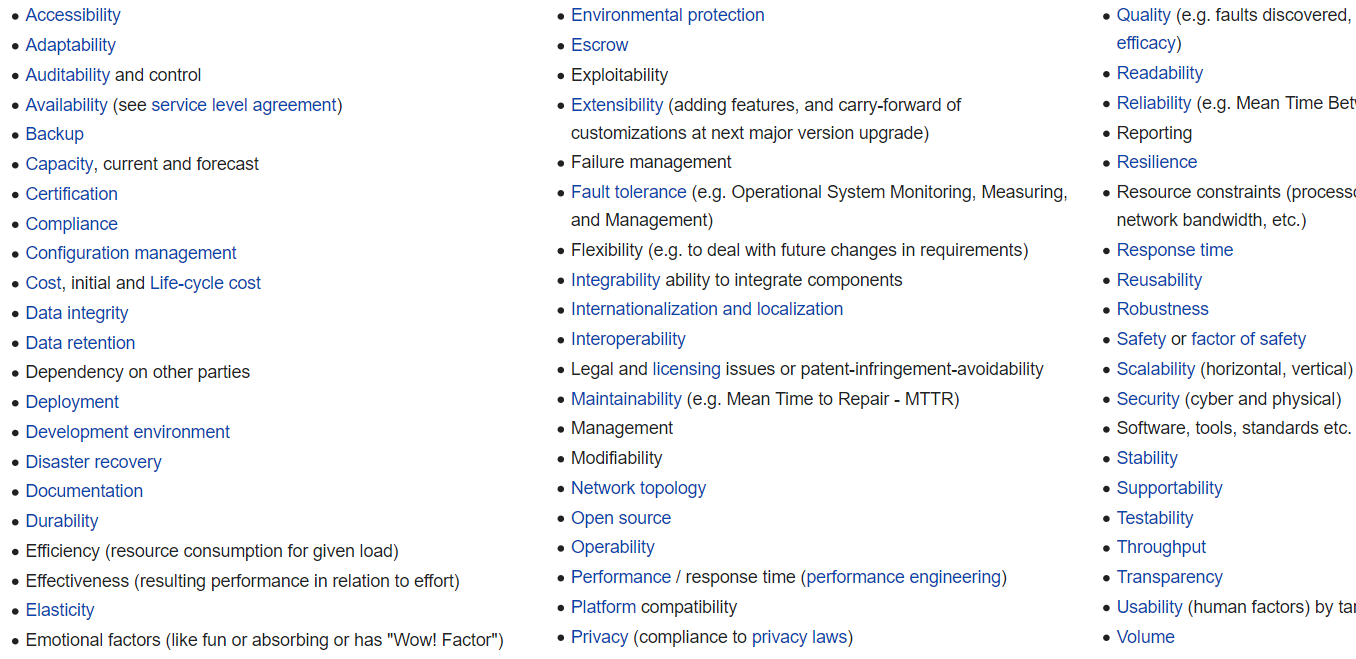
# API Manager

* It is used to implement non functional requirements
* Theses Non-functional requirements are:



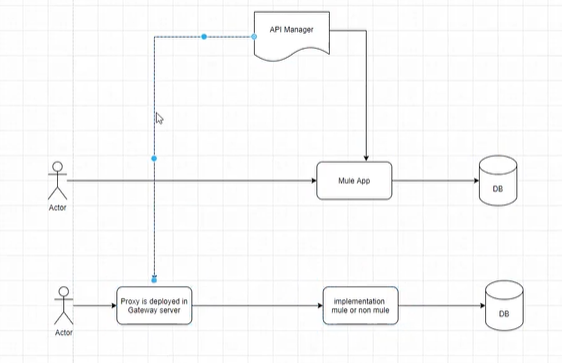
# Manage API

* It is used to enforce policies
* There are three ways to enforce policies

1. Basic Endpoints
2. Endpoint with proxy
3. Service Mesh

## Basic Endpoints

* In this we configure policies with implementation
* How it works
  + API manager is in control plane
  + Then configure policies with control plane
  + When mule runtime starts, the polices that are configured in API Manager gets downloaded to runtime where mule application is running



## Endpoint with proxy

* In the we create a separate proxy application
* Here we deploy the proxy application in gateway server [ i.e. Mule Runtime]
* The policies are configured to control plane
* When the gateway server starts it downloads that policies into the gateway server
* If the request is validated by the implemented policies, then the request is forwarded to the implementation. In this case the implementation can be mule app or non- mule app

Use-cases

1. For **non mule applicat**ion

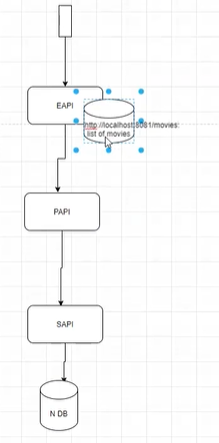
* Used for implementation in non-mule application example .NET, spring boot
* And you want to secure, manage those APIs with Mule

1. **Security Requirement**

* Suppose mule application are running on different n/w **demilitarized zone**
* If you have any security requirements then use it
* Here all request must pass through proxy so it is more secure

In **computer** security, a **DMZ** or **demilitarized zone** (sometimes referred to as a perimeter network or screened subnet) is a physical or logical subnetwork that contains and exposes an organization's external-facing services to an untrusted, usually larger, network such as the Internet

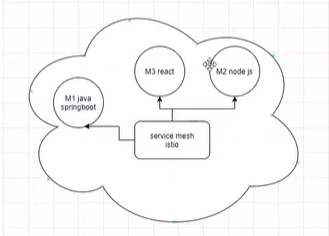
1. **Caching**

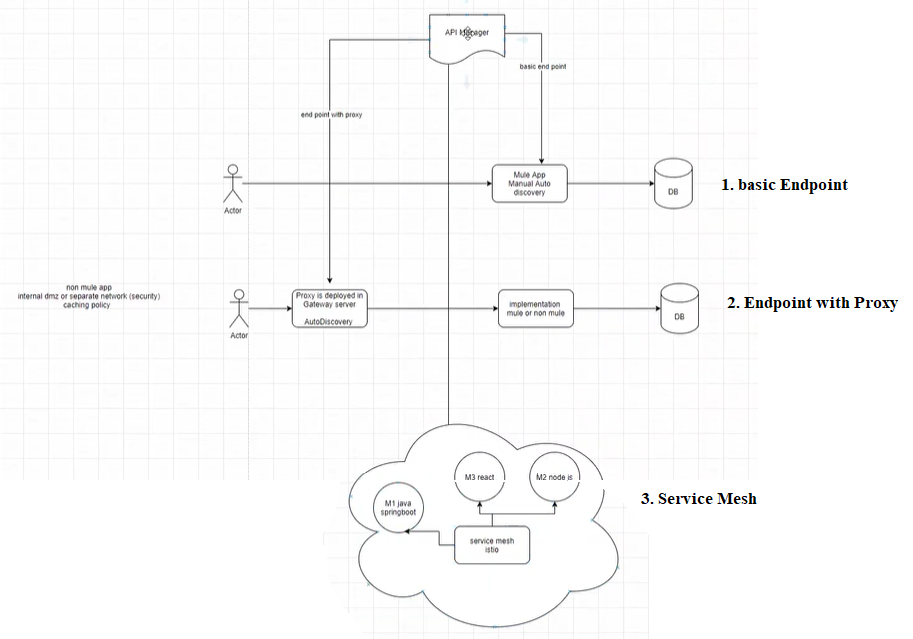
* To enable caching we must use caching policy
* Caching is implemented for non-frequently changing data
* Caching policy always uses key-value pair
* It is persistent and distributed
* Pros:
  + It helps to reduce n/w latency
* Cons:
  + It will cache complete response. So user donot have a choice to cache the partial / select specific response
* Example : Access a movie list
  + If we donot use caching certain amount of time is spent on sending request. Where request flows through
    - EAPI to PAPI
    - PAPI to SAPI
    - SAPI to DB
* Now if we implement caching it will help to reduce the n/w traffic
* It uses key- value pair to store responses
* Here when the request is made for the first time The EAPI stores the response in its cache as key- value
* Example: movieList: list of movies
* Now if any one makes the same request next time, instead of sending the request to DB, data are fetched from the cache
* Caching can be done in multiple ways
  + - **Cache scope**: use **anypoint studio cache** **scope**
    - **Cache policy: use the anypoint cache policy**
      * Here instead of using key-value
      * It stores the url as key
      * Example: <http://localhost/movies> : list of movies

### Auto Discovery

* Autodiscovery is managing an instance within API manager to a corresponding Mule Runtime
* It is the process of associating Mule Runtime with API manager
* Two ways to implement auto discovery
  + **Manually** in basic endpoint: It is used to implement policy within the implementation
  + **Automatic** in end point with proxy
* Mule Runtime can be anywhere:
  + Cloudhub
  + Standalone
  + Anypoint studio locally
* **GateKeeper**
* If we don not configure the autodiscovery, **GateKeeper** will prevent the access of the API
  + So inorder to access the API we need to make sure that the auto discovery is configured

## Service Mesh

* Istio based service mesh
* **Istio** is an open source **service** mesh platform that provides a way to control how microservices share data with one another. ... Together, these proxies form a mesh network that intercepts network communication between microservices. The control plane manages and configures proxies to route traffic
* The Anypoint service mesh is implemented in customers who have microservices running example of such microservices can be application developed using spring boot, node js based, react
* **Example: If there are 100’s of microservices in the Kubernetes cluster then how do we perform govrnance and security in these micro services**
* Service Mesh is an architectural style to implement governance and security on all the microservices in Kubernetes cluster
* Anypoint Service mesh allows to apply different policies in single place. It is easy to manage policy and governance
* Anypoint Service mesh : Using an API manager it allows to configure policies to Istio using secure mesh capability
  + It connects to non-mule microservices which is working in Kubernetes
  + It has a limited list of API policy compared to API manager
  + It requires Istio to be running in the Kubernetes clustered microservice

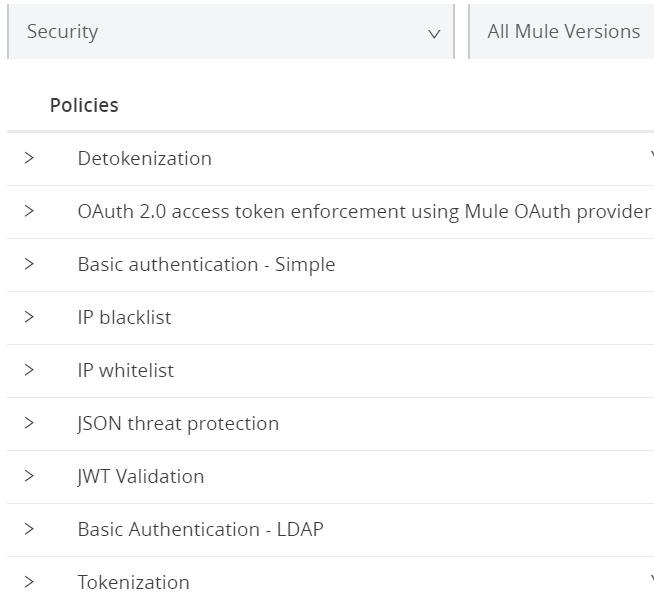
****

# Policies

Policies can be categorized into 6 categories

* Security
* QoS
* Transformation
* Compliance
* Troubleshoot

## Security



1. JWT thread protection and XML threat protection

* It validates the payload, checks the size of the payload, check the nested elements in the payload for XML, validates the structure of the payload

1. IP Whitelist: process only specific IP or range of IPs
2. IP Blacklist: blocks specific IP or range of IPs
3. Basic authentication- Simple
4. Basic authentication- LDAP
5. JWT Validation and Oauth 2. Access token

* Oauth 2. Access token and OpeIDConnect are used to configure Oauth 2 based policies
* Oauth 2. Access token and OpeIDConnect are used to protect the APIs with the oauth 2 mechanism
* Working of Oauth 2 policy:
  + First authenticate with identity provider
  + That identity provider will give a token
  + Now when you send a request using the token
  + Mule runtime will validate the token, if the token is valid mule runtime will allow to access to the resource
* In Oauth 2. Access token enforcement using Mule Oath provider, Mule Runtime can act as a oauth provider
* <https://dzone.com/articles/implementing-mulesoft-as-oauth-provider-for-securi>

Oauth Policy

* It can be done in two ways

1. Mule Oath provider

* Mule Runtime acts as a identity provider

1. External Identity provider

* External services are used for as identity provider
* Example: Okta, pingfederate
* To access the external Identity provider
  + First you need to configure a client provider in the access management
  + Multiple client providers can be configured in the anypoint platform
  + ID of client provider can be user/ employees name
  + Example: user name or user id are stored in the identity management system
* If an organization has their own identity management, then you can integrate you anypoint platform with such system

**Oauth 2.0** is a protocol. It has a handshake process and generated token. It ensure that only the authorized users are able to access the resources. So first it authenticates the user then authorizes them. It uses a JWT token internally

**JWT Policy**:

* It is used to validate the token.
* token is created using header+payload+signature
* JWT policy cnfigureation
  + Authorization header
  + Algorithm used: RSA. HMAC
  + Key length
  + Key origin: text based validation or public keystore validation
  + url: add the url of keystore
  + validate claim: validates the claim propagates the claim

**Identity provider**

* Why?
* You can configure your Identity provider the identity management system so that users of the organization can login to anypoint.mulesoft.com
* identity management system **must** **support OpenID connect or SAML 2.0 protocol**
* used for identity management. Where users start to use anypoint platform

You can invite each users separately

Anypoint has its internal identity management, which will store id

This is not feasible for large organization.

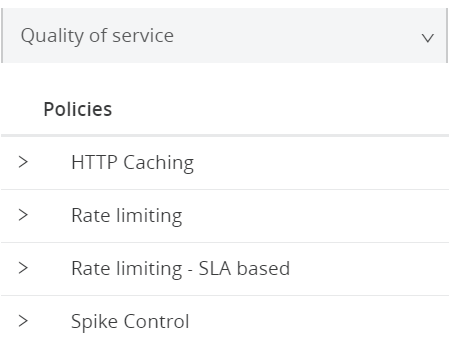
**Client provider**

* it is used for protecting your APIs with Oauth 2 policy
* configure the client provider to use external Identity provider

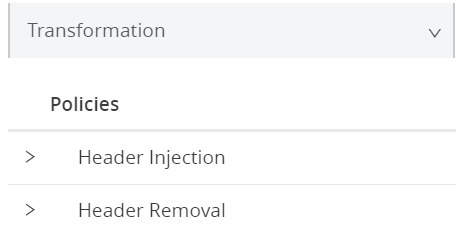
Create Oauth 2 Policy

* create client provider
* associate the client provider with an environment
* Then goto api manager, select the api instance, go to policy and configure a policy : OpenID connect policy
* Then go to exchange, open the API, request the access. First create the application
* Get the client Id and client secret
* It need to have client management or use mule oauth

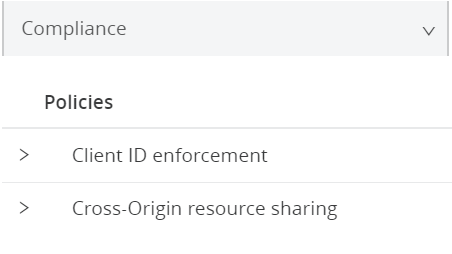
## QoS



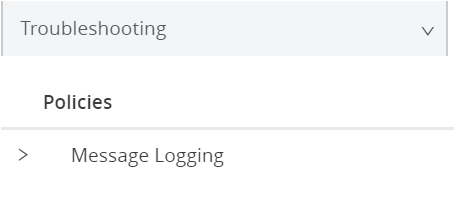
## Transformation



## Compliance



## Troubleshoot



Automated Policies

* You can have multiple instance of the same application
* To apply a policy to all the instances

API group

* Suppose one client uses 10 API
* When a client makes a request you can group all the apis so this helps in easy management
* You can publish the API in exchanges
* Helpful to reduce administrative activities