# Object Store

* There are couple of components that make the concept of object store
  + Mule OS
  + OS v1
  + OS v2
* An Object Store Connector
* OS is useful for **storing the state of the application**
* Stores the data in **key-value pair**
* Use the key to get the respective data

## OS connector operations:

The object store connector can perform various operations :

1. Contains
2. Clear
3. Remove
4. Retrieve
5. Retrieve all keys
6. Retrieve All
7. Store.

**Why use OS?**

* Storing synchronization information, such as watermarks.
* Storing temporal information such as access tokens.
* Storing user information.
* It is used in various filters, routers, and other message processors that need to store state between messages
* The Cache module uses an Object Store to maintain all of the cached data.
* The OAuth module (and every OAuth enabled connector) uses Object Stores to store the access and refresh tokens.

## OS use case

### Caching

In Mule 4 caching can be achieved in mule using cache scope and/or object-store. Cache scope internally uses Object Store to store the data. Here the previous request is stored as key-value pair

* It reduce the processing load on the Mule instance and increase the speed of message processing within a flow.
* Reduce latency
* It have option to use private Object Store to store the data. It restrict the access to data so it prevent manipulation

### Watermark

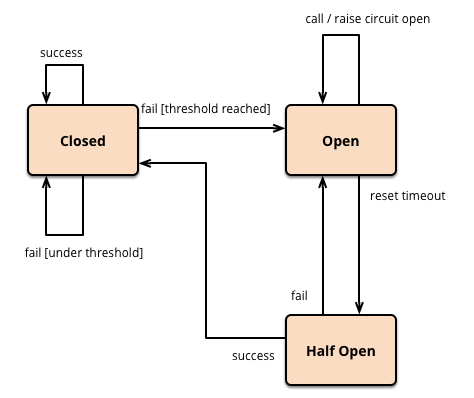
Watermarking is typically used to perform data synchronization, for example, when polling a legacy resource to retrieve new data and to sync it with another destination endpoint. The watermarking technique stores and retrieves the point at which a periodic synchronization resumes the next time it executes. Another example: Your application may need to keep track of the last synchronized record, so that the next time the application synchronizes, it can resume where it left off Watermarking is a typically used alongside ObjectStore.

Watermarking can be done automatically in On table row, New or Updated File. Additionally we can manually perform watermarking by customization using store and retrive

### Token Storage:

It is used for storing temporal information such as access tokens. Example OAuth enabled connectors can store tokens inside an Object Store with a few lines of configuration code.

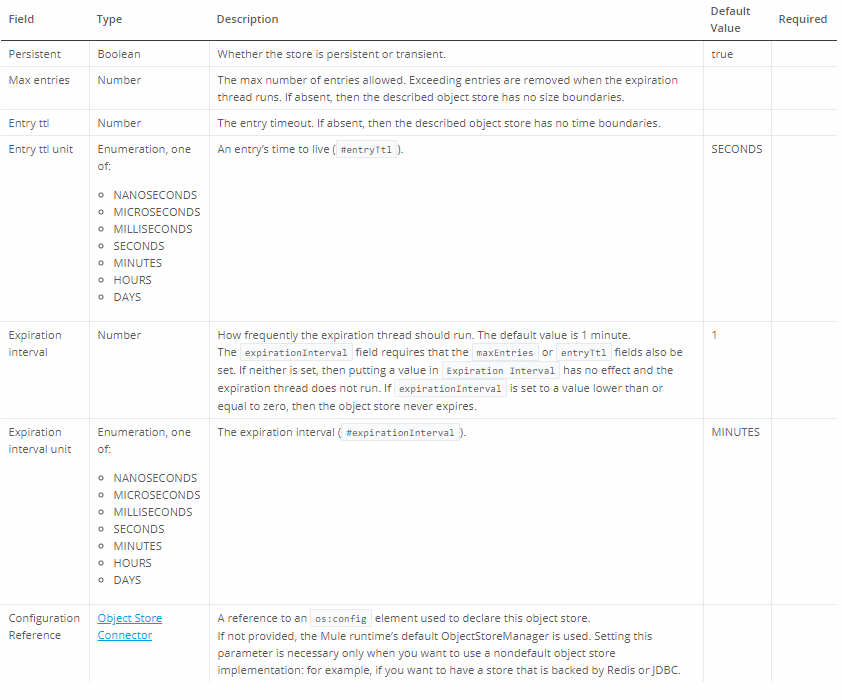
### Circuit Breaker:

Circuit Breaker monitors API calls. When everything is working as expected, it is in the state **closed**. When the number of fails, like timeout, reaches a specified threshold, Circuit Breaker will stop processing further requests. We call it the open state. As a result, API clients will receive instant information that something went wrong without waiting for the timeout to come.

The Circuit is opened for a specified period of time. After timeout occurs, the circuit breaker goes into the **half-opened** state. Next, the API call will hit the external system/API. After that, the circuit will decide whether to close or open itself.

In this object store is used to increase the counter incase of failures

Configuration



Two ways to configure

* + - **Default:** 
      * for each application it is created as default.
      * app *without any configuration*
    - **Custom- used to create multiple instances**
      * must specify an objectStore attribute
      * configured to behave differently

## **Two ways to configure Object Stores:**

|  |  |
| --- | --- |
| **In-memory store** | **Persistent store** |
| objects in local Mule runtime memory | Data stored in HDD or external storage |
| Objects are lost on the shutdown of the Mule runtime. | Data are retained |
| Fast performance | Provide high availablity |

# OS v2

* It is a service which is only used in the cloud hub based application.
* It is implemented used AWS component called dynamo DB
* Object Store v2 lets CloudHub applications store data and states across batch processes, Mule components and applications, from within an application or by using the Object Store REST API.
* Features
* Supports both static and rolling TTL (time to live) for data persistence.
* Allows for an unlimited number of entries. There is no limit on the total size of v2 object-stores.
* Stores values up to 10
* Is available in all supported regions and availability zones within each region.
* Is co-located in the same region as your workers. For example, workers hosted in Singapore would use Object Store v2 hosted in Singapore.
* Provides a Mule connector and REST interface for access by external applications.
* Provides end-to-end secure TLS-transport.
* Encrypts persistent storage to FIPS 140-2 compliant standards.
* **The only way on-premises Mule applications can access Object Store v2 is via the Object Store REST API**. You can configure a Mule app to use the Object Store REST API to store and retrieve values from an object store in another Mule app.

V1V1

* API servers and data in 1 region
* Database encryption
* Available for all Cloud Hub users
* 10,000 keys per app
* 1 MB max size of value
* 1 GB total storage
* \_defaultUserObjectStore

V2

* API servers and data are co-located across all Cloud Hub region
* FIPS 140-2 row level encryption
* Unlimited keys per app
* 10 MB max size of value
* Unlimited total storage
* 30 day TTL
* Free to use under 10 TPS

## Circuit Breaker:

* + It prevents you process from getting over loaded in a situation where you api uses backend service that the service gets down
  + Suppose that backend service is down
  + Assume that the clients are continuously invoking the APIs while the server is down
  + It will bombard with large number of transaction and bring down the API
  + So to overcome this you might want to close the traffic if certain criteria is made
  + There are 10 errors in period of 60 then I want the block the traffic for next 630 sec
  + The circuit will have three phase: open, close and half open state

**Private Object Store**

You can define a private Object Store that is not defined as a global element and does not have a referable name. Usecases for private object store:

* For cases where shared state is a security risk, you should use a private Object Store.
* For cases where you do not want anyone to manipulate the store from the connector level. For example, you want to avoid the chance that someone changes the configuration of a Clear operation so that it deletes all your authorization data.

Dynamo DB

* + - It is based on key value pair storage
    - Uses no SQL and is serverless
    - Datatype can vary for each item level
    - Adv
      * Fast n Flexible
      * Low latency
      * High durability

Things to do

<https://javastreets.com/blog/2017/9/mule-batch-processing-part2-munit-testing.html>

https://www.youtube.com/watch?v=EJ3r5uEPhKU

https://www.youtube.com/watch?v=r-jjcHPEP34