

pOS Acquiring dollar

Solution Design Document (SDD)

|  |  |
| --- | --- |
| **Process ID** | P-001 |
| **PU** | EPO |
| **Process Group** | EPO |
| **Process Name** | Pos Acquiring Dollar |
| **Document ID** | 1.0 |

|  |  |
| --- | --- |
| **Version** | 0.1 |
| **Status** | Initial Version |
| **Date** | 24-12-2022 |

# Contents

[Contents 2](#_Toc478342575)

[Document Control 3](#_Toc478342576)

[Template 3](#_Toc478342577)

[Completion stages 3](#_Toc478342578)

[Version history 3](#_Toc478342579)

[1. Introduction 4](#_Toc478342580)

[1.1. Document purpose 4](#_Toc478342581)

[1.2. Process summary 4](#_Toc478342582)

[1.3. Reference artefacts 4](#_Toc478342583)

[2. Solution Overview 5](#_Toc478342584)

[2.1. High level design (HLD) 5](#_Toc478342585)

[2.2. Solution description 5](#_Toc478342586)

[3. Solution Detail 15](#_Toc478342587)

[3.1. Object model 15](#_Toc478342588)

[3.2. Solution components 15](#_Toc478342589)

[4. Operations 16](#_Toc478342590)

[4.1. Business exceptions 16](#_Toc478342591)

[4.2. System exceptions 16](#_Toc478342592)

[4.3. Management information (MI) **Error! Bookmark not defined.**](#_Toc478342593)

[4.4. Scheduling and manual execution 17](#_Toc478342594)

[4.5. Optimization and scaling 17](#_Toc478342595)

[4.6. Alerting 17](#_Toc478342596)

[4.7. Logging 17](#_Toc478342597)

[5. Data Management 18](#_Toc478342598)

[5.1. Storage 18](#_Toc478342599)

[5.2. Privacy 18](#_Toc478342600)

[5.3. Security 18](#_Toc478342601)

[5.4. Preservation 18](#_Toc478342602)

[6. Considerations 19](#_Toc478342603)

[i. Business Glossary 20](#_Toc478342604)

[ii. Attachments 21](#_Toc478342605)

# Document Control

## Template

The following person(s) own the format and information requested in this document template.

|  |  |  |
| --- | --- | --- |
| Team | Name | Relevant Sections |
|  |  |  |

The version history of this template is as follows.

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Change Summary |
| 0.1 | 23/12/2022 | Bakare Sodiq | * Initial Version |

## Completion stages

This document will be completed as per the following sign-off points.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Phase | Relevant Sections | Producer | Sign-off |
| 0.1 | Design | All | Delivery Team | Process Owner ( ) |

## Version history

This document’s change history is as follows.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Date | Author | Approver | Change Summary |
| 0.1 | 23/01/2022 | Bakare Sodiq |  |  |

# 1. Introduction

## 1.1. Document purpose

The purpose of the Solution Design Document (SDD) is to describe the technical solution developed to meet the requirements outlined in the Process Design Document (PDD), including any technical prerequisites and considerations required to deploy, operate, and maintain the process. It is a living document that is incrementally developed as the technical solution is built and is finalized prior to deployment into the production automation infrastructure.

This document will refer to the automation package (“The Solution” or “Solution”) throughout, which represents the Business Objects and Processes, as well as any other peripheral technical components (e.g credentials, templates, databases etc.) used to deliver the automated process.

## 1.2. Process summary

The Pos Acquiring Dollar captures the steps for settling all account transactions done on Visa Transactions. This process is aimed at settling different merchants for transactions processed on the Unified Payment platform at our First Bank branches

## 1.3. Reference artefacts

The Process Design Document (PDD) which captures the business-related details of the process being automated and describes how the automated process is intended to work, including risk and data management control has been included as a link in this manual

The following artefacts should be read in conjunction with this document.

|  |  |  |
| --- | --- | --- |
| Title | Version | Location |
| Process Design Document (PDD) –  Pos Acquiring Dollar | 0.1 |  |

# 2. Solution Overview

This section describes the high-level design of the automated solution.

## 2.1. High level design (HLD)

Get AddDayAsset

Bot load local and External config

Validate Work folders

NO

Download Pos ACQ Report

FBN ACQ DOLLAR MERCHANT

Read Dollar Sheet

add three columns

FEE, VAT and Amount Due Merchant

Treat Dollar Trans

Download MiGS Report

Read Sheet dd three columns: FEE, VAT and Amount Due Merchant

Build Upload

Is amount found In VSS120

Send Account balanced Report to Process Owner

Send Account Not Balanced Report to Process Owner

YES

Download VSS 120 Report

vssreport120

Read Naira Sheet

add Four columns

FEE, VAT, Actual Amount and Amount Due Merchant

Read Vss120 Sheet

Is Account balanced?

Find Total Amount in VSS120 Report

Treat Dollar Reversal Trans

Treat Naira Trans

Treat Naira Reversal

NO

## 

## 2.2. Solution description

* Bot Kill all application
* Bot check the day set on Orchestrator, if it is Monday; Bot will run Saturday, Sunday and Monday. Else the bot only treats the day.
* Sequence: Load all Config Portal
  + Load Internal Config Values: These load all assets in the internal config excel

In the project directory. Output Argument: Config

* + Validate Work Folders: Create folder its does not exist
  + Load External Config workflow: Read system settings output Argument Sysfig
  + Delete work folders and validate again
  + Assign report Date
  + Sequent to check for Monday Logic
* Download Logic workflow
* Assign all WF Variables: all needed variables
* Pre-Load
  + Get all Credentials
  + Create Processing File
* Sequence Download: Logic of Downloading Pos ACQ
* If Download File exist
* Downloading is skip else:
* Else: Sequence: Login and Download
* Do While: Try to login 3 Times
* Logout from FTP workflow
* Login FTP workflow
* If Login Successfully: Download Report FTP
* Sequence If Can't Login: Exception error throw
* If File Unable to Download: Exception error throw
* Sequence Download: Logic of Download VSS 120
* If Download File exist
* Downloading is skip else:
* Else: Sequence: Login and Download
* Do While: Try to login 3 Times
* Logout from FTP workflow
* Login FTP workflow
* If Login Successfully: Download Report FTP
* Sequence If Can't Login: Exception error throw
* If File Unable to Download: Exception error throw
* `READ ALL SHEET: Download VSS 120
  + Sequence: Read POS ACQ Naira Sheet
  + Sequence: Read POS ACQ Dollar Sheet
  + Sequence: Read VSS Sheet
* Delete Download File
* Bot Run Pos Acq Dollar
  + Multiple Assign: all needed variables
  + Sequence: RUN Dollar Sheet
* Filter out other Row for Dollar
  + Multiple Assign
  + Read Work Log: DollarWorkLogDT
  + Filter Data Table: TransType by “VISA” and Sign by “+” PositiveWorkLogdDT
  + Filter Data Table: TransType by “VISA” and Sign by DollarNegaWorkLogDT
* Assign PostiveAmountDollar: Convert.ToDecimal(DollarPosWorkLogDT.AsEnumerable(). Sum (row => row.Field<double>(ColNameOrgAmount.ToString())))
* Assign NegativeAmountDollar: Convert.ToDecimal(DollarNegaWorkLogDT.AsEnumerable(). Sum (row => row.Field<double>(ColNameOrgAmount.ToString())))
* Assign out\_DollarTotalMerchant: PostiveAmountDollar
  + Sequence: Get the Credit on VSS 120 Report
  + Read Range: DT
  + If Dollar Amount Found and MIGS: DollarPosWorkLogDT.Rows.Count > 0
    - Positive Dollar check
    - Assign CheckRow: DT.AsEnumerable().Where(x => x["Column18"].ToString().Replace(",","") == out\_DollarTotalMerchant.ToString()).FirstOrDefault()
    - If the Amount is found: If the Amount is found
      * Amount Found in VSS
    - Else : Check in Migs Report
      * Sequence: Download and Read MIGS Sheet
      * Filter out other Row for Migs
        + Multiple Assign
        + Read Work Log: MIGSWorkLogDT
        + Filter Data Table : keep trans only with 840
        + Assign MigsAmount: Convert.ToDecimal(MIGSWorkLogDT.AsEnumerable().Sum(row => row.Field<double>(ColNameMigsOrgAmount.ToString())))
      * Assign out\_DollarTotalMerchant : MigsAmount
    - If Negative Amount Found
    - Assign Check Row:
    - DT.AsEnumerable().
    - Where(x => x["Column18"].ToString().Replace(",","") == NegativeAmountDollar.ToString()).FirstOrDefault()
  + If amount is in VSS 120 Sheet
  + If amount is in VSS 120 Sheet Reversal
  + If Dollar Trans is Avail
    - Assign distinctDT: DollarPosWorkLogDT.AsEnumerable().GroupBy(x=>x.Field<string>(ColNameRetailerName)).Select(g=>g.First()).CopyToDataTable()
    - For Each Row in Data Table: distinctDT
      * Assign currentMerchant: CurrentMerchantRow[ColNameRetailerName].ToString().Trim()
      * Filter the Merchant to work on
      * CreateMerchantSheet
      * Read Each Current Merchant new sheet created in prev step: runMerchantDT
      * For Each Row in Data Table: runMerchantDT
        + GetMerchantDetails workflow
        + Assign merchantRate: Convert.ToDecimal(merchantDatarow["RATE"].ToString())
        + Assign FEE: Convert.ToDecimal(EachMerchant[ColNameOrgAmount]) \* merchantRate
        + Assign AMOUNT DUE MERCHANT: Math.Abs(Convert.ToDecimal(EachMerchant[ColNameOrgAmount]) - Convert.ToDecimal(EachMerchant["FEE"]))
        + Assign VAT: (7.5/107.5) \* Convert.ToDouble(EachMerchant["FEE"])
      * Write Current Merchant to WorkLog
      * Sequence: Get Total VAT and FEE
        + Read Range: DTS
        + Multiple Assign:

out\_TotalAmtDollar : out\_TotalAmtDollar + Convert.ToDecimal(DTS.AsEnumerable().Sum(row => row.Field<double>("FEE")))

out\_TotalVatDollar: out\_TotalVatDollar + Convert.ToDecimal(DTS.AsEnumerable().Sum(row => row.Field<double>("VAT")))

* + Sequence: RUN Dollar Reversal Sheet
    - Filter out other Row for Dollar
      * Multiple Assign
      * Read Work Log DollarWorkLogDT
      * Filter Data Table: DollarNegaWorkLogDT
    - If there is a Reversal Transactions
      * Assign distinctDT: DollarNegaWorkLogDT.AsEnumerable().GroupBy(x=>x.Field<string>(ColNameRetailerName)).Select(g=>g.First()).CopyToDataTable()
    - For Each Row in Data Table: distinctDT
      * Assign currentMerchant: CurrentMerchantRow[ColNameRetailerName].ToString().Trim()
      * Filter the Merchant to work on
      * CreateMerchantSheet
      * Read Each Current Merchant new sheet created in prev step: runMerchantDT
      * For Each Row in Data Table: runMerchantDT
        + GetMerchantDetails workflow
        + Assign merchantRate: Convert.ToDecimal(merchantDatarow["RATE"].ToString())
        + Assign FEE: Convert.ToDecimal(EachMerchant[ColNameOrgAmount]) \* merchantRate
        + Assign AMOUNT DUE MERCHANT: Math.Abs(Convert.ToDecimal(EachMerchant[ColNameOrgAmount]) - Convert.ToDecimal(EachMerchant["FEE"]))
        + Assign VAT: (7.5/107.5) \* Convert.ToDouble(EachMerchant["FEE"])
      * Write Current Merchant to WorkLog
      * Sequence: Get Total VAT and FEE
        + Read Range: DTS
        + Multiple Assign:

Assign out\_TotalAmtREVDollar: out\_TotalAmtREVDollar + Convert.ToDecimal(DTS.AsEnumerable().Sum(row => row.Field<double>("FEE")))

Assign out\_TotalVatREVDollar: out\_TotalVatREVDollar + Convert.ToDecimal(DTS.AsEnumerable().Sum(row => row.Field<double>("VAT")))

Assign out\_ActualTotalREVDollar: out\_ActualTotalREVDollar +Convert.ToDecimal(DTS.AsEnumerable().Sum(row => row.Field<double>("TRANAMOUNT")))

* + - * + Assign out\_ActualTotalVatREVDollar: Math.Abs(out\_ActualTotalREVDollar)
* Sequence: RUN Naira Sheet
  + - Filter out other Row for Dollar
      * Multiple Assign
      * Read Work Log NairaWorkLogDT
      * Filter Data Table: CurrentMerchantDT
    - If there is a Reversal Transactions
      * Assign distinctDT: NairaWorkLogDT.AsEnumerable().GroupBy(x=>x.Field<string>(ColNameRetailerName)).Select(g=>g.First()).CopyToDataTable()
    - For Each Row in Data Table: distinctDT
      * Assign currentMerchant: CurrentMerchantRow[ColNameRetailerName].ToString().Trim()
      * Filter the Merchant to work on
      * CreateMerchantSheet
      * Read Each Current Merchant new sheet created in prev step: runMerchantDT
      * For Each Row in Data Table: runMerchantDT
        + GetMerchantDetails workflow
        + Assign merchantRate: Convert.ToDecimal(merchantDatarow["RATE"].ToString())
        + Assign FEE: Convert.ToDecimal(EachMerchant[ColNameOrgAmount]) \* merchantRate
        + Assign AMOUNT DUE MERCHANT: Math.Abs(Convert.ToDecimal(EachMerchant[ColNameOrgAmount]) - Convert.ToDecimal(EachMerchant["FEE"]))
        + Assign VAT: (7.5/107.5) \* Convert.ToDouble(EachMerchant["FEE"])
      * Write Current Merchant to WorkLog
      * Sequence: Get Total VAT and FEE
        + Read Range: DTS
        + Multiple Assign:

Assign out\_TotalAmtREVNaira: out\_TotalAmtREVNaira + DTS.AsEnumerable().Sum(row => Convert.ToDecimal(row["FEE"].ToString()))

* + - * + out\_TotalVatREVNaira: out\_TotalVatREVNaira + DTS.AsEnumerable().Sum(row => Convert.ToDecimal(row["VAT"]))
        + out\_ActualTotalREVNaira: out\_ActualTotalREVNaira +DTS.AsEnumerable().Sum(row => Convert.ToDecimal(row["TRANAMOUNT"]))
        + out\_ActualTotalREVNaira: Math.Abs(out\_ActualTotalREVNaira)
  + Sequence: RUN Naira Reversal Sheet
    - Filter out other Row for Dollar
      * Multiple Assign
      * Read Work Log NairaWorkLogDT
      * Filter Data Table: NairaNegaWorkLogDT
    - If there is a Reversal Transactions
      * Assign distinctDT: NairaNegaWorkLogDT.AsEnumerable().GroupBy(x=>x.Field<string>(ColNameRetailerName)).Select(g=>g.First()).CopyToDataTable()
    - For Each Row in Data Table: distinctDT
      * Assign currentMerchant: CurrentMerchantRow[ColNameRetailerName].ToString().Trim()
      * Filter the Merchant to work on
      * CreateMerchantSheet
      * Read Each Current Merchant new sheet created in prev step: runMerchantDT
      * For Each Row in Data Table: runMerchantDT
        + GetMerchantDetails workflow
        + Assign merchantRate: Convert.ToDecimal(merchantDatarow["RATE"].ToString())
        + Assign AMOUNT DUE MERCHANT: Convert.ToDecimal(EachMerchant[ColNameOrgAmount]) / Convert.ToDecimal(EachMerchant[ColFilterNaExrate])
        + Assign FEE: Math.Abs(Convert.ToDecimal(EachMerchant["AMOUNT DUE DOLLAR"]) \* merchantRate)
        + Assign AMOUNT DUE MERCHANT: Math.Abs(Convert.ToDecimal(EachMerchant[ColNameOrgAmount]) - Convert.ToDecimal(EachMerchant["FEE"]))
        + Assign VAT: (7.5/107.5) \* Convert.ToDouble(EachMerchant["FEE"])
        + Assign ACTUAL AMOUNT: Math.Abs(Convert.ToDecimal(EachMerchant["TRANAMOUNT"]) / Convert.ToDecimal(EachMerchant["EXRATE"])) - Convert.ToDecimal(EachMerchant["FEE"]
      * Write Current Merchant to WorkLog
      * Sequence: Get Total VAT and FEE
        + Read Range: DTS
        + Multiple Assign:

Assign out\_NairaTotalMerchant: out\_NairaTotalMerchant +Convert.ToDecimal(Dt.AsEnumerable().Sum(row => row.Field<double>("AMOUNT DUE DOLLAR")))

Assign out\_TotalAmtNaira: out\_TotalAmtNaira +Convert.ToDecimal(Dt.AsEnumerable().Sum(row => row.Field<double>("FEE")))

Assign out\_TotalVatNaira: out\_TotalVatNaira +Convert.ToDecimal(Dt.AsEnumerable().Sum(row => row.Field<double>("VAT")))

* + - * + Assign out\_ActualTotalVatREVDollar: Math.Abs(out\_ActualTotalREVDollar)
* Build Upload File
* Multiple Assign: all needed variables
* Read Work Log: DT
* Build Upload File Data Table: UploadFileDT
* Sequence; Update All Dollar Row
  + Sequence: Dollar Trns
    - Update the VISA MERCHANT FUND
    - Sequence: Dollar
      * Filter out other Row
      * Read Work Log
      * Filter Data Table: DollarPosWorkLogDT
      * Assign distinctDT: DT.AsEnumerable().GroupBy(x=>x.Field<string>(ColNameRetailerName)).Select(g=>g.First()).CopyToDataTable()
      * For Each Row in Dollar Sheet: Update all Merchant
        + Update Each MERCHANT FUND
        + If retailer is DESIGN TRADE AND COM

Update Design Trade Row

* + - * + Else Do Others

If current merchant is Raddison

Sequence: Rassion Merchant

Read Range the Current Merchant

Update Each MERCHANT FUND

Filter Data Table ANCHORAGE LEISURE

Filter Data Table RADISSON BLU

If ANCHORAGE LEISURE

Update Row

If RADISSON BLU

Update Row

Else: Do Other

Update Row

* Sequence: Update Migs If Exist
  + Read Range
  + Filter Data Table: CurentMercMigsDT
  + Sequence: Migs
    - Sequence: Update Migs Fee, Amt, Vat
      * For Each Row in Data Table
        + GetMerchantDetails:
        + Assign merchantRate
        + Assign FEE
        + Assign AMOUNT DUE MERCHANT
        + Assign VAT
      * Write Range
      * Sequence: Get Total VAT and FEE
        + Read Range
        + Multiple Assign

TotalVatMigs: Convert.ToDecimal(DT.AsEnumerable().Sum(row => row.Field<double>("VAT")))

TotalAmtMigs: Convert.ToDecimal(DT.AsEnumerable().Sum(row => row.Field<double>("FEE")))

Amount: Convert.ToDecimal(DT.AsEnumerable().Sum(row => row.Field<double>("AMOUNT DUE MERCHANT")))

* + - Assign distinctDT: CurentMercMigsDT.AsEnumerable().GroupBy(x=>x.Field<string>("MERCHANTTITLE")).Select(g=>g.First()).CopyToDataTable()
    - For Each Row in Dollar Sheet: Update all Merchant
      * Update Each MERCHANT FUND
        + GetMerchantDetails
        + Assign merAccountNum
        + If African Torisum

Treat African Torisum

GetMerchantDetails

Assign merAccountNum

Assign DT: DT.AsEnumerable().Where(x => x["MERCHANTTITLE"].ToString() == "AFRICAN TOURISM CORPOR").CopyToDataTable()

For Each Row in Data Table

Update Row

Multiple Assign

VatMigs

AmtMigs

UPDATE FEE

UPDATE VAT

* + - * + Else: Update Row
    - Multiple Assign
    - Logic For Vat and Fee (Invoke Code)
      * If Send mail for manual Net off Vat and Fee
        + Send Mail
        + Sequence Credit Acct
    - Sequence: Update Vat and Fee
      * GetMerchantDetails
      * Update the FEE
      * Update the VAT
    - Update the VISA MERCHANT FUND REV
      * GetMerchantDetails
      * Assign merAccountNum
      * Debit Sequence
        + Update Row
* Sequence: Reversal Dollar
  + Read Work Log
  + Filter Data Table: DollarPosWorkLogDT
  + Assign distinctDT: DT.AsEnumerable().GroupBy(x=>x.Field<string>("RETAILER ID")).Select(g=>g.First()).CopyToDataTable()
  + For Each Row in Dollar Sheet: Update all Merchant
    - Update Each MERCHANT FUND
    - Read Range the Current Merchant
    - For Each Row in Data Table
      * GetMerchantDetails
      * Assign merAccountNum
      * Assign Amount: Convert.ToDecimal(CurrentRow["AMOUNT DUE MERCHANT"])
      * Debit Sequence
        + Update Row\
* Sequence: Update all Naira Row
  + Update Naira MERCHANT FUND
    - GetMerchantDetails
    - Assign merAccountNum
    - Debit Sequence
      * Multiple Assign
      * Add Data Row
  + Sequence: Update All Merchant
    - Filter out other Row
    - Filter Data Table: DollarPosWorkLogDT
    - Assign distinctDT: DT.AsEnumerable().GroupBy(x=>x.Field<string>(ColNameRetailerName)).Select(g=>g.First()).CopyToDataTable()
    - For Each Row in Dollar Sheet: Update all Merchant
      * If current merchant is Raddison
        + Sequence: Rassion Merchant
        + Read Range the Current Merchant

Update Each MERCHANT FUND

Filter Data Table ANCHORAGE LEISURE

Filter Data Table RADISSON BLU

If ANCHORAGE LEISURE

Update Row

If RADISSON BLU

Update Row

Else: Do Other

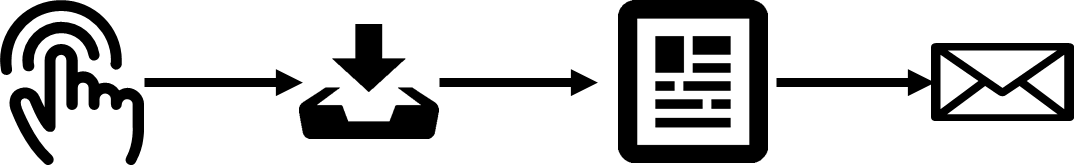
Update Row

* + Sequence: Update Vat and Fee
    - GetMerchantDetails
    - Update the FEE
    - Update the VAT
* Sequence: Reversal Naira
  + Read Work Log
  + Filter Data Table: DollarPosWorkLogDT
  + Update the Naira MERCHANT FUND REV
    - Assign Amount: Convert.ToDecimal(DT.AsEnumerable().Sum(row => row.Field<double>("DOLLAR AMOUNT".ToString())))
    - GetMerchantDetails
    - Assign merAccountNum
    - Credit Sequence
      * Update Row
  + Assign distinctDT: DT.AsEnumerable().GroupBy(x=>x.Field<string>("RETAILER NAME")).Select(g=>g.First()).CopyToDataTable()
  + For Each Row in Dollar Sheet: Update all Merchant
    - Read Range the Current Merchant
    - For Each Row in Data Table
      * GetMerchantDetails
      * Assign Amount: Convert.ToDecimal(CurrentRow["ACTUAL AMOUNT"])
      * Debit Sequence
        + Update Row
* END the Upload File
  + Update Row
* Assign debit: Convert.ToDecimal(UploadFileDT.AsEnumerable().Where(x => x.Field<string>("Type") == "D").Sum(z=>z.Field<decimal>("Amount".ToString())))
* Assign Credit : Convert.ToDecimal(UploadFileDT.AsEnumerable().Where(x => x.Field<string>("Type") == "C").Sum(z=>z.Field<decimal>("Amount".ToString())))
* If Account is Balanced: Math.Round(debit, 1) - Math.Round(credit, 1) == 0
  + Assign out\_IsBalanced : True
* Else
  + Assign out\_IsBalanced : False
* Assign out\_UploadFileName
* Copy File: Upload Template File to Processing folder
* Write Range: UploadFileDT
* Is Account Balanced?
* If True :
* Send\_Job\_Report workflow: Account is Balanced
* Else
* Send\_Job\_Report workflow: Account not Balanced

# 3. Solution Detail

This section describes the low-level design of the automated solution.

## 3.1. Object model



Open Browser Download report Excel Activities Send Email

## 

## 3.2. Solution components

|  |  |  |  |
| --- | --- | --- | --- |
| Ref. | Type | Name | Purpose |
| C1 | Process | Open Application | Opens unified Payment portal to download the required reports |
| C2 | Process | Excel Activities | * Read Excel document into datatable * Add rows * Sort document by columns * Perform computations i.e Add, Subtract, Multiply |
| C3 | Process | Send Email | * Attach excel file to email * Send email |

# 4. Operations

This section describes the controls, reporting and alerting required to operate the solution.

## 4.1. Business exceptions

Events classified as Business Exceptions are those that are not expected to be handled by the virtual worker. That is, they are out of scope of what is described in the PDD.

Business Exceptions are marked as follows.

|  |  |  |
| --- | --- | --- |
| No | Exception | Solution |
| 1 | **Change in report Format**   * Change in number of columns. | * Send email alert to process owner |
| 2 | **Inability to find report**   * The bot is unable to download report due to change in naming convention of the required report. | * Send email alert to process owner |
| 2 | **Inability to find report**   * The bot is unable to download report due to change in naming convention of the required report. | * Send email alert to process owner |
|  |  |  |
|  |  |  |

## 4.2. System exceptions

System exceptions can fall in one of two categories:

1. Known system exceptions – which are known problem or risky areas in the process (e.g. to common system unreliability) that have been specifically catered for with extra retries or redundancies, or at least a specific error description.
2. Unknown system exceptions – which are unplanned errors.

### Known system exceptions

System Exceptions with specific catches are marked as follows.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Scenario | Work Queue | Status | Tags | Required Action |
| N/A | N/A | N/A | N/A | N/A |

### Unknown system exceptions

Unknown System Exceptions will be represented as follows.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Scenario | Work Queue | Status | Tags | Required Action |
| N/A | N/A | N/A | N/A | N/A |

## 4.4. Scheduling and manual execution

The robot will run everyday.

## 4.5. Optimization and scaling

To be determined.

## 4.6. Alerting

Any alerting built into The Solution is described below as per PDD specification.

|  |  |  |  |
| --- | --- | --- | --- |
| Ref. | Scenario | Method | Recipient(s) |
| AL1 | Bot is unable to find Approved/Failed Transaction report | Send Email | Process owner |
| AL2 | Bot has completed its execution | Send Email | Process owner |

## 4.7. Logging

* There is an in-built audit trail which captures actions and timelines of robot’s activities

# 5. Data Management

## 5.1. Storage

The downloaded files i.e Approved/Failed Transactions report and the final report will be stored in a specified folder on the system/server running the bot

## 5.2. Privacy

The bot will not transmit documents/files to external locations (outside FirstBank Bank) and access will be restricted to assigned members of the E-business team

## 5.3. Security

At a specified date (to be determined by members of the COE), the bot will delete all downloaded reports from its download folder.

## 5.4. Preservation

The bot will log onto the applications using credentials supplied by FirstBank bank’s IT

# 6. Considerations

* Stable internet connectivity will be readily available for the bot to function
* Any changes in the naming conventions of documents which are downloaded by the bot may require some updates to robot configuration/process design

# i. Business Glossary

Acronyms and terms used throughout this document are described below.

|  |  |  |
| --- | --- | --- |
| Acronym or Term | Synonym(s) | Full Description |
| VW | PAC, Robot, Bot | Virtual Worker |
| HW |  | Human Worker |
| PDD |  | Process Design Document |
| SDD |  | Solution Design Document |
| VM | VM | Virtual Machine |
| VDI | VDI | Virtual Desktop Interface |
| RPA | RPA | Robotic Process Automation |
| SSO |  | Single Sign-On |

# ii. Attachments

The following attachments relate to this document.

|  |  |
| --- | --- |
| Attachment | Description |
|  |  |
|  |  |