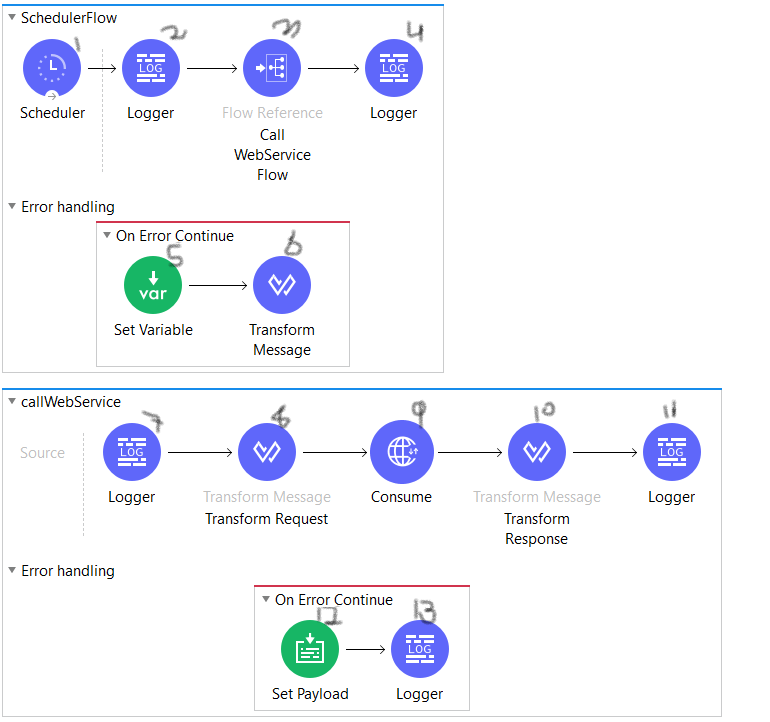
On-Error Continue catches the error, and do not report it as an error; thus the processing of the flow continues even after the error has occurred. This error handler can be used in flows where you don’t want to stop the flow processing even if an error has occurred.

**For example** in the below flow, the parent flow will execute till the end even if web consumer has returned an error.



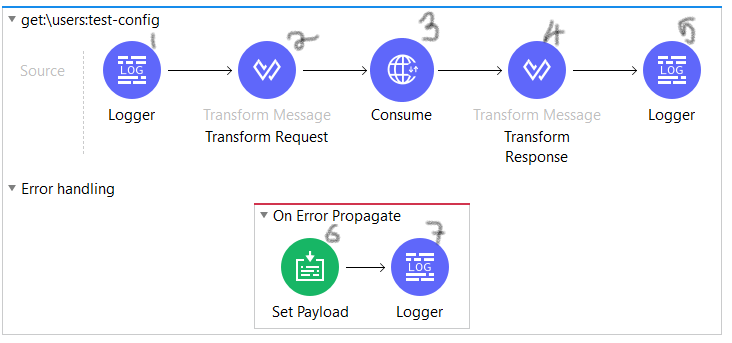
**SchedulerFlow** is calling flow **callWebService** flow**,** in case of any error at point 9 (at web service consumer) the flow will process as follows: 1->2->3->7->8->9->12->13->4.  
Here at point 13 the error is send to its parent flow (**SchedulerFlow**) as flow message, and parent flow executes its processing further.

**ON ERROR PROPAGATE**

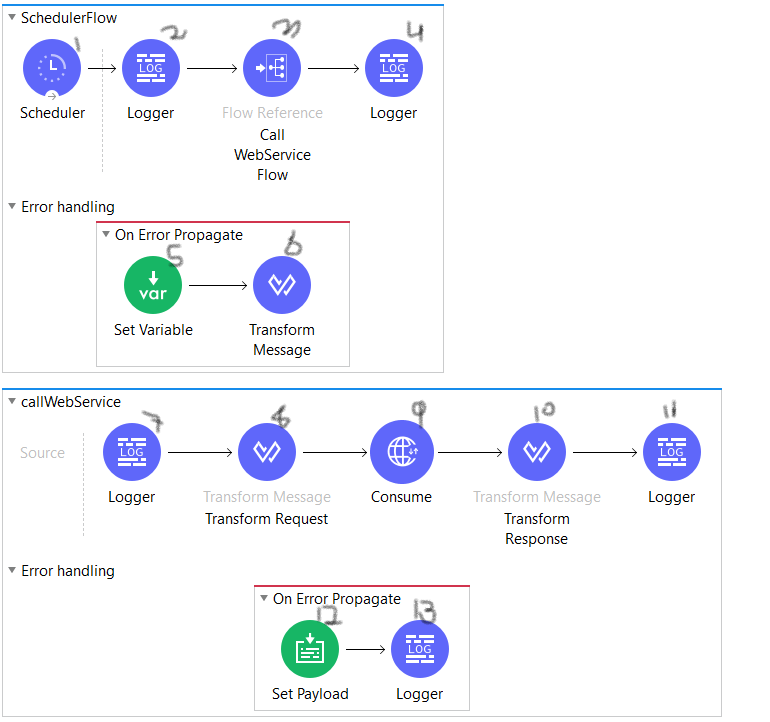
On Error Propagate works exactly as Mule 3 Catch exception strategy. In case on any error, On Error Propagate processes the error message and re-throws the error to its parent flow. No further processing is done on that particular flow.

**For example** in the below Flow, when flow execution starts, point 1, 2, 3 will execute first, on error at point 3 the error is catch by on-error propagate and error processing begins with point 6, 7; once the error handling flow is completed the flow processing ends and an error is re-thrown to its parent flow.

In can of no error or happy scenario point 1,2,3,4,5 are executed, in case of error at point 3; point 1,2,3,6,7 are executed.



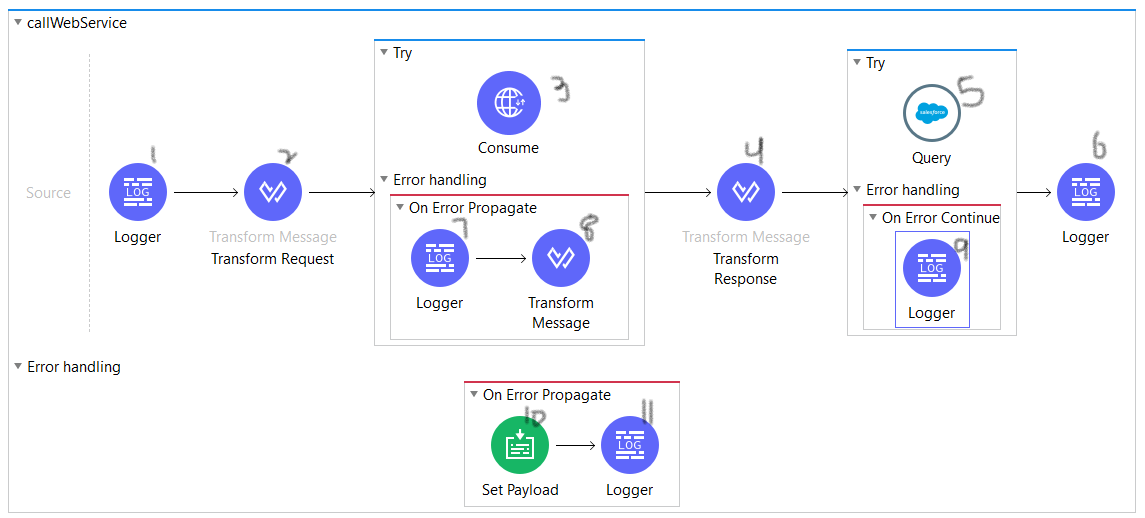
In the second example below, **SchedulerFlow** is calling flow **callWebService** flow**,** in case of any error at point 9 (at web service consumer) the flow will process as follows: 1->2->3->7->8->9->12->13->5->6.  
Here at point 13 the error is thrown to its parent flow (**SchedulerFlow**), and parent flow error handler is invoked.



TRY CATCH SCOPE

Try catch scope can be used within a flow to do error handling of just inner components. Try catch scope can be very useful in cases where we want to add separate error processing strategy for various components in the flow.

For example: In case of error at point 3 (at web service consumer) the flow will process as follows: 1->2->3->7->8->10->11.  
In case of error at point 5 (at saleforces connector) the flow will process as follows: 1->2->3->4->5->9->6.

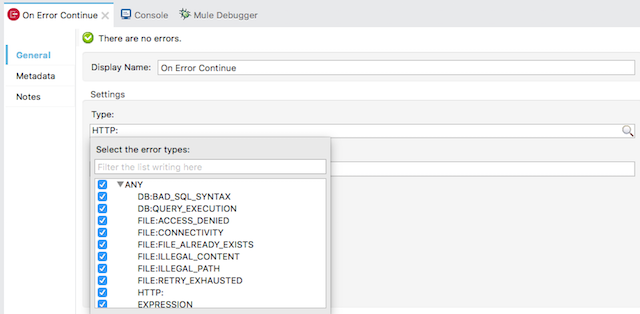


CONFIGURING ON-ERROR CONTINUE AND ON-ERROR PROPAGATE

As in Mule 3 we had to specify which error is to be catch inside the catch exception strategy, same we can do in Mule 4 with even more control.

In Mule 4 we can specify **Error Type** and/or **When Condition** which when is evaluated true that particular error handler is executed. In case none error handler catches the error the error is re-thrown to its parent flow.

**Error Type**: This matches with the type of error that is thrown. Error Type are auto populated based on connectors used in the flow. It contains the list of errors that the connectors can throw in the flow.



**When Condition:**The expression that will be evaluated to determine if the exception strategy could be executed. This should always be boolean expression.

In below example when variable **errorCount** is greater than 3 then only that particular error handler is invoked.

