Mule Capabilities:

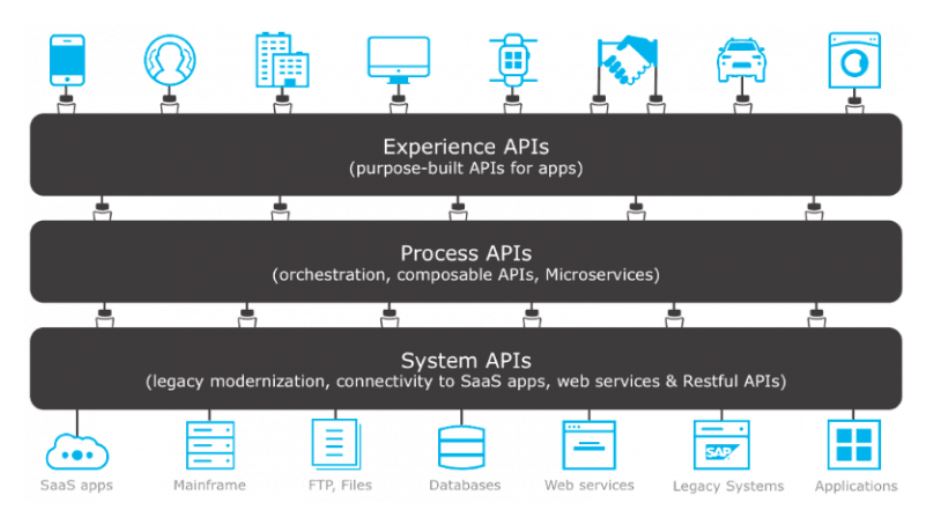
1. Service creation and hosting: Mule will expose and host reusable services using ESB as a light weight service container.
2. Service Mediation: Mule will shield services from message formats and protocols, separate business logic from messaging and enable location independent service calls
3. Message Routing: Route, filter, aggregate and re-sequence messages bases on the context and rules
4. Data Transformation: Exchange data across varying formats and transport protocols

API-led connectivity is an approach which defines methods for connecting the assets using reusable API building blocks. This will be consisting of 3 distinct layers

APIs provides a means of hiding the complexity from the user while exposing data and providing downstream insulation from interface changes.

* API template encapsulate 5 common integration patterns

1. Migration
2. Broadcast
3. Aggregation
4. Correlation
5. Bi-directional synchronization



System API:

* API template provides system APIs for creating the building blocks that can be utilized by anyone within the organization looking for the access of the same data
* System APIs will expose the data via set of RESTful Services that are designed in RAML, which makes it easily consumable by any other developer in the organization

Process API:

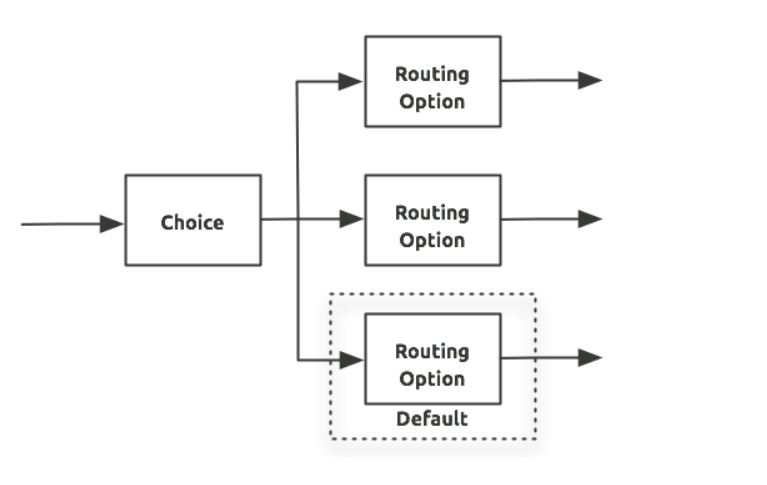
* This will encapsulate the underlying business processes that interact with the source and target systems

Experience API:

* Consuming client

**Routers (Flow Control):**

Routers will route the messages to various destinations in a mule flow. Some may incorporate the logic and analyze and possibly transform messages before the routing takes place (reordering, splitting, combining, broadcast to multiple building blocks)

* This is used to send the same message to multiple target message processors. All flow control can be used to broadcast the same message to more than one processor component. Mostly all the message will be sent to the connectors. The connectors may have filter associated to it.
* **Choice Flow Control** will dynamically route the messages based on the payload or the properties, based on the conditional programming (if/ then/ else).
* Expressions are utilized to evaluate the content of a message, then it routes the message to one of the routing options within the scope. If none of the expressions evaluate to true, then choice of flow control directs to the default route.
* 
* **Scatter-gather:** The routing message processor will send a request to multiple targets concurrently, collecting the responses from all the routes and aggregates into a single message. It will be replacing the All message processor. This will be increasing the efficiency of the application.

**Munit:** Munit is a mule application testing (unit & functional) framework which will be allowing us to build the automated tests for our mule integration of the application.

It can be integrated with Maven and Surefire

Use cases:

* Call mule client from mule code
* Mock outbound endpoints
* Mock message processors
* Verify message processor calls
* Spy and message processor

**Exceptions:**

Faults that will be encountered with in a flow can be called as exception. Generally, Default exception will be invoked. If we want to provide more details about an exception or if we want to customize an exception, then we can utilize on of the exception provide in the MuleSoft.

1. **Catch Exception Strategy**: This exception will be handling all the exceptions that are occurred within the parent flow and process them overriding the default exception strategy.

Ex: Consider the flight booking where you need to have the access to book the seat, if there are no seats available then we will catch the exception then respond to the client

1. **Choice Exception Strategy:** It will handle all the exceptions that are encountered within the parent flow – examines the message content and then it will route to the 1st exception strategy that is true
2. Any number of catch and rollback exceptions can be placed with in the choice exception
3. Choice with in a choice cannot be done

Ex: A catch exception strategy to process and discard the already processed exceptions

A second catch exception strategy to process all validation orders and send them to invalid queue

A roll back exception to roll back the order transaction to rollback

1. **Reference Exception Strategy**: Create a global exception strategy and within the flow make a reference to the global exception file so whenever exception arises it will be referred to the global exception strategy
2. **Rollback Exception Strategy:** If a message throws an exception then a roll back is done to reprocess the message. Mostly roll back exception will be implemented in the cases where transaction is utilized. Additionally, you can use to
3. Manage unhandled exception- exception that the application fails to handle.
4. Put inflows in which messages require redelivery

**Collection Splitter:**  This connector will split a Mule message into fragments and pass them to the next processor. These are generally written in MEL

**Collection Aggregator:**  Whenever message splits, 3 new outbound variables into each of its output fragments.

1. Mule\_Corelation\_group\_size
2. Mule\_corelation\_sequence
3. Mule\_corelation\_Id

As the group id, sequence and id of a message is known then those messages can be aggregated easily

**Shared Resources:**  Mule will be allowing us to share the common resources that are present in a domain, these are known as shared resources, to use this, we need to create a Mule Domain Project and then reference it with the applications that require this element. Mule application must be associated with only one domain at a time.

* We can share multiple services within the domain through the same port.
* Services can be shared with a well-defined interface
* Consistency will be maintained upon any changes in the file.

**Flow Ref vs VM:**  VM endpoints enable message redelivery in your exception handling strategy which is not possible in flow-ref’s. VM’s can provide this internally using queue to hold the messages.

In case of flow-ref’s, if an exception occurs in the flow, even we have a rollback strategy, it will not be executed because there is no internal queue involved.

**Flow vs Sub-Flow:** Flow is a message processing block that has its own methods, exceptions,

Types of Flows:

1. Sub-flow: A sub-flow will basically process the messages synchronously and will inherit both the processing and the exception strategy from the parent. When a sub-flow gets invoked the process in the flow will be resumed until it will receive a response from the sub-flow.
2. Synchronous flow
3. Asynchronous flow

A flow can have multiple sub-flows and the sub-flow can have multiple triggering

**Types of Variables:**

1. Flow Variables: The scope of this variable is restricted with in the same flow and its sub-flows
2. Session variables: These variables can be accessed across all the projects i.e. they are global
3. Record Variables: Accessed with in the variable

Difference between SOAP and REST

|  |  |
| --- | --- |
| **SOAP** | **REST** |
| Traditional and complex | Architectural style and simple |
| Has inbuilt security | Inbuilt security by TLS |
| Permits XML format only | Permits xml, html, Json |
| Uses service interfaces to expose the business logic | Utilizes URI to expose the business logic |
| Everything within the soap is tightly coupled | Loosely coupled |
| It is Asynchronous | Synchronous |
| Has more bandwidth | Less bandwidth |
|  | Consider everything as a resource |
|  | Speed is more as of less parsing |