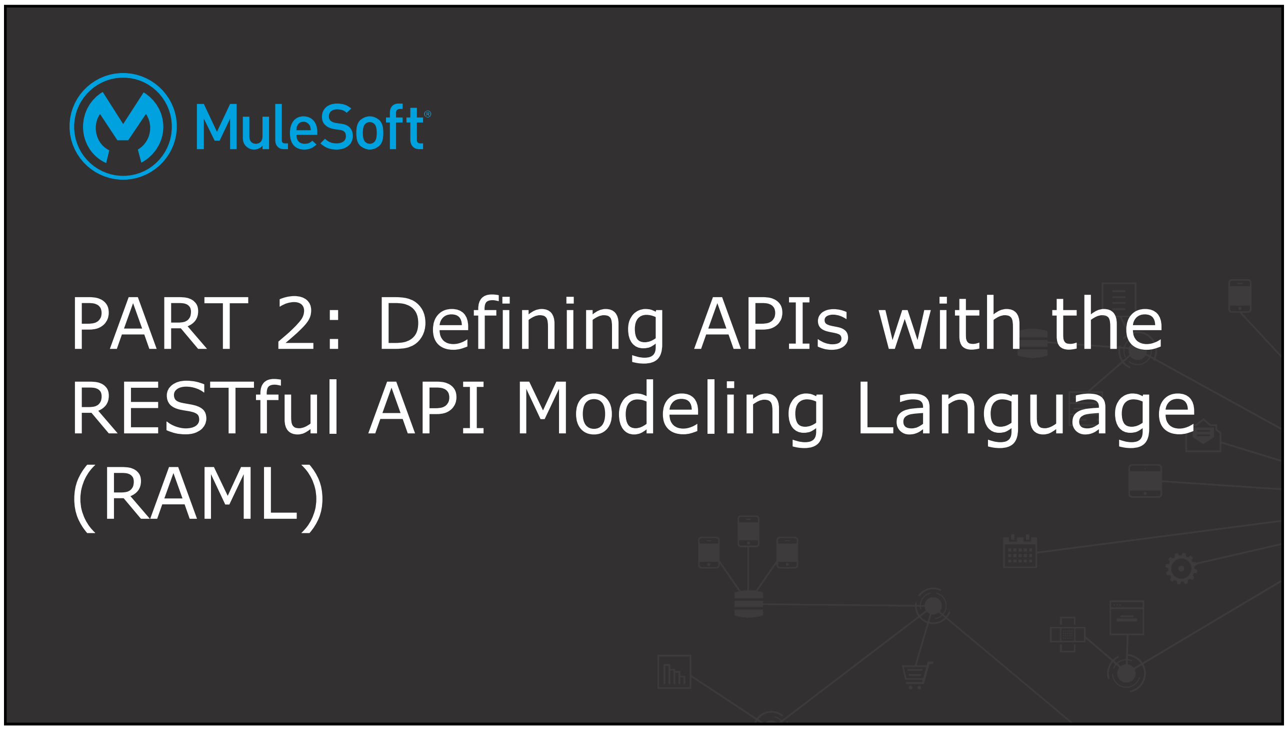
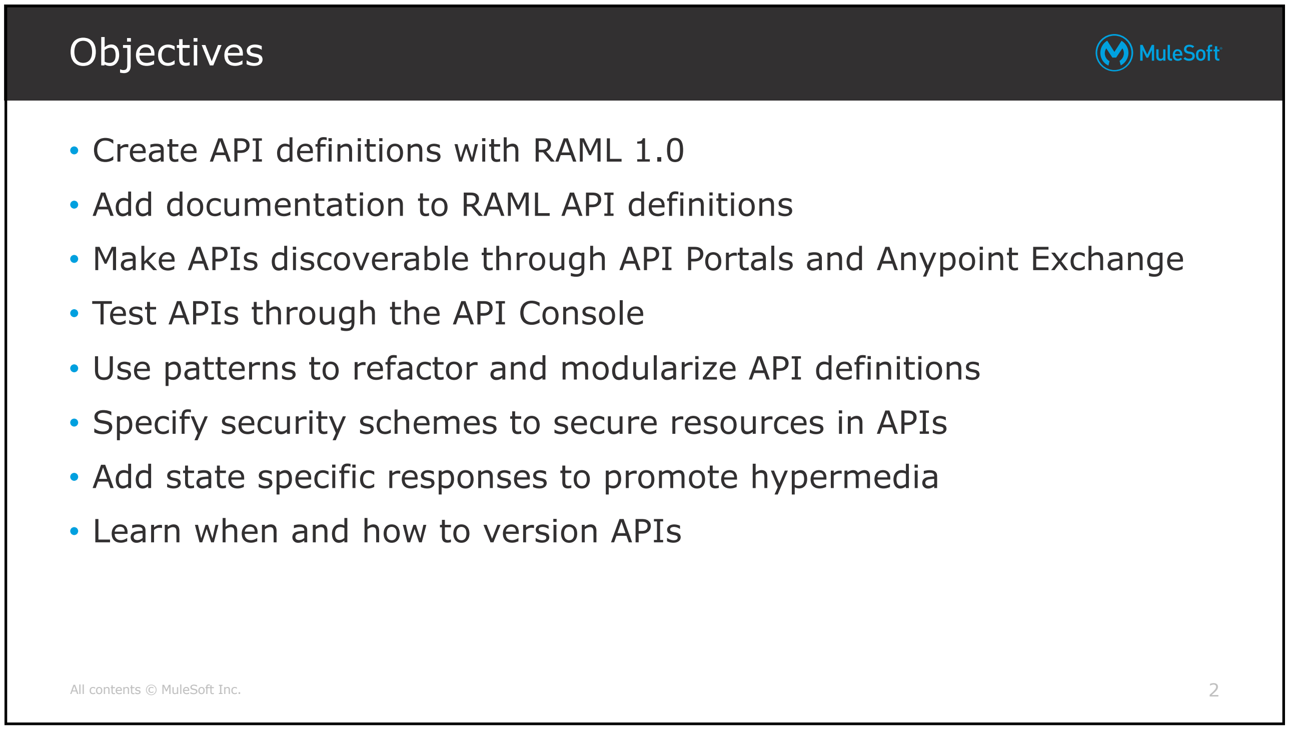


PART 2: Defining APIs with the RESTful API Modeling Language (RAML)



Objectives

- Create API definitions with RAML 1.0
- Add documentation to RAML API definitions
- Make APIs discoverable through API Portals and 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

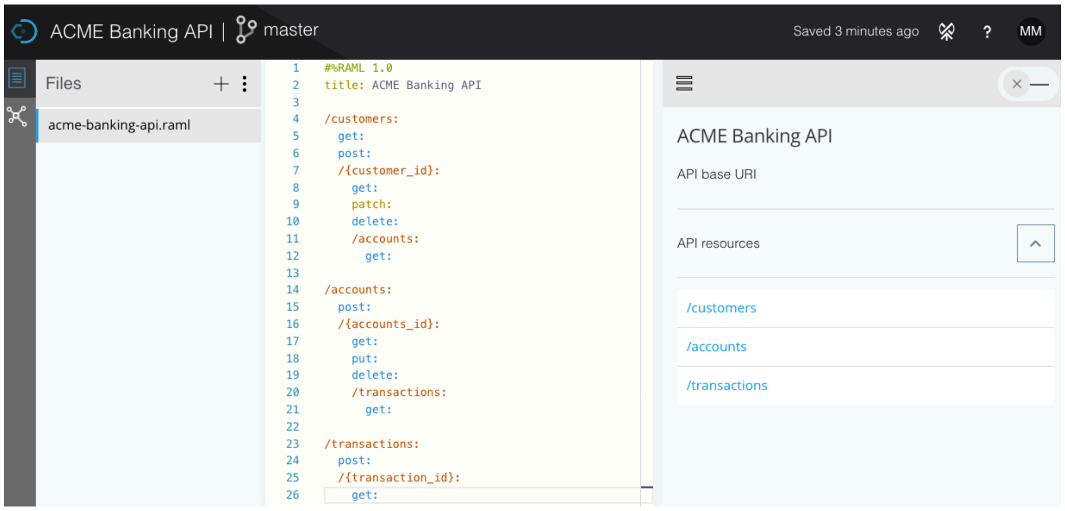
2

MuleSoft

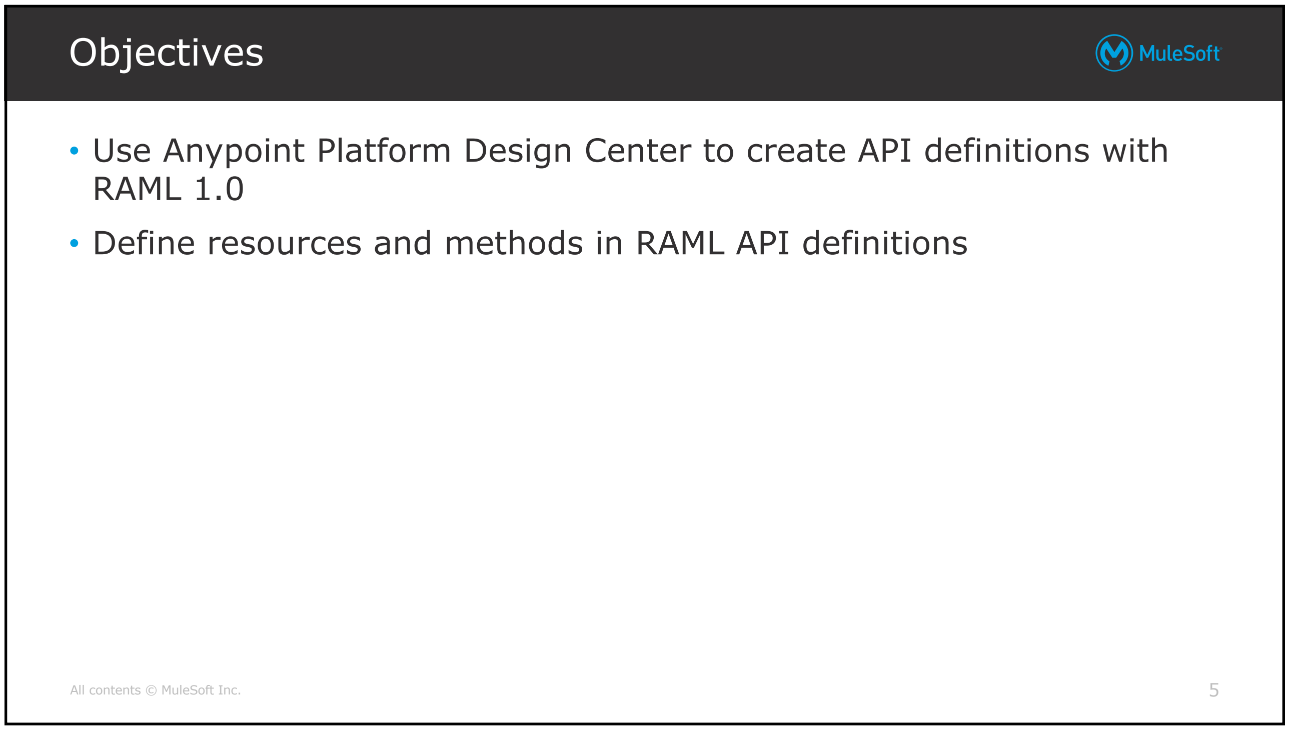
Module 4: Defining API Resources and Methods

Goal

MuleSoft



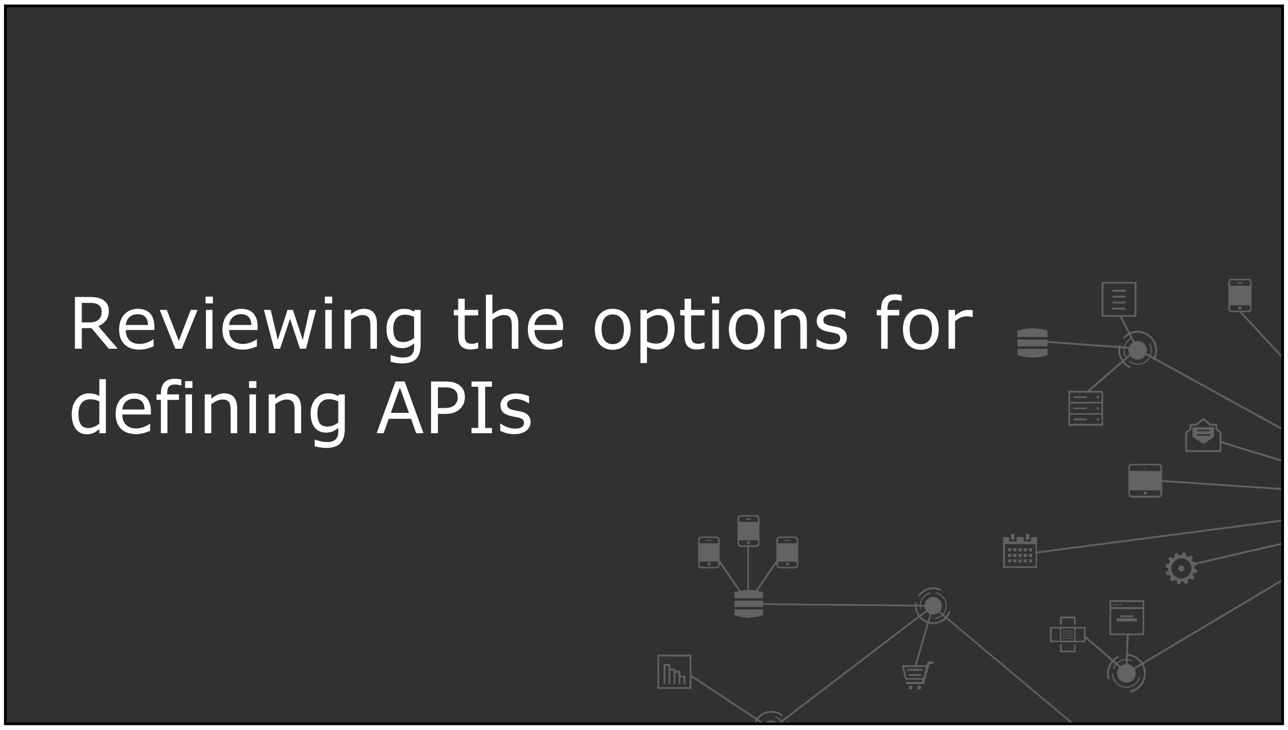
```
#%RAML 1.0
title: ACME Banking API

/customers:
  get:
  post:
  /{customer_id}:
    get:
    patch:
    delete:
    /accounts:
      get:

/accounts:
  post:
  /{accounts_id}:
    get:
    put:
    delete:
    /transactions:
      get:

/transactions:
  post:
  /{transaction_id}:
    get:
```



Objectives

MuleSoft

- Use Anypoint Platform Design Center to create API definitions with RAML 1.0
- Define resources and methods in RAML API definitions



Reviewing the options for defining APIs

Approaches to API design

MuleSoft

- Major description languages like WSDL and WADL were not preferred for describing REST APIS
  - Poor human readability
- RESTful APIs require dynamic discovery and interaction with the endpoints than just serve as a static documentation

Hand coding WSDL/WADL

API Blueprint

apiary

OpenAPI Spec

OPEN {API} INITIATIVE

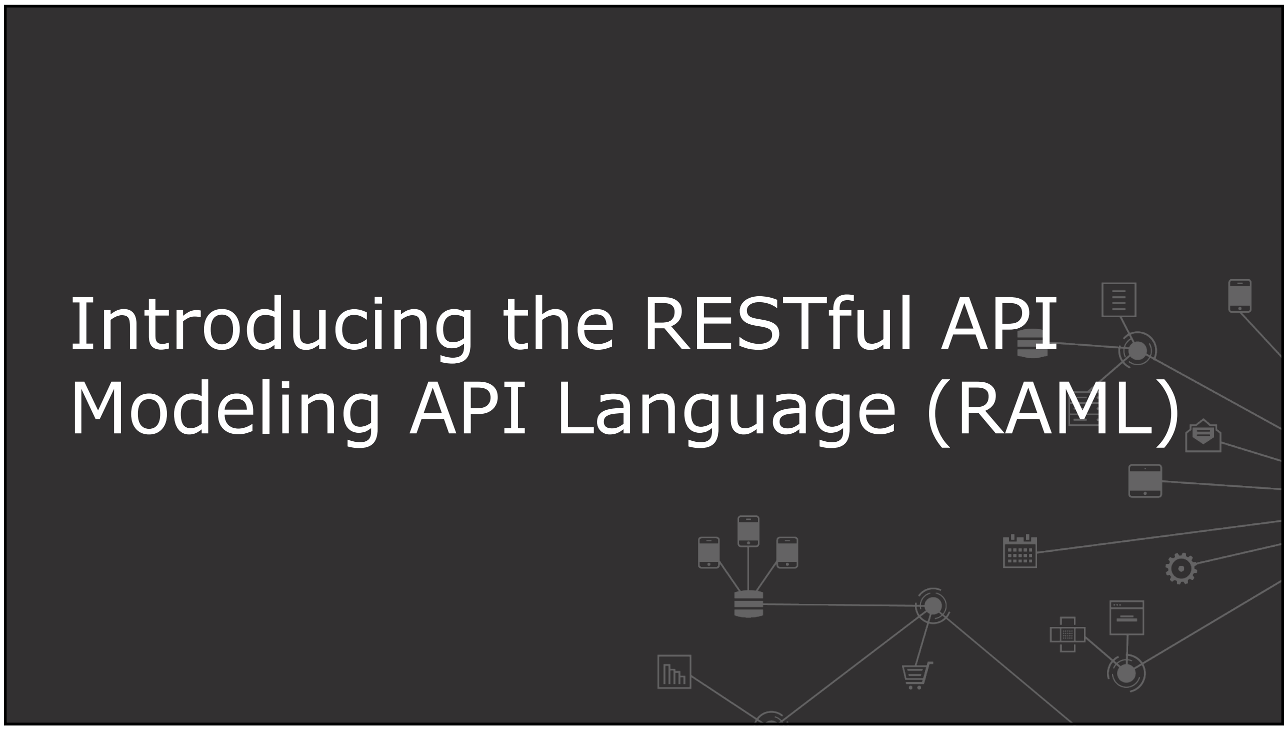
RAML

RAML

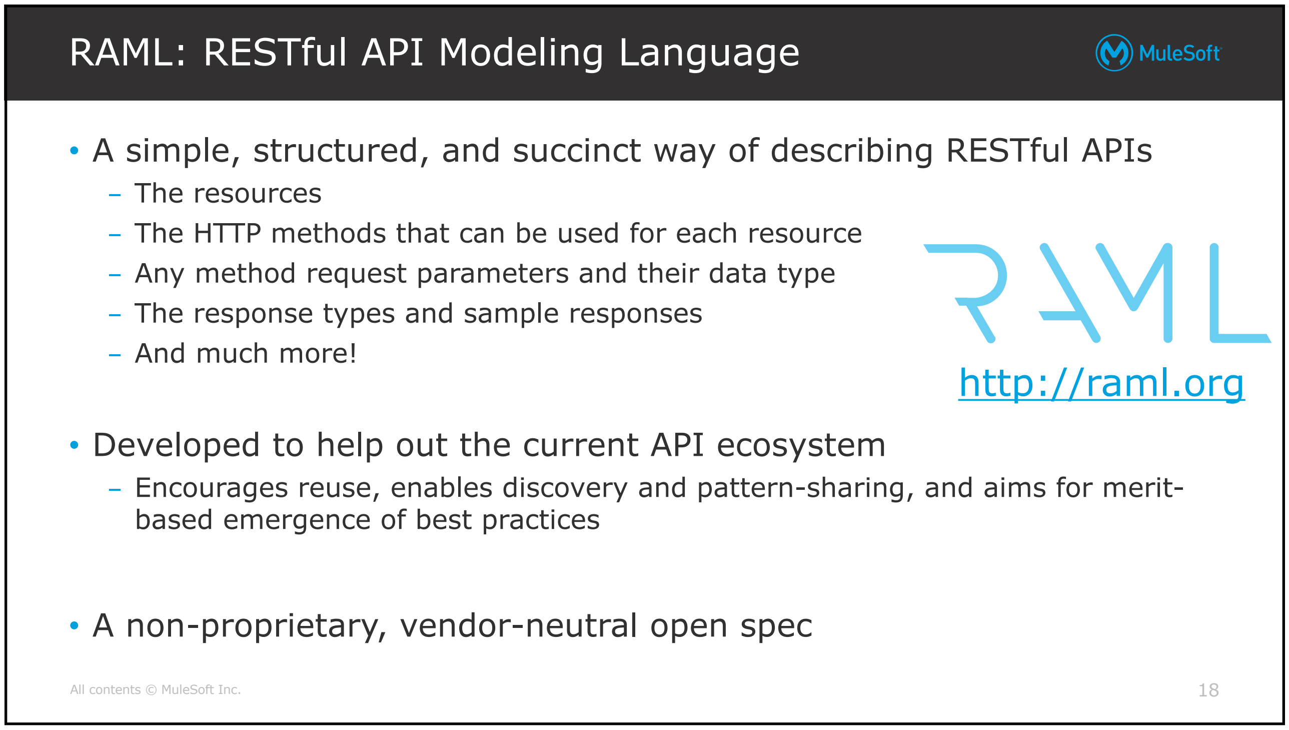
Overview of API description languages

MuleSoft

- OpenAPI Specification previously known as the Swagger specification
  - Generates documentation of REST API methods, parameters and models
  - JSON based code – creates client and server stubs by parsing the OpenAPI definition
- Apiary’s API Blueprint
  - Based on Markdown language
  - The API structure blends in with the documentation
- Several other description languages like I/O Docs(Mashery), Open Data Protocol etc.



Introducing the RESTful API Modeling API Language (RAML)



RAML: RESTful API Modeling Language

- 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!

http://raml.org

- Developed to help out the current API ecosystem
  - Encourages reuse, enables discovery and pattern-sharing, and aims for merit-based emergence of best practices
- A non-proprietary, vendor-neutral open spec

18

RAML features

- RAML is a blueprint to define and model an API
- It helps manage the entire API lifecycle from design to testing and sharing
- It is a machine readable language that is human friendly too
- Two versions available
  - RAML 0.8
  - RAML 1.0
- MuleSoft joined the OpenAPI Initiative to support using RAML and Swagger together
  - Enabled interoperability by providing RAML modeling atop of the OpenAPI Specification
  - Provide common programmatic capabilities and facilitate collaboration

Additional features in RAML 1.0

- RAML 1.0 helps create more modular, reusable API specifications
- It includes new features such as
  - Libraries
  - Overlays and extensions
  - Annotations
  - Datatypes
- 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 specifications can be used to …

MuleSoft

- Auto-generate API documentation
  - For an API Console in an API Portal (interactive docs)
  - 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
- To enable auto-discovery of endpoints for users in tools like Studio

Important terminology in RAML 1.0

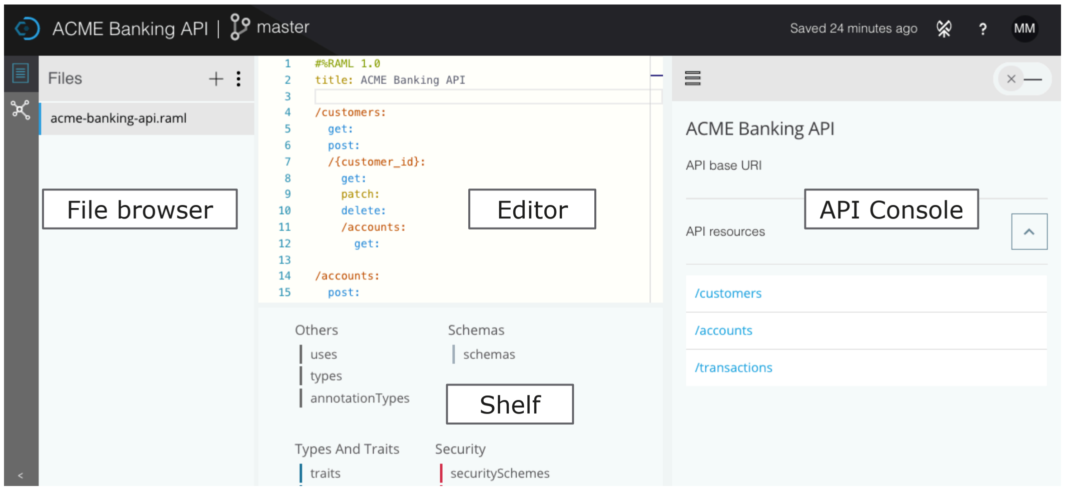
MuleSoft

- YAML and JSON based modeling language
- Consists of nodes – nodes are keys accepting values in the form of
  - Map
    - Consists of multiple key-value pairs
  - Scalar valued
    - Single value, for example *description: This is an example*
  - Sequence
    - Array of values 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

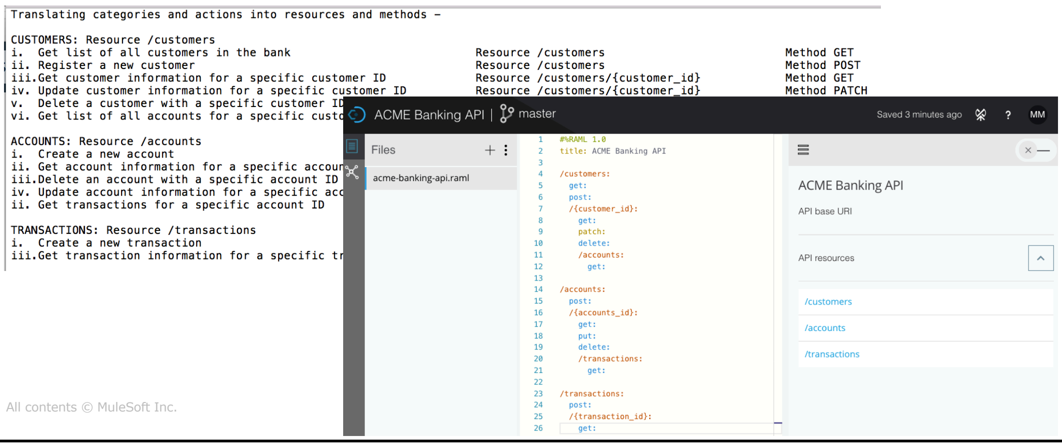
- API Designer is a part of the API Manager entitlement



```
#%RAML 1.0
title: ACME Banking API

/customers:
  get:
  post:
  /{customer_id}:
    get:
    patch:
    delete:
    /accounts:
      get:

/accounts:
  post:
```

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

- 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
i.  Get list of all customers in the bank                      Resource /customers                     Method GET
ii. Register a new customer                                    Resource /customers                     Method POST
iii.Get customer information for a specific customer ID        Resource /customers/{customer_id}       Method GET
iv. Update customer information for a specific customer ID     Resource /customers/{customer_id}       Method PATCH
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 accour
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
```

```
#%RAML 1.0
title: ACME Banking API

/customers:
  get:
  post:
  /{customer_id}:
    get:
    patch:
    delete:
    /accounts:
      get:

/accounts:
  post:
  /{accounts_id}:
    get:
    put:
    delete:
    /transactions:
      get:

/transactions:
  post:
  /{transaction_id}:
    get:
```

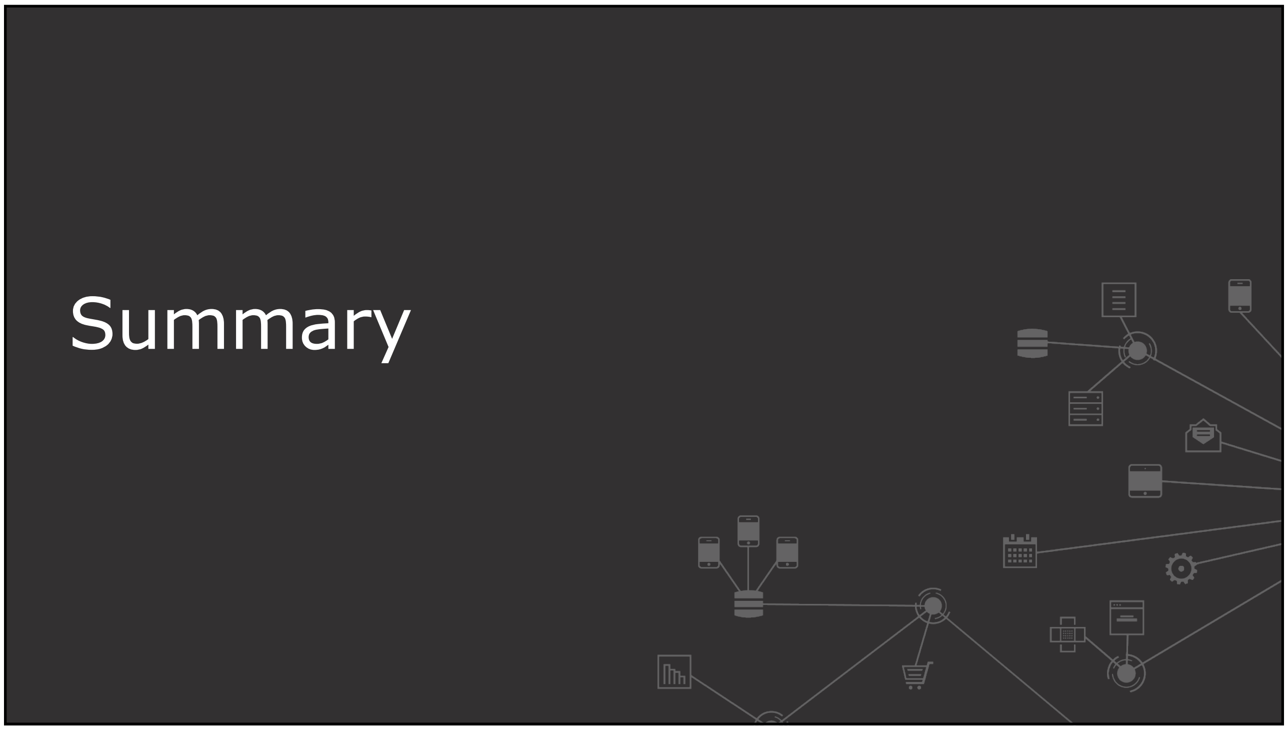
Passing data to methods

Passing data into methods

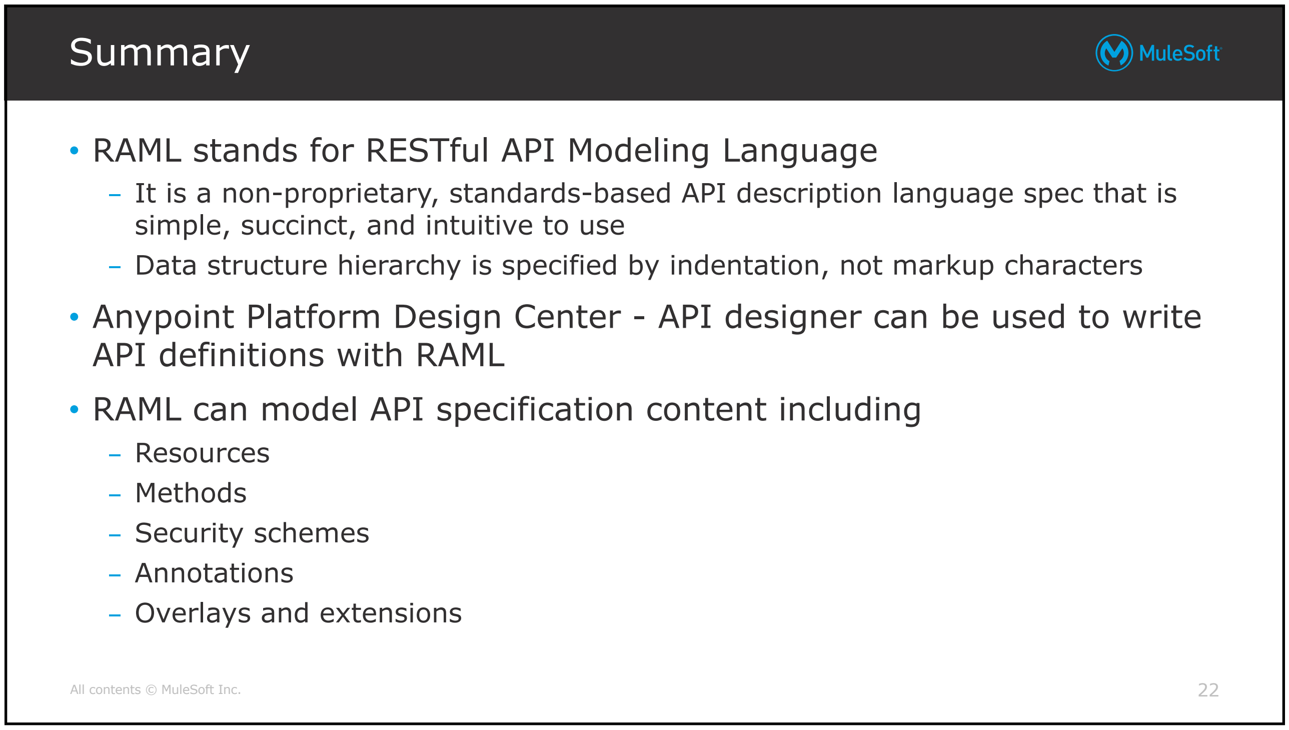
- URI parameters
  - Represented as a nested resource in curly braces
  - Example
    - /users/{userID}, the value of {userID} is dynamic i.e. /users/21gnoe9/
  - Best practice
    - Use for unique identifiers, because they affect a subtree of resources in the URL (if a subtree exists)

Passing data into methods

- 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

- 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