**Technical Design Document**

**REF-UIPATH-RHENRY**

Table of Contents

# **1 Introduction**

Infrastructure d'entreprise robotique de renaud HENRY

# **2 Purpose of this document**

This is a Technical Design Document that explains the technical aspects of the robot designed and developed using UiPath in detail. This will give an overview of the design of the bot and can be used by developers or other stakeholders to understand the prerequisites and requirements to execute the bot successfully.

# **3 Scope**

**The scope of this document includes:**

• Environment Specification  
• System Requirements  
• Prerequisites  
• UiPath Enterprise  
• File Folder Structure  
• Robot Design  
• Issues and Risks

**The scope of this document does not include:**

• Availability of Systems  
• System Changes  
• Changes to Input Files or Data Format  
• Process Changes

*Note – this is a Technical Design Document which only covers the technical aspects. Please refer to the Business Requirements Document for any other information about business processes*

# **4 System Requirements**

The Developer System (specifications below) was used to develop this Technical Design Document. It is important to note that the system the robot is migrated to should also have similar specifications to ensure a proper functioning of the robot.  
The system specifications for REF-UIPATH-RHENRY process:

|  |  |
| --- | --- |
| **Operating System** |  |
| **Processor** |  |
| **RAM** |  |
| **Hard Disk** |  |
| **Components** |  |

# **5 Prerequisites**

The prerequisites for the robot to successfully run are as follows:

a. The following applications are installed in the system:  
 • UiPath 21.10.3.0  
b. The system has a valid studio and back office robot licence for UiPath  
c. Robot has access to all required applications  
d. Files are located in the system following the File Structure indicated below (Section 7).

# **6 UiPath Enterprise**

UiPath version 21.10.3.0 is the software used for developing the Technical Design Document.

# **7 File/Folder Structure**

**• Temporary Folder:**

Path at which all the temporary files are stored.

**• Input Folder:**

Path at which all the input files are stored.

**• Output Folder:**

Path at which all the output files are stored.

**• Code Repository**

Path at UiPath process workflows are be stored.

**• Config File Path**

Path at config files are be stored.

|  |  |
| --- | --- |
| File/Folder | File/Folder Location |
| Temporary Folder |  |
| Input Folder |  |
| Output Folder |  |
| Code Repository |  |
| Config File Path |  |

# **8 Bot Design**

8.1 Main.xaml

[Process title]  
[Process description]  
[Additional information (e.g., author, contact information and applications involved and required external setup)]  
1 Get the next transaction to be processed.  
2 Retrieve a new transaction data to be processed.  
The TransactionNumber variable holds the current transaction number and incrementing this variable makes the framework retrieve the next transaction. If the framework is retrying a failed transaction, this variable is not incremented until the maximum number of retry attempts is reached.  
3 Process a single transaction.   
The result of the processing can be: 1) Success, 2) Business Exception, 3) System Exception.   
In the case of a system exception, the transaction can be automatically retried.  
4 Read configuration file and initialize applications used in the process.  
5 Kills all Windows processes representing applications used in this business process to assure that the execution starts in a clean state.  
Since the applications are assumed to be already closed, CloseAllApplications is skipped and just KillAllProcess is invoked.  
6 Add the process name to the logs generated after this point.   
This log field can be used to create reports and visualizations about the process.  
7 Failures during the initialization are considered system exceptions and lead to the End Process state, thus finalizing the execution.  
8 End process and close all applications used.  
9 Business Rule Exception  
10 There is no need for any action in case of successful transaction.  
The process should simply go to next transaction.  
11 This is a simple mecanism to stop the process.  
In reallife scenario you can stop the process when there is no more data to process or on a schedule.  
12 Transaction item to be processed. The type of this variable can be changed to match the transaction type in the process. For example, when processing data from a spreadsheet that is read into a DataTable, this type can be changed to DataRow.   
13 Used during transitions between states to represent exceptions other than business exceptions.  
14 Used during transitions between states and represents a situation that does not conform to the rules of the process being automated.  
15 Sequential counter of transaction items.  
16 Dictionary structure to store configuration data of the process (settings, constants and assets).  
17 Used to control the number of attempts of retrying the transaction processing in case of system exceptions.  
18 Optionally used to include additional information about the transaction item.  
19 Optionally used to include additional information about the transaction item.  
20 Transaction ID used for information and logging purposes. Ideally, the ID should be unique for each transaction.   
21 Used in case transactions are stored in a DataTable, for example, after being retrieved from a spreadsheet.  
22 Used to control the number of consecutive system exceptions.

***Location: \Main.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_Env | InArgument(x:String) |  |

8.2 Main.xaml

[Process title]  
[Process description]  
[Additional information (e.g., author, contact information and applications involved and required external setup)]  
1 Get the next transaction to be processed.  
2 Retrieve a new transaction data to be processed.  
The TransactionNumber variable holds the current transaction number and incrementing this variable makes the framework retrieve the next transaction. If the framework is retrying a failed transaction, this variable is not incremented until the maximum number of retry attempts is reached.  
3 Process a single transaction.   
The result of the processing can be: 1) Success, 2) Business Exception, 3) System Exception.   
In the case of a system exception, the transaction can be automatically retried.  
4 There is no need for any action in case of successful transaction.  
The process should simply go to next transaction.  
5 Business Rule Exception  
6 Initialize applications used in the process.  
7 If MaxConsecutiveSystemExceptions number was reached, throw Exception at initialization and go to End Process state, thus finalizing the execution.   
If MaxConsecutiveSystemExceptions is 0, then any number of consecutive System Exceptions is allowed.  
8 Failures during the initialization are considered system exceptions and lead to the End Process state, thus finalizing the execution.  
9 End process and close all applications used.  
10 This is a simple mecanism to stop the process.  
In reallife scenario you can stop the process when there is no more data to process or on a schedule.  
11 Read configuration file  
12 Failures during the initialization are considered system exceptions and lead to the End Process state, thus finalizing the execution.  
13 Create TrasactionItems  
14 Failures during the initialization are considered system exceptions and lead to the End Process state, thus finalizing the execution.  
15 Transaction item to be processed. The type of this variable can be changed to match the transaction type in the process. For example, when processing data from a spreadsheet that is read into a DataTable, this type can be changed to DataRow.   
16 Used during transitions between states to represent exceptions other than business exceptions.  
17 Used during transitions between states and represents a situation that does not conform to the rules of the process being automated.  
18 Sequential counter of transaction items.  
19 Dictionary structure to store configuration data of the process (settings, constants and assets).  
20 Used to control the number of attempts of retrying the transaction processing in case of system exceptions.  
21 Optionally used to include additional information about the transaction item.  
22 Optionally used to include additional information about the transaction item.  
23 Transaction ID used for information and logging purposes. Ideally, the ID should be unique for each transaction.   
24 Used in case transactions are stored in a DataTable, for example, after being retrieved from a spreadsheet.  
25 Used to control the number of consecutive system exceptions.

***Location: \Main\_REFC.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_Env | InArgument(x:String) |  |

8.3 InitiAllSettings.xaml

Initialize, populate and output a configuration Dictionary to be used throughout the project.   
Settings and constants are read from the local configuration file, and assets are fetched from Orchestrator.   
Asset values overwrite settings and constant values if they are defined with the same name.  
1 Read settings and constants from the configuration file and add them to the Config dictionary.

***Location: \Framework\01-InitAllSettings.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_ConfigFile | InArgument(x:String) | Path to the configuration file that defines settings, constants and assets. |
| in\_Env | InArgument(x:String) |  |
| out\_Config | OutArgument(scg:Dictionary(x:String, scg:Dictionary(x:String, x:Object))) | Dictionary structure to store configuration data of the process (settings, constants and assets). |

8.4 GetTransactionData.xaml

Get a transaction item from a specified source (e.g., Orchestrator queues, spreadsheets, databases, mailboxes or web APIs).   
  
If there are no transaction items remaining, out\_TransactionItem is set to Nothing, which leads to the End Process state.   
  
For cases in which there is only a single transaction (i.e., a linear process), use an If activity to check whether the argument in\_TransactionNumber has the value 1 (meaning it is the first and only transaction) and assign the transaction item to out\_TransactionItem. For any other value of in\_TransactionNumber, out\_TransactionItem should be set to Nothing.  
  
If there are multiple transactions, use the argument in\_TransactionNumber as an index to retrieve the correct transaction to be processed. If there are no more transactions left, it is necessary to set out\_TransactionItem to Nothing, thus ending the process.  
1 Get a transaction item from the specified Orchestrator queue.  
If queues are not used in this process, replace this activity with the appropriated logic to retrieve transaction items.  
For example, if transactions are rows from a DataTable, the row corresponding to the current transaction is retrieved at this point.

***Location: \Framework\GetTransactionData.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_TransactionNumber | InArgument(x:Int32) | Sequential counter of transaction items. |
| in\_Config | InArgument(scg:Dictionary(x:String, scg:Dictionary(x:String, x:Object))) | Dictionary structure to store configuration data of the process (settings, constants and assets). |
| out\_TransactionItem | OutArgument(ui:QueueItem) | Transaction item to be processed. |
| io\_dt\_TransactionData | InOutArgument(sd:DataTable) | This variable can be used in case transactions are stored in a DataTable (for example, after being retrieved from a spreadsheet). |

8.5 ReacapAll.xaml

***Location: \Framework\ReacapAll.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_NoSessions | InArgument(x:String) |  |
| in\_Config | InArgument(scg:Dictionary(x:String, scg:Dictionary(x:String, x:Object))) |  |

8.6 RetryCurrentTransaction.xaml

Manage the retrying mechanism for the framework and it is invoked in SetTransactionStatus.xaml when a system exception occurs.   
The retrying method is based on the configurations defined in Config.xlsx.

***Location: \Framework\RetryCurrentTransaction.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_Config | InArgument(scg:Dictionary(x:String, scg:Dictionary(x:String, x:Object))) | Dictionary structure to store configuration data of the process (settings, constants and assets). |
| io\_RetryNumber | InOutArgument(x:Int32) | Used to control the number of attempts of retrying the transaction processing in case of system exceptions. |
| io\_TransactionNumber | InOutArgument(x:Int32) | Sequential counter of transaction items. |
| in\_SystemException | InArgument(s:Exception) | Used during transitions between states to represent exceptions other than business exceptions. |
| in\_QueueRetry | InArgument(x:Boolean) | Used to indicate whether the retry procedure is managed by an Orchestrator queue. |

8.7 SetTransactionStatus.xaml

Set and log the transaction's status along with extra log fields.   
There can be three possible statuses: Success, Business Exception and System Exception.  
  
Business Rule Exception characterizes an irregular situation according to the process's rules and prevents the transaction to be processed. The transaction is not retried in this case, since the result will be the same until the problem that causes the exception is solved.  
For example, it can be considered a BusinessRuleException if a process expects to read an email's attachment, but the sender didn't attach any file. In this case, immediate retries of the transaction will not yield a different result.  
  
On the other hand, system exceptions are characterized by exceptions whose types are different than BusinessRuleException. When this kind of exception happens, the transaction item can be retried after closing and reopening the applications involved in the process. The rationale behind this is that the exception was caused by a problem in the applications, which might be solved by restarting them.  
  
If Orchestrator queues are the source of transactions, the Set Transaction Status activity is used to update the status. In addition, the retry mechanism is also implemented by Orchestrator.  
  
If Orchestrator queues are not used, the status can be set, for example, by writing to a specific column in a spreadsheet. In such cases, the retry mechanism is covered by the framework and the number of retries is defined in the configuration file.  
  
At the end, io\_TransactionNumber is incremented, which makes the framework get the next transaction to be processed.  
1 If the transaction item is processed without any exception, its status is updated as Successful.  
2 Includes custom log fields to the log message.  
They are removed after logging to prevent duplicated status messages about a single transaction.  
3 Increment the TransactionNumber to get the next transaction to be processed.  
4 Reset the counter of retries to allow the next transaction to be retried the correct amount of times.  
5 Reset the counter of consecutive system exceptions.  
6 If a BussinessRuleException is thrown during the process, the transaction item's status is updated as Failed (Exception Type: Business).  
  
7 Includes custom log fields to the log message.  
They are removed after logging to prevent duplicated status messages about a single transaction.  
8 If a system exception occurs during the process, the transaction item's status is updated as Failed (Exception Type: Application).  
9 Take a screenshot of the current state of the screen to facilitate debugging.  
10 Close all applications before returning to the Initialization state and opening them again.  
If applications cannot be closed, kill their respective processes.

***Location: \Framework\SetTransactionStatus.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_BusinessException | InArgument(ui:BusinessRuleException) | Exception variable that is used during transitions between states and represents a situation that does not conform to the rules of the process being automated. |
| in\_TransactionField1 | InArgument(x:String) | Optionally used to include additional information about the transaction item. |
| in\_TransactionField2 | InArgument(x:String) | Optionally used to include additional information about the transaction item. |
| in\_TransactionID | InArgument(x:String) | Used for information and logging purposes. Ideally, the ID should be unique for each transaction. |
| in\_SystemException | InArgument(s:Exception) | Used during transitions between states to represent exceptions other than business exceptions. |
| in\_Config | InArgument(scg:Dictionary(x:String, scg:Dictionary(x:String, x:Object))) | Dictionary structure to store configuration data of the process (settings, constants and assets). |
| in\_TransactionItem | InArgument(ui:QueueItem) | Transaction item to be processed. |
| io\_RetryNumber | InOutArgument(x:Int32) | Used to control the number of attempts of retrying the transaction processing in case of system exceptions. |
| io\_TransactionNumber | InOutArgument(x:Int32) | Sequential counter of transaction items. |
| io\_ConsecutiveSystemExceptions | InOutArgument(x:Int32) | Used to control the number of consecutive system exceptions. |
| in\_DicoAnalyse | InArgument(scg:Dictionary(x:String, x:String)) |  |
| in\_DicoOutput | InArgument(scg:Dictionary(x:String, x:String)) |  |

8.8 TakeScreenshot.xaml

Capture a screenshot, log its name and location and save it with the PNG extension.  
If no specific filepath is passed as argument, it saves the image in the folder specified by in\_Folder.  
1 The whole screen is captured.

***Location: \Framework\TakeScreenshot.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_Folder | InArgument(x:String) | Path to the folder where the screenshot should be saved. |
| io\_FilePath | OutArgument(x:String) | Optional argument that specifies the path and the name of the screenshot to be taken. |

8.9 Get\_Asset\_Dynamique.xaml

***Location: \Framework\Lib\GetAssetDynamique.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_NameAsset | InArgument(x:String) |  |
| in\_TypeAsset | InArgument(x:String) |  |
| out\_Value | OutArgument(x:Object) |  |
| in\_Config | InArgument(scg:Dictionary(x:String, scg:Dictionary(x:String, x:Object))) |  |

8.10 GetAssetTryCatch.xaml

***Location: \Framework\Lib\GetAssetTryCatch.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| io\_Config | InArgument(scg:Dictionary(x:String, scg:Dictionary(x:String, x:Object))) |  |
| in\_NameAsset | InArgument(x:String) |  |
| in\_FolderAsset | InArgument(x:String) |  |
| in\_Rubrique | InArgument(x:String) |  |
| in\_Name | InArgument(x:String) |  |

8.11 ReadSheetExcelAssets.xaml

Read non-empty rows in the sheet.

***Location: \Framework\Lib\ReadSheetExcelAssets.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_ConfigFile | InArgument(x:String) |  |
| in\_SheetName | InArgument(x:String) |  |
| in\_IsLogAdvanced | InArgument(x:Boolean) |  |
| io\_Config | InOutArgument(scg:Dictionary(x:String, scg:Dictionary(x:String, x:Object))) |  |

8.12 ReadSheetExcel.xaml

Read non-empty rows in the sheet.

***Location: \Framework\Lib\ReadSheetExcelSettings.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_SheetName | InArgument(x:String) |  |
| io\_Config | InOutArgument(scg:Dictionary(x:String, scg:Dictionary(x:String, x:Object))) |  |
| in\_ConfigFile | InArgument(x:String) |  |
| in\_IsLogAdvanced | InArgument(x:Boolean) |  |

8.13 SetConfigFromAsset.xaml

***Location: \Framework\Lib\SetConfigFromAsset.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_Name | InArgument(x:String) |  |
| in\_Rubrique | InArgument(x:String) |  |
| in\_Asset | InArgument(x:String) |  |
| in\_Type | InArgument(x:String) |  |
| in\_OrchestratorAssetFolder | InArgument(x:String) |  |
| io\_Config | InOutArgument(scg:Dictionary(x:String, scg:Dictionary(x:String, x:Object))) |  |
| in\_IsPatern | InArgument(x:String) |  |

8.14 SetConfig.xaml

***Location: \Framework\Lib\SetConfigFromExcel.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_Rubrique | InArgument(x:String) |  |
| in\_Name | InArgument(x:String) |  |
| in\_Value | InArgument(x:String) |  |
| io\_Config | InOutArgument(scg:Dictionary(x:String, scg:Dictionary(x:String, x:Object))) |  |
| in\_IsPaterne | InArgument(x:String) |  |
| in\_IsLogAdvanced | InArgument(x:Boolean) |  |

8.15 CloseAllApplications.xaml

Do the necessary procedures for ending the process (e.g., logout) and close the used applications.

***Location: \Implementation\00-Framework\CloseAllApplications.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_Config | InArgument(scg:Dictionary(x:String, scg:Dictionary(x:String, x:Object))) |  |

8.16 KillAllProcesses.xaml

Use the Kill Process activity to force the termination of the Windows processes representing applications used in the business process being automated.  
Note that killing processes might have undesirable outcomes, such as losing unsaved changes to files.

***Location: \Implementation\00-Framework\KillAllProcesses.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_Config | InArgument(scg:Dictionary(x:String, scg:Dictionary(x:String, x:Object))) |  |

8.17 RunStart.xaml

***Location: \Implementation\01\_RunStart\RunStart.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_Config | InArgument(scg:Dictionary(x:String, scg:Dictionary(x:String, x:Object))) |  |

8.18 DispatcherRoot.xaml

***Location: \Implementation\02\_Dispatcher\DispatcherRoot.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_Config | InArgument(scg:Dictionary(x:String, scg:Dictionary(x:String, x:Object))) |  |
| out\_NoSession | InArgument(x:String) |  |

8.19 DispatcherUsed.xaml

***Location: \Implementation\02\_Dispatcher\DispatcherUsed.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_RAZ | InArgument(x:String) |  |
| in\_QueueName | InArgument(x:String) |  |
| in\_QueuePath | InArgument(x:String) |  |
| in\_SiteWeb | InArgument(x:String) |  |
| in\_Config | InArgument(scg:Dictionary(x:String, scg:Dictionary(x:String, x:Object))) |  |
| out\_NoSession | OutArgument(x:String) |  |

8.20 Initialize\_Applications.xaml

Open applications used in the process and do necessary initialization procedures (e.g., login).

***Location: \Implementation\03\_InitApps\InitAllApplications.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_Config | InArgument(scg:Dictionary(x:String, scg:Dictionary(x:String, x:Object))) | Dictionary structure to store configuration data of the process (settings, constants and assets). |

8.21 Process.xaml

Invoke major steps of the business process, which are usually implemented by multiple subworkflows.  
  
If a BusinessRuleException is thrown, the transaction is skipped.   
If another kind of exception occurs, the current transaction can be retried.

***Location: \Implementation\04\_Process\Process.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_TransactionItem | InArgument(ui:QueueItem) | Transaction item to be processed. |
| in\_Config | InArgument(scg:Dictionary(x:String, scg:Dictionary(x:String, x:Object))) | Dictionary structure to store configuration data of the process (settings, constants and assets). |
| out\_DicoAnalyse | OutArgument(scg:Dictionary(x:String, x:String)) |  |
| out\_DicoOutput | OutArgument(scg:Dictionary(x:String, x:String)) |  |

8.22 Workflow\_Main.xaml

***Location: \Implementation\04\_Process\Workflow\_Main.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_Nom | InArgument(x:String) |  |
| in\_SiteWeb | InArgument(x:String) |  |
| in\_Config | InArgument(scg:Dictionary(x:String, scg:Dictionary(x:String, x:Object))) |  |
| in\_FolderWotking | InArgument(x:String) |  |
| io\_DicoAnalyse | InOutArgument(scg:Dictionary(x:String, x:String)) |  |

8.23 SetTranscationItemWithAnalyse.xaml

***Location: \Implementation\05\_SetTransactionItem\SetTranscationItemWithAnalyse.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_Config | InArgument(scg:Dictionary(x:String, scg:Dictionary(x:String, x:Object))) |  |
| in\_TransactionItem | InArgument(ui:QueueItem) |  |
| in\_DicoAnalyse | InArgument(scg:Dictionary(x:String, x:String)) |  |

8.24 RunStart.xaml

***Location: \Implementation\06\_RunEnd\RunEnd.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_Config | InArgument(scg:Dictionary(x:String, scg:Dictionary(x:String, x:Object))) |  |

8.25 SendEmailBySMTP.xaml

***Location: \Librairy\Email\SendEmailBySMTP.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_To | InArgument(x:String) |  |
| in\_Cc | InArgument(x:String) |  |
| in\_Objet | InArgument(x:String) |  |
| in\_Body | InArgument(x:String) |  |
| in\_Port | InArgument(x:Int32) |  |
| in\_Serveur | InArgument(x:String) |  |
| in\_ListeFiles | InArgument(scg:List(x:String)) |  |
| in\_Credential\_Login | InArgument(x:String) |  |
| in\_EmailFrom | InArgument(x:String) |  |
| in\_NameFrom | InArgument(x:String) |  |
| in\_Credential\_Pwd | InArgument(x:String) |  |

8.26 Monkeys.xaml

***Location: \Librairy\Errors\Monkey.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_Config | InArgument(scg:Dictionary(x:String, scg:Dictionary(x:String, x:Object))) |  |
| in\_StringChance | InArgument(x:String) |  |
| in\_Actived | InArgument(x:String) |  |
| in\_Exception | InArgument(s:Exception) |  |
| out\_WithErrors | OutArgument(x:Boolean) |  |

8.27 SendEmailError.xaml

***Location: \Librairy\Errors\SendEmailError.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_Config | InArgument(scg:Dictionary(x:String, scg:Dictionary(x:String, x:Object))) |  |
| in\_SystemeException | InArgument(s:Exception) |  |
| in\_BusinessException | InArgument(ui:BusinessRuleException) |  |

8.28 Get\_Env.xaml

***Location: \Librairy\Orchestrateur\Get\_Env.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| out\_Env | OutArgument(x:String) |  |

8.29 GetInfoOfQItems.xaml

***Location: \Librairy\Orchestrateur\NoDeSession\Libs\GetInfoOfQItems.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_ClientId | InArgument(x:String) |  |
| in\_RefreshToken | InArgument(x:String) |  |
| in\_TenantName | InArgument(x:String) |  |
| in\_AccountLogicalName | InArgument(x:String) |  |
| in\_FolderOrchrestrator | InArgument(x:String) |  |
| in\_QueueName | InArgument(x:String) |  |
| out\_JSON\_ITEMS | OutArgument(njl:JObject) |  |

8.30 GetLastSession.xaml

***Location: \Librairy\Orchestrateur\NoDeSession\Libs\GetInfoOfSession.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_FolderOrchrestrator | InArgument(x:String) |  |
| in\_QueueName | InArgument(x:String) |  |
| in\_NoSession | InArgument(x:String) |  |
| in\_FolderRecap | InArgument(x:String) |  |
| out\_PathExcel | OutArgument(x:String) |  |
| out\_DT | OutArgument(sd:DataTable) |  |
| out\_Rouges | OutArgument(x:String) |  |
| out\_Verts | OutArgument(x:String) |  |
| out\_Oranges | OutArgument(x:String) |  |

8.31 GetLastSession.xaml

***Location: \Librairy\Orchestrateur\NoDeSession\Libs\GetLastSession.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_FolderOrchrestrator | InArgument(x:String) |  |
| in\_QueueName | InArgument(x:String) |  |
| out\_LastID | OutArgument(x:Int32) |  |

8.32 Séquence\_Etape1.xaml

***Location: \Tests\Séquence-Etape1.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Config | OutArgument(scg:Dictionary(x:String, scg:Dictionary(x:String, x:Object))) |  |

8.33 Test\_EmailSMTP.xaml

***Location: \Tests\Test-EmailSMTP.xaml***

# **9 Workflows**

|  |  |  |
| --- | --- | --- |
| Name | Invoked Workflow | Invoked In |
| \Main.xaml | •Framework\GetTransactionData.xaml •Implementation\04\_Process\Process.xaml •Framework\SetTransactionStatus.xaml •Framework\01-InitAllSettings.xaml •Implementation\00-Framework\KillAllProcesses.xaml •Implementation\00-Framework\CloseAllApplications.xaml |  |
| \Main\_REFC.xaml | •Framework\GetTransactionData.xaml •Librairy\Errors\SendEmailError.xaml •Implementation\04\_Process\Process.xaml •Framework\SetTransactionStatus.xaml •Implementation\00-Framework\CloseAllApplications.xaml •Implementation\00-Framework\KillAllProcesses.xaml •Implementation\03\_InitApps\InitAllApplications.xaml •Implementation\06\_RunEnd\RunEnd.xaml •Framework\ReacapAll.xaml •Framework\01-InitAllSettings.xaml •Implementation\01\_RunStart\RunStart.xaml •Implementation\02\_Dispatcher\DispatcherRoot.xaml |  |
| \Framework\01-InitAllSettings.xaml | •Framework\Lib\ReadSheetExcelSettings.xaml •Framework\Lib\ReadSheetExcelAssets.xaml | •Main.xaml •Main\_REFC.xaml •Tests\Séquence-Etape1.xaml |
| \Framework\GetTransactionData.xaml |  | •Main.xaml •Main\_REFC.xaml |
| \Framework\ReacapAll.xaml | •Librairy\Orchestrateur\NoDeSession\Libs\GetInfoOfSession.xaml •Librairy\Email\SendEmailBySMTP.xaml | •Main\_REFC.xaml |
| \Framework\RetryCurrentTransaction.xaml |  |  |
| \Framework\SetTransactionStatus.xaml | •Implementation\05\_SetTransactionItem\SetTranscationItemWithAnalyse.xaml •Framework\TakeScreenshot.xaml •Framework/RetryCurrentTransaction.xaml •Implementation\00-Framework\CloseAllApplications.xaml •Implementation\00-Framework\KillAllProcesses.xaml | •Main.xaml •Main\_REFC.xaml |
| \Framework\TakeScreenshot.xaml |  | •Framework\SetTransactionStatus.xaml •Librairy\Errors\SendEmailError.xaml |
| \Framework\Lib\GetAssetDynamique.xaml |  | •Framework\Lib\SetConfigFromAsset.xaml |
| \Framework\Lib\GetAssetTryCatch.xaml |  |  |
| \Framework\Lib\ReadSheetExcelAssets.xaml | •Framework\Lib\SetConfigFromAsset.xaml | •Framework\01-InitAllSettings.xaml |
| \Framework\Lib\ReadSheetExcelSettings.xaml | •Framework\Lib\SetConfigFromExcel.xaml | •Framework\01-InitAllSettings.xaml |
| \Framework\Lib\SetConfigFromAsset.xaml | •Framework\Lib\GetAssetDynamique.xaml | •Framework\Lib\ReadSheetExcelAssets.xaml |
| \Framework\Lib\SetConfigFromExcel.xaml |  | •Framework\Lib\ReadSheetExcelSettings.xaml |
| \Implementation\00-Framework\CloseAllApplications.xaml |  | •Main.xaml •Main\_REFC.xaml •Framework\SetTransactionStatus.xaml •Implementation\04\_Process\Workflow\_Main.xaml |
| \Implementation\00-Framework\KillAllProcesses.xaml |  | •Main.xaml •Main\_REFC.xaml •Framework\SetTransactionStatus.xaml •Implementation\04\_Process\Workflow\_Main.xaml |
| \Implementation\01\_RunStart\RunStart.xaml |  | •Main\_REFC.xaml •Tests\Séquence-Etape1.xaml |
| \Implementation\02\_Dispatcher\DispatcherRoot.xaml | •Implementation\02\_Dispatcher\DispatcherUsed.xaml | •Main\_REFC.xaml |
| \Implementation\02\_Dispatcher\DispatcherUsed.xaml | •Librairy\Orchestrateur\NoDeSession\Libs\GetLastSession.xaml | •Implementation\02\_Dispatcher\DispatcherRoot.xaml |
| \Implementation\03\_InitApps\InitAllApplications.xaml |  | •Main\_REFC.xaml |
| \Implementation\04\_Process\Process.xaml | •Implementation\04\_Process\Workflow\_Main.xaml •Librairy\Errors\SendEmailError.xaml | •Main.xaml •Main\_REFC.xaml |
| \Implementation\04\_Process\Workflow\_Main.xaml | •Implementation\00-Framework\CloseAllApplications.xaml •Implementation\00-Framework\KillAllProcesses.xaml | •Implementation\04\_Process\Process.xaml |
| \Implementation\05\_SetTransactionItem\SetTranscationItemWithAnalyse.xaml |  | •Framework\SetTransactionStatus.xaml |
| \Implementation\06\_RunEnd\RunEnd.xaml |  | •Main\_REFC.xaml |
| \Librairy\Email\SendEmailBySMTP.xaml |  | •Framework\ReacapAll.xaml •Librairy\Errors\SendEmailError.xaml •Tests\Test-EmailSMTP.xaml |
| \Librairy\Errors\Monkey.xaml |  |  |
| \Librairy\Errors\SendEmailError.xaml | •Framework\TakeScreenshot.xaml •Librairy\Email\SendEmailBySMTP.xaml | •Main\_REFC.xaml •Implementation\04\_Process\Process.xaml |
| \Librairy\Orchestrateur\Get\_Env.xaml |  |  |
| \Librairy\Orchestrateur\NoDeSession\Libs\GetInfoOfQItems.xaml |  | •Librairy\Orchestrateur\NoDeSession\Libs\GetInfoOfSession.xaml •Librairy\Orchestrateur\NoDeSession\Libs\GetLastSession.xaml |
| \Librairy\Orchestrateur\NoDeSession\Libs\GetInfoOfSession.xaml | •Librairy\Orchestrateur\NoDeSession\Libs\GetInfoOfQItems.xaml | •Framework\ReacapAll.xaml |
| \Librairy\Orchestrateur\NoDeSession\Libs\GetLastSession.xaml | •Librairy\Orchestrateur\NoDeSession\Libs\GetInfoOfQItems.xaml | •Implementation\02\_Dispatcher\DispatcherUsed.xaml |
| \Tests\Séquence-Etape1.xaml | •Framework\01-InitAllSettings.xaml •Implementation\01\_RunStart\RunStart.xaml |  |
| \Tests\Test-EmailSMTP.xaml | •Librairy\Email\SendEmailBySMTP.xaml |  |

# **10 Dependencies**

• Bibliothèque\_Log: [1.0.2]  
• MongoDB.Bson: [2.13.2]  
• MongoDB.Driver: [2.13.2]  
• MongoDB.Driver.Core: [2.13.2]  
• UiPath.Excel.Activities: [2.11.4]  
• UiPath.Mail.Activities: [1.12.2]  
• UiPath.System.Activities: [21.10.2]  
• UiPath.Testing.Activities: [1.4.3]  
• UiPath.UIAutomation.Activities: [21.10.3]  
• UiPath.WebAPI.Activities: [1.9.2]

# **11 Issues and Risks**

Below are the issues and risks identified during development and testing