Anypoint Platform: API Design

Summary

This course is for API designers, developers, and architects who want to get hands-on experience creating well-designed, modular API definitions using RAML 1.0 and Anypoint Platform™.

Duration

2 days (in-person or online)

Objectives

At the end of this course, students should be able to:

- Translate functional design requirements into API resources and methods.
- Use API Designer to create API specifications.
- Define API resources, methods, parameters, and responses using RAML.
- Model data in APIs using data types.
- Document and test APIs.
- Make APIs discoverable.
- Minimize repetition in APIs using resource types and traits.
- Modularize APIs using libraries, overlays, and extensions.
- Specify API security schemes.
- Enhance API responses using hypermedia.
- Version APIs.

Prerequisites

None

Setup requirements

- A computer with a minimum screen resolution of 1024x768
- Unrestricted internet access to port 80 (with > 5Mbps download and > 2Mbps upload)
- The latest version of Chrome, Safari, Firefox, or Edge
- An Anypoint Platform account

Get a detailed setup document here.

Outline

PART 1: Designing APIs

Module 1: Introducing RESTful API design

- Describe common web API formats including SOAP, RPC, and REST
- Describe REST API architecture
- List the rules for enforcing REST principles in APIs
- Describe the design-first approach for REST APIs

Module 2: Translating functional requirements for APIs

- Identify different categories and actions for REST APIs
- Translate categories to resources
- Select HTTP methods to support the actions on the categories

Module 3: Introducing API-led connectivity and the API lifecycle

- Describe the API lifecycle
- Explain MuleSoft's API-led connectivity approach
- Describe the API lifecycle with Anypoint Platform
- Navigate Anypoint Platform

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

Module 4: Defining API resources and methods

- Describe API specification languages used to create API definitions
- Use API Designer to create RAML API definitions
- Define resources and methods in RAML API definitions

Module 5: Specifying responses

- Create HTTP method responses
- Use status codes in HTTP responses
- Add error handling and caching information to HTTP responses
- Select and specify the types of content returned in HTTP responses

Module 6: Modeling data

- Identify datatypes and attributes used in resource methods
- Create datatype fragments
- Set request and response body types to datatypes
- Create examples for datatype fragments
- Include examples in datatype fragments

Module 7: Documenting and testing APIs

- Add documentation and description nodes to API definitions
- Use the mocking service to create API endpoints
- Use the API console to test API endpoints

Module 8: Making APIs discoverable

- Publish API specifications and fragments to Anypoint Exchange for discovery
- Create API portals for learning about and testing APIs in Anypoint Exchange
- Customize public portals with themes
- Create sample use cases with API Notebook in API portals
- Gather feedback from API consumers

Module 9: Reusing patterns

- Create and reference resource types for reusability
- Use traits to modularize methods

Module 10: Modularizing APIs

- Use libraries for greater API composability
- Use overlays to override resource information
- Use extensions to enhance resources

Module 11: Securing APIs

- Define API security requirements
- Use security schemes to apply resource-level and method-level policies
- Define custom security schemes for APIs
- Apply an OAuth2.0 external provider policy to resource methods

Module 12: Enhancing API responses using hypermedia

- Describe hypermedia
- Simplify API discoverability and enhance responses using hypermedia
- Modify API definitions to generate state-specific client responses

Module 13: Versioning APIs

- Explain when and when not to version APIs
- Describe the methods of versioning APIs
- Document changes in new API versions using API portals
- Deprecate old API versions