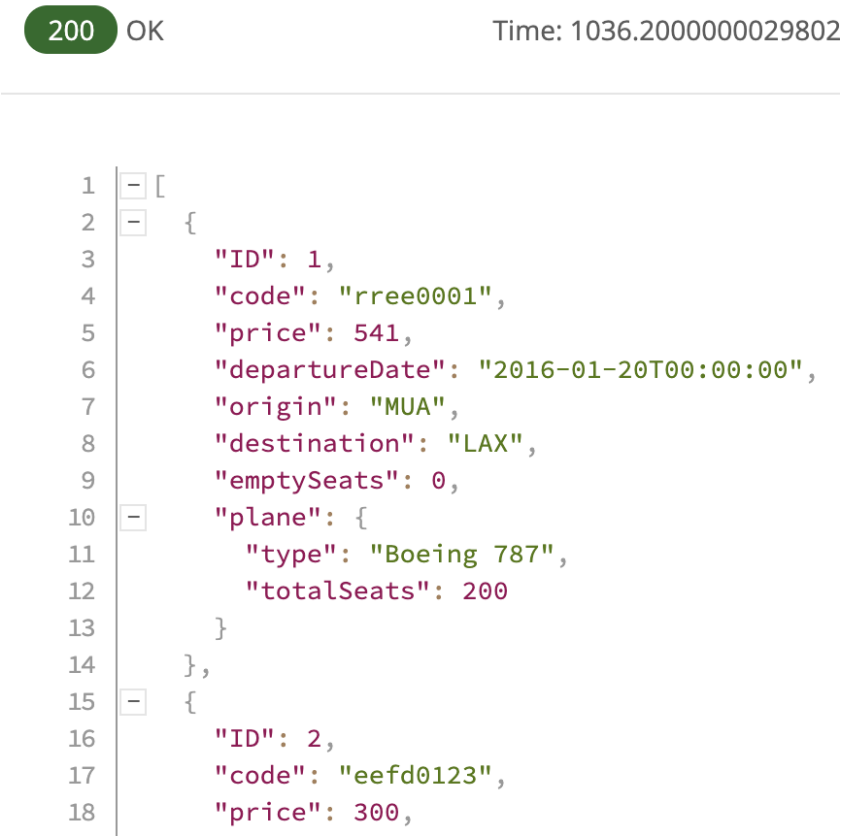
Add authorization headers to test the rate limiting – SLA based policy from an API portal

30. Return to the private/incognito browser window and tab with the American Flights API portal in the public portal.
31. Try again to make a call to the Sandbox – Rate limiting – SLA based policy; you should still get a 401 Unauthorized response.
32. In the Headers section, click the Add link.
33. Set the header name to client_id.
34. Set the value of client_id to the value you copied.

35. Return to the browser tab with My applications in the public portal.
36. Click the Show link next to Client Secret then copy its value.
37. Return to the browser tab with the API console in the public portal.
38. Add another header and set the name to client_secret.

39. Set the client_secret header to the value you copied.

40. In the course snippets.txt file, record these client_id and client_secret values in the section reserved for this module.
41. Click Send; you should now get a 200 response with flight results.

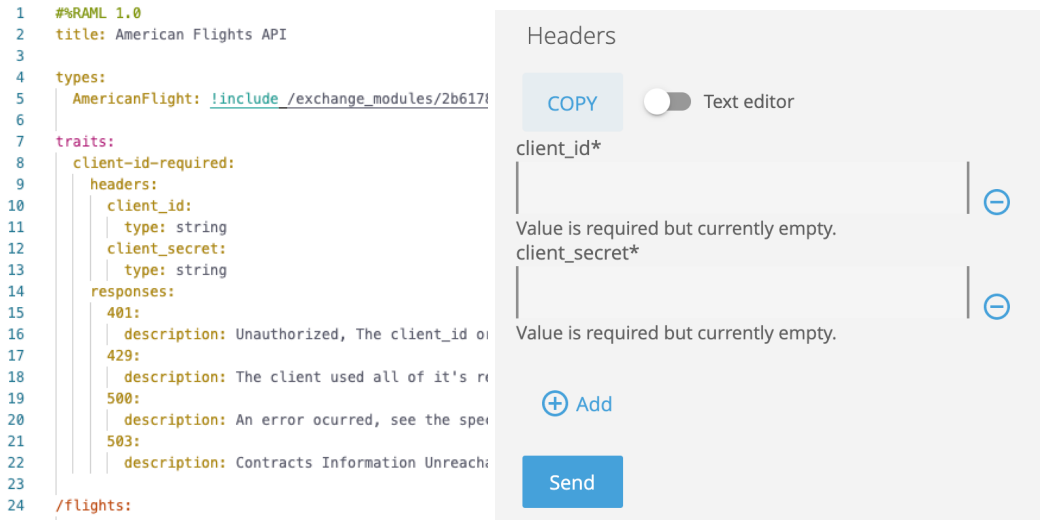


Walkthrough 5-5: Add client ID enforcement to an API specification

In this walkthrough, you add client ID enforcement to the API specification. You will:

- Modify an API specification to require client id and client secret headers with requests.
- Update a managed API to use a new version of an API specification.
- Call a governed API with client credentials from API portals.

Note: If you do not complete this exercise for Fundamentals, the REST connector that is created for the API and that you use later in the course will not have client_id authentication.



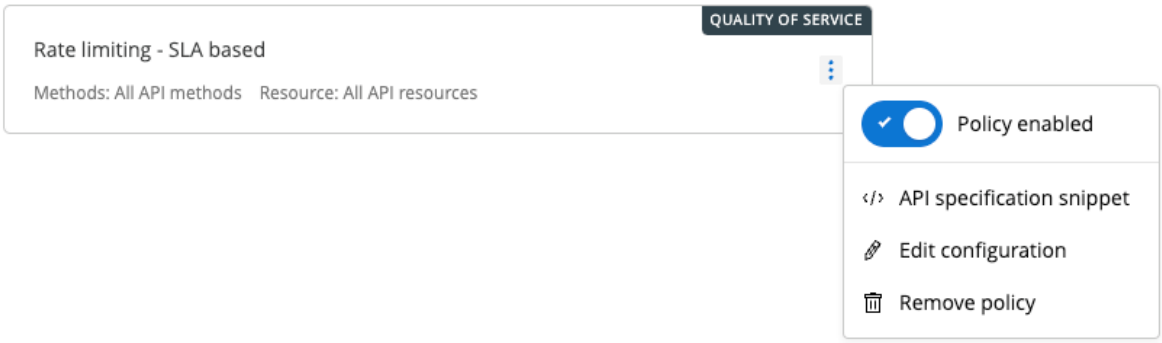
Starting file

This walkthrough uses Anypoint Platform. There is no starting file. To complete the walkthrough, you must have completed the preceding walkthrough.

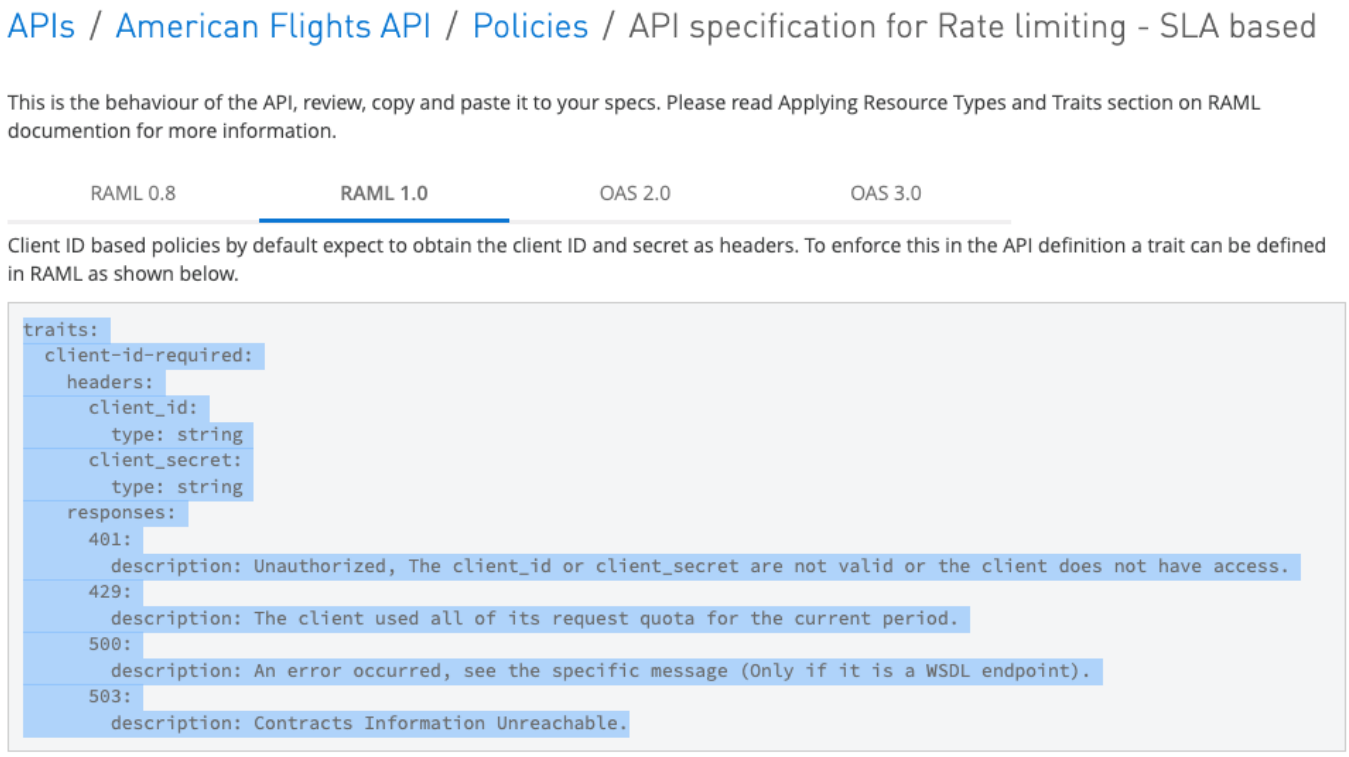
Copy the traits required to add authentication to the API specification

1. Return to the browser tab with American Flights API v1 in API Manager.
2. In the left-side navigation, select Policies.

3. In the Rate limiting – SLA based policy options menu, select API Specification snippet.



4. On the API specification for Rate limiting - SLA based page, select the RAML 1.0 tab.
5. Copy the value for the traits.



6. In the API Manager asset path near the top of the page, click American Flights API.

Add authentication headers to the API specification

7. Return to the browser tab with your API in Design Center.
8. Go to a new line after the types declaration and paste the traits code you copied.

```
#%RAML 1.0
title: American Flights API

types:
  AmericanFlight: !include /exchange_modules/2b617

traits:
  client-id-required:
    headers:
      client_id:
        type: string
      client_secret:
        type: string
    responses:
      401:
        description: Unauthorized, The client_id o
      429:
        description: The client used all of it's r
      500:
        description: An error ocurred, see the spe
      503:
        description: Contracts Information Unreach

/flights:
```

9. Go to a new line after the /flights resource declaration and indent.

```
      503:
        description: Contracts Information Unreachable.

/flights:

  get:
    queryParameters:
      destination:
        required: false
```

10. Add a nested is node with an empty array.

```
is: []
```

```
      503:
        description: Contracts Information Unreachable.

/flights:
  is: []
  get:
    queryParameters:
      destination:
```

11. Make sure the cursor is inside the array brackets and add the client-id-required trait name as an array element.

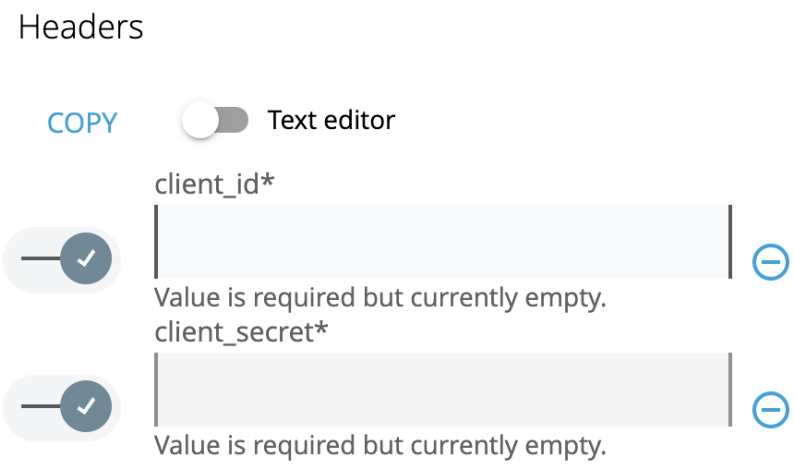
```
24 /flights:
25   is: [client-id-required]
26   get:
27     queryParameters:
```

12. Repeat this process so the trait is applied to all methods of the {ID} resource as well.

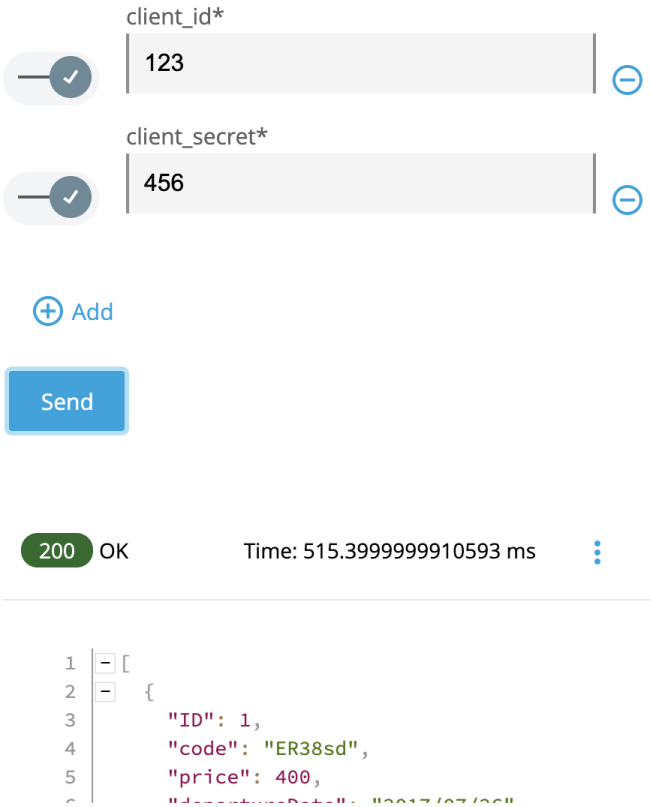
```
55   /{ID}:
56     is: [client-id-required]
57     get:
58       responses:
```

Test the API in the API console in Design Center

13. In the API console, select one of the resources and click Try it.
14. In the Headers section, you should now see fields to enter client_id and client_secret.
15. Look at both fields; you should see value is required messages for each.



16. Enter any values for the client_id and client_secret and click Send; you should get a 200 response with the example results.

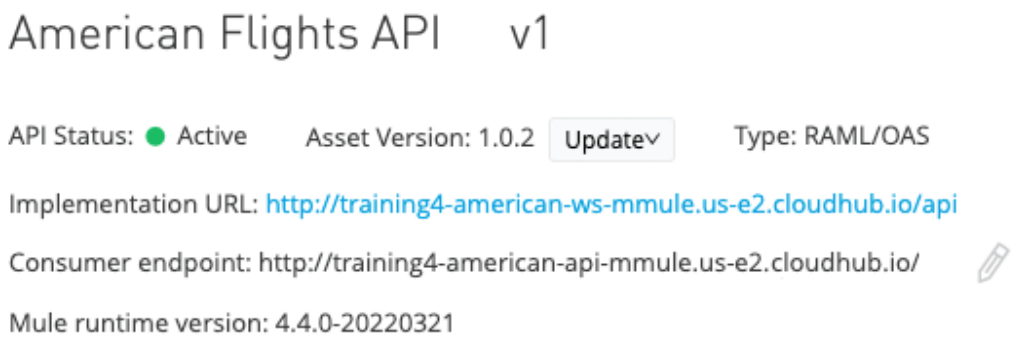


Publish the new version of the API to Exchange

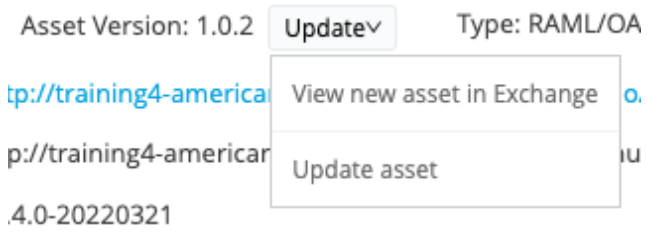
17. Click the Publish button.
18. In the Publishing to Exchange dialog box, examine the asset version then click the Publish to Exchange button.
19. Wait for the API to publish then in the resultant dialog box, click Close.

Update the managed API instance to use the new version of the API specification

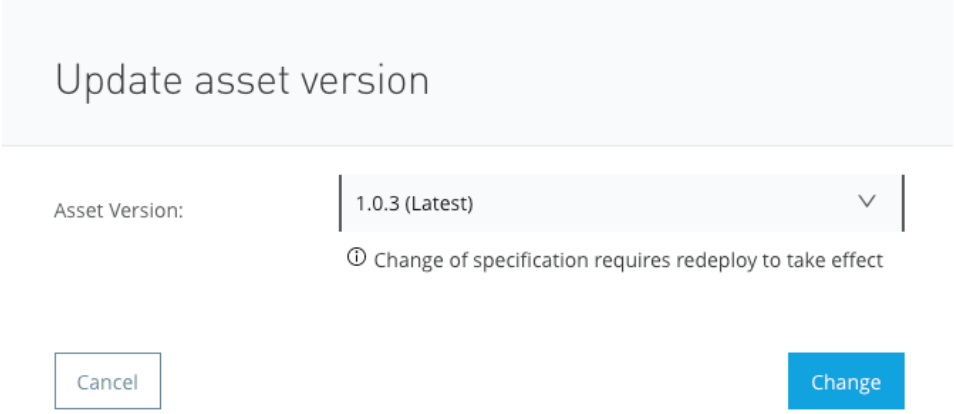
20. Return to browser tab with American Flights API v1 in API Manager.
21. Refresh the page then locate the asset version displayed at the top of the page; you should see 1.0.2 and a new Update drop-down menu next to it.



22. Select Update asset in the Update drop-down menu.



23. In the Update asset version dialog box, select 1.0.3 (Latest) in the drop-down menu.

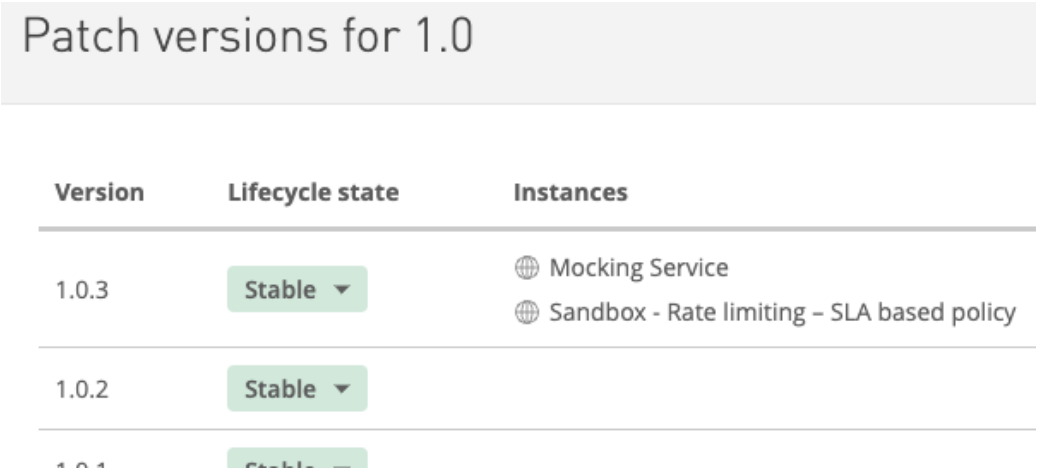


24. Click Change; you should see the 1.0.3 asset version displayed at the top of the page with the Latest label next to it.

Note: You may need to refresh the page to see the new version and label.

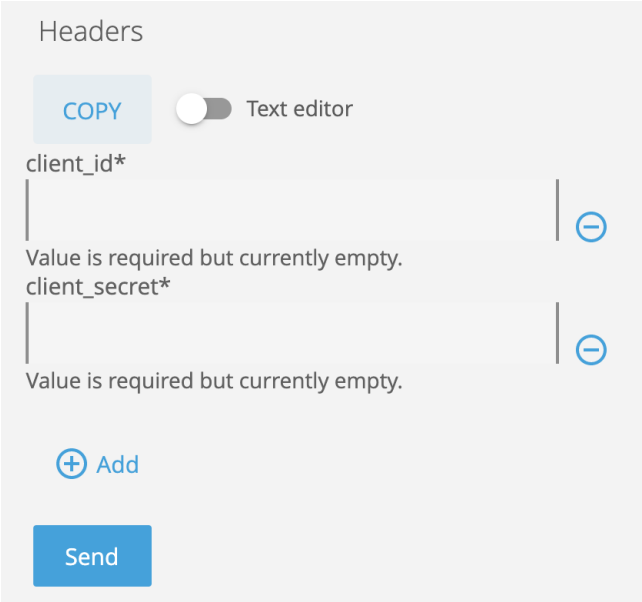
Test the rate limiting – SLA based policy in the API console in Exchange

25. Return to the browser tab with Exchange.
26. Return to the home page for the API and click the Manage versions button; you should see the new version listed with instances for both the Mocking Service instance and the new proxy.



27. Click Close.
28. Click the GET method for the flights resource; you should see required text fields for client_id and client_secret and no longer need to add the headers manually for each request.

Note: You will test and use the authentication with the REST connector later in the Fundamentals course.



29. Close all Anypoint Platform browser windows and tabs.