Error Handling

To easily implement error handling, it is recommended to use executable BPMN.

- A best practice that works for most scenarios is to have an **automated retry of error objects** from error state to a history state.
  This automatically fixes technically errors like network down, system temporarily down, etc.

  > The tasks must be atomic - meaning, all actions within this task can be executed again.
  > **Hint:** Use dedicated error types to mark retry-able errors.

- Retry can be implemented by a timer that periodically polls for error objects and sends the history signal (retry timer).
  - define a Timeout-Boundary at the error object
  - With this approach you prevent that you have to manually handle a a large amount of error objects.
- Errors in the model (e.g. mapping errors) can easily be solved using this approach. After fixing the error in the model and re-deployment of the service, it is sufficient to just trigger a retry and the erroneous transaction will be performed again with the rectified code. Depending on the project requirements, this can be a simple mechanism valid for all kinds of errors!

Instead of having complicated error handling in the flow, implement **dedicated validation task(s)**. Check if fields are filled resp. correctly formatted etc. This prevents that, later in the process, you have to check values before using functions like concat() Operation, substring etc.
Do not misuse exceptions for validation checks but use constraints for validation instead.

Reaction on Errors

Use the Monitoring Service With UI for a **dedicated reaction on errors**.

On this Page:

Related Pages:

- Project Organization
- Naming Conventions and Containment Tree Organisation
- Model Documentation
- Settings
- Mappings
- Sub-activities
- Logging
- Error Handling

- BPMN execution on the Bridge
- Constraints
- Monitoring Service With UI