1. What are Web Services?

It's a method of communication between two devices over network, it's accessible via HTTP.

Two machines which are working in different languages eg one is Java another is .net can talk with the help of Web Service using XML/JSON. (it's Language independent way of communication)

1. SOAP web services.
2. RESTful web services.

Any web service has server-client relationship. Any web service can have multiple clients. Eg: a travel portal like Make My Trip is selling tickets of an airliner, So in this, this Portal is client and the Airline is the server where the service is running.

2. What is REST?

REST stands for Representational State Transfer or we say it's RESTful web service. REST is a client-server architecture, in this each unique URL is representation of some resource.

REST API uses HTTP methods.

For example

To create a resource on the server, use POST.

To retrieve a resource, use GET.

To change the state of a resource or to update it, use PUT.

To remove or delete a resource, use DELETE.

Example: If we want to create a REST service that fetches the record of a customer then our URI will be:

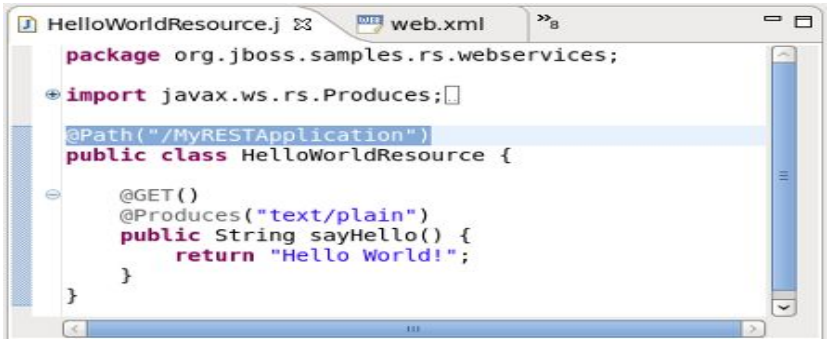
– GET http://tutorialsAtoZ.com/customer/{customerID}

where HTTP method is GET,

Resource is customer

And URI parameter is customerID which depicts

for which resource we want to fetch the records.



3. Which HTTP method is safest?

Ans Safe methods are methods that can be cached, prefetched without any impact on the resource.

An idempotent HTTP method is a HTTP method that can be called many times without different outcomes

HTTP	Idempotent	Safe
OPTIONS	yes	yes
GET	yes	yes
HEAD	yes	yes
PUT	yes	no
POST	no	no
DELETE	yes	no
PATCH	no	no

4. What is a web service API?

An API (Application Programming Interface) is the means by which third parties can write code that interfaces with other code. A Web Service is a type of API, one that almost always operates over HTTP (**though some, like SOAP, can use alternate transports, like SMTP**)

a Web service API almost always uses HTTP (hence the Web part), and definitely involves communication over a network.

5. What's the difference between SOAP and REST?

SOAP is a protocol	REST is architectural style
SOAP means Single Object Access Protocol	REST means Representational state transfer
SOAP cannot use REST	REST can use SOAP as a protocol
P uses service Interfaces to expose the business logic	REST uses the URI to expose the business logic
JAX-WS is java api for soap	JAX-RS is java api for REST
It permits XML only	It permits XML,JSON,HTML and others
Consumes more bandwidth	Consumes less bandwidth

6. What is RAML and why we use it?

RAML – RESTful API Modeling Language
it's similar to WSDL(Web Services Description Language), it contains endpoint URL, request/response schema, HTTP methods and query and URI parameter.
it helps client (I mean a consumer of the service) in knowing, what the service is and what are all operations and how can be they be invoked.
it helps the us in creating the initial structure of this API.
And can also be used for documentation purpose.

7. What Is Mule ?

Mule ESB is a Java-based enterprise service bus (ESB) and integration platform, developer can connect their application with ESB. Mule use service oriented architecture. Apart from of the different technologies the applications use, including JMS, Web Services, SMTP, HTTP. The advantage of ESB, it allows different applications to communicate with each other. Messages can be of any format XML to JSON. Mule ESB Development provide messaging framework that enable exchange of data among application.

8. Why we use ESB? What is the basic responsibility of ESB?

Normally when we connect two application for ex using webservices it would be
Point-to-Point Integration Challenges

- Applications connect to other applications in a point to point manner
- Approach works for small scale integration projects
- Complexity increases as
  - integration points grow
    - Data received over multiple protocols
    - Need of such data format which could be understood by all
  - applications become mission critical
    - Centralization of concerns like security, auditing, routing
    - Appropriate error handling
      An ESB should provide means to handle all these challenges properly.
- Mule provides solutions for the following technical integration problems
  - Connectors for integrating with different systems
  - Different protocols for input – REST, File etc
  - Different data Format for exchange – JSON, XML, Java Object
  - Support for multiple Invocation Styles – Synchronous, Asynchronous, RPC, Messaging,

Batch call
- Life Cycle and Management of mule elements
- Error Handling

9. Why Mule Esb?

Mule ESB is lightweight but highly scalable, allowing us to start small and connect more applications over time. Mule manages all the interactions between applications and components transparently, regardless of whether they exist in the same virtual machine or over the Internet, and regardless of the what is transport protocol used.

There are currently several commercial ESB implementations on the market. However, many of these provide limited functionality or are built on top of an existing application server or messaging server, locking you into that specific vendor. Mule is vendor-neutral, so different vendor implementations can plug in to it. You are never locked in to a specific vendor when you use Mule.

10. What Difficulties Mule Does Encompass?

Transport: applications can accept input from a variety of means, from the file system to the network.

Data format: speaking the right protocol is only part of the solution, as applications can use almost any form of representation for the data they exchange.

Invocation styles: synchronous, asynchronous, or batch call semantics entail very different integration strategies.

Lifecycles: applications of different origins that serve varied purposes tend to have disparate development, maintenance, and operational lifecycles.

11. Why The Name Mule ?

There is a lot of infrastructure work to be done before we can really start thinking about implementing any logic. So this infrastructure work is regarded as "donkey work" as it needs doing for every project. A Mule is also commonly referred to as a carrier of load, moving it from one place to another. The load it specializes in moving is our enterprise information.

12. What Are Available ESbs Apart From Mule ?

All major JEE vendors (BEA, IBM, Oracle, Sun) have an ESB in their catalog. It is unremarkably based on their middleware technologies and is usually at the core of a much broader SOA product suite. There are also some commercial ESBs that have been built by vendors not in the field of JEE application servers, like the ones from Progress Software, IONA Technologies, and Software AG.

13. What Are Differences Between Mule And Other Commercial Esbs ?

Prescriptive deployment model, whereas Mule supports a wide variety of deployment strategies. Prescriptive SOA methodology, whereas Mule can embrace the architectural style and SOA practices in place where it is deployed. Mainly focused on higher-level concerns, whereas Mule deals extensively with all the details of integration. Strict full-stack web service orientation, whereas Mule's capacities as an integration framework open it to all sorts of other protocols. Comprehensive documentation, a subject on which MuleSource has made huge progress recently.

14. What Is Model Layer In Mule ?

The first logical layer is the model layer. A Mule model represents the runtime environment that hosts services. It defines the behavior of Mule when processing requests handled by services. The model provides services with supporting features, such as exception strategies. It also provides services with default values that simplify their configuration.

15. What Is Service Layer In Mule ?

A Mule service is composed of all the Mule entities involved in processing particular requests in predefined manners.A service is defined by a specific configuration. This configuration determines the different elements, from the different layers of responsibility, that will be mobilized to process the requests that it will be open to receive. Depending on the type of input channel it uses, a service may or may not be publicly accessible outside of the ESB.

16. What Is Transport Layer In Mule ?

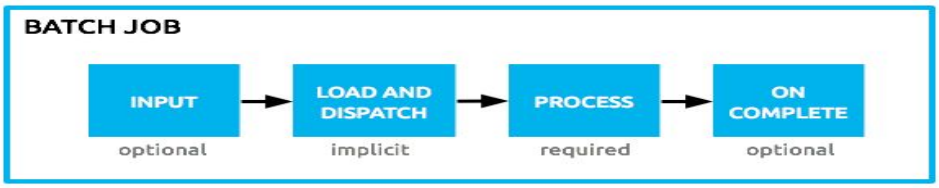
The transport layer is in charge of receiving or sending messages. This is why it is involved with both inbound and outbound communications. A transport manifests itself in the configuration in connectors, endpoints and transformers.

A transport also defines one message adapter. A message adapter is responsible for extracting all the information available in a particular request (data, meta information, attachments, and so on) and storing them in transport-agnostic fashion in a Mule message.

17. Batch processing

Mule possesses the ability to process messages in batches. A batch job is a block of code that splits messages into individual records and perform operations on each record.

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.



Batch Processing in Mule has four phases.

➢ **Input Phase**

• Input Phase is an optional part in a batch job. You can trigger a batch job, by placing an inbound connector (a message source) in the input Phase of batch job.

• If using batch executor (in Mule flow) to trigger the batch job, then we can accommodate any transformations or adjustments to a message payload before Mule begins processing it as a batch, as batch Step accepts only payloads of type interface java.util.Collection, java.lang.Iterable..

➢ **Load and Dispatch**

• This is implicit and is running behind the scenes. During this phase, Mule creates a batch- job-instance for a message received.

• Mule creates a persistent queue for a batch-job-instance and stores all the records of the payload. Then each record is sent to the first BatchStep for processing.

➢ **Process Phase**

• We can use multiple batch steps here to process the records.

• Each batch Step has the following configuration

– Accept Expression: BatchStep processes a particular record only when this expression is evaluated to true (corresponding to that record). If false the corresponding record will be skipped, and batchStep tries to execute next record in the queue.

– AcceptPolicy: There are three categories under this. They are whether to accept all the records (ALL), Only FailedRecords (ONLY_FAILURES), Only SuccessfulRecords (NO_FAILURES). The default Accept policy is NO_FAILURES.

➢ **OnComplete Phase**

• This is the fourth phase in the batch job. After mule has executed entire batch, the output is an object BatchJobResult.

• This can be used to create report of the no. of Records processed, Count of successful and failed records. So, that we can take appropriate action on what type of input data records are getting failed.

Batch Commit

• We can use a batch Commit in a batch Step to accumulate a subset of records within a batch for bulk upsert to an external source or service.

• We can set the chunk size or the number of records to be taken as subset using Commitsize attribute.

• However, if we want all the records to be written in one shot, we can use the Streaming attribute of BatchCommit. This will wait until all the records reach batchCommit and write all records in one go with streaming enabled.

• BatchCommit will be helpful particularly when writing to Database, SalesForce connectors, as records number is high we cannot write each record everytime to them.

18. What are different elements of a Mule Application

Elements of a Mule Application

a. Message Sources
b. Messages
c. Flows, Sub-flows and Batch Processing
d. Message Processors
e. Exception Strategies

19. Diff between async scope and asynchronous flow

Async Scope: An **async scope** is a branch processing block that executes simultaneously with the parent message flow, Async scope invokes the contained message processors in a separate thread

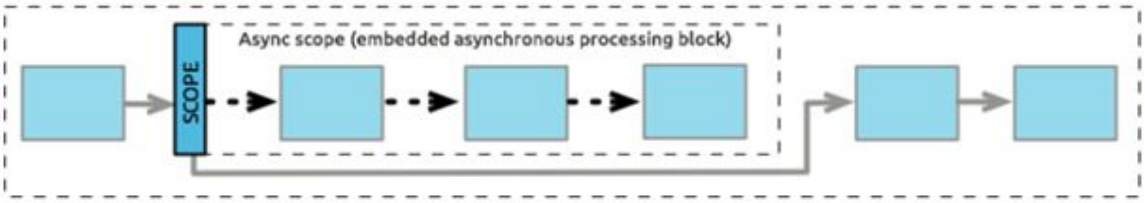
- Cannot be reused
- Source cannot be added
- Cannot have its own exception strategy

Asynchronous Flow: contents of the flow will be processed asynchronously with the triggering flow, it can perform **time-consuming tasks**, without stalling the flow that triggered its execution. In clusters of Mule servers, messages can migrate between nodes when sent to an asynchronous flow. This allows for load balancing between nodes and higher performance of application.This type of flow does not inherit processing or exception strategies from the triggering flow instead they have their own.

20. What is async scope

An **async scope** is a branch processing block that executes simultaneously with the parent message flow. This type of processing block can prove useful for executing time-consuming operations (such as printing a file or connecting to a mail server) — as long as those operations do not require sending a response back to the initiating flow. In other words, the main flow can continue execution while it initiates and processes the asynchronous scope; it does not have to pause until the last message processor embedded in the asynchronous flow has completed its task.

To facilitate this simultaneous branch processing, the async scope sends one copy of the message it has received to the first embedded message processor in its own processing block; at the same time it sends another copy of the message to the next message processor in the main flow (see below).



21. What happens if async scope is inside until successful

The flow will stuck.

As async scope will not return anything and Until successful will expect some response to fulfill until-successful condition.

22. What are the different ways to connect a database with Mule DB connector?

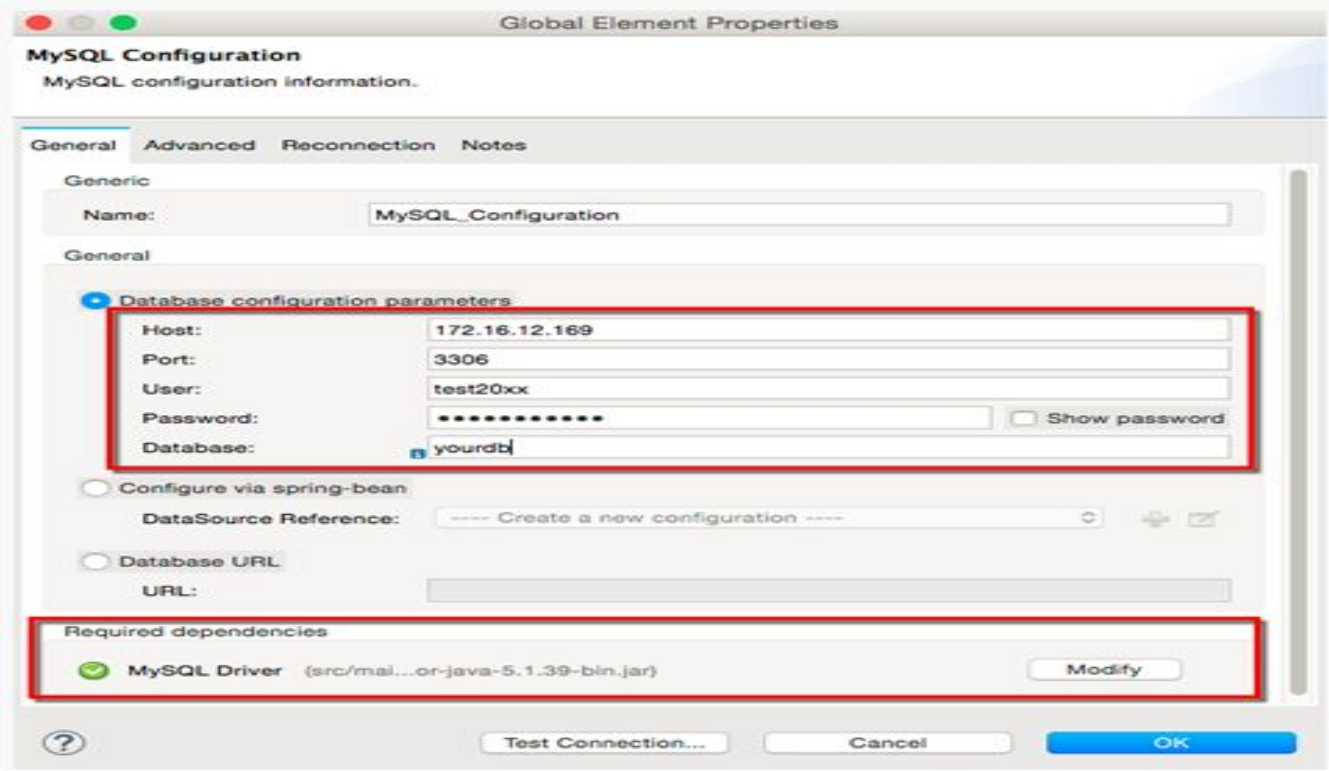
The Database connector allows you to connect with database with almost any Java Database Connectivity (JDBC) relational database.

Database connector and its required fields

Required fields:

- Host
- Port
- Username
- Password
- Database

Adding jar is very important. (In required dependencies)



23. What are Shared Resources in Mule and how are they been used?

We can make connectors as an reusable component by defining them as common resources and expose them to all applications deployed under a same domain, these resources are known as shared resources. These shared resource needs to be defined inside Mule Domain Project and then referred to each of the projects that are meant to use the elements in it.

Mule project is normal project, Normal project will have the all dependencies with in the project scope it self, if you Created Mule Domain Project where we can keep all the common fuctionality,so that we can access from the multiple projects.

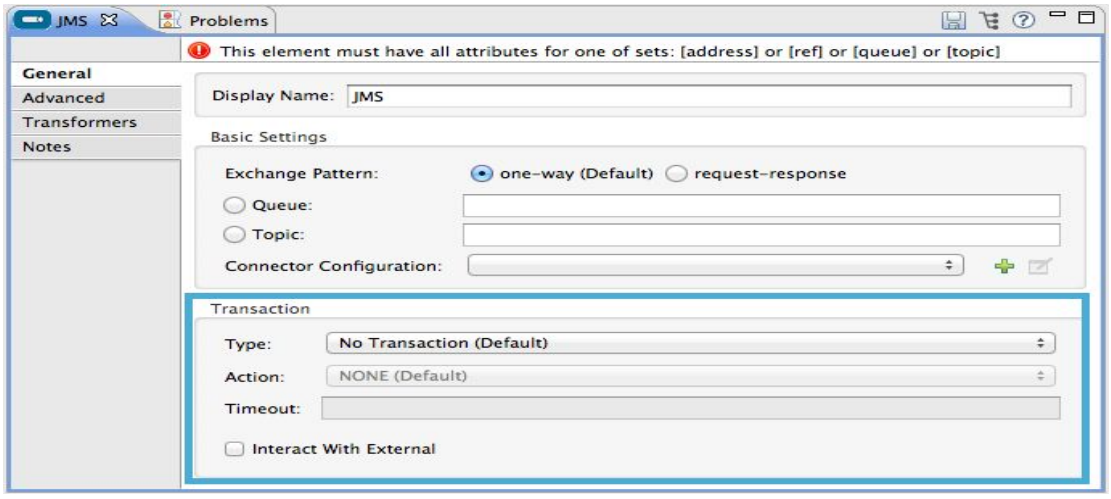
to get this, we need to create domain project, to access in Mule Project we need to add the the domain project name in the ***mule-deploy.properties***

24. What and how to use transaction management in Mule?

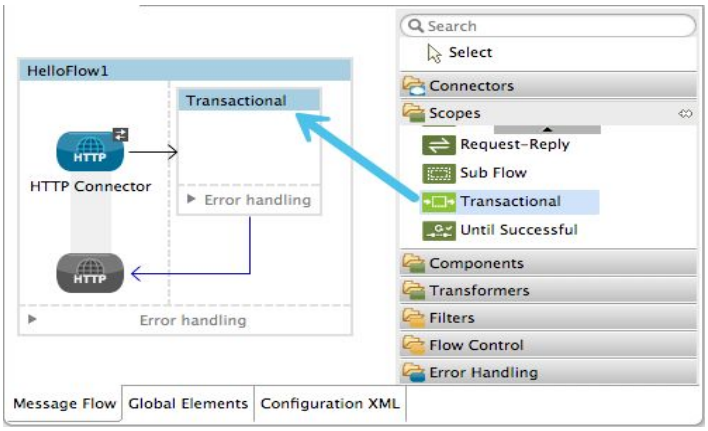
Transactions in mule

where a series of steps in a flow must succeed or fail as one unit, Mule uses a transaction to denote such a unit. For example, you might use a transaction to encapsulate several steps in a flow for which the end result involves committing information to a database. In this type of scenario, the commit is either entirely complete and succeeds, or is incomplete and it fails. Even if partially complete, the commit – or transaction – fails. Where a transaction fails, Mule rolls back the operations within the transaction so that no one part results in partial completion.

Transaction to a Connector

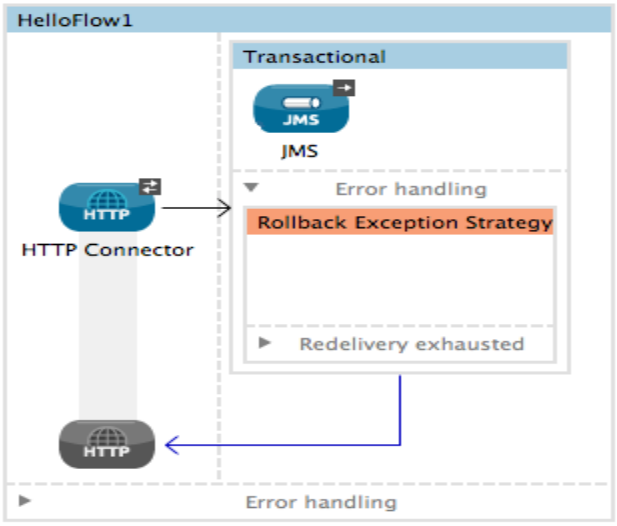


Transaction as a Scope



Exception strategy in Mule

Add a **Transactional** scope to your flow , then add building blocks within the scope to build a transaction.
Search for an exception strategy and drag and drop it into the exception strategy section at the bottom of the scope.



Configure the exception strategy as needed, keeping in mind Mule uses this exception strategy to handle any exceptions thrown while processing the transaction.

25. what is Mule Flow?

Mule applications are built around one or more flows, flow of message from message source to one or more message processors. Typically, a Mule application begins processing a message it receives at an inbound endpoint in a flow.

This flow can then either:

a. implement all processing stages, or
b. route the message to other flows or subflows to perform specific tasks.

26. What are the different types of flow

Subflow: A subflow processes messages synchronously (relative to the flow that triggered its execution) and always inherits both the processing strategy and exception strategy employed by the triggering flow. While a subflow is running, processing on the triggering flow pauses, then resumes only after the subflow completes its processing and hands the message back to the triggering flow.

Synchronous Flow:A synchronous flow, like a subflow, processes messages synchronously (relative to the flow that triggered its execution). While a synchronous flow is running, processing on the triggering flow pauses, then resumes only after the synchronous flow completes its processing and hands the message back to the triggering flow. However, unlike a subflow, this type of flow does not inherit processing or exception strategies from the triggering flow.
It's like single thread processing. Can be used in transactional processing

Asynchronous Flow: An asynchronous flow simultaneously and asynchronously processes messages in parallel to the flow that triggered its execution. When a flow passes a message to an asynchronous flow, thus triggering its execution, it simultaneously passes a copy of the message to the next message processor in its own flow. Thus, the two flows – triggering and triggered – execute simultaneously and independently, each finishing on its own. This type of flow does not inherit processing or exception strategies from the triggering flow instead they have their own.
This type of flow processes messages along multiple threads.

Advantages of Using Multiple Flows in an Application

- Asynchronous Flow B can perform time-consuming tasks, such as writing data to an external database or emailing a message, without stalling Flow A, the flow that triggered its execution.

- Flow A and Flow B can respond differently to errors.
- Breaking up complex operations into a series of smaller flows or subflows makes applications – whether in a GUI or in XML code – easier to read.
- The processing actions in a flows or subflows can be called and used by multiple flows in an application.

27. Difference between flow and sub-flow.

Flow is a message processing block that has its own processing strategy and exception handling strategy. Used in integration tasks, data processing, connecting applications, event processing, etc.

Subflow always processes messages synchronously but inherits processing strategy and exception handling strategy from the calling flow. It can be used to split common logic and be reused by other flows.

28. Difference between private flow and sub-flow.

Private flow has Processor and own Exception Strategy. **Private flow** does not have source define. It can be synchronous or asynchronous based on the processing strategy selected.

Sub-flow consists only of Processor. **Subflow** always processes messages synchronously but inherits processing strategy and exception handling strategy from the calling flow. It can be used to split common logic and be reused by other flows.

29. What are Flow Processing Strategies?

Flow processing strategies A flow processing strategy determines how Mule implements message processing for a given flow . Mule automatically determines best processing strategy Implicitly, set to either synchronous or queued-asynchronous .

A flow is synchronous when – (single thread)

1. The flow's exchange pattern is request-response –
2. The flow partakes in a transaction –
3. The flow is really a sub-flow

A flow is asynchronous in all other cases (multi thread) : flows with one-way exchange patterns and which are not transactional

(Unless overridden by manually setting a processing strategy)

Synchronous vs asynchronous flow difference

Synchronous – (You cannot force a flow with a request-response exchange pattern and/or transactionality to be asynchronous)

1. Same thread is used across the flow
2. If outbound endpoint is one way (JMS), then the thread will be used to continue the message processing
3. If outbound endpoint is request-response, then the thread will wait for the response from the endpoint and continue the message processing after receiving the response

Asynchronous-(You can override an implicitly selected queued-asynchronous processing strategy by explicitly declaring a synchronous processing strategy (or, in rare use cases, a different kind of asynchronous processing strategy) instead.)

1. Receiver thread receives the message and places it in Staged Event-Driven Architecture (SEDA) Queue
2. Receiver thread will be released for receiving message after placing in queue
3. Message will be placed in a dispatcher queue for sending out

In Async Mule uses a queue to decouple the receiver thread from the rest of the flow.

30. Default processing strategy of flow

Exchange Pattern	Transactional?	Flow Processing Strategy
Request-Response	Yes	Synchronous
Request-Response	Yes	Synchronous

One-way	Yes	Synchronous
One-way	No	Queued-Asynchronous

31. What is a Mule Message Processor?

Message Processors operate on the messages in a flow.

- **Connectors**
- **Transformers**
- **Routers**
- **Filters**
- **Component*s***
- **Scopes**
- **Exception Strategies**

It is possible to create custom message processors by implementing the org.mule.api.processor.MessageProcessor in a **custom** class

32. What Is Connector In Mule ?

Connectors are message processor in Mule which are used for:

Sending Data to a flow

Receiving Data from a flow

Interacting with APIs

Types of Connectors

Endpoint based Connectors

Used for sending and receiving data over a protocol like Http, JMS

Follow either a one-way or request-response exchange pattern

Operation Based Connectors

Used for interacting with APIs like Twitter, Salesforce, MongoDB

Operation-based connectors follow an information exchange pattern based on the operation selected

EndPoint Based connectors

- Endpoint based connectors are configured as either inbound or outbound endpoints in a flow.
- Inbound endpoints serve as a message source for a flow.
- Outbound endpoints can occur mid-flow or at the end of flows, and send information to external systems.
  Eg File,Http,JMS,Email,FTP,JDBC,VM

Operation Based connector

- **Operation-based connectors** define a specific operation for the connector to perform.

Global Connector Configuration

- Connectors can be configured as a global element rather than at the level of the message processor within the flow.
- The global connector can be referenced by connectors of the same type in an application.
- For operation-based connectors, the global connector configuration is mandatory.
- For endpoint-based connectors global connector configuration it is optional.
- Eg:

<connectorName>:config for operation-based connectors

<connectorName>:connector for endpoint-based connectors.

33. What is the difference between Endpoints and connectors?

Endpoints are a type of connector. Connectors in Mule are either endpoint-based or operation-based. Endpoint-based connectors follow either a one-way or request-response exchange pattern and are often (but not always) named and based around a standard data communication protocol, such as FTP, JMS, and SMTP. Operation-based connectors follow an information exchange pattern based on the operation that you select and are often (but not always) named and based around one or more specific third-party APIs.

34. Which connectors you have used.

HTTP, HTTPS, FTP, SFTP, Database, File…. Etc.

35. What are required fields of connectors you have used

Remember the fields of connectors which u have used. For example: What if file Age and polling frequency in FILE connector.

36. When u use FTP and SFTP connector

FTP: The FTP Connector implements a file transport channel so that your Mule application can exchange files with an external FTP server. You can configure FTP as an inbound endpoint (which receives files) or outbound endpoint (which writes files to the FTP server).

SFTP: The SFTP Connector implements a secure file transport channel so that your Mule application can exchange files with external resources. SFTP uses the SSH security protocol to transfer messages. You can implement the SFTP endpoint as an inbound endpoint with a one-way exchange pattern, or as an outbound endpoint configured for either a one-way or request-response exchange pattern

37. Diff between HTTP and HTTPS

HTTPS is secure HTTP, HTTP with SSL(secure socket layer)
It uses .jks file for handshake, which is present in src/main/resource folder.
https://docs.mulesoft.com/mule-user-guide/v/3.8/tls-configuration

TLS (Transport layer security its a successor of SSL) configuration Element is used to provide security with the help of trust store and keystore

```
<tls:context name="customContext">
    <tls:trust-store path="trustStore" password="mulepassword"/>
    <tls:key-store path="clientKeystore" keyPassword="mulepassword"
password="mulepassword"/>
 </tls:context>
```

The tls:context element defines a configuration for TLS, which can be used from both the client and server sides. It can be referenced by other configuration objects of other modules (or defined as a nested element of one of them).

Inside it, you can include two nested elements: key-store and trust-store. You don't need to include both, but at least one of the two must be present:

- From the server side: the trust store contains certificates of the trusted clients, the key store contains the private and public key of the server.
- From the client side: the trust store contains certificates of the trusted servers, the key store contains the private and public key of the client.

Adding a trust store or a key store to a TLS configuration implicitly implements the corresponding kind of authentication. Adding both a keystore and a trust store to one same config (as in the code example above) implicitly implements **two way TLS authentication**, also known as **mutual authentication**.

38. How to build a Custom connector

https://docs.mulesoft.com/anypoint-connector-devkit/v/3.8/
Annotations:
@Connector: For java class, for giving name to connector
@processor: java methods
@Configurable: For data members

```
@Connector(name="connpom", friendlyName="Connpom")
	public class ConnpomConnector {
		@Config
		ConnectorConfig config;
			* @param friend Name to be used to generate a greeting message.
		 * @return A greeting message
			@Processor
		public String greet(String friend) {
				return config.getGreeting() + " " + friend + ". " + config.getReply();
		}
		public ConnectorConfig getConfig() {
			return config;}
		 public void setConfig(ConnectorConfig config) {
			this.config = config;}}
ConnectorConfig Class:
@Configuration(friendlyName = "Configuration")
	public class ConnectorConfig {
			@Configurable
		@Default("Hello")
		private String greeting;
			@Configurable
		@Default("How are you?")
		private String reply;
		  public void setGreeting(String greeting) {
			this.greeting = greeting;
		}
		   public String getGreeting() {
			return this.greeting;
		}
		 public void setReply(String reply) {
			this.reply = reply;
		}
		public String getReply() {
			return this.reply;
		}
	}
```

39. What Is Endpoint In Mule ?

An endpoint represents the specific usage of a protocol, whether it is for listening/polling, reading from, or writing to a particular target destination. Hence it controls what underlying entities will be used with the connector they depend on. The target destination itself is defined as a URI. Depending on the connector, the URI will bear a different meaning; for example, it can represent a URL or a JMS destination.

40. What Is Transformer In Mule ?

A transformer takes care of translating the content of a message from one form to another. It is possible to chain transformers to cumulate their effects. Transformers can kick in at different stages while a message transits through a service.

41. What Is Router In Mule ?

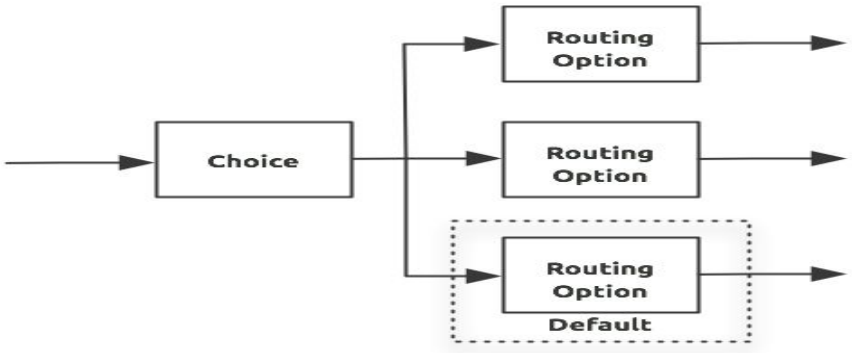
Routers play a crucial role in controlling the trajectory a message will follow when it transits in Mule. Routers are used to route data between applications

Eg are choice router, splitter , scatter gather

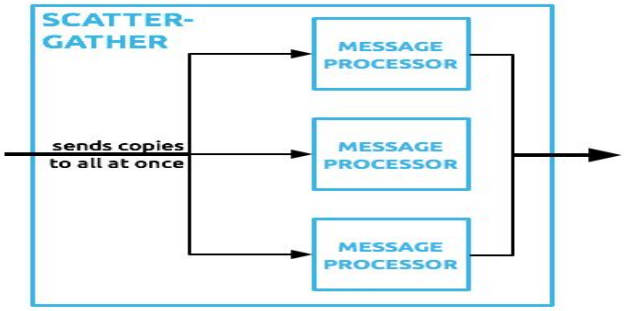
42. Choice router, scatter gather and splitter flow

Choice Router: The choice flow control dynamically routes messages based on message payload or properties. It adds conditional programming to a flow, similar to an if/then/else code block.

A choice flow control uses expressions to evaluate the content of a message, then it routes the message to one of the routing options within its scope (see image below). It directs messages to the first routing option in the scope that matches the routing configurations (evaluates to true). If none of expressions evaluate to true, the choice flow control directs the message to the default (else) route.

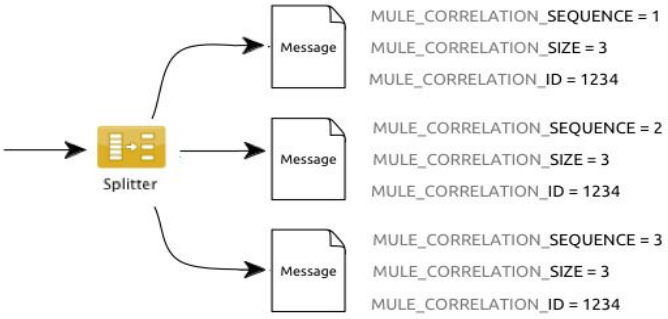


Scatter-Gather: The routing message processor Scatter-Gather sends a request message to multiple targets concurrently. It collects the responses from all routes, and aggregates them into a single message.



Splitter: The Splitter Flow Control splits a message into separate fragments, then sends these fragments one at a time to the next message processor in the flow. Segments are identified based on an expression parameter, usually written in Mule Expression Language (MEL), but other formats can be employed also. You can then use a Collection Aggregator Flow Control to reassemble the parts of the original message. You can also include a **Resequencer** Flow Control to put the parts back into the original sequence in case they are shuffled out of order.

Splitting and aggregating the message is especially useful when you intend to process the split parts in asynchronous flows running on separate servers. Together, the splitter and aggregator flow controls allow you to share the workload among several servers and still be able to reassemble the message after it's processed.



43. Explain Scatter Gather. If 1 of the Scatter Gather o/p will have exception then other will be executed or not? What will be final output?

If there are no failures, Mule aggregates the results from each of the routes into a message collection (MessageCollection class). Failure in one route does not stop the Scatter-Gather from sending messages to its other configured routes, so it is possible that many, or all routes may fail concurrently.

By default, if any route fails, Scatter-Gather performs the following actions:

- sets the exception payload accordingly for each route
- throws a CompositeRoutingException, which maps each exception to its corresponding route using a sequential route ID

Catching the CompositeRoutingException allows you to gather information on all failed routes**.**

44. What Is Filter In Mule ?

Filters are a powerful complement to the routers. Filters provide the brains routers need to make smart decisions about what to do with messages in transit.

- Filters are used to decide on what message continues processing through a flow.
- Filtering can be done based on the message type, payload or headers.
- Payload-type-filter
  - Message passes through this filter only if the payload is of a required type
- Expression Filter
  - Is used to filter based on the result of a MEL expression
- Wildcard Filter
  - Messages that match a shell-like wildcard will be allowed to pass.
  - Is generally used for filtering text messages.
- Regular Expression Filter
  - Filtering is done based on a regular expression.
  - Is very useful for filtering text messages.

45. What Is Component In Mule ?

Components are message processors which execute business logic on messages

, Spring bean, Java bean, or script can be configured as components in a Mule application, Mule includes several scripting components that can be used in flows

eg of components are Groovy script component, Java Component, Web Service Components etc

46. Difference between Java transformer and Java component

If you want to trigger some more complex set of processes coded in Java, or your complex business logic you should better use Java component

On the other hand Java transformer can be used to create a custom transformer that can change or modify the message payload

47. Java component has 2 functions, how to invoke one as per requirement

Two methods:

1. In xml use

<method-entry-point-resolver>
<include-entry-point method = "xyz">
</method-entry-point-resolver>

2. Use invoke component : using invoke component from palette and giving object ref of which we want to invoke method and method name in method in General tab.

48. What is scope?

Mule **scopes** act as wrappers for message processors to make them function as a single unit.

a. Async
b. For each
c. Transactional
d. Until Successful

49. What is For-each scope

The **For Each** scope splits a collection into elements and processes them iteratively through the processors which are embedded in the scope, then returns the original message to the flow.

50. Until successful is used for?

The **until-successful** scope processes messages through its processors until the process succeeds. By default, until-successful processing occurs asynchronously from the main flow. After passing a message into the until-successful scope, the main flow immediately regains control of the thread. but, we can configure until-successful to run synchronously relative to the main flow.

Until-successful repeatedly retries to process a message that is attempting to complete an activity such as:

Dispatching to outbound endpoints, for example, when calling a remote web service that may have availability issues.

51. Various types of Exception Handling?

Types of Errors that can occur in Mule Applications:

- System Exceptions
- Messaging Exceptions.

Exception Handling Strategies

- Default Exception handling
- Catch Exception handling
- Rollback Exception handling
- Reference
- choice

System Exception Strategy

- System Exception Strategy is invoked when an exception is thrown at the system- level (i.e., when no message is involved).
- System exceptions can occur during application start-up or when a connection to an external system like JMS Broker fails
- When a system exception occurs, Mule sends an exception notification to registered listeners, logs the exception. It executes a reconnection strategy if the exception was caused by a connection failure

System Exception Strategies are not configurable in Mule but Reconnection strategy can be configured.

Messaging Exception Strategy

- Messaging Exception occurs when an exception is thrown within a flow whenever a message is involved.
- When a message being processed through a Mule flow throws an exception, normal flow of execution stops. Mule transfers the message to the message processor sequence within the exception strategy.

For the same we have multiple exception strategies

❖ ***Default Exception Strategy***
– Applied by default for all Messaging Exceptions if no other exception strategy matches for that exception
– Causes rollback of any transaction if it exists and then logs the exception
❖ ***Catch Exception Strategy***
– Invokes a customized flow to handle the exception
– Consumes the inbound message and commits the transaction
– Mule's catch exception strategy behavior is similar to a Java catch block, except that you cannot throw a new exception or catch another exception within a catch exception strategy
❖ ***Rollback Exception Strategy***
– Inbound message is not consumed
– Causes rollback of the message. In some cases, where the message supports, causes redelivery of the message.
– If the message exceeds the redelivery attempts (defined by maxRedeliveryAttempts attribute), the flow defined by <on-redelivery-attempts-exceeded> section is invoked, failed transaction is committed and the message is consumed.
❖ ***Reference Exception Strategy***
– Does not itself consume the message or cause rollback or commit
– References a global exception handler defined with a global catch, rollback or choice exception strategy.
❖ ***Choice Exception Strategy***
– Does not itself consume the message or cause rollback or commit
– Routes to an exception strategy from a set of strategies based on an expression defined with the when attribute for each exception strategy.
– If a suitable exception strategy is not found then the default exception strategy is invoked.

Characteristics of Messaging Exception Strategies

- Each flow can contain only one exception strategy.
- Each exception strategy can contain any number of message processors.
- Choice exception strategies can contain one or more catch and/or rollback exception strategies.
- Rollback and catch exception strategies cannot contain other exception strategies.
- Global exception strategies can be created and reused in flows.

52. What is custom exception and global exception?

53. What u should do if you want to refer same exception strategy in all flows present?

Two ways:

1. Make it as global exception strategy and refer it through Reference exception strategy
2. Make it as default exception strategy through global element.

54. How Message In Mule Is Composed and What Is Payload In Mule ?

Answer :

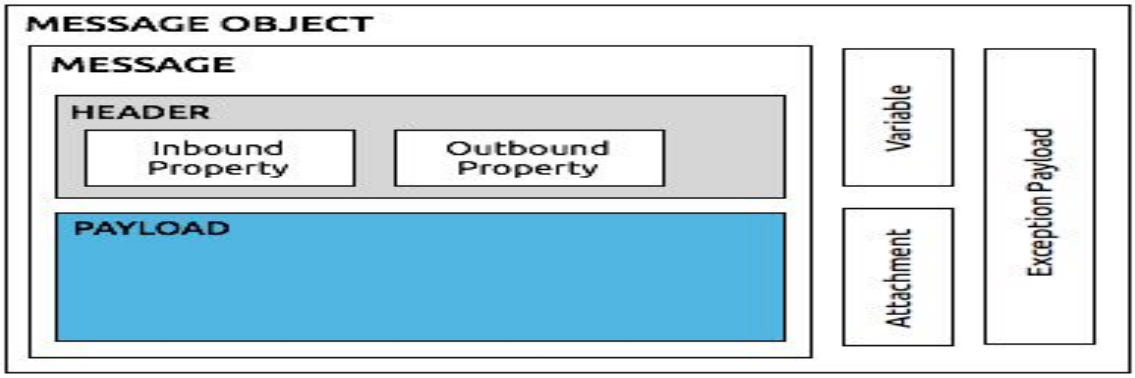
The Mule message is the data that passes through an application via one or more flows. It consists of two main parts, The message header, which contains metadata about the message, contains inbound and outbound properties,Inbound properties are immutable and outbound properties are mutable. The message payload, which contains your business-specific data.

***Inbound properties** are automatically generated by the message source and are immutable*

- *They are immutable and cannot be set or manipulated by the user.*
- *They contain metadata specific to the message source*
- *A message retains its inbound properties only for the duration of the flow*
- *Inbound properties are not carried forward in case a message moves from one flow to another.*

Outbound properties are mutable

- ***Outbound properties** are set automatically by Mule or a user can set them by using one or more transformer elements in the flow.*
- *They contain metadata about the message similar to that of an inbound property*
- *Outbound properties become inbound properties when the message passes from the outbound endpoint of one flow to the inbound endpoint of a different flow through a connector.*
- *Outbound properties remain outbound properties if the message moves to a new flow via a flow-ref rather than a connector.*



The content of a message, also known as payload. It is wrapped in an instance of org.mule.api.MuleMessage, which provides different means of accessing the payload under different forms. A MuleMessage also contains properties, much like the header of a SOAP envelope or the properties of a JMS message, and can also have multiple named attachments.

To set message payload we use **Set Payload message processor** ,can completely replace the content of the Message's payload.

55. What Are Different Type Of Messages In Mule ?

Bridge messages: Pass messages from inbound to outbound routers.
Echo and Log messages: Log messages and move them from inbound to outbound routers.
Build messages: Create messages from fixed or dynamic values

56. What are the different types of variables in mule?

Variables are name value pairs that represent data about a message.

Different kind of variables depending upon their scope
– Flow variables exist only in the flow in which they exist.
– Session variables exist across all flows within the same application.
– Record variables represent records processed as part of a batch.

57. What is message enricher

a message enricher enriches the current payload with some additional message or information and this is done without disturbing the current payload.

Enricher is used if the target system needs more information than the source system can provide. It enriches the Mule message by calling an external system or doing some transformation to the existing payload and saving it into some scope of variable, like session, and the transformation happening in the enricher scope doesn't affect the actual payload.

which method is used for it?

Set-property: Save some information extracted from payload or original payload to some invocation or flow scope variable.

how Message Enricher works.

- Enricher sends a copy of the original message into the processor.
- The original message waits.
- The copy is processed.
- The copy's response is a message.
- Part(s) of the response are added to part(s) of the original message.
- The enriched message moves forward.

58. What Are Configuration Builders In Mule ?

Answer :

Mule uses configuration builders that can translate a human-authored configuration file into the complex graph of objects that constitutes a running node of this ESB. The main builders are of two kinds: a Spring-driven builder, which works with XML files, and a script builder, which can accept scripting language files.

59. Why Spring-driven Configuration Builder Is Important Than Script Builder ?

Answer :

The advantages of Spring-driven configuration builder

It is the most popular — you are more likely to find examples using this syntax.

It is the most user friendly — Spring takes care of wiring together all the moving parts of the ESB, something you must do by hand with a script builder.

It is the most expressive — dedicated XML schemas define the domain-specific language of Mule, allowing you to handle higher-level concepts than the scripting approach does.

60. What Is Bridge Component In Mule ?

Answer :

A bridge component is used to pass the messages from the inbound router to the outbound one. A bridge is a neutral component: it does not perform any action or modify messages that it processes.

61. What Tags Are Used To Configure Spring Elements In Mule ?

Answer :

Tags like <spring:bean/> <spring:entry/> etc. are used to configure spring stuff.

62. Give An Example Of Stdio Connector In Mule ?

Answer :

<stdio:connector name="SystemStreamConnector" promptMessage="Please enter something: "
messageDelayTime="1000"/>

63. Give An Example Of Http Connector In Mule ?

Answer :

<http:connector name="HttpConnector"proxyHostname="proxyHostname"
proxyPort="proxyPort"
proxyUsername="proxyUsername"proxyPassword="proxyPassword"/>

64. When Does Mule Instantiates A Connector ?

If Mule figures out that one of our endpoints needs a particular connector, it will automatically instantiate one for us, using all the default values for its different configuration parameters. This is a perfectly viable approach if we are satisfied with the behavior of the connector when it uses its default configuration. This is often the case for the VM or HTTP transports. Note that Mule will name these default connectors with monikers such as connector.http.0.

65. What Is Transport Service Descriptor In Mule ?

The connector has a technical configuration known as the Transport Service Descriptor (TSD). This hidden configuration is automatically used for each instance of the connector. It defines technical parameters such as what classes to use for the message receivers, requesters, and dispatchers; or the default transformers to use in inbound, outbound, and response routers. Knowing these default values is essential to grasping the behavior of a transport.

66. How Many Endpoints Are There In Mule ?

Answer :

There are two endpoints :

inbound

outbound.

You will use inbound and outbound endpoints to communicate between components and services inside Mule as well as with the outside world.

67. what is Inbound Endpoint in Mule?

Inbound endpoints are message sources which as the name suggests is where messages are created. They can be created based on external events (like an incoming HTTP request or JMS message)

68. What Is An Outbound Endpoint In Mule ?

Outbound endpoints are used to send data. An outbound endpoint is used to do things such as write to file streams, and send email messages.

Outbound endpoints are message processors that send the current message to "destinations" like a JMS queue, an HTTP server, a file, ...

69. What Is Global Endpoint In Mule ?

An endpoint destination that is shared by several routers, it is worth creating a global endpoint. A global endpoint is not typified for inbound or outbound routing, making it usable in many different places in a configuration file. It must be named so it can actually be used in a service, which will reference the global endpoint by its name.

70. Why Does An Endpoint In Mule Offer An Address Attribute ?

This allows us to configure a generic endpoint using the Mule 1.x style of URI-based destination addresses instead of the dedicated attributes of the specific endpoint element.

71. Give An Example Of File Endpoint In Mule ?

<file:endpoint name="tmpPoller" path="/tmp" fileAge="1000"pollingFrequency="2000"/>

72. What Is Streaming Property In File Connector In Mule ?

The value of this streaming property can be either true or false. If it is set to true then we are actually working on stream of file data otherwise we are working with file itself.

73. What Is Pollingfrequency Property In File Connector In Mule ?

When we want file inbound endpoints to poll their source directories for new content. This is accomplished by setting the pollingFrequency to some milliseconds value.

74. What Is Autodelete Property In File Connector In Mule ?

The default value of autoDelete is true. Therefore, a file inbound endpoint will, by default, remove the file from the source directory once it is read by the inbound endpoint. If you do not want to delete file automatically then you can set autoDelete property to false.

75. What Is Fileage Property In File Connector In Mule ?

The fileAge property specifies how long the endpoint should wait before reading the file again. For instance, a fileAge of 60000 indicates Mule should wait a minute before processing the file again.

76. How To Send Only Certain Types Of File From One Directory To Another In Mule ?

Use the below element in file inbound to filter certain types of files.

<file:filename-wildcard-filter pattern="file*.xml"/></file:inbound-endpoint> pattern indicates what pattern of file names should move from one directory to another directory.

77. What Is Multicasting Router In Mule ?

The multicasting router can send messages to multiple endpoints over different transports. The multicasting router allows you to easily move the same messages across these different endpoints.

78. What Is Mule Transformer ?

Mule transformers are used to convert the data received into transformed output

- Payload Type Transformation

  Data type of the message payload is transformed from one binary form to another eg Object to String transformation

- Payload Format Transformation

  Data format of message payload is transformed to another format. JSON TO XML Transformer and XML to JSON transformer

- Properties Transformation

  Properties of the message are modified in this transformation

- Transformers often come in pairs. The transformer that is able to reverse the changes of another one is called a round-trip transformer.

79. What Is Mule Context ?

The Mule context is composed of references to different objects, including security credentials, the session in which the request is processed. All internals of the ESB are accessible through Mule context.

80. What is object-store

An **object store** is a facility for storing objects in or across Mule applications. Mule uses object stores to persist data for eventual retrieval. Internally, Mule uses object stores in various filters, routers, and other message processors that need to store state between messages. In most cases, Mule creates and manages object stores automatically.

81. Why we use VM transport?

You can use the Java Virtual Machine (VM) transport for intra-JVM communication between Mule flows.This transport by default uses in-memory queues but can optionally be configured to use persistent queues(persistent queues are use to help prevent data loss). **Note:** VM file persistency does not work on clusters.

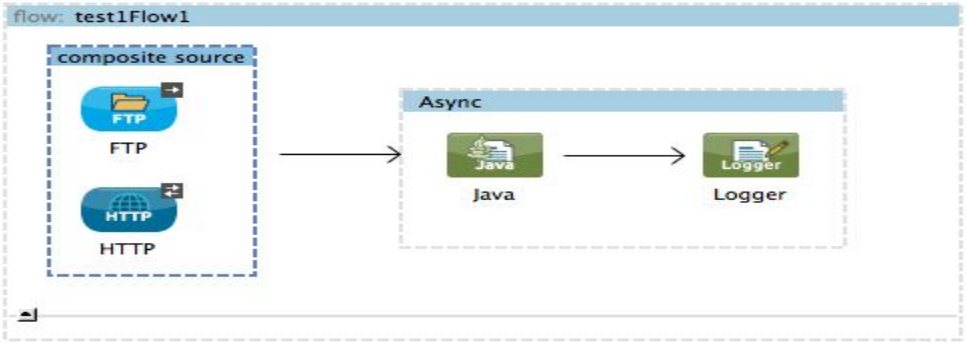
The VM transport has often been used to implement complex integrations made up of multiple applications.

VM is still useful in certain situations. Suppose, for instance, that most of the parts of your solution are local, but some need to be decoupled for testing, or because some need to be made remote.

You can use Flow References to directly reference one flow from another without a transport in the middle.use VM transports if you need redelivery of messages, this won't be possible in private flow and flow ref because for rollback exception strategy even if we define that no internal queue exists.

82. What is composite source

In the case of **Composite**, the embedded building blocks are message sources (i.e. inbound connectors) that listen in parallel on different channels for incoming messages. Whenever any of these receivers accepts a message, the **Composite** scope passes it to the first message processor in the flow, thus triggering that flow.



83. what is Munit

MUnit is a Mule application testing framework that allows us to easily build automated tests for our integrations and APIs. It provides a full suite of integration and unit test capabilities, can be used with Maven (software project management tool)

With MUnit you can: examples

- Create your Mule test by writing Mule code
- Create your Mule test by writing Java code
- Disable flow inbound endpoints
- Disable endpoint connectors

84. Custom Transformers – what class we inherit

The Mule message is the data that passes through an application via one or more flows. It consists of two main parts:

- The message header, which contains metadata about the message contains inbound and outbound properties
  Inbound properties are immutable and outbound properties are mutable
- The message payload, which contains your business-specific data.

AbstractTransformer – override doTransform() method
It is used when only payload needs to be access or used within Transformer

```java
import org.mule.transformer.AbstractTransformer;

public class OrderToHtmlTransformer extends AbstractTransformer
{
    public Object doTransform(Object src, String encoding) throws TransformerException
}
```

AbstractMessageTransformer – override transform() method
It is used when whole Mule Message needs to be access or used within Transformer

```java
import org.mule.transformer.AbstractMessageTransformer;

public class OrderToHtmlTransformer extends AbstractMessageTransformer
{
    public Object transform(MuleMessage message, String encoding) throws TransformerException
}
```

85. Git commands

To add your other files please follow the below commands.
git status – All the non-committed code will appear in RED.

git add *file-to-be-added (Follow this step till your all files are added.)*
git status – All added files will appear in green color
git commit
i. Proper comment
ii. Then press ESC :wq
iii. Enter
git push

86. Difference between call out and service invoke?

Service Invoke: The Service Invoke is used to make a service request in either a request or response flow. The service may be Request/Response or One-Way. Multiple instances of the Service Invoke (primitive) are permitted in a flow, allowing a series of service invocations to be performed.

Callout: The Callout receives the message and calls the requested service and operation. There is a Callout node for each connected target operation in the flow.

– If the call is successful, the Callout Response node in the response flow receives the response message.
– If the call is unsuccessful, the Callout can be set to retry service invocations depending on the type of fault received.

87. How can you implement loop in flow?

Fan-in and Fan-out is used to implement the loop

88. What is the functionality of Fan-in and Fan-out?

This is used to Implement a loop in mediation

Fan-out: We can use the Fan Out to fire the output terminal once (with the input message) or to fire the output terminal *multiple* times. You can use Fan Out in isolation or as part of a Fan Out and Fan In combination.

Fan-In: Fan In is always partnered with a Fan Out in the same flow and acts as a decision point for when to continue flow execution. It receives a number of messages until a decision point is reached, at which point the last message to be received is propagated to the output terminal. The Fan In primitive can only be used in combination with Fan Out.

89. What is Shared Context?

Shared Context: Context is a temporary area which is created along with Service Message Object (SMO) in the Flows. Shared Context is a type of context.it is mainly used when we are using Aggregation process where we need to Iterate for Certain times. Shared Context maintains Aggregation data between Aggregation (FanOut and FanIn) primitives. The Content (data) which is present in the shared context BO does not persist across Request and Response flows i.e. The Data in the Shared Context which is used in Request flow can not be used again in Response flow.

90. What is Transient Context?

Transient Context: Used for passing values between connectors within the current flow — either the request flow or the responses flow. The transient context cannot link requests and responses and hence cannot be used across.

Used when you want to save an input message before a service invokes call (within a request or response flow). After the services invoke call, the next primitive can create another message by combining the service invoke response and the original message stored in the transient context.

91. What is the Difference between Stop and fail?

Stop: Stops a particular path in the flow, without generating an exception.
Fail: Generates an exception in the flow.

92. What is mule MMC? Why it is used?

MMC Mule Management Console is an management and monitoring tool designed specifically for Mule ESB instances. MMC provides a set of functionality for managing and monitoring running Mule instances, Mule clusters, applications within Mule instances, and the flows within those applications. We can look into specific transactions also with the help of it.