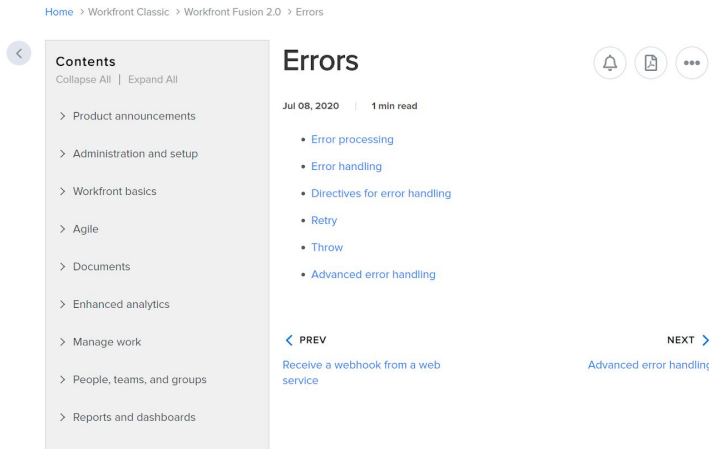


Errors

The top-level error article needs some revision

https://one.workfront.com/s/document-item?bundleId=workfront-classic&topicId=Content%2FWorkfront_Fusion%2FErrors%2F_errors.htm



https://one.workfront.com/s/document-item?bundleId=workfront-classic&topicId=Content%2FWorkfront_Fusion%2FErrors%2FError-handling.htm

Fusion's Native Error Handling

Scheduled

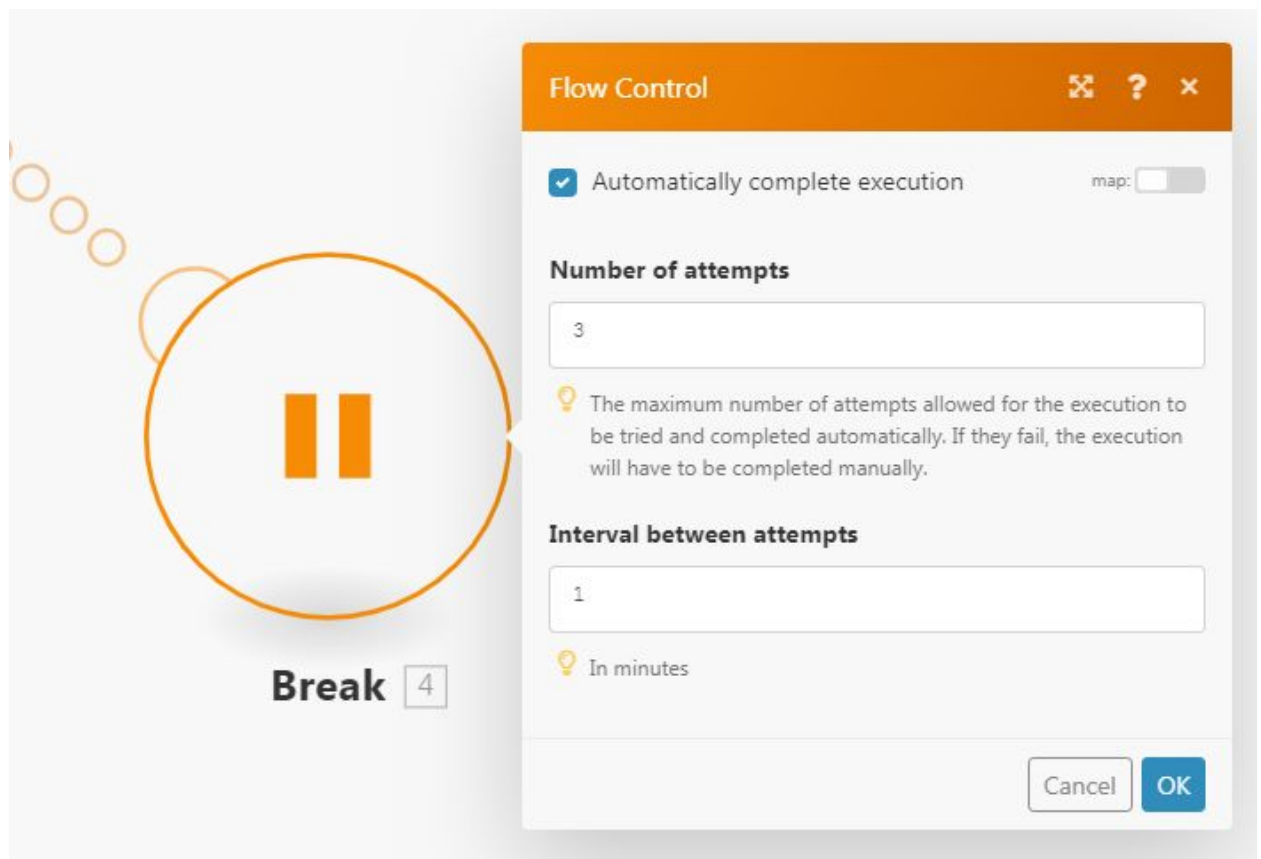
Real-time

Retries

Link to Workfront One article: <https://one.workfront.com/s/article/Retry-377721170>

Using the Break directive

1. In the [Advanced scenario settings](#) enable the "Allow storing of Incomplete Executions" option.
2. Attach an error handler route to the module (see [Introduction to Error Handling](#)).
3. Link the Break directive to the error handler route (see [Directives for error handling](#)) and configure it e.g. like this:



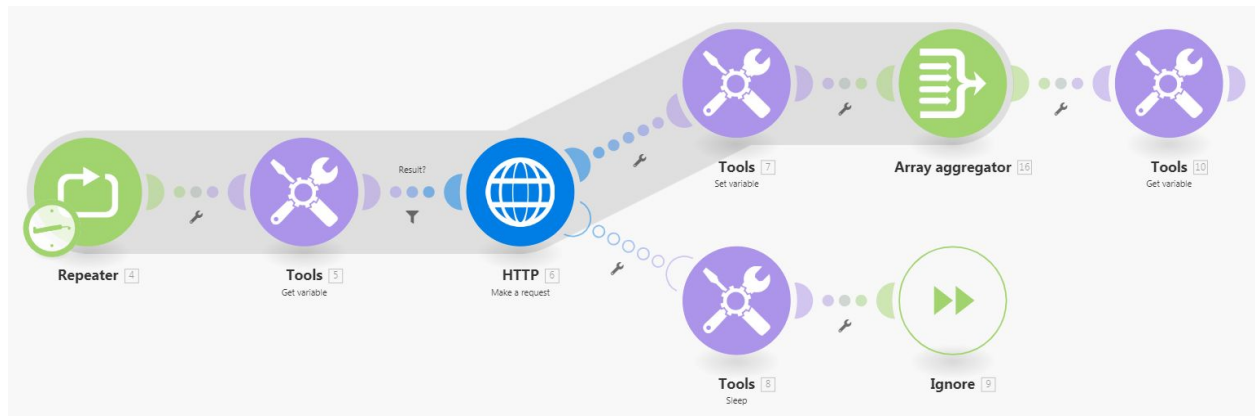
Drawbacks:

- The minimum retry interval is one minute.
- If the module is processing multiple bundles and the processing of a bundle fails, the partial execution (only the bundle that caused the error) is moved to the [Incomplete executions](#) folder and scheduled for retries according to the Break directive settings. However, the current execution continues and the module continues to process the subsequent bundles. At least you may enable the "Sequential processing" option in the Scenario settings to prevent the scenario from executing again until the execution stored in the the Incomplete executions folder has been successfully resolved.

Using the Repeater module

1. Employ the Repeater module and set its *Repeats* field to the maximum number of attempts.
2. Link the potentially failing module to the Repeater module.
3. Attach an error handler route to this module (see [Introduction to Error Handling](#)).
4. Link the Tools > Sleep module to the error handler route and set its *Delay* field to the number of seconds between the attempts.
5. Link the Ignore directive after the Tools > Sleep module (see [Directives for error handling](#)).
6. Link the Tools > Set variable module after the the potentially failing module and configure it to store the module's result in a variable named e.g. `Result`.
7. Link the Array aggregator module after the Tools > Set variable and choose the Repeater module in its *Source Module* field.
8. Link the Tools > Get variable module to the Array aggregator module and configure it to obtain the value of the `Result` variable.
9. Insert the Tools > Get variable module between the Repeater module and the potentially failing module and configure it obtain the value of the `Result` variable.
10. Insert a filter between this Tools > Get variable module and the potentially failing module to continue only if the `Result` variable does not exist.

Here is a sample scenario where the HTTP > Make a request module represents the potentially failing module (you may create this scenario easily from the [Retry template](#)):



If the result of the potentially failing module is too complex to be stored in a simple variable, you may employ [Data store](#) to store/retrieve the result. The Data store would contain just one record, the record's key can be e.g. `Result`.

Drawbacks:

- This workaround might appear a bit too complex and it is also more demanding in terms of operations.