**For each scope:**

\* Splits payload collection and processes individual elements

- Collection can be any supported data type including application/json,

application/xml, application/java.

\* Returns the original payload regardless any modifications on scope.

\* Stops processing and invokes an error handler if any element throws an

exception.

\* We can examine payload before, during and after the scope.

\* For Each does not modify the current payload. The output payload is the same

as the input.

\* The For each scope stores each item of the collection in payload during each

iteration.

\* You can also split an array into batches to enable quicker processing. Each batch

is treated as a separate Mule message. For example, if a collection has 200

elements and you set Batch Size to 50, the For each scope iteratively

processes 4 batches of 50 elements, each as a separate Mule message.

**Batch job scope:**

\* Provides the ability to splits the large messages into records that are processed asynchronously in a batch job.

\* Created especially for data sets that are:

- Splits the large or streamed messages into records.

- Performs action on each individual record.

- Reports on results

\* Handling quantities of incoming data from an API into legacy system.

**How does it work? :**

A batch job contains one or more batch steps that act upon records as they move through batch job.

**Batch job contains 3 different phases:**

**Load and dispatch (implicit):**

- Splits payload into collection of records.

- Creates persistent queue and store in it.

**Process required:**

- Asynchronously process the records.

- Contains one or more batch steps.

**On complete:**

- Reports summary of records processed.

- Provides insight that which record failed so you can address it to rectify.

**Filtering:**

\* Using filtering to specify when a batch step is executed.

\* Batch job has two attributes to filter records.

- Accept policy

- An accept expression

**Examples:**

\* In one step check and see if the record exists in some data store and then In

next only upload it to that data store if it doesn’t exist.

\* Prevent a step from processing any records which failed in preceding step.

**Aggregating:**

\* To accumulate records, use batch aggregator scope inside aggregator section of

a batch step.

**Aggregating and filtering in batch step:**

\* use a batch job to synchronize records in Salesforce.

\* use batch aggregator scope to commit records in batches.