VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Visa Ekblom

ACCRUALS WITH RPA

Technology and Communication
2020

ACKNOWLEDGEMENTS

I would like to thank my lovely wife for supporting me when I decided to go back to school to get a higher degree. I would also like to thank my dogs for keeping me company while writing this thesis.

VAASAN AMMATTIKORKEAKOULU
Tietotekniikka

TIIVISTELMÄ

Tekijä	Visa Ekblom
Opinnäytetyön nimi	Accruals with RPA
Vuosi	2020
Kieli	englanti
Sivumäärä	52
Ohjaaja	Ghodrat Moghadampour

Tämän opinnäytetyön tavoite oli käyttää ohjelmistorobotiikkaa parantamaan ostolaskujen kustannusjaksotusten kirjaamista palvelukeskuksen ostolaskuprosessissa. Ostolaskujen kustannusjaksotusten kirjaaminen on uusi tehtävä palvelukeskukselle ja vaatisi useita tunteja jos hoidettaisiin käsin. Ohjelmistorobotiikka korvaa tarpeen palkata lisää henkilökuntaa.

Ohjelmistorobotiikan infrastruktuuri oli valmiiksi paikallaan ennen tämän opinnäytetyön alkua, joten kirjoittajan tehtäväksi jäi pelkän ohjelmistorobotiikkaprosessin rakentaminen jo olemassa olevilla työkaluilla. Ostolaskujen jaksotusprosessia ei ollut paikallaan silloin olevassa järjestelmässä, joten kirjoittajalla oli vapaat kädet ohjelmistorobotiikan rakentamisessa. Pelkästään alku sekä loppu oli määritelty.

Ohjelmistorobotiikka alkoi suppealla prosessilaajuudella ja kasvoi hitaasti kattamaan jopa 80% koko ostolaskujen jaksotusprosessin laajuudesta, mikä oli enemmän kuin mitä oli alun perin oletettu ohjelmistorobotiikka pystyvän hoitamaan. Joka kerta kuin laajuus lisääntyi ohjelmistorobotiikkaa jouduttiin muuttamaan että se pystyisi hoitamaan uuden dokumenttimäärän. Viimeinen versio ohjelmistorobotiikasta ostolaskujen kustannusjaksotuksia varten on rakennettu selviämään kaikesta laajuuden kasvamisesta.

Avainsanat	Ostolaskujen jaksotus, ohjelmistorobotiikka, palvelukeskus ja RPA

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietotekniikka

ABSTRACT

Author	Visa Ekblom
Title	Accruals with RPA
Year	2020
Language	English
Pages	52
Name of Supervisor	Ghodrat Moghadampour

This thesis had as an aim to use Robotic Process Automation to handle accruals in a Shared Service Center for the Accounts Payable team. Handling accruals was a new process for the Shared Service Center and would require many working hours to be processed manually. Instead of having to hire more people, an RPA was to be built.

The Robotic Process Automation environment was already set in place before the thesis, so the task was to build the RPA Accruals process itself using the available tools. No process existed for doing the accruals in the current system, so the project had free hands on how to do it. Only the start and end points were given.

The process started with a narrow scope and slowly increased to handle over 80% of the accrual process, which was more than expected out of the RPA Accruals. Every time the scope increased the RPA process was challenged and had to be changed to accommodate the volumes. RPA Accruals was finally built to withstand any volume increase without needing to change the process.

Keywords	Accruals, robotic process automation, shared service center and RPA

CONTENTS

LIST OF FIGURES, TABLES AND CODE SNIPPETS

1 INTRODUCTION

The objective of this thesis is to create a Robotic Process Automation (RPA) to handle Accruals for the Accounts Payable Team in a Shared Service Center. Handling accruals was a new task introduced to the Shared Service Center in 2018. From the beginning it was decided that RPA would be used for this process, because of the volumes of documents it would not be feasible for Accounts Payable Accountants to accrue all Open Purchase Invoice documents with complete accuracy and quality. While the RPA Accruals were being built, the instructions given for accruals for Accounts Payable Accountants was to accrue every Open Purchase Invoice document over 10.000 euros on the real cost objects that were on the documents and everything under 10.000 euros could be accrued to a company default cost object. For the RPA Accruals we will gradually increase both the scope of companies and types of Open Purchase Invoice documents, starting with a handful of companies and slowly getting up to the full scope.

This study contains eight chapters. Chapter 1 introduces the study by outlining the background of the thesis. It states the objective that the research is set to achieve and explains the research problem and research questions. Chapter 1 also touches the companies relevant to the project and a bit of their history. Chapter 2 is about the different technologies and programs used in the project. Chapter 3 we go through what the project should accomplish. In Chapter 4 we look at the design of the environments and the reports that the users will see. Chapter 5 explains the implementation part of the project. In Chapter 7 we will see how the project worked once done. Chapter 8 will be the conclusion of the project.

1.1 Company X Shared Service Center

Company X Shared Service Center (SSC) supports the Company X companies all over the world in financial accounting and other operational tasks and acts as a gatekeeper to ensure compliance. SSC is striving for harmonized one global way of working worldwide with high quality, timely, effective and efficient processes. SSC is sharing best practices and knowledge with focus on continuous development and operational excellence throughout all companies. Due to the thesis being within a area of finance, Company X is anonymous for this thesis.

Some of the services SSC is offering to its companies include:

- Purchase Invoice handling
- Travel Expense handling
- Payments
- Intragroup Matching
- Purchasing
- Customer Invoicing
- Closing and Reporting
- Admin billing
- Master Data

These services are offered globally to all companies in the Company X group. SSC strives to improve the quality and speed of its services all the time so WSSC was the logical choice to get to be the pilot group for RPA. /9/

1.2 UiPath

The Shared Service Center got the task of piloting Robotic Process Automation (RPA) in March 2016. UiPath, a smaller RPA company from Romania had a great tool and was one of the two that were tested during a one-week workshop. In the end, UiPath was a lot more user-friendly compared to the competitor. Once the workshop was over, we arranged the software and infrastructure for RPA, and started a pilot period at full speed. Three departments were involved in the pilot, Shared Service Center, Treasury and Service. After the pilot period Shared Service Center had 10 RPAs running in production on two robots. The pilot was a huge success and RPA has been promoted and spread all over Company X since then.

Company X and UiPath have had very good relations from the start with UiPath listening well to Company X's feedback and Company X getting those new innovative changes and features to new releases into UiPath Studio and UiPath Orchestrator. Between the original pilot period and today, UiPath had become one of the fastest growing and biggest impactors in the RPA tools and technology market, with offices all around the world.

2 TECHNOLOGIES AND SYSTEMS USED

This chapter explains tools and technologies used to achieve the aim of the project.

2.1 RPA

Robotic Process Automation (RPA) is a technology that enables anyone to configure computer software, or a "robot" to mimic and copy the actions of a human interacting within digital systems to execute different kinds of business processes. RPA robots utilize the user interface to process data and use applications just like humans would do. They interpret, trigger actions and communicate with other variety of systems to perform a vast variety of repetitive tasks and processes with one major different: an RPA software robot never sleeps and makes zero mistakes that a human being would do.

RPA is a great tool for cases in instances where a company has a legacy tool, which has no way to link into a newer system or tool. RPA can either transfer all the legacy tool data to the new system or be the link between these two systems and tools. Another good way to use RPA is if a system or tool is missing a feature, but the development costs for that features overweighs the benefits. For this RPA will be a cost-efficient replacement for that development. /6/

2.2 .NET Core

The .NET Core is an open source developer platform developed by Microsoft. With .NET developers can use multiple coding languages such as C#, and Visual Basic. There are endless different editors and libraries the developers have access to. UiPath Studio is built on .NET platform, which means that the coding language inside the UiPath Studio is VB.net based. We also had access to the vast number of .NET libraries whenever we need to build something not available in the UiPath Studio ready-made activities. /2/

2.3 SQL

Structured Query Language (SQL) is a domain-specific language used in programming and designed for managing structured data in relational databases systems. SQL statements are used for common database tasks, such as retrieve data from a

database or update data to a database. The most common SQL commands used to accomplish almost everything needed on a database are such as *Select, Insert, Update, Delete, Create table* and *Drop table*.

In this project we used SQL to maintain a couple of tables in our database. The tables were used are pre-created, so we only needed to manipulate the data is in those tables. /5/

2.4 VBA Macro

For parts of this project, we used Visual Basic for Applications (VBA) which is an implementation of Microsoft's event-driven programming language Visual Basic 6. VBA is built into most Microsoft Office applications.

Visual Basic for Applications enables building faster and more complex calculations in Excel compared to just using Excel functions. It also gives us access to user-defined functions (UDFs), accessing Windows API and other low-level functionality through dynamic-link libraries (DLLs). With these we can create files and access our databases from within the Excel Macro.

In this project we used VBA for manipulating the data and exporting to a separate file for Accruals to be in the correct format which is used by the SAP transaction ZFI_GL_UPLOAD.

2.5 SAP ERP System

Enterprise resource planning (ERP) system is a software used for managing a company’s main business processes. Company X uses SAP Enterprise for its ERP system and it is used for things including but not limited to Accounting, Human Resources, Sales and Production.

An ERP system consists of different databases from which it continuously provides a real time updated view. This is a big of data from hundreds of different kinds of data tables. The task of an ERP system is to be able to combine all this data and provide it to the users in a user interface. The RPA will be interacting the user interface with most of the time during this RPA process.

The SAP application is navigated through user interfaces called Transactions. Each transaction gives us access to the specific data we need at that current moment, for example, a view of all Open Purchase Invoice documents or Purchase Orders. An example of the transactions we use for this project is xFlow Invoice Overview, which gives us access to the Purchase Invoice documents, but we also accessed some of the background tables directly using the SAP transaction General Table Display. /4/

2.6 UiPath Studio

UiPath Studio was the main software used to create this RPA process. UiPath Studio is used for both simple and complex solutions for application integration and automating third-party applications. It is an easy to learn, hard to master kind of software. The basic building of an automation happens with drag and drop blocks called Activities (Figure 1), which range from delete file, click button and read text to more complex activities such as data-table manipulation and doing HTTP Requests. UiPath studio has the option to record interactions happening on the computer and creating a readymade RPA process of all the different clicks and keystrokes. The recorder is great for new Citizen RPA Developers, but for more advanced projects like in this thesis all the activities have been handpicked and refined.

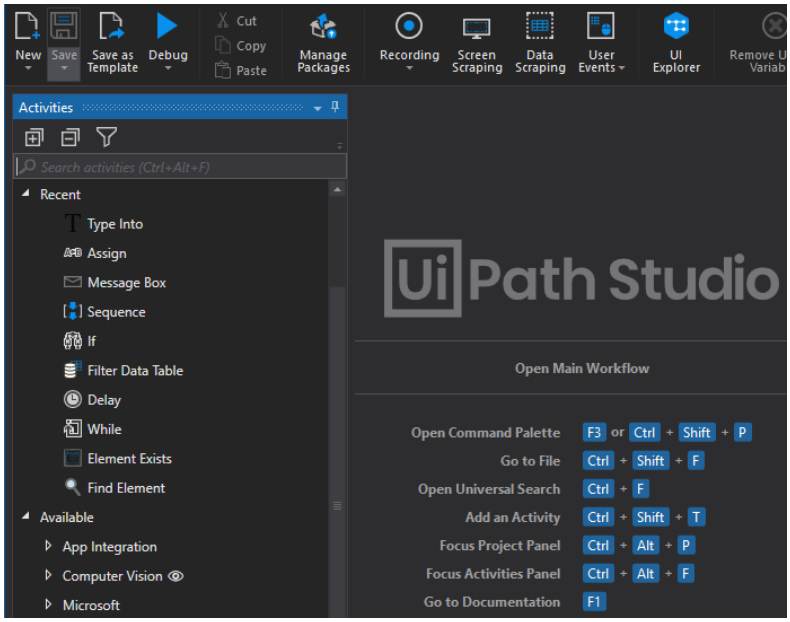

Figure 1. UiPath Studio user interface.

The whole RPA process is mainly built in UiPath Studio. UiPath Studio was used for debugging on the local machine where the RPA is being developed. It would be possible to run the whole RPA process manually with the UiPath studio, but the local machines do not have access to the Orchestrators Assets for all the passwords and usernames. To run the production version, it must be published to Orchestrator and then the Orchestrator will start the RPA process on the unattended robots. /8/

2.7 UiPath Orhcestrator

UiPath Orchestrator is a web application, which enables the control of the RPA environment. The Orchestrator contains all the individual RPA processes with an inbuilt version control. The RPA processes can be deployed to smaller environments which each contain one or more robots. A robot can also be in multiple environments. Orchestrator contains all Queues for RPA processes that use Queues in their process. Orchestrator also holds all the necessary assets that the RPA processes might need while running, for example usernames and passwords for many different applications.

At the time of this project we had access to three robots (Figure 2), which they all had access to their own remote servers. The servers are running the Windows 10 operation system and have all the necessary applications installed on them, such as SAP and Microsoft Office. These robots are so called unattended robots, which means that they run independently without anyone needing to monitor them in real time in a virtual environment. We utilized all three robots for the RPA Accruals and they could run in parallel with the queue system. /7/

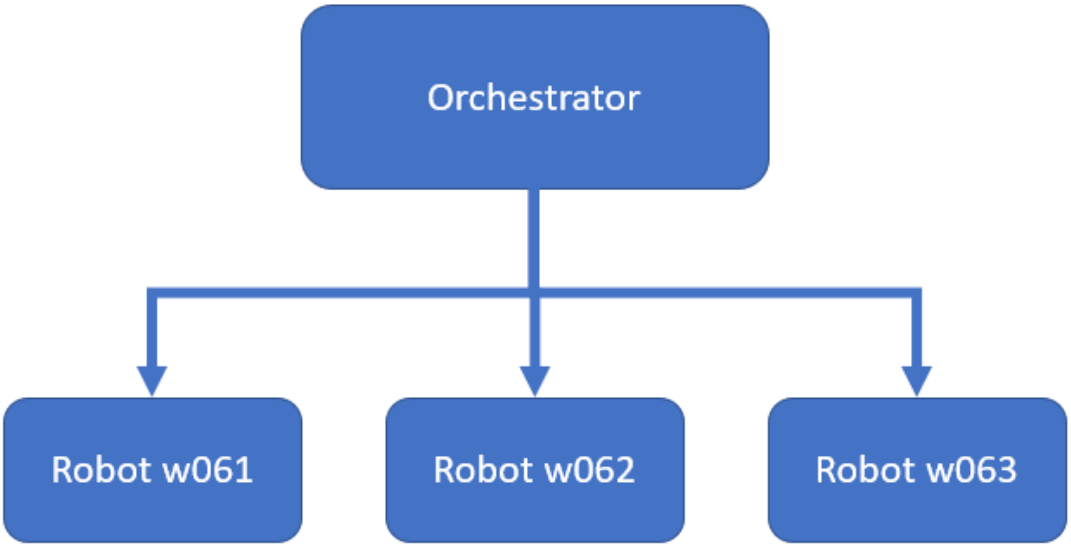

Figure 2. Orchestrator and robot setup.

3 APPLICATION DESCRIPTION

This chapter explains the application requirements and objectives.

3.1 Accruals

Accruals are expenses incurred or revenues earned which affect the company's net income on the income statement, but the actual money for these transactions have not yet been transferred between company and supplier. Accrual accounts include for example accounts payables, accounts receivables and bonus accruals earned or payable. The benefits of doing accruals is to have a better visibility of the company's cash flow. /1/

The accruals are done at the end of each accounting period, in case of this project at the change from one month to the next month. This is done by adjusting in the journal entries for each expense received but not yet recorded in their respective Cost areas, also called accruing in this project. Reasons for not being yet recorded can be that the invoice is still waiting for approval from the buyer or missing goods-receipt.

As an example, the company has received an invoice for 10000 euros on January 15th for materials already received. The invoice has a two-month payment term, setting it to be paid on March 15th. As a policy the company will hold out until the last payment day before the due date before paying the invoice. But as the invoice has already been inserted into the Accounts Payable ERP system, so the 10000 euros is accrued to the cost area which is responsible for the costs. Now the costs are visible on the cost area even before the actual invoice is paid out.

3.2 Quality Function Deployment

The following sections provide a detailed description of the key features and functionalities of the application. The features are classified into three categories: Must-have, Should-have, and Nice-to-have. Must-have is any requirement that absolutely must be delivered for the project to be considered successful, while Should-Have are requirements that should be in the final application but are not needed for an

early production version. Nice-to-have are objectives or requirements that are considered desired or even important to the overall project but can be considered as optional and not necessary for a project success.

The accruals are to be done during the first day of each month. Due to there being thousands of Open Purchase Invoice documents, it is impossible for the Accounts Payable Accountants to finish these manually without taking shortcuts and thus losing quality and detail. Due to this the Accruals RPA is to be run during the night when the month changes and all possible Open Purchase Invoice documents are to be accrued before the Accounts Payable accountants come to work in the morning. Any leftovers that the RPA was unable to accrue are to be handled manually by the Accounts Payable accountants.

The whole RPA process is mimicking how a human would do the accruals on a higher level, but the details in the middle will be a bit different since the RPA can process data better on a massive scale. So instead of looking at data one document at a time the RPA will do mass data manipulation.

3.2.1 Normal Requirements (Must have)

The robot will accrue all the Open Purchase Invoice documents that are not yet paid that are in the release step in SAP transaction xFlow Invoice Overview. Open Purchase Invoice documents in the release step will have all cost objects filled, thus making it easy for the RPA to decide to which costs these Open Purchase Invoice documents will be accrued.

The robot will send Excel reports of which Purchase Invoice documents were still open at the time the RPA was run.

The robot will report which Open Purchase Invoice documents it has accrued and which Open Purchase Invoice documents it has not.

The RPA Accruals process is to be finished before 8.00 in the morning on the first working day of the month.

3.2.2 Expected Requirements (Should have)

The robot will try to accrue Open Purchase Invoice documents that are not yet paid and are in other steps than release in SAP. Some of the Open Purchase Invoice documents have already been filled in cost objects but are still waiting to be approved or have not yet been sent forward by the accountant for various reasons.

Some companies have certain vendors, which should not be accrued, these need to be excluded. Some companies have certain Cost objects that should not be accrued, these need to be excluded.

3.2.3 Optional Requirements (Nice to have)

The robot could report in the Excel report which Open Purchase Invoice documents had Goods Receipt done. Since the robot is checking these fields while its deciding on what to accrue, giving the same information to accountants saves some time.

3.3 Process Cases

A process case is a structure for documenting the functional requirements for the project. Each process case provides a set of scenarios that shows how the system should interact with the case.

There are multiple different kind of steps where the Purchase Invoice documents are open in SAP. There are different kinds of rules how the cost objects are fetched for these and if they can be accrued or not.

The main two differences are if the Purchase Invoice documents are non-PO or PO invoices (purchase order). Which one of these two the Purchase Invoice documents belong to is determined by the purchase that generated the invoice. With a purchase requisition process in use, the purchase will be initiated by a pre-approved purchase order that is delivered to the supplier. With non-PO invoices the purchase is done outside of the regulated process.

Some of the PO invoices also have Goods Receipt, which is a process in the SAP system that tells us when physical inbound goods or materials have arrived at the warehouse.

3.3.1 Non-PO Documents

Non-PO documents are purchase invoices that do not have a purchasing order number. For Non-PO Documents we take out all current open purchase invoice documents from the system that fall within the Company scope. The RPA only checks the Open Purchase Invoice documents that are within the steps scope.

Table 1. SAP Transactions xFlow Invoice Overview non-PO document steps.

Technical name	Description
AAC0	Accountant review for selecting approver
AAC1	Accountant review for completing document
AAC2	Accountant review for rejected document
AAC3	Accountant review for manual posting
AACC	Accountant review for edit document
AREL	Cost object owner review for releasing document for payment
AVAG	Approver review of document

The RPA checks for each of the Open Purchase Invoice documents within the company and step scopes if the Open Purchase Invoice documents have all necessary data filled in to be accrued. The data that is checked is that the total amount of line items and tax is equal to the Open Purchase Invoice documents full amount and that each line item has cost objects assigned to them.

3.3.2 PO Documents

PO documents are purchase invoices that have a purchase order. For PO Documents all current Open Purchase Invoice documents are taken from the system that fall within the Company scope. The RPA only checks the Open Purchase Invoice documents that are within the steps scope.

Table 2. SAP Transactions xFlow Invoice Overview PO document steps.

Technical name	Description
LAC0	Accountant review for selecting approver
LAC1	Accountant review for completing document
LAC2	Accountant review for rejected document
LAC3	Accountant review for manual posting
LACC	Accountant review for edit document
LREL	Cost object owner review for releasing document for payment
LVAG	Approver review of document

The RPA checks for each of the Open Purchase Invoice documents within the company and step scopes if the Open Purchase Invoice documents have all necessary data filled in to be accrued. The data that is checked is that the total amount of Purchase Order lines, line items and tax is equal to the Open Purchase Invoice documents full amount. The Purchase Orders are checked that they have cost objects assigned to them in the Purchase Orders assignments and each line item is checked for cost objects assigned to them. Difference between Purchase Order lines and line items are that the PO lines are the items ordered and line items are usually extra costs, for example freight or packaging costs.

3.3.3 Result and reporting

Final report of what was is done by the robot is to be accessible for the Accounts Payable Accountants in the Excel format. This is done by sending an Excel report per company by email to the Accounts Payable accountants.

3.3.4 Goods receipt reporting

The goods receipt report is to be accessible for the Accounts Payable Accountants in Excel format. This is done by attaching the Goods Receipt report to the Final report Excel.

4 DATABASE AND REPORT DESIGN

This chapter will show how parts of the SQL database and the final reports are designed to look like in the RPA Accruals.

4.1 Database diagrams

A SQL database is used to store the data of the processes outside the RPA process itself. The tables were modelled into two groups to classify the data and identify the relationships between the tables. RPA Accrual scope (Figure 3) is separated as its own group, while the RPA Accrual data is in another group. The RPA process uses four tables (Figure 4), of which the main one is Full_List and the joined tables are PO_Documents, Non-PO_Documents and Goods Receipt Documents. The All Documents table will have a comprehensive list of all open documents, while the Non-PO_Documents and PO_Documents tables will have specific data for each single document. As the last fixed table, we have Goods_Receipts table, which simply contains any good receipt markings for documents where relevant. When the RPA Accruals process runs for each Queue item, these are the tables that are fetched per Company Code.

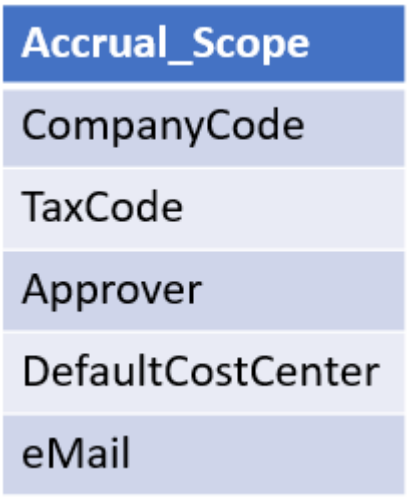

Figure 3. Table setup for Accrual Scope table.

The scope table has all the necessary configuration data according to the company code. It contains the default Tax Code, default Cost Center and the email of the Accounts Payable Accountant team email address for that company code. There is

also the Approver for the company code, which is no longer used in the current version of RPA Accruals, but was left in the table for possible future use. The email field can hold multiple emails, some companies have wished to have their Controllers also receive the RPA Accruals report by email.

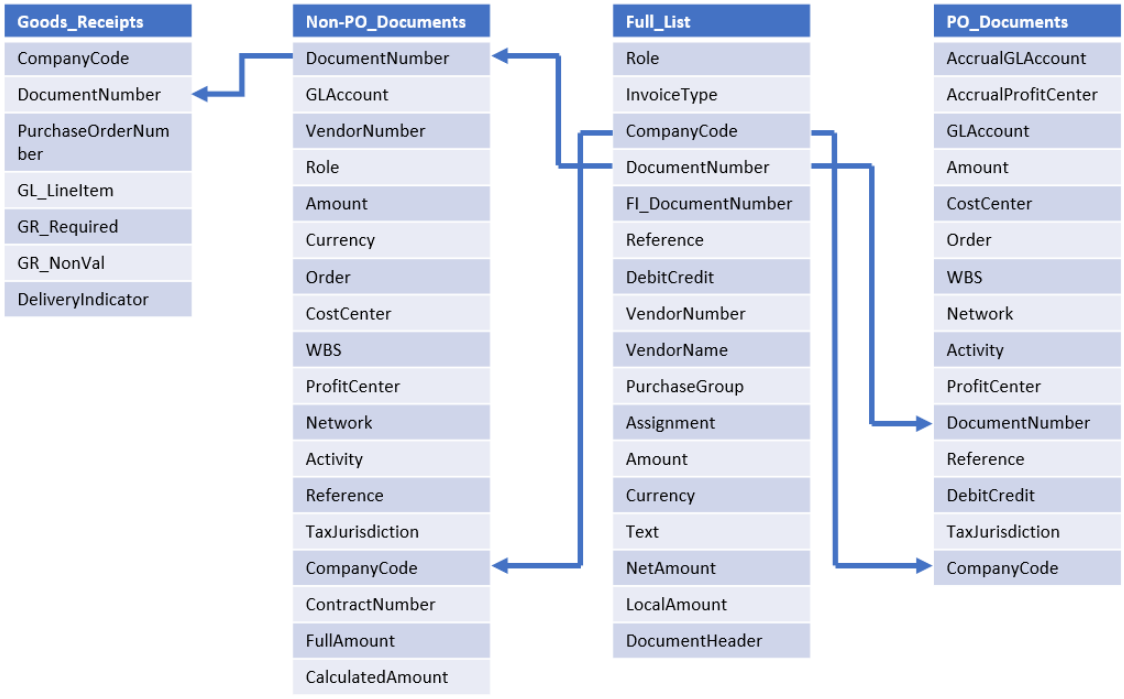

Figure 4. Table setup for collected Open Purchase Invoice document data.

4.2 Report design

For the Excel report that is sent to the Accounts Payable Accountant teams we used the same layout as what is visible in the SAP transaction xFlow Invoice Overview. The report has color coded lines. Green Open Purchase Invoice documents have been accrued. Red ones are Open Purchase Invoice documents that the RPA Accruals failed to accrue. Blanks are Open Purchase Invoice documents that the RPA Accruals skipped due to there not being enough information to accrue. Open Purchase Invoice documents that are set to blue are ones that did not get accrued but have goods receipt information on the second sheet in the report.

Table 3. Final report headers.

Field	Description
Company Code	The company's unique identifier
Document Number	Documents unique number
FI Document Number	Documents number which is only unique per company code
Reference	Documents invoice number
Debit / Credit	Indicator if the document was debit or credit
Vendor Number	Suppliers unique identifier
Vendor Name	Suppliers name
Purchase Group	Purchase Group identifier
Document Amount	Documents total amount in documents currency
Document Currency	Documents currency
Text	Any additional text found on the document
Net Amount	Documents net amount in documents currency
Tax Amount	Documents tax amount in documents currency
Document Header Text	Any additional header text found on the document
GL Flow Document Number	The accrued documents number
GL Flow Sequence Number	The accrued documents sequence number

The Goods receipts report is added to a second sheet in the final report.

Table 4. Goods Receipt report headers.

Field	Description
Company Code	The company’s unique identifier
Document Number	Documents unique number
Purchase Order Number	Purchase Order Number
Line Item	The line item of the purchase order
GR Required	Goods receipt required or not
GR Non Valuated	Goods receipt non valuated
Delivery Indicator	Delivery indicator

5 IMPLEMENTATION

This section illustrates how the application was implemented, with some insights to the early development phase and how the project evolved over time.

The instructions given on how to build the RPA Accruals were very open ended, with only the starting point and end- point defined (Figure 5). The starting point is to use the list of Open Purchase Invoice documents in SAP transaction xFlow Invoice Overview. The end- point is to have Open Purchase Invoice Documents accrued in SAP and have an Excel report on what was accrued and what was not accrued.

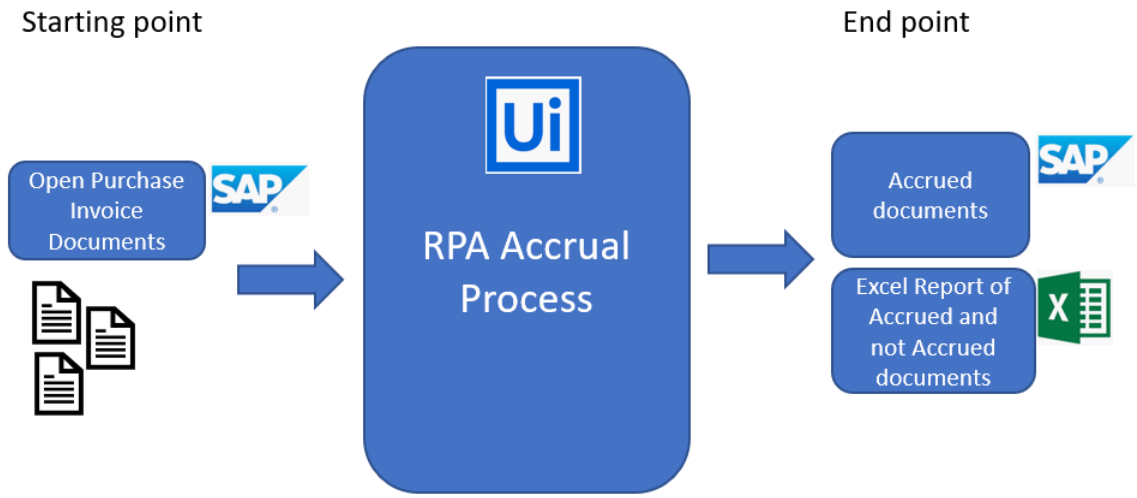

Figure 5. Objectives of RPA Accruals.

In the first working version of the RPA Accruals, we mimicked human actions almost to full detail, including all button presses and field typing (Figure 6). This worked to an extent with some fails while running, due to system slowness and objects changing randomly names and the robot being unable to interact with them. In that version the robot would go one Company at a time, get the list of Open Purchase Invoice documents then one at a time, go into the Purchase Invoice document view, read all the data and then write them into the Accrual file. After all the Purchase Invoice documents had been sorted through, the robot would use the SAP transaction ZFI_GL_REQUEST to upload the Accrual excel file. In SAP transaction ZFI_GL_REQUEST the robot would choose an approver from the company side who would get the posted items into a separate transaction where they could

view the accrued documents and then inform the accountants in case there had been mistakes. With the original scope of 10 company codes and only Open Purchase Invoice documents in Release step, it would take the process around 5 hours to run on one robot, if there were no failures. This version would run without any queue systems, which also meant that if the robot would fail, we would be forced to change the company scope data table manually to remove the ones that already ran and then rerun the process from the start.

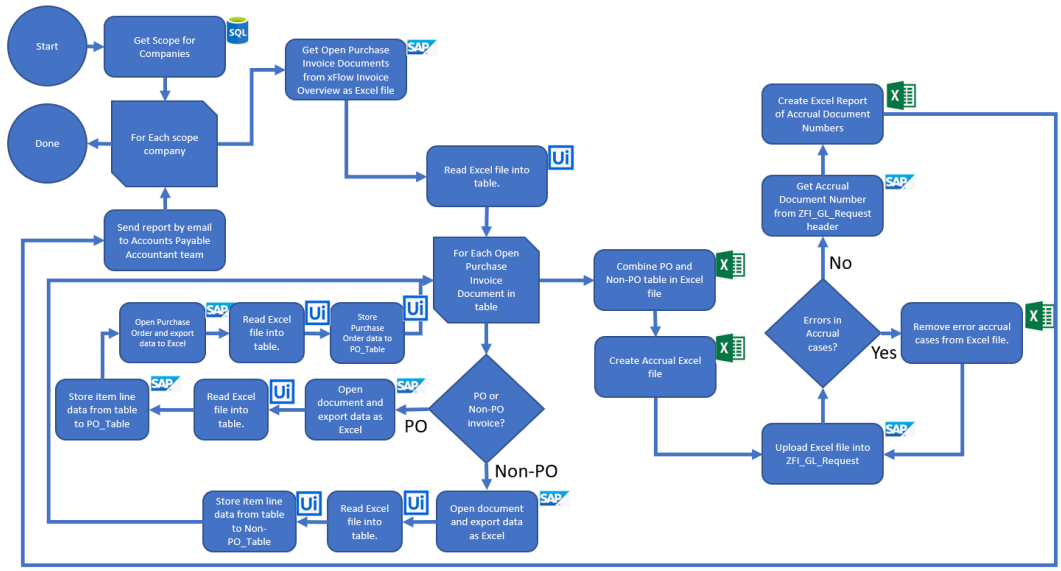

Figure 6. RPA Accruals first production version flow.

Once the Accrual scope started to increase to over 20 companies and we were able to add Open Purchase Invoice documents in other steps than just the release step, the time it took to run through the whole process was no longer feasible to finish before the 8:00 deadline in the morning. The process was changed to run one company at a time, get the list of Open Purchase Invoice documents and then through help of SAP tables get the line level data for all the documents in one go (Figure 7). Using the SAP tables made the whole process much faster and with about 4 times more items, we were able to finish in the same time as the original version in 5 hours.

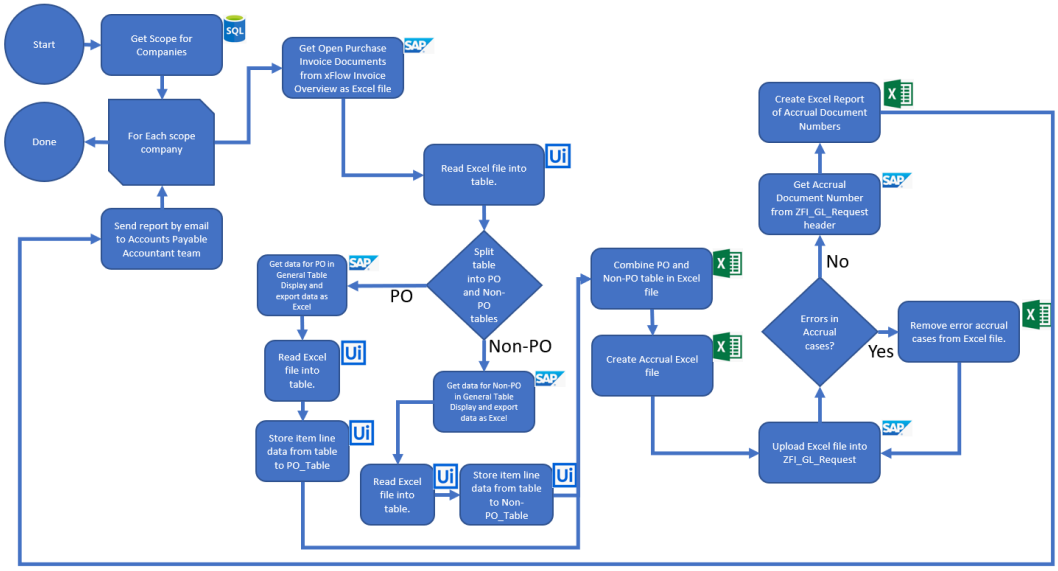

Figure 7. RPA Accruals middle version flow.

The current version has 98 companies in the scope and can handle Open Purchase Invoice documents in 14 different steps in the SAP transaction xFlow Invoice Overview (Figure 8). One robot would not have been able to handle these amounts of volumes, so the only solution was to start involving more robots for the task. One idea was to split up the scopes and give it to different robots, but there was no way to know beforehand what companies had what amount of volumes open in the system. A queue system was built, and the Data Collection part of the RPA was changed. With this version, the process would start with finding all Open Purchase Invoice documents for all companies in scope with a single search. That list would be uploaded to an SQL database table. Then the robot would use SAP tables to get all the line item data for both non-PO and PO documents and upload them to separate SQL tables. After all the data for the run was collected by the Data Collector part of the process, the robot would create Queue items, one for each company code in the scope. Now we could have three different robots running through the queue, each picking up the next available item and processing it. The only tasks that was left for each robot to do when running the queue items was to download that queue items company's pre-sorted data from the SQL table. With the data, the robot would create an Accrual Excel file, which will be uploaded through the SAP transaction ZFI_GL_UPLOAD. The change from SAP transaction ZFI_GL_REQUEST to SAP transaction ZFI_GL_UPLOAD has the benefits of having less UI elements and no need for a prefilled approver. SAP transaction ZFI_GL_UPLOAD is not

available to Accounts Payable Accountants due to the missing approver fields, but this was acceptable for RPA use. With the current version, we can run the whole process through in less than 3 hours.

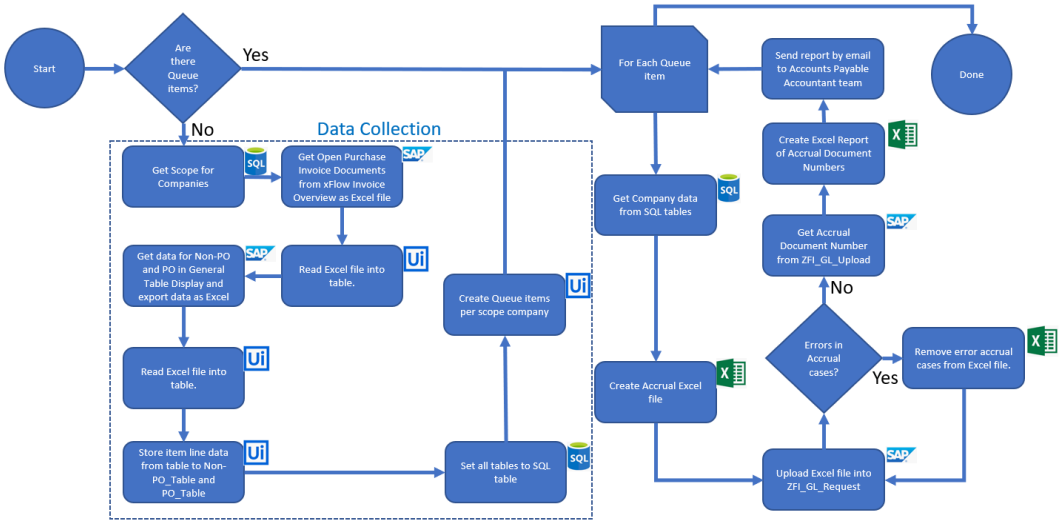

Figure 8. RPA Accruals current process flow.

5.1 Handling the queues

The three robots that are in use for RPA Accruals are scheduled to start the run at midnight and try to run again every 20 minutes until 8.00 in the morning. When the robots starts running RPA Accruals, they begin by checking the Accrual queue if there are any queue items newer than 2 days and if a robot is already running Data Collection RPA module. If none of these are true, one robot will run the Data Collection RPA module and create a queue item for each company that has Open Purchase Invoice documents. Then for each sequenced run, the robots will notice that there are available queue items and process one at a time (Figure 9). With this setup, we can have three robots working on the Accrual process at the same time without the fear of double bookings.

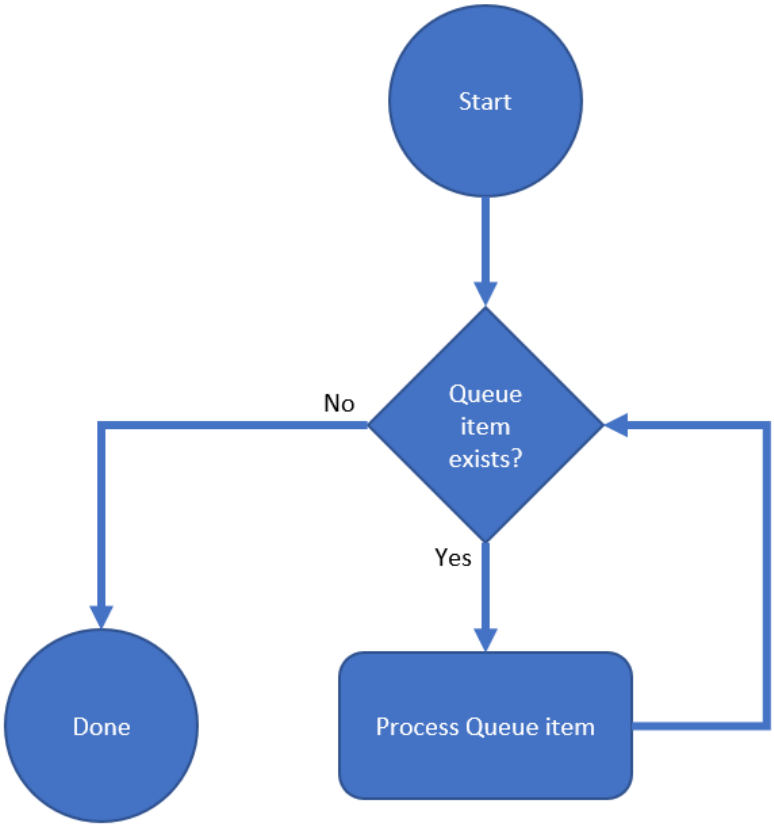

Figure 9. Queue item handling logic.

The queue is named AP_Accruals. Automatic retry is disabled for this queue as we do not want the robot to rerun the process in case it was able to create Accrual documents before failing. In case a queue item fails, the RPA Admin will check how far that queue item was processed and then either retry the queue or inform the Accounts Payable Accountant team that the specific Company Code failed and needs to be handled manually.

Table 5. AP Accruals Queue item statuses.

Status	Description
New	Queue Item has been created but not started.
In Progress	A Robot has started working a queue item
Successful	A Robot has successfully processed a queue item
Failed	A Robot has failed to process a queue item. The queue item needs to be reprocessed.

5.2 SAP Automation with UiPath

Parts of the RPA Accruals process is done in the same way the Account Payable Accountants would do. The robot goes to SAP transaction xFlow Invoice Overview to get a view of all current Open Purchase Invoice documents. Instead of going through the Open Purchase Invoice documents one by one looking for cost objects and details, the robot instead uses SAP Tables for that information. Normally regular SAP users do not have access to the SAP Table transactions, such as General Table Display, as these are reserved for admin users. For the robot, it was a more logical choice to fetch data from tables instead of using the layouts created for accountants.

With RPA, we still access the SAP transactions in the same way as any human being would do by writing in the transaction codes and clicking through the interface. When getting the data that we need from the transaction user interfaces, it would have been possible to read the visible data objects with the robot, but in all cases where there is more data than a couple of rows, it is faster to export the data into an Excel report, then read the Excel file into a data-table within the RPA process (Figure 10).

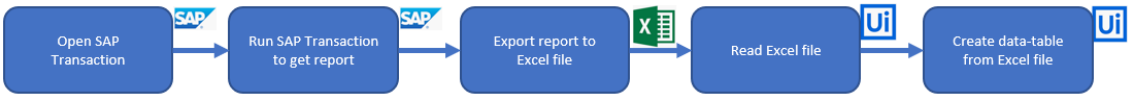

Figure 10. Flow of getting data from SAP to the RPA.

The way the robot interacts with objects, such as buttons and fields is done by using the UiPath Studio Activities. Most of the time the objects are identified by their underlying selectors instead of looking for a visual object (Figure 11). As an example, an OK button in SAP would have an underlying selector name behind the user interface, to which UiPath is able to send a simulated click command. This would click the button without the robot needing to take control over the mouse.

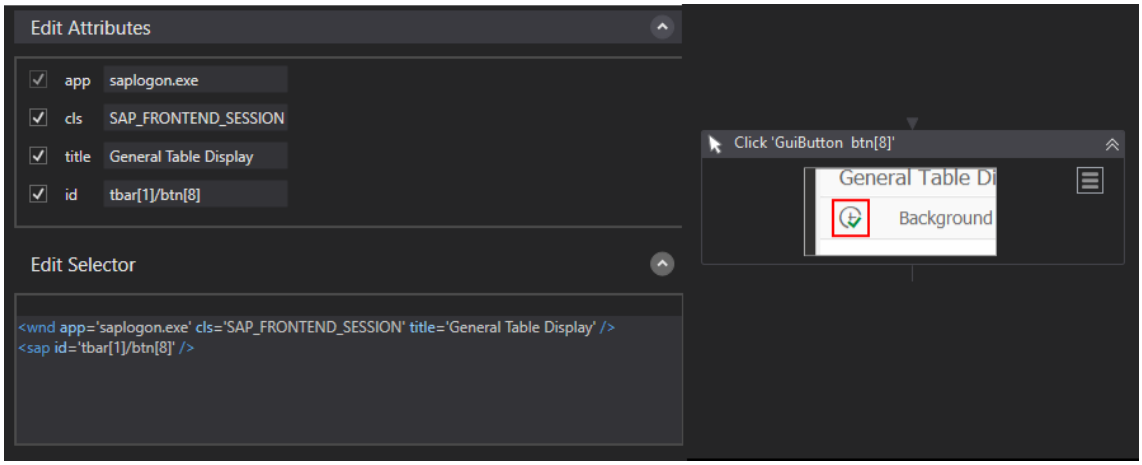

Figure 11. Example of a selector of an Execute button in SAP transaction General Table Display.

The interaction with SAP transactions objects is built in UiPath Studio as a sequence. UiPath Studio activities have an inbuilt wait function, which means that it will wait for a certain amount of time to be able to interact with an object before throwing an exception. This is especially important when moving from one SAP transaction user interface to another, as there might be some delays with loading between. The wait time can be adjusted on an activity level, so when we know that loading a report might take even up to 5 minutes, we can adjust the wait time to be a bit longer.

Table 6. SAP Transactions used in the RPA Accruals.

Technical Name	Name	Description
/WMD/XF_IV_ADMIN	xFlow Invoice Overview	Used to give an overview of all Purchase Invoice documents and for processing said documents.
SE16n	General Table Display	Used to get access to SAP tables. Takes as input a table name, which allows for further filtering before running the view.
SE38	ABAP Editor	Used for running custom programs. In this project we use it to run a program which consists of 4 joined tables and gives us specific fields for filtering the data.
ZFI_GL_UPLOAD	Report ZFI_GL_UPLOAD	Custom built transaction used to mass post documents from an Excel file. In this project used to post the Accrual documents.

5.3 How Tables are Joined and Processed

The PO documents lines are processed inside the RPA process by manipulating data-tables. Each PO document accrual line has 16 fields, which are picked up from a joined table named PO_Documents, which consists of three SAP tables, xFlowLinesTable, PO_OpenDocumentsTable and afvcTable (Figure 12). Furthermore, the PO_OpenDocumentsTable table is joined together from five other tables, rsegTable, rbkpTable, ekpoTable, ekknTable and msegTable (Figure 13).

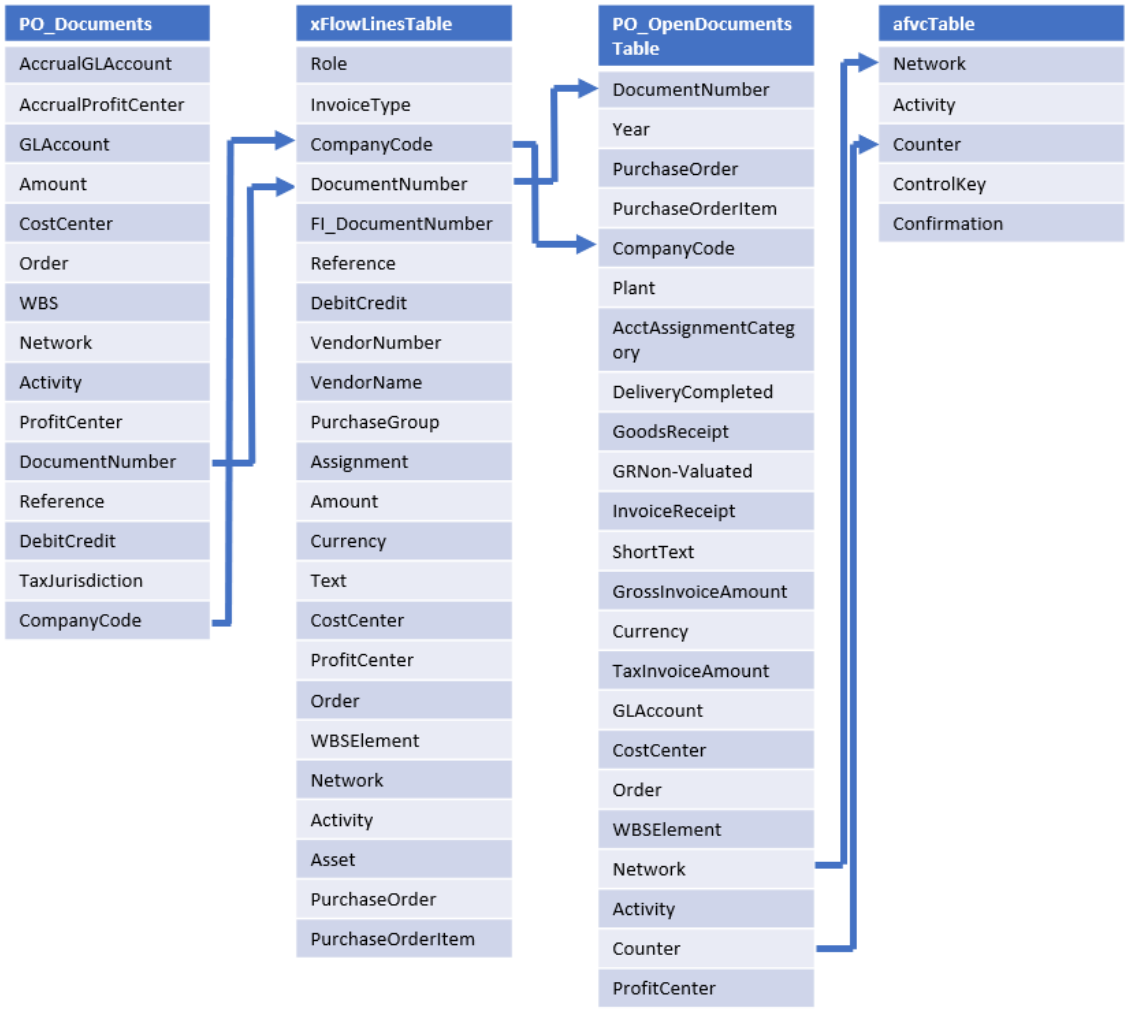

Figure 12. Table joins for PO_Documents table.

The xFlow table has been fetched from the SAP transaction xFlow Invoice Overview and consists of current Open Purchase Invoice documents on the item line detail level. The afvcTable table has been fetched from the SAP transaction General Table Display and is for getting the Activity code for individual Network objects. The third table, which is “PO_OpenDocumentsTable” has been taken from a programming SAP transaction ABAP Editor where we have pre-joined the five tables. Not all of the tables have been added to the “PO_OpenDocumentsTable” table in the program, because xFlow table is a report instead of a table and is not available to be joined with the other tables inside SAP transaction ABAP Editor.

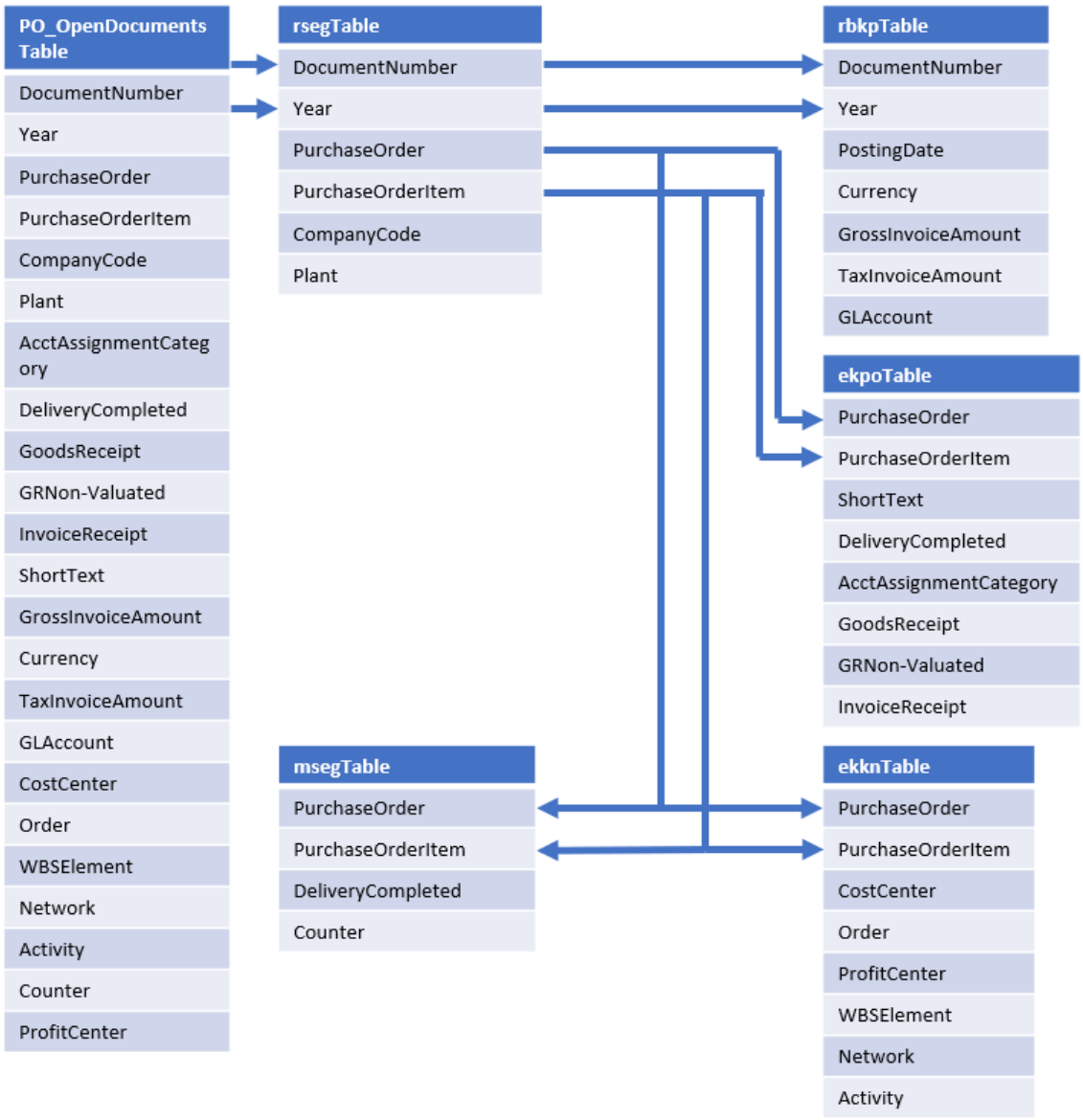

Figure 13. Table joins for PO_DocumentsTable table.

For non-PO documents, we take out the full list of Open Purchase Invoice documents only on the header level from the SAP transaction xFlow Invoice Overview, creating a table named "Full_List" out of it. For the line level data, we fetch two tables from the SAP transaction General Table Display called vbsegsTable and vbsegkTable (Figure 14). These tables are temporary joined to each other to build the final Non-PO_Documents table.

The non-PO Documents and PO documents tables are slightly different from each other, since the data are in different tables and thus very different in the easiness of accessibility of the data. For non-PO documents, we have to get tax amount lines

from the table vbsegkTable, which adds its own lines to the Non-PO_Documents. The tax lines from vbsegkTable are summed up together with any non-PO documents line item data per Open Purchase Invoice Document to the CalculatedAmount field. The CalculatedAmount field is later compared to the FullAmount field. In case these differ from each other the whole Open Purchase Invoice document is removed from the Non-PO_Documents table.

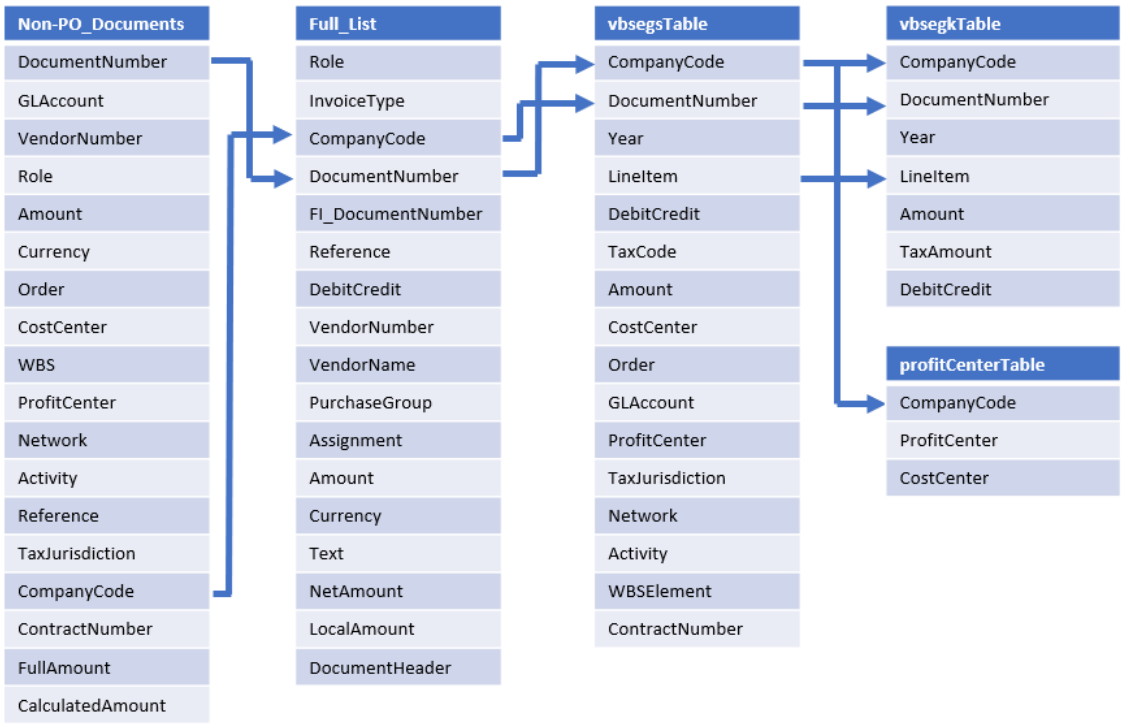

Figure 14. Table joins for Non-PO_Documents table.

Table 7. SAP Transactions Display General Table different tables used in RPA Accruals.

Table Technical name	Table Name
RSEG	Document Item: Incoming Invoice
RBKP	Document Header: Invoice Receipt
EKPO	Purchasing Document Item
EKKN	Account Assignment in Purchasing Document
MSEG	Document Segment: Material
AFVC	Operation within an order
VBSEGS	Document Segment for Document Parking – G/L Account Database
VBSEGK	Document Segment for Vendor Document Parking

5.4 Excel Macro Process

Excel macro scripting is used to perform correct formatting to the data for both non-PO and PO Documents. As the SAP transaction ZFI_GL_UPLOAD requires an Excel file as input and Excel sometimes tries to be too helpful with formatting of data, running into problems where Excel turns dates into amounts or vice versa. Instead of using raw data when creating the Excel files, we first go through all the data with the Excel Macro and then output a separate Excel file.

For the early versions of RPA Accruals, the Excel Macro would get the Open Purchase Invoice Documents list by copying the sheets from the exported Excel file into the Excel Macro. An example of a macro is given below.

```
Dim x As Workbook
Dim y As Workbook
Set x = Workbooks.Open(filepath & "full_list.XLSX") '//Defines the Excel file to be imported.
Set y = Workbooks("MACROACCRUALS.xlsm") '//Defines the working excel file.
Workbooks("full_list.XLSX").Sheets("Sheet1").Copy _
After:=Workbooks("MACROACCRUALS.xlsm").Sheets(1) '//Copies a specific sheet from Excel file to the working file.
ActiveSheet.Name = "FullList" '//Names the copied sheet.
x.Close
```

Code Snippet 1. Import Excel sheet within an Excel macro.

This way of getting the data for the Excel Macro worked for a while, but when the Open Documents export file was a bit bigger in size, the RPA Accrual would completely lose the Excel Macro object after the macro was done and crash the RPA process. In the current version the importing of data is instead handled by the RPA Process, directly writing the tables into the correct sheets in the Excel Macro file by using the UiPath Studio activities.

The Excel Macro is split into three Main parts, “Main”, “MainClean” and “MainReport”. “MainClean”-macro is run at the very start of the RPA Accrual Queue Item process; this is to delete old data that might be in the Excel Macro file. “Main”-macro is run before Accruals are posted in SAP transaction ZFI_GL_UPLOAD and “MainReport”-macro is run afterwards. “Main” consists of seven sub scripts, while “MainReport” consists of four sub-scripts.

The macros are structured in a way to handle any errors and making it easy for the robot to evaluate if there was a problem or not. For this, we used a separate sheet named “Error”, if the macro fails, it will write the sub- scripts name in the Error sheet. When the robot has run one of the main macros, it will afterwards read the Error sheet to see if there was any problems while running the macro. The structure of this can be seen in the following code.

```
Sub MainClean()
Worksheets("Error").Cells.ClearContents '//Clears the error sheet.
On Error GoTo ErrHandler:
NameSheets '//Sub macro.
CleanWholeExcel '//Sub macro.
```

```
GetTaxFromSQL '//Sub macro.
Exit Sub
ErrHandler: '//Error handler, writes Error and the sub macro into the error sheet.
Workbooks("MACROACCRUALS.xlsm").Worksheets("Error").Cells(3, 1).Value = "Error"
Workbooks("MACROACCRUALS.xlsm").Worksheets("Error").Cells(3, 2).Value = errorcode
End Sub
```

Code Snippet 2. Excel Macro MainClean.

The sub- macros in MainClean are

- NameSheets – Initializes all the variables for sheets and file paths.
- CleanWholeExcel – Deletes all extra sheets and clears content on all constant sheets.
- GetTaxFromSQL – Fetches company codes, tax codes and default cost center from the SQL database Accrual Scope table.

Accessing the SQL database with an Excel Macro is done with the following code.

```
errorcode = "GetTaxFromSQL error"
Dim rs As ADODB.Recordset
Set rs = New ADODB.Recordset
Dim SQLQUE As String
sTax.Cells.ClearContents '//Clears content on the Tax sheet.
Connect.Open "Provider=sqloledb.1;Data Source=FISERVER1\WARTSILA_SQL;Trusted_connection=yes;" '//Opens an SQL connection.
SQLQUE = "SELECT [CoCode], [TaxCode], [DefaultCC] FROM [RPA_Processing].[dbo].[Accrual_Scope]" '//The SQL command to select data from the table.
rs.Open SQLQUE, Connect '//Executing the SQL command.
sTax.Range("A1").CopyFromRecordset rs '//Writing the data received from the SQL table into the Tax sheet.
Connect.Close
Set Connect = Nothing
```

Code Snippet 3. SQL query through Excel Macro.

The sub macros in Main are

- NameSheets – Initializes all the variables for sheets and file paths.

- GetCurrDoc – Gets all unique currencies in the Open Purchase Invoice Documents list and gets the company specific tax codes from the SQL scope table import.
- GetLinesforFI – Goes through the non-PO Documents accrual list. Removes documents that fall into a couple company specific rules. Combines similar lines of accruals to reduce the run time in later steps of the process.
- GetLinesforMM - Goes through the PO Documents accrual list. Removes documents that fall into a couple company specific rules. Combines similar lines of accruals to reduce the run time in later steps of the process.
- ProjectExclude – Removes certain project related Open Purchase Invoice documents from the Accrual list.
- CreateAccrSheets – Creates the separate accrual sheets from the non-PO and PO Documents accrual lists.
- SaveExcelsAsNewFiles – Creates new Excel files of the sheets created by CreateAccrSheets.

Parts in the GetLinesforFI and GetLinesforMM sub- macros that checks for the data if it is to be removed or not is done with a for each loop like the following code

```
lRow = slineitems.Range("B" & Rows.Count).End(xlUp).Row '//Define the last row on the sheet.
For i = lRow To 1 Step -1
  If Left(slineitems.Range("H" & i).Value, 1) = "6" Or Left(slineitems.Range("H" & i).Value, 3) =
"999" '// Checks if the leftmost characters in column H are 6 or 999.
Then
    documentnumber = slineitems.Range("A" & i).Value '//Defines the document number to be
deleted.
    For y = lRow To 1 Step -1
      If documentnumber = slineitems.Range("A" & y).Value Then '//Looks for document num-
ber per row.
        slineitems.Range("A" & y).EntireRow.Delete '//Deletes the row.
      End If
    Next y
  End If
Next i
```

Code Snippet 4. Excel Macro removal of non-PO documents that have orders starting with 6 or 999.

Parts in the GetLinesforFI and GetLinesforMM sub- macros that combine data if it is similarly done with a for each loop like the following code

```
slineitems.Columns("A:R").Sort Key1:=slineitems.Range("G1"), order1:=xlAscending, Key2:=slineitems.Range("C1"), order2:=xlAscending, Header:=xlNo
slineitems.Columns("A:R").Sort Key1:=slineitems.Range("A1"), order1:=xlAscending, Key2:=slineitems.Range("D1"), order2:=xlAscending, Key3:=slineitems.Range("H1"), order3:=xlAscending, Header:=xlNo '//Sorts the data according to columns G, C, D and H.
lRow = slineitems.Range("B" & Rows.Count).End(xlUp).Row
For i = lRow To 1 Step -1 '//Checks for duplicate Orders.
  If slineitems.Range("A" & i).Value = slineitems.Range("A" & i + 1).Value And slineitems.Range("B" & i).Value = slineitems.Range("B" & i + 1).Value And slineitems.Range("D" & i).Value = slineitems.Range("D" & i + 1).Value And slineitems.Range("G" & i).Value = slineitems.Range("G" & i + 1).Value And slineitems.Range("B" & i).Value = slineitems.Range("B" & i + 1).Value Then
    If slineitems.Range("B" & i).Value = slineitems.Range("B" & i + 1).Value And slineitems.Range("C" & i).Value = slineitems.Range("C" & i + 1).Value Then
    If slineitems.Range("H" & i).Value = slineitems.Range("H" & i + 1).Value And slineitems.Range("H" & i).Value <> "" Then '//Compares various columns if they are the same as row above.
      slineitems.Range("F" & i).Value = CDbl(slineitems.Range("F" & i).Value) + CDbl(slineitems.Range("F" & i + 1).Value) '//Adds amounts together.
      slineitems.Range("A" & i + 1).EntireRow.Delete '//Deletes duplicate rows.
    End If
    End If
  End If
Next i
```

Code Snippet 5. Combine similar data in Excel Macro.

The sub- macros in MainReport are

- NameSheets - Initializes all the variables for sheets and file paths.
- ReportDocuments – Creates a copy of the full Open Purchase Invoice Documents and names it Report. Goes through the list of Accrual cases that the robot got from the SAP transaction ZFI_GL_UPLOAD while posting the accruals. Compares this list with the report sheet and marks them accordingly.
- SaveReport – Creates a new Excel File from the Report sheet.

The way the macro finds what to report out of the Open Purchase Invoice documents is done with a loop between the full Open Purchase invoice documents list and the Accrual Documents report with the following code.

```
lRowFL = sFL.Range("A" & Rows.Count).End(xlUp).Row '//Define the last row on the sheet FL.
lRowGLF = sDataS.Range("E" & Rows.Count).End(xlUp).Row '//Define the last row on the sheet GLF.
  For i = 2 To lRowFL
    For y = 1 To lRowGLF
      If sFL.Range("S" & i).Value = sDataS.Range("E" & y).Value Then
        sFL.Range("R" & i).Value = "'" & sDataS.Range("F" & y).Value
        If sDataS.Range("G" & y).Value = "OK" Then '//Checks if value is OK.
          sFL.Range("A" & i).EntireRow.Interior.ColorIndex = 50 '//Colors the row green.
        ElseIf sDataS.Range("G" & y).Value = "ERROR" Then '//Checks if value is ERROR.
          sFL.Range("A" & i).EntireRow.Interior.ColorIndex = 3 '//Colors the row red.
        End If
        GoTo GLFound '//Goes to the next item if match was found.
      End If
    Next y
GLFound:
  Next i
```

Code Snippet 6. Excel Macro for matching Accrual documents and Open Purchase invoice documents.

Once the data has been checked and manipulated by the macro, it will create Excel sheets per company code. These are in a predefined format that the SAP transaction ZFI_GL_UPLOAD requires.

5.5 Posting the Accruals

We use the SAP transaction ZFI_GL_UPLOAD to post the accruals. ZFI_GL_UPLOAD is a simple SAP transaction with options for only filling in the Excel file path and choosing if a test run or not. Once the Excel file has been processed by SAP transaction ZFI_GL_UPLOAD, either a report of successfully done accruals or a report of invalid/failed accrual lines is received.

In the case of a successful report, the view gives us the Accrual Document numbers for each individual Accrual case required for reporting. To get this the robot exports

the report view to an Excel file, reads the Excel file to a table and then imports that table to our Excel Macro file.

In the case of a failed report view the SAP transaction ZFI_GL_UPLOAD informs which individual Accrual cases failed and why. A common reason would be that the cost object is closed for bookings at the time of the run. The robot exports the report view to an Excel file, reads the Excel file to a table and then compares the failed report table with the original run and removes all the failed lines. The failed report is also imported to the Excel Macro file on a separate sheet, including the informed reason why the Accrual cases failed. Once the original Accrual file has been purged of any failed Accrual cases, the file is run again through SAP transaction ZFI_GL_UPLOAD. This is repeated until either the SAP transaction ZFI_GL_UPLOAD gives a successful report, or the Accrual file runs out of Accrual cases. Usually the Accrual file gives invalid cases not at all or just once, but there have been rare cases where the file had to be purged twice.

The layout of the Accrual File goes according to the SAP transaction ZFI_GL_UPLOAD template. Important fields to be filled in are Document Type and Header Text, which identifies the accruals as done by the robot.

Table 8. Part 1 of 2, Accrual Excel file headers required for SAP Transactions ZFI_GL_UPLOAD.

Field Name	Description
Sequence	Identifies grouped up accrual as one Accrual Case.
Company code	The identifier of the Company X's specific company
Document type	SAP document type. For RPA Accruals "R3" is used to easily identify them.
Currency	Currency of the accrual.
Document Date	Accrual document date. Set as the last day of previous month.
Posting Date	Accrual posting date. Set as the last day of previous month.
Reference	Reference of the accrual document. (not used)
Document Header Text	Header text of the accrual document. Set to "External Accruals" month name and year.
Exchange Rate	Manual exchange rate where applicable, currently not used.
Calculate Tax	Calculate tax indicator where SAP would automatically create the tax for documents. The indicator is turned off.
Posting Period	Not in use.
Posting Key	Not in use.
GL Account	The General Ledger account of the accrual line.
Amount in Document Currency	The amount of the accrual line.

Table 9. Part 2 of 2, Accrual Excel file headers required for SAP Transactions ZFI_GL_UPLOAD.

Field Name	Description
Tax Code	Tax code. This is a default for each company code. Only the line with a cost object will have the tax code filled in.
Assignment	Default text “External Accruals RPA” is used.
Text	“External Accrual RPA” and document number of the specific accrual is used.
Long Text Field	Not in use.
Trading Partner	Not in use.
Cost Center	The cost object “Cost Center” is filled here where applicable.
Order	The cost object “Order” is filled here where applicable.
Profit Center	Profit Center
WBS Element	The cost object “WBS Element” is filled here where applicable.
Network	The cost object “Network” where applicable.
Oper: / Act, Number	The cost object “Activity” where applicable.
Tax Jurisdiction Code	Tax Jurisdiction code is filled in for a couple companies, for example US and Brazil.
Business Place	Business place is filled in for Brazil.
Contract Number	The cost object “Contract Number” is filled here where applicable.

5.6 Final report

When the RPA process has gone through the posting of accruals, the Excel macro is run to mark each Open Purchase Invoice document in the Full List either Accrued with a note of the Accrual document number, failed with the system given reason why it failed or leave them blank if they were not added to the accrual list. This Full List Excel sheet and the Goods Receipt Excel sheet are both exported to one separate Excel file, which the robot emails to the Accounts Payables Accountant teams shared email boxes.

5.7 Error Handling

Both the Data Collection module and the process Queue item module of the RPA Accrual have their own but similar error handling. In the case where the Data Collection would fail, the whole process can be restarted no matter where it failed. As the first step of the process is to always delete all old data in the SQL tables and delete all Excel files that have been exported from SAP, so there is no risk of doing something twice that should not be done twice.

When a queue item process fails, the robot turns the Queue into failed and goes to process the next item. The process must be checked manually for the Queue item on how far it got. If the fail took place before the robot got to the SAP transaction ZFI_GL_UPLOAD, the queue item can be retried. If the fail happened after or during SAP transaction ZFI_GL_UPLOAD, the process cannot be retried and instead all the relevant logs and Excels are sent to the Accounts Payable Accountant team to check.

Both parts in the RPA Accrual process have their Try Catch error handling taking down the Exception Message, timestamps and is configured to take a screenshot of the current view. The screenshot will help with finding out where the fail happened and how far the process was. All this data is sent by the robot automatically by email to the RPA Admins.

5.8 Security Handling

To ensure that no one can insert malicious documents into the RPA Accruals runs or start the RPA Accrual process outside of the schedule, the robot can only be started from the Orchestrator. Each robot has its own usernames and passwords for all the systems it interacts with, which makes following all systems logs easy to see what changes the robots have done.

All passwords for the SQL databases are stored in the Orchestrators Assets and these are called on demand by the robots. SAP logins happen with single sign on (SSO), so if the RPA Process is run from another location it will not be able to access SAP with the robot usernames.

6 TESTING

In this chapter, the testing of the RPA Accruals between production versions is described. Due to the anonymity of Company X the testing data is not reported in great detail.

6.1 Testing Environment

We had two copies of the SAP Production system, SAP Quality and SAP Development. The robots have access to all three systems, but we only used SAP Quality for testing since SAP Development is used for all kinds of development and would give wrong kind of results for testing. SAP Quality system is updated twice per year with a copy from SAP Production. To get the best accuracy on testing we used both SAP Production and SAP Quality for testing. We first received Open Purchase Invoice documents from the SAP Production environment and built our Accrual cases out of these. Then we switched to SAP Quality and run the Accrual files through SAP transaction ZFI_GL_UPLOAD (Figure 15). This way we always had Open Purchase Invoice documents available for testing as SAP Quality will always have outdated Open Purchase Invoice documents.

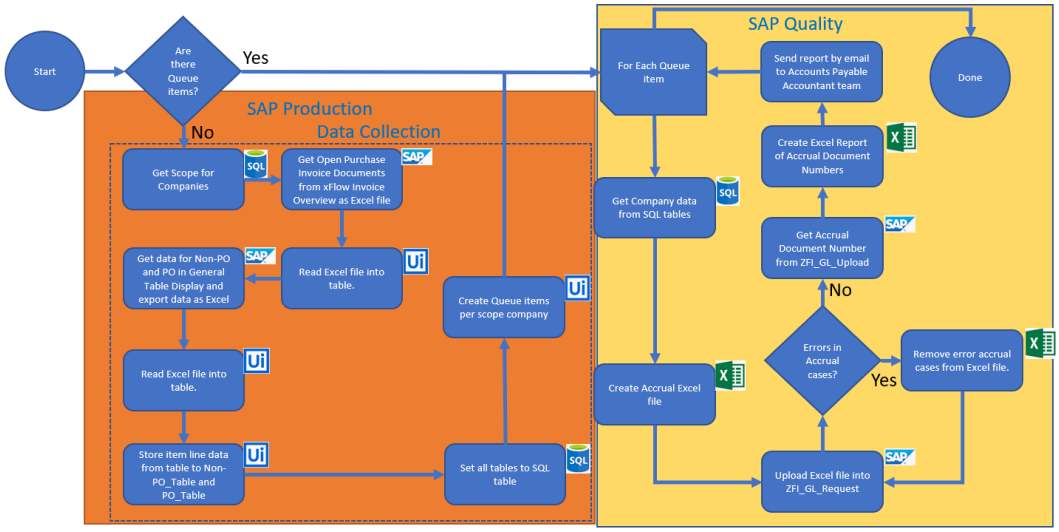

Figure 15. Testing RPA Accruals flow.

6.2 Test Cases

When we wanted to have specific test cases, we were able to do a full Data Collection from SAP Production. Once we have all the data, we went through the SQL tables Full_List, PO_Documents and non-PO_Documents to pick out all the specific test cases that we wanted to run through to see that RPA Accruals were working in an expected manner. We deleted all unwanted cases from the SQL tables.

6.2.1 Non-Purchase Order Open Purchase Invoice Document Test Cases

For non-PO Open Purchase Invoice documents, we had six different test cases. The test cases were from various SAP transaction xFlow Invoice overview steps and mixed Company codes, as they all have the same setup for getting accrual lines:

- Test 1, Accruable with cost center as cost object.
- Test 2, Accruable with WBS as cost object.
- Test 3, Accruable with Order as cost object.
- Test 4, Accruable with separate tax amount.
- Test 5, not Accruable with cost center as cost object without all lines filled in.
- Test 6, not Accruable with no lines filled in.

6.2.2 Purchase Order Open Purchase Invoice Document Test Cases

For non-PO Open Purchase Invoice documents, we had 6 different test cases. The test cases were from various SAP transaction xFlow Invoice overview steps and mixed Company codes, as they all have the same setup for getting accrual lines:

- Test 7, Accruable with cost center as cost object in Purchase Order.
- Test 8, Accruable with WBS as cost object in Purchase Order.
- Test 9, Accruable with Order as cost object in Purchase Order.
- Test 10, Accruable with separate tax amount.
- Test 11, not Accruable with cost center as cost object without all Purchase Order lines filled in.
- Test 12, not Accruable with no Purchase Order filled in.

6.3 Test Results

For the test data to give a good testing result, we picked 20 to 50 documents of each test case. The minimum number in each case was 20, as some of cases were uncommon and the maximum of 50 as some test cases were very common. The test data has 485 different documents from 10 different scope company codes.

Table 10. Test results for RPA Accruals run.

Test Case	Number of test documents	Number of different companies	Expected amount to be accrued	Number of accrued documents
Test 1	50	10	50	50
Test 2	20	5	20	20
Test 3	20	6	20	20
Test 4	30	6	30	30
Test 5	35	7	0	0
Test 6	50	10	0	0
Test 7	50	10	50	50
Test 8	50	10	50	50
Test 9	50	10	50	50
Test 10	40	10	40	40
Test 11	40	9	0	0
Test 12	50	10	0	0

6.4 Testing Conclusion

From the tests, we can see that the robot was able to accrue 310 documents out of the 485 Open Purchase Invoice documents in the test scope. The remaining 175 Open Purchase Invoice documents are from test cases 5, 6, 11 and 12, which were

not supposed to be accrued. The tests were successful on the part of processing the correct documents and leaving out the ones meant to be left out. All the amount and cost objects were observed, and all accrual documents had the correct data.

7 SUMMARY

The project was a success and the process has been running in production in the turn of every month change. During the first 20 runs of the process, the RPA admins would monitor the success of the RPA and fix any errors. As the RPA would run during the night this meant the RPA Admins would keep nightshifts once per month when the month end did not take place on a Friday or Saturday. On those days, the RPA could be run in the morning, as the Accountants would not come to work before Monday. At this moment, the process is reliable and fast enough so it can be started by scheduling through the orchestrator and the RPA Admins can sleep during the night. Instead, the RPA Admins will check the process at 6.00 in the morning, restart any single Queue items that have failed and still be ready before the deadline.

Very rough estimates had been done at the start of the project on how much time the robot would save for the Accounts Payable accountants. As this was a completely new process handled by WSSC, the estimate could not be used as a real measure. Once we were satisfied with the performance of the RPA Accrual process, a process time saved monitor module was added. The monitor module would create logs for each document that was handled and for each step an Accounts Payable Accountant would do to perform an accrual. Durations were clocked as to how long on average it would take an Accounts Payable Accountant to perform an Accrual. These were the times:

Table 11. Calculated times for each step in the Accruals process.

Sub process	Duration in seconds to process
Check if non-PO Document is accruable	8 seconds
Check if PO Document is accruable	12 seconds
Fill a non-PO Document into the Accrual file	60 seconds
Fill a PO Document into the Accrual file	120 seconds
Mark down a Goods Receipt	15 seconds
Post an Accrual file	300 seconds
Remove failed Accrual case	30 seconds

With the current setup, RPA Accruals can accrue a bit over 80% of all Open Purchase Invoice Documents, leaving less than a fifth to be handled manually. These were mostly Open Purchase Invoice documents that had been inserted into the system on the previous day and the Accounts Payable Accountants had not yet handled them in any way. With a theoretical 1000 Open Purchase Invoice Documents the robot would save 40 hours of Accounts Payable Accountant work (Figure 16). This time is also effective time without any breaks or distractions in between, so the time saving would be even higher.

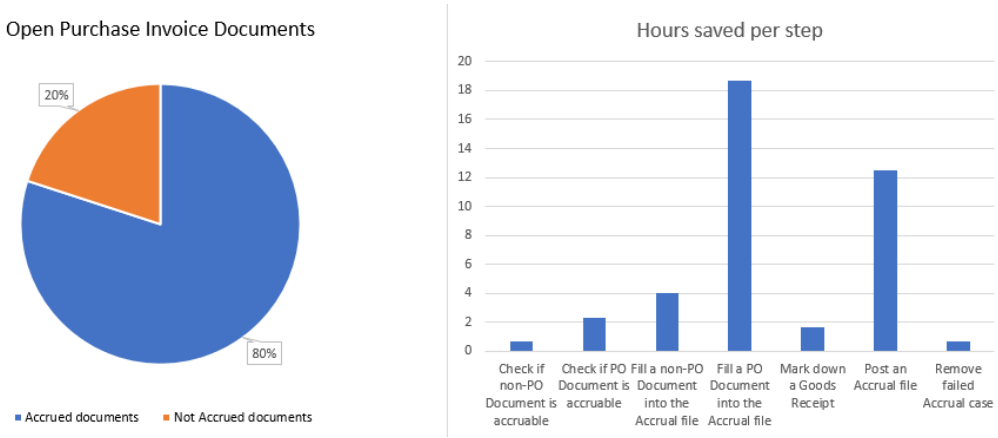

Figure 16. Volume of Open Purchase Invoice documents accrued and time saved.

8 CONCLUSION

This thesis project took a bit longer than was initially planned, as the RPA Accruals is an ongoing project, with hopes that it can one day accrue 100% of all Open Purchase Invoice documents. So eventually, the decision had to be made that the RPA Accruals was good enough for the thesis.

As was planned from the beginning the scope for RPA Accruals would increase over time, from a small scale, where we knew most of the Open Purchase Invoice documents had all necessary data filled in, to be able to handle more difficult companies and data eventually. In the beginning, we chose a handful of companies where both the Purchase Invoice document handling process and Purchase Order process were streamlined and on a good quality level. This enabled the first production version of RPA Accruals to accrue most of the Open Purchase Invoice documents that were in its scope, giving confidence to both the management and the Accounts Payable Accountants that the RPA would really be able to help with this process. At the time we had access to two robots, with one running RPA Accruals at the month change, while the other was busy with other month change RPA tasks. With the current scope, one robot was enough for RPA Accruals.

The scope was slowly increased by adding more companies; this also meant that the robot needed more time to run the whole scope. This was a major problem especially if the robot failed during the night, meaning that the code required fixing in the middle of the night and then the RPA Accruals were rerun, getting close to the 8:00 deadline. To solve this there were two options, get more robots which would have been the expensive solution as we did not need the capacity of more than 2 robots at the time. The other option was to make RPA Accruals work faster somehow. As the starting point was to get the Purchase Invoice Document header and line item data faster, getting them one by one was not working out. After some investigation, a way was discovered how to use the SAP transaction General Table Display, which gave access to the SAP tables behind all the normal transactions. Accessing tables is not something a normal Accountant can do, but for the RPA projects, authorizations was obtained to start using the SAP transaction General Table Display. With the help of the SAP tables, the time the robot needed to run RPA Accruals could be reduced significantly. At the same time now that we had

access to the SAP Tables we could improve on our existing RPAs and make new RPAs faster from the start.

With the SAP tables, it was possible to start involving other Open Purchase Invoice documents than just the ones in Release step. Open Purchase Invoice documents from steps that did not always have the necessary data filled in or even had faulty data filled in, started failing the RPA Accruals more and more. RPA Accruals were built as one long process where it would accrue one company and then move to the next company, meaning when it failed on an unexpected data, the whole process would be stopped and had to be restarted manually. This also meant that every time RPA Accruals failed, we would have to stop the whole process, modify the scope table to remove any companies that had already run from the table, all this delaying the whole process from running in a timely fashion. This caused problems and concern when the RPA Accruals were run.

In addition to this inconvenient manual error handling process the whole RPA Accruals was once again taking the full 8 hours to run. At this point, we already had access to three robots and natural solution would be to utilize them all for RPA Accruals. The first idea would be to split the scope table to two scope tables, but it would be hard to get two equally timed scopes. Instead, it was decided to start using the Queue feature, which we had in Orchestrator. In an earlier version of the Orchestrator the queue had some bugs, so we had not yet utilized the feature, but the current version was debugged. So, the RPA Accruals was split into one queue item per company in the scope, and this allowed for much easier error handling and allowed multiple robots to work on RPA Accruals at the same time. If one Queue item failed, the robot could just leave it as Failed in the queue and continue to the next. Then the RPA Admins could check the failed queue items, while the robots continued to run other queue items, which was an ideal error handling way of working as it does not delay the RPA Accruals.

RPA Accruals and RPA in general was an excellent thesis subject, it gave an opportunity to explore so many different aspects of the project. Normally when starting an RPA, it is done to an existing process, meaning process mapping and interviewing the people currently doing the process. With Accruals, I got free hands as the process was completely new, and the Accounts Payable Accountants were going

to them with some very heavy shortcuts. For this project, I was acting as both Lead Developer and Project Leader. RPA Accruals was a good learning experience, since it was the first RPA where I had to start using the Queue system effectively. This has helped immensely in other RPA projects. With the large variety of software developing aspects, such as different coding languages (.Net and VB), SQL database handling and design, code fine- tuning to run as fast and reliably as possible and environment setup, the project has been a good learning experience.

REFERENCES

/1/ Investopedia. 2019. Accrual Accounting. Accessed 10.2.2020. https://www.investopedia.com/terms/a/accrualaccounting.asp

/2/ Microsoft. 2020. .NET documentation. Accessed 7.2.2020. https://docs.microsoft.com/en-us/dotnet/

/3/ SAP Help Portal. 2001. Goods Receipt. Accessed 8.2.2020. https://help.sap.com/doc/saphelp_46c/4.6C/en-US/c6/f83f084afa11d182b90000e829fbfe/content.htm?no_cache=true

/4/ SAP. 2019. Intelligent ERP for Industrial Manufacturing. Accessed 8.2.2020. https://www.sap.com/products/s4hana-erp.html

/5/ w3schools.com. 2020. SQL Tutorial. Accessed 9.2.2020. https://www.w3schools.com/sql/default.asp

/6/ UiPath. 2019. Robotic Process Automation (RPA). Accessed 2.2.2020. https://www.uipath.com/rpa/robotic-process-automation

/7/ UiPath. 2019. Be an Orchestrator of Robots. Accessed 2.2.2020. https://www.uipath.com/product/orchestrator

/8/ UiPath. 2019. Your Automation Canvas: UiPath Studio. Accessed 2.2.2020. https://www.uipath.com/product/studio

/9/ Company X. 2020. Shared Service Center. Accessed 5.2.2020.