1. How many APIs you have created in overall Mule Experience. List down all the APIS. -- they wanted to know the count.

Although there are many api’s created in the Mule learning experience, I have created 2 API’s in the Airline management project. One is Master data syncup, here in this API we get an Object name from amazon SQS and fetch the objectName from S3 bucket where flight information is present, transforming the data accordingly, and uploading the failed records and new records to S3.

Another API is the daily flight scheduler which does a little similar process but gives daily flight status to other applications like KAFKA.

Apart from the project related in my training and POC’s I have worked and implemented API’s on DynamoDB, Ecommerce related applications.

2. Can you explain some RAML that you have developed -- they wanted actual business cases RAML.

One of the RAML I have worked on is Airline Management master syncup RAML,

Here based on the business requirement, we created a RAML.

Let me explain little about the business requirement and elucidate how we implemented this.

So here, the business requirement was to fetch and post information about flights based on the origin and destination.

For this, we created a resource called “flights”, and to this flights resources, we added methods like POST, PATCH, GET.

Every method will have these 3 as basic skeleton, Body, responses, and examples.

In our POST method, we a give a description about the resource what its action is

In the body, we declare the type of resource, In this we can either directly write in the format, the type should be or we can use Datatypes.

In general, we have Traits, Datatype, Example, Extention etc as specification in RAML, This allows us for reusability and cleaner code.

In the responses, we can use the appropriate Response code which we are expecting if it passed or failed.

In general we used 200 status for pass and 400 for failed method. Again we can include datatypes, examples, in the body for these status codes.

Like wise we created other resources based on the business requirement.

Coming to the over all structure of RAML, we contain Title, Declaration Types, like variables, then resources and their corresponding responses.

In general, for fetching elements or records we use GET operation, we can add queryParameters to identify the record, using their attributes which we can declare in the GET method,

For example in our project in the RAML, when assigned GET method to the flights resources,

We used displayName attributes to get the information about that method,

Assigned queryParameters for date, declared the parameter type, and specifying whether that parameter is required or not by using required attribute which takes in Boolean value, by default it is false.

For this queryParameter, we assigned added a response by the same method.

Likewise we can create as many resouces as business is required. This is the Skeleton of the Mule Flow and describes the way to implement and easily understand the requirement.

3. How many APIs are running in production that you have developed.

4. Explain AirLine Management Mule implementation

I worked on a Project called Airline management, this project includes Flight Master Data Sync-Up and Daily Flight Schedule Updates. Each module we are maintaining the process and system layer as part of API-Led connectivity, however the experience layer is not part of the first version.

As part of this module, we will be receiving an AWS SQS message for the S3 object placed, as part of business implementation, we are transforming and grouping the planned flight level information in the process layer. Secondly in the system layer we are storing the data in various DB by using stored procedures. Additionally we are creating Success/Failed record files in local and uploading as S3 object in AWS. This module is completely developed in Mule.

We are reusing the same concepts in this module however we are adding another feature in the process layer that is publishing flight information to Confluent KAFKA Topics per airport and connecting Kafka with React for UI real time streaming about the flight live status. To put it in short, here in this module, we were integrating different applications that is MuleSoft, Kafka, and React.

6. What new thing I did in AirLine Management.

The new things I have learnt and implemented in Airline management are:

Amazon web services like S3, SQS, DynamoDB, KAFKA. Most of all Websockets

7. What thing I reused in projects

In the Airline management project, between the two Master data syncUp and Daily flight status modules, we have utilized DWL functions which mainly implemented the data orchestration appropriate to the business model. Apart from that, in the flow level, we have utilized exception handling mechanisms.

8. How did you manage your RAML dataTypes

RAML datatypes can be used either isolating from the mail RAML or implementing in the same RAML. However, it is better to use separate RAML because to solve reusability and cleaner code. Here we managed RAML datatypes by creating separate RAML files and invoking them using include functionality.

Here this type of implementation is called ChildRAML specification where we are calling RAML away from main flow wherever and whenever that type is required.

9. Development Best practices that you followed.

The best practices are the way the business model works as required. Here, in my project starting from the very skeleton of the application that is from RAML, we have implemented child ramls along with RAML validations for cleaner code and sustainable code. Later in application perspective, we have used global properties for configuring the values of connectors.

[**KP:** We have segregated the implementations in the process layer and system layer]

10. How many connectors that you used.

In the project, we have used around connectors such as JDBC connectors, S3 connectors, SQS connectors and HTTP connectors. Explain them in detail.

11. Have you used Salesforce connectors?

Yes, this is not a part of the project, but yes for my practise and a POC purpose I worked on salesforce connectors.

***Can you explain me what have you worked on***

Basically salesforce connectors enables us to create apps that react with salesforce events like creating, adding updating or deleting objects or records. Basic idea was to query some records from salesforce and inserting few customer records in the contacts topic/event. I do not know the exact details of the POC but I had to deliver this task of query and inserting in salesforce.

However, I found it very similar to the way we connect to any other connector. Here we need have the information of the secret key and domain name.

**DWL Functions - Self Questioning**

**From 3.5 DWL came**

**It is JSON like language**

**Mule performance by using DWL increased**

**--- => separator**

*In the project, I have used dwl functions like*

++ => for concatenation

-- => for removing elements it identifies the object

abs => return the absolute value ( +ve numbers)

avg => returns the average of the list of values

**ceil => rounds the nearest number (if decimal)**

**contains=> returns boolean (True if present and false if not)**

**$ => returns the index**

**$$ => returns the value**

**& => clubs the list of elements having same tag**

**distinctBy=> returns unique**

**entriesOf => returns key,value pair and if any attributes are present**

**Arun’s Questions:**

**121. What is XSLT**

**XSLT stands for extensible styling languague transformation. Mostly this is used for converting or transforming XML into XHTML or any other form. XSL is the styling sheet which helps in the conversion process.**

**122. How to use an existing Java code in Mule.**

**Lets say there is a JAVA code which is having some transformations or some functionality and we would like to use it as it is. Obvious thing to implement in mule is we can either convert the whole functionality into DWL and use in transformer.**

**However, if the business requirement mandates the JAVA code as it is or there is a lots of code to convert which can be cumbersome or for interoperability we can use JAVA module.**

**How do we create this**

**First we have to create a New Java instance, in the src, main, JAVA, we can define this by creating a class and using the existing JAVA code.**

**Additionally we have to declare the dependencies. In POM.xml**

**However, in Mule 4 we got these dependencies and required options directly in the mule palette. Thus we can create new and invoke them.**

**12. How to use the VM connector**

**JAVA Modules**

**Lets say there is a JAVA code which is having some transformations or some functionality and we would like to use it as it is. Obvious thing to implement in mule is we can either convert the whole functionality into DWL and use in transformer.**

**However, if the business requirement mandates the JAVA code as it is or there is a lots of code to convert which can be cumbersome or for interoperability we can use JAVA module.**

**How do we create this**

**First we have to create a New Java instance, in the src, main, JAVA, we can define this by creating a class and using the existing JAVA code.**

**Additionally we have to declare the dependencies. In POM.xml**

**However, in Mule 4 we got these dependencies and required options directly in the mule palette. Thus we can create new and invoke them.**

**Deployment:**

**Standalone: we need to have the standalone wrapper. Once we have that go to**

**RunTime Manager : by creating application and servers and invoking those servers**

**HTTP:**

**How do you declare in the Arguments**

**In the arguments section use this as env variable**

**-Dmyenv=dev**

**These are the place holders, if we want to send any environment variables then we will be passing as VM arguments using -D option as the extension**

**secure properties:**

1. **get from exchange**
2. **In global search for secure properties**
3. **filename = give the yaml**
4. **key = give the key**
5. **encryption algo : AES**
6. **Mode = CBC**

**secure properties jar => cmd => java file encryption => choose the above appropriately => encrypted file will be created => place in main/rsrcs/ => prefix ${secure::hvkjhv} => OK**

**Note : values must be inside “”**

**MuleDomains :**

**To avoid monolithic application, we use this**

**only common global elements should be kept**

**go to properties -> change to domain project -> remove redundant dependencies from pom**

**12. How to use the VM connector**

**VM connector manages intra-app and inter-app communication through either transient or persistent asynchronous queues**

**o   Transient queues are faster than persistent queues, but they are not reliable if the system crashes.**

**o   Persistent queues are slower but reliable.**

**How do we do it?**

**DWL for XML:**

**whatever we should declare a tag**

**we need a root tag**

**product:{**

**}**

**Inside this there will be things like the product.**

**product @(pid: payload.productid):{}**

**Externalize the DWL:**

**drag the dropdown,**

**unselect the inline**

**select the file and save as a file**

**We can see that DWL will be stored in resources section of project folders**

**We can even set multiple DWL functions using create operators**

**For complex dataweave transformations**

**array = []**

**dw format is same as json except “”**

**map = iterate over the payload items**

**$ = current element**

**creating a variable**

**it is done above the separator**

**evaluation**

**() => wrapper for objects => flattens and gives the merged output**

**filter**

**example: (myproduct filter $.product.brandName == “HP”)**

**Lets say we are expecting from queryParams**

**myproduct filter $.product.brandName == attributes.queryParams.bname**

**Sorting**

**orderBy**

**ascending orderBy $.op**

**descending orderBy -$.op**

**conversion**

**var conversionRate  = 70**

**set a variable and its price or some number**

**how to write a function**

**fun**

**fun getConvertedPrice(myprice) = myprice \* 70**

**In the DWL functions:**

**op: getConvertedPrice($.offer.offerprice)**

**Lambda functions:**

**var convert = (myprice) -> conversionRate**

**In DWL functions:**

**op : convert(product.offer.offerprice)**

**if**

**?**

**lookup(‘flowName’, “”)**

**lookup(“flowName”, attributes.queryParams.country)**

**Converting XML to JAVA**

**We need to be very precise on the type we are expecting**

**DataSense Graphics**

**XML will always be converted to stream of data if JAVA is specified.**

**KEY and Value based**

**key is the tag name value is based on the tag attributes**

**ns = namespace**

**\*  is the reference we use to separate from the tags and travenrse by each element**

**type: specifying our own types**

**116. Difference between map and mapobject in DWL**

**Map helps in iteration of the elements present in the payload - Array**

**MapObject in iteration of the elements returns along with the keys and values - Object**

**13. Where did you deploy your code -- cloud, on-prem**

**we deployed our application in on-premise. Our project Airline management is in nascent phase and I think once there are more requirements coming in, we will be deploying in cloudHub.**

**1. How many APIs you have created in overall Mule Experience. List down all the APIS. -- they wanted to know the count.**

**Although, there are many api’s created in the Mule learning experience, I have created 5 API’s in Airline management project where each API has its own functionality.**

**2. Can you explain some RAML that you have developed -- they wanted actual business cases RAML.**

**One of the RAML I have worked on is Airline Management master syncup RAML,**

**Here based on the business requirement, we created a RAML.**

**Let me explain little about the business requirement and elucidate how we implemented this.**

**So here, the business requirement was to fetch and post information about flights based on the origin and destination.**

**For this, we created a resource called “flights”, and to this flights resources, we added methods like POST, PATCH, GET.**

**Every method will have these 3 as basic skeleton, Body, responses, and examples.**

**In our POST method, we a give a description about the resource what its action is**

**In the body, we declare the type of resource, In this we can either directly write in the format, the type should be or we can use Datatypes.**

**In general, we have Traits, Datatype, Example, Extention etc as specification in RAML, This allows us for reusability and cleaner code.**

**In the responses, we can use the appropriate Response code which we are expecting if it passed or failed.**

**In general we used 200 status for pass and 400 for failed method. Again we can include datatypes, examples, in the body for these status codes.**

**Like wise we created other resources based on the business requirement.**

**Coming to the over all structure of RAML, we contain Title, Declaration Types, like variables, then resources and their corresponding responses.**

**In general, for fetching elements or records we use GET operation, we can add queryParameters to identify the record, using their attributes which we can declare in the GET method,**

**For example in our project in the RAML, when assigned GET method to the flights resources,**

**We used displayName attributes to get the information about that method,**

**Assigned queryParameters for date, declared the parameter type, and specifying whether that parameter is required or not by using required attribute which takes in Boolean value, by default it is false.**

**For this queryParameter, we assigned added a response by the same method.**

**Likewise we can create as many resouces as business is required. This is the Skeleton of the Mule Flow and describes the way to implement and easily understand the requirement.**

**3. How many APIs are running in production that you have developed.**

**4. Explain AirLine Management Mule implementation**

**I worked on a Project called Airline management, this project includes Flight Master Data Sync-Up and Daily Flight Schedule Updates. Each module we are maintaining process and system layer as part of API-Led connectivity, however experience layer is not in part of the first version.**

**As part of this module, we will be receiving AWS SQS message for the S3 object placed, as part of business implementation, we are transforming and grouping the planned flight level information in the process layer. Secondly in the system layer we are storing the data in various DB by using store procedures. Additionally we are creating Success/Failed record files in local and uploading as S3 object in AWS. This module is completely developed in Mule.**

**We are reusing the same concepts in this module however we are adding another feature in the process layer that is publishing flight information to Confluent KAFKA Topics per airport and connecting Kafka with React for UI real time streaming about the flight live status. To put it in short, here in this module, we were integrating different applications that is MuleSoft, Kafka, React.**

**6. What new thing I did in AirLine Management.**

**The new things I have learnt and implemented in Airline management are:**

**Amazon web services like S3, SQS, DynamoDB, KAFKA. Most of all Websockets**

**7. What thing I reused in projects**

**In the Airline management project, between the two Master data syncUp and Daily flight status modules, we have utilized DWL functions which mainly implemented the data orchestration appropriate to the business model. Apart from that, in the flow level, we have utilized exception handling mechanisms.**

**8. How did you manage your RAML dataTypes**

**RAML datatypes can be used either isolating from the mail RAML or implementing in the same RAML. However, it is better to use separate RAML because to solve reusability and cleaner code. Here we managed RAML datatypes by creating separate RAML files and invoking them using include functionality.**

**Here this type of implementation is called ChildRAML specification where we are calling RAML away from main flow wherever and whenever that type is required.**

**9. Development Best practices that you followed.**

**The best practices are the way the business model works as required. Here, in my project starting from the very skeleton of the application that is from RAML, we have implemented child ramls along with RAML validations for cleaner code and sustainable code. Later in application perspective, we have used global properties for configuring the values of connectors.**

**10. How many connectors that you used.**

**In the project, we have used around connectors such as JDBC connectors, S3 connectors, SQS connectors and HTTP connectors. Explain them in detail.**

**12. How to use the VM connector**

**VM connector manages intra-app and inter-app communication through either transient or persistent asynchronous queues**

**o   Transient queues are faster than persistent queues, but they are not reliable if the system crashes.**

**o   Persistent queues are slower but reliable.**

**How do we do it?**

**1. How many APIs you have created in overall Mule Experience. List down all the APIS. -- they wanted to know the count.**

**In total apart from the API’s created in my MuleSoft training, I have created over all 2 API’s which are running and used. These are from my Airline Management project. One is Master Data SynUp API and the other is daily flight status API.**

**In the master data SyncUp, we have used API-LED connectivity and following that approach we segregated the whole flow into system layer and process layer. So the same with the Daily status. Here we have not utilized Experience layer because according to our requirement I have been given the task for System and process layer. Would like to elaborate on each of the API’s functionalities ?**

**If yes, Explain the overall flow => Answer from Q**

**If No, Continue with next question.**

**2. Can you explain some RAML that you have developed -- they wanted actual business cases RAML.**

**Yes, RAML is the skeleton or the primary thing which we develop for any Mule Application. In any RAML, on an overview, it contains Body, Responses and examples.**

**It is used for basic understanding of business model and requirements.**

**Let me explain what I have implemented in my Airline management project.**

**Once we go to design centre and click on creating a new RAML specimen**

**We will by default have some RAML, baseUri, title**

**In this we declared types such as flights, segments and whatall the requirement was**

**We declared resources like flights.**

**In this flight resources, we declared methods like GET POST and PATCH**

**Let me elucidate on one Method which is POST:**

**Every method contains body, response.**

**In the body we declare the datatype we use either it be application/json or any.**

**In the response also we declare the format of the response we are expecting.**

**In general we use 200 Status code for good and 400 for bad.**

**Apart from these basic things, we can have our RAML validation, we can have basic authentications, Oauth providers, defining traits, securityschemas, DataTypes, examples etc;**

**We can not only manually write them down, but we can even create child RAMLS and include them in our main RAML.**

**3. How many APIs are running in production that you have developed.**

**API’s which are running in production are both Airline management API’s. Maintainance is taken care by another team, we initially deployed it On-Premise.**

**4. Explain Airline Management Mule implementation**

**Our Airline management project was to facilitate the Airport authority to display the arriving scheduled and delayed flight status. We made an application where for the backend API design we used MuleSoft and for messaging system we used Kafka and for displaying front end we used REACT.**

**Coming specific to Mule implementation of the project.**

**As part of this module, we will be receiving AWS SQS message for the S3 object placed, as part of business implementation, we are transforming and grouping the planned flight level information in the process layer. Secondly in the system layer we are storing the data in various DB by using store procedures. Additionally we are creating Success/Failed record files in local and uploading as S3 object in AWS. This module is completely developed in Mule.**

**We are reusing the same concepts in this module however we are adding another feature in the process layer that is publishing flight information to Confluent KAFKA Topics per airport and connecting Kafka with React for UI real time streaming about the flight live status. To put it in short, here in this module, we were integrating different applications that is MuleSoft, Kafka, React.**

**6. What new thing I did in Airline Management**

**Every project teaches a new thing and so as this project. Here the thing I learnt and implemented was establishing connection and using AWS as a service for our mule implementation. Other than in perspective of mule, we learnt new things and had a very innovative way to club three different technologies with websockets. While working with websockets, I went through R&D on spring and we implemented as realtime connection between Kafka and React. I learnt and implemented more in AWS, like connecting to S3, creating an event in S3 and notifying that event using SQS service.**

**7. What thing I reused from AIM.**

**The reused elements are Error handling, global configurations and few YAML and property files. In the Airline management project, between the two Master data syncUp and Daily flight status modules, we have utilized DWL functions which mainly implemented the data orchestration appropriate to the business model. Apart from that, in the flow level, we have utilized exception handling mechanisms.**

**8. How did you manage your RAML dataTypes**

**RAML datatypes can be used either isolating from the mail RAML or implementing in the same RAML. However, it is better to use separate RAML because to solve reusability and cleaner code. Here we managed RAML datatypes by creating separate RAML files and invoking them using include functionality.**

**Here this type of implementation is called ChildRAML specification where we are calling RAML away from main flow wherever and whenever that type is required.**

**9. Development Best practices that you followed.**

**The best practices I have followed was to include all the possible error handling configurations so that whenever there are any error relating issue we can easily find out what went wrong.**

**Later I maintained cleaner code by dividing the task or flow into chunks and maintaining their level of integrity.**

**Using global properties for configuring values and RAML descriptions for better understanding for new comers or while explaining to business users**

**10. How many connectors that you used.**

**Apart from the connectors which I have used in pratise, In the project I used most of the cnnectors like DB connectors, http connectors, VM connectors and JMS connectors. Apart from these connectors while working with AWS and KAFKA I used other connectors like SQS, S3, and publish KAFKA connector.**

**Go on explaining the connectors if he doesn’t say to stop**

**11. Have you used salesforce connectors**

**Cloud hub**

**Enivronment**

**Roles**

**We can add our own custom roles**

**we can add users from other organizations also**

**we can send the invitation and we can assign the role**

**external identity**

**Deploying in cloudHub**

**When we are configuring the http, we should never hardcode, we need to**

**configure the yaml files.**

**externalize all the things**

**1.IF we want to deploy directly from anypoint studio we can directly do it**

**once provided the credentials we will get into runtime manager**

**runtime version = 4.2.0**

**worker size = trial we habe only 1**

**configure the unique name and deploy**

**go to manage and click on the end point we can visualize our API in action**

**additionally we can see the logs in the browser itself.**

**CI/CD pipelining**

**bitbucket**

**whenever there is a change in code we wanted JENKINS to do code checkout**

**in the local maven deploy should happen. This deployment should happen either in CloudHub or on premise through Runtime manage**

**Mule maven plugin - configure**

**How do we do this configure**

**In the POM.xml inside the plugins tag, we have to add it.**

**By default it always contains**

**maven clean package deploy -DmuleDeploy**

**-------------------------------------------------------------------------**

**Deploying with JENKINS**

**we need to have bit-bucket or Github client and Jenkins installed**

**In the JENKINS -> create new project -> freestyle project -> specify the folder workspace you want your build -> specify the URL or Git or BitBucket source code management -> POLL SCM -> \* \* \* \* which means poll every minute -> Add build step -> maven version -> deploy -DmuleDeploy -> Build**

**API-Led connectivity**

**The main use of API-led connectivity is reusability. Rather than making a monolithic large flow or service we are trying to separate the entire service into micro services.**

**In Api-Led connectivity, we are segregating the flow into system layer, process layer and experience layer.**

**Experience layer - exposes our API’s to the end user , in general frontend**

**Process layer - exposes the System Api’s and does the transformation or related process required.**

**System layer -  it is the low level API which geneally interacts with all the core systems and exposes to process layer.**

**What is API management policy?**

**There are few API management policies which we can configure or apply either in runtime or in develpment.**

**Basic Authenticatinon : SIMPLE**

**requires username and password**

**It protects the APIS from access by forcing applications to specify username and password**

**Basic Authentication : LDAP**

**The policy then extracts the username and password encoded in Base64 and then requests the configured LDAP instance to determine if the user credentials are correct in the provided LDAP context.**

**Client ID enforcement policy**

**Allows access to authorized client applications**

**CORS:**

**enables access to resources residing in external domains**

**There is no return error code or status for this**

**IP Blacklist**

**Blocks a single IP address or a range of IP addresses from accessing an API endpoint**

**OAuth Access token enforcement:**

**allows access to an API for valid OAuth tokens**