<https://docs.mulesoft.com/object-store/osv2-faq#can-one-mule-app-access-the-object-store-of-another-mule-app>

<https://docs.mulesoft.com/object-store/osv2-faq>

**11. What are the real time problems API led connectivity is solving? What are the problems faced by IT?**

https://blogs.mulesoft.com/dev/api-dev/what-is-api-led-connectivity/

**API led connectivity** helps to fulfill the IT delivery gap

An API-led connectivity approach to delivering IT projects ensures you are not only on time and budget with your first projects, but you have built the reusable assets that will save your company time and money, created an infrastructure which is designed for change, built in visibility, compliance and governance and, most importantly, met the needs of the business, which is long-term sustained agility.

It enables you to move fast on your first project, but then actually accelerate further from your second project onwards, due to reusable assets and a built-up organizational capability; API-led connectivity liberates resources, allowing you to innovate and to move quickly.

The problems of IT are:

* - Meet the changing market demand on time
* - resources are tightly coupled
* - limited opportunity to reuse
* - difficult to govern

**12. New IT operating Model proposed by MuleSoft? Does it focus on Consumption?**

MuleSoft encourages production consumption model.

It focus on the consumption and this is enforce by implementing C4E

**13. How is Modern API a core enabler of a new operating Model? What are the features of a Modern API?**

Modern AP is a core enabler as it focues on production consumption model. To achieve this it implement API led connectivity approach. And this governed by C4E. The requirement of Modern API are:

- discoverable and accessible assets through self service

- Design and productized for ease of consumption

- easy to manage for security, performance and scalability

**14. What are the roles LOB IT / Central IT and Developers play in API led connectivity?**

**Central IT**: unlocks the data in core system of records. It is responsible for producing system APIs and they are also responsible to governed and monitored the access of these APIs

**LOB IT:** Discover and consume data made accessible by system API. Then usinf these system API it creates process API which compose data transforming enriching, aggregating, orchestration as need.

**Developers:** Discover, self-server, resuse and consume the APIs for the experience layer

**15. What are the major outcomes you think are driven by API Led connectivity?**

The major outcome of API Led connectivity are:

* complete project on time and in budget
* everything is reusable
* Designed such way that it can be easily changed
* Connection are secured and monitored so it is agile and productive
* Still meet governance, compliance. Security, scalability

**16. What is C4E? What is the goal for C4E?**

C4E is a cross functional team that venturalizes and drive API led connectivity approach

- It ensure applications are build using existing assets as much as possible and develop new assets as needed for functionality if that does not yet exists

17. How can you achieve Speed/ Agility with application networks?

Speed and agility in achieved with the application network .

**18. What is an API? What is API Spec? Why do you need it?**

The term “**API**” stands for [Application Programming Interface](https://nordicapis.com/focus-topics/what-is-an-api/).  An API is an interface that can be used to program software that interacts with an existing application.

Example: When you use an application on your mobile phone, the application connects to the Internet and sends data to a server. The server then retrieves that data, interprets it, performs the necessary actions and sends it back to your phone. The application then interprets that data and presents you with the information you wanted in a readable way. This is what an API is - all of this happens via API

<https://www.mulesoft.com/resources/api/what-is-an-api>

https://www.mulesoft.com/resources/api-management

**An API spec** consists of a plan of how your API should look structurally - like a blueprint of a house. It’s a key part of API development because it can help you isolate design flaws or problems before you write a line of code. API provides information for how to communication with software component.

Adv:

* help you catch any glitches or inconsistencies early on
* can save you months and perhaps years of hassle as you struggle with poor design or even having to build a new API from scratch.

<https://www.mulesoft.com/resources/api/api-spec#:~:text=An%20important%20part%20of%20API,a%20blueprint%20of%20a%20house.&text=The%20idea%20behind%20a%20REST,be%20flexible%20enough%20to%20endure>.

**API documentation** is a technical content deliverable, containing instructions about how to effectively use and integrate with an API. It’s a concise reference manual containing all the information required to work with the API, with details about the functions, classes, return types, arguments and more, supported by tutorials and examples.

<https://dzone.com/articles/what-is-api-documentation-and-why-does-it-matter#:~:text=API%20documentation%20improves%20the%20developer,API%20and%20increase%20user%20awareness.&text=We're%20in%20the%20multi,glue%20of%20the%20digital%20landscape.&text=They're%20analytical%2C%20precise%2C,important%20problems%20with%20your%20API>.

**19. Difference between API Interface definition file / Web service and API proxy?**

**An API** is a set of **definitions** and protocols for building and integrating application software.

An API is an abstraction of a set of components. It encapsulates the complexity of your Mule application development, and it presents an easy and reusable way of accessing your assets.

**Web services** are the methods of communication that allows two software system to exchange data over the internet

**API proxy** is an application that control access to web services, restricting access and usage through the use of API gateway

20. What is a web service?

Web services are the methods of communication that allows two software system to exchange data over the internet

Difference between API and web services are:

* web service is an API — since it exposes an application’s data and/or functionality — but not every API is a web service
* **Web services require a network.** While APIs can be on- or offline, web services must use a network.
* **APIs are protocol agnostic**. While APIs can use any protocols or design styles, web services usually use SOAP (but sometimes REST, UDDI, and XML-RPC).
* API is an interface that allows you to build on the data and functionality of another application, while a web service is a network-based resource that fulfills a specific task

21. Difference between Soap and Rest web services?

* SOAP stands for Simple Object Access Protocol whereas REST stands for Representational State Transfer.
* SOAP is a protocol whereas REST is an architectural pattern.
* SOAP uses service interfaces to expose its functionality to client applications while REST uses Uniform Service locators to access to the components on the hardware device.
* SOAP needs more bandwidth for its usage whereas REST doesn’t need much bandwidth.
* SOAP only works with XML formats whereas REST work with plain text, XML, HTML and JSON.
* SOAP cannot make use of REST whereas REST can make use of SOAP.

<https://www.guru99.com/comparison-between-web-services.html>

Below are the main differences between SOAP and REST

|  |  |
| --- | --- |
| **SOAP** | **REST** |
| * SOAP stands for Simple Object Access Protocol | * REST stands for Representational State Transfer |
| * SOAP is a protocol. SOAP was designed with a specification. It includes a WSDL file which has the required information on what the web service does in addition to the location of the web service. | * REST is an Architectural style in which a web service can only be treated as a RESTful service if it follows the constraints of being   1. Client Server   2. Stateless   3. Cacheable   4. Layered System   5. Uniform Interface |
| * SOAP cannot make use of REST since SOAP is a protocol and REST is an architectural pattern. | * REST can make use of SOAP as the underlying protocol for web services, because in the end it is just an architectural pattern. |
| * SOAP uses service interfaces to expose its functionality to client applications. In SOAP, the WSDL file provides the client with the necessary information which can be used to understand what services the web service can offer. | * REST use Uniform Service locators to access to the components on the hardware device. For example, if there is an object which represents the data of an employee hosted on a URL as http://demo.guru99 , the below are some of URI that can exist to access them   http://demo.guru99.com/Employee  http://demo.guru99.com/Employee/1 |
| * SOAP requires more bandwidth for its usage. Since SOAP Messages contain a lot of information inside of it, the amount of data transfer using SOAP is generally a lot.   <?xml version="1.0"?>  <SOAP-ENV:Envelope  xmlns:SOAP-ENV  ="http://www.w3.org/2001/12/soap-envelope"  SOAP-ENV:encodingStyle  =" http://www.w3.org/2001/12/soap-encoding">  <soap:Body>  <Demo.guru99WebService  xmlns="http://tempuri.org/">  <EmployeeID>int</EmployeeID>  </Demo.guru99WebService>  </soap:Body>  </SOAP-ENV:Envelope> | * REST does not need much bandwidth when requests are sent to the server. REST messages mostly just consist of JSON messages. Below is an example of a JSON message passed to a web server. You can see that the size of the message is comparatively smaller to SOAP.   {"city":"Mumbai","state":"Maharastra"} |
| * SOAP can only work with XML format. As seen from SOAP messages, all data passed is in XML format. | * REST permits different data format such as Plain text, HTML, XML, JSON, etc. But the most preferred format for transferring data is JSON. |

**22. What are the different ways you can secure APIs?**

There are different ways to secure the APIs

1. Custom code using existing frameworks

The upside of this approach is that developers could write custom functions that fit the security requirements 100%. However, the major challenge with the custom code approach is that these libraries will have to be embedded into all deployed Mule application that need them (probably all of them). So once security requirements or endpoints change over time, the security libraries will have to be maintained and updated

1. External third art or cloud services

Some customers are heavily invested in cloud platforms such as [AWS](https://blogs.mulesoft.com/dev/cloudhub-developer/anypoint-platform-and-serverless-computing-aws-lambda/), Google Cloud, and Microsoft Azure. Many are considering using capabilities from these cloud platforms to secure their Mule endpoints using a “best-of-breed” approach.

While this approach gives companies flexibility to pick the best tools for the job, it also has a major drawback: a lot of thinking has to go into end-to-end security since the actual Mule endpoints are still exposed on CloudHub

1. Anypoint Manager
2. The third option is to use an out-of-the-box component within MuleSoft’s Anypoint Platform — the [API Manager](https://www.mulesoft.com/resources/api/what-is-api-management).
3. It basically creates an [API Proxy](https://blogs.mulesoft.com/dev/api-dev/proxying-with-api-manager/) for each backend API running on Anypoint Platform and thereby secures requests coming into the platform again the API.
4. All of the proxies run on the out-of-the-box API Gateway that services as the point of enforcement for API policies.
5. Developers can now quickly attach API policies to the endpoints and secure them efficiently without changing the underlying code being dependent on external solutions.

* Eliminate vulnerabilities at the network edge based on observed attack patterns at the API gateway
* Enforce security by configuring mandatory policies
* Hide sensitive data with format-preserving tokenization to reduce compliance scope

Write down.

1. **What is control plane and runtime plane?**

The Anypoint Platform control plane provides a set of cloud services that simplify the design, reuse, and management of integrations and APIs.

The Anypoint Platform runtime plane is where applications are deployed, and also where the Mule runtime engine and other application-related services, such as Anypoint Connectors run. The runtime engine includes Anypoint Security edge policies and tokenization, MQ, Object Store, and Connectors.

1. **Key capabilities of Runtime manger, exchange, API manager?**

**Anypoint runtime manager** is the interface to anypoint platform that provides a unified view of your applications, servers, and APIs.

It is used to deploy, manage, monitor our Mule applications in a sandbox, staging, or production environment from one central location, whether your apps are deployed in the cloud or on-premises.

**Exchange** is a global repository that allows to share, discover and incorporate assets and resources in our application. It helps to create API developer portal, view and test APIs, simulate data to APIs ( Mocking service), create assests and use API Notebooks to describe and test API functions.

**API Manager** is a component of Anypoint Platform for managing APIs. API Manager leverages the runtime capabilities of API Gateway, which enforces policies, collects and tracks analytics data, manages proxies, provides industry standard encryption and authentications, and manages client applications.

1. How do API's in runtime plane communicate with control plane?

To use Mule runtime engine within the EU Control Plane, you must redefine the configuration properties that enable Mule runtime engine to point to the EU Control Plane. You must also provide the client ID and client secret of the environment belonging to your organization.

By default, Mule runtime engine is configured to point to the US control plane. If you do not override these properties, the API gateway capability of Mule runtime engine is disabled in the EU Control Plane.

<https://docs.mulesoft.com/eu-control-plane/runtime-configure-eu>

# Active MQ Terminology:

https://www.youtube.com/watch?v=4AgWmtez9I4

<https://docs.mulesoft.com/mq/mq-understanding>

Anypoint MQ is an enterprise, multi-tenant, cloud messaging service that performs advanced asynchronous messaging scenarios between applications. Anypoint MQ is fully integrated with Anypoint Platform, offering role-based access control, client app management, and connectors.

The Anypoint MQ messaging broker enables applications to communicate by publishing a Mule message to a queue so that another app can consume the message.

Multi-tenant: you can create a queue and to acess the queue you need to access via client app.

Cloud messaging: hosted in the cloud. So don’t need to worry about the scalability feature. On premise application need to use API to access it

Performance async messaging scenario: bank picks all the transaction that occur in day at the end of the day. This can be done using queue

Fully integrated platform:

Roll based access control: conntect different clients to the available queue

Message queues provide temporary storage that enables applications to communicate in a scalable and reliable way. Publishers can publish messages into queues, and subscribers can receive messages from queues. You can create a dead-letter queue to store undeliverable messages.

Anypoint MQ includes these features:

* **Queues and Message Exchanges**
* **Management console**
* **Anypoint MQ connector**
* **Usage information**
* **REST API**
* **Environments and role-based access control**
* **Client management**
* **Large payloads**
* **Disaster recovery**
* **Delay queues**
* **Delayed delivery for messages**
* **Message groups**
* **Encrypted queues**

Queue

Message queues provide temporary storage that enables applications to communicate in a scalable and reliable way. Publishers can publish messages into queues, and subscribers can receive messages from queues. You can create a dead-letter queue to store undeliverable messages.

Dead Queue

Anypoint MQ enables you to ensure that messages that aren’t delivered are sent to a queue known as the dead-letter queue (DLQ). You can then analyze the messages sent to the DLQ to determine why those messages were not delivered.

A DLQ is essentially the same as any other queue except that it receives only undelivered messages. You can specify the time-to-live (TTL) value, encryption, and delivery delay when you create the queue.

You must have at least two queues for one of them to be assigned as a DLQ. Both queues must be:

* The same type (standard or FIFO)
* In the same geographical [region](https://docs.mulesoft.com/mq/mq-faq#regions)
* In the same environment and owned by the same Anypoint Platform account.

<https://docs.mulesoft.com/mq/mq-queues#dead-letter-queues>

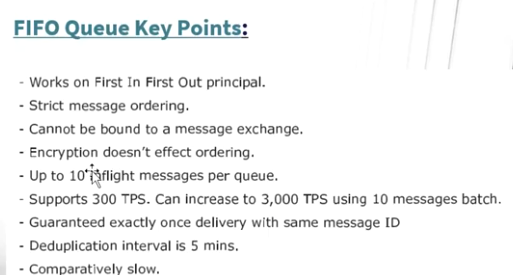
Queues provide these benefits:

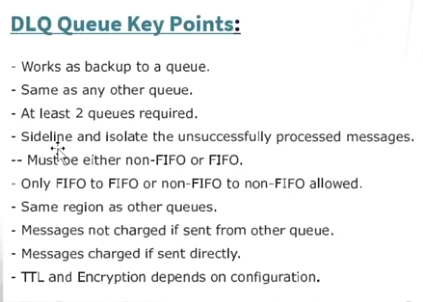
* **Horizontal scalability**
* **Reliability**
* **Decoupling: diff application can communicate**
* **Aggregation: different application pushing to same queue**
* **Message TTL**
* **Default Acknowledgement Timeout**
* **Unlimited messages in a queue**
* **Long polling**
* **Billing and charging**

Types: Standardized Queue, FIFO Queue and exchange

FIFO: for application where the order of the messages needs to be strictly preserved and enforced. Also used for de-duplication of message

Standardized: used if the ordering of the message is not necessary



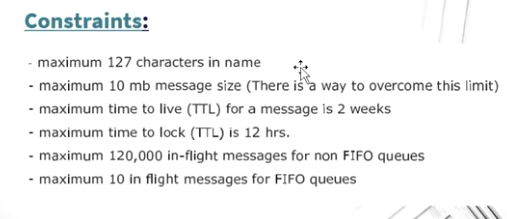


Queue Configuration:

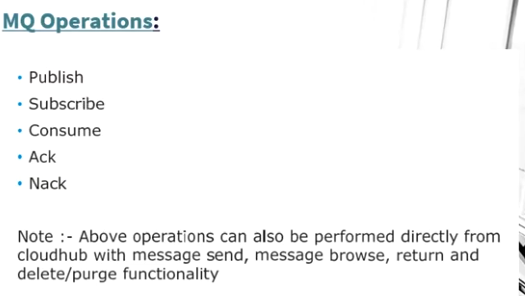
* **ID: alpha numeric, periods and hyphen**
* **Message TTL: message can stay in the queue for the mentioned time**
* **ACK timeout: ACK timeout happens**
* **Default delivery delay: define delay before sending message**
* **Encryption:**
* **Dead letter queue: you must define the dead letter queue in the same region where the queue is define**
* **Attempts before re-route:**
* **In queue: when you send the message to the broker the message is signed to the broker**
* **In flight: when the consumer consumes the message from the queue or ACK time expires**

**Ack mode: way to know the broker whether the messages is consumes or not ..**

* 1. **Auto: if sunscriber received msg without any exception ack is send to the broker. If exception occur the server get nack and msg is not deleted from the server**
  2. **Immediate: as soon as the message is received by the flow it sends the ack to the broker. Do not consider the message processing**
  3. **Manual: ackToken is unique.**



**Default subscriber mode is prefetch. Prefetch mode takes few of the message from the broker and store is in buffer. In scenario where high through put is required this mode is used.**



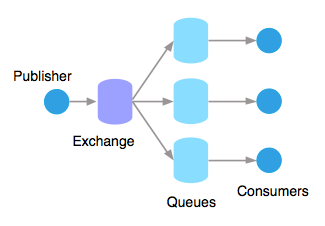
**Polling mode the size is 10 message. And you can define the time to fetch.**

**Clients**

In Anypoint MQ, each consumer must be registered as a client app with a name, ID, and a secret. The client app uses the ID and secret to authenticate and get access to queues and message exchanges within a single environment. Client apps cannot be used across multiple environments or business groups.

Message Exchange

With message exchanges you can distribute a single message to multiple consumers



For example, if a publisher publishes changes in the weather, and you have multiple apps that want to receive that data, you can use a message exchange to publish a single message to multiple queues for each app. A binding is a relationship between the queue and message exchange, which tells the message exchange where to send messages. A queue can be bound to multiple message exchanges.

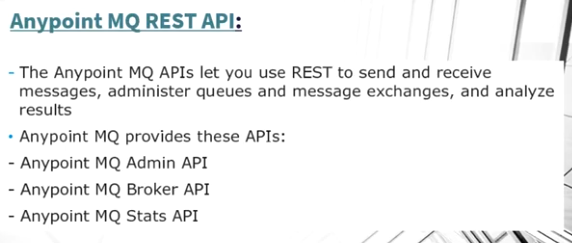
Example: In this example, an app running Salesforce customer relationship manager (CRM) wants to communicate with two applications that handle different databases. Using Anypoint MQ, the Salesforce app sends a pub/sub message to a message exchange, and the two applications receive the message from the message exchange and convey the message to their respective databases.

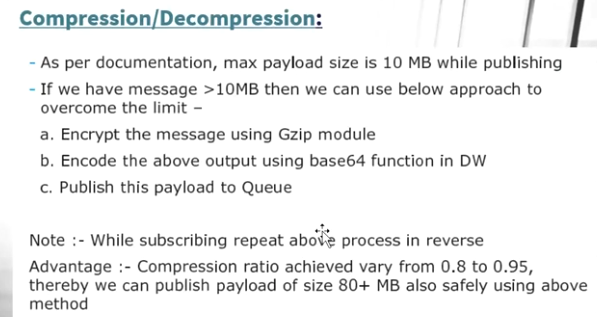
Difference between VM and Anypoint queue

VM in in memory queue. As soon as the application is down the message is lost

Anypoint used for relaiblity

To secure the message you can use encrpt option in the AM queue. But you can add extra level of security by encrypt a message





Client-id enforcement:

It allows only the authorized users to access the API.

# Mule 4 - Client ID Enforcement

<https://dzone.com/articles/mule-4-client-id-enforcement>

The Client Id Enforcement policy is used to restrict access to a protected resource by allowing requests only from registered client applications. The policy ensures that each request, which contains valid client credentials is able to access protected resources.

The client application has to be registered on the AnyPoint platform to generate client credentials (client\_Id  and  client\_secret). After client application registration, all subsequent requests have to pass client\_id and client\_secret, as part of the request while invoking an API.

There are some policies which internally enforce client application credentials. Those are:

* Rate Limiting - SLA-Based Policies.
* OAuth 2.0 Access Token Enforcement.
* [JWT Validation](https://dzone.com/articles/state-of-api-security).

<https://docs.mulesoft.com/api-manager/2.x/client-id-based-policies>

When a client application is registered in Anypoint Platform, a pair of credentials consisting of a client ID and client secret is generated. When the client application requests access to an API, a contract is created between the application and that API. An API that is protected with a Client ID Enforcement policy is accessible only to applications that have an approved contract.

When you apply a Client ID Enforcement policy, access to your API is tracked by reporting the client ID along with the analytics events. You can configure Mule runtime engine (Mule) to encrypt all sensitive information that pertains to policies, contracts, and initialization data.

# Splitter Pattern

Splitters are not longer available in Mule 4. To process a collection of elements sequentially, use the [For Each component](https://docs.mulesoft.com/mule-runtime/4.3/for-each-scope-concept).

foreach element is a scope, meaning that the components used to process the iterated elements must be defined within foreach

Also, the foreach element does not return a collection with the results of all the processed values. Instead, it returns the same collection received as input. So, the behavior of previous examples is not the same.

It does not matter if a Mule 3 aggregator is present. To store the results of the processed values, you need to use a Mule 4 aggregator.

Besides For each Splitter pattern is used in Batch and Parallel For each

# For Each Scope

# <https://docs.mulesoft.com/mule-runtime/4.3/for-each-scope-concept>

The For Each scope splits a payload into elements and processes them one by one through the components that you place in the scope. It is similar to a for-each/for loop code block in most programming languages and can process any collection, including lists and arrays. The collection can be any supported content type, such as application/json, application/java, or application/xml.

General considerations about the For Each scope:

* By default, For Each tries to split the payload. If the payload is a simple Java collection, the For Each scope can split it without any configuration. The payload inside the For Each scope is each of the split elements. Attributes within the original message are ignored because they are related to the entire message.
* For Each does not modify the current payload. The output payload is the same as the input.
* For non-Java collections, such as XML or JSON, use a DateWeave expression to split data. Use the **Collection** field for this purpose.

Discussion:

**Why do we use object store instead of normal variable:**

We use object store over normal variable in following conditions:

* When configuring an Idempotent Message Validator, to store the unique IDs in the specified object store.
* When configuring a Redelivery policy in a listener (HTTP, File, JMS, VM, Socket), to store the redelivery count in the specified object store.
* When configuring a Cache scope that uses a custom caching strategy, to store the cached data in the specified object store.
* When a custom component uses an object store to persist information.
* When storing or retrieving information from a Mule flow, by using Anypoint Connector for Object Store (Object Store Connector).

**When to use messaging Queue**

MQ are used for asynchronous messaging for batch jobs and decouple applications

In modern cloud architecture, applications are decoupled into smaller, independent building blocks that are easier to develop, deploy and maintain. [Message queues](https://aws.amazon.com/message-queue/) provide communication and coordination for these distributed applications.

Message queues can significantly simplify coding of decoupled applications, while improving performance, reliability and scalability. You can also combine message queues with [Pub/Sub messaging](https://aws.amazon.com/pub-sub-messaging/) in a fanout design pattern.

**Data synchronization**

Data synchronization is the ongoing process of synchronizing data between two or more devices and updating changes automatically between them to maintain consistency within systems.

* Data synchronization ensures [accurate, secure, compliant data](https://www.talend.com/resources/5-pillars-gdpr-compliance/) and successful team and customer experiences
* Data must always be consistent throughout the data record. If data is modified in any way, changes must upgrade through every system in real-time to avoid mistakes, prevent privacy breaches, and ensure that the most up-to-date data is the only information available.

**Watermarking**

The watermarking technique stores and retrieves the point at which a periodic synchronization resumes the next time it executes. Watermarking is a typically used alongside ObjectStore.

**Poll Scope:**

The poll scope feature allows the developer to poll a particular source endpoint based on a timed interval

Repeatable Stream

Mule 4 introduces repeatable streams as its default framework for handling streams. Repeatable streams enable you to:

* Read a stream more than once.
* Have concurrent access to the stream.

<https://docs.mulesoft.com/mule-runtime/4.3/streaming-about>

There are two streaming strategies used in mule 4:

1. File Stored Repeatable Stream

2. In memory repeatable stream