* Data Weave is an essential tool that lets you weave complex data transformations into your Mule applications to quickly transform and map Mule message payloads to other data structures.
* Mule, the runtime engine of Any point Platform, is a lightweight Java-based enterprise service bus (ESB) and integration platform that allows developers to connect applications together quickly and easily, enabling them to exchange data.

Mule capabilities

1. Service creation and hosting
2. Service mediation
3. Message routing
4. Data transformation

* Sub flows are like functions which can be used in main flows. Processing strategy and exception strategies were override by sub flow from parent flow.
* In two different flows the processing strategy and exception strategies were different for both.
* Two flows were connected by flow reference.
* Flow variables are visible in sub flow only when we use flow reference.
* Session variables can be visible without using flow reference. For session variables we have to use VM.
* Session variables and flow variables are not accessible when we use HTTP. But when we use VM and flow reference they are accessible.
* When you are calling child flow from parent flow it must be request responses.
* Variables (session and flow) are assigned in URL.

Mule Expression Language (MEL):

* We can use variable name directly, mule first checks in flow variable and then in session variable, if it is not found in both then it will through error.
* #[firstname] is same as #[flowVars.firstname].
* To disable auto-resolution just place an cfg: **<configuration><expression-language autoResolveVariables=”false”/></configuration>**
* A transport or "provider", is a set of objects that add support to Mule to handle a specific kind of transport or protocol.
* A connector is the object that sends and receives messages on behalf of an endpoint.
* A router is the object that do something with messages once they have been received by a connector, or prior to being sent out by the connector
* A filter optionally filters incoming or outgoing messages that are coming into or going out from a connector, Filters are used in conjunction with Routers.
* A transformer optionally changes incoming or outgoing messages in some way
* This is usually done to make the message format useable by a downstream function

Mule Event Flow

1. Message Receiver Endpoint: Some event triggers a message flow

A file being written into a folder

A message arriving on a message queue

A record in a database

Data written to a socket

1. Inbound Router: The inbound router is the fist step in a message. Functions typically performed by an inbound router

Filtering

Remove duplicate messages

Matching messages

Aggregation (combining)

Re-sequence data

Forwarding

1. Interceptor: Used to intercept message flow into your service component

Used trigger monitor/events or interrupt the flow of the message

Example: an authorization interceptor could ensure that the current request has the correct credentials to invoke the service.

1. Inbound Transformer: If the inbound data is not in the correct format for the service it must be transformed at this point
2. Service Invocation: The actual service is performed

In mule, this is generally a Java object

Service invocation can also be a "pass through"

1. Outbound Router: Dispatching the data to all the relevant endpoints
2. Outbound Transformer: Any transformations that needs to be done on the message after a service has been performed on the message can be executed before it is put into the endpoint

* Web service is a function or program in any language that can be accessed over HTTP. Message format can be XML or JSON or any other program as long as the other programs can understand and communicate. Web services can be synchronous or asynchronous. Any web service has server-client relationship. Any web service can have multiple clients. Eg: When a travel portal is selling tickets of an airliner, Portal is client and the Airline is the server as it is selling its service.
* REST stands for representaional State Transfer or RESTful web service. REST is a client-server architecture which means each unique URL is representation of some object or resource. Any REST API developed uses HTTP methods explicitly and in a way that’s consistent with the protocol definition. This basic REST design principle establishes a one-to-one mapping between create, read, update, and delete (CRUD) operations and HTTP methods.
* RAML – RESTful API Modeling Language
* RAML is similar to WSDL, it contains endpoint URL, request/response schema, HTTP methods and query and URI parameter.
* RAML helps client (a consumer of the service) know, what the service is and what/how all operations can be invoked.
* RAML helps the developer in creating the initial structure of this API. RAML can also be used for documentation purpose.
* A RAML contains:
* Endpoint URL with its Query parameters and URI parameters,
* HTTP methods to which API is listening to (GET, POST, PUT, DELETE),
* Request and response schema and sample message,
* HTTP response code that an API will return (eg: 200, 400, 404, 500)
* Sub flow – A sub flow is always synchronous. It is similar to a synchronous flow, sub flow executes in the same thread of the calling process. Calling process triggers the sub-flow and waits for it to complete and resumes once the sub-flow has completed.

Sub flow always processes messages synchronously (relative to the flow that triggered its execution).

Sub flow executes in the same thread of the calling process. Calling process triggers the sub-flow and waits for it to complete and resumes once the sub-flow has completed.

Sub flow inherits processing strategy and exception handling strategy from the parent/calling flow.

Use – It can be used to split common logic and be reused by other flows.

* Synchronous Flow– Same as sub-flow, the only difference is that in synchronous flows you need to separately define an exception strategy to it, it does not inherit the exception strategy of its calling flow.
* Asynchronous Flow – As in sub-flow and synchronous flow, calling process triggers the sub-flow and waits for it to complete; for asynchronous flow the flow, calling process triggers an asynchronous flow and moves ahead to its next activity. An asynchronous flow executes in parallel to its calling/parent flow in a different thread. An asynchronous flow does not return its output it its parent/calling flow.
* Private Flow – flow that does not have an inbound connector in the source. Means a private flow cannot start of its own on receiving the inbound message as it does not have any inbound connector, A private flow can only be called using flow-ref same as sub-flow.

Types of variables

1. Flow Variable
2. Session Variable
3. Record Variable

Types of Exception Handling

1. Default Exception Handling
2. Global Exception Handling
3. Catch Exception Handling
4. Choice Exception Handling
5. Rollback Exception Handling

Flow Processing Strategies

1. Synchronous Flow Processing Strategy
2. Queued Flow Processing Strategy
3. Asynchronous Flow Processing Strategy
4. Thread Per Processing Strategy
5. Queued Asynchronous Flow Processing Strategy
6. Non-blocking Flow Processing Strategy
7. Custom Processing Strategy

* Creating SOAP Service – We can create a SOAP service same as we create Mule Project with RAML, the only change is instead of RAML we need to import Concert WSDL.
* Consuming SOAP Service – We can use Web Service Consumer or CXF component in our mule flow to access/consume SOAP service.