Parallel For-each

* The Parallel For Each scope enables you to process a collection of messages by splitting the collection into parts that are simultaneously processed in separate routes within the scope of any limitation configured

Error Handling

* Every route is processed in parallel, if an error is thrown in one route, processing continues in all of the other routes until all finish processing. After that, all results (and any errors) are aggregated and then processed

**#[error.errorMessage.payload.failures['0']]**

**#[error.errorMessage.payload.failures['1']]  
#[error.errorMessage.payload.results['0']]**

**#[error.errorMessage.payload.results['1']]**

**For each future**

🡪If any point of time you want to check which iteration is going on

Use #[vars.counter] function

🡪 If you want to check what is the initial payload

Use #[vars.rootMessage] function

🡪Output of the payload is unmatchable (it can be change)

**Variable Activity**

🡪 not able to access the internally constructed variable on the output parallel

for-each loop.

🡪 Before the parallel for-each attributes can be access inside of the parallel for-each

Differences between For Each and Parallel For Each Scopes

Both For Each and Parallel For Each split the defined collection, and the components within the scope process each element in the collection. Also, in both cases, each route runs with the same initial context. The difference between these two scopes are:

* For Each works sequentially, while the Parallel For Each processes in parallel. This difference affects error handling:

Because of the processing differences, the execution of For Each execution is interrupted when an error is raised (and the Error Handler is invoked), while Parallel For Each processes every route before invoking the Error Handler with a MULE:COMPOSITE\_ROUTE error type.

* For Each does not modify the payload, while the Parallel For Each outputs a collection of the output messages from each iteration.