**Technical Design Document**

**DO and Invoice Process Automation - Bot 2**

Table of Contents

# **1 Introduction**

Robotic Enterprise Framework

# **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 DO and Invoice Process Automation - Bot 2 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 22.4.6.0  
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 22.4.6.0 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

DO and Invoice Process Automation  
2nd bot for fetching all the manual entries from input file submitted by user and create invoices for indivual customers  
Applications used are Outlook, Excel, Chrome browser.  
1 Get the next transaction to be processed. Get the Next DO for processing  
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.  
Processing will be done and output will be stored in output excel for respective Daily, Consolidated and Monthly scenario.  
4 Start Processing of transactions  
5 Read configuration file and initialize applications used in the process. This will check if all the folders, Shared drives are accessible along with reading Config file.   
6 An uninitialized Config dictionary indicates that it is the first run of the process.  
7 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.  
8 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.  
9 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.  
10 Failures during the initialization are considered system exceptions and lead to the End Process state, thus finalizing the execution.  
11 End process and close all applications used.  
12 All the applications like outlook, SDMS web portal, Chrome will be closed in this .xaml  
13 Business Rule Exception  
14 There is no need for any action in case of successful transaction.  
The process should simply go to next transaction.  
15 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.  
16 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.   
17 Used during transitions between states to represent exceptions other than business exceptions.  
18 Used during transitions between states and represents a situation that does not conform to the rules of the process being automated.  
19 Sequential counter of transaction items.  
20 Dictionary structure to store configuration data of the process (settings, constants and assets).  
21 Used to control the number of attempts of retrying the transaction processing in case of system exceptions.  
22 Optionally used to include additional information about the transaction item.  
23 Optionally used to include additional information about the transaction item.  
24 Transaction ID used for information and logging purposes. Ideally, the ID should be unique for each transaction.   
25 Used in case transactions are stored in a DataTable, for example, after being retrieved from a spreadsheet.  
26 Used to control the number of consecutive system exceptions.

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

8.2 CreateMasterFileFolders.xaml

This .xaml will check if the input file is available for Consolidated process on the network drive or input folder specified.

***Location: \ExtractInput\CheckInputFileConso.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_Config1 | InArgument(scg:Dictionary(x:String, x:Object)) |  |
| io\_ProcessCounter1 | InOutArgument(x:Int32) |  |
| in\_ProcessType | InArgument(x:String) |  |
| out\_dt\_TransactionData1 | OutArgument(sd:DataTable) |  |
| out\_IOExcelFilePath | OutArgument(x:String) |  |

8.3 CreateMasterFileFolders.xaml

This .xaml will check if the input file is available for Daily process on the network drive or input folder specified.

***Location: \ExtractInput\CheckInputFileDaily.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_Config1 | InArgument(scg:Dictionary(x:String, x:Object)) |  |
| io\_ProcessCounter1 | InOutArgument(x:Int32) |  |
| in\_ProcessType | InArgument(x:String) |  |
| out\_InputOutputFilePath | OutArgument(x:String) |  |
| out\_dt\_TransactionData1 | OutArgument(sd:DataTable) |  |

8.4 CreateMasterFileFolders.xaml

This .xaml will check if the input file is available for Monthly process on the network drive or input folder specified.

***Location: \ExtractInput\CheckInputFileMonthly.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_Config1 | InArgument(scg:Dictionary(x:String, x:Object)) |  |
| io\_ProcessCounter1 | InOutArgument(x:Int32) |  |
| in\_ProcessType | InArgument(x:String) |  |
| out\_dt\_TransactionData1 | OutArgument(sd:DataTable) |  |
| out\_IOExcelFilePath | OutArgument(x:String) |  |

8.5 CloseAllApplications.xaml

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

***Location: \Framework\CloseAllApplications.xaml***

***Parameters:***

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

8.6 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.

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

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_Config | InArgument(scg:Dictionary(x:String, x:Object)) | Dictionary structure to store configuration data of the process (settings, constants and assets). |
| out\_TransactionField1 | OutArgument(x:String) | Allow the optional addition of information about the transaction item. |
| out\_TransactionField2 | OutArgument(x:String) | Allow the optional addition of information about the transaction item. |
| out\_TransactionID | OutArgument(x:String) | Transaction ID used for information and logging purposes. Ideally, the ID should be unique for each transaction. |
| 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). |
| io\_ProcessType | InOutArgument(x:String) |  |
| io\_ProcessCounter | InOutArgument(x:Int32) |  |
| io\_TransactionNumber | InOutArgument(x:Int32) | Sequential counter of transaction items. |
| io\_IOExcelFilePathDaily | InOutArgument(x:String) |  |
| io\_IOExcelFilePathConso | InOutArgument(x:String) |  |
| io\_IOExcelFilePathMonthly | InOutArgument(x:String) |  |
| io\_ProcessedRowCount | InOutArgument(x:Int32) |  |
| io\_FilterFlag | InOutArgument(x:String) |  |
| io\_UniqueColCount | InOutArgument(x:Int32) |  |
| out\_dt\_TransactionItem | OutArgument(sd:DataTable) | Transaction item to be processed. |
| io\_dt\_FilteredDT | InOutArgument(sd:DataTable) |  |
| io\_dt\_ScheduleDetailsDT | InOutArgument(sd:DataTable) |  |

8.7 Initialize\_Applications.xaml

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

***Location: \Framework\InitAllApplications.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_Config | InArgument(scg:Dictionary(x:String, x:Object)) | Dictionary structure to store configuration data of the process (settings, constants and assets). |
| io\_ScreenshotFilePath | InOutArgument(x:String) |  |
| out\_dt\_BillingEntityDT1 | OutArgument(sd:DataTable) |  |
| out\_dt\_PublicHolidayDT1 | OutArgument(sd:DataTable) |  |
| out\_dt\_VehicleTypeMapDT1 | OutArgument(sd:DataTable) |  |

8.8 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.  
2 Read non-empty rows in the sheet.  
3 Get from Orchestrator the values of assets listed in the Assets sheet.  
4 Logs a message and throw exception in case the asset is specified in the Config file, but it could not be loaded from Orchestrator.

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

***Parameters:***

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

8.9 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: \Framework\KillAllProcesses.xaml***

***Parameters:***

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

8.10 Process.xaml

This sequence wll enter all the data into quickbooks application for all Daily, Consolidated and Monthyl scenarios. Quickbooks will generate the invoices and the PDF files will be merged.  
1 Processing steps for Daily process. Calculations are done in this step and entering the data to quickbooks

***Location: \Framework\Process.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_Config | InArgument(scg:Dictionary(x:String, x:Object)) | Dictionary structure to store configuration data of the process (settings, constants and assets). |
| io\_ProcessType | InOutArgument(x:String) |  |
| io\_TransactionNumber | InOutArgument(x:Int32) |  |
| io\_IOExcelFilePathDaily | InOutArgument(x:String) |  |
| io\_IOExcelFilePathConso | InOutArgument(x:String) |  |
| io\_IOExcelFilePathMonthly | InOutArgument(x:String) |  |
| out\_InvoiceNumber | OutArgument(x:String) |  |
| out\_InvoiceFilePath | OutArgument(x:String) |  |
| in\_dt\_TransactionItem | InArgument(sd:DataTable) | Transaction item to be processed. |
| io\_dt\_VehicleTypeMap | InOutArgument(sd:DataTable) |  |
| in\_dt\_Sundays | InArgument(sd:DataTable) |  |
| in\_dt\_PublicHolidays | InArgument(sd:DataTable) |  |
| in\_dt\_BillingEntity | InArgument(sd:DataTable) |  |

8.11 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, 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. |
| io\_UniqueColCount1 | InOutArgument(x:Int32) |  |
| io\_ProcessedRowCount1 | InOutArgument(x:Int32) |  |
| in\_dt\_TransactionItem | InArgument(sd:DataTable) |  |
| io\_FilterFlag | InOutArgument(x:String) |  |

8.12 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\_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, 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. |
| io\_ConsecutiveSystemExceptions | InOutArgument(x:Int32) | Used to control the number of consecutive system exceptions. |
| io\_ProcessType | InOutArgument(x:String) |  |
| io\_ScreenshotFilePath | InOutArgument(x:String) |  |
| io\_IOExcelFileDaily | InOutArgument(x:String) |  |
| io\_IOExcelFileConsolidated | InOutArgument(x:String) |  |
| io\_IOExcelFileMonthly | InOutArgument(x:String) |  |
| in\_InvoiceFilePath | InArgument(x:String) |  |
| in\_InvoiceNumber | InArgument(x:String) |  |
| io\_ProcessedRowCount | InOutArgument(x:Int32) |  |
| io\_FilterFlag | InOutArgument(x:String) |  |
| io\_UniqueColCount | InOutArgument(x:Int32) |  |
| in\_dt\_TransactionItem1 | InArgument(sd:DataTable) |  |
| io\_dt\_TransactionData | InOutArgument(sd:DataTable) |  |

8.13 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 | InOutArgument(x:String) | Optional argument that specifies the path and the name of the screenshot to be taken. |

8.14 CalculateWorkingDays.xaml

***Location: \Quickbooks\CalculateWorkingDaysMonthly.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_DateString | InArgument(x:String) |  |
| out\_NoOfWorkingDays | OutArgument(x:Int32) |  |
| out\_InputMonth | OutArgument(x:Int32) |  |

8.15 DailyProcessSequence.xaml

Calculate the number of hours worked and detect if driver has worked on Sundays or public holidays

***Location: \Quickbooks\DailyCalculations.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_Config1 | InArgument(scg:Dictionary(x:String, x:Object)) |  |
| io\_TransactionNumber1 | InOutArgument(x:Int32) |  |
| in\_TransactionItem1 | InArgument(sd:DataRow) |  |
| out\_TotalStdWorkingHours | OutArgument(x:Double) |  |
| out\_TotalPremWorkingHours | OutArgument(x:Double) |  |
| in\_QuotationFilePath1 | InArgument(x:String) |  |
| in\_dt\_Sundays | InArgument(sd:DataTable) |  |
| in\_dt\_PublicHolidays | InArgument(sd:DataTable) |  |
| in\_DailyQuotationSheetFlag | InArgument(x:Boolean) |  |

8.16 FetchQuotationDetails.xaml

***Location: \Quickbooks\FetchQuotationDetailsDaily.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_DailySheetPresent | InArgument(x:Boolean) |  |
| in\_QuotationFilePath | InArgument(x:String) |  |
| in\_Config1 | InArgument(scg:Dictionary(x:String, x:Object)) |  |
| out\_DailyQuotationSheetFlag | OutArgument(x:Boolean) |  |
| in\_dt\_Quotation | InArgument(sd:DataTable) |  |
| out\_AddressOfJobType | OutArgument(x:String) |  |
| out\_MinWorkingHours | OutArgument(x:Double) |  |
| in\_dt\_VehicleTypeMap1 | InArgument(sd:DataTable) |  |
| out\_VehicleTypeQkbk | OutArgument(x:String) |  |
| out\_StdRate | OutArgument(ui:GenericValue) |  |
| out\_PremRate | OutArgument(ui:GenericValue) |  |
| out\_VehicleTypeDescription | OutArgument(x:String) |  |
| in\_CurrentRow | InArgument(sd:DataRow) |  |

8.17 FindBillingEntity.xaml

Find out the billing entities for the current Company or PO

***Location: \Quickbooks\FindBillingEntity.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| out\_BillingEntity | OutArgument(x:String) |  |
| in\_DefaultBusinessEntity | InArgument(x:String) |  |
| io\_IOExcelFilePath | InOutArgument(x:String) |  |
| in\_ProcessType | InArgument(x:String) |  |
| in\_dt\_BillingEntity1 | InArgument(sd:DataTable) |  |
| in\_dt\_TransactionItem | InArgument(sd:DataTable) |  |
| in\_TimeoutShort | InArgument(x:Int32) |  |
| in\_HeavyEquipmentPath | InArgument(x:String) |  |
| in\_Config1 | InArgument(scg:Dictionary(x:String, x:Object)) |  |
| in\_DefaultEntityPath | InArgument(x:String) |  |

8.18 MergeFiles.xaml

This sequence will merge all the PDF files like Quotation file, DO/Timesheet file, Invoice file and PO file(if exists)

***Location: \Quickbooks\MergeFiles.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_InvoiceNumber | InArgument(x:String) |  |
| in\_InvoiceFilePath | InArgument(x:String) |  |
| in\_InvoicePath | InArgument(x:String) |  |
| in\_Config1 | InArgument(scg:Dictionary(x:String, x:Object)) |  |
| io\_ProcessType2 | InOutArgument(x:String) |  |
| in\_dt\_TransactionItem1 | InArgument(sd:DataTable) |  |

8.19 MergeFiles.xaml

This sequence will merge all the PDF files like Quotation file, DO/Timesheet file, Invoice file and PO file(if exists)

***Location: \Quickbooks\MergeFilesMonthly.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_InvoiceNumber | InArgument(x:String) |  |
| in\_InvoiceFilePath | InArgument(x:String) |  |
| in\_InvoicePath | InArgument(x:String) |  |
| in\_Config1 | InArgument(scg:Dictionary(x:String, x:Object)) |  |
| io\_ProcessType2 | InOutArgument(x:String) |  |
| in\_dt\_TransactionItem1 | InArgument(sd:DataTable) |  |

8.20 DailyProcessSequence.xaml

Calculate the number of hours worked and detect if driver has worked on Sundays or public holidays

***Location: \Quickbooks\MonthlyCalculations.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_Config1 | InArgument(scg:Dictionary(x:String, x:Object)) |  |
| io\_TransactionNumber1 | InOutArgument(x:Int32) |  |
| in\_TransactionItem1 | InArgument(sd:DataRow) |  |
| out\_TotalStdWorkingHours | OutArgument(x:Double) |  |
| out\_TotalPremWorkingHours | OutArgument(x:Double) |  |
| in\_QuotationFilePath1 | InArgument(x:String) |  |
| out\_TotalOTHours | OutArgument(x:Double) |  |
| in\_MinWorkingHrs | InArgument(x:Double) |  |
| out\_NoOfDaysWorked | OutArgument(x:Int32) |  |
| in\_dt\_Sundays | InArgument(sd:DataTable) |  |
| in\_dt\_PublicHolidays | InArgument(sd:DataTable) |  |

8.21 QuickbooksLogin.xaml

Login to Quickbooks to create invoices

***Location: \Quickbooks\QuickbooksLogin.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_QuickbooksExePath | InArgument(x:String) |  |
| in\_RecipientsEmailList | InArgument(x:String) |  |
| in\_CurrentBillingEntity | InArgument(x:String) |  |
| in\_Config1 | InArgument(scg:Dictionary(x:String, x:Object)) |  |

8.22 QuickbooksLogout.xaml

This .xaml will logout system from Quickbooks.

***Location: \Quickbooks\QuickbooksLogout.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_Config1 | InArgument(scg:Dictionary(x:String, x:Object)) |  |

8.23 Sequence.xaml

This will create invoice in the quickbooks as per Daily and Consolidated scenario. It will also calculate the total number of standard and premium working hours.

***Location: \Quickbooks\QuickbooksProcess.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_DelayAfterShort | InArgument(x:Int32) |  |
| in\_Config1 | InArgument(scg:Dictionary(x:String, x:Object)) |  |
| in\_QuotationFilePath | InArgument(x:String) |  |
| io\_TransactionNumber1 | InOutArgument(x:Int32) |  |
| in\_InvoicePdfFolderPath | InArgument(x:String) |  |
| io\_ProcessType1 | InOutArgument(x:String) |  |
| out\_InvoiceFilePath1 | OutArgument(x:String) |  |
| out\_InvoiceNumber1 | OutArgument(x:String) |  |
| in\_dt\_Sundays1 | InArgument(sd:DataTable) |  |
| in\_dt\_PublicHolidays1 | InArgument(sd:DataTable) |  |
| in\_dt\_VehicleTypeMap1 | InArgument(sd:DataTable) |  |
| in\_dt\_TransactionItem1 | InArgument(sd:DataTable) |  |

8.24 Sequence.xaml

This .xaml will create invoices for   
Monthly process. It will also calculate the total standard working hours and working hours on Weekends.

***Location: \Quickbooks\QuickbooksProcessMonthly.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_DelayAfterShort | InArgument(x:Int32) |  |
| in\_Config1 | InArgument(scg:Dictionary(x:String, x:Object)) |  |
| in\_QuotationFilePath | InArgument(x:String) |  |
| io\_TransactionNumber1 | InOutArgument(x:Int32) |  |
| in\_InvoicePdfFolderPath | InArgument(x:String) |  |
| out\_InvoiceNumber1 | OutArgument(x:String) |  |
| out\_InvoiceFilePath1 | OutArgument(x:String) |  |
| io\_ProcessType1 | InOutArgument(x:String) |  |
| in\_dt\_VehicleTypeMap | InArgument(sd:DataTable) |  |
| in\_dt\_TransactionItem1 | InArgument(sd:DataTable) |  |
| in\_dt\_PublicHolidays | InArgument(sd:DataTable) |  |
| in\_dt\_Sundays | InArgument(sd:DataTable) |  |
| in\_BillingEntity | InArgument(x:String) |  |
| in\_BillingEntityForHE | InArgument(x:String) |  |

8.25 SetStatusDaily.xaml

***Location: \Quickbooks\SetStatusDaily.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_FilterFlag | InArgument(x:String) |  |
| in\_IOExcelFileDaily | InArgument(x:String) |  |
| in\_InvoiceFilePath1 | InArgument(x:String) |  |
| in\_InvoiceNumber1 | InArgument(x:String) |  |
| in\_Config1 | InArgument(scg:Dictionary(x:String, x:Object)) |  |
| in\_Status | InArgument(x:String) |  |
| in\_Comments | InArgument(x:String) |  |
| io\_dt\_TransactionData1 | InOutArgument(sd:DataTable) |  |
| in\_dt\_TransactionItem | InArgument(sd:DataTable) |  |

8.26 SetStatusDaily.xaml

***Location: \Quickbooks\SetStatusDailyBESE.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_FilterFlag | InArgument(x:String) |  |
| in\_IOExcelFileDaily | InArgument(x:String) |  |
| in\_Config1 | InArgument(scg:Dictionary(x:String, x:Object)) |  |
| in\_Status | InArgument(x:String) |  |
| in\_Comments | InArgument(x:String) |  |
| io\_dt\_TransactionData1 | InOutArgument(sd:DataTable) |  |
| in\_dt\_TransactionItem | InArgument(sd:DataTable) |  |

8.27 CheckAllColumnsMasterFile.xaml

Check if all required columns are available in the master file.

***Location: \Reusable Code\CheckAllColumnsMasterFile.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_Config1 | InArgument(scg:Dictionary(x:String, x:Object)) |  |
| out\_dt\_BillingEntity | OutArgument(sd:DataTable) |  |
| out\_dt\_PublicHoliday | OutArgument(sd:DataTable) |  |
| out\_dt\_VehicleTypeMap | OutArgument(sd:DataTable) |  |

8.28 CheckFileIsDownloaded.xaml

***Location: \Reusable Code\CheckFileIsDownloaded.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_dt\_TransactionItem1 | InArgument(sd:DataTable) |  |
| in\_FilePath | InArgument(x:String) |  |
| in\_Config1 | InArgument(scg:Dictionary(x:String, x:Object)) |  |

8.29 CheckInputDataAvailable.xaml

***Location: \Reusable Code\CheckInputDataAvailable.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| io\_ProcessType1 | InOutArgument(x:String) |  |
| in\_Config1 | InArgument(scg:Dictionary(x:String, x:Object)) |  |
| in\_dt\_TransactionItem1 | InArgument(sd:DataTable) |  |

8.30 GetQuotationFile.xaml

Check if the Required quotation file is exist or not

***Location: \Reusable Code\CheckQuotationFile.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_Config1 | InArgument(scg:Dictionary(x:String, x:Object)) |  |
| out\_QuotationFilePath | OutArgument(x:String) |  |
| io\_ProcessType1 | InOutArgument(x:String) |  |
| in\_dt\_TransactionItem1 | InArgument(sd:DataTable) |  |

8.31 GetQuotationFile.xaml

Check if the Required quotation file is exist or not

***Location: \Reusable Code\CheckQuotationFileMonthly.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_Config1 | InArgument(scg:Dictionary(x:String, x:Object)) |  |
| out\_QuotationFilePath | OutArgument(x:String) |  |
| io\_ProcessType1 | InOutArgument(x:String) |  |
| in\_dt\_TransactionItem1 | InArgument(sd:DataTable) |  |

8.32 CleanScreenshotFolder.xaml

***Location: \Reusable Code\CleanScreenshotFolder.xaml***

***Parameters:***

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

8.33 CreateDOFolder.xaml

Create Invoice folder paths for saving invoices

***Location: \Reusable Code\CreateInvoicePath.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_ProcessType | InArgument(x:String) |  |
| in\_InvoicePdfPath | InArgument(x:String) |  |
| out\_InvoicePdfFolderPath | OutArgument(x:String) |  |

8.34 NetworkFolderFileAccessible.xaml

This .xaml will check if the requited file placed on Network drive is accessible or not.

***Location: \Reusable Code\NetworkFileAccessible.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_NetworkFileFolderPath | InArgument(x:String) |  |
| in\_RetryNumber | InArgument(x:Int32) |  |
| in\_Word | InArgument(x:String) |  |
| in\_RetryIntervalMedium | InArgument(x:TimeSpan) |  |

8.35 NetworkFolderFileAccessible.xaml

This .xaml will check if all the required network folders are accessible for the process to run.

***Location: \Reusable Code\NetworkFolderAccessible.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_NetworkFileFolderPath | InArgument(x:String) |  |
| in\_RetryNumber | InArgument(x:Int32) |  |
| in\_Word | InArgument(x:String) |  |
| in\_RetryIntervalMedium | InArgument(x:TimeSpan) |  |

8.36 SendProcessNotificationAndStatus.xaml

This .xaml file Sends process start notification to the required recipients  
1 Try catch added for sending process end notifiation and status   
2 Sends exception email to the reqired recipients.  
Below are the arguments which are used:  
1. in\_RecipientsEmailList = Specify list of recipients.  
2. in\_Subject = Specify email subject.  
3. in\_Body = Specify email body.  
4. in\_AccountName = Specify account name from which email should be send.  
5. in\_ExceptionAttachmentPath = Specify exception screenshot path which needs to be attached to exception email

***Location: \Reusable Code\ProcessStartNotification.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_RecipientsEmailList | InArgument(x:String) |  |
| in\_Subject | InArgument(x:String) |  |
| in\_Body | InArgument(x:String) |  |
| in\_CcEmailList | InArgument(x:String) |  |

8.37 SendExceptionEmail.xaml

Sends exception email to required recipients  
1 Try catch added for sending exception  
2 Sends exception email to the reqired recipients.  
Below are the arguments which are used:  
1. in\_RecipientsEmailList = Specify list of recipients.  
2. in\_Subject = Specify email subject.  
3. in\_Body = Specify email body.  
4. in\_AccountName = Specify account name from which email should be send.  
5. in\_ExceptionAttachmentPath = Specify exception screenshot path which needs to be attached to exception email  
6.

***Location: \Reusable Code\SendExceptionEmail.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_Subject | InArgument(x:String) |  |
| in\_Body | InArgument(x:String) |  |
| in\_CcEmailList | InArgument(x:String) |  |
| in\_RecipientsEmailList1 | InArgument(x:String) |  |
| in\_ExScreenshotsFolderPath | InArgument(x:String) |  |
| in\_ExceptionOccured | InArgument(x:String) |  |
| io\_ProcessType1 | InOutArgument(x:String) |  |

8.38 SendExceptionEmail.xaml

Sends exception email for Config file error to required recipients  
1 Try catch added for sending exception  
2 Sends exception email to the reqired recipients.  
Below are the arguments which are used:  
1. in\_RecipientsEmailList = Specify list of recipients.  
2. in\_Subject = Specify email subject.  
3. in\_Body = Specify email body.  
4. in\_AccountName = Specify account name from which email should be send.  
5. in\_ExceptionAttachmentPath = Specify exception screenshot path which needs to be attached to exception email  
6.

***Location: \Reusable Code\SendExceptionEmailConfig.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_Subject | InArgument(x:String) |  |
| in\_Body | InArgument(x:String) |  |
| in\_CcEmailList | InArgument(x:String) |  |
| in\_RecipientsEmailList1 | InArgument(x:String) |  |

8.39 SendExceptionEmail.xaml

Sends No data exception email to required recipients  
1 Try catch added for sending exception  
2 Sends exception email to the reqired recipients.  
Below are the arguments which are used:  
1. in\_RecipientsEmailList = Specify list of recipients.  
2. in\_Subject = Specify email subject.  
3. in\_Body = Specify email body.  
4. in\_AccountName = Specify account name from which email should be send.  
5. in\_ExceptionAttachmentPath = Specify exception screenshot path which needs to be attached to exception email  
6.

***Location: \Reusable Code\SendExceptionEmailNoData.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_Subject | InArgument(x:String) |  |
| in\_Body | InArgument(x:String) |  |
| in\_CcEmailList | InArgument(x:String) |  |
| in\_RecipientsEmailList1 | InArgument(x:String) |  |
| io\_ProcessType1 | InOutArgument(x:String) |  |

8.40 SendExceptionEmail.xaml

Sends exception email to required recipients  
1 Try catch added for sending exception  
2 Sends exception email to the reqired recipients.  
Below are the arguments which are used:  
1. in\_RecipientsEmailList = Specify list of recipients.  
2. in\_Subject = Specify email subject.  
3. in\_Body = Specify email body.  
4. in\_AccountName = Specify account name from which email should be send.  
5. in\_ExceptionAttachmentPath = Specify exception screenshot path which needs to be attached to exception email  
6.

***Location: \Reusable Code\SendExceptionEmailNoSS.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_Subject | InArgument(x:String) |  |
| in\_Body | InArgument(x:String) |  |
| in\_CcEmailList | InArgument(x:String) |  |
| in\_RecipientsEmailList1 | InArgument(x:String) |  |
| in\_ExceptionOccured | InArgument(x:String) |  |
| io\_ProcessType1 | InOutArgument(x:String) |  |

8.41 SendReport.xaml

This .xaml file Sends process end notification and status report to the required recipients  
1 Try catch added for sending process end notifiation and status for Daily process  
2 Sends exception email to the reqired recipients.  
Below are the arguments which are used:  
1. in\_RecipientsEmailList = Specify list of recipients.  
2. in\_Subject = Specify email subject.  
3. in\_Body = Specify email body.  
4. in\_AccountName = Specify account name from which email should be send.  
5. in\_ExceptionAttachmentPath = Specify exception screenshot path which needs to be attached to exception email

***Location: \Reusable Code\SendReport.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_ReportBody | InArgument(x:String) |  |
| in\_ReportSubject | InArgument(x:String) |  |
| in\_RecipientsEmailList | InArgument(x:String) |  |
| in\_CCEmailList | InArgument(x:String) |  |
| io\_OutputExcelFile | InOutArgument(x:String) |  |
| io\_ProcessType1 | InOutArgument(x:String) |  |

8.42 SetStatusConsolidated.xaml

***Location: \Reusable Code\SetStatusConsolidated.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| in\_Config | InArgument(scg:Dictionary(x:String, x:Object)) |  |
| in\_dt\_TransactionItem1 | InArgument(sd:DataTable) |  |
| in\_InvoiceFilePath | InArgument(x:String) |  |
| in\_InvoiceNumber | InArgument(x:String) |  |
| in\_Comments | InArgument(x:String) |  |
| in\_Status | InArgument(x:String) |  |
| io\_ProcessType | InOutArgument(x:String) |  |
| io\_IOExcelFileConso | InOutArgument(x:String) |  |

8.43 SetStatusMonthly.xaml

***Location: \Reusable Code\SetStatusMonthly.xaml***

***Parameters:***

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| io\_IOExcelFileMonthly | InOutArgument(x:String) |  |
| in\_Config | InArgument(scg:Dictionary(x:String, x:Object)) |  |
| in\_InvoiceFilePath | InArgument(x:String) |  |
| in\_InvoiceNumber | InArgument(x:String) |  |
| in\_Status | InArgument(x:String) |  |
| in\_Comments | InArgument(x:String) |  |
| in\_dt\_TransactionItem1 | InArgument(sd:DataTable) |  |
| io\_ProcessType | InOutArgument(x:String) |  |

8.44 GeneralTestCase.xaml

GeneralTestCase.xaml is a data driven test case based on the Tests.xlsx, Tests sheet where the developer will write the workflow paths of the workflows to be tested and the expected exception - AppEx, BRE or Success. Running GeneralTestCase with data variations is going to be a clear result of comparing the expected result with the actual result after passing through this list of workflows as test data.   
There are two possible statuses - PASS or FAIL for each workflow that was run. The status is PASS if the actual exception caught is the one previously defined in the Tests sheet and FAIL otherwise.

***Location: \Tests\GeneralTestCase.xaml***

***Parameters:***

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

8.45 GetTransactionDataTestCase.xaml

Given the TransactionNumber, verify if GetTransactionData workflow works as expected.  
Once a Transaction Item has been processed, its status will be In Progress.  
Queue name should be configured in order for the test case to run.  
1 Note to also change the transaction status

***Location: \Tests\GetTransactionDataTestCase.xaml***

8.46 InitAllApplicationsTestCase.xaml

Verify if the InitAllApplications workflow works as expected.  
The verification should check if after opening the applications, the expected state is reached.  
1 Check if the applications are in the expected state

***Location: \Tests\InitAllApplicationsTestCase.xaml***

8.47 InitAllSettingsTestCase.xaml

Verify if the InitAllSettings workflow works as expected.  
The verification should check if initalization of settings was successful: if the Config dictionary was created, if it contains a certain key etc.  
1 Check if the Config contains key-value pairs  
Check if the Config contains certain key information

***Location: \Tests\InitAllSettingsTestCase.xaml***

8.48 MainTestCase.xaml

Verify if the Main workflow works as expected.  
The verification should check whether the status file or report built after the process run is the expected one.  
1 Please add the path to the status file  
2 Please add the path to the expected status file

***Location: \Tests\MainTestCase.xaml***

8.49 ProcessTestCase.xaml

Verify if the Process workflow works as expected.  
The verification should check whether the output of the Process workflow is the expected one.

***Location: \Tests\ProcessTestCase.xaml***

8.50 TestWorkflowTemplate.xaml

Template workflow used to create tests for workflows in the process.  
Create a new test workflow by copying and renaming this file.  
1 Dictionary structure to store configuration data.  
2 Verification to pe performed

***Location: \Tests\WorkflowTestCaseTemplate.xaml***

# **9 Workflows**

|  |  |  |
| --- | --- | --- |
| Name | Invoked Workflow | Invoked In |
| \Main.xaml | •Framework\GetTransactionData.xaml •Reusable Code\SendExceptionEmailNoSS.xaml •Framework\Process.xaml •Framework\SetTransactionStatus.xaml •Reusable Code\SendExceptionEmail.xaml •Reusable Code\SendExceptionEmailConfig.xaml •Framework\InitAllSettings.xaml •Framework\KillAllProcesses.xaml •Framework\InitAllApplications.xaml •Reusable Code\CleanScreenshotFolder.xaml •Framework\CloseAllApplications.xaml | •Tests\MainTestCase.xaml |
| \ExtractInput\CheckInputFileConso.xaml | •Reusable Code\SendExceptionEmailNoData.xaml | •Framework\GetTransactionData.xaml |
| \ExtractInput\CheckInputFileDaily.xaml | •Reusable Code\SendExceptionEmailNoData.xaml | •Framework\GetTransactionData.xaml |
| \ExtractInput\CheckInputFileMonthly.xaml | •Reusable Code\SendExceptionEmailNoData.xaml | •Framework\GetTransactionData.xaml |
| \Framework\CloseAllApplications.xaml | •Quickbooks\QuickbooksLogout.xaml •Reusable Code\SendExceptionEmail.xaml | •Main.xaml •Framework\SetTransactionStatus.xaml •Tests\InitAllApplicationsTestCase.xaml •Tests\ProcessTestCase.xaml |
| \Framework\GetTransactionData.xaml | •Reusable Code\SendReport.xaml •ExtractInput\CheckInputFileDaily.xaml •ExtractInput\CheckInputFileConso.xaml •ExtractInput\CheckInputFileMonthly.xaml | •Main.xaml •Tests\ProcessTestCase.xaml |
| \Framework\InitAllApplications.xaml | •Reusable Code\NetworkFolderAccessible.xaml •Reusable Code\NetworkFileAccessible.xaml •Reusable Code\CheckAllColumnsMasterFile.xaml •Quickbooks\QuickbooksLogin.xaml •Reusable Code\SendExceptionEmail.xaml •Reusable Code\ProcessStartNotification.xaml •Reusable Code\SendExceptionEmailNoSS.xaml | •Main.xaml •Tests\ProcessTestCase.xaml |
| \Framework\InitAllSettings.xaml |  | •Main.xaml •Tests\GetTransactionDataTestCase.xaml •Tests\InitAllApplicationsTestCase.xaml •Tests\ProcessTestCase.xaml •Tests\WorkflowTestCaseTemplate.xaml |
| \Framework\KillAllProcesses.xaml | •Quickbooks\QuickbooksLogout.xaml •Reusable Code\SendExceptionEmail.xaml | •Main.xaml •Framework\SetTransactionStatus.xaml |
| \Framework\Process.xaml | •Reusable Code\CheckInputDataAvailable.xaml •Reusable Code\CreateInvoicePath.xaml •Reusable Code\CheckQuotationFile.xaml •Quickbooks\QuickbooksProcess.xaml •Quickbooks\MergeFiles.xaml •Quickbooks\FindBillingEntity.xaml •Reusable Code\CheckQuotationFileMonthly.xaml •Quickbooks\QuickbooksProcessMonthly.xaml •Quickbooks\MergeFilesMonthly.xaml | •Main.xaml •Tests\ProcessTestCase.xaml |
| \Framework\RetryCurrentTransaction.xaml |  |  |
| \Framework\SetTransactionStatus.xaml | •Quickbooks\SetStatusDaily.xaml •Reusable Code\SetStatusConsolidated.xaml •Reusable Code\SetStatusMonthly.xaml •Framework\TakeScreenshot.xaml •Framework/RetryCurrentTransaction.xaml •Framework\CloseAllApplications.xaml •Framework\KillAllProcesses.xaml | •Main.xaml |
| \Framework\TakeScreenshot.xaml |  | •Framework\SetTransactionStatus.xaml •Reusable Code\SendExceptionEmail.xaml |
| \Quickbooks\CalculateWorkingDaysMonthly.xaml |  | •Quickbooks\QuickbooksProcessMonthly.xaml |
| \Quickbooks\DailyCalculations.xaml |  | •Quickbooks\QuickbooksProcess.xaml |
| \Quickbooks\FetchQuotationDetailsDaily.xaml |  | •Quickbooks\QuickbooksProcess.xaml |
| \Quickbooks\FindBillingEntity.xaml |  | •Framework\Process.xaml |
| \Quickbooks\MergeFiles.xaml | •Reusable Code\SendExceptionEmail.xaml | •Framework\Process.xaml |
| \Quickbooks\MergeFilesMonthly.xaml | •Reusable Code\SendExceptionEmail.xaml | •Framework\Process.xaml |
| \Quickbooks\MonthlyCalculations.xaml |  | •Quickbooks\QuickbooksProcessMonthly.xaml |
| \Quickbooks\QuickbooksLogin.xaml |  | •Framework\InitAllApplications.xaml |
| \Quickbooks\QuickbooksLogout.xaml |  | •Framework\CloseAllApplications.xaml •Framework\KillAllProcesses.xaml |
| \Quickbooks\QuickbooksProcess.xaml | •Reusable Code\SendExceptionEmail.xaml •Quickbooks\FetchQuotationDetailsDaily.xaml •Quickbooks\DailyCalculations.xaml •Reusable Code\CheckFileIsDownloaded.xaml | •Framework\Process.xaml |
| \Quickbooks\QuickbooksProcessMonthly.xaml | •Quickbooks\MonthlyCalculations.xaml •Quickbooks\CalculateWorkingDaysMonthly.xaml •Reusable Code\SendExceptionEmailNoSS.xaml | •Framework\Process.xaml |
| \Quickbooks\SetStatusDaily.xaml |  | •Framework\SetTransactionStatus.xaml |
| \Quickbooks\SetStatusDailyBESE.xaml |  |  |
| \Reusable Code\CheckAllColumnsMasterFile.xaml |  | •Framework\InitAllApplications.xaml |
| \Reusable Code\CheckFileIsDownloaded.xaml |  | •Quickbooks\QuickbooksProcess.xaml |
| \Reusable Code\CheckInputDataAvailable.xaml | •Reusable Code\SendExceptionEmail.xaml | •Framework\Process.xaml |
| \Reusable Code\CheckQuotationFile.xaml |  | •Framework\Process.xaml |
| \Reusable Code\CheckQuotationFileMonthly.xaml |  | •Framework\Process.xaml |
| \Reusable Code\CleanScreenshotFolder.xaml |  | •Main.xaml |
| \Reusable Code\CreateInvoicePath.xaml |  | •Framework\Process.xaml |
| \Reusable Code\NetworkFileAccessible.xaml |  | •Framework\InitAllApplications.xaml |
| \Reusable Code\NetworkFolderAccessible.xaml |  | •Framework\InitAllApplications.xaml |
| \Reusable Code\ProcessStartNotification.xaml |  | •Framework\InitAllApplications.xaml |
| \Reusable Code\SendExceptionEmail.xaml | •Framework\TakeScreenshot.xaml | •Main.xaml •Framework\CloseAllApplications.xaml •Framework\InitAllApplications.xaml •Framework\KillAllProcesses.xaml •Quickbooks\MergeFiles.xaml •Quickbooks\MergeFilesMonthly.xaml •Quickbooks\QuickbooksProcess.xaml •Reusable Code\CheckInputDataAvailable.xaml |
| \Reusable Code\SendExceptionEmailConfig.xaml |  | •Main.xaml |
| \Reusable Code\SendExceptionEmailNoData.xaml |  | •ExtractInput\CheckInputFileConso.xaml •ExtractInput\CheckInputFileDaily.xaml •ExtractInput\CheckInputFileMonthly.xaml |
| \Reusable Code\SendExceptionEmailNoSS.xaml |  | •Main.xaml •Framework\InitAllApplications.xaml •Quickbooks\QuickbooksProcessMonthly.xaml |
| \Reusable Code\SendReport.xaml |  | •Framework\GetTransactionData.xaml |
| \Reusable Code\SetStatusConsolidated.xaml |  | •Framework\SetTransactionStatus.xaml |
| \Reusable Code\SetStatusMonthly.xaml |  | •Framework\SetTransactionStatus.xaml |
| \Tests\GeneralTestCase.xaml | •[in\_WorkflowFile] |  |
| \Tests\GetTransactionDataTestCase.xaml | •Framework\InitAllSettings.xaml •Framework/GetTransactionData.xaml |  |
| \Tests\InitAllApplicationsTestCase.xaml | •Framework\InitAllSettings.xaml •Framework/InitAllApplications.xaml •Framework\CloseAllApplications.xaml |  |
| \Tests\InitAllSettingsTestCase.xaml | •Framework/InitAllSettings.xaml |  |
| \Tests\MainTestCase.xaml | •Main.xaml |  |
| \Tests\ProcessTestCase.xaml | •Framework\InitAllSettings.xaml •Framework\InitAllApplications.xaml •Framework\GetTransactionData.xaml •Framework\Process.xaml •Framework\CloseAllApplications.xaml |  |
| \Tests\WorkflowTestCaseTemplate.xaml | •Framework\InitAllSettings.xaml |  |

# **10 Dependencies**

• UiPath.Excel.Activities: [2.16.1]  
• UiPath.Mail.Activities: [1.18.2]  
• UiPath.PDF.Activities: [3.11.0]  
• UiPath.System.Activities: [22.4.5]  
• UiPath.Testing.Activities: [22.4.3]  
• UiPath.UIAutomation.Activities: [22.4.7]  
• UiPathTeam.Documents.TDDGenerator: [1.4.7221.18964]

# **11 Issues and Risks**

Below are the issues and risks identified during development and testing