

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Anypoint | [https://anypoint.mulesoft.com](https://anypoint.mulesoft.com/) | Cognizant | Cognizant Employee Id | Your choice | Cognizant | |  | | --- | | [mahalingam.muthusamypandiyan@cognizant.com/GanapathySubramanian.Rajagopal@cognizant.com  will send an invite upon your request to him through email ; Use the Invite link and SIGN-UP and do not use Sign-In button ;](mailto:mahalingam.muthusamypandiyan@cognizant.com/GanapathySubramanian.Rajagopal@cognizant.com%20will%20send%20an%20invite%20upon%20your%20request%20to%20him%20through%20email%20;Use%20the%20Invite%20link%20and%20SIGN-UP%20and%20do%20not%20use%20Sign-In%20button%20;) | |  |

ADO

Access

ECOM

Create API with 2 tier, exp and sys API

Pessyapi(PBM Team)

Pharmaciessysapi(PBM team)

http:

port: "8081"

sfdc:

username: "sujijothy@gmail.com"

password: "Ayyappa1"

token: "srbNMy9AsEnSrdSDuEUzrRqN3"

file:

accountsDir: "C:/suja/APDevFundamentals4.3\_studentFiles\_13jul2020/APDevFundamentals4.3\_studentFiles\_13jul2020/resources"

db:

host: "iltdb.learn.mulesoft.com"

port: "3306"

user: "mule"

password: "mule"

database: "training"

maven:

1. Maven is a powerful project management tool that is based on POM (project object model). It is used for projects build, dependency and documentation.
2. We can add jars and other dependencies of the project easily using the help of maven.

### Pros and Cons of using Maven

**Pros:**

* 1. Maven can add all the dependencies required for the project automatically by reading pom file.
  2. One can easily build their project to jar, war etc. as per their requirements using Maven.
  3. Maven makes easy to start project in different environments and one doesn’t needs to handle the dependencies injection, builds, processing, etc.
  4. Adding a new dependency is very easy. One has to just write the dependency code in pom file.

**Cons:**

* 1. Maven needs the maven installation in the system for working and maven plugin for the ide.
  2. If the maven code for an existing dependency is not available, then one cannot add that dependency using maven.

**When should someone use Maven?**

One can use the Maven Build Tool in the following condition:

* 1. When there are a lot of dependencies for the project. Then it is easy to handle those dependencies using maven.
  2. When dependency version update frequently. Then one has to only update version ID in pom file to update dependencies.
  3. Continuous builds, integration, and testing can be easily handled by using maven.
  4. When one needs an easy way to Generating documentation from the source code, Compiling source code, Packaging compiled code into JAR files or ZIP files.

Dns – domain name system:

### [What Is DNS](https://www.cloudflare.com/learning/dns/what-is-dns/)

The Domain Name System (**DNS**) is the phonebook of the Internet. Humans access information online through domain names, like nytimes.com or espn.com

### The Domain Name System resolves the names of internet sites with their underlying IP addresses adding efficiency and even security in the process.

DNS is a directory of names that match with ip address.

## How DNS servers work

The DNS directory that matches name to numbers isn’t located all in one place in some dark corner of the internet. With [more than 332 million domain names listed at the end of 2017](http://www.verisign.com/en_US/domain-names/dnib/index.xhtml?section=cc-tlds), a single directory would be very large indeed. Like the internet itself, the directory is distributed around the world, stored on domain name servers (generally referred to as DNS servers for short) that all communicate with each other on a very regular basis to provide updates and redundancies.

Each domain can correspond to more than one IP address. In fact, some sites have hundreds or more IP addresses that correspond with a single domain name. Another reason for the distributed nature of the directory is the amount of time it would take for you to get a response when you were looking for a site if there was only one location for the directory, shared among the millions, probably billions, of people also looking for information at the same time. That’s one long line to use the phone book.

DNS server you use will be established automatically by your network provider when you connect to the internet. If you want to see which servers are your primary nameservers

## **Firewall defined**

A firewall is a [**network security**](https://www.forcepoint.com/cyber-edu/network-security) device that monitors incoming and outgoing network traffic and permits or blocks data [**packets**](https://www.forcepoint.com/cyber-edu/packet-loss) based on a set of security rules. Its purpose is to establish a barrier between your internal network and incoming traffic from external sources (such as the internet) in order to block malicious traffic like viruses and hackers.

Firewalls carefully analyze incoming traffic based on pre-established rules and filter traffic coming from unsecured or suspicious sources to prevent attacks. Firewalls guard traffic at a computer’s entry point, called ports, which is where information is exchanged with external devices.

Think of IP addresses as houses, and port numbers as rooms within the house. Only trusted people (source addresses) are allowed to enter the house (destination address) at all—then it’s further filtered so that people within the house are only allowed to access certain rooms (destination ports), depending on if they're the owner, a child, or a guest. The owner is allowed to any room (any port), while children and guests are allowed into a certain set of rooms (specific ports).

## **Types of firewalls**

Firewalls can either be software or hardware, though it’s best to have both. A software firewall is a program installed on each computer and regulates traffic through port numbers and applications, while a physical firewall is a piece of equipment installed between your network and gateway.

Packet-filtering firewalls, the most common type of firewall, examine packets and prohibit them from passing through if they don’t match an established security rule set. This type of firewall checks the packet’s source and destination IP addresses. If packets match those of an “allowed” rule on the firewall, then it is trusted to enter the network.

Packet-filtering firewalls are divided into two categories: stateful and stateless.

Stateless firewalls examine packets independently of one another and lack context, making them easy targets for hackers. In contrast, stateful firewalls remember information about previously passed packets and are considered much more secure.

correlationId:

The correlationId is a unique value that is assigned automatically when an event is started, at the beginning of a flow. It can be obtained from the [predefined variable](https://docs.mulesoft.com/mule-runtime/4.3/dataweave-variables-context) #[correlationId].

You can think of it as the identifier of a [Mule Event](https://docs.mulesoft.com/mule-runtime/4.3/about-mule-event).

Scripting module:

Using this module we can execute javascript/python/groovy coding, by adding Execute using scripting module

And choosing engine: Nashorn (JavaScript), there are other options to choose ruby, python, groovy and adding code in the code field. For eg: var list = eval(payload);

var sum = 0;

for (var i in list)

sum += list[i];

payload = sum;

to add for passing { "a": 3, "b": 4 } value in post man

Training api

**Client ID:**

951ba9d621404534a1af1cafecf4d678

**Client Secret:**

9B05E05036f742D5Ba49b5017c7D8100

Content-based routing is used to examine messages and route them to the correct channel or destination depending on a message's content. You should always use content-based routing when you want to route messages to the correct destination. nts on the basis of Customer ID.

For debugging,

It takes more time, so before debugging in arc, file->settings->application settings, ->locate request timeout setting , click arrow next to it,change the request timeout to 300 seconds.

Choice router –

It is used to send an event to one route based on conditional logic

Scatter-gather

It is used to send an event concurrently to multiple routes

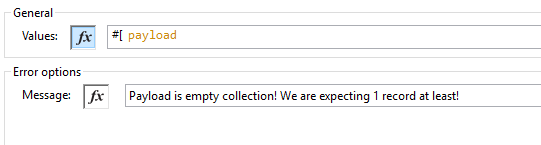
A collection of all result is returned, use dw to flatten the collection.

(eg.,getDeltaflight,getUnitedFlights)yields, all flight details, by processing concurrently deltaflights and united flights yielding all flight details

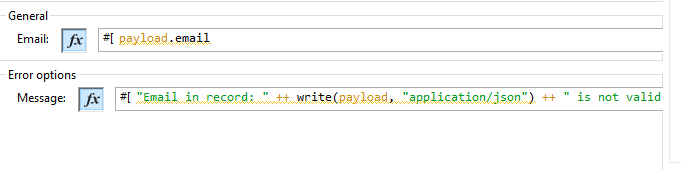
Use Validation (validator) Module:

It is used to specify whether the event can proceed in a flow

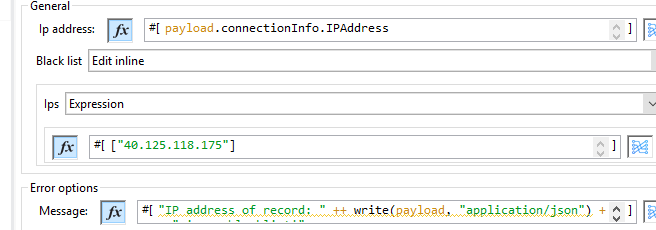
If payload is not empty collection



If payload.email is email:



To validate if ip address is not on blacklist



Scheduler

component to create a new flow that executes at a specific frequency say for eg.every 5 secs/cron at particular condition

JMS Publish operation to send a message to a JMS queue

In the Publish properties view, set the

following values:

•

Display

N

ame: J

MS

accountsQ

•

Connector configuration: JMS\_Config

•

Destination: accountsQ

•

Destination type: QUEUE

Watermarking:

Rename file name:

attributes.fileName ++ 'backup'

By default watermarkmode: disabled

If we did’t enable

modified\_timestamp, the file will be keep on processing every 5 secs, eventhough it is processed already.

Inorder to make it remember, not to process already processed file,in watermarkmode: choose, modified\_timestamp,

If we try to modify the file in the input folder, say for example, if I delete one record, then it will be reprocessed.

Similarly if u chose created\_timestamp option then, then the file endpoint stores the creation time,so even if u modified it won’t reprocess, ie if u want to process the file only once when it is created then u should choose created\_timestamp

How the watermark is stored and not processed even if we restart the system. The previous object store state is stored in mule runtime(.mule folder). The object store state is retained.

If u want to ask u to check, whether u want to clear or retain the state.

**Object stores** are used to store an object. Mule 4 uses object stores to retrieve the objects later and to keep the data persistent. **Object store** has Contains, Dual store, Remove, Retrieve all keys, Retrieve and store, Store operations.

for running, right click run as (mule application configure), there is an option clear application data, by default it is configured as never, if u change to prompt, whenever u r running the appln, u will get an option whether u wan to clear the cache, u can choose whichever option u want, say for example if u give clear it won’t retain previous state.

If u did’t clear, based on watermarking it won’t process the already processed file.

Here file endpoint: on new or updated file

Payload after the file endpoint is stream object, i.e the payload after the file endpoint is not the entire file.

That’s the reason,if u have transform msg and get payload and put in foreach with logger, everytime, it has one record of the file.

OnTableRow:

This operation selects from a table at a regular interval and generates one message per obtained row. Optionally, you can provide watermark and ID columns. If a watermark column is provided, the values taken from that column are used to filter the contents of the next poll, so that only rows with a greater watermark value are returned. If an ID column is provided, this component automatically verifies that the same row is not picked twice by concurrent polls.

If any row added , to check we will use ontable row we can check by cron/particular time

Say for example, based on the watermark column like product\_id is stored in object store. If we r running next time, it picks up from the watermark after that .

Watermarking:

Remembering the previous state

Normally whatever data we want to remember we will keep it in objectstore in mule.

Here in this ontablerow also keeping the last product\_id in the object store. Usually objectstore stores keyvalue pairs, as column as key and value as column values

**Watermark** is used to poll for new set of records instead of polling for same resources repeatedly. **Watermarking** is a common use of the object store.

Mulesoft:

Is a data integration platform built to connect variety of data sources and application in the cloud and on-premises. Mulesoft supports reusable API.

With Anypoint:

We can do

Integrate appln, design API’s, build & test API’s, expose aPI’s on cloud, provide access to library of API’s

API:

It provides info for how to communicate with a s/w component, defining the

i)operation, what to call,

ii)i/ps(what to send)

iii)o/ps(what u get back)

API Gateway runtime – ctrls access to APIby enforcing policies

Is part of mule runtime but requires separate license

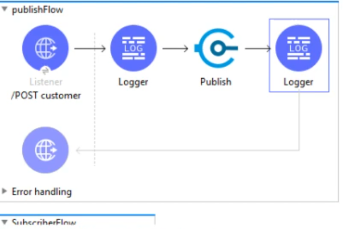
API proxy-

Is an appln that controls access to a web service restricting access and usage through the use of an API gateway

Cloudhub-

Is the platform as a service component of anypoint platform host mule runtime on AWS

VM connector:



If we r assigning first logger with before publish

Second logger- after publish

Then

If we use publish and listenr, then It will function asynchronously.

i.e It will first print before publish(logger) and publish and logger(after publish). In parallel consume will function.

So o/p:

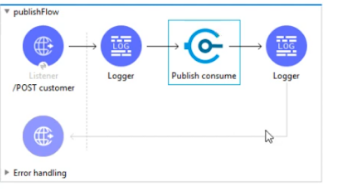
Before publish,after publish then it prints payload

If we want to happen queing synchronously, instead of publish,we shud use publishconsume.

If we use publish consume, it wait for the msg to consume, and then the entire flow will get invoked

o/p(loggerthen publishconsume, once the sonsume flow executed, ctrl comes back to main flow)

before publish, payload, after publish



* **VM Consume**-This component acts as a **event** **processor** only to poll a queue.Since this acts as a event processor it has to be called either from a **event** **source** or through flow-ref and cannot act as a event source.
* **VM Listener**-This acts as the **event source** only and gets triggered whenever a message is post to the queue on which this VM Listener is polling.This component cannot act as a event processor.

It is an interface, which acts as a s/w intermediary that allows two appln to talk to each other

VM is Mule's internal transport for messaging/queueing. The VM transport is for intra-JVM communication between Mule flows. So, that means when you use a VM in your flow, you can communicate between different flows in the application. It can only be used by Mule applications. When creaing a VM queue it can only be accessed by the Mule application that creates it (Cloudhub for example) OR it can be reused by same Mule apps running in a domain project or cluster. No existing broker infrastructure needs to be setup. Supports persistent and transient.

Anypoint MQ is Mulesoft's Cloud Messaging platform. This can be used by other applications - not just Mule. It can also be used across multiple Mule apps regardless of domains or cluster, well suited for Cloudhub applications. No infra setup, all in the cloud. Think Amazon SQS but a lot better and great integration with Anypoint Platform

JMS uses the Java Messaging Service protocol and requires an external JMS broker such as ActiveMQ. Can be used by any application that supports JMS connectivity.

**For reliable we use JMS, there isno chance of losing data.**

**VM –adv queue**

**JMs- more robust**

API Autodiscovery is a mechanism that manages an API from API Manager by pairing the deployed application to an API created on the platform. API Management includes tracking, enforcing policies if you apply any, and reporting API analytics. Critical to the Autodiscovery process is identifying the API by providing the API name and version.

Map : Returns an array that is the result of applying a transformation function (lambda) to each of the elements. ([https://docs.mulesoft.com/mule-runtime/3.9/dataweave-operators#](https://docs.mulesoft.com/mule-runtime/3.9/dataweave-operators) map)

2. MapObject: Similar to Map, but instead of processing only the values of an object, it processes both keys and values as a tuple. (

### Using Map to Return an Array

(':array', ':function') ⇒ :array

Returns an array that is the result of applying a transformation function (lambda) to each of the elements. The lambda is invoked with two parameters: **index** and the **value**. If these parameters are not named, the index is defined by default as **$$** and the value as **$**.

Transform

%dw 1.0

%output application/json

---

users: ["john", "peter", "matt"] map upper $

Output

{

"users": [

"JOHN",

"PETER",

"MATT"

]

}

### Using Map on an Object

(':object', ':function') ⇒ ':array'

Returns an array with the values that result out of applying a transformation function (lambda) to each of the values in the object. The keys of the original object are all ignored by this operation and the object is treated as an array. To have access to the keys, you can use the operation **mapObject** instead. The lambda is invoked with two parameters: **index** and the **value**. If these parameters are not named, the index is defined by default as **$$** and the value as **$**. The index refers to the position of a key:value pair when the object is treated as an array.

|  |
| --- |
| See [Map Object](https://docs.mulesoft.com/mule-runtime/3.9/dataweave-operators#map-object) if what you want is to process both keys and values instead of just values. |

Transform

%dw 1.0

%output application/json

%var conversionRate=13.45

---

priceList: payload.prices map (

'$$':{

dollars: $,

localCurrency: $ \* conversionRate

}

)

### Input: XML

Input

<prices>

<basic>9.99</basic>

<premium>53</premium>

<vip>398.99</vip>

</prices>

### Output: JSON

Output

{

"priceList": [

{

"0": {

"dollars": "9.99",

"localCurrency": 134.3655

}

},

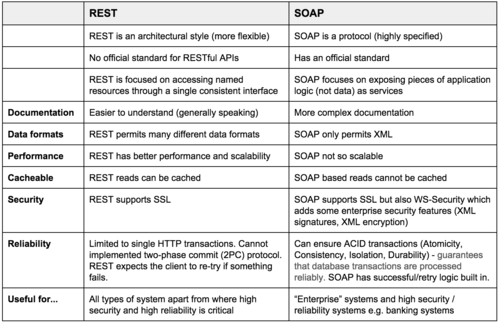
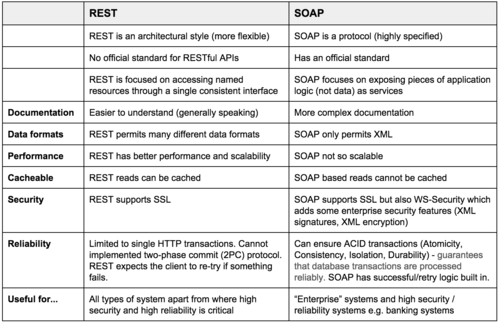
{

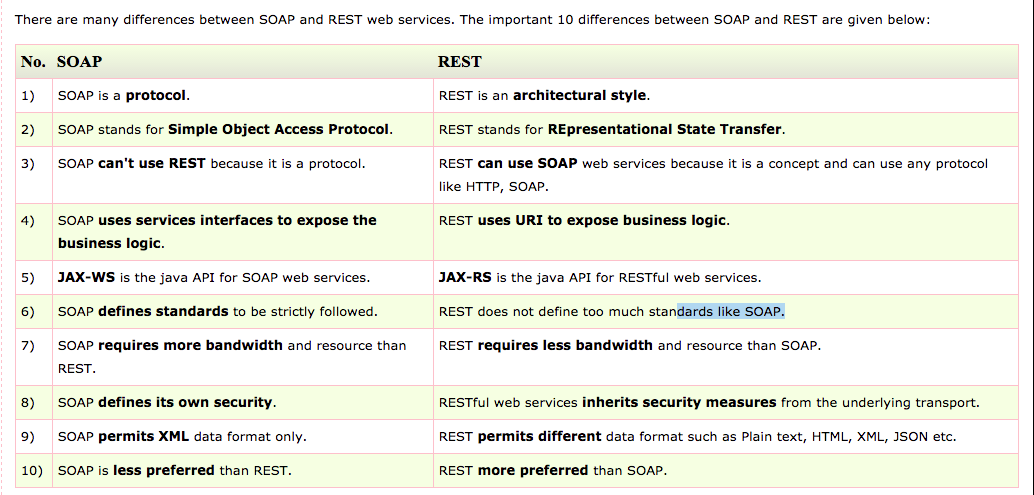
"1": {

"dollars": "53",

"localCurrency": 712.85

}





## What is WSDL?

**Web Services Description Language** (WSDL) is an XML-based file that basically tells the client application what the web service does. The WSDL file is used to describe in a nutshell what the web service does and gives the client all the information required to connect to the web service and use all the functionality provided by the web service.

A WSDL document is used to describe a web service. This description is required, so that client applications are able to understand what the web service actually does.

* The WSDL file contains the location of the web service and
* The methods which are exposed by the web service.

The WSDL file itself can look very complex to any user, but it contains all the necessary information that any client application would require to use the relevant web service.

Below is the general structure of a WSDL file

* Definition
* TargetNamespace
* DataTypes
* Messages
* Porttype
* Bindings
* service

One key thing to note here is that definition of messages, which is what is passed by the SOAP protocol is actually defined in the WSDL document.

The WSDL document actually tells a client application what are the types of SOAP messages which are sent and accepted by the Web service.

In other words, the WSDL is just like a postcard which has the address of a particular location. The address provides the details of the person who delivered the postcard. Hence, in the same way, the WSDL file is the postcard, which has the address of the web service which can deliver all the functionality that the client wants.

<!-- WSDL definition structure -->

<definitions

name="Guru99Service"

targetNamespace=http://example.org/math/

xmlns=http://schemas.xmlsoap.org/wsdl/>

<!-- abstract definitions -->

<types> ...

<message> ...

<portType> ...

<!-- concrete definitions -->

<binding> ...

<service> ...

</definition>

1. The **<types>** tag is used to define all the complex datatypes, which will be used in the message exchanged between the client application and the web service. For example, there could be a data type called EmployeeDataType which could have 2 elements called "EmployeeName" of type string and "EmployeeID" of type number or integer. Together they form a data structure which then becomes a complex data type.
2. The **<messages>** tag is used to define the message which is exchanged between the client application and the web server. These messages will explain the input and output operations which can be performed by the web service. An example of a message can be a message which accepts the EmployeeID of an employee, and the output message can be the name of the employee based on the EmpoyeeID provided.
3. The **<portType>** tag is used to encapsulate every input and output message into one logical operation. So there could be an operation called "GetEmployee" which combines the input message of accepting the EmployeeID from a client application and then sending the EmployeeName as the output message.
4. The **<binding>** tag is used to bind the operation to the particular port type. This is so that when the client application calls the relevant port type, it will then be able to access the operations which are bound to this port type. Port types are just like interfaces. So if a client application needs to use a web service they need to use the binding information to ensure that they can connect to the interface provided by that web service.
5. The **<service>**tag is a name given to the web service itself. Initially, when a client application makes a call to the web service, it will do by calling the name of the web service. For example, a web service can be located at an address such as **http://localhost/Guru99/Tutorial.asmx** . The service tag will actually have the URL defined as **http://localhost/Guru99/Tutorial.asmx**, which will actually tell the client application that there is a web service available at this location.

## Why WSDL

A web service is an important component in building modern day web applications. Their main purpose is to allow multiple applications built on various programming languages to talk to each other. For instance, we can have a .Net web application talks to a[Java](https://www.guru99.com/java-tutorial.html)application via a Web service.

## **What is HTTP?**

The Hypertext Transfer Protocol (HTTP) is designed to enable communications between clients and servers.

HTTP works as a request-response protocol between a client and server.

Example: A client (browser) sends an HTTP request to the server; then the server returns a response to the client. The response contains status information about the request and may also contain the requested content.

## **The GET Method**

**GET is used to request data from a specified resource.**

**GET is one of the most common HTTP methods.**

Note that the query string (name/value pairs) is sent in the URL of a GET request:

/test/demo\_form.php?name1=value1&name2=value2

**Some other notes on GET requests:**

* GET requests can be cached
* GET requests remain in the browser history
* GET requests can be bookmarked
* GET requests should never be used when dealing with sensitive data
* GET requests have length restrictions
* GET requests are only used to request data (not modify)

## **The POST Method**

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

The data sent to the server with POST is stored in the request body of the HTTP request:

**Some other notes on POST requests:**

* POST requests are never cached
* POST requests do not remain in the browser history
* POST requests cannot be bookmarked
* POST requests have no restrictions on data length

## **The PUT Method**

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

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

## **The HEAD Method**

**HEAD is almost identical to GET, but without the response body.**

In other words, if GET /users returns a list of users, then HEAD /users will make the same request but will not return the list of users.

HEAD requests are useful for checking what a GET request will return before actually making a GET request - like before downloading a large file or response body.

**The DELETE method deletes the specified resource.**

## **The OPTIONS Method**

**The OPTIONS method describes the communication options for the target resource.**

# [Status codes](https://www.w3.org/Protocols/HTTP/HTTP2.html)

## Success 2xx

These codes indicate success. The body section if present is the object returned by the request. It is a MIME format object. It is in MIME format, and may only be in text/plain, text/html or one fo the formats specified as acceptable in the request.

### OK 200

The request was fulfilled.

### CREATED 201

## 304-not modified, nothing was modified by reqest

401-unauthorized

Does not have access to resource

404-resource not found, uri is not recognized by the server

500-server error

## Error 4xx, 5xx

The 4xx codes are intended for cases in which the client seems to have erred, and the 5xx codes for the cases in which the server is aware that the server has erred. It is impossible to distinguish these cases in general, so the difference is only informational.

The body section may contain a document describing the error in human readable form. The document is in [MIME](https://www.w3.org/Protocols/HTTP/References.html" \l "z1) format, and may only be in text/plain, text/html or one for the formats specified as acceptable in the request.

### Bad request 400

The request had bad syntax or was inherently impossible to be satisfied.

### Unauthorized 401

### The parameter to this message gives a specification of authorization schemes which are acceptable. The Not found 404

The server has not found anything matching the URI given

### Internal Error 500

The server encountered an unexpected condition which prevented it from fulfilling the request.

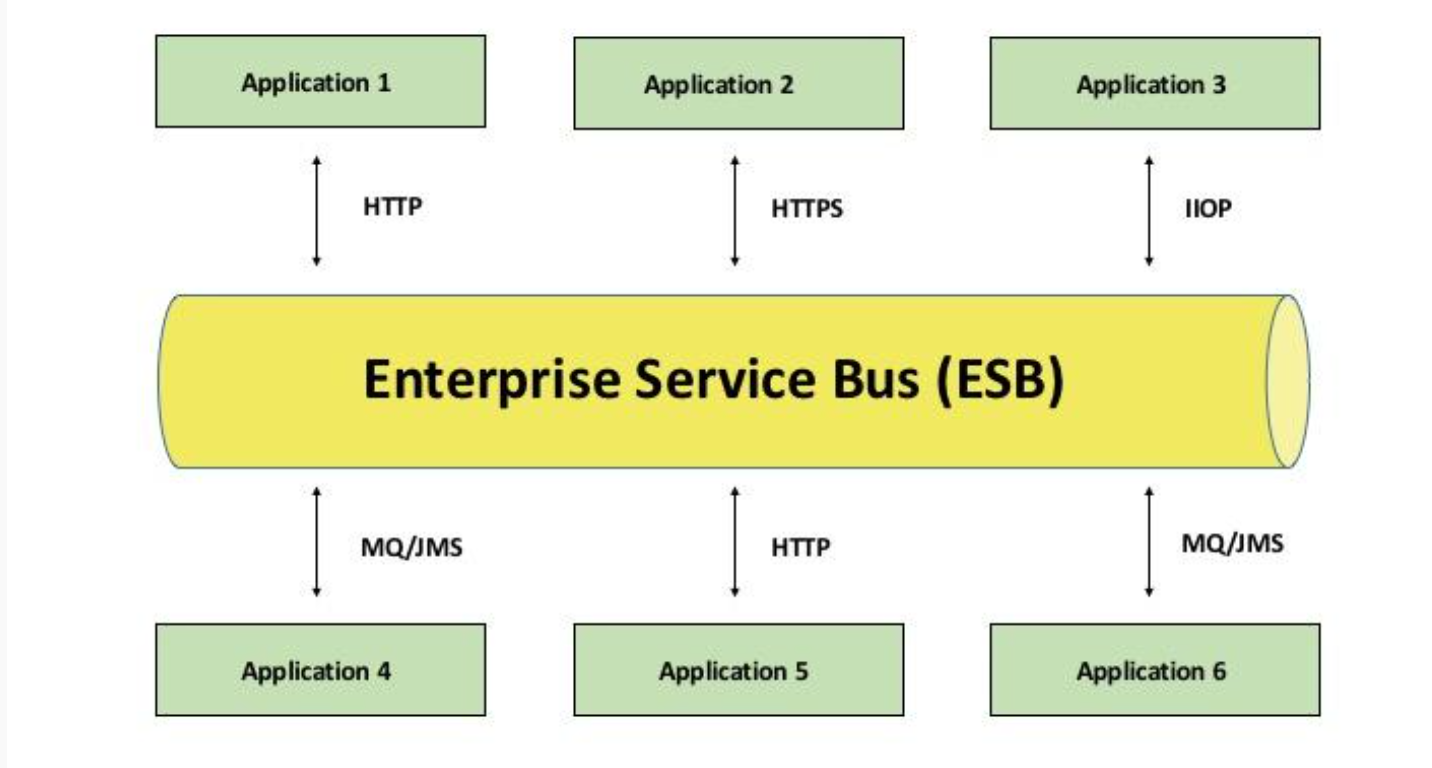
**Define Mule ESB**

It is an integration platform and a lightweight enterprise service bus (ESB) based on Java. It enables developers to easily and quickly connect applications and exchange data. With Mule ESB, developers can easily integrate existing systems irrespective of how different are the technologies that the application use such as Web Services, HTTP, JDBC, JMS and the like.

An ESB provides a secure, scalable and cost-effective infrastructure that enables real-time data exchange among many systems.

An ESB can provide

Messaging, Web Services , Data Transformation, Routing



## An Enterprise Service Bus (ESB) is a

## 1. Middleware software platform

## 2. It used Application to Application communication.

## Why ESB:

## Lets Consider that there are two systems that need to exchange data.

## Team plan and implement a solution (point to point) that allows these systems to communicate.

## Over couple of years, team deploys several more systems that need to interact with each other as well as the existing two systems.

## 

**What is the benefit of using Mule ESB?**

Mule ESB is an integration framework that is lightweight and highly scalable. It enables developers in starting small applications and also in connecting different applications. With Mule managing the exchanges between components and applications transparently and ESB taking care of a variety of applications, it is easy to integrate third-party applications with the help of Mule.

**What are the various types of Exception Handling?**

* Global Exception Handling
* Catch Exception Handling
* Choice Exception Handling
* Default Exception Handling
* Rollback Exception Handling

**What are the characteristics of Mule ESB?**

An ESB is used for the purpose of integration with an approach that is service-oriented. Its features include:

* Message Routing Service
* Message Transformation Service
* Set of Service Container
* Web Service Security

**In Mule, how do you develop and consume SOAP services?**

SOAP services can be created just like how we create a Mule project by using RAML. The difference here is that we need Concert WSDL importing rather than RAML. And SOAP services can be consumed by using our Mule flow CXF component or Web Service Consumer.

**How can you find out whether your project requires ESB?**

As every project might not require an ESB, you should analyze first to see if your project might benefit from ESB implementation. Certain things that should be at the front of your mind while you analyze the need for ESB are:

* If the project requires integration of more than 3 applications or services and if communication between two application is needed, it would be enough to use point to point integration
* Sometimes there will be a need for you to scale the project in the future where there might arise a need to interact with multiple services. This is required only by a few projects that perform heavy tasks
* If the project requires message routing abilities such as aggregating and forking message flows. This feature is not necessary for all projects
* You should have clarity on the architecture of the thing that needs to be achieved. A simple POC integration of small parts to find out the benefits is much better
* As most of the ESBs are on the expensive side, first evaluate whether your project budget permits ESB use

**Name the various kinds of Primitives that are used in Mediation.**

The following are the various kinds of primitives in mediation

* Endpoint Lookup
* Service Invoke
* DB lookup
* Data Handler
* Type Filter
* Message Element Setter
* Custom MediationFan-out
* Fan-in
* Header Setters
* Message Logger
* Even Emitter
* XSLT
* BO MapMessage Filter
* Fail
* Stop
* Sub Flow

**Name the various ESBs that are in the market**

There are different ESBs in the market, both licensed and open source. They are:

* JBoss Fuse ESB
* Mule ESB
* Talend

Microservices:

The central idea behind microservices is that some types of applications become easier to build and maintain when they are broken down into smaller, composable pieces which work together. Each component is continuously developed and separately maintained, and the application is then simply the sum of its constituent components. This is in contrast to a traditional, "monolithic" application which is all developed all in one piece.

Applications built as a set of modular components are easier to understand, easier to test, and most importantly easier to maintain over the life of the application. It enables organizations to achieve much higher agility and be able to vastly improve the time it takes to get working improvements to production. This approach has proven to be superior, especially for large enterprise applications which are developed by teams of geographically and culturally diverse developers.

There are other benefits:

* **Developer independence**: Small teams work in parallel and can iterate faster than large teams.
* **Isolation and resilience**: If a component dies, you spin up another while and the rest of the application continues to function.
* **Scalability**: Smaller components take up fewer resources and can be scaled to meet increasing demand of that component only.
* **Lifecycle automation**: Individual components are easier to fit into continuous delivery pipelines and complex deployment scenarios not possible with monoliths.
* **Relationship to the business**: Microservice architectures are split along business domain boundaries, increasing independence and understanding across the organization.

The common definition of microservices generally relies upon each microservice providing an API endpoint, often but not always a stateless REST API which can be accessed over HTTP(S) just like a standard web page. This method for accessing microservices make them easy for developers to consume as they only require tools and methods many developers are already familiar with.

## **What is Middleware?**

Middleware is the software that connects software components or enterprise applications. Middleware is the software layer that lies between the operating system and the applications on each side of a distributed computer network. **The function of middleware is to mediate interaction between the parts of an application, or between applications.**

Middleware is the infrastructure which facilitates creation of business applications, and provides core services like concurrency, transactions, threading, messaging, and the SCA framework for service-oriented architecture (SOA) applications. Middleware includes Web servers, application servers, content management systems, and similar tools that support application development and delivery.

We can write functions or create variables in dataweave to perform something complex or store values and reuse them later by calling them anywhere in dataweave body.

## **Service-Oriented Architecture**

SOA :

Service-oriented architecture is essentially a collection of services.

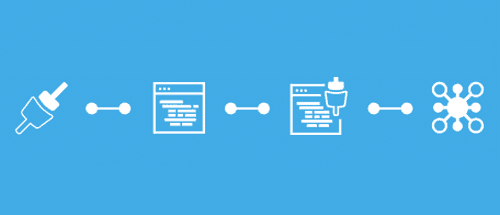
These services communicate with each other.

Services can range in size anywhere from small application services to very large enterprise services.



Service-Oriented Architecture (SOA) is an architectural style whose goal is to achieve loose coupling among diverse interacting software applications, enabling organizations to take advantage of existing investments in applications and systems. SOA facilitates the development of modular business services that can be easily integrated and reused, thus creating a flexible and adaptable infrastructure. Using a SOA approach, an organization can focus more resources and budget on innovation and on delivering new business services. Systems that can successfully use SOA can minimize the disruption of planned or unplanned outages in an enterprise.

Some of the advantages of using SOA are:

* **Reduction in development time and cost**: SOA services are easily reused and can be rapidly assembled into new, composite applications.
* **Lower maintenance cost:** Reusable services reduce the number and internal complexity of enterprise services.
* **High-quality services**: Increased service reuse creates high-quality services through multiple testing cycles from different service consumers.
* **Lower integration costs**: Standardized services know how to work together, enabling disparate applications to quickly and easily connect.
* **Reduce risk**: Fewer, reusable services provide greater control over corporate and IT governance policies, and reduce the overall compliance risk to an enterprise.
* 

[API-led connectivity](https://www.mulesoft.com/lp/whitepaper/api/api-led-connectivity)

Mulesoft's methodical way of connecting data and applications through reusable APIs to decouple between the implementation and the API

Three layers

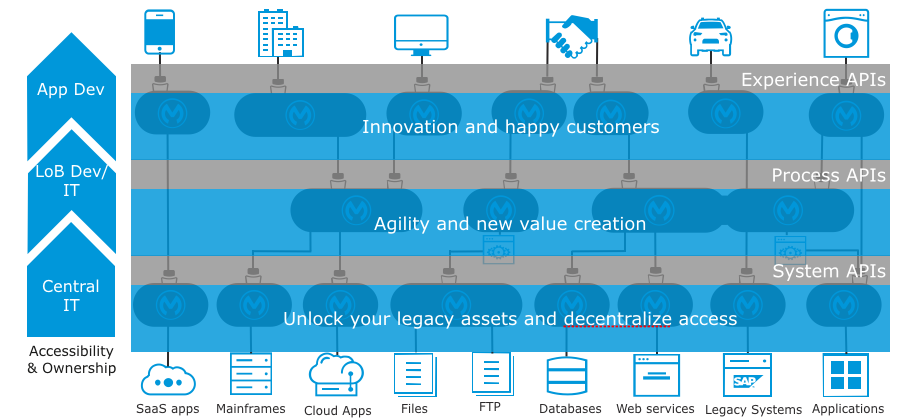
* Experience Layer
* Process Layer
* System Layer

It is a methodical way to connect data to applications through reusable and purposeful [APIs](https://www.mulesoft.com/resources/api/what-is-an-api). **API-led connectivity that goes far beyond software integration.** The system is made up of three different layers of APIs with a single purpose – to make your business composable & agile:

* Governed by IT, **system APIs** are those APIs which extract sensitive information from your systems.
* **Process APIs** are designed specifically for processes in an organisation. For example, if you need to fetch your bank balance from a smartphone or laptop, the process is the same.
* **Experience APIs** would be developed according to how the information is displayed on any particular device.

This methodology allows organisations to easily **add more and more devices and solutions into the mix, while maintaining high performance of the whole system.** Instead of taking months to do so, it can be done within a matter of days or weeks because the processes are already there. This applies to the other end of the spectrum, even after merge or acquisition, systems, processes & devices of  the organisations can be interconnected through the APIs.

With API-led Connectivity, your IT infrastructure would look more like the below, creating a composable, transparent and decentralized structure. This allows different lines of business to take more control over their systems, while system data is still governed by the IT department.



No matter how many years your organisation has been around & how old your systems are, things don’t have to take an eternity to happen anymore. You can beat competition by doing things first, by transforming a business idea through technology within days, weeks instead of months, years.

Functions

Dataweave functions have to be defined with %function in the header part

%dw 1.0

%output application/json

%function getName() {

name: payload.user.name

}

---

getName()

With the introduction of datawaeve version 2 in Mule 4, the syntax for declaring functions will change.

Variables

Dataweave variables have to be defined with %var in the header part

%dw 1.0

%output application/json

%var name = payload.user.name

---

name: name

Lets have the below XML as input to both the scripts.

<user>

<name>Jane</name>

</user>

Output of both scripts for the above XML is same.

{

"name": "Jane"

}

**Serialization** is a mechanism of converting the **state of an object into a byte stream.** **Deserialization** is the reverse process where the byte stream is used to recreate the actual **Java** object in memory.

The byte stream created is platform independent. So, the object serialized on one platform can be deserialized on a different platform.

To make a Java object serializable we implement the **java.io.Serializable** interface.  
The ObjectOutputStream class contains **writeObject()** method for serializing an Object.

public final void writeObject(Object obj)

throws IOException

The ObjectInputStream class contains **readObject()** method for deserializing an object.

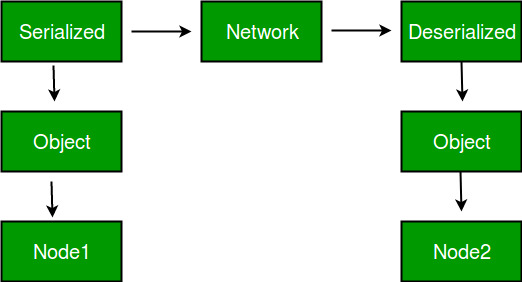
public final Object readObject()

throws IOException,

ClassNotFoundException

**Advantages of Serialization**

1. To save/persist state of an object.  
2. To travel an object across a network.



Only the objects of those classes can be serialized which are implementing **java.io.Serializable** interface.

**Points to remember**  
1. If a parent class has implemented Serializable interface then child class doesn’t need to implement it but vice-versa is not true.  
2. Only non-static data members are saved via Serialization process.  
3. Static data members and transient data members are not saved via Serialization process.So, if you don’t want to save value of a non-static data member then make it transient.  
4. Constructor of object is never called when an object is deserialized.  
5. Associated objects must be implementing Serializable interface.  
Example :

class A implements Serializable{

// B also implements Serializable

// interface.

B ob=new B();

}

What connectors have u used?

Api autodiscovery

Localhost:8081/api/hello -> mule run time return the response

After api autodiscovery

Localhost:8081/api/hello -> mule run time -> api gateway -> api policy in run time -> mule flow -> return the response

If u want to manage ur api from api manager, u can enable api autodiscovery for that particular Mulesoft application

Difference between api autodiscovery and proxy applicationS:

When u deploy application on cloudhub without api autodiscovery, then u have to create proxy application from api manager

Basically on top of main application u r creating proxy application, so basically two application running on the cloudhub, so with api autodicovery we can avoid multiple applications deployed on the cloudhub. So, by using api autodiscovery we can manage apis and by applying various policies, without creating proxy on the cloudhub

If u r using endpoint with proxy then it will create a separate application on top of main application, ie nothing but proxy application

For using api autodiscovery, we should choose basic endpoint in managing type option in api configuration

For setting api autodiscovery

1.First create an raml api specification

2.publish to exchange

3. create mule project by importing from exchange

4.in api manger, manage api by selecting same api u published, we should choose basic endpoint in managing type option in api configuration, choose check box mule 4 option and save

5. copy that apiId autodiscovery and in studio fo to global elements and click create and component configuration and choose APIautodiscovery, in that paste the id u copied and give flow name as apimain

6.in anypointplatform, goto access management and copy clientid and client secret in sandbox

And paste it in notepad in this format,

anypoint.platform.client\_id=d1506544fed84237a68d3a49d158d006

anypoint.platform.client\_secret=9f7945edFd9B46E2911927ACD53ef829

7. Either u can pass the properties directly from mule properties file or directly pass from cloudhub.

If we want to pass directly from cloudhub, then first export the project.

In studio, export the project by right clicking the project and select export and select anypoint studio project to Mule Deployable Archieve

And store zip file in a location

8.in runtime mgr, deploy app by selecting that jar file u created

Wait for deployment, once deployed if u go to api manager u can see it turned unregistered to active

Now u add consumer endpoint:

By copying app url test-appsuja.us-e2.cloudhub.io from run time manager and add it to consumer endpoint in api manager for eg: http://test-appsuja.us-e2.cloudhub.io/ and add a label for eg. Sandbox api

Now u go to post man and run http://test-appsuja.us-e2.cloudhub.io/api/test?name=sai

It works perfectly for 3 req

More than that it says authentication error, since u applied rate limiting policy with 3 req allowed

If u want to add clientID enforcement

Remove rate limiting policy in api manager

Add clientIDenforcement policy, copy snippet content in raml1.0

Add those lines in api design center, and add is: [client-id-required] in below get:

Publish to exchange

Again go to api manager and click v1, click update button next to assetversion: and choose the version u published recently

Try again with exchange

Access denied

To get access

Click request access

For api instance Choose test Sandbox API

For application, create application with test app

U can give request access directly also

If u want u can create SLA tier with silver, free access too

It will generate client id and secret copy and try to run with that

Scatter-gather:

Sends request msg to multiple target concurrently, collects all responses from all routes and aggreagates them tinto a single message

Send copies to all at once to different message processor

Failure in one route, not stop sending msg

Sets exceptionpayload accordingly for each route throws composite Routing exception

getExceptionForRouteIndex(int)- returns exception of requested route ID

scatter gather donot start with less than 2 processing route

|  |  |  |
| --- | --- | --- |
|  | REST | SOAP |
|  | Rest is an architectural style(more flexible) | SOAP is a protocol, highly specified |
|  | No official standards for RESTFul API’s | Has an official standard |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |