Exception Handling Mechanism

The exceptions raised in your module are handled in a flow starting with an Exception Handler element. In an action, you can have more than one Exception Handler flow, in order to handle different types of exceptions.

An exception can be raised by OutSystems or in your logic at any point of your module. For typical UI requests, you can handle the raised exceptions by:

- Adding an Exception Handler element and its logic in your action’s flow.
- Adding an On Exception action in your UI Flows.
- Let the Global Exception Handler of your module do the work. By default, Global Exception Handler property of your module is set to On Exception action of Common UI Flow.

In action flows starting in Timers you can only handle the raised exceptions by adding Exception Handler elements in your logic, otherwise, the execution flow is interrupted and the error is logged.

When an exception is raised, the current execution flow is interrupted and the flow restarts in the first Exception Handler element which handles that type of exception.

As an example, consider an Action B raising a User Exception named MyUserException. Action B is invoked by Action A, which is a screen action. When MyUserException is raised in Action B, the exception handling mechanism works as follows:
You should have, at least, one Exception Handler in your application flow to inform and allow the end-user to continue to navigate.

Handling Exceptions Raised by Integrations

When you are handling exceptions raised by an integration you are consuming (such as an action of an Extension or a method of a REST API) you will not be able to determine the type of exception. In these situations, you should handle the exception with an All Exceptions Handler. Then, you can use the ExceptionMessage property of the Exception Handler element to identify the exception.

Managing Database Transactions when Handling Exceptions

When you handle an exception in an action running on server side, you can choose what happens to the database transaction by setting the Abort Transaction property of the Exception Handler:

- Set Abort Transaction property to Yes if you want that all the database transactions that weren't committed to being aborted and changes rolled back in the database.
• Set Abort Transaction property to **No** if you want that the database transaction continues to be processed as if no exception had occurred. This transaction will be committed by the next Commit Transaction action or implicitly by OutSystems, typically at the end of request execution.

In Reactive Web and Mobile apps, Abort Transaction property is available only when you are handling exceptions in server actions.