Solution Design Document

BACS List Updates - Council Tax Direct Debits

DigitalNL Programme

Project reference number: M3 – RPA Proof of Concept

1.Introduction

1.1 Purpose

The purpose of this document is to describe in detail the processes, components, objects and tasks which were necessary to support development of the BACS List Updates - Council Tax Direct Debits robot. The document is intended for those developing and supporting the robot. It is a dynamic document which will continually be revised as changes are applied. Its accuracy is essential if different personnel are to successfully support the process.

1.2 Scope

This document has been generated in line with PwC Good Practice, for the development of the Proof of Concept BACS List Updates - Council Tax Direct Debits robot for North Lanarkshire Council. The robot will be attended and will not be using Orchestrator for Queue management or creation of reports for the purpose of the Proof of Concept. Instead, alternative solutions and components have been designed to replace orchestrator functionality where necessary, including generation of a transaction log.

The process will interact with the following applications;

- Civica OpenRevenues
- Microsoft Outlook
- Microsoft Excel
- Temporary Save Location (\TemporarySaveLocation)

1.3 Reference Documents

- Process Definition Document (PDD) - M3 RPA PoC PDD - Council Tax Direct Debit
- M3 RPA Proof of Concept - Process Assessment Report

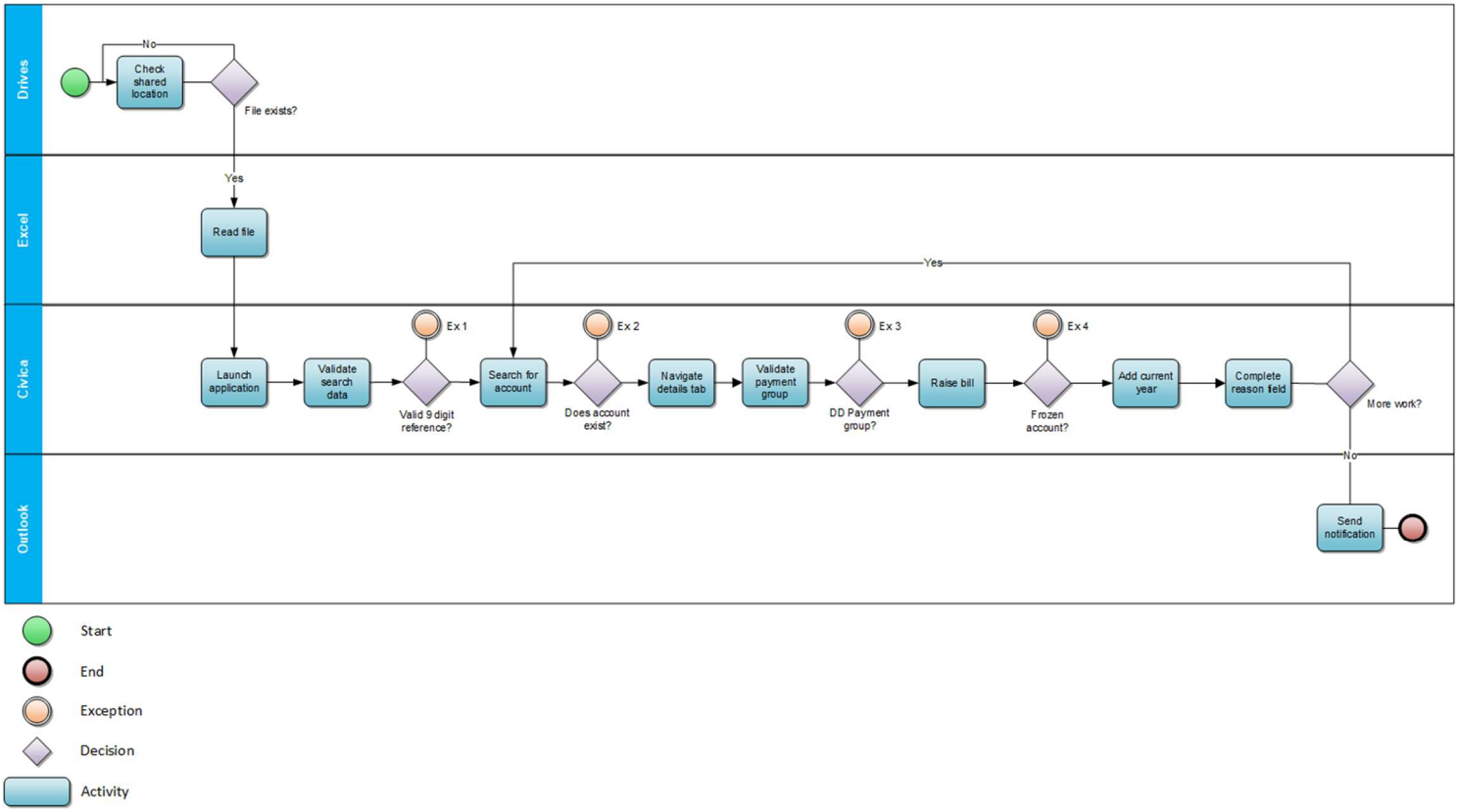
1.4 Definitions

- PDD - Process Definition Document
- SDD - Solution Design Document
- RPA - Robotic Process Automation
- BACS - Bankers Automated Clearing Services
- CTax - Council Tax

2. Solution Design

2.1 Solution Overview

The flow diagram below is a visual representation of the robot which has been built and tested. It reflects the process route, and shows the interaction with the relevant applications required to complete the process.

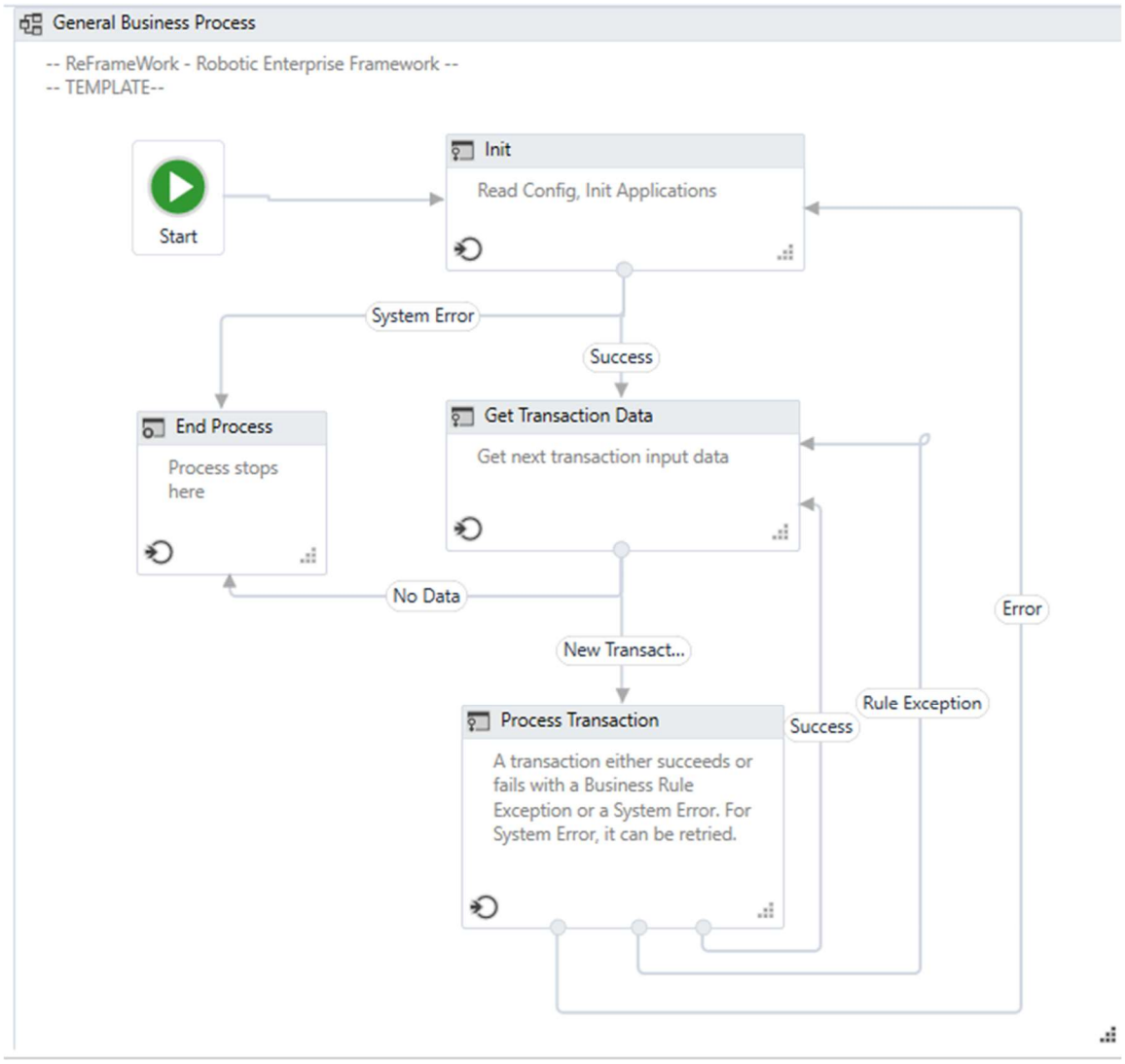


Exception Ref.	Business rule exception description	Exception handling
Ex 1	The account reference number is not a valid 9 digit code	Marked in the transaction log as “Invalid account reference number”
Ex 2	An account does not exist for the reference number searched	Marked in the transaction log as “Account not Found”
Ex 3	The account does not have a Direct Debit payment group	Marked in the transaction log as “Not a Direct Debit Payment group”
Ex 4	The account has a bill freeze assigned	Marked in the transaction log as “Complete (Frozen Account)”

2.2 Solution Components

The robot has been developed in the UiPath Robotic Enterprise Framework, using UiPath Studio 2019.4. The Framework incorporates a number of states and components to execute each transaction. States are a way of separating each stage in the process flow, and are used to invoke each individual automation component. Each automation component carries out an individual step in the process flow. The Framework provides a way to store, read, and easily modify project configuration data, a robust exception handling scheme and event logging for all exceptions and relevant transaction information.

Example Robotic Enterprise Framework;



2.3 Framework States

Component	Description	Outcomes
1. Init (Initialise)	Outputs a settings Dictionary with key/value pairs to be used in the project. Settings are read from the local config file.	At the end of the Initialise State, the robot will have: Read the configuration file into Config; a global dictionary variable, cleaned the working environment by calling the "KillAllApplications" workflow, and initialised all the applications required to complete the full process.
2. Get Transaction Data	Gets data from spreadsheets, databases, email, web API or UiPath server queues. If no new data, set `out_TransactionItem` to Nothing. Assigns the `TransactionItem` by using the `in_TransactionNumber` to index the datarow in the Array which has been passed from the Init state.	From the Get Transaction Data state, the robot can proceed with two possible outcomes: The first outcome is that the robot has obtained new transaction data in the `TransactionItem` variable and therefore will attempt to move on to the Process Transaction state. The second outcome is that the robot will have completed its data collection and as a result, will set the `TransactionItem` variable to Nothing. Alternatively the robot will encounter an Application Exception while processing GetTransactionData.xaml. In this case the robot is unable to get process data, which will cause the robot to proceed to the End Process state.
3. Process Transaction	In this state all other process specific files will be invoked. If an application exception occurs, the current transaction can be retried. If a Business Rule Exception is encountered the exception details will be recorded in the Database	After the Process.xaml file is executed: ● The robot looks for an exception having been generated (either Business Rule or Application). If no exception was caught, it means

	file, the robot will navigate back to the starting position of the process, and the next transaction item will be processed instead. <u>SetTransactionStatus workflow</u> Within this workflow, the following activities take place: ● Sets the TransactionStatus and logs that status in the desired database. The flowchart branches out into the three possible Transaction Statuses: Success, Business Exception and Application Exception. ● Incrementing of the `io_TransactionNumber` variable takes place. If there is an application exception and the MaxRetryNumber has not been reached, an increment is applied to the `io_RetryNumber` variable and not the `io_TransactionNumber` variables. ● Manages both the logging of the Process.xaml output, as well as the management of the next transaction or the retrying of the current one. ● Where `TransactionNumber` and `RetryNumber` are written, allowing for automatic retry in case of an Application Exception.	that the process was successful.
4. End Process	● This is the final state, out of which there are no transitions. <u>CloseAllApplications workflow</u> ● Here all open, working applications will be closed, this workflow ensure that no	The process ends with all applications closed

	applications are left open once all transactions have been processes. KillAllProcesses workflow ● Here all working processes will be killed. This is the default workflow which is invoked, should the “CloseAllApplications” workflow fail to complete successfully.	

2.4 Automation Components

Name	Framework State	Description	Inputs	Outputs
Excel_Check for CSV	Init (Initialise)	This component is responsible for monitoring the shared drive for transaction data	in_Config	out_FileToProcess
Excel_Read CSV	Init (Initialise)	This component is responsible for reading CSV files	in_FileToProcess	out_AccountArray
TransactionLog_Initialise	Init (Initialise)	This component is responsible for building the data table required for the transaction log	-	out_TransactionLog
Civica_Launch Application	Init (Initialise)	This component is responsible for launching the Civica OpenRevenues application	in_Config	-
Civica_Login	Init (Initialise)	This component is responsible for logging into the Civica OpenRevenues application	in_Username	-
Civica_Navigate to CTax Display	Init (Initialise)	This component is responsible for navigating to CTax Display page within Civica	-	-
Civica_Search for account number	Process Transaction	This component is responsible for searching for the account number for customer account	in_TransactionItem, in_ReferenceNumber	out_AccountFound, out_ValidReference
Civica_Close Account	Process Transaction	This component is responsible for closing the open account and returning to the search screen	-	-
Civica_Navigate Details tab	Process Transaction	This component is responsible for navigating to Details tab where validation of payment group occurs	-	-
Civica_Validate DD payment group	Process Transaction	This component is responsible for checking whether payment group contains a DD code	-	out_DDPayment Group
Civica_Navigate to request a bill	Process Transaction	This component is responsible for navigating to section for requesting a bill	in_ReasonCode1	out_Frozen Account
Civica_Request a new bill	Process Transaction	This component is responsible for requesting a new bill	-	in_Reason Code1
Outlook_Send completion email notification	End Process	This component is responsible for sending email notifications to the R&B Team upon completion	in_EmailRecipient, in_EmailSubject	-
Outlook_Send failure email notification	End Process	This component is responsible for sending email notification to the R&B Team open system failure	in_EmailRecipient, in_EmailSubject	-

TransactionLog_UpdateForCompletion	Process Transaction	This component is responsible for updating the transaction log upon a successful completion of a transaction	in_EmailSubject, in_EmailBody, in_EmailRecipient in_LongDate in_TransactionLogPath	-
TransactionLog_UpdateForFailure	Process Transaction	This component is responsible for updating the transaction log for a transaction that has been marked as an exception as per the defined business rules	in_FailureReason, in_EmailSubject, in_EmailBody, in_EmailRecipient	-
Civica_Close Application	End Process	This component is responsible for closing the Civica application after all transactions on the BACS spreadsheet have been processed	-	-
TransactionLog_Export	End Process	This component is responsible for exporting the data for the Transaction Log	in_TransactionLog, in_TransactionLogFileName	in_LongDate

3. Configuration File

3.1 Overview

The Robotic Enterprise Framework designed by UiPath is reliant upon on a Configuration file which is automatically created and aligned to a project upon its creation. The purpose of the config file is to store constants and project specific settings in an easily editable file and avoid hard coding into the main codebase.

3.2 Configuration File Values

The table below lists the constant variables and robot settings, and a description of their use, stored in the single config file used throughout the process. Note that should any of the values in the config file be changed, it is possible the robot will not behave in the expected manner, or potentially prevent the robot from intialising completely.

Name	Value Description
SaveLocation	This value of this item is the full folder path to the temporary save location, where all CSV files will be saved that are to be processed.
ApplicationPath	This value of this item is the full application path to be used by the robot to launch the civica application.
Username	The value of this item is the username to be used by the robot to log into Civica.
ReasonCode1	The value of this item is the reason code to be assigned to all recalculated bills.
CompletionEmailRecipient	This is the email address that all completion emails will be sent to by the robot.
CompletionEmailSubject	This is the subject of all completion emails that will be set by the robot.
CompletionEmailBody	This is the body text of all completion emails that will be set by the robot.
FailureEmailRecipient	This is the email address that all system failure emails will be sent to by the robot.
FailureEmailSubject	This is the subject of all system failure emails that will be set by the robot.
FailureEmailBody	This is the body text of all system failure emails that will be set by the robot.
TransactionLogPath	The value of this item is the full folder path of the save location for the process transaction logs.

4. Exception Management

The robot has been built in line with the UiPath Robotic Enterprise Framework. As a result this robot has been configured to deal with two specific types of exceptions; System Exceptions and Business Rule Exceptions (BRE).

- System Exceptions (also known as Application Exceptions) are all exceptions that occur as a result of the processing system or application operating in an unpredictable manner. Due to the varied nature of these exceptions, all of these specified exceptions are handled in the same manner, whereby the robot will close all of the relevant applications and attempt to retry the same process for the transaction item.

- Business Rule Exceptions (BRE) are exceptions which occur as the result of certain data that the robot depends on being either incomplete or missing. The robot handles these exceptions by recording them, before navigating back to the start position of the process to try the next transaction.

5. Performance Considerations

The following table describes the performance factors which should be considered for the robot as these have a direct impact on the performance of the robot;

Factor	Description
Infrastructure	● The performance of the robot will be directly impacted by the hardware on which it resides. ● For the purpose of the PoC the robot will reside on an NLC assigned laptop which provides the required level of performance for the PoC.
Application	● The performance of the robot will be directly impacted by the performance of the CivicaOpen Revenues test application.
Network	● The robot requires access to the internal NLC network to launch all required applications required to process all transactions. ● Network speed will have a direct impact on the performance of the robot.

6. Testing Considerations

For the full process of design, build and test of the Proof of Concept, real client data was provided for use by the Revenue and Benefits team. This meant that realistic scenarios and cases could be planned and tested, ensuring that the robot would behave in a realistic and appropriate manner.