Self Details

1. I am Siva Kishor Reddy.
2. I was Graduated my engineering degree from B.I.T.S.
3. I did My +2 in ShirdiSai Junior College.
4. I am Schooling at Z.P.H.S.
5. Coming to my experience I have around 6 years of experience in Software industry. I stared my carrier as software developer with Fabex tech Solutions.
6. Right now am working with capjemini
7. I have excellent ability to work in groups and as well as individual with minimum super vision.

ROLES

1. As a part of the project I have used both Mule 3 (3.8.5) and Mule 4 (4.1.5) versions.
2. I have worked in All the Phases Like Design, Development and Deployment in Mulesoft.
3. In Design Phase I have worked in RAMLs.
4. In Development phase I have worked in Flows, Sub Flows, Exception Strategies, Dataweave
5. and Batch Processing.
6. In Deployment phase, we deploy our applications using Jenkins (maintained by DevOps team)

ROLES

1. **Completed Code migration from Mule 3.8.5 version to 4.1.5 in 90 days and was rewarded for the same.**
2. **Completed Migration of complex XSLTs to Dataweave.**
3. **○ Extensively worked on Dataweave**
4. **○ Implemented reusable global Dataweave modules.**

Realtime Project Currently Working

In JDA,
We use JDA products TMS (Transport Management System) and WMS (Warehouse Management System).

In this project we use Mulesoft to share information between TMS and WMS. We use Mulesoft within JDA's infrastructure. We are not exposing any APIs. Where as we use HTTP connectors to communicate among multiple Mulesoft projects inside our architecture.

Source: TMS/WMS
Destination: WMS/TMS

API LED Connectivity

the methodical way to transfer data between applications through reusable and purposeful apis.
The Definition itself is saying that it is reusable and purposeful apis,
If there is any purpose then only you can go with api led connectivity.
If there is no purpose then don't go with api led connectivity.

System APIS:

System api means what and all the systems that we are connecting DataBase that is one system.
we are connecting SAP that is one system
we are connecting salesforce that is one system.

Reusable system calls will be developed as system API's.Once we built system api many users can access the data without any need to learn underlying systems can reuse these API's in multiple project.

When ever you are making integration with any particular system then you just develop a rest service or api on top of systems those are system apis.

For suppose I would like to fetch the employee employee info from employee master database then I can develop 1 system api on top of that DataBase table I can expose that api's outside world.

When I go with System API?

Go for system api if and only if there is a need that system api is reused in multiple applications. If not don't go for it.

Process API:

The process API is one which will consume system api together and it will shape the data into single system by consuming multiple system apis.

Bit confusing right let me give you 1 example
Assume I would like to get customer information or order information from the backend system

Assume I have customer info in salesforce & order info in sap so I need to develop 2 apis.

So obesely I need to develop 2 system apis. System api is separate project like salesforce-sapi that will consume salesforce internally and respond back us with customer details.

In similar way way I can develop 1 more project as System api.sap-sapi that will connect to sap and give you order info
Now I would like have the data for each customer what and all the orders placed.
Means I can develop a single application. which connect to salesforce next it will connect to sap once we get data I can combine orders & customer data into 1 respond then respond back in 1 application itself.

NOTE: For suppose I have system API I want to fetch the data from database and respond back in that case I don’t need process api.

When I go with Process API?

When I would like to call multiple system apis across the systems & I would like to prepare a single response by using those system calls there I can do enrichment and transformations in that case go for process api.

Experience API:

An Experience API is one where who are all want to Experience you process api.That means your Experience api will consume process api with respect to the audience.

For suppose what ever the process apis we developed that need to be exposed to 2 Audience.Assume that 1 is web Application & 2 one is Mobile Application.

Experience API are responsible which will expose 2 audience .

For suppose what ever process api you developed that is going to be consumed by one audience then you don’t need Experience Api.

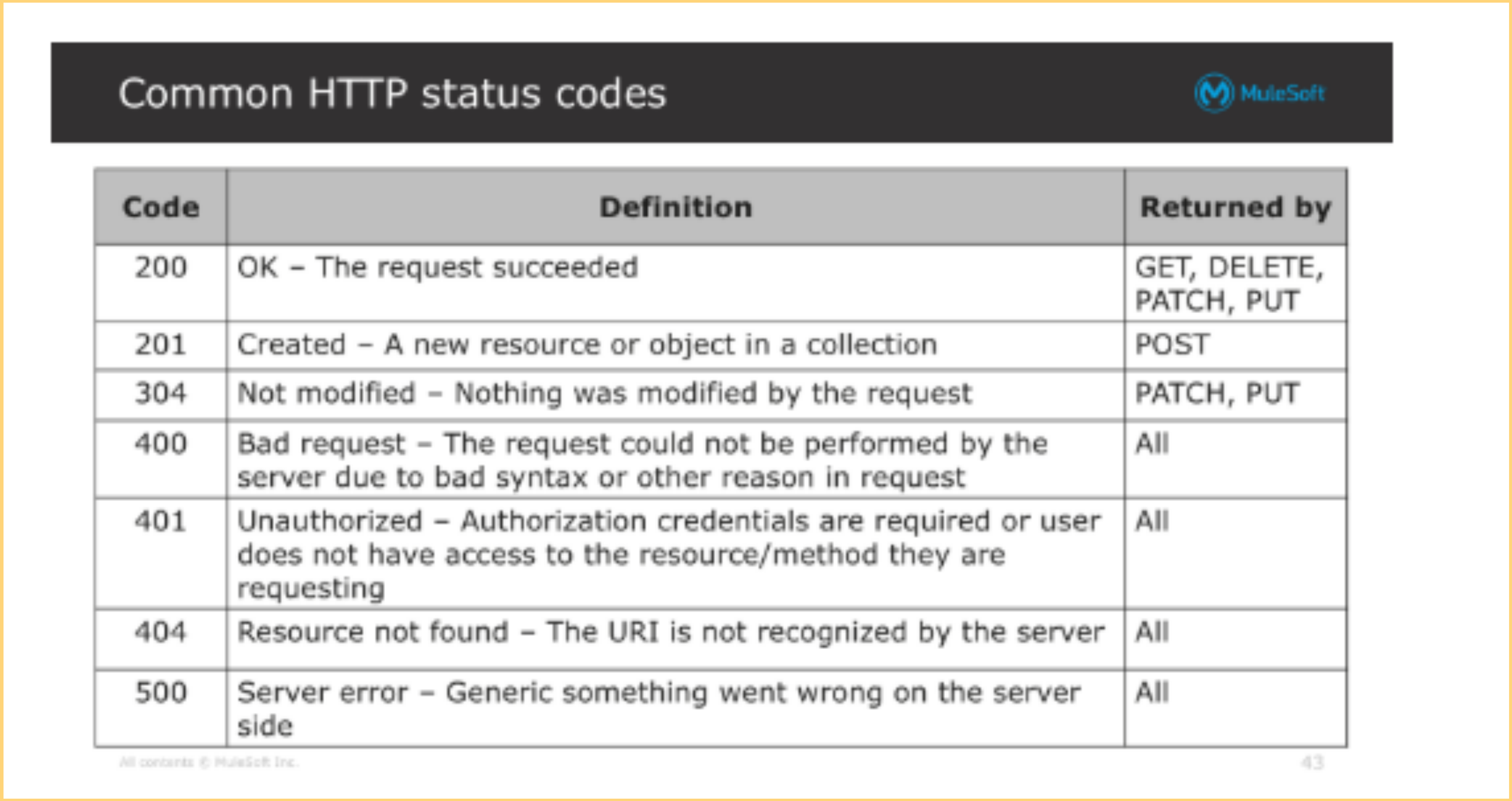
When I go with Experience API?

Experience apis also should be chosen if and only if multiple audience need, multiple way of response or any difference needed with respect to the url or responses or enrichments or transformations so then only go for experience api.

Coding Standards: Naming Conventions and Project Structure

1. Project Structure: (To Improve readability of code)
   - Mule Configuration Files:
     - Main xml: application-name.xml (Main flow with   APIKitRouter)
   - Business Logic: business-logic.xml (Private Flows Generated fromRAML)
   - Error Handler: error-handler.xml (Error Handling Logic, with error- handler block)
   - Global Configurations: global-config.xml (Connector Configuration, Properties Placeholders)

2. Camel-Casing: Variables, Properties, External DWL File names
   Example: sourceSystem, messageId
3. Application Name :(Low Caps):
   companyName-sourceSystem-endSystem-api :Example: mulebytes-sap-workday-api
4. Flow Names (Camel-Casing):
   Example: sendEmailReportFlow



Common HTTP status codes

MuleSoft

Code	Definition	Returned by
200	OK – The request succeeded	GET, DELETE, PATCH, PUT
201	Created – A new resource or object in a collection	POST
304	Not modified – Nothing was modified by the request	PATCH, PUT
400	Bad request – The request could not be performed by the server due to bad syntax or other reason in request	All
401	Unauthorized – Authorization credentials are required or user does not have access to the resource/method they are requesting	All
404	Resource not found – The URI is not recognized by the server	All
500	Server error – Generic something went wrong on the server side	All

RAML

Restful Api Modelling Language

RAML STRUCTURE

IN RAML 3 SECTIONS ARE THERE

ROOT SECTION

HEADER SECTION

BODY SECTION

SCHEMA

Template created for validation request or response message body using data format standard.

Traits:

To configure reusable assets like headers query parameters and responses

IF U GENERATE FLOW FROM RAML
U WILL GET

1.Main Flow
2.Api Console Flow
3.Private Flows
4.Mapping Exception Strategies

How many private flows will be generated

Based on Methods In RAML
ex: if 3 methods are there 3 Private Flows will be generated

1. **Life Cycle of API Development using RAML:**
a. **Design Center:** Create RAML in Design Center. With API Specifications (Resource Paths, Methods, Input Schemas, Traits (If there are reusable headers, query params etc.))
b. **Exchange:** Post Review from Business on the RAML, Publish it to Anypoint Exchange (So that Other Developers and Client Applications, can check your API Specification before implementation and provide additional reviews)
c. **API Manager:** Import API That is Published to Anypoint Exchange into APi manager, and take a note of **API ID** (This will further be used during Development and Deployment steps)
d.**Development:** While Generating Flows from RAML, it creates a
Main Flow (With API KitRouter), API Console Flow, Number of Private Flows as many Number of Methods in the RAML, with Naming Convention (**method:/resourcePath:apikit-config-name**) So that request will be automatically re-routed to appropriate flow matching with **Request-URI - Flow-Name**
You can also defining custom Mappings in APIKit Router.
Write Munit Test Cases if Applicable.
Configure API-Auto Discovery, with API ID that was capture during API Manager Import from Exchange step.

Deployment:

Go to Runtime Manager, Click on Deploy Application, Upload the Artifact (.jar, .zip) and configure Number of Workers and Work Size.
[Worker (vCore): Virtual Mule Server Instance,
Worker Size (vCore Size) : Virtual Memory allocated to Mule Server Instance]
And then deploy the application.

RAML Data Types:

1. String
2. Number
3. Boolean
4. Custom Data Types (**types)**

API Fragment

These are Reusable components that are used to create and publish in order to be utilised across different API's in your organisation you could also have fragments with in your api specification to improve the molecularity of your api specification. Api fragments can be of datatypes, traits, security schemes, resource types and any others.

RAML Best Practices:
1. Identify Commonalities amongst the Resource Paths, so that you can decide what to put inside traits
2. Maintain Standard Response Structures and Codes with Appropriate Examples, which makes it easier in Anypoint Studio Metadata resolution for easier development.
3. Externalise Examples and Schemas into separate files and refer to the using **!include /filePath so that RAML will be more readable.**

How to Read Keys From Properties Files:
1. ${propertyPath} : Example: ${smtp.username}
2. **p ('propertyPath')**: Example: p('smtp.username')

If they are using Secured Properties Placeholder:
1. ${**secure::**propertyPath} : Example: ${**secure::**smtp.username}
2. **p ('secure::propertyPath')**: Example: p('**secure::**smtp.username')

How to define Security in RAML:
securitySchemes

200 vs 201 HTTP Status Codes:
200: Success (Example: Normal API Calls that involves FETCHING records from End system)
201: Created (Example: API Calls that involves INSERTING records into End system)

APIKit Router:
Used to Re-Route Input Requests to Appropriate Private Flows, those are either maintained as Custom Mappings List or Standard Generated as per Naming convention. By Matching Input Request URI to the Flow Names in the Routing Table

How do you refer an external file/resources in RAML

Floder Location : !include /types/all/examples.raml

URI Param VS QUERY Param

URI Param is basically used to identify a specific resource or resource type

QUERY Param is basically used to sort or filter those resources

Library in RAML:

Library is single RAML file, in which we can define data types, examples, resources etc.

ResourceType:

ResourceType is basically a template that is used to define the descriptions, methods, and parameters that can be used by multiple resources without writing the duplicate code or repeating code.

Resource:

Resource that uses resourceTypes can inherit its nodes

Traits:

Traits are like functions, using traits we can group Common Attributes like Headers, QueryParams etc

Cache scope

Cache scope is for storing and reusing frequently called data.you can use cache scope to reduce the processing load on the mule instance and to increase the speed of message processing with in a flow. It is particularly effective for processing repeated request for the same information.processing request for information that involves large, non-consumable payloads.

What is payload

payload is a mule runtime variable that stores arrays of objects.

How to consume soap

Using web service consumer component .configure the connector with WSDL or SOAP web service configuration details like Service name, URL, port etc.

FLOWS

WHAT IS FLOW ?

Flow is collection of mule components

Each flow may have a Processing Strategy as well as an Exception handling strategy associated with it.

FLOW TYPES :

Main Flow

SUB FLOWS

Private

SUB FLOWS:

Sub-flows are processed synchronously

Sub-flows inherit the processing strategies and exception strategies of the calling flow

SYNCHRONOUS:

Synchronous flow is similar to sub flow

This type of flow process the message with in single thread

It Doesn’t inherit the processing strategies and exception strategies of the calling flow

ASYNCHRONOUS:

It will process the message parallel

It means it will run in Background it will not wait for response

Private Flow:

A Flow with out message source is called Private Flow

Private flows are another type of reusable flows, much similar to sub-flows but with a very different behaviour in term of threading and exception handling

They can also be called using the flow-reference element.

Using:

used to create local variables for a scope inside data weave .

Mule 3	Mule 4
Mule Message: Payload: Properties (Inbound/Outbound): (Inbound properties are immutable) Attachments Inbound (Outbound):	**Mule Message:** Payload: Attributes: Attachments:
Types of Exception Strategies: • Mapping Exception Strategy (RAML generated along with API Kit Router) • Choice Exception Strategy • Catch Exception Strategy • Rollback Exception Strategy	**Types of Exception Strategies:** • On Error Propagate • On Error Continue
Look and Feel of Anypoint Studio (6): • If a place a connector example (HTTP) in source it acts a Listener, if I place it in Source it acts a Requester • It's not possible to read File in between the flow, we will have to use custom connector like Mule	**Look and Feel of Anypoint Studio (7):** • All Connectors are pre build in a Modularized a way. • Each Connector has individual components for all the operations.
Here Dataweave has version 1.0	Here Dataweave has version 2.0 with few changes syntax
Mule 3 Support MEL (Mule Expression Language) and Dataweave (Write Dataweave functions outside Transform Message component using dw("") expression)	Mule 4 has Dataweave everywhere

MIGRATION:

```
java -jar mule-migration-assistant-runner-0.5.1.jar -muleVersion 4.1.5
-projectBasePath <<mule 3 project path> > -destinationProjectBasePath
<< mule 4 project path>> >
```

Will face lot problem while migrating 3 to 4
Message Enricher ———— Target specified for each component
filters —————— validation with try and catch block
Exception handling ———— Error Type (IDENTIFIER:NAME)
Here we will use class files
Ex: java.lang.exception

Here exception Strategies r there; on error propogate
On error continue

Mapping Exception(inbound Http)
Choice Exception(Multiple catch Exception)

Catch Exception(single block eException)
Rollback Exception

How do you Secure your APIs:

Transport Level you can secure your APIs using HTTPS listener configuration along with TLS (Transport Layer Security) or SSL (Secured Socket Layer)By providing Keystore or Truststore.

Keystore : Usually a JKS (Java Keystore) file created using Keystore explorer which is not signed.

Truststore : Signed certificate issued by Certificate Authority

API Level :

you can secure your APIs using Security Policies (we are using Basic Authentication policy) in API Manager.
[Before applying policies, we have to register our API with Runtime instance by deploying a Proxy or by using API Autodiscovery]

Tell me the steps to configuring HTTPS

SSL{Secure Sockets Layer} certificates are enable websites to move from HTTP to HTTPS, which is more secure.

Ans: One-way SSL

For normal SSL, on the server connector(Http Listener) we need a keystore where the servers certificate and private key reside. If we need one way SSL in http Request then we need to add trust store to call external HTTPS requests where trust store have external cer imported

Two Way SSL

Http Request

configure an HTTPS client connector with both client keystore and truststore. The client keystore shall contain the clients public certificate and private key. The client truststore shall contain the servers certificate.

HTTP Listener

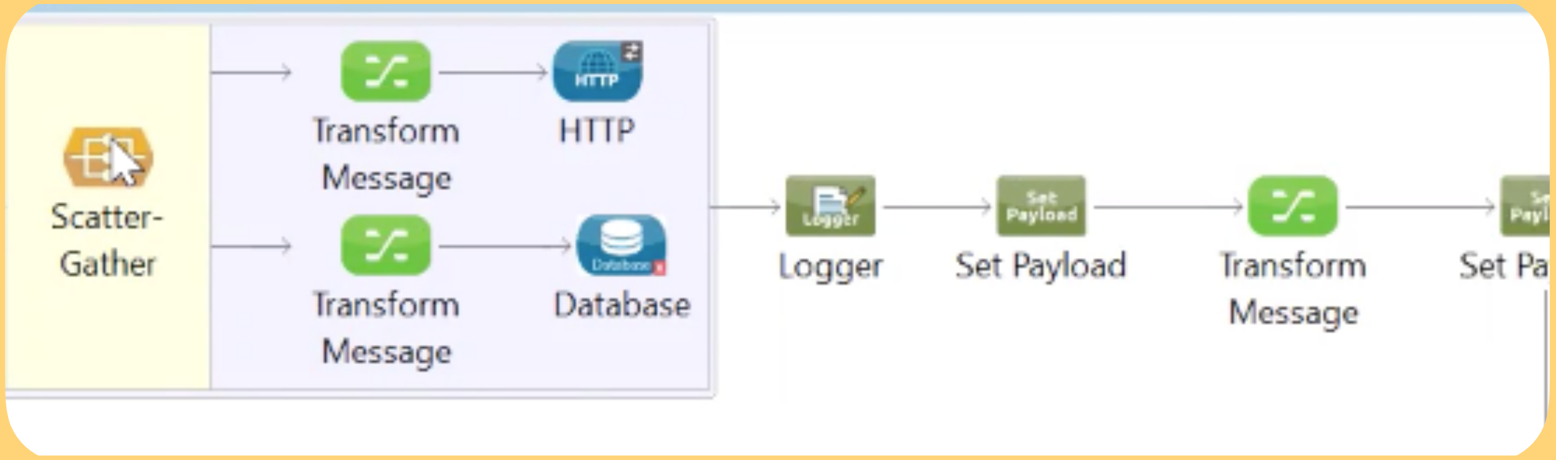
configure the server connector with both server keystore and truststore as well as set 'requireClientAuthentication' to 'true' on the 'tls-server' (i.e. truststore) attribute. This shall force the server connector to check client requests in the trust store prior to granting access.

REF: https://dzone.com/articles/configuring-https-mule

Scatter Gather

O/p: Array of Payload

It is used to transform messages parallel



Out of 2 roots if 1 fail it will throw composite routing exception will occur . To avoid this we will use Aggregation strategy

Aggregation strategy(Java Class)

It Consists of 2 methods

Event with Exceptions And Event without Exceptions

Integration Patterns

Fan-In

Fan-In will receive message from multiple sources

Ex:Composite Source

Fan-out

Fan-Out will send message to multiple sources

Ex:Scatter Gather

Pub-Sub In ActiveMQ we used this model

Ex:JMS

Mule 3	Mule 4
Mule Message: Payload: Properties (Inbound/Outbound): (Inbound properties are immutable) Attachments Inbound (Outbound):	Mule Message: Payload: Attributes: Attachments:
Types of Exception Strategies: • Mapping Exception Strategy (RAML generated along with API Kit Router) • Choice Exception Strategy • Catch Exception Strategy • Rollback Exception Strategy	Types of Exception Strategies: • On Error Propagate • On Error Continue
Look and Feel of Anypoint Studio (6): • If a place a connector example (HTTP) in source it acts a Listener, if I place it in Source it acts a Requester • It's not possible to read File in between the flow, we will have to use custom connector like Mule	Look and Feel of Anypoint Studio (7): • All Connectors are pre build in a Modularized a way. • Each Connector has individual components for all the operations.
Here Dataweave has version 1.0	Here Dataweave has version 2.0 with few changes syntax
Mule 3 Support MEL (Mule Expression Language) and Dataweave (Write Dataweave functions outside Transform Message component using dw("") expression)	Mule 4 has Dataweave everywhere

Dataweave

It is a functional programming languageDataWeave is used to transform message either from 1 format to another or to enrich a message in same format.

https://docs.mulesoft.com/mule-runtime/3.9/dataweave-operators

Filter:

Filter is used to Filter records from array with matching conditions

Map:

To iterate over an array

$ = current value
$$ = index

Var:

To create global variables inside data weave

Size of:

To calculate size of an array

Order by:

To sort an array in ascending or Descending(-)

Map object:

To iterate an object key by key

$ = current value
$$ = key

Join by:

Concatenating values in array using common Delimiter

Split by:

Split values in array using common Delimiter

Pluck:

To extract key list or value list from an object

```
objectPath pluck $$ - Extract Keys
objectPath pluck $ - Extract Values
```

Lookup:

used to call private flows from inside data weave by passing flow name and payload as arguments.

Modules:

to enhance reusability in dwls we created a modules which will have all common functions and variables so that we import these modules in any data weave and reuse.

Flatten:

Flatten is used to Merge multi-level array into single-level array

```
%dw 2.0
output application/json
---
flatten((payload pluck $$) + (payload pluck $))
```

Reduce:

Used for Summation of items in an Array

```
[10,20,30,40,50,16,11,14] reduce ((item, accumulator) ->
accumulator + item)
[10,20,30,40,50,16,11,14] reduce ($ + $$)
```

Format:

```
%dw 2.0
output application/json
---
now() as String {format: "dd-MMM-yyyy"}
```

ZIP:

zip to rearrange pairs of long arrays

Externalise Data-weave Modules:

Create Folder in src/main/resources
Inside that you can create a file—> Right click new —> file
You can keep all u r reusable functions or code

when ever u want reuse u can import

Import * from my modules:: filename

If u want give any alias name to the function u can use ‘as’

Functions:

By using fun keyword we can create function.

Fun name(arg1, arg2)

IN data Weave core module is there in that lot of functions are there.

UPPER:
LOWER:

IN data Weave string module is there in that lot of functions are there.

U need to import

Import dw::core::strings

EX: strings::camalize(siva)
                    Capitalize
                    Pluralize
                    Singularize
                    Underscore——> where ever space is there in between words _ will come
                    Dasherize

Types of Variables:

Variables are temporary memory locations used to store some information, and expire after their scope has ended.

1.Flow Variables:

Are Accessible in the defined Flow also in the Subsequent flows where flow references are going.

2.Session Variables

Are Accessible across Applications throughout the session.

3.Record Variables

Are useful only in Batch Processing

Batch Processing

Batch Processing is used to process large number of records simultaneously.
Batch Job can be triggered either by (Batch Execute - Manual) or (Poll – Based on Frequency)

In Batch Processing 4 stages are there

Input

To Configure Connector or Poll

Load And Dispatch

It Will Prepare Records

Process

In Process we are using Batch Step is there

In Batch Step 2 options are there

Accept Policy

All Records

No Failure Records

Only Failure Records

Accept Expiration

Matching Conditions to Filter Records

On Complete

Here we will get complete Batch Job Result

Like Total Records, Failed Records, Successful Records

Batch Job is Asynchronous.

Cost center data syncup:
SAP ==> Workday
1. Get Company Code List by SAP BAPI Call BAPI_COMPANYCODE_GETLIST
2. Transform Company Code List from SAP as an Array
3. Trigger Batch Job to iterate over Company Codes using Flow reference
4. For each Company Code Get Cost center List using BAPI_COSTCENTER_GETLIST
5. Trigger Batch Job for each Cost Center to Push it to Workday

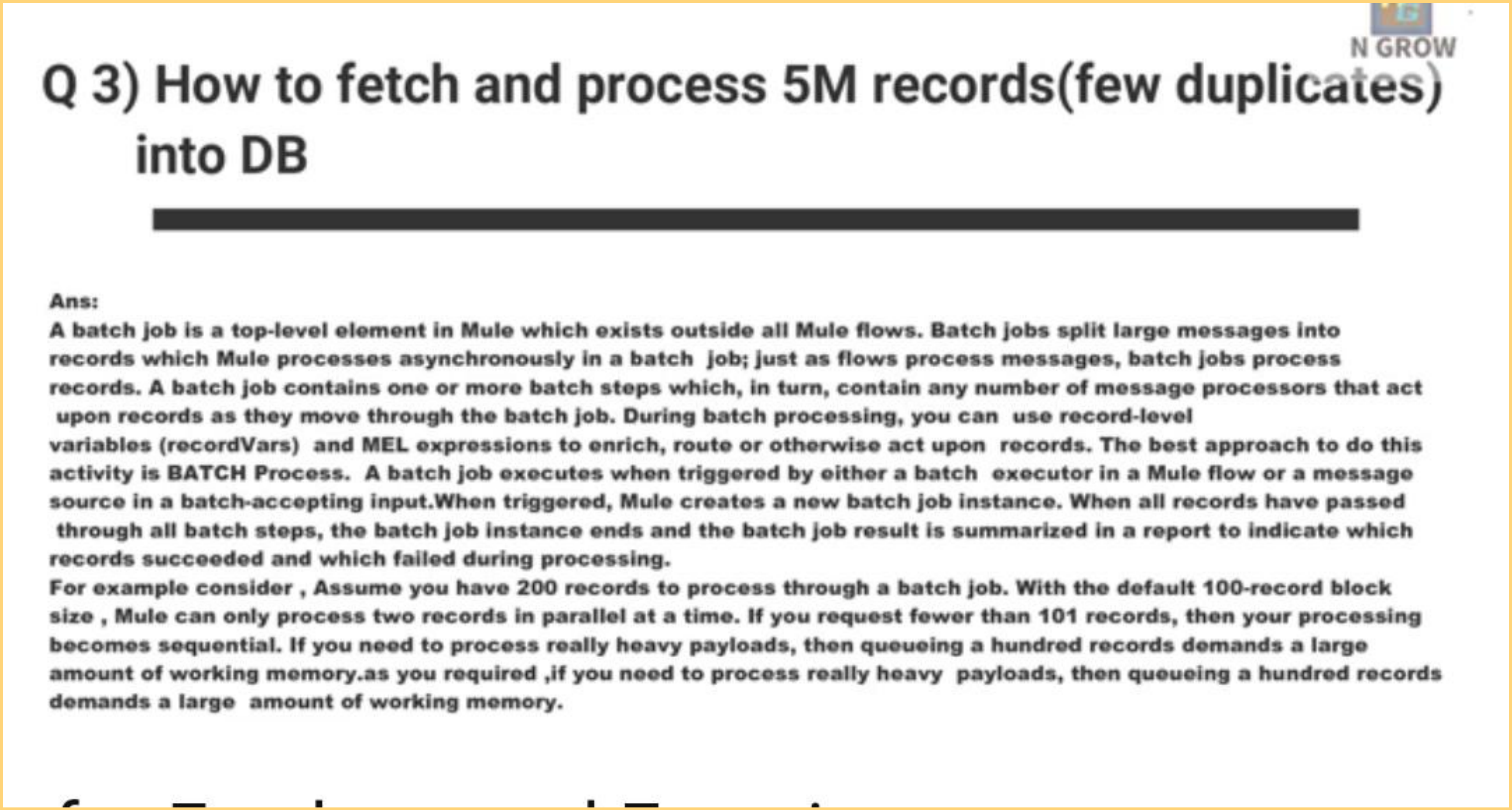
Max Failed Records:
-1 : Continue to Process Records even if N number of them are failing
0 : Fail/Stop Batch Job even if 1 record fails

Two Batch Steps:
1. Normal Batch Step (1st step)
    Accept Policy: Default (NO_FAILURES)
    Accept Expresion: No Use case
    (Processors) Business Case: Send Cost Center List of Each Company Code to Workday
    Aggregation: Can be used to process records input to Batch Step in Sub Groups
    Aggregation Size <= Batch Block Size

2. Failures Step (2nd Step)
First Batch Step will consider a record as a failed record if there are any exceptions while processing.
Accept Policy: ONLY_FAILURES
Capture Exceptions: #[Batch::getStepExceptions()] or #[Batch::failureExceptionForStep(StepName)]
Accept Expression: No Use case
Aggregation: Generate a CSV File and Store it in File System for reprocessing these records later point of time
Or
Send these Failed Records to a Message Broker (DLQ-Dead Letter Queue) to reprocess or report at later of time

3. OnComplete:
We are sending a HTML-Template based email at end of each Batch Completion about the status and metrics of batch job
by using batchjobresult object (totalrecords, failedrecords etc.)
For this we have prepared a HTML and are using Parse Template to dynamically generate HTML Body for Email.

<u>Planning:</u>
1. Identify Volume of Records
2. Do Some Math to calculate Batch Block Size
3. Number of Threads in Thread Pool= 16
Batch Block Size = Total Number Records / 16
Aggregation Size = Depending End System Capability and <= Batch Block Size
4. Increase Heap Size for Mule Runtime (in MULE_RUNTIME_HOME/conf/wrapper.conf) (If it's OnPremise), vCore Size (If it's cloudHub)
5. Number of Batch Instances Created =
If Total Number Records <= Batch Block Size = 1
Else Total Number Records / Batch Block Size



Q 3) How to fetch and process 5M records(few duplicates) into DB

Ans:
A batch job is a top-level element in Mule which exists outside all Mule flows. Batch jobs split large messages into records which Mule processes asynchronously in a batch job; just as flows process messages, batch jobs process records. A batch job contains one or more batch steps which, in turn, contain any number of message processors that act upon records as they move through the batch job. During batch processing, you can use record-level variables (recordVars) and MEL expressions to enrich, route or otherwise act upon records. The best approach to do this activity is BATCH Process. A batch job executes when triggered by either a batch executor in a Mule flow or a message source in a batch-accepting input.When triggered, Mule creates a new batch job instance. When all records have passed through all batch steps, the batch job instance ends and the batch job result is summarized in a report to indicate which records succeeded and which failed during processing.
For example consider , Assume you have 200 records to process through a batch job. With the default 100-record block size , Mule can only process two records in parallel at a time. If you request fewer than 101 records, then your processing becomes sequential. If you need to process really heavy payloads, then queueing a hundred records demands a large amount of working memory.as you required ,if you need to process really heavy payloads, then queueing a hundred records demands a large amount of working memory.

Foreach

1.Synchronous :
IT means Record By Record even 1 record fail it will stop process

2.It is used for small data set

3. It is private Scope

Batch Processing

1.ASynchronous :
IT means Record By Record even 1 record fail it will stop not process

2.It is used for Large data set

3. It is public Scope

Error Handling (Mule 3 and Mule 4)

Mule 3:

- **Types of Exception Strategies:**
  a. Mapping Exception Strategy
  b. Rollback Exception Strategy
  c. Choice Exception Strategy
  d. Catch Exception Strategy

- **Exceptions are identified as Java Exception Classes**

- **When there is an Error, Exception Strategy that is configured to the Flow/ Application (Global) will be executed and stop message transaction**

- **If you want to ignore Exceptions in Mule 3,**
  Identify the Connectors/Components for which you want to ignore error.
  Move those into a Private-Flow (Because you can write flow specific exception strategies).
  Define Exception Strategy for Exception Classes, for which want to ignore errors.
  Once the Exception Strategy of that particular flow is executed,
  Since Error is already handled, it will go back to the Next Message Processor from where this Flow was called.

- **If you want to throw Custom Errors you will have to use either Groovy Script or Java Class to throw custom Errors**
  Example: throw java.lang.IllegalArgumentException("This is an Invalid Parameter")

Catch Exception Strategies:

It is a single block Exception Strategy where all types of errors will be handled in same manner.

Choice Exception Strategies

To handle errors conditionally using list of catch Exception Strategies

Mapping Exception Strategies

Mule 4:

- **Types Error Handling:**

Basically If we will get any exception it will check in local error handler if I dint catch any logic to catch that exception then mule default error handler comes into picture it will stop the execution of process and it will logs the info on the console.

When an error is thrown an error object is created.
For Every object we have properties
ERROR.DESCRIPTION
ERROR.TYPE

Error types are identified by namespace & identifier
VALIDATION:INVALID_BOOLEAN

- **a. On Error Propagate:**
- If there is any error in the Flow or Application where On Error Propagate, It will Catch the Error, Execute Error Logic and Stop that Message processing.

- All Processors in the error handling scope are executed
- At the end of the scope
- —> the rest of the flow that thrown the error is not executed
- —-> the error is thrown upto the next level and handled there
- —> An http listener returns an error response.

- b. On Error Continue
  If there is any error in the Flow or Application where On Error Continue, It will Catch the Error, Execute Error Logic and Continue with next component onwards.

- Flow A >>> is Calling >>> Flow B , if you want to continue in Flow A When there is error in Flow B, then you should use On-Error Continue in Flow B.
  If On Error Continue is used in Flow A, Whoever is calling Flow A will ignore the Error occurred in Flow B, But it won't continue executing Flow A.

- All Processors in the error handling scope are executed
- At the end of the scope
- —> the rest of the flow that thrown the error is not executed
- —> the event is passed upto the next level as if the flow execution had completed successfully.
- An http listener returns an Successful response.

- If you want to Throw Custom Errors, you can use Raise Error component
  For Raise Error, configure ERROR_TYPE => NAMESPACE:IDENTIFIER (Example: MYAPI:RECORDS_NOT_FOUND)

- Try/Catch - On-Error Continue, it will Ignore the Error and Continue executing same flow itself

- Global Error Handler:
  Create Seperate Mule Configuration File, Define Stack of On-Error-Propagate or On-Error-Continue depending on requirements.
  Go to Global Elements in and Create a Configuration with Default Error Handler, by selecting the name of Error Handler.

Framework (Global Error Handler, Common Utils, Reusable Frameworks):

- Define Error Handler in a Separate Mule Project
  Example: Common-utils
- Build Common-Utils as a JAR (mule-plugin)
  <classifier>mule-plugin</classifier>

- Add this Common-utils JAR as a Dependency in the APIs
- In Global Elements of API, Create a **Import** configuration with **XML-Name** that is defined in Common-Utils
- Create a Configuration with Default Error Handler, that is available from Common-utils JAR.

S

Munit

Before mule 3.7 there is no Munit. Only JUNIT was there it is very difficult to non java people. In mule 3.7 they introduced MUNIT.

Unit testing & Integration testing

In unit testing we will do only perticulr
 Flow & will mock

We need to create test suit.
Every test contains 3 parts.
1.Behaviour
2.execution
3.validation

How to create Munit test suite?
Right click on the flow
Select M-Unit
Create a blank test for this flow.
And it is located in src/test/munit

U can use setevent

If u want validate u r test u can use assert that operation
If u want verify any processor is executed or not u can use verify call
And pick processor and select the processor

If u have any repeated to u need to externalise that one by using BEFORE TEST operator

If u want mock anything u can use MOCK WHEN and pick processor

You have 3 flows u want to ignore 2 flows & u need to execute only 1 flow

U can use IGNORE TEST

Right click on test
Click on ignore test

1 more way is

Click on the test
U can see option like
Ignore test by default it is false make it true.

LOGS:

Audit Logs:

Audit Logs Will be IN Runtime Manager

Accessmanagement contains audit logs.

It contains user interaction with any point studio like when u logged in & logged out.

In audit logs u can see

1. **When app is started**
2. **When app is stooped**
3. **When app is deleted**
4. **When app is redeployed**
5. **When flow is started, stoped**

These audit will keep 6 years in any point studio.
By default mule runtimes shows info level logs.
Debug or trace level log messages or ignored.
If you want see those logs u need to configure log4j2.
Log4j2 supports both synchronous & asynchronous

System logs:
Specific to mule runtime
Configured by log4j2.xml
Contains log messages about mule runtime lifecycle{startup & shutdown}

Application Logs:

Specific to mule application
Configured by log4j2.xml

It contains all log messages generated inside the mule application.

Third party log systems:
You can use spunk or ELK plugins in Runtime Manager :
Use the runtime manager agent to exchange logs & analytics data with Spunk & ELK

GO to Run time manager—> click on server—>manage servers

Click plugins

Select events like tracking or debugging
Enable elk or splunk

It will ask host, user name, password.
In Elk it will ask log file

JMS:

When we are going to work with JMS
We usually publish a message with either queue or topic

Queue: one to one
Topic: one to many

Configuration:

JMS jars : needs to be uploaded
User name: admin
Password: admin

Click on factory configuration
Edit inline
Enter

Broker url : tcp://localhost:61616

Provide Destination : from which Queue its should listen

PUBLISH:
Connector configuration:
If u working with activemq select ActiveMq connection
If u want work with any other MQ use generic connection

FTP:

FTP is used to connect remote locations.

FTP CONFIG

WORKING DIRECTORY
host: local host
Port : 21 default
User name:
Password:

Next file path:

Schedular:

Rate-limiting: Restrict number of requests for API to accept per unit time, If requests are rejected during ratelimiting we cannot reprocess it, it will simply reject.

Throttling: Maximum number of requests an API can accept, if number of requests exceeds the throttling limit, it'll queue the excess request for retry

GET

BY USING THE GET METHOD WE WILL GET TOTAL DATA AS A RESPONSE IN OUR APPLICATION OR DATABASE
In get method no there is no bodyGET requests can be cached

1.GET requests remain in the browser history
2.GET requests can be bookmarked
3.GET requests should never be used when dealing with sensitive data
4.GET requests have length restrictions
5GET requests is only used to request data (not modify)

POST

POST is used to send data to a server to create/update a resource.

POST requests are never cached

POST requests do not remain in the browser history

POST requests cannot be bookmarked

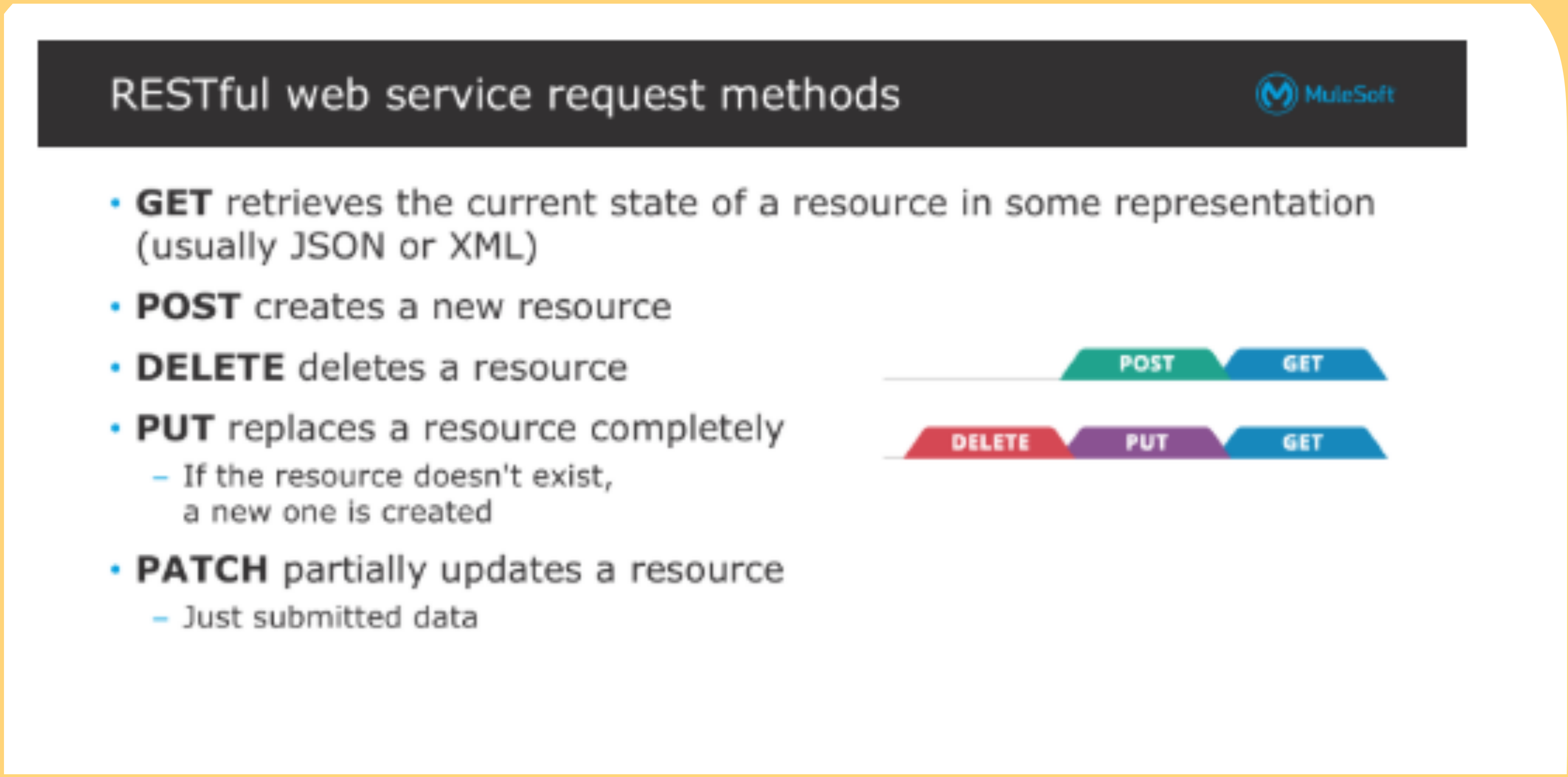
POST requests have no restrictions on data length

PUT

PUT : it is used to display message as a response to the user. In Put method there is body

PUT is used to send data to a server to create/update a resource.

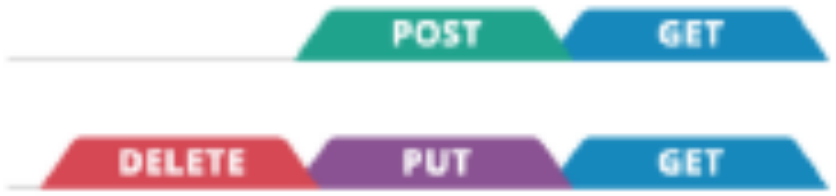
The difference between POST and PUT is that PUT requests are idempotent. That is, calling the same PUT request multiple times will always produce the same result. In contrast, calling a POST request repeatedly have side effects of creating the same resource multiple times.

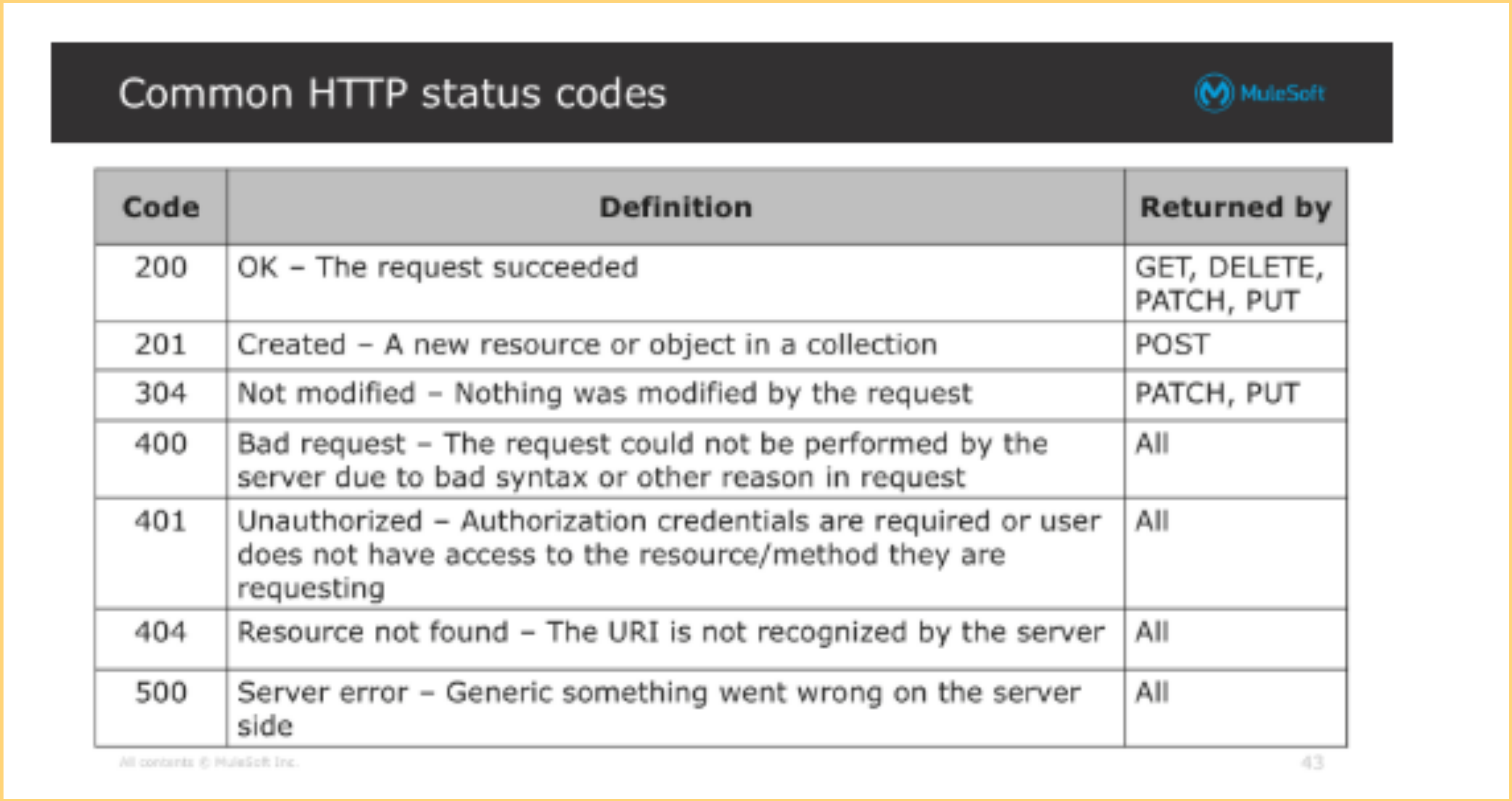


RESTful web service request methods

MuleSoft

- **GET** retrieves the current state of a resource in some representation (usually JSON or XML)
- **POST** creates a new resource
- **DELETE** deletes a resource
- **PUT** replaces a resource completely
  - If the resource doesn't exist, a new one is created
- **PATCH** partially updates a resource
  - Just submitted data





Common HTTP status codes

Code	Definition	Returned by
200	OK – The request succeeded	GET, DELETE, PATCH, PUT
201	Created – A new resource or object in a collection	POST
304	Not modified – Nothing was modified by the request	PATCH, PUT
400	Bad request – The request could not be performed by the server due to bad syntax or other reason in request	All
401	Unauthorized – Authorization credentials are required or user does not have access to the resource/method they are requesting	All
404	Resource not found – The URI is not recognized by the server	All
500	Server error – Generic something went wrong on the server side	All

After consuming rest, exactly what you do?

First i will verify the Status code that is returned by the External system and if it is valid success response the i will check if my return type is not Same as the response what I have received I will convert it using transform Message and send it back to the Listener

Is Rest API is good are Soap?

When it comes to Performance and simplicity to get the info it is REST
When it comes to in built standards and security then SOAP is the best option
REF: https://smartbear.com/blog/test-and-monitor/understanding-soap-and-rest-basics/

What are other ESB's other than Mule

1)Apigee
2)Tibco
3)Dell Boomi
4)IBM ACE
5)Talend
6)Oracle Service Bus

===

=============

1) What are different Processing strategies

. synchronous

. queued asynchronous

. non blocking (3.9)

2)What are the scopes in mule

A sync

messageEnricher

untilSuccessful

CompositeSource

batch

Commit

request-reply

Object Store

Object stores are used to store an object. Mule uses object stores whenever it needs data persistent for later retrieval.

Object Store Operations

The object store connector can perform various operations like

Contains,
Remove,
Retrieve,
Retrieve all keys,
Retrieve and Store and
Store.

- **Contains** is used to verify if given key is present in the object store.
- **Remove** is to remove the respective key from the object store.
- **Retrieve** is used to retrieve an object from the object store and make it available in the specified property scope of a Mule Message.
- **Retrieve all keys** will return a list of all keys present in the object store.
- **Retrieve and Store** is used to retrieve and store an object in the same operation.
- **Store** is used to store the object in an object store.

Types of Object Store

- In-memory store: This allows you to store objects in memory at runtime. Generally, objects are lost when runtime is shutdown.

- Persistent store: Mule persists data when an object store is explicitly configured to be persistent. Mule creates a default persistent store in the file system.

Object Store Limitations

Object Stores are not a universal solution for data storage. They do not replace a database, and they are not suitable for every use case. Most importantly, they do not support transactional access or modification.

What Is API Autodiscovery?

API Auto-discovery to *pair an API in API Manager to its deployed Mule application*.
It allows API Manager to manage the API. When set up correctly, it shows the API status represented as a green dot if the API is being tracked (*active*) or a gray dot if untracked
By configuring autodiscovery, you can use policy management and API Analytics accessed from within API Manager.

https://dzone.com/articles/using-api-autodiscovery-in-anypoint-platform

https://dzone.com/articles/using-api-autodiscovery-in-anypoint-platform

JWT

A signed JWT token consists of three parts: the header, the payload, and the signature.

JWT header

The header essentially tells you whether or not a JWT is signed. If the token is not signed, the value of would be “none”. If it is signed, the value would indicate the type of algorithm used to sign the token,

JWT payload

Depending on the type of token, the payload may contain different types of information, which we call claims. For instance, if a JWT is a id token, then the claims may contain information about the user including the username, id, email, name etc … If the JWT is an access token, the claims can also include scopes, roles, audience, etc …

Run Time Manager:

Runtime Manager is the Anypoint Platform tool used to deploy and manage all of your Mule applications from one central location, whether your apps are running in the cloud or on-premises.

https://dzone.com/articles/managing-mule-schedules-with-anypoint-runtime-mana#:~:text=Runtime%20Manager%20is%20the%20Anypoint,the%20cloud%20or%20on%2Dpremises.

API Manager:

API Manager is a component of Anypoint Platform for designing, building, managing, and publishing APIs. Anypoint Platform uses Mule as its core runtime engine. You can use API Manager on a public cloud, such as CloudHub, a private cloud, or a hybrid.

https://www.mulesoft.com/platform/api/manager

MuleSoft API Manager Policies

Rate-limiting: Restrict number of requests for API to accept per unit time, If requests are rejected during rate-limiting we cannot reprocess it, it will simply reject.

Throttling: Maximum number of requests an API can accept, if number of requests exceeds the throttling limit, it'll queue the excess request for retry

JWT (JSON Web Token) Validation - https://lnkd.in/e57kcPs
OAuth 2.0 Implementation Using Mule OAUTH2 Provider In Mule 4 - https://lnkd.in/eeRYsEN
Basic Authentication – Simple - https://lnkd.in/euPpjeM
Basic Authentication – Client ID enforcement - https://lnkd.in/ehqHHew
Basic Authentication – LDAP - https://lnkd.in/eDKQtR4
Spike Control - https://lnkd.in/eFqYkfh
Rate Limiting – SLA based - https://lnkd.in/e9BjbKu
Rate Limiting - https://lnkd.in/eEKniPD
Message Logging - https://lnkd.in/eD3bYPx
XML Threat Protection - https://lnkd.in/e9ipqDX
JSON Threat Protection - https://lnkd.in/e7sEN48
HTTP Caching - https://lnkd.in/excvwUz
Header Removal - https://lnkd.in/ePi_8_3
Header Injection - https://lnkd.in/e9dmnSN
IP Whitelist - https://lnkd.in/emSJ8aq
IP Blacklist - https://lnkd.in/eCdgYBh

#APIManager #Policies #MuleSoft #Mulesy #mule4 #oauth

CloudHub: L

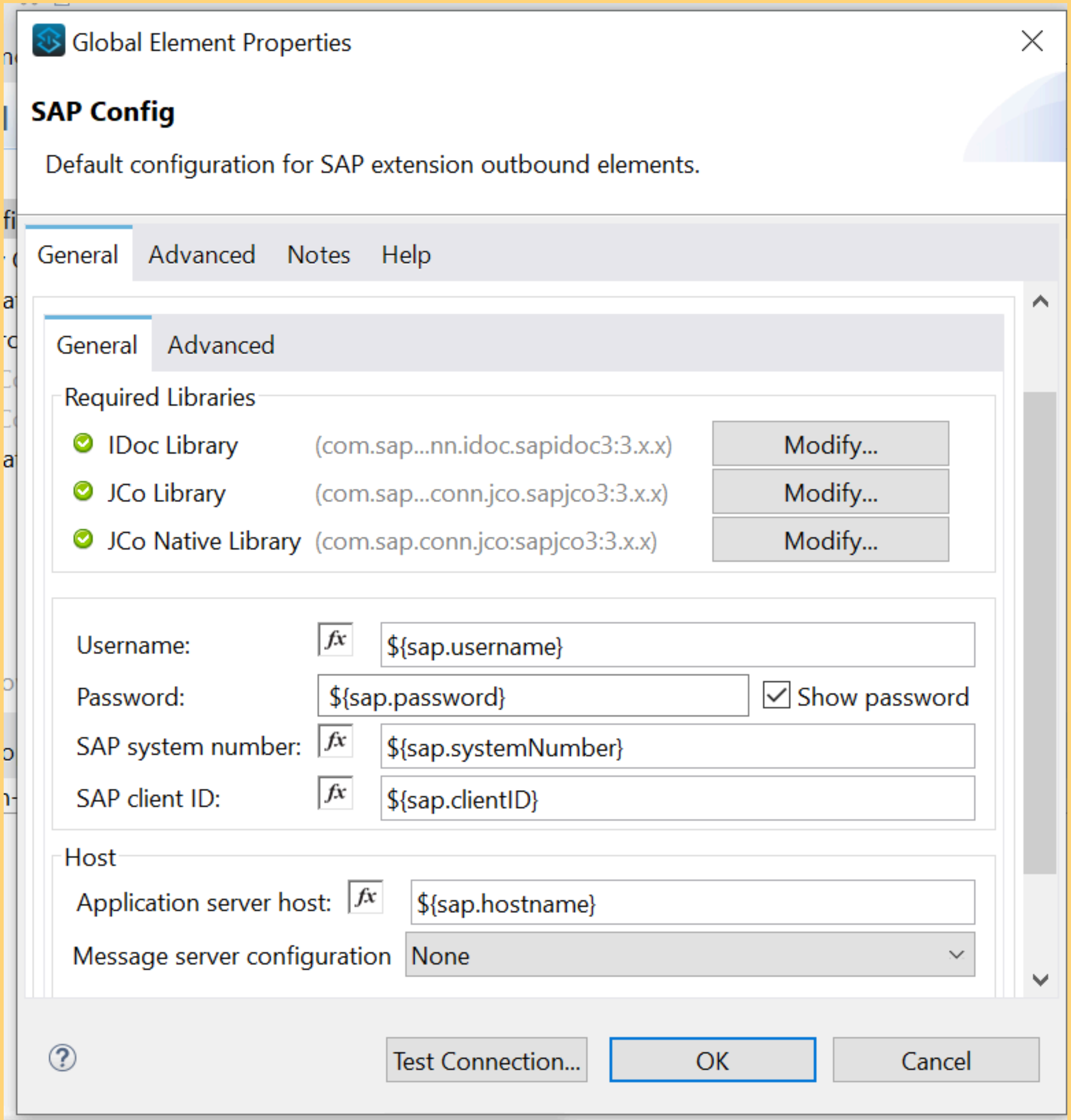
Custom Logging:

1. **Elasticsearch - Logstash - Kibana (ELK), Splunk, SumoLogic**
   If you use Logstash, you can log details using the normal Logger component. Because in Logstash, while pushing data to Elasticsearch you can convert normal Pattern Log to JSON message

   Once JSON message is pushed to Elasticsearch, Kibana Dashboard tool can pick data from Elasticsearch automatically (.yml configuration has Elasticsearch URL)

   Elasticsearch - Kibana (EK)
   a. Since Elasticsearch Exposes REST API, you can push Log data from flow in Async scope to Elasticsearch using HTTP Request/connector.
   b. Change Log Pattern to JSON Layout in Log4j2.xml and use Custom JSON Logger to print logs in JSON format.
   Add Socket Appender to Send Log messages to Elasticsearch.
   Log Messages can be tracked using **CorrelationID** passed as part of custom logger component

SAP CONNECTOR



Global Element Properties

SAP Config

Default configuration for SAP extension outbound elements.

Required Libraries

✓ IDoc Library	(com.sap...nn.idoc.sapidoc3:3.x.x)	Modify...
✓ JCo Library	(com.sap...conn.jco.sapjco3:3.x.x)	Modify...
✓ JCo Native Library	(com.sap.conn.jco:sapjco3:3.x.x)	Modify...

Username: ${sap.username}

Password: ${sap.password} [✓] Show password

SAP system number: ${sap.systemNumber}

SAP client ID: ${sap.clientID}

Host

Application server host: ${sap.hostname}

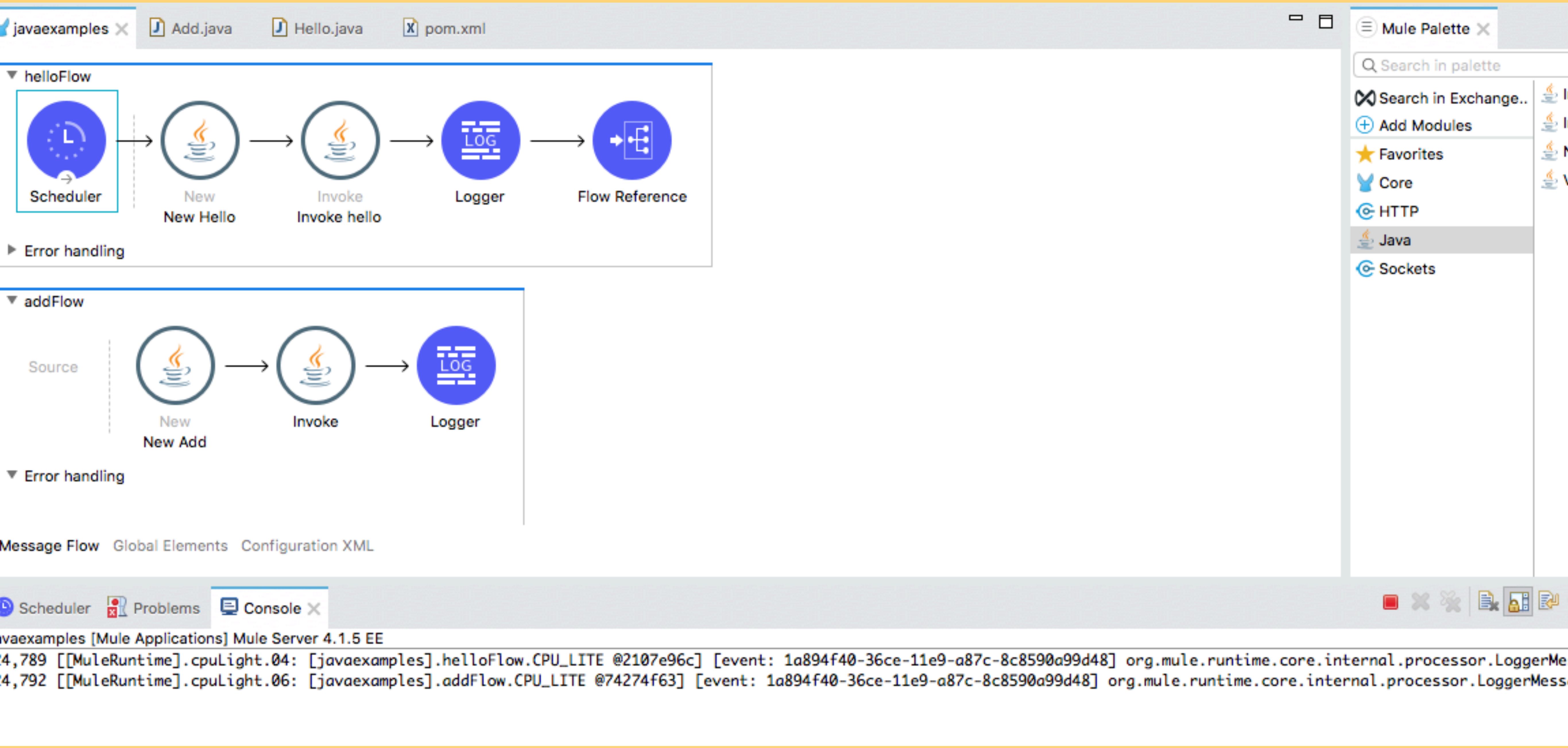
Message server configuration: None

Invoking java Methods

Java methods (either instance or static) can be called through the invoke and invoke static operations in the Java module.

Invoke Methods with DataWeave

The Java module also has a DataWeave function (`Java::invoke`) to provide the same functionality as the `invoke` operation but inside a DataWeave expression.



```
vaexamples [Mule Applications] Mule Server 4.1.5 EE
:4,789 [[MuleRuntime].cpuLight.04: [javaexamples].helloFlow.CPU_LITE @2107e96c] [event: 1a894f40-36ce-11e9-a87c-8c8590a99d48] org.mule.runtime.core.internal.processor.LoggerMe
:4,792 [[MuleRuntime].cpuLight.06: [javaexamples].addFlow.CPU_LITE @74274f63] [event: 1a894f40-36ce-11e9-a87c-8c8590a99d48] org.mule.runtime.core.internal.processor.LoggerMess
```

ESB

ESB stands for Enterprise Service Bus which is basically a middleware tool for integrating various applications together over a bus-like infrastructure.

ESB Functionality − VETRO abbreviation is generally used to summarize the functionality of ESB.

- V(Validate) − As the name implies, it validates the schema validation. It requires a validating parser and up-to-date schema. One example is an XML document confirming to an up-to-date schema.

- E(Enrich) − It adds additional data to a message. The purpose is to make message more meaningful and useful to a target service.

- T(Transform) − It converts the data structure to a canonical format or from a canonical format. Examples are conversion of date/time, currency, etc.

- R(Routing) − It will route the message and act as a gatekeeper of the endpoint of a service.

- O(Operate) − The main job of this function is to invoke the target service or interacts with the target app. They run at the backend.

VETRO pattern provides overall flexibility to the integration and ensures that only consistent and validated data will be routed throughout the ESB.