Contents

[Course content 6](#_Toc147005642)

[Section 1: Introduction 6](#_Toc147005643)

[1. Mule ESB integration 6](#_Toc147005644)

[2. Integration introduction 8](#_Toc147005645)

[3. Mule API development life cycle and components 10](#_Toc147005646)

[Section 2: Mule 4 installation set-up 10](#_Toc147005647)

[6. Local Environment setup 10](#_Toc147005648)

[7. Cloud (anypoint platform) registration-> 11](#_Toc147005649)

[Section 3: Mule basics - develop first mule app, run, debug, export & import mule app 11](#_Toc147005650)

[9. Web services - rest services basics 12](#_Toc147005651)

[10. Let's develop first mule app - simple rest service development & run/deploy 13](#_Toc147005652)

[11. Mule global elements & why connector configurations ? 13](#_Toc147005653)

[12. Multiple flows in a mule configuration file vs multiple configuration files 13](#_Toc147005654)

[13. Mule application debugging introduction 13](#_Toc147005655)

[14. Export a mule application & Import a mule application 13](#_Toc147005656)

[Section 4: Mule basics - Mule events and mule message structure 13](#_Toc147005657)

[16. Mule event message structure 14](#_Toc147005658)

[17. Access mule event payload and attributes – old 15](#_Toc147005659)

[18. Mule events - extract JSON fields 15](#_Toc147005660)

[19. Mule events - extract XML elements and XML attributes 15](#_Toc147005661)

[20. Mule events - extract header parameters and query parameters 15](#_Toc147005662)

[21. Mule events - extract URI parameters 15](#_Toc147005663)

[22. Mule variables - create and extract variables 15](#_Toc147005664)

[23. Mule variables - need of variables in mule flow 15](#_Toc147005665)

[24. Rest services - setup http hard coded response in listener connector 15](#_Toc147005666)

[25. Rest services - setup http dynamic response in listener connector 15](#_Toc147005667)

[**Section 5: Mule basics - Basic transformations** 26. Transformations introduction 15](#_Toc147005668)

[27. Transform JSON input to XML output 15](#_Toc147005669)

[28. Transform XML input to JSON output 15](#_Toc147005670)

[29. Transform Message connector preview feature 15](#_Toc147005671)

[30. Basic transformations - old 15](#_Toc147005672)

[**Section 6: Mule rest-full services (database) as provider - simple use case to get started** 31. Install oracle and sql developer 15](#_Toc147005673)

[32. Mule rest service to post JSON payload to insert to database 15](#_Toc147005674)

[33. Mule rest service to post XML payload to update to database 15](#_Toc147005675)

[34. Mule rest service to fetch records from database using query parameter 15](#_Toc147005676)

[35. Mule rest service by URI param to delete record in table using stored proc call 15](#_Toc147005677)

[**Section 7: DataWeave language – basics** 36. DataWeave language introduction 15](#_Toc147005678)

[37. DataWeave selectors part 1 15](#_Toc147005679)

[38. DataWeave selectors part 2 15](#_Toc147005680)

[39. DataWeave datatypes 15](#_Toc147005681)

[40. DataWeave variables - pre defined variables 15](#_Toc147005682)

[41. DataWeave variables - Create flow level variables in transform message 15](#_Toc147005683)

[42. DataWeave variables - Create variables in DataWeave headers 15](#_Toc147005684)

[43. DataWeave - type coercion 15](#_Toc147005685)

[44. DataWeave - date type coercion 15](#_Toc147005686)

[45. DataWeave - introduction to now() function 15](#_Toc147005687)

[**Section 8: Mule database module - first system api uhub (database) sapi implementation** 46. Real time use case uhub-sapi requirement walkthrough & db scripts execution 15](#_Toc147005688)

[47. Real time use case uhub-sapi rest provider development 15](#_Toc147005689)

[48. Mule database auto-generated keys feature 15](#_Toc147005690)

[49. Calling database stored procedure with in and out parameters 15](#_Toc147005691)

[50. Real time use case - practice activity 15](#_Toc147005692)

[51. Transaction ID, Correlation ID over integrations & Choice router introduction 15](#_Toc147005693)

[52. Database - Writing dynamic database queries 16](#_Toc147005694)

[53. Database - connection pooling 16](#_Toc147005695)

[**Section 9: Mule validation module and flow types** 54. Mule validations introduction 16](#_Toc147005696)

[55. Develop required validations in real time use case 16](#_Toc147005697)

[56. Types of mule flows 16](#_Toc147005698)

[57. Payload and varaibles propagation among main flows and sub flows 16](#_Toc147005699)

[58. JSON schema validation 16](#_Toc147005700)

[59. XML schema validation 16](#_Toc147005701)

[**Section 10: Mule error handling** 60. Error handling introduction & mule default error handling 16](#_Toc147005702)

[61. Error handling using on-error-propagate vs on-error-continue 16](#_Toc147005703)

[62. Multiple errors handling in flow 16](#_Toc147005704)

[63. Multiple error types selection & common error handling in a error handling block 16](#_Toc147005705)

[64. Error handling - error mapping feature on connectors 16](#_Toc147005706)

[65. Error handling - Raise Error - Raising custom errors and handling custom errors 16](#_Toc147005707)

[66. Error handling on group of event processors using try block & scenarios 16](#_Toc147005708)

[67. Error handling b/w main flows & sub flows and scenarios 16](#_Toc147005709)

[68. Error handler - conflicts b/w global error handler vs local error handler 16](#_Toc147005710)

[69. Error handler - solution to solve conflict b/w global & local error handlers 16](#_Toc147005711)

[**Section 11: Anypoint platform introduction & access management** 70. Anypoint platform capabilities 16](#_Toc147005712)

[71. Configure business groups, environments in an organization 16](#_Toc147005713)

[72. Configure users , roles and permissions within an organization 16](#_Toc147005714)

[**Section 12: Anypoint Platform - Design Center - Design API's Using RAML** 73. Design API specification using RAML introduction 16](#_Toc147005715)

[74. Design API specification using RAML for a simple use case to get started 16](#_Toc147005716)

[75. Mocking API specification & scenarios 16](#_Toc147005717)

[76. Resource's as same name with multiple http methods & nested JSON, XML payload's 16](#_Toc147005718)

[77. API spec with headers, URI parameters, query parameters & deal array of objects 16](#_Toc147005719)

[**Section 13: Anypoint Platform - Design Center - Design Fragments Based API's Using RAML** 78. RAML fragments introduction 16](#_Toc147005720)

[79. Fragments - data types 16](#_Toc147005721)

[80. Fragments - Library 16](#_Toc147005722)

[81. Fragments - example 16](#_Toc147005723)

[82. Fragments - traits 16](#_Toc147005724)

[83. Fragments - security scheme - basic auth 16](#_Toc147005725)

[84. Fragments - security scheme - oauth 2.0 16](#_Toc147005726)

[85. Introducing health check api 16](#_Toc147005727)

[**Section 14: Anypoint Platform - Anypoint Exchange - Publishing API's to Exchange** 86. Anypoint exchange introduction 16](#_Toc147005728)

[87. Publish API specification to exchange 16](#_Toc147005729)

[88. API Exchange portal - API documentation 17](#_Toc147005730)

[89. Simulate, feedback and validation of API 17](#_Toc147005731)

[90. Add contributors to exchange API assets 17](#_Toc147005732)

[91. Share API exchange portal access to external clients 17](#_Toc147005733)

[**Section 15: API based project development and real time scenarios** 92. Create API based project 17](#_Toc147005734)

[93. API kit router 17](#_Toc147005735)

[94. API validations using RAML 17](#_Toc147005736)

[95. Mule application real time naming standards 17](#_Toc147005737)

[96. Mule application real time project structure 17](#_Toc147005738)

[97. API based project implementation 17](#_Toc147005739)

[98. Real time global error handler 17](#_Toc147005740)

[99. Deploy API to mule runtime cloud hub 1.0 17](#_Toc147005741)

[100. Setup a free cloud database 17](#_Toc147005742)

[101. Point mule application to cloud database 17](#_Toc147005743)

[Section 16: Mule configure properties per environment 102. Mule configuration properties file to read properties 17](#_Toc147005744)

[103. Mule configuration YAML file to read properties 17](#_Toc147005745)

[104. Mule secure configuration properties to read properties 17](#_Toc147005746)

[105. Load property files dynamically by environment 17](#_Toc147005747)

[**Section 17: Anypoint platform - Runtime Manager - Deployment to CloudHub 1.0** 106. Configure CloudHub runtime properties & deploy application to CloudHub 17](#_Toc147005748)

[107. Runtime monitoring, dashboard, vCores, workers - vertical & horizontal scaling 17](#_Toc147005749)

[108. Safely hide application runtime manager properties 17](#_Toc147005750)

[109. Runtime manager - Monitoring insights of runtime application 17](#_Toc147005751)

[110. Runtime - Changing log levels to debug & enable verbose logs on mule components 17](#_Toc147005752)

[111. Runtime manager - Setup alerts on applications & servers critical conditions 17](#_Toc147005753)

[**Section 18: Anypoint platform - API Manager - API Management** 112. API Management & API gateway introduction 17](#_Toc147005754)

[113. Create API in API Manager and about API instance id 17](#_Toc147005755)

[114. API auto discovery 17](#_Toc147005756)

[115. Policies - basic authentication 17](#_Toc147005757)

[116. Policies - client id enforcement policy 17](#_Toc147005758)

[117. Policies - rate limiting policy 17](#_Toc147005759)

[118. Policies - Rate limiting - SLA based policy 17](#_Toc147005760)

[119. Policies - JSON and XML threat protection policies 17](#_Toc147005761)

[120. Policies - IP allow list and IP block list policies 17](#_Toc147005762)

[121. Policies - Setup automatic policies across API's and CORS policy 17](#_Toc147005763)

[122. Policies - Header injection and header removal policies 17](#_Toc147005764)

[123. Policies - Message logging policy 17](#_Toc147005765)

[124. Policies - Setup oauth provider and apply oauth 2.0 policy 17](#_Toc147005766)

[125. API Proxy-Create proxy application on external services, decode & API Management 18](#_Toc147005767)

[126. Promote API to higher environment 18](#_Toc147005768)

[Section 19: Anypoint platform - Runtime Manager - Deployment to CloudHub 2.0 18](#_Toc147005769)

[Section 20: Anypoint Platform - Design Center - Design API's Using OAS 18](#_Toc147005770)

[Section 21: Mule JMS 18](#_Toc147005771)

[Section 22: On premise deployment and hybrid deployment 18](#_Toc147005772)

[Section 23: Mule HTTP Request - Consume REST services 18](#_Toc147005773)

[Section 24: Mule SOAP services (web services) 18](#_Toc147005774)

[Section 25: Mule File and FTP 18](#_Toc147005775)

[Section 26: Mule Salesforce integration 18](#_Toc147005776)

[Section 27: Mule scatter-gather router 18](#_Toc147005777)

[Section 28: Mule scopes 18](#_Toc147005778)

[Section 29: Dataweave 2.0 18](#_Toc147005779)

[Section 30: Munit 18](#_Toc147005780)

[Section 31: Mule with DevOps 18](#_Toc147005781)

[Section 32: API LED Connectivity 18](#_Toc147005782)

[Section 33: Bonus - Mule real time project (Health domain) 18](#_Toc147005783)

# Course content

## Section 1: Introduction

1. Mule ESB integration  
Mule ESB is a lightweight java-based enterprise service bus (ESB) and integration platforms that allows developers to connect applications together quickly and easily, enabling them to exchange data.

Benefits:

Reduced development time

Cloud enablement

Designing API’s

API Management

Mule community vs enterprise edition

A blue background with white text

Description automatically generated

Mule API SDLC

A diagram of a software development process

Description automatically generated

Web-application->anypoint platform

* Design center-To design the api’s
* Exchange – To maintain the API artifact as a repository
* Management Center –
  + Access Management – To do identity management and configurations of users, runtimes, environments
  + API Manager – To
  + Runtime Manager – To deploy
  + API Governance –
  + Visualizer –
  + Monitoring –
  + Secrets Manager –

A screenshot of a phone

Description automatically generated

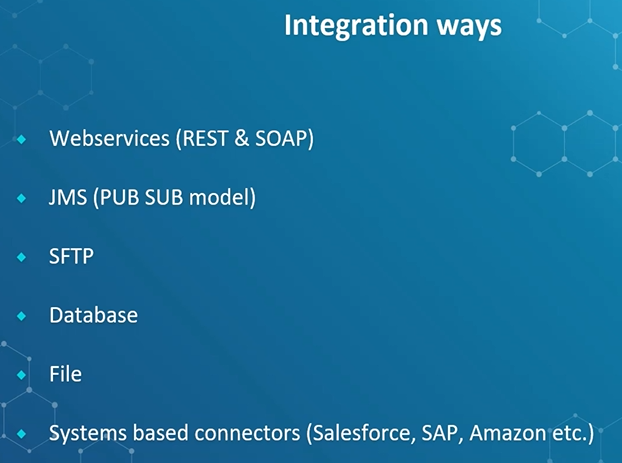
Ide->anypoint studio

### 2. Integration introduction



A blue background with white text

Description automatically generated



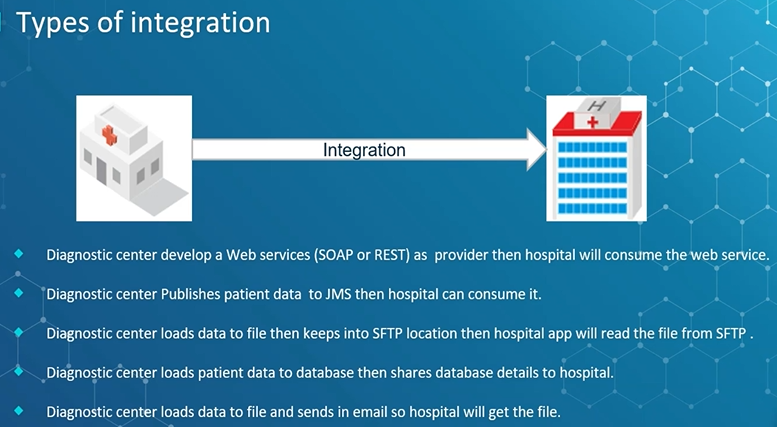
Webservices>JMS> System Based Connectors(salesforce,SAP,Amazon)

Usecase

Types of Integration

A Diagoniost Chain->Diagnosed Data-> Has lots of Patients data

A Hospital chain wants to use this



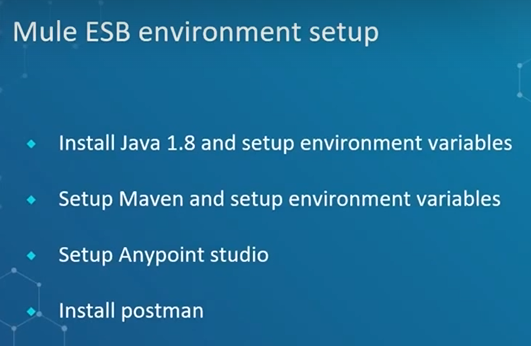
### 3. Mule API development life cycle and components

A diagram of a process

Description automatically generated

## Section 2: Mule 4 installation set-up

### 6. Local Environment setup



Java installation

url -> Java 1.8 -> jdk-8u281-windows-x64.exe

Maven Binary Zip -> apace-maven-3.6.3-bin.zip

Env\_Variables

JAVA\_HOME-> path

Path->java\_path/bin

MAVEN\_HOME-> path

Path->java\_path/bin

For javaw.exe setup

Configure it to point to your JDK installation as the default VM

Java\jdk..\bin\javaw.exe

In .ini file -> replace -vmargs to -vm c:\java\jdk..\bin\javaw/exe

### 

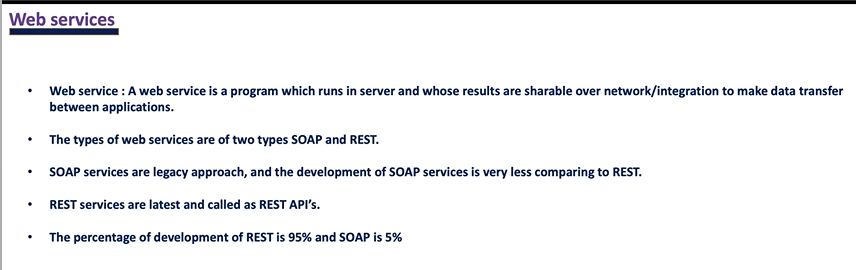
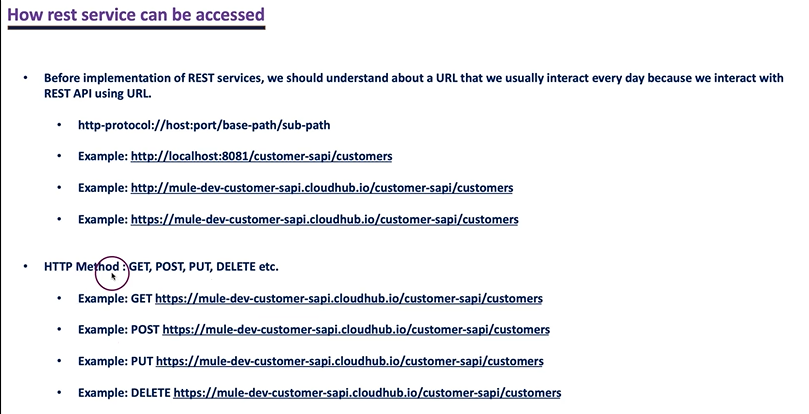
### 7. Cloud (anypoint platform) registration->

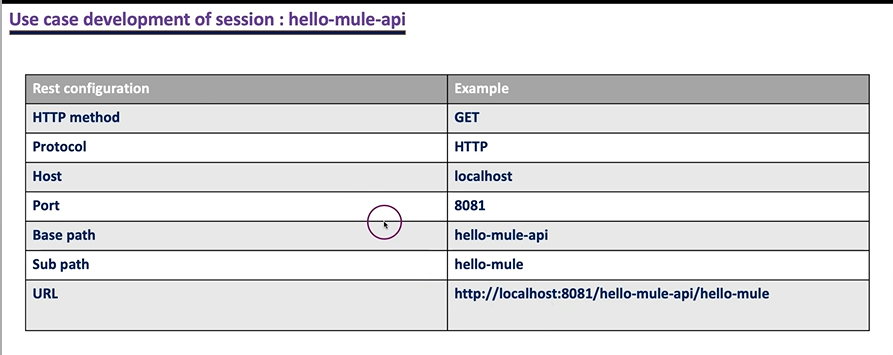
REGISTER ONLINE

## Section 3: Mule basics - develop first mule app, run, debug, export & import mule app

8. Overview of Anypoint Studio  
  
src/main/mule ->sample.xml-> mule configuration file  
src/main/resources -> files  
src/main/munit -> munit files  
src/main/resources -> munit files

src/main/ -> .classpath files generated  
src/test/ -> .classpath files generated  
target->deployable jar file generated  
mule-artifact.json -> mule version  
pom.xml-> All mule projects are maven projects  
log4j.xml -> logger component  
connectors added from pom  
mule server-> server running

9. Web services - rest services basics  
  
  
  
  
A computer screen shot of a computer

Description automatically generated  
  


### 10. Let's develop first mule app - simple rest service development & run/deploy

|  |  |
| --- | --- |
|  | Make a mule app using above methods <http://localhost:8081/hello-mule-api/hello-mule> |

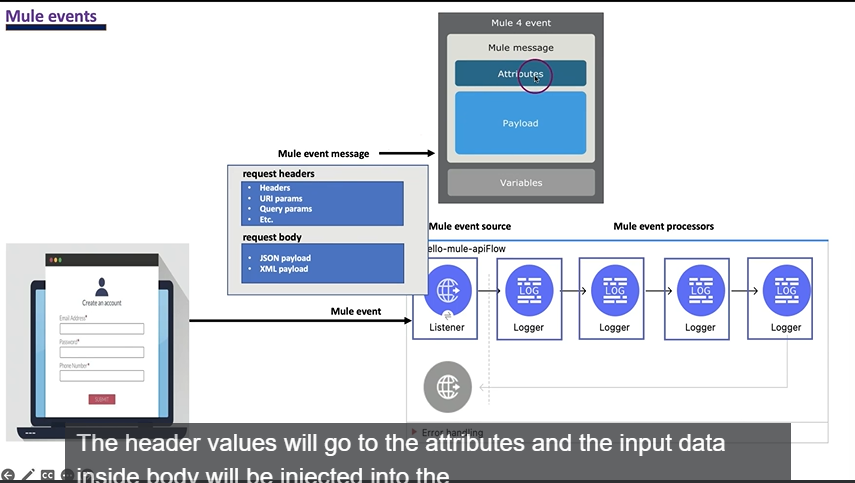
11. Mule global elements & why connector configurations ?  
Global elements are used in different services also.-> base-path same but different sub-path  
We should not use same ports for multiple configurations when configurations are running in one application/server

12. Multiple flows in a mule configuration file vs multiple configuration files  
Ans: create multiple mule configuration files

13. Mule application debugging introduction  
x+y,breakpoint,resume next

### 14. Export a mule application & Import a mule application

Import and export mule projects

Section 4: Mule basics - Mule events and mule message structure  
  
15. Mule event sources, event processors & event message  
  
  
Mule event sources(listners)-> Mule event sources are the trigger points which allows client requests to process this business logic.  
Mule event processors(components)-> The mule event processors are responsible to do the business that we developed.

16. Mule event message structure  
A screenshot of a computer

Description automatically generated

### 17. Access mule event payload and attributes – old attributes.queryParams attributes.uriParams payload.age

### 18. Mule events - extract JSON fields

### 19. Mule events - extract XML elements and XML attributes

### 20. Mule events - extract header parameters and query parameters

### 21. Mule events - extract URI parameters

### 22. Mule variables - create and extract variables

### 23. Mule variables - need of variables in mule flow

### 24. Rest services - setup http hard coded response in listener connector body: “success”, status code :200 , reason phase : “ok”

### 25. Rest services - setup http dynamic response in listener connector

### Section 5: Mule basics - Basic transformations 26. Transformations introduction A diagram of a flowchart Description automatically generated A diagram of a process Description automatically generated

### 27. Transform JSON input to XML output A screenshot of a computer code Description automatically generated

### 28. Transform XML input to JSON output

### 29. Transform Message connector preview feature A screenshot of a computer Description automatically generated

### 30. Basic transformations - old A screen shot of a computer Description automatically generated for json payload root element is optional but in xml it is mandatory xml: for making root element -> emp: {} title : payload.book.title.@lang | <title lang=”en”>Everyday Italian</title> (output is json) title @(lang: payload.item.properties.language) : payload.item.properties.title(to assign attributes when json to xml) for schema generation: xml:

Freeformatter.com/xsd-generator-> xml schema generation  
 employee\_schema.xsd

### Section 6: Mule rest-full services (database) as provider - simple use case to get started 31. Install oracle and sql developer

Sql workbench->shell,server,bench->give bin path of server in env variables.

### 32. Mule rest service to post JSON payload to insert to database

**Pm curl**

curl --location 'http://localhost:8091/emp-sapi/create-employee' \

--header 'Content-Type: application/json' \

--data '{

"employeeID": 123,

"employeeName": "Chinna",

"employeeStatus": "A"

}'

insert into emp(emp\_id, emp\_name, emp\_status) values(:empID, :empName, :empStatus)

{

empID: payload.employeeID,

empName: payload.employeeName,

empStatus: payload.employeeStatus

}

33. Mule rest service to post XML payload to update to database  
<?xml version="1.0" encoding="UTF-8"?>

<employee>

    <employeeID>123</employeeID>

    <employeeStatus>inActive</employeeStatus>

</employee>

update emp set emp\_status=:empStatus where emp\_id=:empID

{

empStatus: payload.employee.employeeStatus,

empID: payload.employee.employeeID

}

34. Mule rest service to fetch records from database using query parameter

### 35. Mule rest service by URI param to delete record in table using stored proc call stored procedure DELIMITER //

### CREATE PROCEDURE delete\_emp(IN id INT, IN name VARCHAR(255))

### BEGIN

### DELETE FROM emp

### WHERE emp\_id = id AND emp\_name = name;

### END;

### //

### DELIMITER ;

### Section 7: DataWeave language – basics 36. DataWeave language introduction

### 37. DataWeave selectors part 1

### 38. DataWeave selectors part 2

### 39. DataWeave datatypes

### 40. DataWeave variables - pre defined variables

### 41. DataWeave variables - Create flow level variables in transform message

### 42. DataWeave variables - Create variables in DataWeave headers

### 43. DataWeave - type coercion

### 44. DataWeave - date type coercion as Date {format: “dd-MM-yyyy”}

### 45. DataWeave - introduction to now() function

### Section 8: Mule database module - first system api uhub (database) sapi implementation 46. Real time use case uhub-sapi requirement walkthrough & db scripts execution

### 47. Real time use case uhub-sapi rest provider development

### 48. Mule database auto-generated keys feature

### 49. Calling database stored procedure with in and out parameters

### 50. Real time use case - practice activity

### 51. Transaction ID, Correlation ID over integrations & Choice router introduction

### 52. Database - Writing dynamic database queries

### 53. Database - connection pooling

### Section 9: Mule validation module and flow types 54. Mule validations introduction

### 55. Develop required validations in real time use case

### 56. Types of mule flows

### 57. Payload and varaibles propagation among main flows and sub flows

### 58. JSON schema validation

### 59. XML schema validation

### Section 10: Mule error handling 60. Error handling introduction & mule default error handling

### 61. Error handling using on-error-propagate vs on-error-continue

### 62. Multiple errors handling in flow

### 63. Multiple error types selection & common error handling in a error handling block

### 64. Error handling - error mapping feature on connectors

### 65. Error handling - Raise Error - Raising custom errors and handling custom errors

### 66. Error handling on group of event processors using try block & scenarios

### 67. Error handling b/w main flows & sub flows and scenarios

### 68. Error handler - conflicts b/w global error handler vs local error handler

### 69. Error handler - solution to solve conflict b/w global & local error handlers

### Section 11: Anypoint platform introduction & access management 70. Anypoint platform capabilities

### 71. Configure business groups, environments in an organization

### 72. Configure users , roles and permissions within an organization

### Section 12: Anypoint Platform - Design Center - Design API's Using RAML 73. Design API specification using RAML introduction

### 74. Design API specification using RAML for a simple use case to get started

### 75. Mocking API specification & scenarios

### 76. Resource's as same name with multiple http methods & nested JSON, XML payload's

### 77. API spec with headers, URI parameters, query parameters & deal array of objects

### Section 13: Anypoint Platform - Design Center - Design Fragments Based API's Using RAML 78. RAML fragments introduction

### 79. Fragments - data types

### 80. Fragments - Library

### 81. Fragments - example

### 82. Fragments - traits

### 83. Fragments - security scheme - basic auth

### 84. Fragments - security scheme - oauth 2.0

### 85. Introducing health check api

### Section 14: Anypoint Platform - Anypoint Exchange - Publishing API's to Exchange 86. Anypoint exchange introduction

### 87. Publish API specification to exchange

### 88. API Exchange portal - API documentation

### 89. Simulate, feedback and validation of API

### 90. Add contributors to exchange API assets

### 91. Share API exchange portal access to external clients

### Section 15: API based project development and real time scenarios 92. Create API based project

### 93. API kit router

### 94. API validations using RAML

### 95. Mule application real time naming standards

### 96. Mule application real time project structure

### 97. API based project implementation

### 98. Real time global error handler

### 99. Deploy API to mule runtime cloud hub 1.0

### 100. Setup a free cloud database

### 101. Point mule application to cloud database

### Section 16: Mule configure properties per environment 102. Mule configuration properties file to read properties

### 103. Mule configuration YAML file to read properties

### 104. Mule secure configuration properties to read properties

### 105. Load property files dynamically by environment

### Section 17: Anypoint platform - Runtime Manager - Deployment to CloudHub 1.0 106. Configure CloudHub runtime properties & deploy application to CloudHub

### 107. Runtime monitoring, dashboard, vCores, workers - vertical & horizontal scaling

### 108. Safely hide application runtime manager properties

### 109. Runtime manager - Monitoring insights of runtime application

### 110. Runtime - Changing log levels to debug & enable verbose logs on mule components

### 111. Runtime manager - Setup alerts on applications & servers critical conditions

### Section 18: Anypoint platform - API Manager - API Management 112. API Management & API gateway introduction

### 113. Create API in API Manager and about API instance id

### 114. API auto discovery

### 115. Policies - basic authentication

### 116. Policies - client id enforcement policy

### 117. Policies - rate limiting policy

### 118. Policies - Rate limiting - SLA based policy

### 119. Policies - JSON and XML threat protection policies

### 120. Policies - IP allow list and IP block list policies

### 121. Policies - Setup automatic policies across API's and CORS policy

### 122. Policies - Header injection and header removal policies

### 123. Policies - Message logging policy

### 124. Policies - Setup oauth provider and apply oauth 2.0 policy

### 125. API Proxy-Create proxy application on external services, decode & API Management

### 126. Promote API to higher environment

## Section 19: Anypoint platform - Runtime Manager - Deployment to CloudHub 2.0

## Section 20: Anypoint Platform - Design Center - Design API's Using OAS

## Section 21: Mule JMS

## Section 22: On premise deployment and hybrid deployment

## Section 23: Mule HTTP Request - Consume REST services

## Section 24: Mule SOAP services (web services)

## Section 25: Mule File and FTP

## Section 26: Mule Salesforce integration

## Section 27: Mule scatter-gather router

## Section 28: Mule scopes

## Section 29: Dataweave 2.0

## Section 30: Munit

## Section 31: Mule with DevOps

## Section 32: API LED Connectivity

## Section 33: Bonus - Mule real time project (Health domain)