

Batch Processing in mule

Batch Processing

Mule possesses the ability to process messages in batches. Within an application, you can initiate a batch job which is a block of code that splits messages into individual records, performs actions upon each record, then reports on the results and potentially pushes the processed output to other systems or queues.

Batch processing is particularly useful when working with the following scenarios:

- Integrating data sets, small or large, streaming or not, to parallel process records.
- Synchronizing data sets between business applications, such as syncing contacts between NetSuite and Salesforce and causing "near real-time" data integration.
- Extracting, transforming and loading (ETL) information into a target system, such as uploading data from a flat file (CSV) to Hadoop.
- Handling large quantities of incoming data from an API into a legacy system.

Batch Job:

Batch job is a top-level element in Mule which exists outside all Mule flows. Batch jobs split large messages into records which processes asynchronously in a batch job.

A batch job contains one or more batch steps which, in turn, contain any number of message processors that act upon records as they move through the batch job. During batch processing, you can use record variables and MEL expressions to enrich, route or otherwise act upon records.

Batch Job Instance:

whenever a Mule flow executes a batch job. Mule creates the batch job instance in the Load and Dispatch phase. Every batch job instance is identified internally using a unique String known as batch job instance id.

Batch Processing Phases:

Phase	Configuration
Input	optional
Load and Dispatch	implicit, not exposed in a Mule application
Process	required
On Complete	optional

- Input:

The first phase, Input, is an optional part of the batch job configuration and is designed to Triggering Batch Jobs via an inbound connector, and/or accommodate any transformations or adjustments to a message payload before Mule begins processing it as a batch.

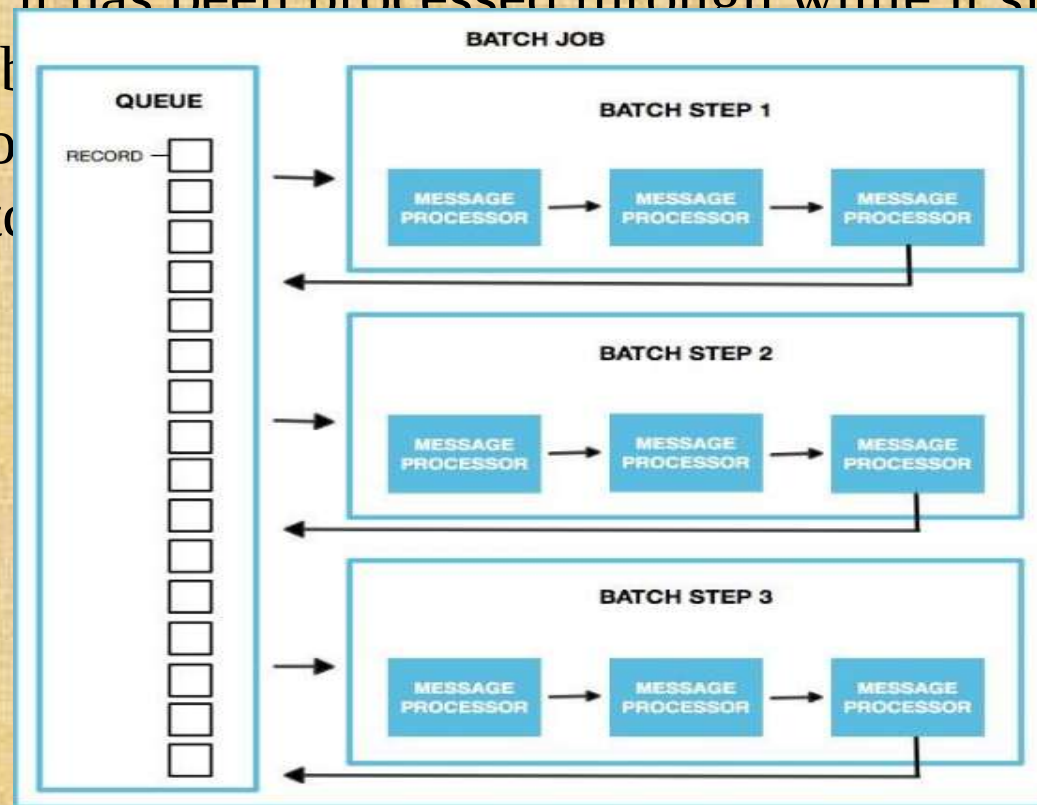
- Load and Dispatch:

The second phase, Load and Dispatch, is implicit and performs all the "behind the scenes" work to create a batch job instance. Essentially, this is the phase during which Mule turns a serialized message payload into a collection of records for processing as a batch. You don't need to configure anything for this activity to occur, though it is useful to understand the tasks Mule completes during this phase.

- Process:

Mule begins asynchronous processing of the records in the batch. Within

- Only one queue exists and records are picked out of it for each batch step, processed, and then sent back to it each record keeps track of what stages it has been processed through while it sits on this queue. Note that a batch job can have many queued records waiting to finish processing before the next batch step.



- On Complete:

we can optionally configure Mule to create a report or summary of the records it processed for the particular batch job instance. This phase exists to give system administrators and developers some insight into which records failed so as to address any issues that might exist with the input data.

- After Mule executes the entire batch job, the output becomes a batch job result object (BatchJobResult).
- you have two options for working with the output.
 - Create a report
 - Reference the batch job result object

- Batch Processing Terminology:

we have term in batch processing.

Batch, Batch Commit, Batch Job, Batch Job Instance, Batch Job Result, Batch Message Processor, Batch Phase, Batch Step, Record

- Batch Elements:

- Batch, Batch Commit, Batch Reference, Batch Threading Profile, Record Variable.

- In Batch Threading Profile, we have some attributes.

PoolExhaustedAction, maxThreadsActive, maxThreadsIdle, threadTTL, threadWaitTimeout, maxBufferSize

- BatchJobResult Processing Statistics:

batchJobInstanceId, elapsedTimeInMillis, failedOnCompletePhase, failedOnInputPhase, failedOnLoadingPhase, failedRecords, inputPhase

- Handling Failures During Batch Processing:

Mule has three options for handling a record-level error.

- Finish processing
- Continue processing
- Continue processing based on limit