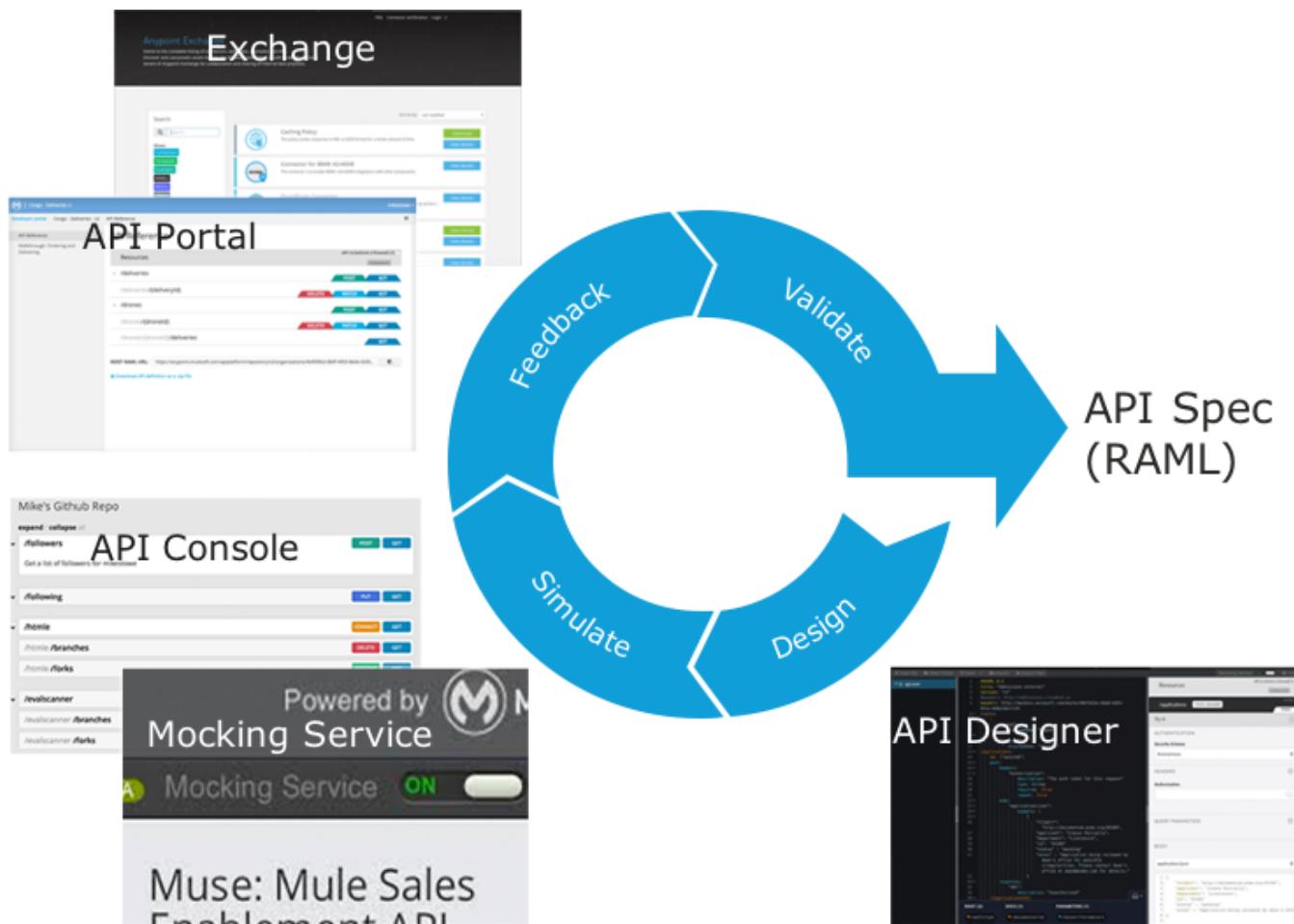


Module 3: Designing APIs



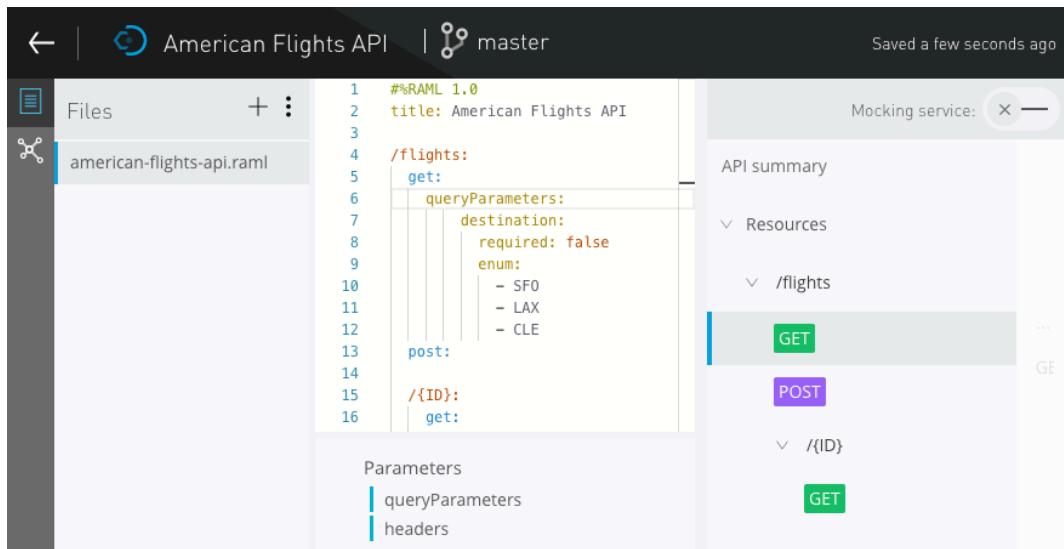
At the end of this module, you should be able to:

- 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.
- Interact with an API using the API console.



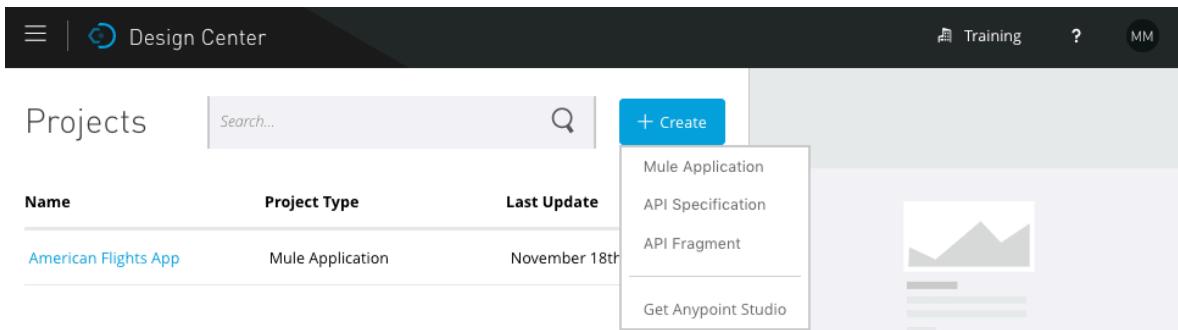
The screenshot shows the MuleSoft Anypoint Studio interface. On the left, there's a sidebar with 'Files' and a '+' button. A file named 'american-flights-api.raml' is selected. The main area displays the RAML code:

```
1  #%RAML 1.0
2  title: American Flights API
3
4  /flights:
5    get:
6      queryParameters:
7        destination:
8          required: false
9          enum:
10         - SFO
11         - LAX
12         - CLE
13    post:
14      /{ID}:
15        get:
```

Below the code editor, there are sections for 'Parameters' (queryParameters, headers) and 'API summary'. The API summary shows a tree structure of resources: 'Resources' → '/flights' (with 'GET' and 'POST' methods) → '/{ID}' (with 'GET' method). A 'Mocking service' button is also present.

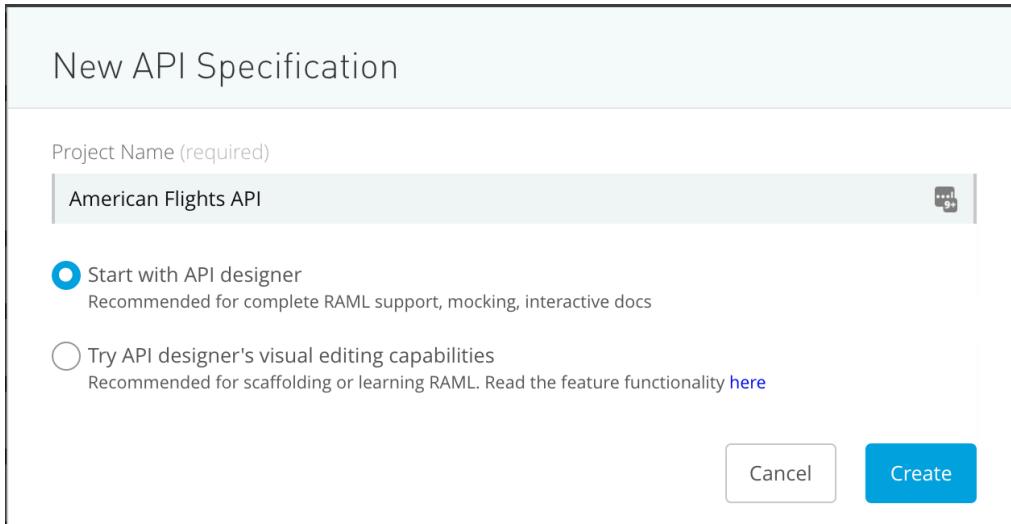
Create a new Design Center project

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

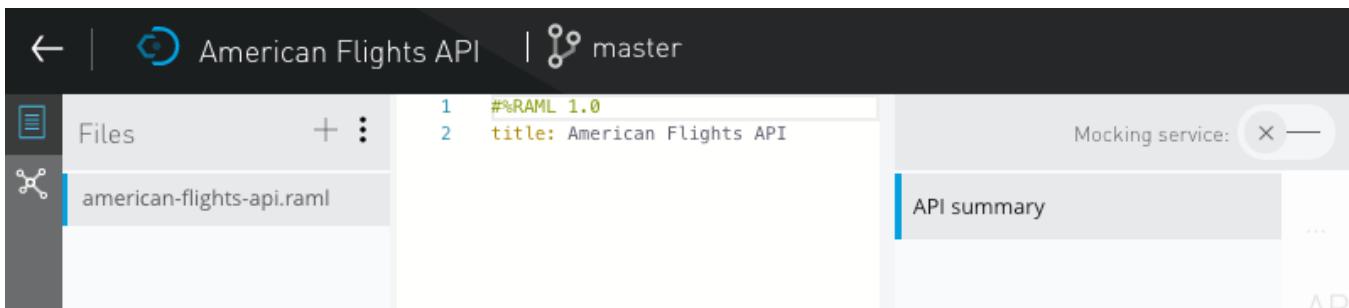


3. In the New API Specification dialog box, set the project name to American Flights API.

4. Select Start with API designer and click Create; API designer should open.



5. In API designer, 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.

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

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 arrow located in the upper-right of the right edge of the web browser window.

The screenshot shows the API designer shelf interface. At the top, there's a header bar with a back arrow, a logo, the title "American Flights API", a gear icon, and the branch name "master". To the right, it says "Saved a minute ago". Below the header, on the left, is a sidebar with "Files" and a plus sign, and a "Mocking service" button with an "X" and a minus sign. The main area shows a RAML file named "american-flights-api.raml". The code editor contains the following RAML:

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

To the right of the code editor is the "API summary" section, which includes a tree view of resources: "Resources" and "flights".

Add RAML methods

9. In the editor, go to a new line of code and 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.

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

The screenshot shows the API designer shelf after pressing Tab. The "Methods" section now lists "get", "put", and "post".

Types and Traits	Docs	Security
is	description	securedBy
type	displayName	

Parameters	Methods
uriParameters	get put post

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.

The screenshot shows the Mockingbird interface with the American Flights API RAML file open. The left sidebar shows 'Files' and a new file icon. The main area displays the RAML code:

```
1  #%RAML 1.0
2  title: American Flights API
3
4  /flights:
5    get:
6    post:
7
```

To the right, a panel titled 'Mocking service:' shows an 'API summary' with sections for 'Resources' and '/flights'. Under '/flights', there are 'GET' and 'POST' buttons.

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 and expand the /{ID} resource; you should see the nested resource with a GET method.

The screenshot shows the RAML Editor interface with the following details:

- Header:** American Flights API | master
- Left Sidebar:** Files (selected), american-flights-api.raml
- Code View (Left):** RAML code for the American Flights API.
- Code View (Right):** Generated Mocking service code.
- Right Panel:** API summary and Resources section, showing endpoints for /flights and /{ID}.

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

Mocking service: `Mocking service...`

API summary

Resources

/flights

GET

POST

/ID

GET

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}:
```

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
```

Try to call an API method using the API console

30. In the API console, click the GET method for the /flights resource.

31. Review the information.

The screenshot shows a 'Mocking service' interface. At the top, there's a back arrow, the text 'Mocking service:', and a close button. Below this, a blue button labeled 'Try it' is positioned next to the endpoint path '/flights : get'. The main area is titled 'Request' and shows a 'GET /flights' call. A sidebar on the left lists 'Parameters' and 'Properties'. Under 'Properties', there's a section for 'Query parameters' with a 'destination' field set to 'string'. It lists possible values: '(enum) SFO, LAX, CLE'. Another blue 'Try it' button is located at the bottom of this section.

32. Click the Try it button; you should get a message that the Request URL is invalid; a URL to call to try the API needs to be added to the RAML definition.

The screenshot shows a 'Mocking service' interface. At the top, there's a back arrow, a shield icon, the text 'Mocking service:', and a close button. Below this, a red 'Request URL' input field is empty. A placeholder text 'Fill the URI parameters before making a requ...' is visible below the input field. A blue 'Send' button is at the bottom. The interface includes tabs for 'Parameters' (which is active) and 'Headers'. A 'Query parameters' section is present with a checkbox for 'Show optional parameters'.

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:

- Turn on the mocking service.
- Use the API console to make calls to a mocked API.

The screenshot shows the Anypoint Platform interface. On the left, the API designer shows a RAML file named 'american-flights-api.raml'. The code defines a base URI and a '/flights' resource with 'get' and 'post' methods. The 'post' method has a query parameter 'destination' with values 'SFO', 'LAX', and 'CLE'. On the right, the API console is open, showing a 'Mocking service' slider turned on. A request is being made to the URL 'https://mocksvc.mulesoft.com'. The 'Parameters' tab is selected, showing 'Query parameters' with 'SFO' selected. The response is a 200 OK status with a duration of 118.61 ms. The response body is a JSON object with a single key 'message' containing the value 'RAML had no response information for application/json'.

Turn on the mocking service

1. Return to API designer.
2. Locate the Mocking Service slider in the menu bar at the top of the API console.
3. Click the right-side of the slider to turn it on.

The screenshot shows the API console. The 'Mocking service' slider is turned on, indicated by a checked checkbox icon. Below it, a 'Request URL' field contains 'https://mocksvc.mulesoft.com'.

4. Look at the baseUri added to the RAML definition in the editor.

```
1  #RAML 1.0
2  baseUri: https://mocksvc.mulesoft.com/mocks/6822ede5-6246-4462-a380-6ae4a9d4e1c2 #
3  title: American Flights API
```

Test the /flights:get resource

5. In API console, click the Send button for the flights GET method; you should get a 200 status code, a content-type of application/json, and a general RAML message placeholder.

The screenshot shows the MuleSoft API console interface. At the top, there's a header with a back arrow, a shield icon, and the text "Mocking service: [checkbox checked]". Below that is a "Request URL" field containing "https://mocksvc.mulesoft.c...". Underneath are tabs for "Parameters" (which is selected) and "Headers". A sub-section titled "Query parameters" has a checkbox labeled "Show optional parameters" which is unchecked. A large blue "Send" button is centered below these sections. At the bottom, the response status is shown as "200 OK" in a green box with a timestamp "510.36 ms". Below the status are several small icons: a magnifying glass, a square, a double-headed arrow, and a list icon. The main content area displays a JSON object: { "message": "RAML had no response information for application/json" }.

6. Select the Show optional parameters checkbox.
7. In the text field for the code query parameter, enter SFO and click Send; you should get the same response.

Test the /flights/{ID} resource

8. Click the back arrow at the top of API console twice to return to the resource list.

The screenshot shows the API console's navigation sidebar. It starts with "API summary" and then "Resources". Under "Resources", there's a collapsed section for "/flights" and an expanded section for "/{ID}". Within the "/{ID}" section, there are two methods: a green "GET" button and a purple "POST" button. To the right of the methods, there are three dots (...), a "GE" label, and a small "GET" button at the bottom.

9. Click the GET method for the("/{ID}) nested resource.

10. Click Try it; you should see a message next to the Send button that the request URL is invalid.

The screenshot shows the 'Mocking service' interface. In the 'Request URL' field, the URL 'https://mocksvc.mulesoft.c' is entered, with the 'c' likely a typo for 'com'. Below the URL, a red message says 'Fill the URI parameters before making a requ...'. Under the 'Parameters' tab, there is a single input field labeled 'ID*' containing the value '10'. At the bottom right, a blue 'Send' button has a red message next to it stating 'Request URL is invalid.'

11. In the ID text box, enter a value of 10.

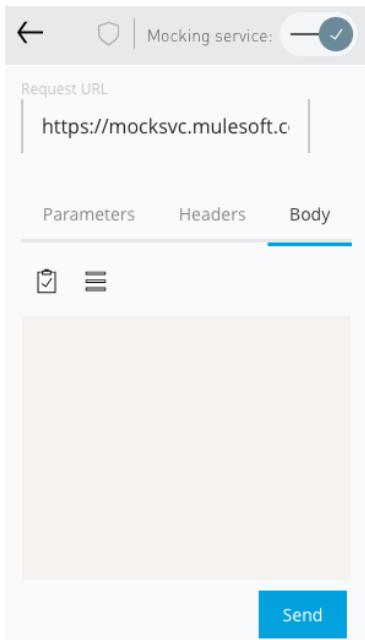
12. Click the Send button.

13. Look at the response; you should get the same default response with a 200 status code, a content-type of application/json, and the general RAML message placeholder.

The screenshot shows the 'Mocking service' interface after the URL has been corrected. The 'Request URL' field now contains 'https://mocksvc.mulesoft.com'. The 'Parameters' tab is selected, showing the 'ID*' field with the value '10'. The blue 'Send' button is now available without any validation messages. Below the interface, the response details are shown: a green '200 OK' status box with a '198.46 ms' latency, followed by a dropdown menu icon. The response body is a JSON object with a single key 'message': { "message": "RAML had no response information for application/json" }.

Test the /flights:post resource

14. Click the back arrow at the top of API console twice to return to the resource list.
15. Click the POST method.
16. Click Try it.
17. Select the Body tab; it should not have any content.



18. Click the Send button.
19. Look at the response; you should get the same generic 200 status code response.

A screenshot of the API response screen. At the top left, it shows "200 OK" and "195.85 ms". To the right is a dropdown arrow. Below that are four icons: a copy icon, a refresh icon, a share icon, and a refresh icon. The main content area displays a JSON object:

```
{  
  "message": "RAML had no  
  response information for  
  application/json"  
}
```

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 in an API.
- Add a data type to be used by resources in an API.
- Specify data types for GET and POST method requests and responses.
- Add example JSON requests and responses.
- Create new files and folders in an API project and import a file.
- Test an API and get example responses.

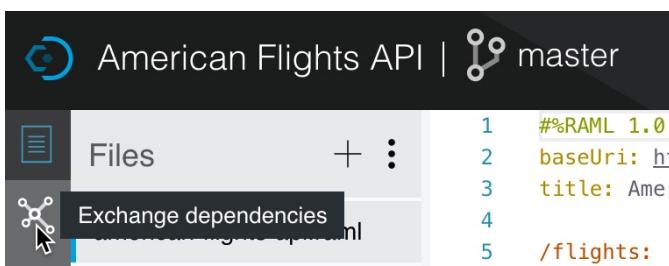
The screenshot shows the MuleSoft Anypoint Studio interface. On the left, the 'Files' sidebar shows a directory structure for an API project. The 'american-flights-api.raml' file is currently selected. The main workspace displays the RAML 1.0 code for the API. The right side of the interface shows a preview of the API's response for a specific endpoint, displaying a 200 OK status with a response time of 125.69 ms. The response body is shown as an array of two flight objects, each with properties like ID, code, price, departure date, origin, destination, and empty seats.

```
%RAML 1.0
baseUri: https://mocksvcs.mulesoft.com/mocks/6822ede5-6246-4462-a380-6ae49d4e1c2 #
title: American Flights API
types:
  AmericanFlight: !include exchange_modules/68ef9520-24e9-4cf2-b2f5-620025690913/training-american-flight-data-type/1.0.1/AmericanFlightDataType.raml
/flights:
  get:
    queryParameters:
      destination:
        required: false
        enum:
          - SFO
          - LAX
          - CLE
    responses:
      200:
        body:
          application/json:
            type: AmericanFlight[]
            example: !include exchange_modules/68ef9520-24e9-4cf2-b2f5-620025690913/training-american-flights-example/1.0.1/AmericanFlightsExample.raml
      post:
        body:
          application/json:
            type: AmericanFlight
            example: !include examples/AmericanFlightNoIDExample.raml
    responses:
      201:
        body:
```

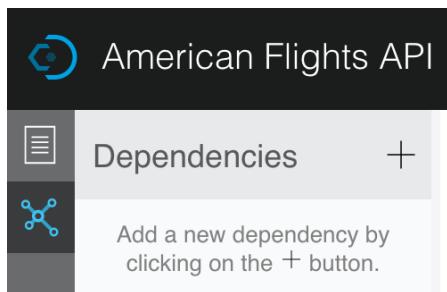
```
200 OK 125.69 ms
[Array[2]
-0: {
  "ID": 1,
  "code": "ER38sd",
  "price": 400,
  "departureDate": "2017/07/26",
  "origin": "CLE",
  "destination": "SFO",
  "emptySeats": 0,
  "plane": {
    "type": "Boeing 737",
    "totalSeats": 150
  }
},
-1: {
  "ID": 2,
  "code": "ER451f",
  "price": 540.99,
  "departureDate": "2017/07/27",
  "origin": "SFO",
  "destination": "ORD",
  "emptySeats": 54,
  "plane": {
    "type": "Boeing 777",
    "totalSeats": 300
  }
}]
```

Add data type and example fragments from Exchange

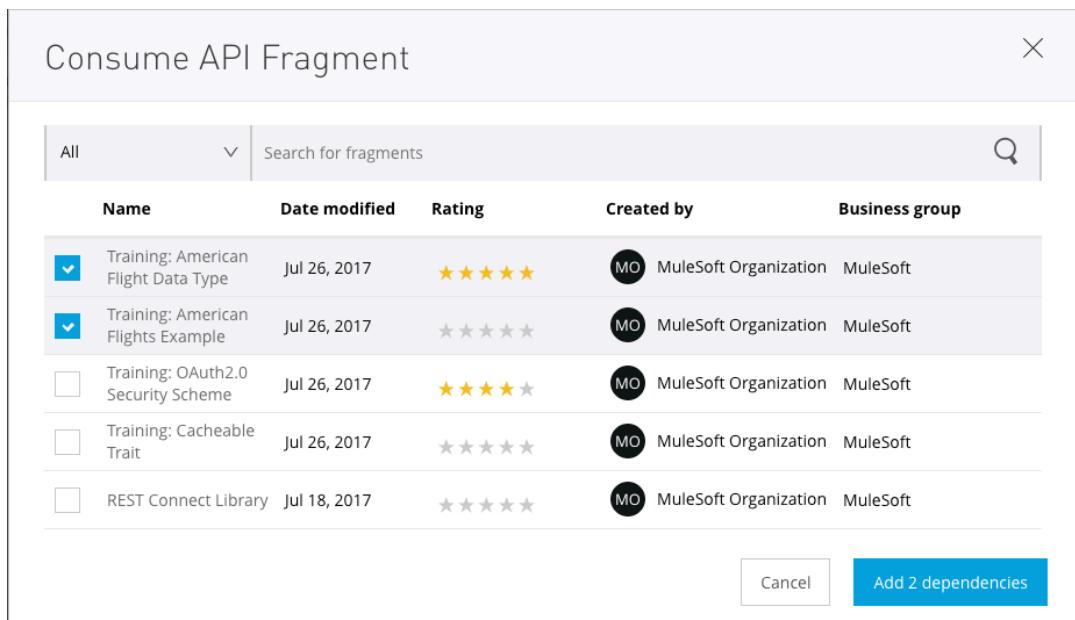
1. Return to API designer.
2. In the file browser, click the Exchange dependencies button.



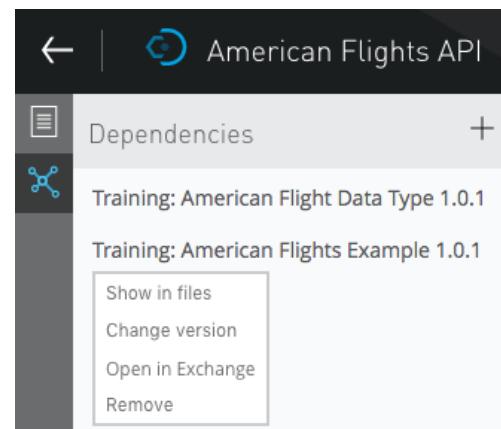
3. Click the Add button.



4. In the Consume API Fragment 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 and review the menu options.



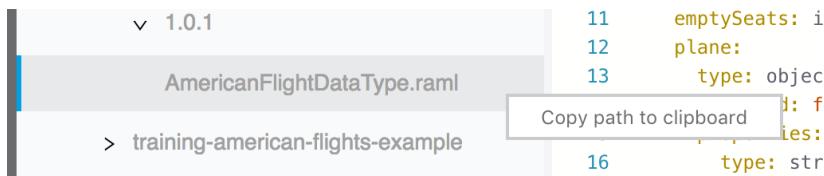
- Click the Files button (above the Exchange dependencies button).
- Expand the exchange_modules section until you see AmericanFlightDataType.raml.
- Click AmericanFlightDataType.raml and review the code.

```

1  %%RAML 1.0 DataType
2
3  type: object
4  properties:
5    ID?: integer
6    code: string
7    price: number
8    departureDate: string
9    origin: string
10   destination: string
11   emptySeats: integer
12   plane:
13     type: object
14     required: false
15     properties:
16       type: string
17       totalSeats: integer
18

```

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



Define an AmericanFlight data type for the API

- Return to american-flights-api.raml.
- Near the top of the code above the /flights resource, add a types element.
- Indent under types and add a type called AmericanFlight.
- 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.

```

1  %%RAML 1.0
2  baseUri: https://mocksvc.mulesoft.com/mocks/3220238a-17c6-4e31-b5fa-413e8129e12b #
3  title: American Flights API
4
5  types:
6    AmericanFlight: !include exchange_modules/68ef9520-24e9-4cf2-b2f5-620025690913/tr
7
8  /flights:
9    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.

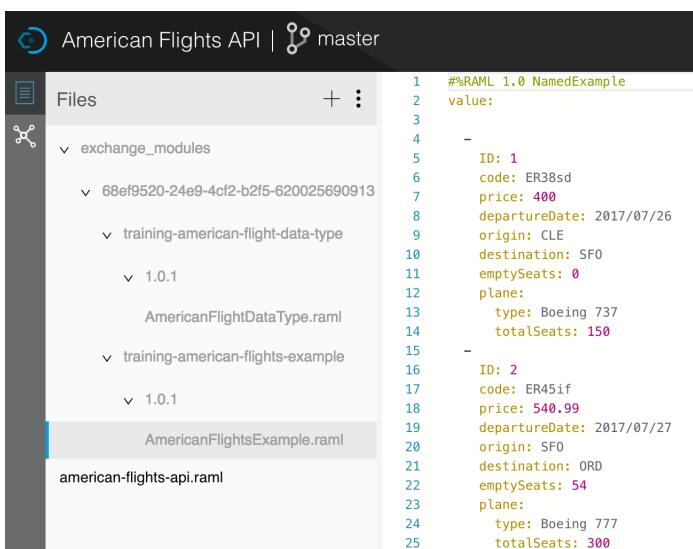
```
8   /flights:  
9     get:  
10    queryParameters:  
11      destination:  
12        required: false  
13        enum:  
14          - SFO  
15          - LAX  
16          - CLE  
17    responses:  
18      200:  
19        body:  
20          application/json:  
21            type: AmericanFlight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

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

```
17  |   responses:  
18  |     200:  
19  |       body:  
20  |         application/json:  
21  |           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.



The screenshot shows the MuleSoft RAML file browser interface. The left sidebar displays a tree view of files under 'exchange_modules'. The 'AmericanFlightsExample.raml' file is currently selected and highlighted in blue. The right pane shows the content of this file, which is a RAML 1.0 document. The code includes a '#%RAML 1.0 NamedExample' block, followed by two examples of AmericanFlight objects (ID 1 and ID 2) with their respective properties like code, price, departure date, origin, destination, empty seats, and plane details.

```
1  #%%RAML 1.0 NamedExample  
2  value:  
3  
4  -  
5  ID: 1  
6  code: ER38sd  
7  price: 400  
8  departureDate: 2017/07/26  
9  origin: CLE  
10 destination: SFO  
11 emptySeats: 0  
12 plane:  
13   type: Boeing 737  
14   totalSeats: 150  
15 -  
16 ID: 2  
17 code: ER45if  
18 price: 540.99  
19 departureDate: 2017/07/27  
20 origin: SFO  
21 destination: ORD  
22 emptySeats: 54  
23 plane:  
24   type: Boeing 777  
25   totalSeats: 300
```

19. In the file browser, click the options menu next to AmericanFlightsExample.raml and select Copy path to clipboard.
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 example.
23. 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.

```

17   responses:
18     200:
19       body:
20         application/json:
21           type: AmericanFlight[]
22           example: !include exchange_modules/86f74f12-decb-452c
23     post:

```

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

24. In API console, click the /flights:get method.
25. Look at the type information in the response information.

The screenshot shows the API console interface for the /flights:get method. The 'Response' tab is selected, showing the 200 status code. Under the 'Type' section, the response type is listed as 'application/json'. Below this, the JSON schema for the response is displayed, showing an array of AmericanFlight objects. Each object has properties like ID, code, price, departureDate, origin, destination, and emptySeats. The 'plane' property is shown as an object with type and totalSeats fields. The 'Examples' tab is also visible but contains no examples.

Parameter	Type	Description
Array of AmericanFlight items		
item. ID	integer	
item. code (required)	string	

26. Click the Examples tab; you should see the example array of AmericanFlight objects.

Type application/json

200

Examples

```
[{"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}}]
```

Parameter	Type	Description
item.ID	integer	
item.code	string	(required)

27. 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

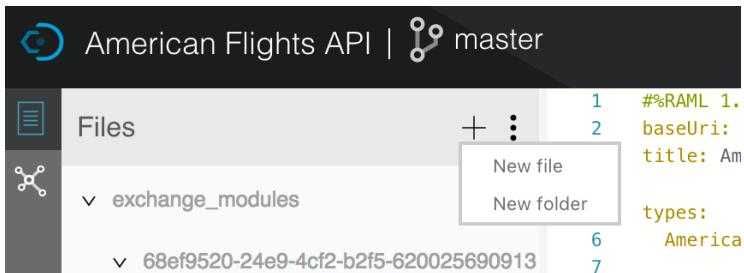
28. In the editor, indent under the `/{ID}` resource get method.

29. In the shelf, click responses > 200 > body > application/json > type > AmericanFlight.

```
/{ID}:
  get:
    responses:
      200:
        body:
          application/json:
            type: AmericanFlight
```

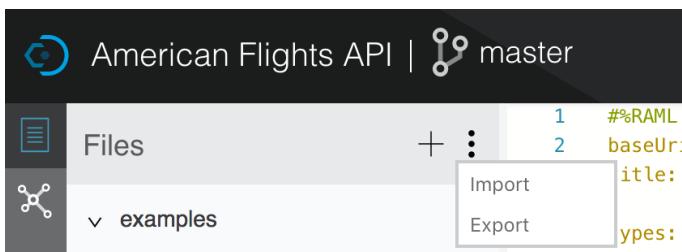
Define an example response for the /flights:get method in a new folder

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



31. In the Add new folder dialog box, set the name to examples and click Create.

32. In the file browser, click the menu button and select Import.



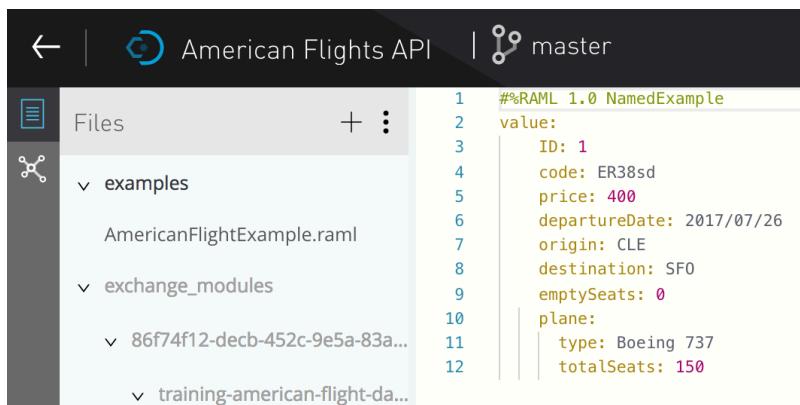
33. In the Import dialog box, leave File or ZIP selected and click the Choose File button.

34. Navigate to your student files and select the AmericanFlightExample.raml file in the examples folder.

35. In the Import dialog box, click Import; you should see the new file in API designer.

36. Review the code.

37. In the file browser, drag AmericanFlightExample.raml into the examples folder.



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

38. Return to american-flights-api.raml.

39. In the editor, go to a new line after the type declaration in {ID}:/get (at the same indentation).

40. In the shelf, click example.

41. Add an include statement and include the example in examples/AmericanFlightExample.raml.

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

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

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

The screenshot shows the API console interface for the /{ID}:get method. The top navigation bar has tabs for 'Type' and 'Examples'. The 'Examples' tab is selected, indicated by a blue underline. Below the tabs, there are two icons: a square with a minus sign and a square with three horizontal lines. The main content area displays a JSON example response:

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

Below the JSON example, there is a table with three columns: 'Parameter', 'Type', and 'Description'. It lists two parameters:

Parameter	Type	Description
ID	integer	
code (required)	string	

43. Select the Examples tab; you should see the example AmericanFlightExample data.

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

The screenshot shows the MuleSoft Anypoint Studio Mocking service interface. At the top, there's a header with a back arrow, a shield icon, and the text "Mocking service: [checkmark]". Below that is a "Request URL" field containing "https://mocksvc.mulesoft.com/mocks/6822ede5-6...". There are two tabs: "Parameters" (which is selected) and "Headers". Under "Parameters", there's a section for "URI parameters" with a single entry: "ID*" set to "10". A "Send" button is located below the parameters. The response section shows a green "200 OK" status with a "205.93 ms" latency. Below the status are several icons: a copy icon, a refresh icon, a dropdown arrow, and a refresh symbol. The response body is a JSON object:

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

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

45. In the editor, indent under the /flights post method.
46. 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

47. Return to AmericanFlightExample.raml and copy all the code.
48. In the file browser, click the add button next to the examples folder and select New file.

49. In the Add new file dialog box, set the following values:

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

50. Click Create.

51. Delete any code in the new file and then paste the code you copied.

52. Delete the line of code containing the ID.

```
1  #%RAML 1.0 NamedExample
2  value:
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
```

53. Return to american-flights-api.raml.

54. In the post method, go to a new line under type and add an example element.

55. Use an include statement to set the example to examples/AmericanFlightNoIDExample.raml.

```
24  post:
25    body:
26      application/json:
27        type: AmericanFlight
28        example: !include examples/AmericanFlightNoIDExample.raml
```

Specify an example response for the /flights:post method

56. Go to a new line of code at the end of the /flights:post method and indent to the same level as body.

57. Add a 201 response of type application/json.

```
24  post:
25    body:
26      application/json:
27        type: AmericanFlight
28        example: !include examples/AmericanFlightNoIDExample.raml
29    responses:
30      201:
31        body:
32          application/json:
33
```

58. In the shelf, click example.

59. Indent under example and add a message property equal to the string: Flight added (but not really).

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

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

60. In API console, return to the /flights:post method.

61. Look at the request information; you should now see information about the body - that it is type AmericanFlight and it has an example.

/flights : post Try it

Request

POST <https://mocksvc.mulesoft.com/mocks/6822ede5-6246-4462-a380-6ae4a9d4e1c2/flights>

Body

Type AmericanFlight

Type Examples

```
{
  "ID": "integer",
  "code": "string",
  "price": "number",
  "departureDate": "string",
  "origin": "string",
  "destination": "string",
  "emptySeats": "integer",
  "plane": {
    "type": "string",
    "totalSeats": "integer"
  }
}
```

Properties

ID integer

code(required) string

62. Click the Try it button and select the Body tab again; you should now see the example request body.

◀ Mocking service:

Request URL: <https://mocksvc.mulesoft.com/mocks/6822ede5-6246-4462-a380-6ae4a9d4e1c2/flights>

Parameters Headers Body

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

Send

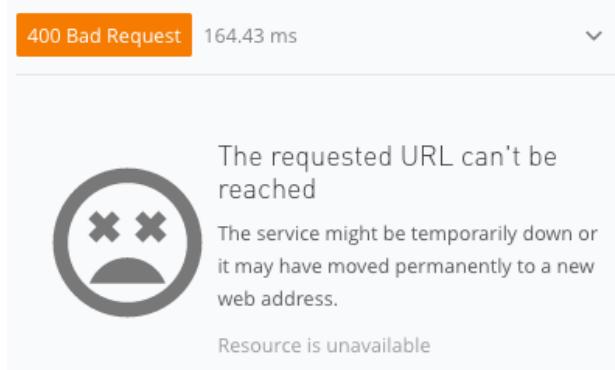
63. Click the Send button; you should now see a response with a 201 status code and the example message.

```
201 Created 494.58 ms ▾  
✖️ ↻ ⌂ ⌂  
{  
  "message": "Flight added (but not really)"  
}
```

64. In the body, remove the emptySeats properties.

Parameters	Headers	Body
		<pre>☑️ ⌂ { "code": "ER38sd", "price": 400, "departureDate": "2017/07/26", "origin": "CLE", "destination": "SFO", "plane": { "type": "Boeing 737", "totalSeats": 150 } }</pre>
		<input type="button" value="Send"/>

65. Click Send again; you should get a 400 Bad Request message.



66. 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.

The screenshot shows the Anypoint Exchange interface with the 'Assets list' sidebar open. The main panel displays the 'American Flights API' asset. Key details shown include:

- API summary:** The American Flights API is a system API for operations on the `american table` in the `training database`.
- Supported operations:** Get all flights, Get a flight with a specific ID, Add a flight, Delete a flight, Update a flight.
- Reviews:** 0 reviews, with a placeholder message: 'Be the first to rate American Flights API'.
- Asset versions for v1:** Version 1.0.1 has one instance (Mocking Service), and Version 1.0.0 has one instance.
- Tags:** None added yet.
- Dependencies:** Training: American Flights Example (1.0.1, RAML Fragment) and Training: American Flight Data Type (1.0.1, RAML Fragment).

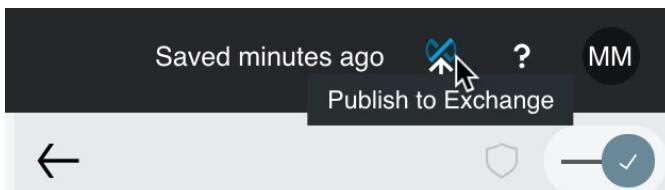
Remove the `baseUri` by turning off the mocking service

1. Return to API designer.
2. Click the slider to turn off the mocking service; the RAML code should no longer have a `baseUri`.

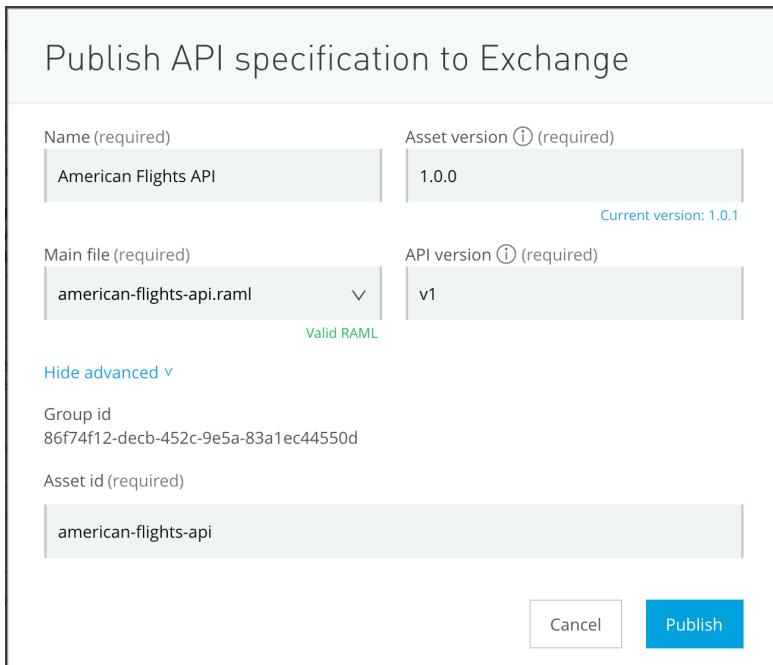
```
1  #%RAML 1.0
2  title: American Flights API
3
```

Publish the API to Anypoint Exchange from API designer

3. Click the Publish to Exchange button.



4. In the Publish API specification to Exchange dialog box, leave the default values:
 - Name: American Flights API
 - Main file: american-flights-api.raml
 - Asset version: 1.0.0
 - API version: v1
5. Click the Show advanced link.
6. Look at the ID values.
7. Click Publish.



8. Click the Publish button.
9. In the Publish API specification to Exchange dialog box, click Done.
10. Look at the RAML file; you should see a version has been added.

```

1  #%RAML 1.0
2  version: v1
3  title: American Flights API
  
```

Locate your API in Anypoint Exchange

11. Return to Design Center.

12. In the main menu, select Exchange; you should see your American Flights API.

The screenshot shows the MuleSoft Exchange interface. The left sidebar has 'All' selected. The main area is titled 'All assets' and shows three items: 'American Flights API Connector' (Connector, 5 stars), 'American Flights API' (REST API, 5 stars), and 'LDAP Connector' (Module, 5 stars). A search bar and a 'New' button are at the top right. The bottom of the screen shows the MuleSoft logo.

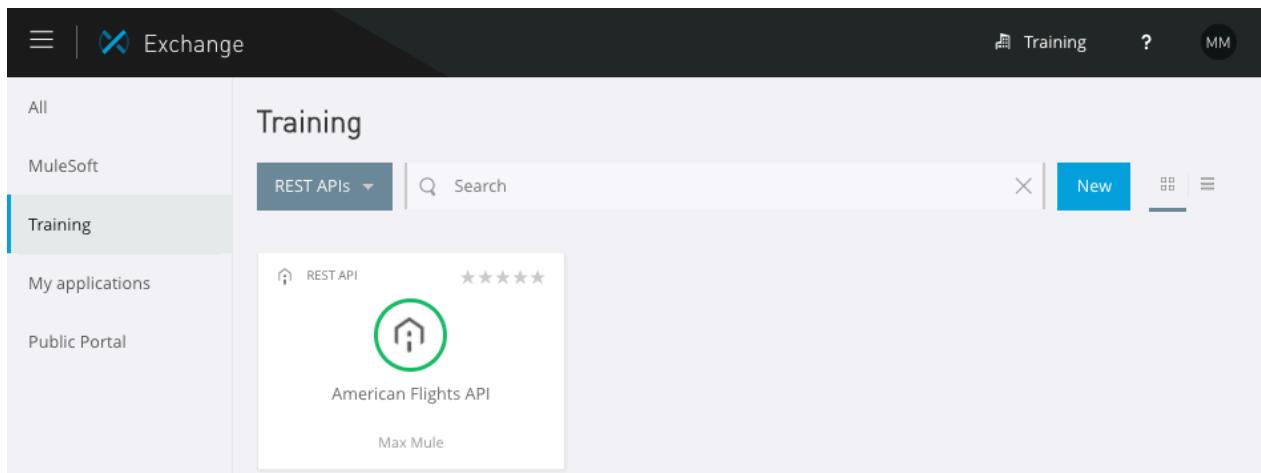
13. In the types drop-down menu, select REST APIs; you should still see your API.

The screenshot shows the MuleSoft Exchange interface with 'REST APIs' selected from the dropdown in the top navigation bar. The main area is titled 'All assets' and shows three items: 'American Flights API' (REST API, 5 stars), 'Optymyze API' (REST API, 5 stars), and 'CardConnect REST API' (REST API, 5 stars). A search bar and a 'New' button are at the top right. The bottom of the screen shows the MuleSoft logo.

14. In the left-side navigation, click MuleSoft; you should not see your American Flights API in the public Exchange (just the Training: American Flights API).

The screenshot shows the MuleSoft Exchange interface with 'MuleSoft' selected from the dropdown in the top navigation bar. The main area is titled 'MuleSoft' and shows three items: 'Optymyze API' (REST API, 5 stars), 'CardConnect REST API' (REST API, 5 stars), and 'Nexmo SMS API' (REST API, 5 stars). A search bar and a 'New' button are at the top right. The bottom of the screen shows the MuleSoft logo.

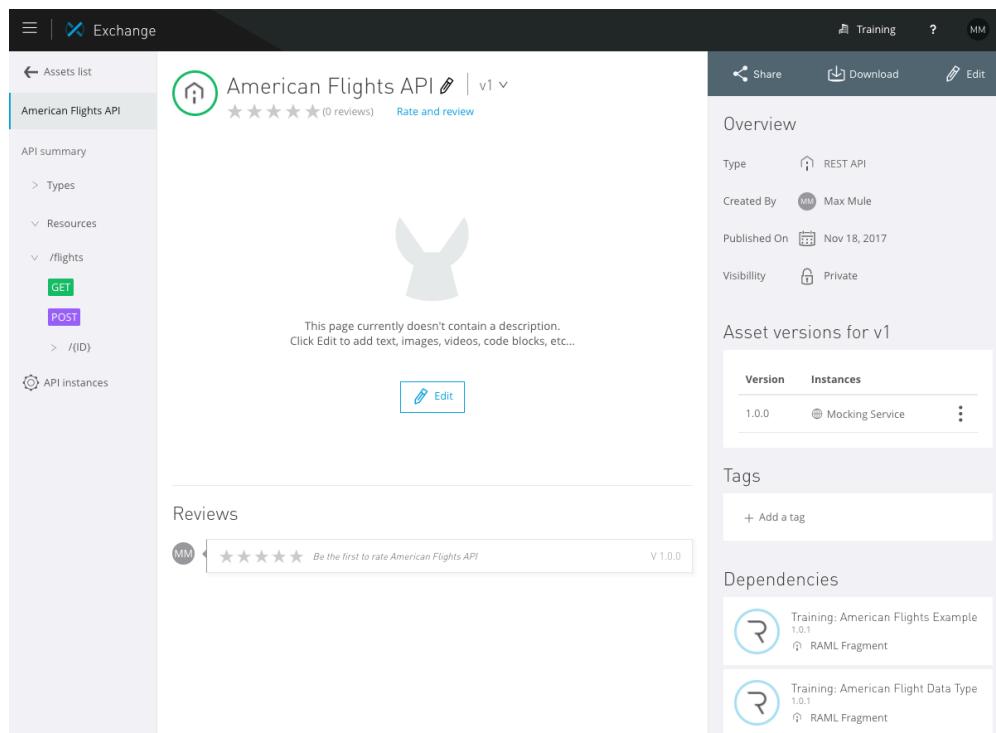
15. In the left-side navigation, click the name of your organization (Training in the screenshots); you should see your American Flights API in your private Exchange.



The screenshot shows the MuleSoft Exchange interface. On the left, there's a sidebar with 'All', 'MuleSoft', and 'Training' (which is selected and highlighted in blue). The main area is titled 'Training' and shows a search bar with 'Search' and a 'New' button. Below the search bar, there's a list of REST APIs. One API is highlighted with a green circle around its icon: 'American Flights API' (type: REST API, created by: Max Mule, published on: Nov 18, 2017, visibility: Private).

Review the auto-generated API portal

16. Click the American Flights API.
17. Review the page; you should see that as the creator of this API, you can edit, review, share, download, and add tags to this version.



The screenshot shows the American Flights API portal page. The left sidebar lists API endpoints: '/flights' (GET, POST), '/(ID)' (PUT, PATCH), and 'API instances'. The main content area shows the API details: 'American Flights API' (version v1), a placeholder image, and a note: 'This page currently doesn't contain a description. Click Edit to add text, Images, videos, code blocks, etc...'. There's an 'Edit' button. Below this is a 'Reviews' section with a placeholder: 'Be the first to rate American Flights API'. The right side has sections for 'Overview' (Type: REST API, Created By: Max Mule, Published On: Nov 18, 2017, Visibility: Private), 'Asset versions for v1' (1.0.0, Mocking Service), 'Tags' (+ Add a tag), and 'Dependencies' (Training: American Flights Example, Training: American Flight Data Type).

18. Locate the API dependencies in the lower-right corner.

19. Locate the API version (v1) next to the name of the API at the top of the page.

The screenshot shows the Exchange interface. At the top, there's a navigation bar with a menu icon and the word "Exchange". Below it is a sidebar with a back arrow labeled "Assets list" and a selected item "American Flights API". The main content area displays the "American Flights API" with a green circular icon containing a house symbol. To the right of the icon, the text "American Flights API" is followed by a pencil icon and "v1". Below this, a star rating of five stars is shown with the text "(0 reviews)" and a "Rate and review" link.

20. Locate the asset versions for v1; you should see one version (1.0.0) of this API specification (v1) has been published and there is one API instance for it and that uses the mocking service.

The screenshot shows a table titled "Asset versions for v1". The table has two columns: "Version" and "Instances". There is one row for version 1.0.0, which lists "Mocking Service" under "Instances". A vertical ellipsis icon is located to the right of the "Instances" column.

Version	Instances
1.0.0	Mocking Service

21. In the left-side navigation, click API instances; you should see information for the API instance generated from the API specification using the mocking service.

The screenshot shows the Exchange interface with the left sidebar expanded. Under "API summary", the "Types" section is expanded, showing "Resources" and "/flights". Under "/flights", "GET" and "POST" methods are listed, with "/{ID}" below them. The "API instances" section is also expanded, showing a table of API instances. The table has columns: "Instances", "Environment", "URL", and "Visibility". There is one entry for "Mocking Service" with the URL "https://mocksvc-proxy.anypoint.mulesoft.com/exchange/800b1585-b6be-4c62-82de-02d8c2adf413/american-flights-api/1.0.0" and "Public" under "Visibility". A blue "+ Add new instance" button is visible at the bottom of the table.

Instances	Environment	URL	Visibility
Mocking Service		https://mocksvc-proxy.anypoint.mulesoft.com/exchange/800b1585-b6be-4c62-82de-02d8c2adf413/american-flights-api/1.0.0	Public

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

23. Click AmericanFlight and review the information.

The screenshot shows the MuleSoft Exchange interface. On the left, there's a navigation sidebar with options like 'Assets list', 'American Flights API', 'API summary', 'Types' (selected), 'AmericanFlight', 'Resources', '/flights' (selected), and 'API instances'. Under '/flights', there are 'GET' and 'POST' buttons. The main content area displays the 'American Flights API' page with a green house icon, a rating of 5 stars (0 reviews), and a title 'Type AmericanFlight'. It shows a JSON schema for the 'AmericanFlight' type and a table of parameters:

Parameter	Type	Description
ID	integer	
code (required)	string	

Test the API in its API portal in Exchange

24. In the left-side navigation, click the /flights GET method; you should now see an API Reference section to test the method on the right side of the page.
25. In the API Reference, select to show optional query parameters.
26. Click the destination drop-down and select a value.

The screenshot shows the MuleSoft Exchange interface with the 'GET /flights' method selected. The left sidebar remains the same. The main content area shows the API Reference for '/flights : get'. It includes a 'Request' section with 'GET /flights' and a 'Parameters' section with a table:

Parameter	Type	Description
Query parameters		
destination	string (enum)	Possible values: SFO, LAX, CLE

To the right, there's a panel for testing the API. It shows a 'GET' button, a dropdown for 'Mocking Service' (set to 'Mocking Service'), a URL field with 'https://mocksvc-proxy.anypoint.mulesoft.com/exc', and tabs for 'Parameters' (selected) and 'Headers'. Below these are sections for 'Query parameters' (with a dropdown containing 'LAX') and 'Show optional parameters' (checkbox checked). A 'Send' button is at the bottom right.

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

The screenshot shows the MuleSoft Anypoint Exchange API designer interface. At the top, a green button labeled "GET" is visible. Below it, the URL "https://mocksvc-proxy.anypoint.mulesoft.com/exc" is entered. Under the URL, there are tabs for "Parameters" and "Headers", with "Parameters" being the active tab. A dropdown menu under "Query parameters" shows "LAX". To the right of the dropdown is a blue "Send" button. Below the "Send" button, the response status is shown as "200 OK" and "582.08 ms". There is also a "Details" link. The main content area displays a JSON response:

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

Add information about the API

28. In the left-side navigation, click the name of the API: American Flights API.
29. Click one of the Edit buttons for the API.
30. Return to the course snippets.txt file and copy the text for the American Flights API description text.
31. Return to the editor in Anypoint Exchange and paste the content.

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

The American Flights API is a system API for operations on the **american** table** in the training database.
Supported operations
Get all flights
Get a flight with a specific ID
Add a flight

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

The American Flights API is a system API for operations on the **american** table** in the training database.
Supported operations

34. Select the words Supported operations and click the heading button four times (the H).

The American Flights API is a system API for operations on the **american** table** in the training database.
Supported operations

35. Select Get all flights and click the bulleted list button.

36. Repeat for the other operations.

The American Flights API is a system API for operations on the **american** table** in the training database.
Supported operations

- Get all flights
- Get a flight with a specific ID
- Add a flight

37. Click the Visual tab in the editor toolbar and view the rendered markdown.

The screenshot shows the MuleSoft Exchange interface for editing APIs. On the left, there's a sidebar with options like 'Assets list', 'American Flights API' (selected), '+ Add new page', 'API summary', '+ Add terms and conditions', and 'API instances'. The main area displays the 'American Flights API' page, version v1. At the top right of this area is a toolbar with various icons (B, I, etc.) and tabs for 'Markdown' and 'Visual' (which is currently selected). Below the toolbar, the content is rendered in a visual format: 'The American Flights API is a system API for operations on the **american** table in the *training database*. Supported operations: • Get all flights • Get a flight with a specific ID • Add a flight'. At the bottom right of the main area are two buttons: 'Discard changes' and 'Save as draft'.

38. Click the Save as draft button; you should now see buttons to exit the draft or publish it.

The screenshot shows the MuleSoft Exchange interface for the 'American Flights API'. The left sidebar remains the same. The main area now shows the published state. A message at the top says 'This is a draft that has not been published yet.' with a pencil icon. To the right of this message are 'Exit Draft' and 'Publish' buttons. The 'Publish' button is highlighted in blue. To the right of the main content area is a sidebar titled 'Overview' with details: 'Type REST API', 'Created By Max Mule', and a profile picture for Max Mule. The main content area displays the API's description and supported operations.

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

The screenshot shows the Mule Exchange API portal. On the left, there's a sidebar with navigation links like 'Assets list', 'American Flights API', 'API summary', 'Types', 'Resources', '/flights', and a 'Supported operations' section listing 'Get all flights', 'Get a flight with a specific ID', and 'Add a flight'. The main content area displays the 'American Flights API' page, which includes a logo, a rating of 0 reviews, and a 'Rate and review' link. It also contains a brief description: 'The American Flights API is a system API for operations on the **american** table in the *training database*'. Below this is an 'Overview' section with details: Type (REST API), Created By (Max Mule), and Published On (Nov 18, 2017).

Modify the API and publish the new version to Exchange

40. Return to your American Flights API in API designer.
41. Return to the course snippets.txt file and copy the American Flights API - /{ID} DELETE and PUT methods.
42. Return to american-flights-api.raml and paste the code after the {ID}/get method.
43. Fix the indentation.
44. Review the code.

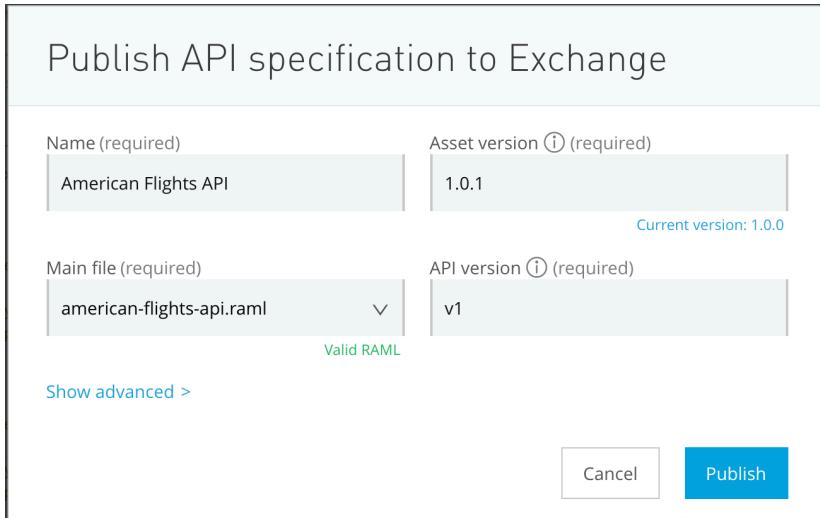
```

/{ID}:
  get:
    responses:
      200:
        body:
          application/json:
            type: AmericanFlight
            example: !include examples/AmericanFlightExample.raml
  delete:
    responses:
      200:
        body:
          application/json:
            example:
              message: Flight deleted (but not really)
  put:
    body:
      application/json:
        type: AmericanFlight
        example: !include examples/AmericanFlightNoIDEExample.raml
    responses:
      200:
        body:
          application/json:
            example:
              message: Flight updated (but not really)

```

45. Click the Publish to Exchange button.

46. In the Publish API specification to Exchange dialog box, look at the asset version.



47. Click Publish.

48. In the Publish API specification to Exchange dialog box, click Exchange; Exchange should open in a new browser tab.

49. Locate the asset versions listed for the API; you should see both asset versions of the API listed with an associated API instance using the mocking service for the latest version.

Asset versions for v1	
Version	Instances
1.0.1	Mocking Service
1.0.0	

50. Click the Edit button.

51. Add two new operations that delete a flight and update a flight.

A screenshot of the API documentation editor interface. At the top, there are toolbar icons for bold, italic, code snippets, and other document controls. To the right of the toolbar are buttons for 'Markdown' and 'Visual' modes, with 'Markdown' currently selected. The main content area contains the following text:

```
The American Flights API is a system API for operations on the **american table** in the _training database_.  
  
#### Supported operations  
  
- Get all flights  
- Get a flight with a specific ID  
- Add a flight  
- Delete a flight  
- Update a flight
```

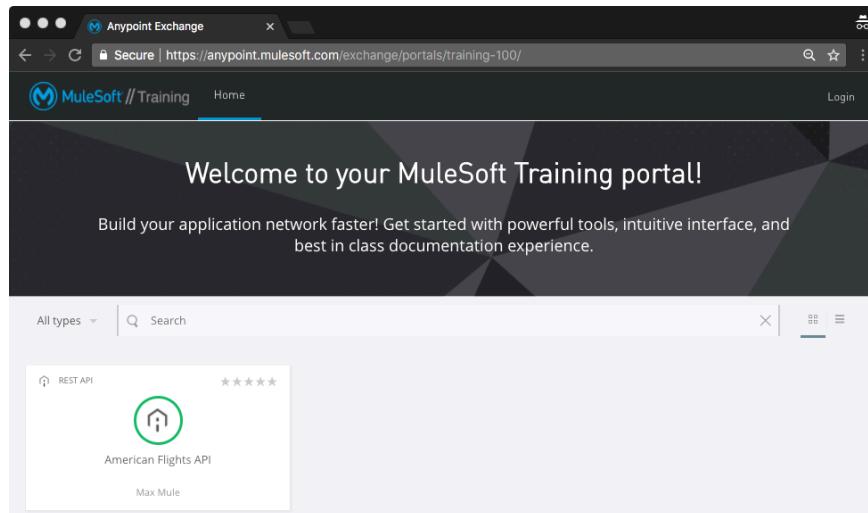
52. Click Save as draft and then Publish.

The screenshot shows the MuleSoft Anypoint Platform interface. On the left, the sidebar lists 'Assets list' and the 'American Flights API'. Under the API summary, there are sections for 'Types', 'Resources', and '/flights'. The '/flights' section shows two buttons: 'GET' and 'POST'. Below this is an 'API instances' section. The main content area displays the 'American Flights API' details. It includes a logo, a rating of 0 reviews, and a 'Rate and review' button. A description states: 'The American Flights API is a system API for operations on the **american table** in the **training database**'. A 'Supported operations' list includes: Get all flights, Get a flight with a specific ID, Add a flight, Delete a flight, and Update a flight. To the right, there's a navigation bar with 'Training', '?', and 'MM'. Below it are 'Share', 'Download', and 'Edit' buttons. The 'Overview' section shows the API is a 'REST API' created by 'Max Mule' on Nov 18, 2017, with 'Private' visibility. It also lists 'Asset versions for v1' (1.0.1 and 1.0.0), 'Tags' (with a '+ Add a tag' button), and 'Dependencies' (Training: American Flights Example and Training: American Flight Data Type, both version 1.0.1 and RAML Fragment).

Walkthrough 3-5: Share an API

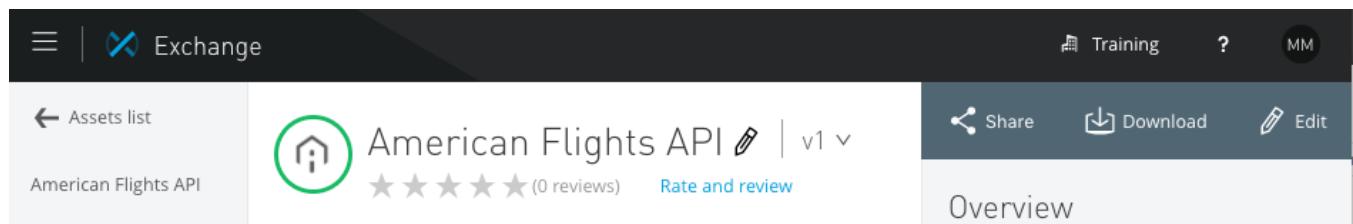
In this walkthrough, you share an API with both internal and external developers to locate, learn about, and try out the API. You will:

- 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.

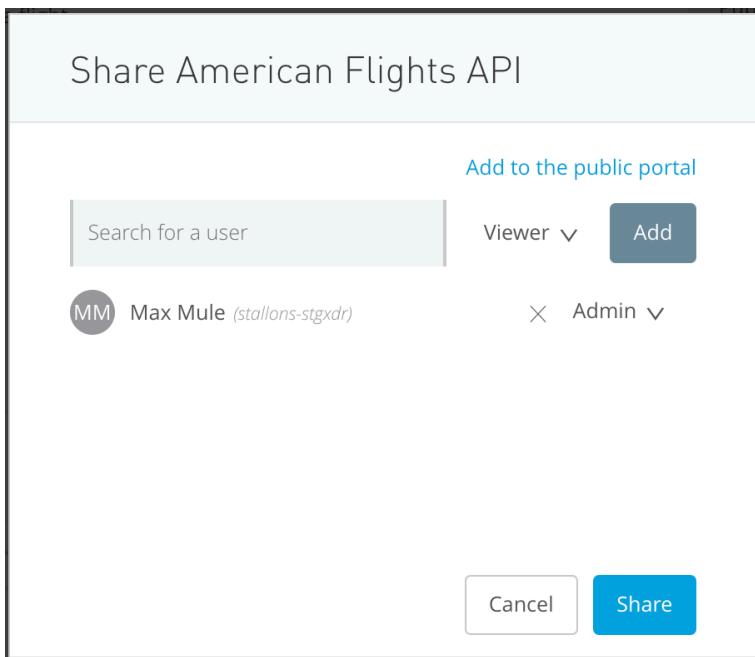


Share the API in the private Exchange with others

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



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



4. Open the Viewer drop-down menu located next to the user search field; you should see that you can add additional users to be of type Viewer, Contributor, or Admin.

Note: In the future, you will also be able to share an asset with an entire business group.

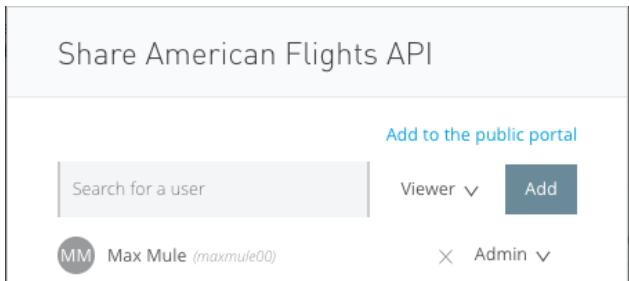
5. Click Cancel.
6. In the left-side navigation, click Assets list.
7. In the left-side navigation, click the name of your organization.
8. Click your American Flights API.

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

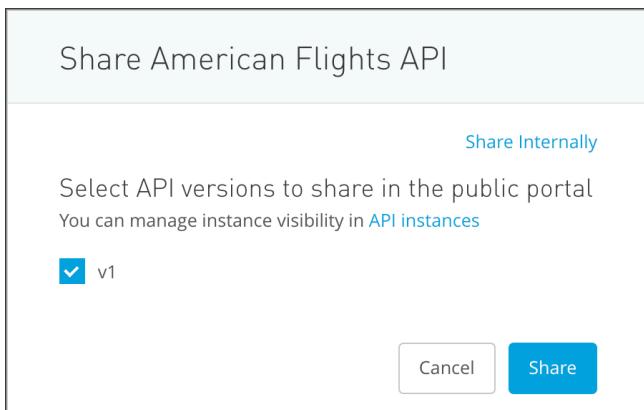
A screenshot of the MuleSoft Exchange interface. The top navigation bar includes 'Exchange', 'Training', a help icon, and a user profile icon. On the left is a sidebar with 'All', 'MuleSoft', and 'Training' sections. Under 'Training', 'My applications' and 'Public Portal' are listed. The main area is titled 'Training' and shows two items: 'American Flights API Connector' (Connector, 5 stars, Max Mule) and 'American Flights API' (REST API, 5 stars, Max Mule). A 'New' button is at the top right of the main content area.

Create a public API portal

9. Click the American Flights API.
10. Click the Share button for the API again.
11. In the Share American Flights API dialog box, click Add to the public portal.



12. In the new dialog box, select to share v1 of the API.
13. Click Share.



Explore the API in the public portal

14. In the left-side navigation, click Assets list.
15. In the left-side navigation, click Public Portal.

A screenshot of the MuleSoft Exchange application. The top navigation bar includes "Exchange", "Training", a help icon, and a user profile icon. On the left, a sidebar shows navigation links: "All", "MuleSoft", "Training" (which is selected and highlighted in grey), "My applications", and "Public Portal". The main content area is titled "Training" and displays two items: "American Flights API Connector" (Connector type, 5 stars, created by Max Mule) and "American Flights API" (REST API type, 5 stars, created by Max Mule). A "New" button is located in the top right corner of the main content area.

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

17. Look at the URL for the public portal.

The screenshot shows a web browser with three tabs open. The active tab is titled 'Secure | https://anypoint.mulesoft.com/exchange/portals/training-100/'. The page content is a developer portal interface. At the top, there's a navigation bar with 'Home' and 'My applications' links, a 'Customize' button, and a 'MM' icon. The main heading is 'Welcome to your developer portal!'. Below it, a sub-headline reads 'Build your application network faster! Get started with powerful tools, intuitive interface, and best in class documentation experience.' A search bar at the bottom left includes dropdowns for 'All types' and 'Search'. In the center, there's a card for the 'American Flights API', which is categorized under 'REST API'. The card features a green circular icon with a white house-like symbol, a five-star rating, and the text 'American Flights API' and 'Max Mule'.

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

The screenshot shows the 'American Flights API' page within the developer portal. On the left, a sidebar navigation includes 'Assets list', 'American Flights API' (which is selected and highlighted in blue), 'API summary', 'Types' (with a 'GET' method listed), 'Resources' (with a 'flights' item and 'POST' method listed), and 'API instances'. The main content area displays the API details for 'American Flights API | v1'. It includes a green circular icon with a white house-like symbol, a five-star rating with '(0 reviews)', and a 'Rate and review' link. Below this, a description states: 'The American Flights API is a system API for operations on the **american** table in the *training database*'. Under 'Supported operations', there's a bulleted list: 'Get all flights', 'Get a flight with a specific ID', 'Add a flight', 'Delete a flight', and 'Update a flight'. To the right, there's a 'API Spec' button with a download icon.

19. In the left-side navigation, click the POST method for the flights resource.

20. Review the body and response information.

21. In the API Reference section, click Send; you should get a 201 response with the example data returned from the mocking service.

The screenshot shows the MuleSoft Anypoint Platform interface. On the left, there's a sidebar with a navigation tree for the "American Flights API". The tree includes sections for "API summary", "Types", "Resources", and "Instances". Under "Instances", there are four items: "GET", "POST", "DELETE", and "PUT", with "POST" being highlighted. The main content area displays the "American Flights API" details, version v1, with a 5-star rating of 0 reviews. It shows the endpoint "/flights : post" with a "Request" section. The "Request" section shows a "POST /flights" call. Below this is a "Body" section where the "Type AmericanFlight" is selected. A JSON example is provided:

```
{
  "ID": "integer",
  "code": "string",
  "price": "number",
  "departureDate": "string",
  "origin": "string",
  "destination": "string",
  "emptySeats": "integer",
  "plane": {
    "type": "string",
    "totalSeats": "integer"
  }
}
```

To the right, a "POST" button is at the top of a panel titled "Mocking Service" with the URL "https://mocksvc-proxy.anypoint.mulesoft.com/". Below the URL are tabs for "Parameters", "Headers", and "Body", with "Body" being active. A note says "This endpoint doesn't require to declare query or URI parameters." At the bottom of the panel is a "Send" button. The response is shown as a 201 Created status with a timestamp of 271.27 ms and a "Details" link. The response body is a JSON object with a message: "Flight added (but not really)".

Customize the public portal

22. In the left-side navigation, click Assets list.

23. Click the Customize button.

The screenshot shows the MuleSoft Anypoint Platform developer portal. At the top, there are "Home" and "My applications" links, and a "Customize" button with a pencil icon. The main content area has a large "Welcome to your developer portal!" heading. Below it is a descriptive text: "Build your application network faster! Get started with powerful tools, intuitive interface, and best in class documentation experience." The background features a blue gradient.

24. Change the text to Welcome to your MuleSoft Training portal!

The screenshot shows the MuleSoft Anypoint Platform interface. On the left, there's a preview of a web page with a header "Welcome to your MuleSoft Training portal!" and a sub-section "Build your application network faster! Get started with powerful tools, intuitive interface, and best in class documentation experience." Below this is a search bar and a list of applications, with "American Flights API" highlighted. On the right, there's a sidebar for configuration:

- Top bar**: Includes a "Logo" field with a "Choose file" button.
- Favicon**: Includes a "Choose file" button.
- Background color**: A color swatch set to #262728.
- Text color**: A color swatch set to #FFFFFF.
- Text color (active)**: A color swatch set to #00A2DF.
- Welcome section**:
 - Hero image**: A field containing "image-default.png" with a "Choose file" button.
 - Text color**: A color swatch set to #FFFFFF.
- Custom pages**: A link to "+ Add new page".

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

30. Review the new logo and banner in the preview.

The screenshot shows the MuleSoft Training portal builder interface. On the left, there's a preview of the portal with a dark header containing the MuleSoft logo and the text "Welcome to your MuleSoft Training portal!". Below the header is a banner with the text "Build your application network faster! Get started with powerful tools, intuitive interface, and best in class documentation experience.". A search bar and a sidebar with API cards (REST API, American Flights API, Max Mule) are also visible. On the right, there are several configuration sections: "Top bar" (Logo set to "MuleSoft_training_logo.png", "Choose file" button), "Favicon" (empty field, "Choose file" button), "Background color" (#262728), "Text color" (#FFFFFF), "Text color (active)" (#00A2DF), "Welcome section" (Hero image set to "banner.jpg", "Choose file" button), "Text color" (#FFFFFF), and "Custom pages" (+ Add new page).

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

Explore the public portal as an external developer

34. In the browser, copy the URL for the public portal.
35. Open a new private or incognito window in your browser.

36. Navigate to the portal URL you copied; you should see the public portal (without the customize button).

The screenshot shows the MuleSoft Training portal homepage. At the top, there's a banner with the text "Welcome to your MuleSoft Training portal!" and a subtext: "Build your application network faster! Get started with powerful tools, intuitive interface, and best in class documentation experience." Below the banner is a search bar with the placeholder "All types" and a "Search" button. A sidebar on the left lists "REST API" and "American Flights API" with a "Max Mule" badge. The main content area displays the "American Flights API" card.

37. Click the American Flights API.

38. Explore the API portal.

39. 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.

The screenshot shows the "American Flights API | v1" details page. On the left, the sidebar shows "Assets list" and "API instances". Under "API instances", there's a "GET /flights" entry. The main content area shows a "GET" method card for "Mocking Service" at the URL "https://mocksvc-proxy.anypoint.mulesoft.com/". It has tabs for "Parameters" and "Headers", and a "Send" button. Below the card, a "200 OK" response is shown with a timestamp of "260.37 ms" and a "Details" link. The response body is an array of flight objects:

```
[Array[2]
  ↗0: {
    "ID": 1,
    "code": "ER38sd",
    "price": 400,
    "departureDate": "2017/07/26",
    "origin": "CLE"
  }
]
```