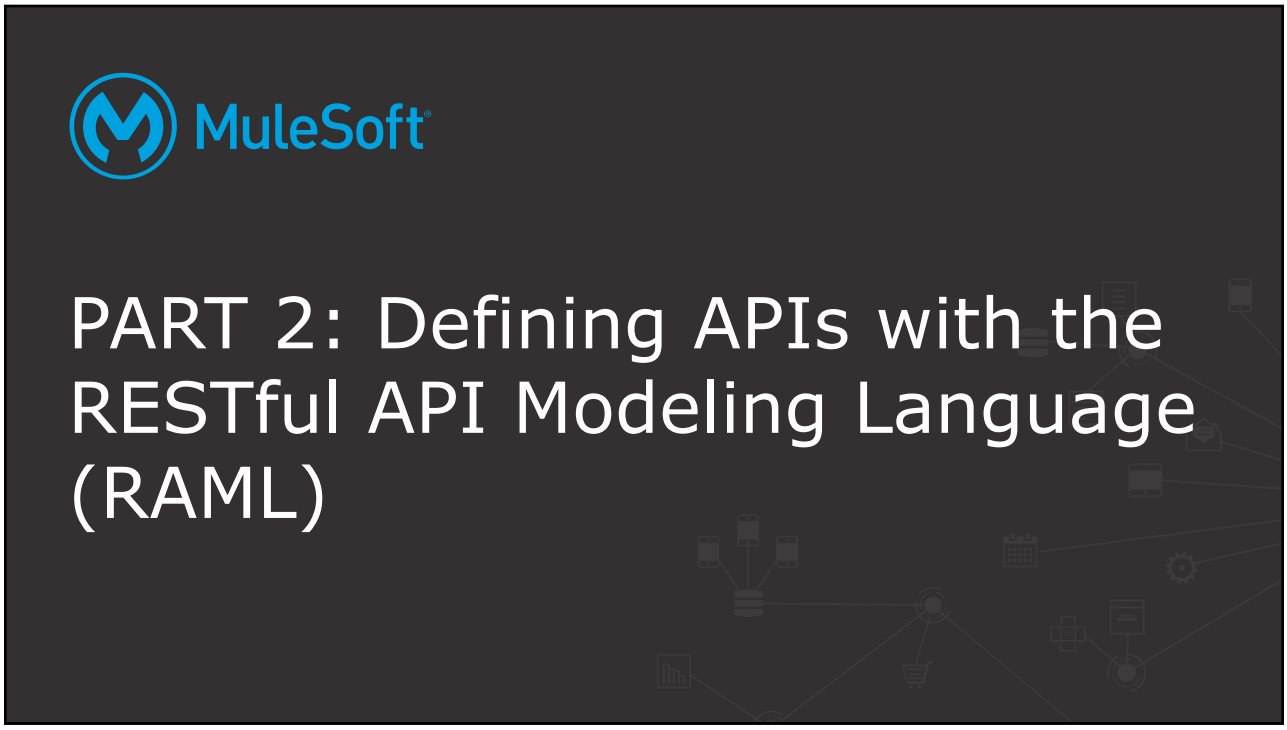


PART 2: Defining APIs with the RESTful API Modeling Language (RAML)

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

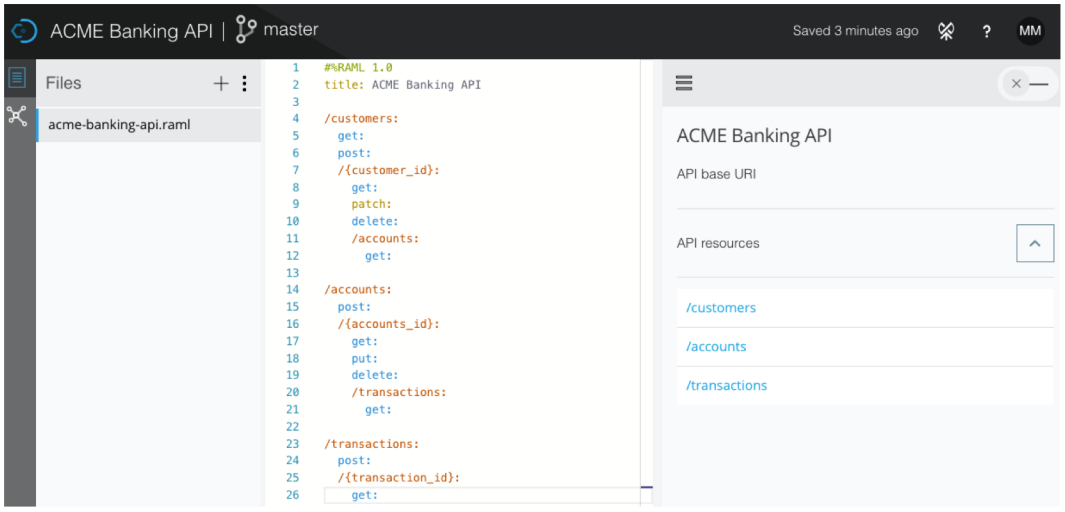
- Create API definitions with RAML 1.0
- Add documentation to RAML API definitions
- Make APIs discoverable through Anypoint Exchange
- Test APIs through the API console
- Use patterns to refactor and modularize API definitions
- Specify security schemes to secure resources in APIs
- Add state specific responses to promote hypermedia
- Learn when and how to version APIs

MuleSoft

Module 4: Defining API Resources and Methods

Goal

MuleSoft



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

- Select a specification language to create a standardized API definition
- Use Anypoint Platform Design Center to create API definitions with RAML 1.0
- Define resources and methods in RAML API definitions

Selecting a powerful specification language for defining APIs

To succeed at Spec-Driven Development, the API definition should be

MuleSoft

- Standardized
  - Creating a standard spec helps with portability among developers
  - Maintains long-term consistency
- Consistent
  - Help make sure resources and methods within and across APIs are formatted similarly
  - Ensures code reuse when possible by using pattern-driven design
- Concrete
  - An API definition is the base of the application being built
  - A solid blueprint covering all aspects of the application helps developers who are coding the application

To succeed at Spec-Driven Development, the API definition should be

MuleSoft

- Tested
  - Testing the API definition takes place in the design phase in order to build a reliable application
  - Testing is carried out by internal as well as external consumers to ensure all the needs are met
- Immutable
  - API definition is the ultimate authority and the application should not deviate from this blueprint at the time of development
  - The API design has a lifecycle of its own and has been thoroughly tested with real API consumers to ensure longevity
- Persistent
  - In the event of making changes to the application, the API definition must carefully go through the design lifecycle and re-evaluated before updating the code/application

Overview of API specification languages

MuleSoft

- **Dynamic discovery and interaction** with the endpoints is very critical in spec driven development
  - Rather than having specifications serve as a static documentation
- Major description languages like WSDL and WADL were not preferred for describing REST APIS
  - Poor human readability

WSDL/WADL

API Blueprint

apiary

OpenAPI Spec

OPEN {API} INITIATIVE

RAML

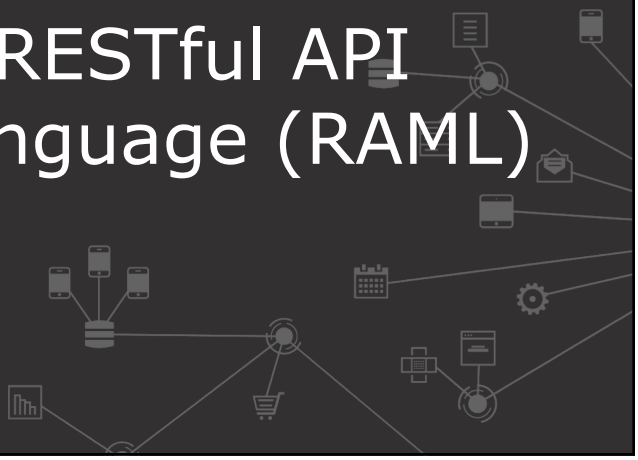
RAML

Overview of API specification languages

MuleSoft

- OpenAPI Specification (OAS) previously known as the Swagger specification
  - Generates documentation of REST API methods, parameters and models
  - Can be JSON/YAML based
    - Creates client and server stubs by parsing the API definition
- Apiary's API Blueprint
  - Based on Markdown language
    - Excellent document generator
  - API structure blends in with the documentation – lacks tooling and language support
- Several other description languages like I/O Docs(Mashery), Open Data Protocol etc.
  - Interactive documentation systems that are not very human-friendly

Introducing the RESTful API Modeling API Language (RAML)



RAML: RESTful API Modeling Language

MuleSoft

- A non-proprietary, vendor-neutral open spec
- A simple, structured, and succinct way of describing RESTful APIs
  - The resources
  - The HTTP methods that can be used for each resource
  - Any method request parameters and their data type
  - The response types and sample responses
  - And much more!

RAML

http://raml.org

- Developed to help out the current API ecosystem by encouraging Spec-Driven Development
  - Encourages reuse, enables discovery and pattern-sharing, and aims for merit-based emergence of best practices

More about RAML

MuleSoft

- Created by a group of members from MuleSoft, PayPal, Intuit, Cisco and more
- RAML is a blueprint to model an API
  - Helps manage the entire API lifecycle from design to testing and sharing
  - Machine readable and human friendly too (since it is defined in YAML)
- Two versions available
  - RAML 0.8
  - RAML 1.0
- RAML joined the OpenAPI Initiative to support working with OAS
  - This enables interoperability by providing RAML modeling atop of the OpenAPI Specification (OAS)
  - Provide common programmatic capabilities and facilitate collaboration

RAML 0.8 vs RAML 1.0

MuleSoft

- RAML 1.0 empowers developers by helping create modular, reusable API specifications
- It includes new features such as
  - Libraries
  - Datatypes (instead of schemas in RAML 0.8)
  - Overlays and extensions
  - Annotations
- Migration information from RAML 0.8 to 1.0 can be found here
  - https://docs.mulesoft.com/release-notes/raml-1-early-access-support

RAML API definitions are used to …

MuleSoft

- Auto-generate API documentation
  - For an API console in an Exchange portal (interactive do
  - Using hundreds of other tools: http://raml.org/developers/document-your-api
- Generate mocked endpoints so an API can be interactively tested before it is built
  - In an API console
  - Using popular testing tools: http://raml.org/developers/test-your-api
- Auto-generate an implementation interface with sever-side generators in Mule, using APIkit
  - In NodeJS, Java, .NET, Python…: http://raml.org/developers/build-your-api
- Enable auto-discovery of endpoints for users in tools like Studio

Important terminology in RAML 1.0

MuleSoft

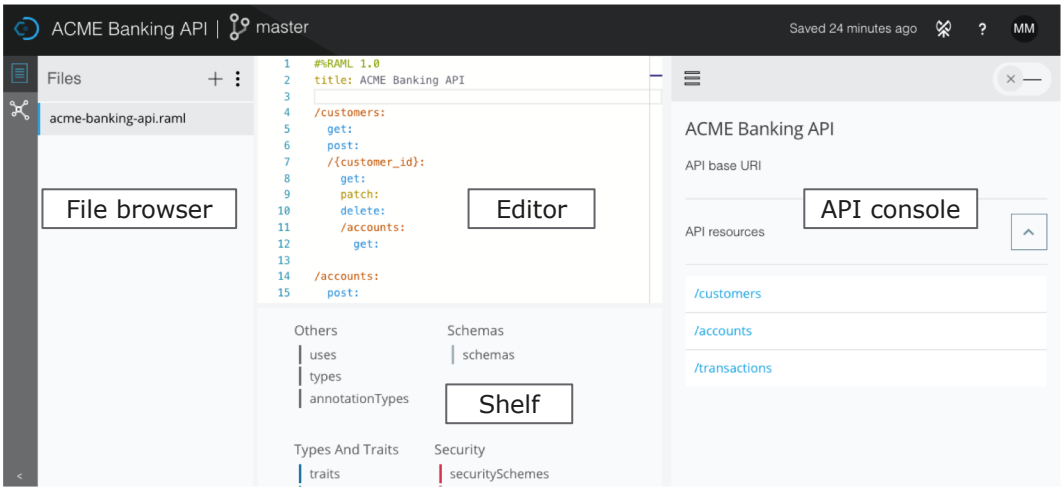
- YAML and JSON based modeling language
  - YAML initially stood for Yet Another Markup Language but later changed to YAML Ain't Markup Language – to emphasize that it is more data-oriented
- Consists of nodes – keys accepting values in the form of
  - Map (multiple key-value pairs)
  - Scalar valued (single key-value pair like *description: This is an example)*
  - Sequence (array of values for a key)
    - For example:
      - *is: [cacheable, searchable]*
      - *is:*
        - *cacheable*
        - *searchable*
- Indentation is important to represent hierarchy in the lines of data
  - Improper indentation results in erroneous code

Creating RAML API definitions in Anypoint Platform

Using API designer for creating API definitions

MuleSoft

- API designer is a part of the Design Center entitlement



Walkthrough 4-1: Create an API and define resources in RAML 1.0

MuleSoft

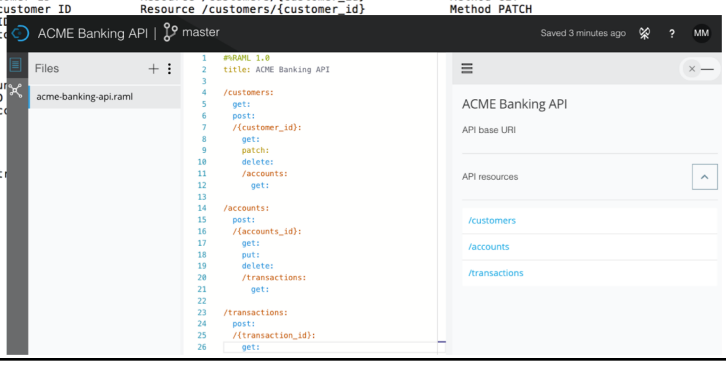
- Create an API in Anypoint Platform Design Center
- Define resources and nested resources identified for the API
- Define HTTP methods for the resources

```
Translating categories and actions into resources and methods –

CUSTOMERS: Resource /customers                    Resource /customers                    Method GET
i.  Get list of all customers in the bank         Resource /customers                    Method POST
ii. Register a new customer                       Resource /customers/{customer_id}      Method GET
iii.Get customer information for a specific customer ID  Resource /customers/{customer_id}  Method PATCH
iv. Update customer information for a specific customer ID
v.  Delete a customer with a specific customer ID
vi. Get list of all accounts for a specific custo

ACCOUNTS: Resource /accounts
i.  Create a new account
ii. Get account information for a specific accoun
iii.Delete an account with a specific account ID
iv. Update account information for a specific acc
ii. Get transactions for a specific account ID

TRANSACTIONS: Resource /transactions
i.  Create a new transaction
iii.Get transaction information for a specific tr
```



Passing data to methods

Passing data into methods

MuleSoft

- URI parameters
  - Represented as a nested resource in curly braces
  - Example
    - /users/{userID}, the value of {userID} is dynamic i.e. /users/21gnoe9/
  - Best practice
    - Use for unique identifiers, because they affect a subtree of resources in the URL (if a subtree exists)

Passing data into methods

MuleSoft

- Query parameters
  - Are an extension of the resource, represented as a key-value pair after a question mark at the end of the URI
  - Example
    - /users?active=true
  - Best practice
    - Use for a subset of the resource or for adding a filter property for the data returned by the resource - not to obtain the data itself
- Headers
  - Covered in Module 5

Summary

Summary

MuleSoft

- RAML stands for RESTful API Modeling Language
  - It 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
- Anypoint Platform Design Center - API designer can be used to write API definitions with RAML
- RAML can model API specification content including
  - Resources
  - Methods
  - Security schemes
  - Annotations
  - Overlays and extensions