**Error:**

\* Error is which stops the normal flow of execution of our program.

There are two types of errors:

1) system error 2) messaging error

\* **System error:**

Mule throws a system error when an exception occurs at the system level and no Mule event is involved. A system error handler manages exceptions that occur:

During application startup.

When a connection to an external system fails.

When a system error occurs, Mule sends an error notification to registered listeners, logs the error, and if the error is caused by a connection failure, executes a reconnection strategy.

System error handlers are not configurable in Mule.

\* **messaging error:**

Mule throws a messaging error (a [Mule error](https://docs.mulesoft.com/mule-runtime/4.2/mule-error-concept)) whenever a problem occurs within a flow of a Mule app, where Mule events and the messages they contain are processed.

You can handle Mule messaging errors in more than one way:

You can rely on the [default error handling](https://docs.mulesoft.com/mule-runtime/4.2/error-handling#default_error_handling) mechanism.

Within a flow, you can set up [On-Error components](https://docs.mulesoft.com/mule-runtime/4.2/on-error-scope-concept) (On Error Continue and On Error Propagate) inside the flow’s built-in Error Handler component. These components can contain any number of components to process the error.

**Error handling:**

a) We have two strategies. It is much simpler than before. We can catch an error and handle it, or we can rethrow it. We have also Try Scope to use it with. This scope encapsulates one or more event processors and handles for this piece of flow error.

There are three types of error handling are there:

1. On-Error Continue

2. On-Error Propagate

3. Try Catch Scope

**On-Error Continue:**

On-Error Continue catches the error and does 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.

\* **On error propagate:**

On-Error Propagate works exactly as the Mule 3 Catch exception strategy. In case of any errors, 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.

**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 a separate error processing strategy for various components in the flow.

**Global Error Handler:**

You can define a global error handler that can be referenced from any flow (except sub-flows, which inherit the calling flow error handler) using reference flow strategy.

Custom Default Global Error Handler:

**\*** You can define custom default global error handlers that will become the default error handler for your flows within the Mule application unless one of the above options is used in the flow.

**\*** Error Handler Within Mule flow

You can define the exception strategy within a flow and it will not be referenced by other flows within the Mule application.

**\*** [Mule Soft](https://www.mulesoft.com/) provides various ways to handle exceptions. Faults in Mule are generally referred to as exceptions. Mule exceptions can be divided into two categories:

System exceptions are generally invoked when the exception is thrown at system level (i.e., when no message is invoked). This type of error occurs during application startup when there is a connection failure with external systems — i.e., let's say Mule is trying to establish a connection with a JDBC database to fetch and read messages. If the connection fails, Mule invokes the system exception strategy.

Messaging exceptions are generally invoked when the messaging exception strategy is thrown into the flow.

Catch Exception Strategy:

The catch exception strategy catches all of the exceptions within the parent flow and processes them, overriding the default exception strategy. Further, you can use the catch exception strategy to:

Avoid propagating exceptions to inbound connectors.

Avoid propagating exceptions to parents via flow reference components.

Ensure that the transaction processed by the flow will not roll back when the error occurs.

Reference Exception Strategy:

You can create one or more global exception strategies in your Mule application to reuse in your flows. First, create a global exception strategy, then add a reference strategy exception to the flow to apply the error-handling behavior of your new global catch exception strategy.

Choice Exception Strategy:

You can define the choice exception strategy to customize how Mule handles the message that throws the exception based on the message's exception type at the moment it throws the exception.