



Module 3: Designing APIs



- We discussed in the last modules about the benefits of designing an API first before actually building it
- This is often referred to as **spec driven development**
 - A development process where your application is built in two distinct phases
 - The creation of a spec (the design phase)
 - Development of code to match the spec (the development phase)
- In this module, we'll
 - Create this API specification using a standardized API description language (RAML)
 - Then learn to test it with users without writing any code

Goal

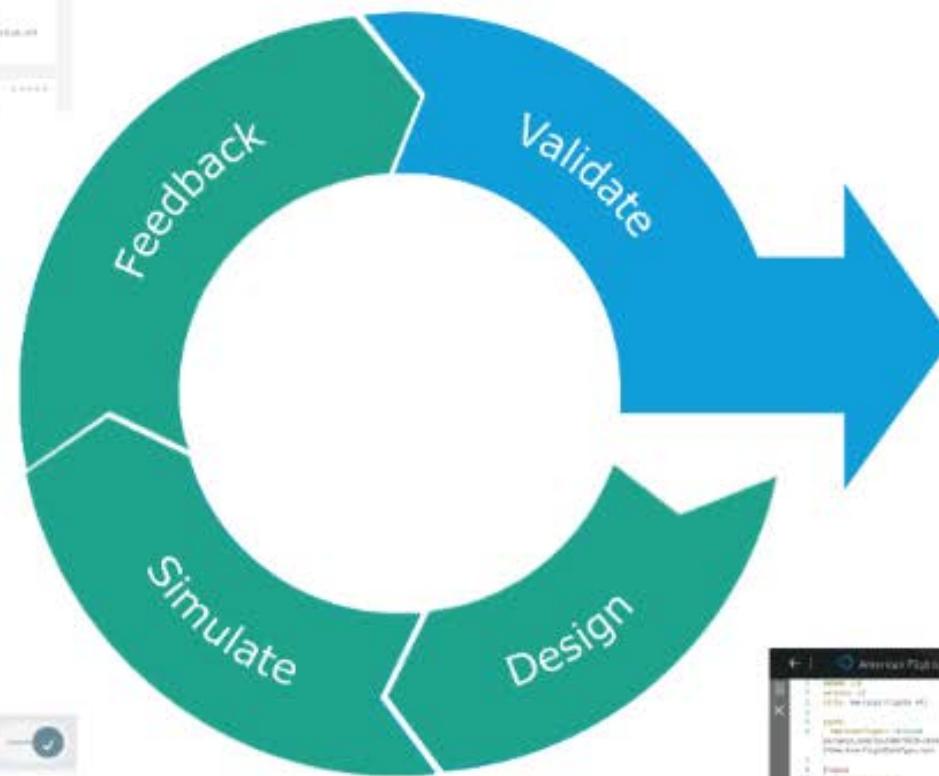


A screenshot of the MuleSoft Exchange API portal. It shows a search bar at the top with the text "American Flights API". Below the search bar, there's a list of APIs under "Assets" and a "Recently used" section. A detailed view of the "American Flights API" is shown in the center, including its description, endpoints like "/flights/:id", and parameters.

API portal

A screenshot of the MuleSoft API console. It shows an "API summary" section with "Resources" and "flights" listed. Under "flights", there are four methods: "GET", "POST", "PUT", and "DELETE". Below this, a "Mocking Service" section displays a URL "https://mocksvc.mulesoft.com/mocks/72406d47-102a-486a-ae97-6" and a "Parameters" section with a "Query parameters" input field.

API console



API Spec
(RAML)

A screenshot of the MuleSoft API designer. It shows the "American Flights API" definition with various sections like "Description", "Operations", "Responses", and "Security". The "Operations" section lists "flights" and "flights/:id" with their respective descriptions and responses.

API designer

- 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

Reviewing the options for defining APIs



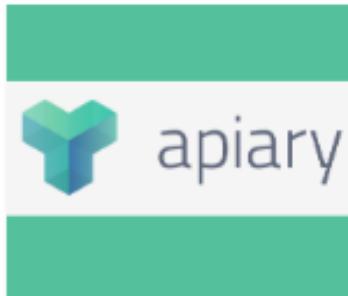
Approaches to API design



Hand coding

```
auditing minFreeAllocPart: Error in Allocating memory for the audit log file
ting present
h: "/pci@0/d/pci-ata1/ata-40/0/0/0/sector-0"
<IOPathMatch></key>(string "0")</dict>
<string></dict>
ioreWire QUID = 0x58e4ff:a
sent:8
t device = IOService::GetSystemPCIAddressForDevice(0)
2PCIIBridge/pci-ata1/CH4400/ata-40/0/0/0/sector-0
geDriver/IOATABlockStorageDevice/10000000000000000000000000000000
titonS
    - HES_Untitled_3018
        - 14. aior 3
            - via virtio-ahci
```

API Blueprint



OpenAPI Spec



RAML



Introducing RAML



- **A simple, structured, and succinct way of describing RESTful APIs**
- A non-proprietary, vendor-neutral open spec
- Developed to help out the current API ecosystem
 - Encourages reuse, enables discovery and pattern-sharing, and aims for merit-based emergence of best practices
- RAML files can be used to auto-generate documentation, mocked endpoints, interfaces for API implementations, and more!



- RAML is based on broadly-used standards such as YAML and JSON
- Uses a human-readable data serialization format where **data structure hierarchy is specified by indentation**
 - Not additional markup characters

```
1  #%RAML 1.0
2  version: v1
3  title: American Flights API
4
5  /flights:
6    get:
7    post:
8
9    /{ID}:
10   get:
11   delete:
12   put:
13   responses:
14     200:
15       body:
16         application/json:
```

Notice the indentation used to specify to what each line applies

- Resources are the objects identified by the web service URL that you want to act upon using the HTTP method used for the request
- All resources begin with a slash
- Any methods and parameters nested under a resource belong to and act upon that resource
- Nested resources are used for a subset of a resource to narrow it
 - URI parameter are enclosed in {}

```
1  #%%RAML 1.0
2  version: v1
3  title: American Flights API
4
5  /flights:
6    get:
7    post:
8
9    /{ID}:
10   get:
11   delete:
12   put:
13   responses:
14     200:
15       body:
16         application/json:
```

Using API designer to define APIs with RAML



American Flights API | master

← | ↻ | ⚙ | Saved a minute ago | X | ? | MM

Files + : Mocking service: X —

examples AmericanFlightExample.raml
AmericanFlightNoIDExample.raml

exchange_modules 68ef9520-24e9-4cf2-b2f5-620025690913/training-american-flight-data-type/1.0.1/AmericanFlightDataType.raml

File browser Editor Shelf

```
1 #%RAML 1.0
2 version: v1
3 title: American Flights API
4
5 types:
6   AmericanFlight: !include
7     exchange_modules/68ef9520-24e9-4cf2-b2f5-620025690913/training-american-
8     flight-data-type/1.0.1/AmericanFlightDataType.raml
9
10 /flights:
11   get:
12     queryParameters:
13       destination:
14         required: false
15         enum:
16           - SFO
17           - LAX
18           - CLE
19     responses:
20       200:
21         body:
22           application/json:
23             type: AmericanFlight[]
```

API summary

> Types

> Resources

> /flights

GET

POST

> /(ID)

GET

DELETE

PUT

API console

12

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



- 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 API designer interface for the American Flights API. On the left, the file structure shows 'american-flights-api.raml'. The main area displays the RAML code:

```
%RAML 1.0
title: American Flights API
...
/flights:
  get:
    queryParameters:
      destination:
        required: false
        enum:
          - SFO
          - LAX
          - CLE
  post:
    /{ID}:
      get:
```

The right side shows the API summary with the following details:

- Mocking service: (button)
- API summary
- Resources
 - /flights
 - GET
 - POST
 - /ID
 - GET

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 API designer interface for the "American Flights API". The left pane displays the RAML code for the API, and the right pane shows the API summary and its resources.

API Summary:

- Resources:** /flights
- /flights:**
 - GET**
 - POST**
- /({ID}):**
 - get:** GET

RAML Code:

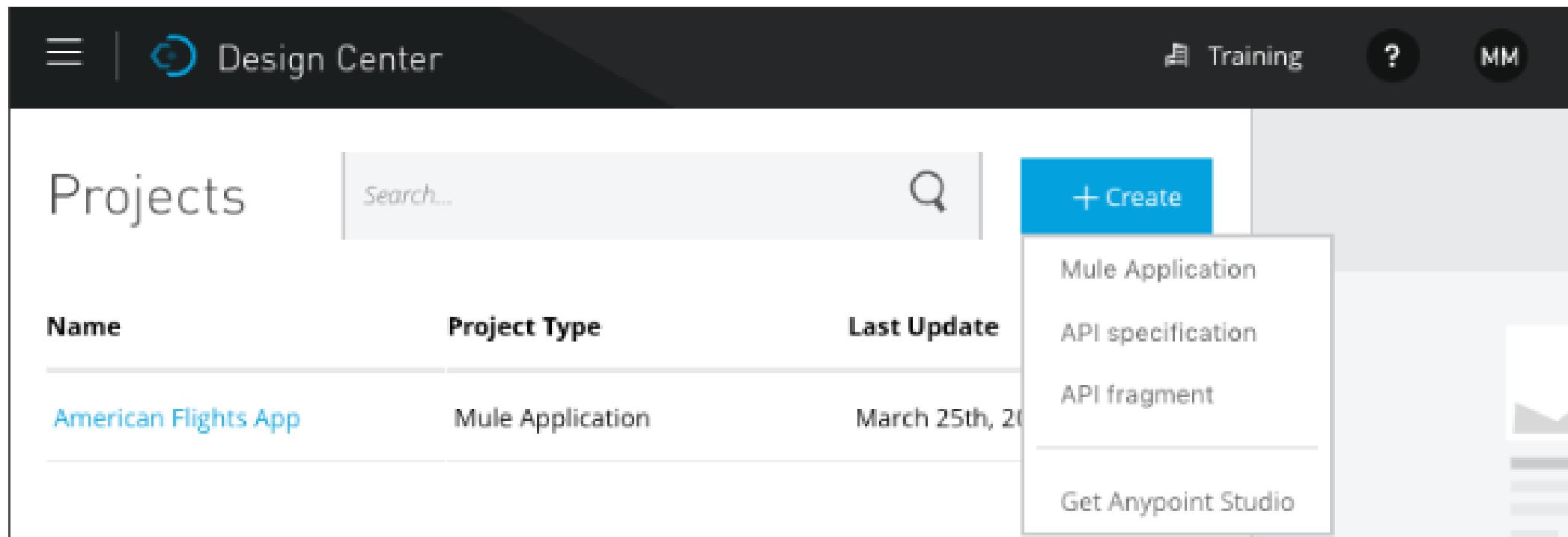
```
1  #%RAML 1.0
2  title: American Flights API
3
4  /flights:
5    get:
6      queryParameters:
7        destination:
8          required: false
9          enum:
10         - SFO
11         - LAX
12         - CLE
13    post:
14    /{ID}:
15      get:
```

Bottom Navigation:

- Docs
- Others
- displayName
- facets
- example
- type

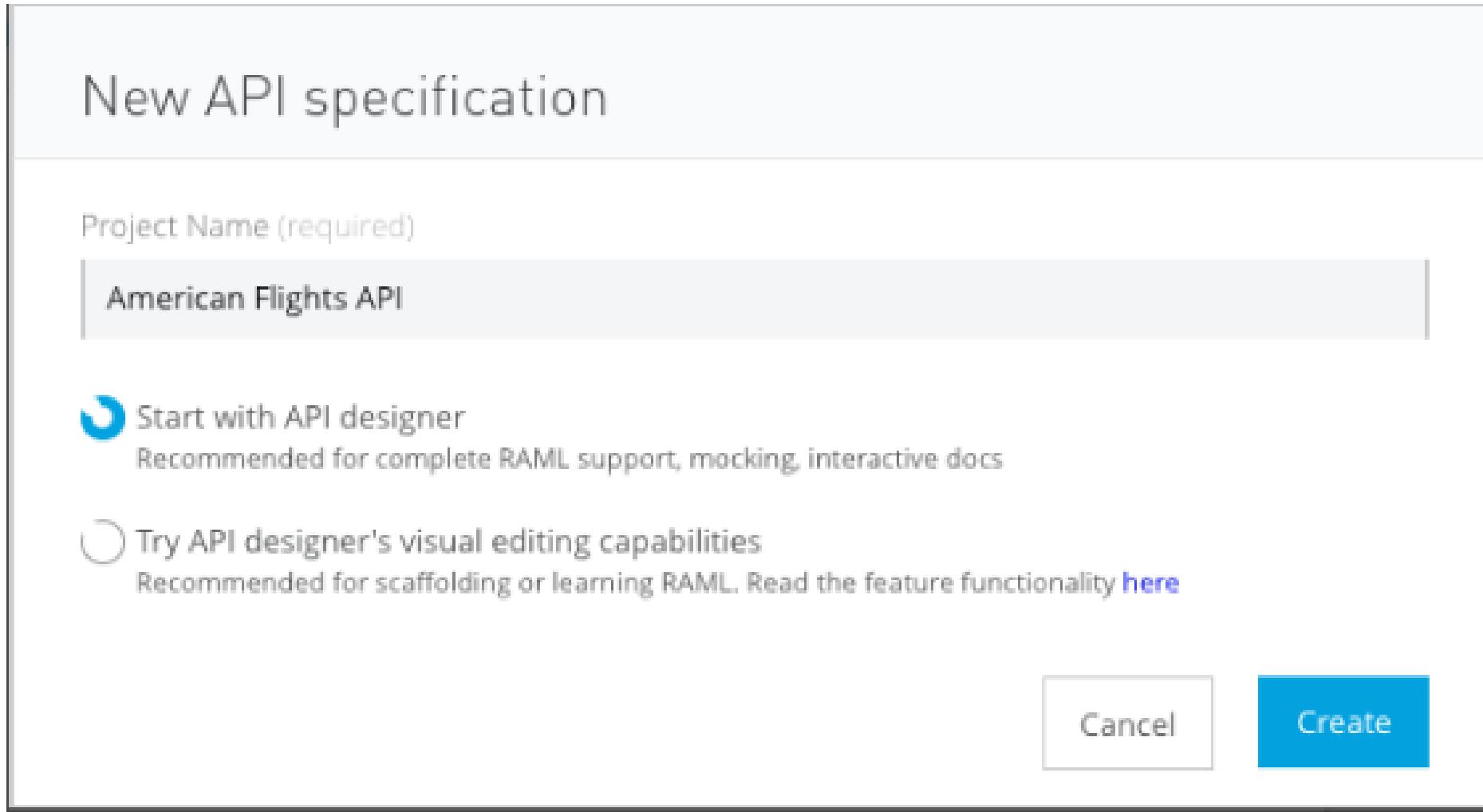
Create a new Design Center project

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



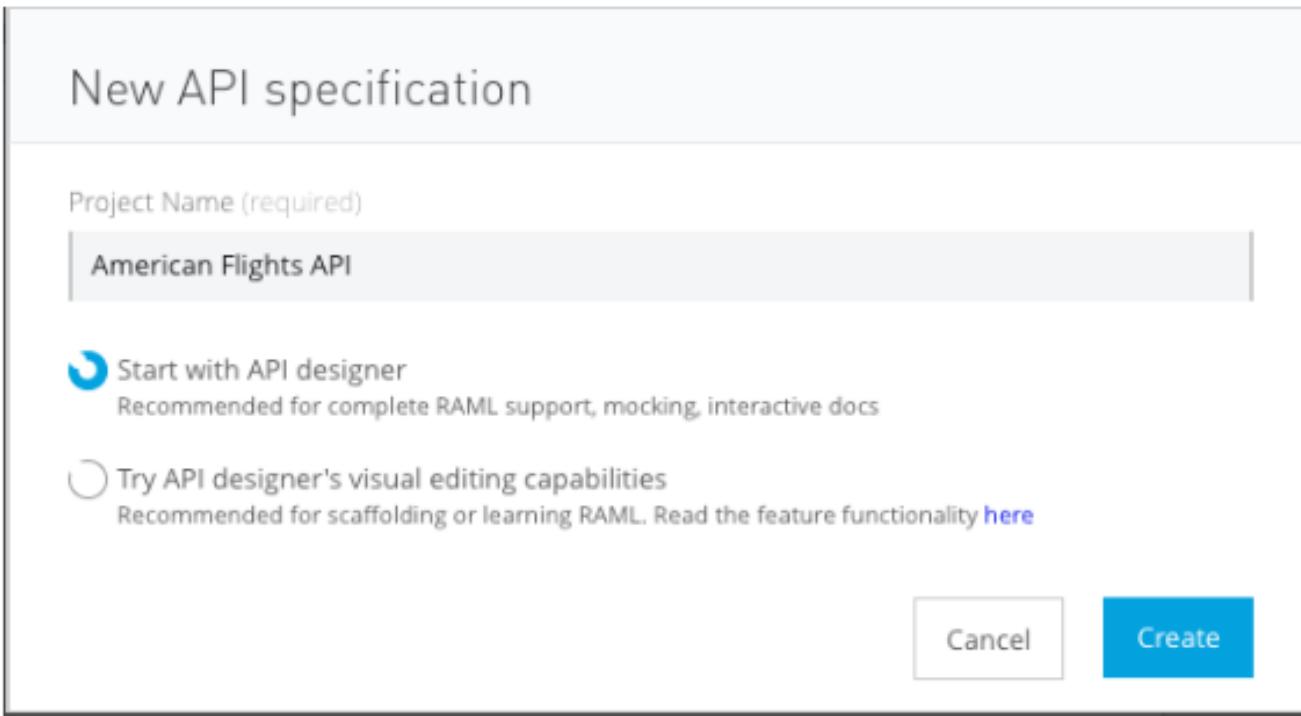
3. In the New API specification dialog box, set the project name to American Flights API.

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

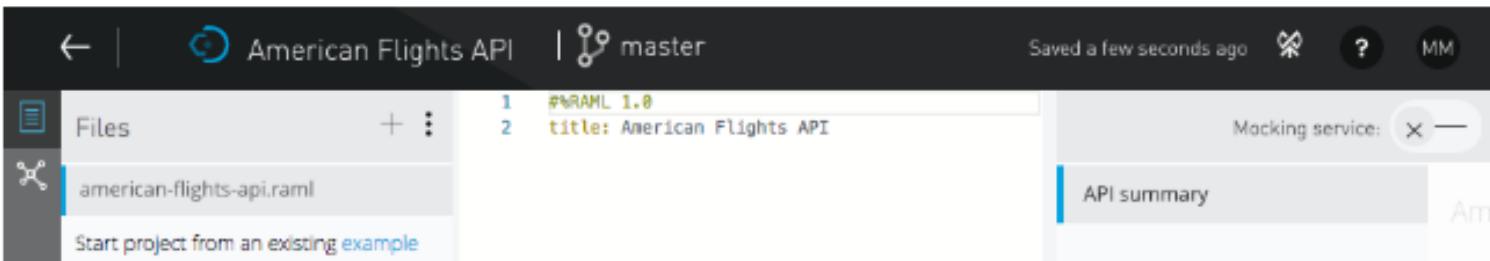


5. In API designer, click the Hide these tips link in the popup window.

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



5. In API designer, click the Hide these tips link in the popup window.
6. Review the three sections of API designer: the file browser, the editor, and the API console.



Add a RAML resource

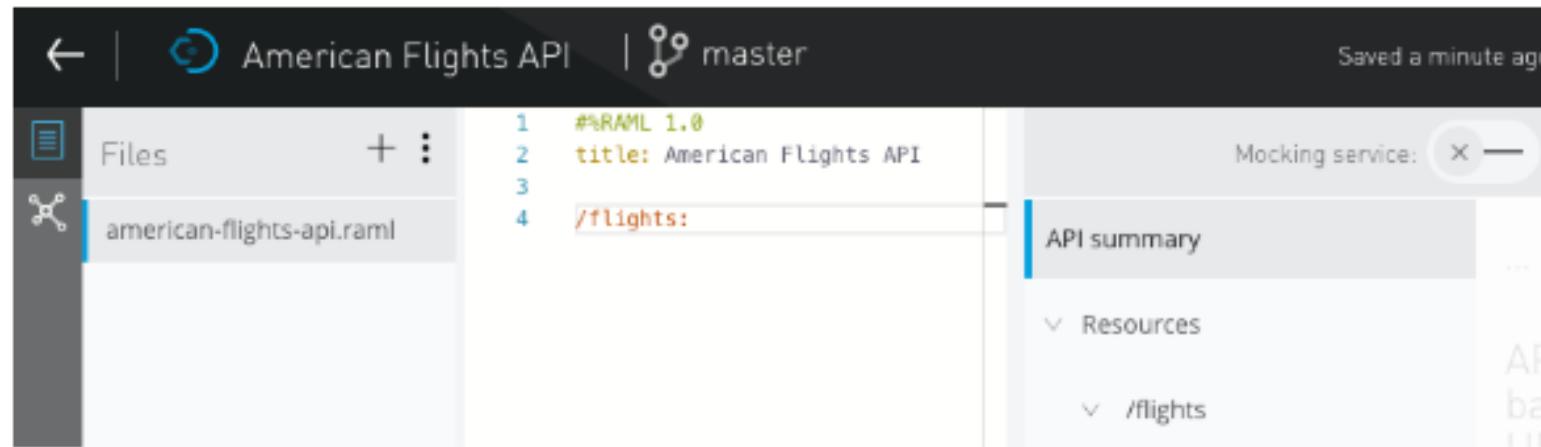
7. In the editor, place the cursor on a new line of code at the end of the file.
8. Add a resource called flights.

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

View the API console

9. 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 a web-based API development environment. At the top, there's a header bar with a back arrow, a logo, the project name "American Flights API", a branch indicator "master", and a timestamp "Saved a minute ago". Below the header is a sidebar on the left containing icons for "Files" and "Dependencies", with "american-flights-api.raml" selected. The main area displays the RAML code for the API:

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

To the right of the code, there's a "Mocking service:" button with a close icon. A large, semi-transparent modal window is overlaid on the page, titled "API summary". It contains a tree view with nodes "Resources" and "/flights". The bottom right corner of the modal has some partially visible text: "AP" and "base".

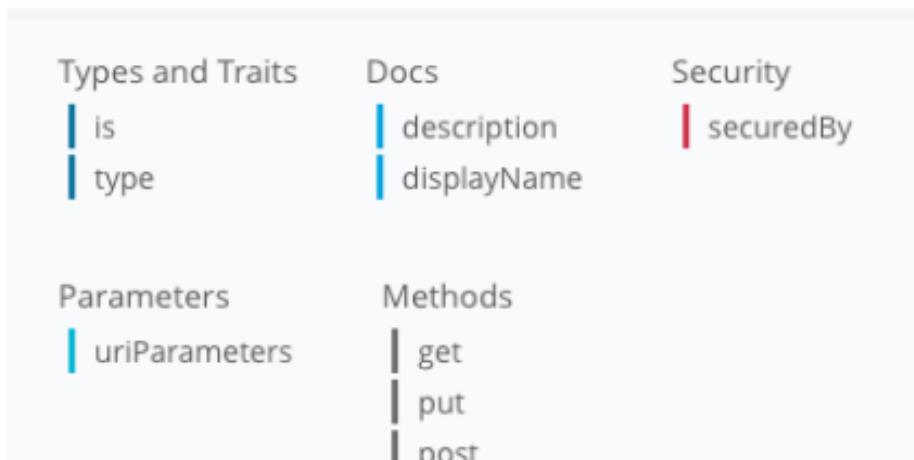
Add RAML methods

10. 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.

11. 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
```



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

16. Look at the API console; you should see GET and POST methods for the flights resource.

The screenshot shows the RAML 1.0 specification for the American Flights API. The title is "American Flights API". The /flights resource has two methods: get (highlighted in green) and post (highlighted in purple). The API summary and resources sections are also visible.

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

API summary

Resources

/flights

GET

POST

Add a nested RAML resource

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

```
/{{ID}}:
```

20. Add a get method to this resource.
21. Look at the API console and expand the /{{ID}} resource; you should see the nested resource with a GET method.

The screenshot shows the API Platform interface. On the left, the RAML editor displays the following code:

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

On the right, the API console shows the API summary and a tree view of resources. Under the /flights resource, there is a GET method. Under the /{{ID}} resource, there is also a GET method.

Add an optional query parameter

22. In the editor, indent under the /flights get method (not the /flights/{ID} get method).
23. In the shelf, click the queryParameters parameter.
24. 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}:
```

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

30. Set enum to a set of values including SFO, LAX, and CLE.

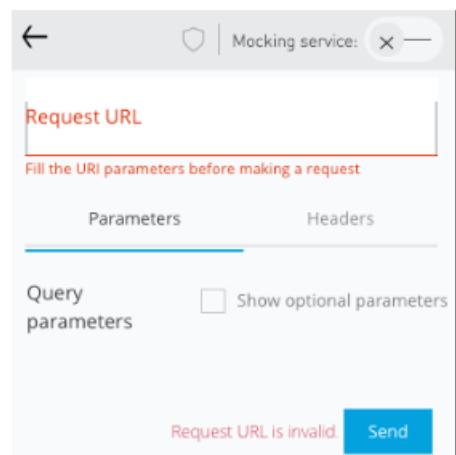
```
4   /flights:  
5     get:  
6       queryParameters:  
7         destination:  
8           required: false  
9           enum:  
10          - SF0  
11          - LAX  
12          - CLE
```

Try to call an API method using the API console

31. In the API console, click the GET method for the /flights resource.
32. Review the information.

The screenshot shows a user interface for an API console. At the top, there is a header with a back arrow, the text "Mocking service:", and a close button. Below the header, the endpoint "/flights : get" is displayed next to a blue "Try it" button. A large "Request" section follows, containing the HTTP method "GET /flights". Underneath, there are sections for "Parameters" and "Properties". A horizontal line separates these from the "Query parameters" section, which contains the parameter "destination string(enum)". Below this, the possible values "SFO, LAX, CLE" are listed. Another blue "Try it" button is located at the bottom right of the main content area.

33. 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.

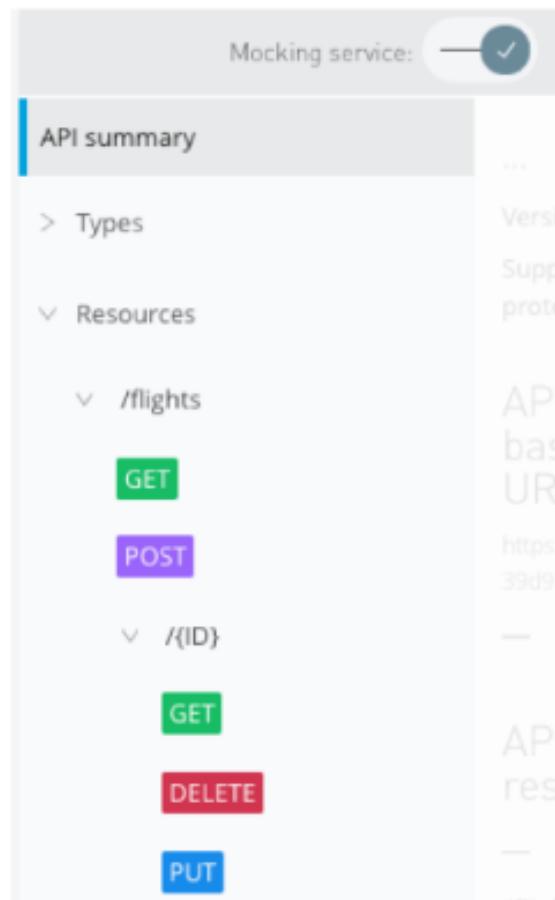


Testing API design without writing code

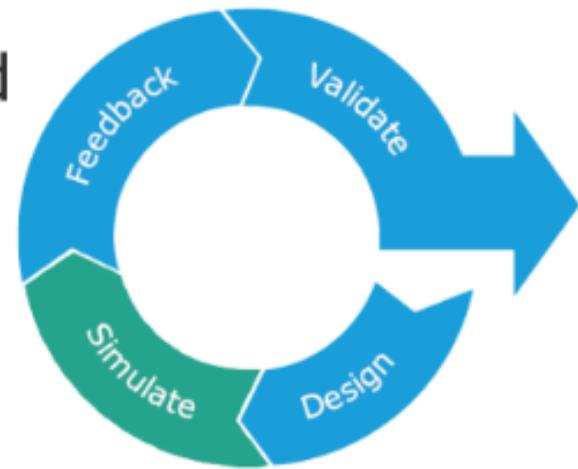


Simulating an API

- You can mock an API to test it before it is implemented
 - Useful to get early feedback from developers
- Use the **API console** and the **mocking service** to run a live simulation
 - Returns sample API responses defined in the API definition
- The API console is available in
 - **API designer** – so the API designer can test it
 - **API portals in Exchange** – so users/developers can test it



The screenshot shows the API summary page of the MuleSoft API console. At the top, there is a checkbox labeled "Mocking service:" which is checked. Below this, the "API summary" section is visible, showing a tree structure of API resources. Under "Resources", there is a node for "/flights" which has two methods listed: "GET" and "POST". Underneath "/flights", there is another node for "/{ID}" which also has two methods listed: "GET" and "DELETE". To the right of the tree structure, there are several vertical columns of information, including "Versi", "Supp", "proti", "AP", "bas", "UR", "https", "39d9", "—", "AP", "res", "—", and "/flights".



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



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

The screenshot shows the MuleSoft API Studio interface. On the left, the 'Files' sidebar shows a single file named 'american-flights-api.raml'. The main workspace displays the following RAML 1.0 code:

```
1 #@RAML 1.0
2 baseUri:
3   https://mocksvc.mulesoft.com/mocks/7a9138df
4   -b911-4d24-a66d-6e906da868b8 #
5   title: American Flights API
6
7 /flights:
8   get:
9     queryParameters:
10       destination:
11         required: false
12         enum:
13           - SFO
14           - LAX
15           - CLE
16
17   post:
18     /{ID}:
19       get:
```

To the right, the 'Mocking service' panel is open. It shows the 'Request URL' as `https://mocksvc.mulesoft.com/mocks/7a9138df`. Under the 'Parameters' tab, there is a dropdown menu set to 'SFO'. A 'Send' button is located below the parameters. At the bottom, a response message is displayed: `200 OK 123.10 ms`, followed by a JSON object with the key `"message": "RAML had no response information for application/json"`.

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 with two main panels. On the left, the 'Files' panel displays a RAML file named 'american-flights-api.raml'. The code content is as follows:

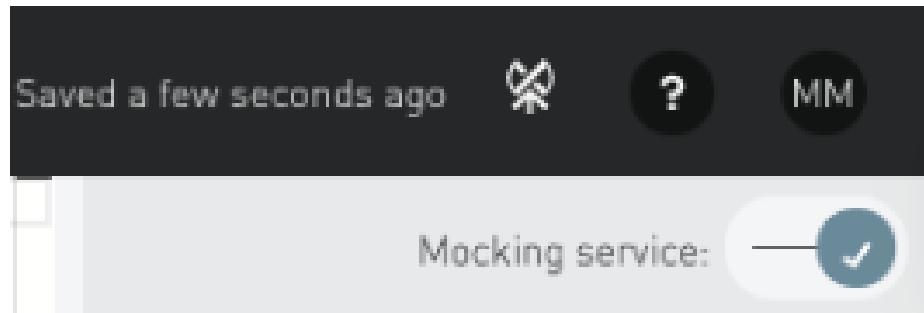
```
1 #RAML 1.0
2 baselUri:
3   https://mocksvc.mulesoft.com/mocks/7a9138df
4   -b911-4d24-a66d-4e90da868bb. #
5   title: American Flights API
6
7 /flights:
8   get:
9     queryParameters:
10       destination:
11         required: false
12         enum:
13           - SFO
14           - LAX
15           - CLE
16
17   post:
18     /{ID}:
19       get:
```

On the right, the 'Mocking service' panel is open, showing a request URL 'https://mocksvc.mulesoft.com/mocks/7a9138df'. Under the 'Parameters' tab, there is a dropdown set to 'SFO'. A 'Send' button is visible. Below the parameters, a status bar shows '200 OK 123.10 ms'. The response body is displayed as:

```
{ "message": "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.



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

```
1  #%RAML 1.0
2  baseUri: https://mocksvc.mulesoft.com/mocks/f02c935c-66b1-4a68-9bf2-d7beb3affe53 #
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.

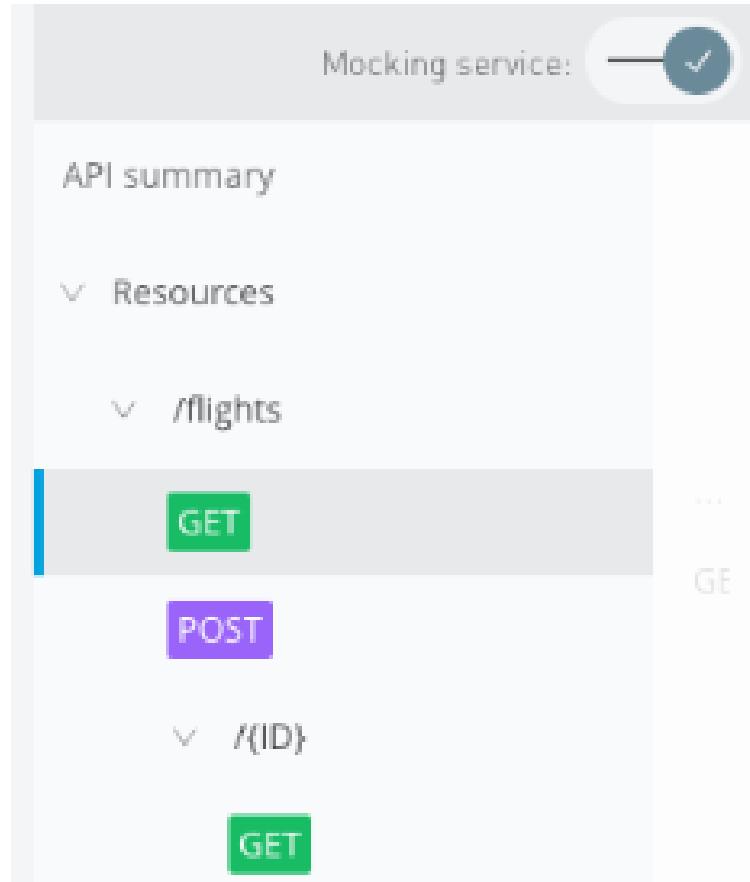
The screenshot shows the Mocking service API console interface. At the top, there's a back arrow, a shield icon, and the text "Mocking service: [checkmark]". Below that is a "Request URL" input field containing "https://mocksvc.mulesoft.com/ mocks/7a". There are two tabs: "Parameters" (which is selected) and "Headers". Under "Parameters", there's a section for "Query parameters" with a checkbox labeled "Show optional parameters" which is unchecked. A large blue "Send" button is centered below the parameters section. At the bottom, the response is shown: a green "200 OK" status box with "119.20 ms" latency, followed by a dropdown menu. Below that is a toolbar with icons for copy, paste, refresh, and others. The main content area displays a JSON placeholder response:

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

6. Select the Show optional parameters checkbox.
7. In the destination text field, select SFO and click Send; you should get the same response.

Test the /flights/{ID} resource

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



- 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. At the top, there is a back arrow, a shield icon, and the text 'Mocking service:' followed by a toggle switch which is turned on (indicated by a checkmark). Below this, the 'Request URL' field contains the value 'https://mocksvc.mulesoft.co'. A red error message 'Fill the URI parameters before making a requ...' is displayed below the URL field. There are two tabs: 'Parameters' (which is selected, indicated by a blue underline) and 'Headers'. Under the 'Parameters' tab, there is a single parameter named 'ID*' in a text input field. At the bottom of the form, there is a red error message 'Request URL is Invalid.' next to the 'Send' button, which is highlighted with a blue background.

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

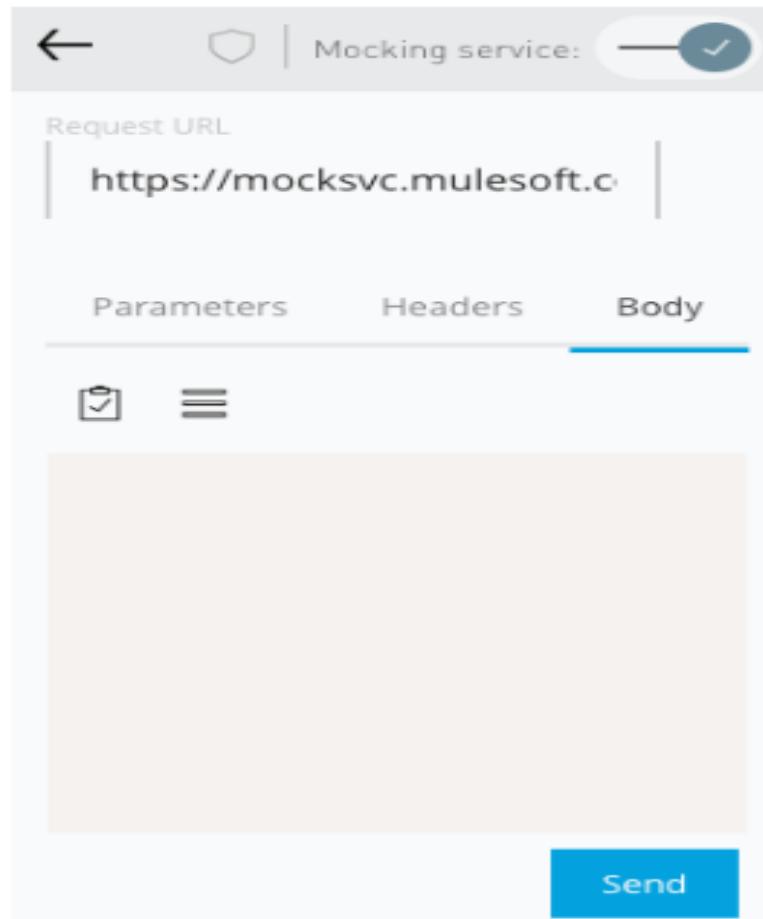
12. Click the Send button.

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 a web-based interface for a 'Mocking service'. At the top, there's a back arrow, a shield icon, and the text 'Mocking service:' followed by a toggle switch that is turned on (indicated by a checkmark). Below this is a 'Request URL' input field containing the value 'https://mocksvc.mulesoft.com/mocks/7a'. There are two tabs: 'Parameters' (which is selected, indicated by a blue underline) and 'Headers'. Under the 'Parameters' tab, there is a section for 'URI parameters' with a single entry: 'ID*' with the value '10'. In the center, there is a large blue 'Send' button. Below the 'Send' button, the response is displayed in a green box with the status '200 OK' and a time of '174.90 ms'. To the right of the status, there is a dropdown arrow. Below the status, there are four small icons: a copy icon, a refresh icon, a comparison icon, and a refresh icon. The main content area shows a JSON response: { "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.

200 OK 116.70 ms ▾

✖️ ↻ ⌂ ⌂

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

Using RAML to define specifications for requests and responses



- Responses must be a map of one or more HTTP status codes
- For each response, specify possible return data types along with descriptions and examples

```
6   /flights:  
7     get:  
8       responses:  
9         200:  
10        body:  
11          application/json:  
12            example:  
13              ID: 1  
14              code: GQ574  
15              price: 399  
16              departureDate: 2016/12/20  
17              origin: ORD  
18              destination: SFO  
19              emptySeats: 200  
20        plane:  
21          type: Boeing 747  
22          totalSeats: 400
```

- For a request, similarly specify the possible request data types along with data types, descriptions, and examples

```
6   /flights:  
7   + get: ...  
23  post:  
24    displayName: Add a flight  
25    body:  
26      application/json:  
27        example:  
28          code: GQ574  
29          price: 399  
30          departureDate: 2016/12/20  
31          origin: ORD  
32          destination: SFO  
33          emptySeats: 200  
34          plane:  
35            type: Boeing 747  
36            totalSeats: 400
```

- Instead of including all code in one RAML file, you can modularize it and compose it of reusable fragments
 - **Data types, examples**, traits, resource types, overlays, extensions, security schemes, documentation, annotations, and libraries
- Fragments can be stored
 - In different files and folders within a project
 - In a separate API fragment project in Design Center
 - In a separate RAML fragment in Exchange

Walkthrough 3-3: Add request and response details



- 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

The screenshot shows the MuleSoft Anypoint Studio interface. On the left, the project tree displays the 'American Flights API' project with various files and modules. The 'american-flights-api.raml' file is currently selected and shown in the main code editor area. The code is a RAML 1.0 specification for the American Flights API, defining endpoints for flights, including query parameters like destination and required fields like enroute. It also defines responses for 200 OK and 400 Bad Request. To the right of the code editor is a preview pane showing a successful 200 OK response with a JSON body containing flight details such as ID, code, price, departure date, origin, destination, empty seats, and plane information.

```
#API 1.0
baseUri: https://mackavec.mulesoft.com/mocks/6221ede5-6246-4462-a300-5a4d9d4a1c2
title: American Flights API
types:
  AmericanFlight: !include exchange_modules/6bef9520-24e9-4cf2-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/6bef9520-24e9-4cf2-b2f5-620025690913/training-american-flights-example/1.0.1/AmericanFlightExample.raml
      400:
        body:
          application/json:
            type: AmericanFlight
            example: !include exchange_modules/6bef9520-24e9-4cf2-b2f5-620025690913/training-american-flights-example/1.0.1/AmericanFlightExample.raml
  post:
    body:
      application/json:
        type: AmericanFlight
        example: !include exchange_modules/6bef9520-24e9-4cf2-b2f5-620025690913/training-american-flights-example/1.0.1/AmericanFlightExample.raml
```

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.

The screenshot shows the MuleSoft Anypoint Studio interface. On the left, the project tree displays files like `AmericanFlightExample.raml`, `AmericanFlightNoIDExample.raml`, and `american-flights-api.raml`. The right panel shows the code editor for `american-flights-api.raml` and a preview of a Mocking service response.

Code Editor Content (`american-flights-api.raml`):

```
1  #@NAME: 1.0
2  baseUri:
3      https://mocksvc.mulesoft.com/mocks/6822ed05-6246-4462-a380-6
4      e499d1c3 #
5  title: American Flights API
6  types:
7      AmericanFlights !include
8          exchange_modules/68ef9520-24e9-4cf2-b2f5-628825698913/training-
9          american-flight-data-type/1.0.1/AmericanFlightDataType.raml
10
11 /flights:
12     get:
13         queryParameters:
14             destination:
15                 required: false
16                 enum:
17                     - SFO
18                     - LAX
19                     - CLE
20         responses:
21             200:
22                 body:
23                     application/json:
24                         type: AmericanFlight []
25                         example: !include
26                             exchange_modules/68ef9520-24e9-4cf2-b2f5-628825698913/training-
27                             american-flights-example/1.0.1/AmericanFlightsExample.raml
28
29         post:
30             body:
31                 application/json:
32                     type: AmericanFlight
33                     example: !include
34             examples/AmericanFlightNoIDExample.raml
35             responses:
36                 201:
37                     body:
```

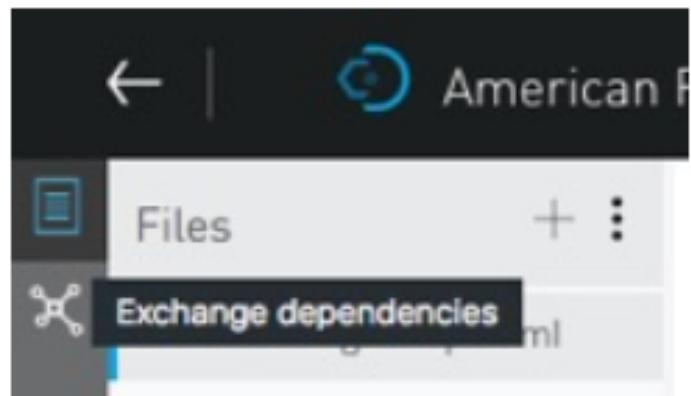
Mocking Service Preview:

200 OK 125.69 ms

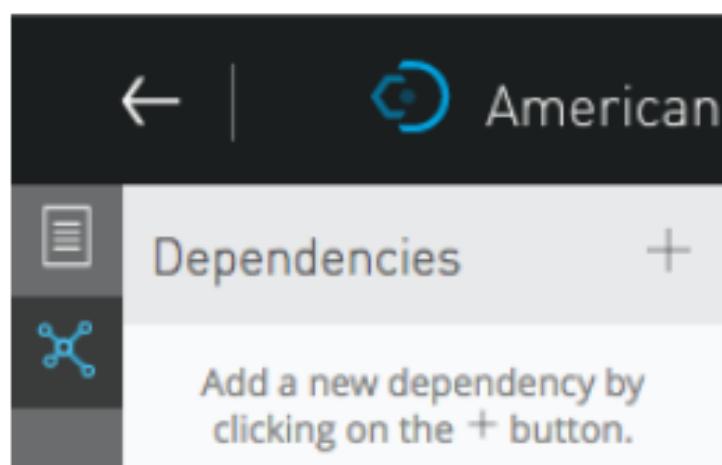
```
[Array(2)
  -0: {
    "ID": 1,
    "code": "ER38sd",
    "price": 400,
    "departureDate": "2017/07/26",
    "origin": "CLE",
    "destination": "SFO",
    "emptySeats": 0,
    "plane": {
      "type": "Boeing 737",
      "totalSeats": 150
    }
  },
  -1: {
    "ID": 2,
    "code": "ER451f",
    "price": 540.99,
    "departureDate": "2017/07/27",
    "origin": "SFO",
    "destination": "ORD",
    "emptySeats": 54,
    "plane": {
      "type": "Boeing 777",
      "totalSeats": 100
    }
  }
]
```

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.

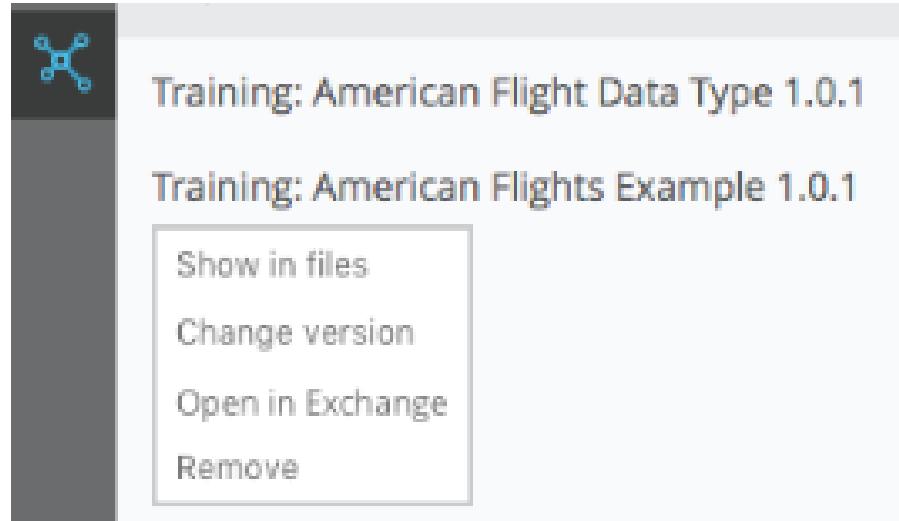
Consume API Fragment

All ▼ 🔍

Name ^	Date modified	Rating	Created by	Business group
<input type="checkbox"/> FHIR DSTU-2 Data Types	Mar 10, 2018	★★★★★	ND Nial Darbey	MuleSoft
<input type="checkbox"/> REST Connect Library	Feb 19, 2018	★★★★★	MO MuleSoft Organization	MuleSoft
<input checked="" type="checkbox"/> Training: American Flight Data Type	Feb 19, 2018	★★★★★	MO MuleSoft Organization	MuleSoft
<input checked="" type="checkbox"/> Training: American Flights Example	Feb 19, 2018	★★★★★	MO MuleSoft Organization	MuleSoft
<input type="checkbox"/> Training: Cacheable Trait	Feb 19, 2018	★★★★★	MO MuleSoft Organization	MuleSoft
<input type="checkbox"/> Training: OAuth2.0 Security Scheme	Feb 19, 2018	★★★★★	MO MuleSoft Organization	MuleSoft

Cancel Add 2 dependencies

5. Click the Add 2 dependencies button.



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

American Flights API | master

Files + :

exchange_modules

```
1  #%RAML 1.0 DataType
2
3  type: object
4  properties:
5    ID?: integer
6    code: string
```

10. 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

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.

```
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.

```
%RAML 1.0 NamedExample
value:
-
  ID: 1
  code: ER38sd
  price: 400
  departureDate: 2017/07/26
  origin: CLE
  destination: SFO
  emptySeats: 0
  plane:
    type: Boeing 737
    totalSeats: 158
-
  ID: 2
  code: ER451f
  price: 548.99
  departureDate: 2017/07/27
  origin: SFO
  destination: ORD
  emptySeats: 54
  plane:
    type: Boeing 777
    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.

```
responses:
  200:
    body:
      application/json:
        type: AmericanFlight[]
        example: !include exchange_modules/86f74f12-decb-452c
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.

Response

Type	application/json									
200	<p>Type</p> <p>Examples</p> <pre>[{ "ID": "integer", "code": "string", "price": "number", "departureDate": "string", "origin": "string", "destination": "string", "emptySeats": "integer", "plane": { "type": "string", "totalSeats": "integer" } }]</pre> <p>Parameter</p> <p>Type</p> <p>Description</p> <table border="1"><tr><td>Array of AmericanFlight items</td><td></td><td></td></tr><tr><td>item. ID</td><td>integer</td><td></td></tr><tr><td>item. code (required)</td><td>string</td><td></td></tr></table>	Array of AmericanFlight items			item. ID	integer		item. code (required)	string	
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

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

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

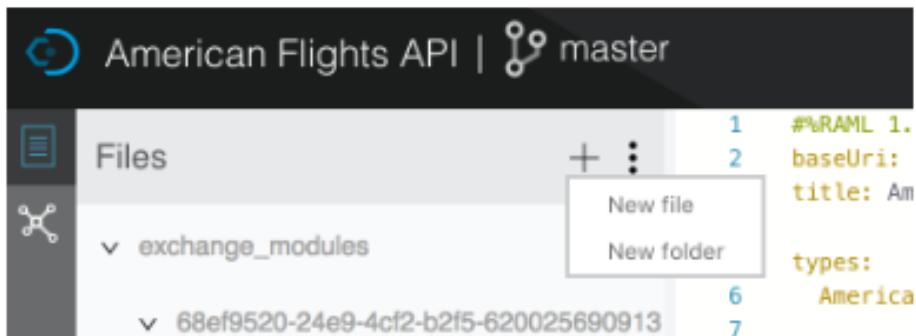
Specify the `/{ID}:get` method to return an `AmericanFlight` object

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

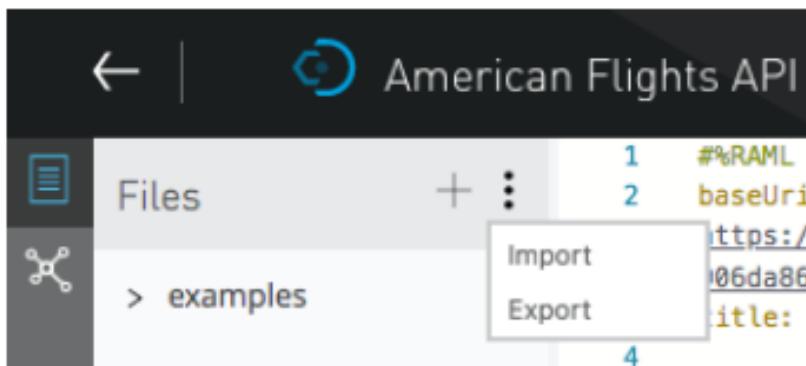
```
/{ID}:
  get:
    responses:
      200:
        body:
          application/json:
            type: AmericanFlight
```

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.

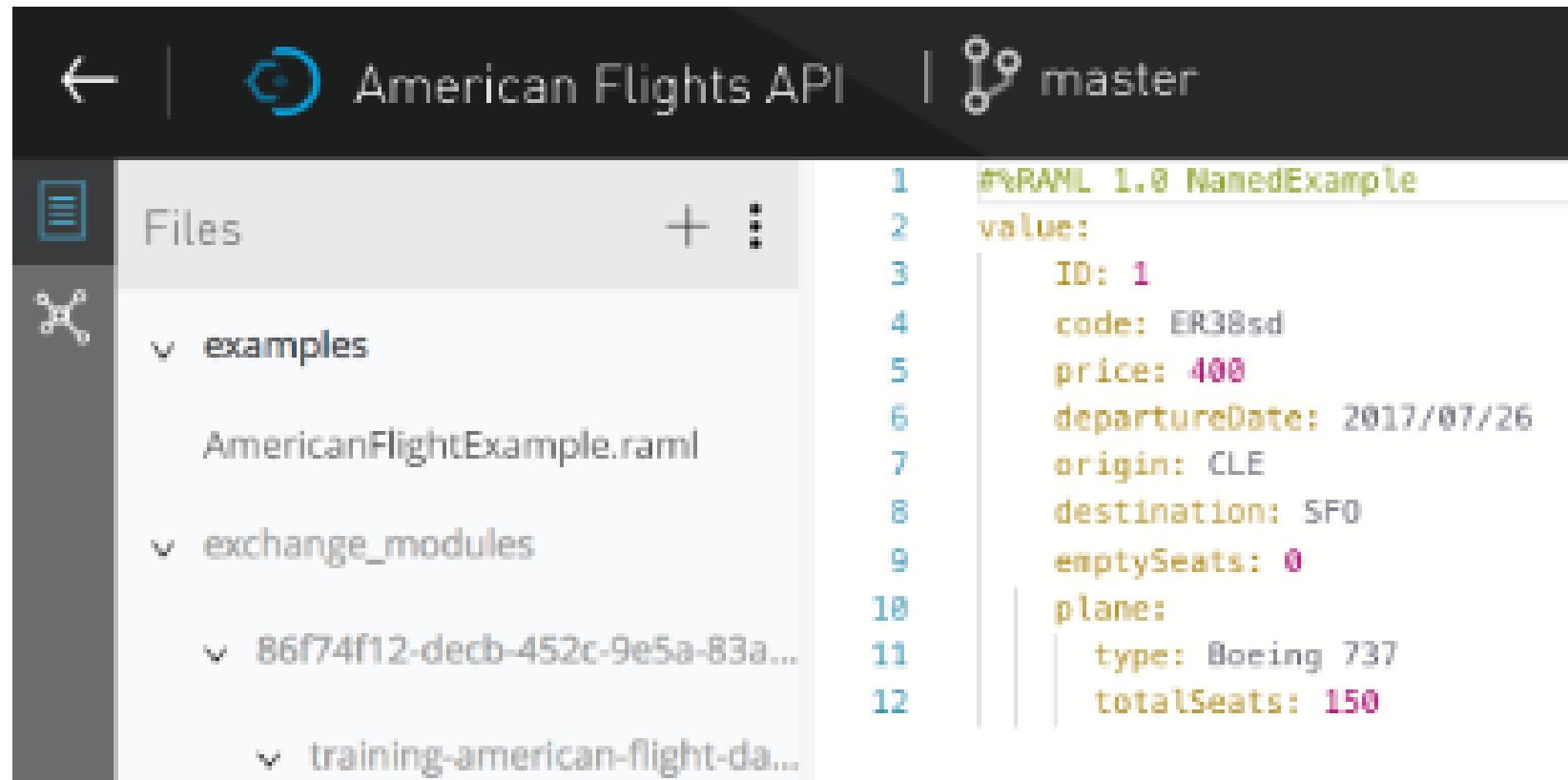


31. In the Add new folder dialog box, set the name to examples and click Create.
32. In the file browser, click the menu button and select Import.



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.



```
1  #%RAML 1.0 NamedExample
2  value:
3    ID: 1
4    code: ER38sd
5    price: 400
6    departureDate: 2017/07/26
7    origin: CLE
8    destination: SFO
9    emptySeats: 0
10   plane:
11     type: Boeing 737
12     totalSeats: 150
```

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.

Type AmericanFlight

200

Type Examples

Copy ☰

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

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 Mocking service interface with the following details:

- Request URL:** `https://mocksvc.mulesoft.com/m`
- Parameters:** A table with one row containing `ID*` and the value `10`.
- Headers:** None listed.
- Send:** A blue button to execute the request.
- Response:**
 - Status:** 200 OK
 - Time:** 384.00 ms
 - Actions:** Copy, Share, Refresh, More
 - Flight Data:**

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

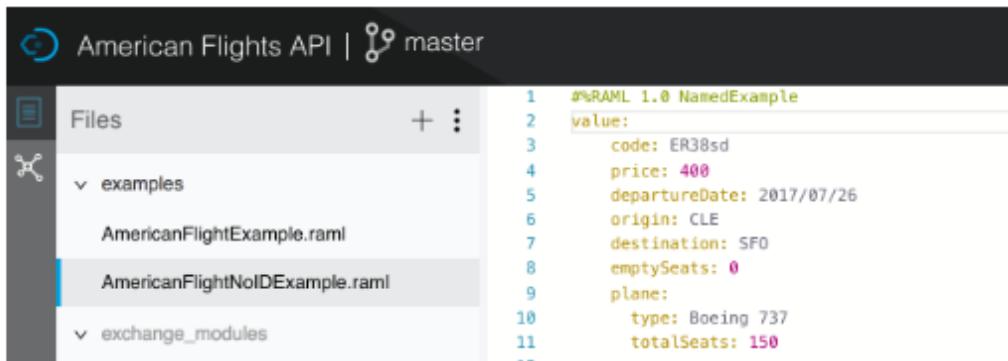
Specify the /flights:post method request to require an AmericanFlight object

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
```

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.

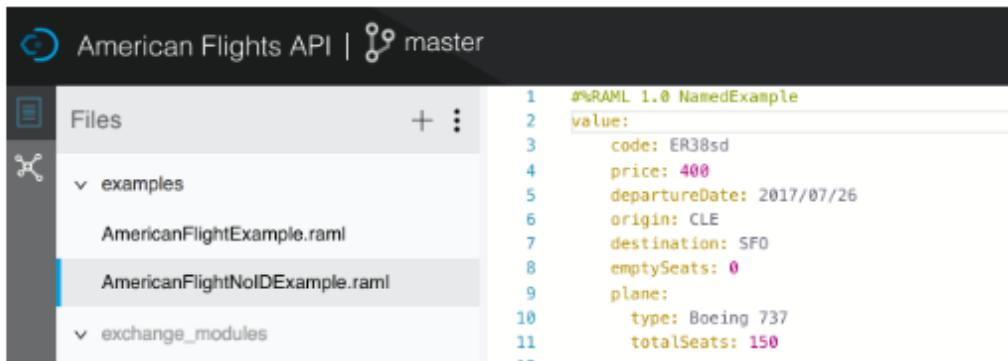
```
24   post:  
25     body:  
26       application/json:  
27         type: AmericanFlight  
28         example: !include examples/AmericanFlightNoIDExample.raml  
29     responses:  
30       201:  
31         body:  
32           application/json:  
33
```

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)
```

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
```

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

Type

```
{  
    "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.

The screenshot shows a web-based interface for a 'Mocking service'. At the top, there's a back arrow, a shield icon, and the text 'Mocking service:' followed by a checkmark. Below that is a 'Request URL' input field containing 'https://mocksvc.mulesoft.com/mocks/6822ede5-'. Underneath the URL, there are three tabs: 'Parameters', 'Headers', and 'Body', with 'Body' being the active tab. The body content area contains a JSON object:

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

At the bottom right of the body content area is a blue 'Send' button.

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

The screenshot shows the response details after clicking the 'Send' button. At the top left, it says '201 Created' and '494.58 ms'. To the right is a dropdown arrow. Below that is a row of icons: a copy icon, a refresh icon, a comparison icon, and a refresh icon. The main content area displays a JSON object:

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

64. In the body, remove the emptySeats properties.

Parameters Headers Body

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

Send

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

400 Bad Request 164.43 ms ▾

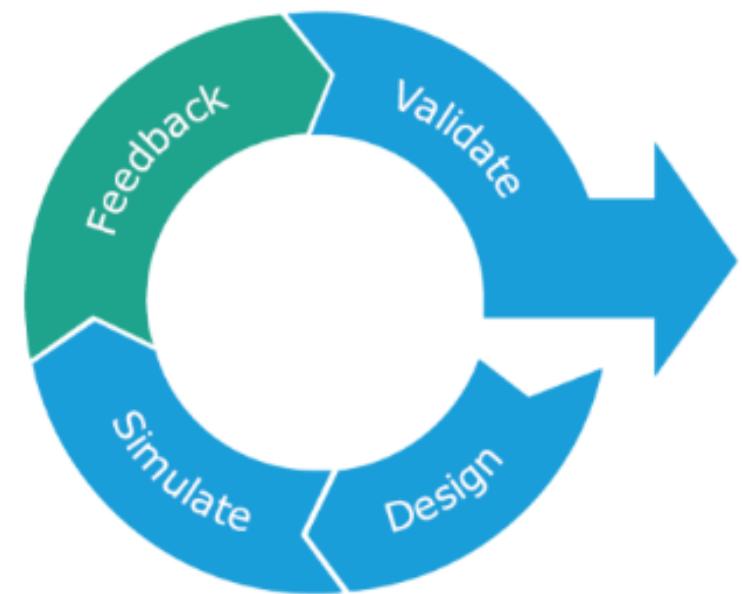
 The requested URL can't be reached
The service might be temporarily down or it may have moved permanently to a new web address.
Resource is unavailable

66. Add the emptySeats property back to the body.

Engaging with users



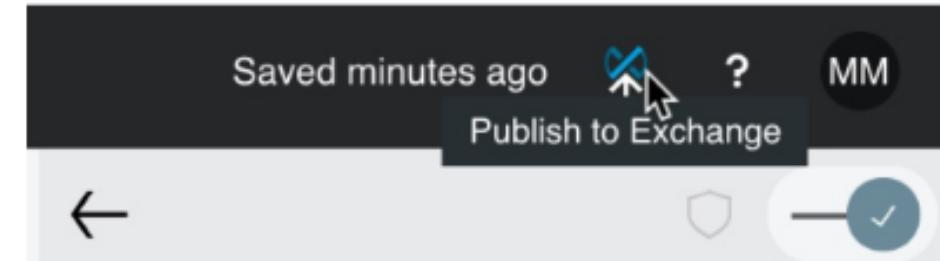
- To build a successful API, you should define it iteratively
 - Get feedback from developers on usability and functionality along the way
- To do this, you need to provide ways for developers to discover and play with the API
- Anypoint Platform makes this easy with
API portals in Exchange
 - In **private Exchange** for internal developers
 - In a **public portal** for external developers



- You publish RAML API Specifications and RAML fragments to the Exchange

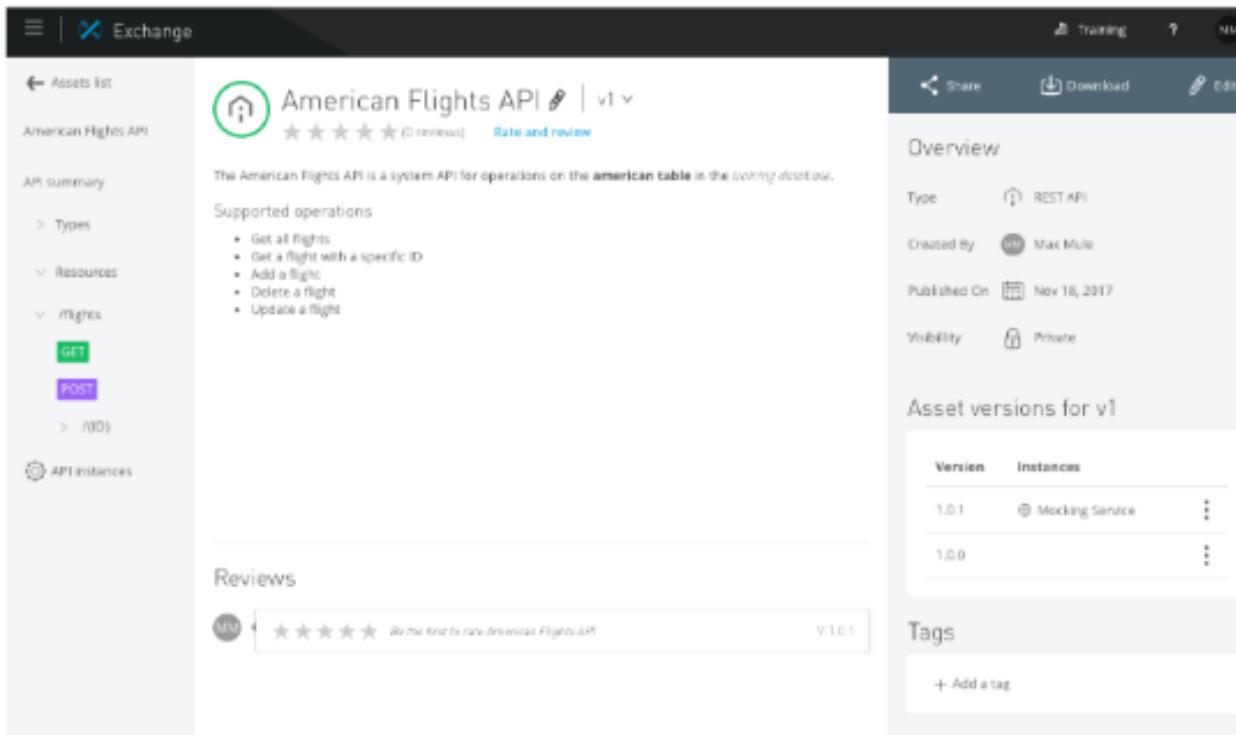
from API designer

- Not from Exchange itself



- **API portals** are automatically created for REST APIs added to Exchange
 - An **API console** for consuming and testing APIs
 - An **automatically generated API endpoint** that uses a **mocking service** to allow the API to be tested without having to implement it
- API portals can be shared with both internal and external users

- 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 following details:

- API Summary:** American Flights API (v1)
- Rating:** ★★★★ (0 reviews) | Rate and review
- Description:** The American Flights API is a system API for operations on the `american` table in the `working` database.
- Supported operations:**
 - Get all flights
 - Get a flight with a specific ID
 - Add a flight
 - Delete a flight
 - Update a flight
- API Instances:** GET /flights, POST /flights
- Reviews:** No reviews yet.
- Overview:**
 - Type: REST API
 - Created By: Max Mule
 - Published On: Nov 18, 2017
 - Visibility: Private
- Asset versions for v1:**

Version	Instances
1.0.1	Mocking Service
1.0.0	
- Tags:** + Add a tag

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 following details:

- API Summary:** American Flights API (v1)
- Rating:** ★★★★☆ (0 reviews) | [Rate and review](#)
- Supported operations:**
 - Get all flights
 - Get a flight with a specific ID
 - Add a flight
 - Delete a flight
 - Update a flight
- API Methods:** GET, POST
- API Instances:** > API
- Reviews:** No reviews yet.
- Overview:**
 - Type: REST API
 - Created By: Max Muñoz
 - Published On: Nov 18, 2017
 - Visibility: Private
- Asset versions for v1:**

Version	Instances
v1.0.1	Marking Secure
v1.0.0	
- Tags:** + Add a tag
- Dependencies:**
 - Training: American Flights Example (1.0.1) RAML fragment
 - Training: American Flight Data Type (1.0.1) RAML fragment

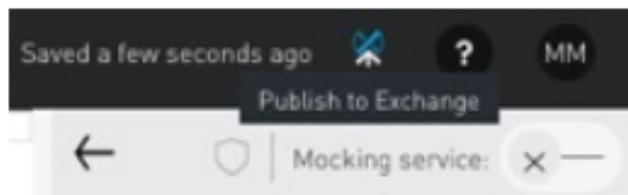
Remove the baseUri by turning off the mocking service

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
```

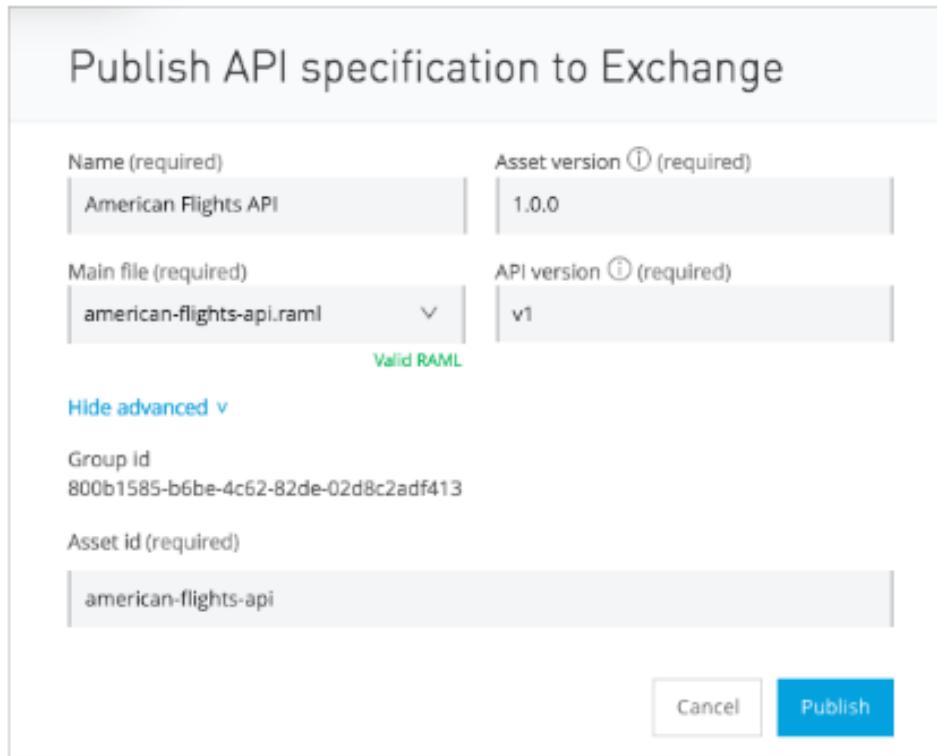
Publish the API to Anypoint Exchange from API designer

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.

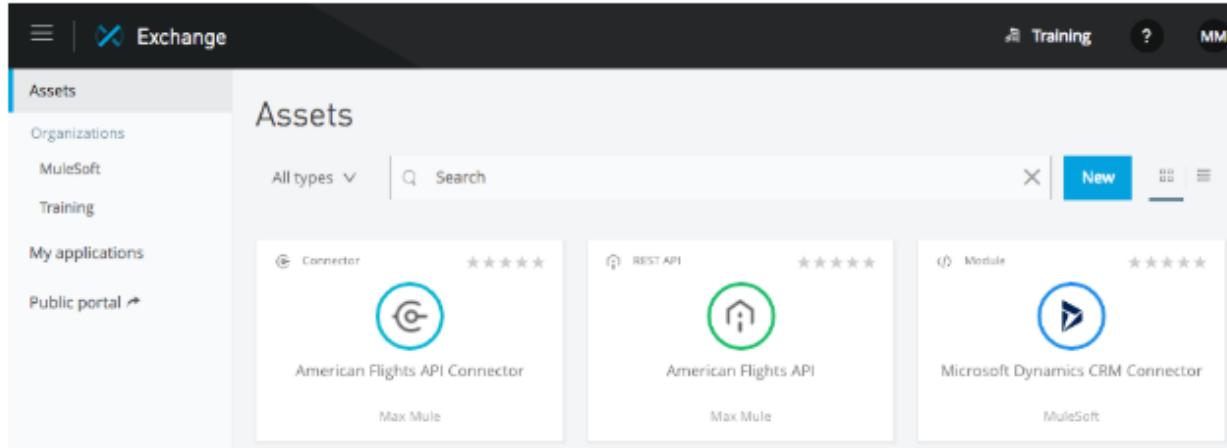


8. Click the Publish button.
9. In the Publish API specification to Exchange dialog box, click Done.
10. Look at the RAML file; you should see a version has been added.

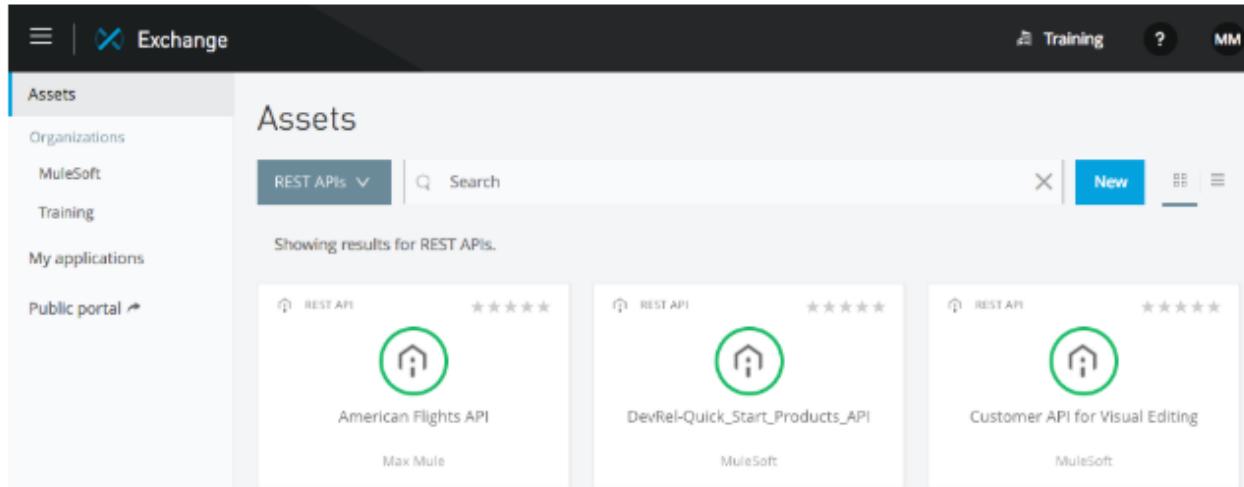
```
1  #%RAML 1.0
2  version: v1
3  title: American Flights API
```

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.



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



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

Assets

REST APIs ▾ Search X New

Showing results for REST APIs.

REST API	Rating	Description	Provider
DevRel-Quick_Start_Products_API	★★★★★	MuleSoft	MuleSoft
Customer API for Visual Editing	★★★★★	MuleSoft	MuleSoft
Account Information (AISP) API - RAML Definition	★★★★★	MuleSoft	MuleSoft

15. 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.

Assets

REST APIs ▾ Search X New

Showing results for REST APIs.

REST API	Rating	Description	Provider
American Flights API	★★★★★	Max Mule	Max Mule

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 Exchange interface for the American Flights API. The left sidebar has a tree view with 'Assets list' at the top, followed by 'American Flights API', 'API summary', 'Types', 'Resources', and 'flights' (expanded) with 'GET' and 'POST' methods. Below that is 'API instances'. The main content area shows the 'American Flights API v1' page. It features a green house icon, a 5-star rating (0 reviews), and a 'Rate and review' button. A large placeholder image of a horse is centered. Below it, text says 'This page currently doesn't contain a description. Click Edit to add text, images, videos, code blocks, etc...' with an 'Edit' button. To the right, there's an 'Overview' section with details: Type (REST API), Created By (Max Mule), Published On (Mar 25, 2018), and Visibility (Private). Below that is 'Asset versions for v1' showing '1.0.0' and 'Mocking Service'. There's also a 'Tags' section with '+ Add a tag' and a 'Dependencies' section listing 'Training: American Flights Example 1.0.1 RAML Fragment' and 'Training: American Flight Data Type 1.0.1 RAML Fragment'.

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 with the title bar "Exchange". On the left, there's a sidebar with "Assets list" and "American Flights API" selected. The main area displays the "American Flights API" with a green edit icon and "v1" next to it. Below it is a rating section with 5 stars and "(0 reviews)".

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". It has two columns: "Version" and "Instances". There is one entry: "1.0.0" with "Mocking Service" listed under "Instances". A three-dot menu icon is also present.

Version	Instances
1.0.0	Mocking Service

21. 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.

The screenshot shows the Exchange interface with the title bar "Exchange". The left sidebar is expanded, showing "API summary", "Types", "Resources", and "API instances" (which is selected). Under "API instances", there's a table with columns "Instances", "Environment", "URL", and "Visibility". One row is shown: "Mocking Service" with "https://mocksvc-proxy.mulesoft.com/exchange/B00b1585-b6be-4c62-82de-00d8c2ad413/api/v1/american-flights-api/1.0.0" and "Public" under "Visibility".

Instances	Environment	URL	Visibility
Mocking Service		https://mocksvc-proxy.mulesoft.com/exchange/B00b1585-b6be-4c62-82de-00d8c2ad413/api/v1/american-flights-api/1.0.0	Public

22. In the left-side navigation, expand Types.
23. Select AmericanFlight and review the information.

The screenshot shows the Exchange API documentation interface. At the top, there's a navigation bar with a menu icon, a search icon, and the text "Exchange". Below the navigation bar, the left sidebar has a "Assets list" button and displays the "American Flights API". The main content area shows the "American Flights API" with a green info icon, a 5-star rating, and "(0 reviews)". The "v1" version is selected. The "API summary" section includes a "Types" section with "AmericanFlight" expanded, showing its properties: ID (integer), code (string), price (number), departureDate (string), origin (string), destination (string), emptySeats (integer), and plane (a nested object with type and totalSeats). Below this, there are sections for "Resources", "/flights" (with GET and POST methods), and "/(ID)". The bottom section, "API instances", lists parameters: ID (integer) and code (string, required).

← Assets list

American Flights API

API summary

Types

AmericanFlight

Resources

/flights

GET

POST

> /{(ID)}

Type AmericanFlight

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

API instances

Parameter	Type	Description
ID	integer	
code (required)	string	

Test the API in its API portal in Exchange

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

The screenshot shows the MuleSoft Anypoint Platform interface. On the left, there's a sidebar with navigation links like 'Assets list', 'American Flights API', 'API summary', 'Types', 'AmericanFlight', 'Resources', 'flights' (which is expanded), and 'API instances'. Under 'flights', there are 'GET' and 'POST' buttons. The main area displays the 'American Flights API' with a green house icon. It shows a rating of 5 stars and 0 reviews. Below that, the endpoint '/flights : get' is listed with a 'Request' section showing a 'GET /flights' call. To the right, the API console for the 'GET /flights' method is shown. It has a 'GET' button at the top. Below it, the URL is set to 'Mocking Service' and 'https://mocksvc-proxy.anypoint.mulesoft.com/exc'. There are tabs for 'Parameters' (which is selected) and 'Headers'. Under 'Parameters', there's a dropdown for 'Query parameters' containing 'LAX', with a checkbox for 'Show optional parameters'. A 'Send' button is at the bottom right of the console area.

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

GET

Mocking Service

https://mocksvc-proxy.anypoint.mulesoft.com/exc

Parameters Headers

Query parameters Show optional parameters

LAX

Send

200 OK 582.08 ms Details ▾

```
[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",
    "price": 350,
    "departureDate": "2017/07/27",
    "origin": "SFO",
    "destination": "CLE",
    "emptySeats": 0,
    "plane": {
      "type": "Airbus A320",
      "totalSeats": 140
    }
  }
]
```

Add information about the API

28. In the left-side navigation, select 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).

Supported operations

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

36. Repeat for the other operations.



Markdown Visual

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

A screenshot of the Microsoft Power BI interface. The top navigation bar shows "Assets list" and "Exchange". The main content area is titled "American Flights API" with a "v1" version indicator. On the left, there's a sidebar with options: "+ Add new page", "API summary" (which is selected), "+ Add terms and conditions", and "API instances". The main content pane displays the rendered markdown from step 35, including the bolded text and the list of supported operations. At the bottom right, there are "Discard changes" and "Save as draft" buttons.

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. On the left sidebar, there are links for 'Assets list', 'American Flights API', 'API summary', and 'Types'. The main content area displays the 'American Flights API' page. A banner at the top says 'This is a draft that has not been published yet.' with buttons for 'Exit Draft', 'Publish', and 'Edit'. The API icon is a green house-like icon. The title is 'American Flights API' with a edit icon and 'v1'. Below the title is a 5-star rating with '(0 reviews)'. A description states: 'The American Flights API is a system API for operations on the american table in the training database.' To the right, the 'Overview' section shows 'Type: REST API' and 'Created By: Max Mule'. The 'Edit' button is highlighted in blue.

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

The screenshot shows the MuleSoft Exchange interface after publishing. The left sidebar remains the same. The main content area now shows the published 'American Flights API' page. The banner at the top now says 'Published' with buttons for 'Share', 'Download', and 'Edit'. The API icon is a green house-like icon. The title is 'American Flights API' with a edit icon and 'v1'. Below the title is a 5-star rating with '(0 reviews)' and a 'Rate and review' link. A description states: 'The American Flights API is a system API for operations on the american table in the training database.' The 'Supported operations' section lists: 'Get all flights', 'Get a flight with a specific ID', and 'Add a flight'. To the right, the 'Overview' section shows 'Type: REST API', 'Created By: Max Mule', and 'Published On: Nov 18, 2017'. The 'Edit' button is highlighted in blue.

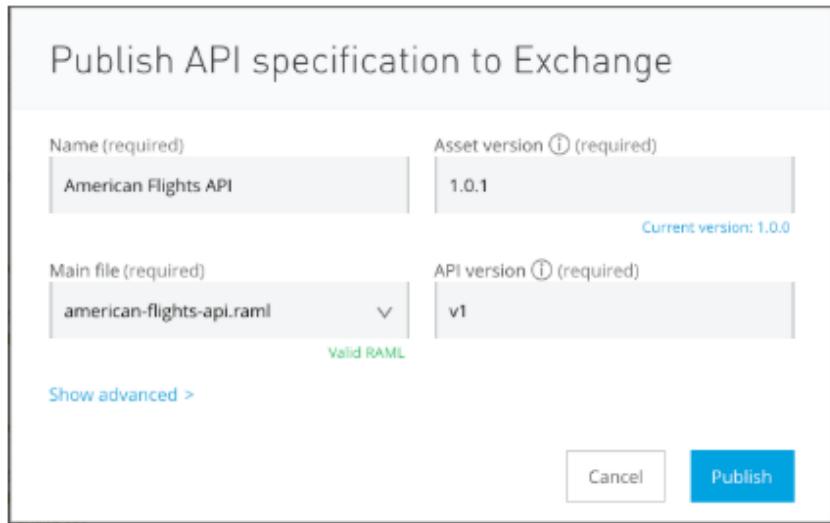
Modify the API and publish the new version to Exchange

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/AmericanFlightExample.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.



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 for managing APIs. The main view is for the "American Flights API".

Left Sidebar: Shows the navigation menu with "Assets List" at the top, followed by "American Flights API". Under "API summary", there are sections for "Types", "Resources", and "Flights". Below these are buttons for "GET", "POST", and a placeholder for "/(ID)". At the bottom is a link for "API instances".

Central Content: The title is "American Flights API" with a green circular icon containing a house symbol. It has a rating of 5 stars and 0 reviews. A "Rate and review" button is present. A description states: "The American Flights API is a system API for operations on the american table in the running database." Below this is a "Supported operations" section with a bulleted list:

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

Right Panel: This panel contains several sections:

- Overview:** Includes fields for "Type" (REST API), "Created By" (Max Mule), "Published On" (Nov 18, 2017), and "Visibility" (Private).
- Asset versions for v1:** A table showing two versions:

Version	Instances
1.0.1	Marketing Service
1.0.0	
- Tags:** A section with a "+ Add a tag" button.
- Dependencies:** A list of two dependencies:
 - Training: American Flights Example (1.0.1, RAML Fragment)
 - Training: American Flight Data Type (1.0.1, RAML Fragment)

Sharing APIs



Sharing APIs



- You can share an API in Exchange with other internal or external users

A screenshot of the MuleSoft Exchange interface. At the top, there's a navigation bar with 'Assets list', 'American Flights API', 'v1', 'Rate and review', 'Share', 'Download', and 'Edit'. Below the navigation is a section titled 'Overview' containing a green circular icon with a white house symbol, the text 'American Flights API v1', and a rating of 0 reviews. There are also 'Share', 'Download', and 'Edit' buttons.

- Share an API within an org through the **private Exchange**
- Share an API with external users in a **public portal** that you create from Exchange

A screenshot of a 'Share American Flights API' dialog box. It includes fields for 'Search for a user', 'Viewer' (set to 'Add'), and a list item 'MM Max Mule (stalions-stgxdr) Admin'. At the bottom are 'Cancel' and 'Share' buttons.

Walkthrough 3-5: Share an API



- 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

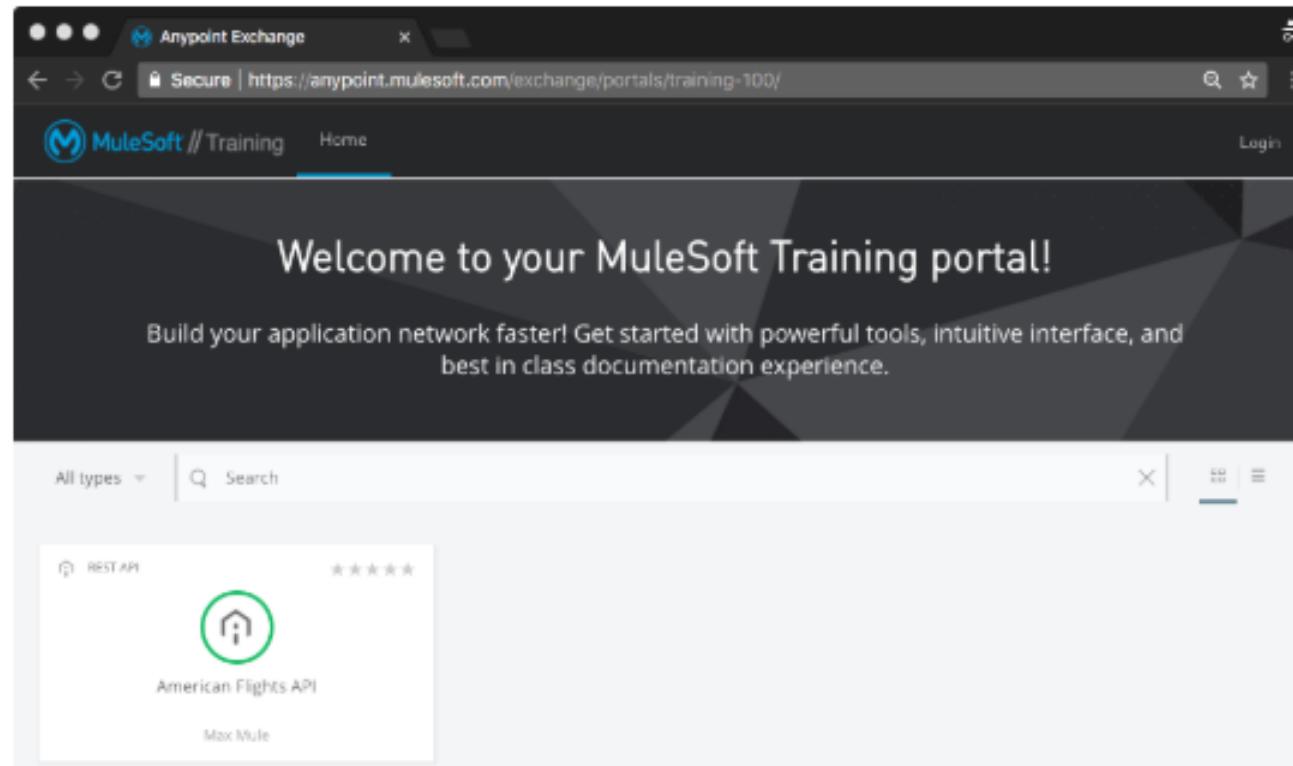
A screenshot of a web browser displaying the MuleSoft Anypoint Exchange portal. The URL in the address bar is https://anypoint.mulesoft.com/exchangeportals/training-100/. The page title is "Anypoint Exchange". The main header says "Welcome to your MuleSoft Training portal!". Below it, a sub-header reads: "Build your application network faster! Get started with powerful tools, intuitive interface, and best in class documentation experience." A search bar at the top has "All types" and a search icon. Below the search bar, there is a card for the "American Flights API", which is categorized as a "REST API". The card features a green circular icon with a white house-like symbol, five stars, and the text "American Flights API" and "Max Mule".

All contents © MuleSoft Inc.

Walkthrough 3-5: Share an API

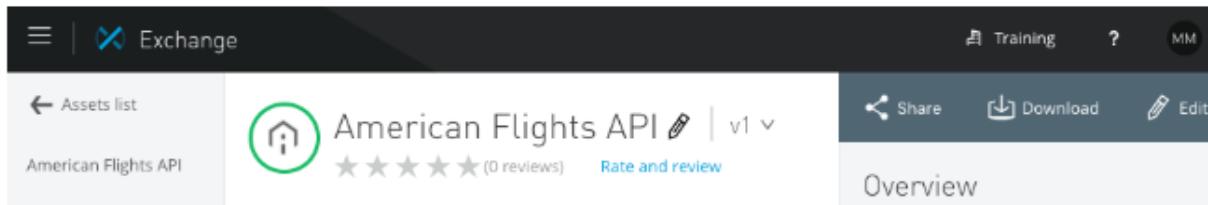
In this walkthrough, you share an API with both internal and external developers to locate, learn about, and try out the API. You will:

- 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.

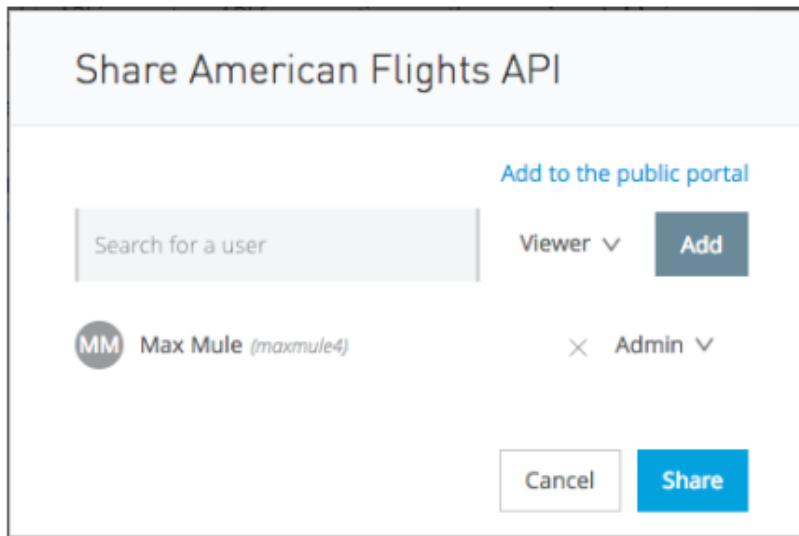


Share the API in the private Exchange with others

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, select the name of your organization.
8. Click your American Flights API.

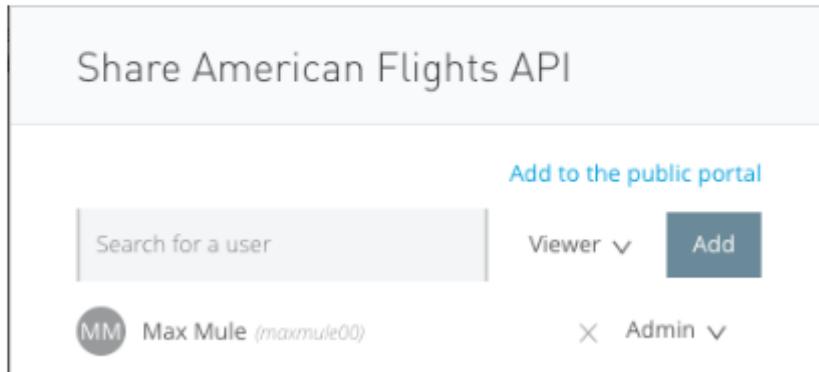
Note: This is how users you share the API with will find the API.

The screenshot shows the MuleSoft Exchange interface. The top navigation bar includes a menu icon, the 'Exchange' logo, 'Training' (which is highlighted in blue), a help icon, and a user icon. The left sidebar has a 'Assets' section with 'Organizations', 'MuleSoft', and 'Training' (highlighted in blue). Below that are 'My applications' and 'Public portal'. The main area is titled 'Assets' with a search bar and a 'New' button. It displays two items: 'American Flights API Connector' (Connector type, 5 stars) and 'American Flights API' (REST API type, 5 stars). Both items have a 'Max Mule' badge below them.

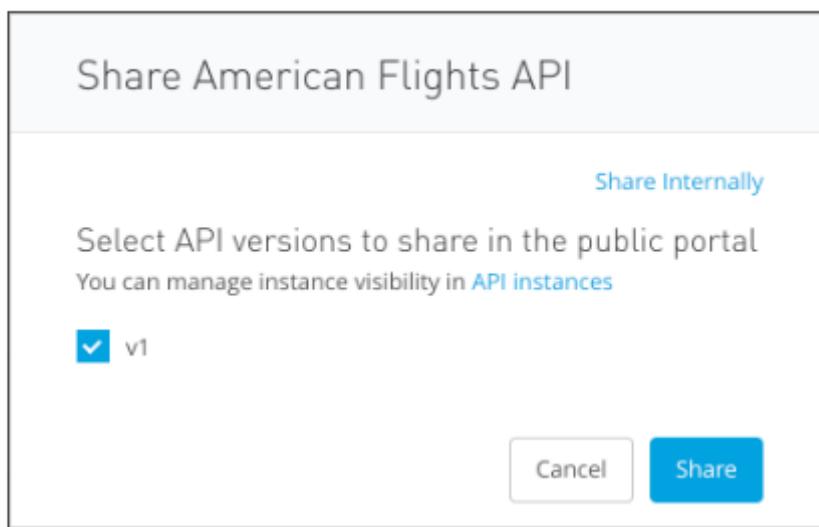
Type	Name	Rating	Author
Connector	American Flights API Connector	★★★★★	Max Mule
REST API	American Flights API	★★★★★	Max Mule

Create a public API portal

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



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



Explore the API in the public portal

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

The screenshot shows the MuleSoft Exchange interface. At the top, there's a navigation bar with icons for Home, Exchange, Training, and Help. The main area is titled "Assets" and has a sub-header "Assets". On the left, a sidebar lists categories: Assets, Organizations, MuleSoft, Training (which is selected and highlighted in blue), My applications, and Public portal. The main content area displays two items: "American Flights API Connector" and "American Flights API". Both items have a "Connector" icon, a five-star rating, and the text "Max Mule". The "American Flights API Connector" item also includes the text "American Flights API". A search bar at the top right says "Search" and a "New" button is also visible.

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

16. Look at the URL for the public portal.

The screenshot shows a web browser window with three tabs open. The active tab is titled 'Secure | https://anypoint.mulesoft.com/exchange/portals/training-100/'. The browser's address bar also displays this URL. The page content is the 'Welcome to your developer portal!' screen of the Anypoint Exchange. It features a search bar at the top with 'All types' and a 'Search' field. Below the search bar, there is a card for the 'American Flights API'. The card includes a green circular icon with a house symbol, the text 'American Flights API', and 'Max Mule' at the bottom. The background has a blue gradient.

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

The screenshot shows a web-based API documentation interface. At the top, there's a dark header with a blue logo, 'Home', 'My applications', and a user icon. Below the header, on the left, is a sidebar with a navigation menu. The menu items include 'Assets list' (with a back arrow), 'American Flights API' (selected and highlighted with a green circle around its icon), 'API summary', 'Types' (with a right arrow), 'Resources' (with a down arrow), '/flights' (with a down arrow), 'GET' (in a green button), 'POST' (in a purple button), and '/(ID)' (with a right arrow). On the right side, the main content area is titled 'American Flights API | v1'. It features a circular icon with a house symbol, a rating of 0 reviews, and a 'Rate and review' link. Below the title, a description states: 'The American Flights API is a system API for operations on the american table in the training database.' Under the heading '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'. At the bottom right of the main content area is a button labeled 'API Spec' with a download icon.

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

19. Review the body and response information.

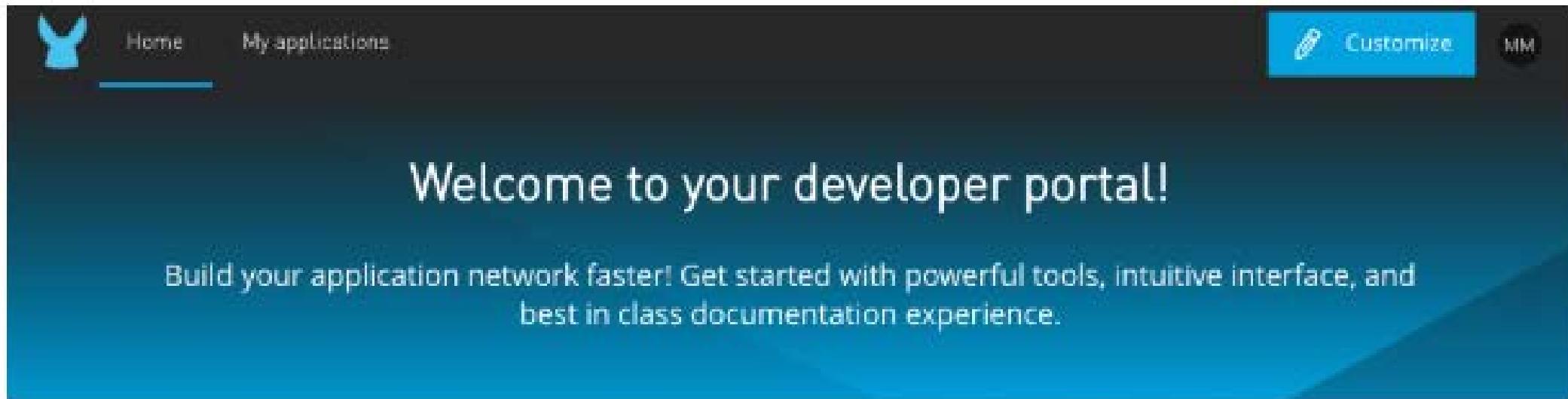
20. In the API console, click Send; you should get a 201 response with the example data returned from the mocking service.

The screenshot shows the MuleSoft Anypoint Platform API Console interface. On the left, there's a sidebar with a navigation tree for the "American Flights API". The "flights" resource is selected, and under it, the "POST" method is highlighted. The main panel displays the "American Flights API | v1" endpoint. A "POST" button is prominently displayed at the top right of the main content area. Below the button, the URL is shown as "https://mocksvc-proxy.anypoint.mulesoft.com/". The "Parameters" tab is selected, and a note states: "This endpoint doesn't require to declare query or URL parameters." At the bottom right of the main panel, a "Send" button is visible. The response section shows a green "201 Created" status box with a timestamp of "271.27 ms" and a "Details" link. Below the status box, there are icons for copy, download, and refresh. The response body is displayed as a JSON object:

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

Customize the public portal

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



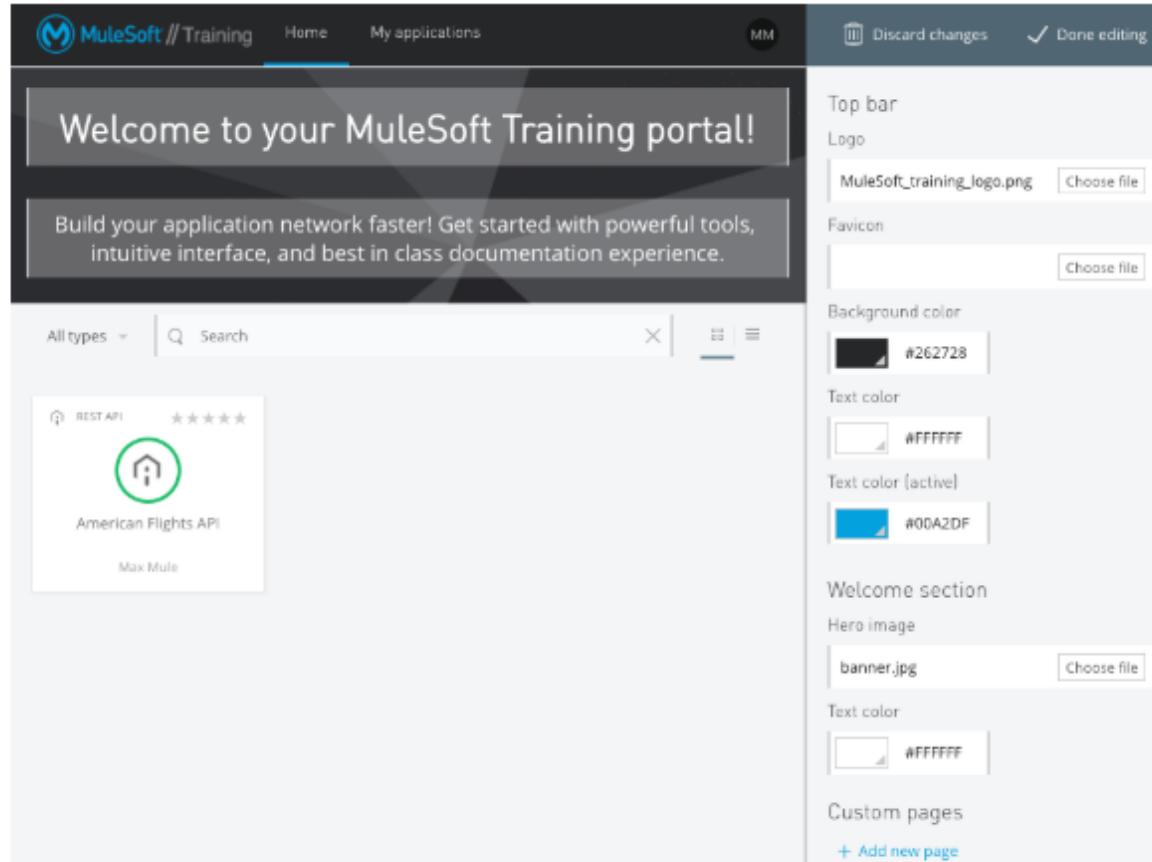
23. Change the text to Welcome to your MuleSoft Training portal!

The screenshot shows the MuleSoft Anypoint Platform interface. On the left, the main application area displays a landing page with the title "Welcome to your MuleSoft Training portal!" and a subtitle: "Build your application network faster! Get started with powerful tools, intuitive interface, and best in class documentation experience." Below this, there's a search bar and a card for the "American Flights API". The "REST API" icon has a green circle around it, and the card includes a rating of 5 stars and the text "Mule Mule". On the right, a sidebar titled "Edit theme" is open, showing various customization options:

- Top bar**: Includes fields for "Logo" and "Favicon", each with 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**: Includes a "Hero image" field containing "image-default.png" and a "Choose file" button, and a "Text color" field set to #FFFFFF.
- Custom pages**: A link to "+ Add new page".

At the top right of the sidebar, there are "Discard changes" and "Done editing" buttons.

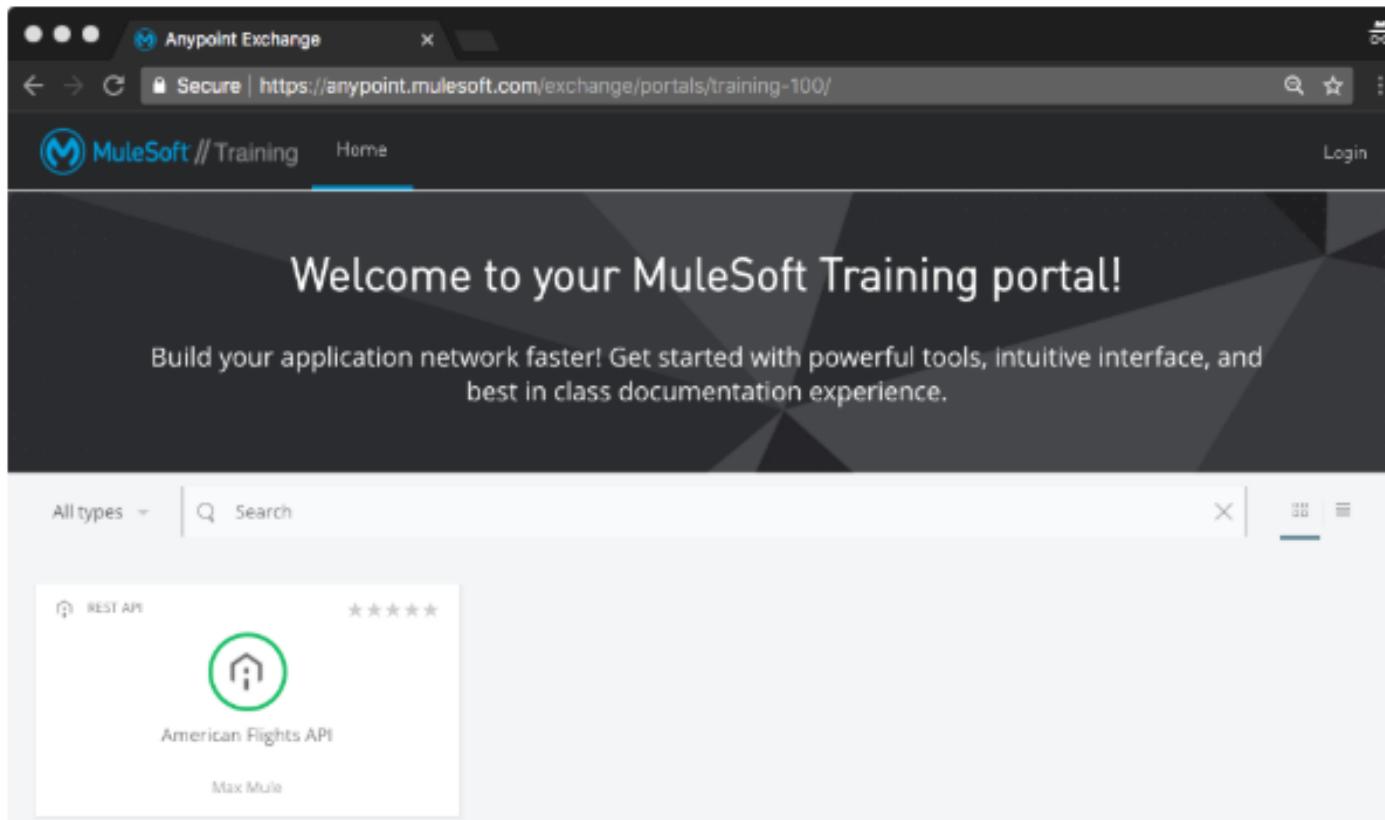
24. In the logo field, click Choose file.
25. In the file browser dialog box, navigate to the student files and locate the MuleSoft_training_logo.png file in the resources folder and click Open.
26. Locate the new logo in the preview.
27. In the hero image field, click Choose file.
28. In the file browser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



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

Explore the public portal as an external developer

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



36. Click the American Flights API.
37. Explore the API portal.

38. 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 MuleSoft Anypoint Platform interface. On the left, the navigation sidebar includes 'Assets list', 'American Flights API' (selected), 'API summary', 'Types', 'Resources', '/flights' (selected), and 'API instances'. Under '/flights', there are 'GET' and 'POST' buttons. The main content area displays the 'American Flights API | v1' details, including a house icon, a 5-star rating (0 reviews), and the endpoint '/flights : get'. The 'Request' section shows a GET request to '/flights'. The 'Parameters' section has a table:

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

The 'Response' section indicates a 200 OK status with a response type of 'application/json'. On the right, a detailed view of the 'GET /flights' endpoint shows the 'Mocking Service' tab selected, displaying the URL <https://mocksvc-proxy.anypoint.mulesoft.com/>. The 'Parameters' tab is active, showing 'Query parameters' with a checkbox for 'Show optional parameters'. A 'Send' button is present. Below, a '200 OK' response is shown with a duration of 260.37 ms and a 'Details' link. The response body is an array of two flight objects:

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


Summary



- **RAML** is a non-proprietary, standards-based API description language spec that is simple, succinct, and intuitive to use
 - Data structure hierarchy is specified by indentation, not markup characters
- Use **API designer** to write API specifications with RAML
- Documentation is auto-generated from a RAML file and displayed in an **API console**
- A **mocking service** can be used in API console to test an API and return the example data specified in RAML

- Make an **API discoverable** by adding it to your **private Exchange**
- **API portals** are automatically created for the APIs with
 - Auto-generated **API documentation**
 - An **API console** that provides a way to consume and test an API
 - An **automatically generated API endpoint** that uses a **mocking service** to allow the API to be tested without having to implement it
- API portals can be shared with both internal and external users
 - Selectively share APIs in your org's **private Exchange** with other internal developers
 - Share APIs with external developers by creating and customizing a **public portal** from Exchange and specifying what APIs you would like to include in it

- RAML definitions can be a lot more complex and sophisticated than what we built here
- Training: training.mulesoft.com
 - *Anypoint Platform: API Design*
- Website: raml.org
 - Documentation
 - Tutorials
 - Full spec
 - Resources



DIY Exercise 3-1: Create an API specification with RAML

Time estimate: 1 hour

Objectives

In this exercise, you create an API specification using RAML. You will:

- Appropriately specify GET and POST methods.
- Appropriately define URI parameters, query parameters, and headers.
- Restrict possible values using an enumeration.

Scenario

Your company needs to expose a customer accounts database as a System API for the rest of the organization. The first step is to create the RAML specification and post it to your company's private Anypoint Exchange so all stakeholders can review and provide feedback. This RAML specification needs to separate out data type definitions from the main API RAML file into separate reusable RAML files.

Create the API specification

In Design Center, create a RAML specification called Accounts API with the following requirements:

- The API has a resource called accounts.
- All client requests must include the required header Requester-ID. This header helps identify the person in the organization making account requests.
- The /accounts resource has a GET method with a required queryParameter named type, where type must be either personal or business. The type query parameter is used to return records that match that account's type.
- The GET method for the /accounts resource has two optional string queryParameters named name and country. The name query parameter value is used to filter and return all account records for a particular account owner. The country query parameter value is used to filter and return records matching the account owner's country of residence.
- The GET method's response data is an array of Account objects. To represent the response data, define and use an Account data type that contain the following fields: id, firstName, lastName, address, postal, country, miles, creationDate, and type. The creationDate field is of type datetime and miles is of type integer.
- The API defines and uses a data type for the Account object that is stored in a separate RAML file.
- The Account data type definition has a corresponding example that is included in the main API RAML file or in the Account data type RAML file.
- The /accounts resource has a POST method that accepts an array of Account objects (with no id and no creationDate field) in JSON format and returns a JSON message: {"message": "Account created (but not really)"}. This Account object is represented as a new data type that does not have id or creationDate fields, makes the type field required, and has a corresponding example.
- Each method should have a description.
- Each method should have a custom 400 error status message example in JSON format.

Publish the API to Anypoint Exchange

Publish the API to Anypoint Exchange and then test the mocked endpoint using the API Console and create a public portal for the API.

Verify your solution

Load the solution [/files/module03/accounts-mod03-api-solution.zip](#) (in the MUFundamentals_DIYexercises.zip that you can download from the Course Resources) and compare your solution.