REFramework as Library (for Power Users)

# Introduction

The goal of the REFramework is to facilitate the development of complex automations by providing the most common functionalities out-of-box:

* Exception Handling
* Retry Mechanism
* Configuration Mechanism
* Logging

Compared to classic REFramework, the current implantation as Custom Activity has the advantage of hiding the complexity of the architecture from the user, allowing him to focus specifically on the process to automate.

# How Does It Works?

The REFramework is very well suited for iterative, transactional processes, where the data can be split into several items (or parts). For that reason, the REFramework is built around *TransactionData* (the entire data to be processed) and *TransactionItem* (chunk of data to be processed during one iteration). In order to use the REFramework for an automation the developer must start by modeling the process data and defining the *TransactionData* and *TransactionItem.* For medium to complex processes this step should be straightforward. Some examples on how to choose these elements are:

* If one has to process an Excel file, *TransactionData* would be the entire datatable while the *TransactionItem* is one row from the spreadsheet
* If one uses Orchestrator queues, the *TransactionData* is the current Item Queue retrieved from the Orchestrator

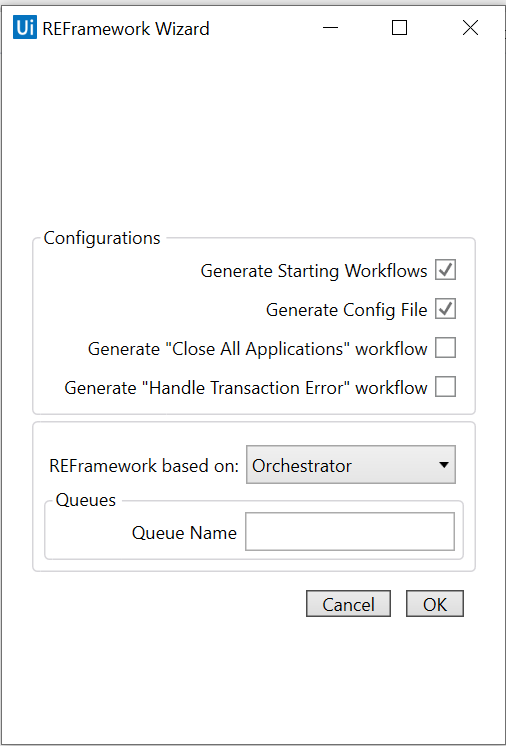
Once the developer decides which type of data he needs, he can start using the *Basic REFramework* activity. When the activity is added to a workflow, the user is presented with a wizard (see Figure 1). The wizard should be used to generate the workflows and files needed by the activity. The files will be slightly different depending on the *TransactionItem* type so careful attention must be payed when selecting the REFramework source. If the developer does not need files generation he can close/cancel the wizard, in this case an empty *Basic REFramework* being created.

Figure REFramework Wizard

Once the wizard is confirmed, an activity of type *Base REFramework* is configured and added to the workflow. At this point, the developer should have a workflow ready to run without any further modifications.

# Implementing an Automation Using REFramework Activity

In order to implement a process, the developer needs to modify following workflows:

1. *Framework/InitAllApplications.xaml* – is invoked at the beginning of the execution and should be used to initializes all applications needed by the robot and to initialize the *TransactionData.* The workflows it is also invoked after an *ApplicationException* is thrown during *ProcessTransaction*. *InitAllApplications* is always preceded by *CloseAllApplications.*
2. *Framework/GetTransactionItem.xaml* – used to retrieve the next item to be processed. Most of the times the workflow can be used as generated by the wizard without further modifications
3. *Framework/ProcessTransaction.xaml* – used to process each transaction item
4. *Framework/EndProcess.xaml* – invoked at the end of the execution, should contain final actions to be executed before the process finishes (send reports, save data, emails, etc.)
5. *Framework/CloseAllApplications.xaml* [optional]– symmetrical to *InitAllApplications,* should contain actions to close all applications used by the robot, in order to leave the environment as found.
6. *Framework/HandleTransactionStatus.xaml* [optional]– the workflow is invoked after each *ProcessTransaction.xaml.* It should handle with the status of the processing, e.g. send an email in case of exception, log the status of each transaction, etc.
7. *Data/Config.xlsx* – contains a dictionary of configurations that can be accessed in every workflow through the variable *Config.* This is in line with UiPath best practices, that discourages the usage of hard-coded strings (URLs, filepaths, …) inside a workflow. Instead, they should be grouped in a centralized external file (excel, json) easy to modify without requiring recompilation for the code.

# Exceptions Handling and Retry Mechanism

The REFramework activity has implemented an exception handling and retry mechanism as following:

* In case of any exception during *InitAllApplications,* the workflow is retried a number of times according to configuration *MaxInitRetryNumber.*
* In case of a *BusinessRuleException* in *ProcessTransaction* current processing is canceled and the next item is retrieved. In case of using Orchestrator, the current queue status is sent to the server.
* In case of an application exception during *ProcessTransaction* , the environment is reset through *CloseAllApplications* and *InitAllApplications* and the item is retried according to the configuration parameter *MaxInitNumber.*

# Logging

REFramework activity propose some predefined logs:

* When REFramework starts and ends
* When an external workflow is invoked
* When a new transaction is started.

# Conclusions

REFramework as custom activity implements a powerful framework for developing automations. By providing a wizard, an user friendly interface and by hiding the complexity behind a library, this framework targets users with medium experience in UiPath, that can focus now on the process to automate, instead of worrying about issues such as exception handling, retry mechanism and configuration files.