



Module 3: Designing APIs



Spec driven development



- 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

The diagram illustrates the API development cycle, represented by a circular process with four main phases: Design, Simulate, Validate, and Feedback. The cycle is represented by a large blue arrow pointing clockwise. To the right of the arrow is the text "API Spec (RAML)". Below the cycle are three screenshots of MuleSoft tools:

- Exchange:** Shows the Anypoint Exchange interface with various assets and applications listed.
- API portal:** Shows the API portal interface for the American Flights API, displaying endpoints like /flights and /flights/{id} with their respective parameters and descriptions.
- API designer:** Shows the API designer interface for the American Flights API, allowing for detailed configuration of endpoints, security, and responses.

All contents © MuleSoft Inc.

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

The slide lists the following objectives:

- 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

All contents © MuleSoft Inc.

Reviewing the options for defining APIs



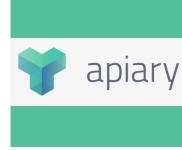
Approaches to API design



Hand coding



API Blueprint



OpenAPI Spec



RAML



Introducing RAML



RAML: RESTful API Modeling Language



- **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
<http://raml.org>

RAML syntax



- 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      /{ID}:
9          get:
10         delete:
11
12     put:
13         responses:
14             200:
15                 body:
16                     application/json:

```

Notice the indentation used to specify to what each line applies

All contents © MuleSoft Inc.

9

Defining resources and methods



- 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:

```

All contents © MuleSoft Inc.

Using API designer to define APIs with RAML



API designer

MuleSoft

American Flights API | master

Saved a minute ago

Mocking service: —

API summary

Types

Resources

/flights

GET

POST

/ID}

GET

DELETE

PUT

Editor

File browser

Types and Traits Others Docs

is type description

uriParameters displayName Shelf

1 #%RAML 1.0
2 version: v1
3 title: American Flights API
4
5 types:
6 | AmericanFlight: !include
exchange_modules/68ef9520-24e9-4cf2-b2f5-620025690913/training-american-
flight-data-type/1.0.1/AmericanFlightDataType.raml
7
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[]

american-flights-api.raml

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 MuleSoft API Designer interface. On the left, there's a file tree with a single file named "american-flights-api.raml". The main area displays the RAML code for the American Flights API:

```
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
14    post:
15      /{ID}:
16        get:
```

Below the code, there are sections for "Parameters", "queryParameters", and "headers". To the right, the "API summary" pane shows the "Resources" section expanded, revealing the "/flights" resource with its "GET" and "POST" methods. Further down, the "/{ID}" resource is shown with its "GET" method.

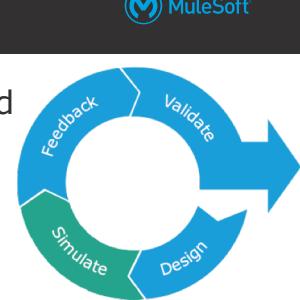
13

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



All contents © MuleSoft Inc.

15

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

All contents © MuleSoft Inc.

16

```

1  #!/RAML 1.0
2  baseUri:
3    https://mocksvc.mulesoft.com/mocks/7a9130df-
4    -b911-d424-466d-0e906d8688b8 #
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  /{ID}:
18    get:

```

Using RAML to define specifications for requests and responses



Defining method `response` details with RAML



- 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
```

Defining method `request` details with RAML



- 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

```

All contents © MuleSoft Inc.

19

Modularizing APIs



- 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

All contents © MuleSoft Inc.

20

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

All contents © MuleSoft Inc.

21

```

American Flights API | 99 master
+-- Files
  +-- examples
    +-- AmericanFlightExample.raml
    +-- AmericanFlightNotExample.raml
  +-- exchange_modules
    +-- 68e95920-24e9-4cf2-b2f5-620025690913
      +-- training-american-flight-data-type
        +-- 1.0.1
          +-- AmericanFlightDataType.raml
        +-- training-american-flights-example
          +-- 1.0.1
            +-- AmericanFlightsExample.raml
  +-- american-flights-api.raml

Saved 5 minutes ago
Mocking service
200 OK 125.69 ms
[{"ID": 1, "code": "BB38ad", "price": 400, "departureDate": "2017/07/26", "origin": "sfo", "destination": "aro", "emptySeats": 0, "plane": {"type": "Boeing 737", "totalSeats": 150}}, {"ID": 2, "code": "BB451f", "price": 540.99, "departureDate": "2017/07/27", "origin": "sfo", "destination": "osp", "emptySeats": 54, "plane": {"type": "Boeing 737", "totalSeats": 150}}]

```

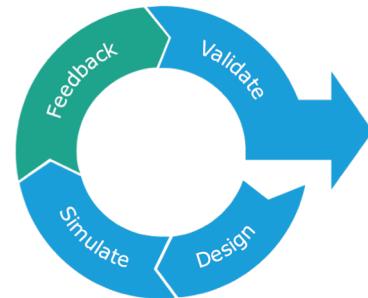
Engaging with users



Engaging users during the API design phase



- 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



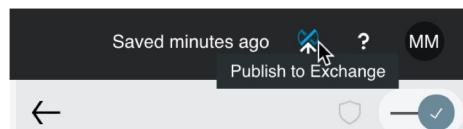
All contents © MuleSoft Inc.

23

Publishing RAML APIs to Anypoint Exchange



- 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



All contents © MuleSoft Inc.

24

Walkthrough 3-4: Add an API to Anypoint Exchange

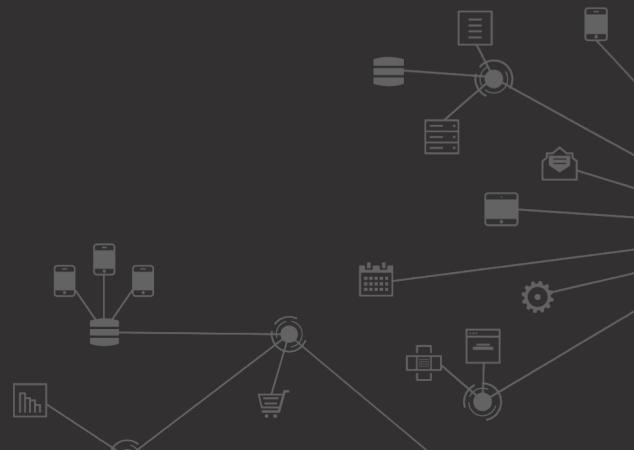


- Publish an API to Exchange from API designer
- Review an auto-generated API portal in Exchange and test the API
- Add information about an API to its API portal
- Create and publish a new API version to Exchange

The screenshot shows the Anypoint Exchange interface for the 'American Flights API' version 1.0.1. The left sidebar includes sections for 'Assets list', 'API summary' (with 'Types', 'Resources', and 'Flights' expanded), and 'API instances'. The main content area displays the API's title, a green icon, a star rating of 4 stars, and a link to 'Rate and review'. Below this is a section titled 'Supported operations' listing: '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 an 'Overview' panel with details like 'Type: REST API', 'Created By: Max Mule', 'Published On: Nov 18, 2017', and 'Visibility: Private'. A table titled 'Asset versions for v1' shows two rows: '1.0.1' (Mocking Service) and '1.0.0'. At the bottom, there's a 'Tags' section with a '+ Add a tag' button.

25

Sharing APIs



Sharing APIs

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

The screenshot shows the Exchange interface. At the top, there's a navigation bar with 'Assets list' and a search bar. Below it, the 'American Flights API' is listed with a green icon, version v1, and a rating of 5 stars (0 reviews). To the right, there are 'Share', 'Download', and 'Edit' buttons, and a 'Training' link. A modal window titled 'Share American Flights API' is open, showing a search bar for users, a dropdown for roles ('Viewer' selected), and a list of users ('Max Mule (stallions-stgdr)') with a role of 'Admin'. There are 'Add' and 'Cancel' buttons at the bottom.

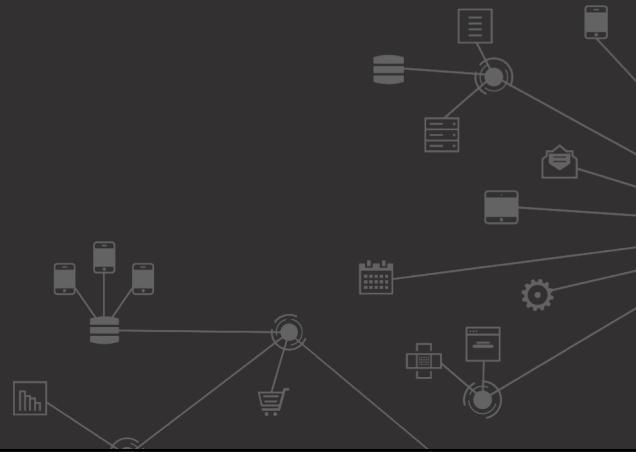
All contents © MuleSoft Inc.

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

The screenshot shows the MuleSoft Training portal. It has a header with the MuleSoft logo and a 'Training' tab. The main content area says 'Welcome to your MuleSoft Training portal!' and '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 list of APIs. One API, 'American Flights API' by 'Max Mule', is listed with a green icon and a 5-star rating. The footer contains the copyright notice 'All contents © MuleSoft Inc.' and the page number '28'.

Summary



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

Summary



- 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 resources



- 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

