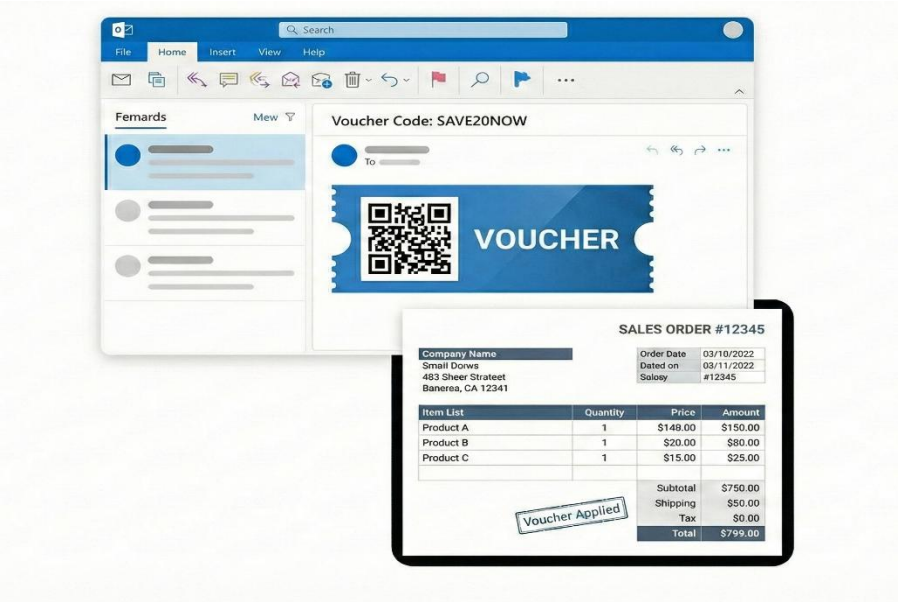
VOUCHERS BOT

UiPath RPA Automation Project

Technical Solution Documentation



Version 3.0 (Final)

Melvin Ramos

RPA Developer

Contents

Summary

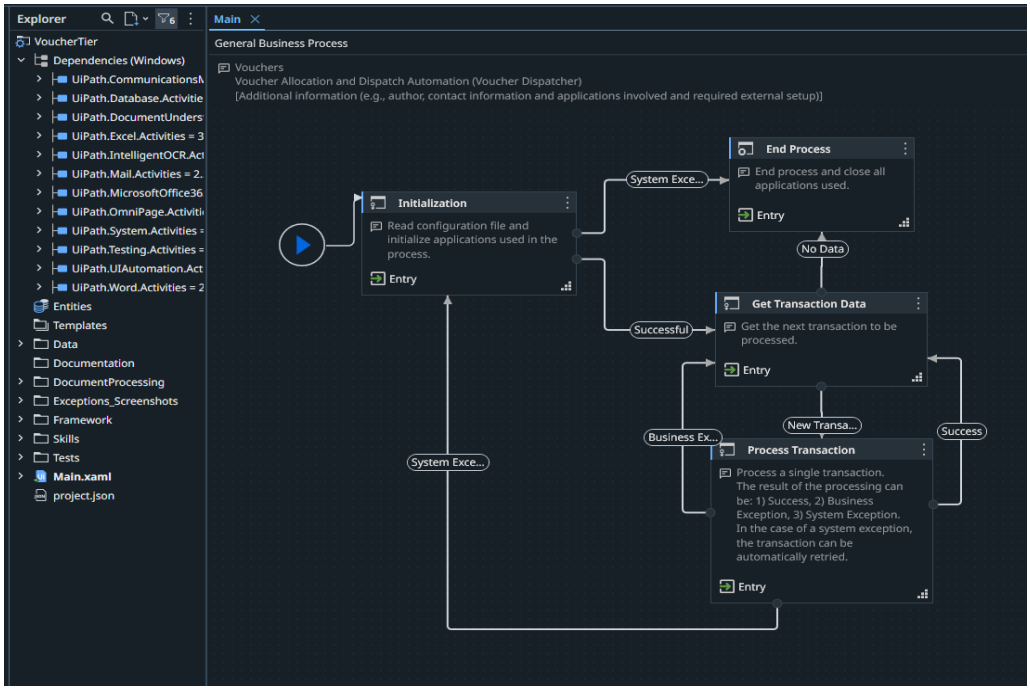
The **Voucher** robot is an automated solution designed to manage the reception of **Sales Orders**, inventory verification, and the dispatch of digital codes. The system eliminates manual intervention, reduces processing time from hours to minutes, and guarantees inventory integrity through transactional database validations.

Key Benefits

- **Speed:** Capable of processing multiple orders in minutes.
- **Data Integrity:** Prevents double-selling by locking records in SQL (Status 0 to 1).
- **Auditability:** Maintains a detailed historical record (VoucherAssignments) for total traceability.
- **Resilience:** Proactive notification system with visual evidence (screenshots) in case of critical failures.

Solution Architecture

The solution is built upon the UiPath **Robotic Enterprise Framework (REFramework)**, customized for a **Batch Processing** model.



Component Diagram

1. **Input:** Emails in Outlook with attached PDF files (**Sales Orders**).
2. **Processing:** UiPath Robot (Background Process).
3. **Backend (Data):** Microsoft SQL Server (Database VoucherOrderDB).
4. **Output:** Confirmation emails containing the generated Voucher PDF and the original Sales Order.

System Prerequisites

- **Software:** UiPath Studio/Robot, Microsoft Outlook, Microsoft Word, Microsoft Excel, SQL Server.
- **Access:** Read/Write permissions on the Database and access to local folders.

Detailed Workflow Description

The process follows the standard REFramework stages but implements specific logic for mass ingestion:

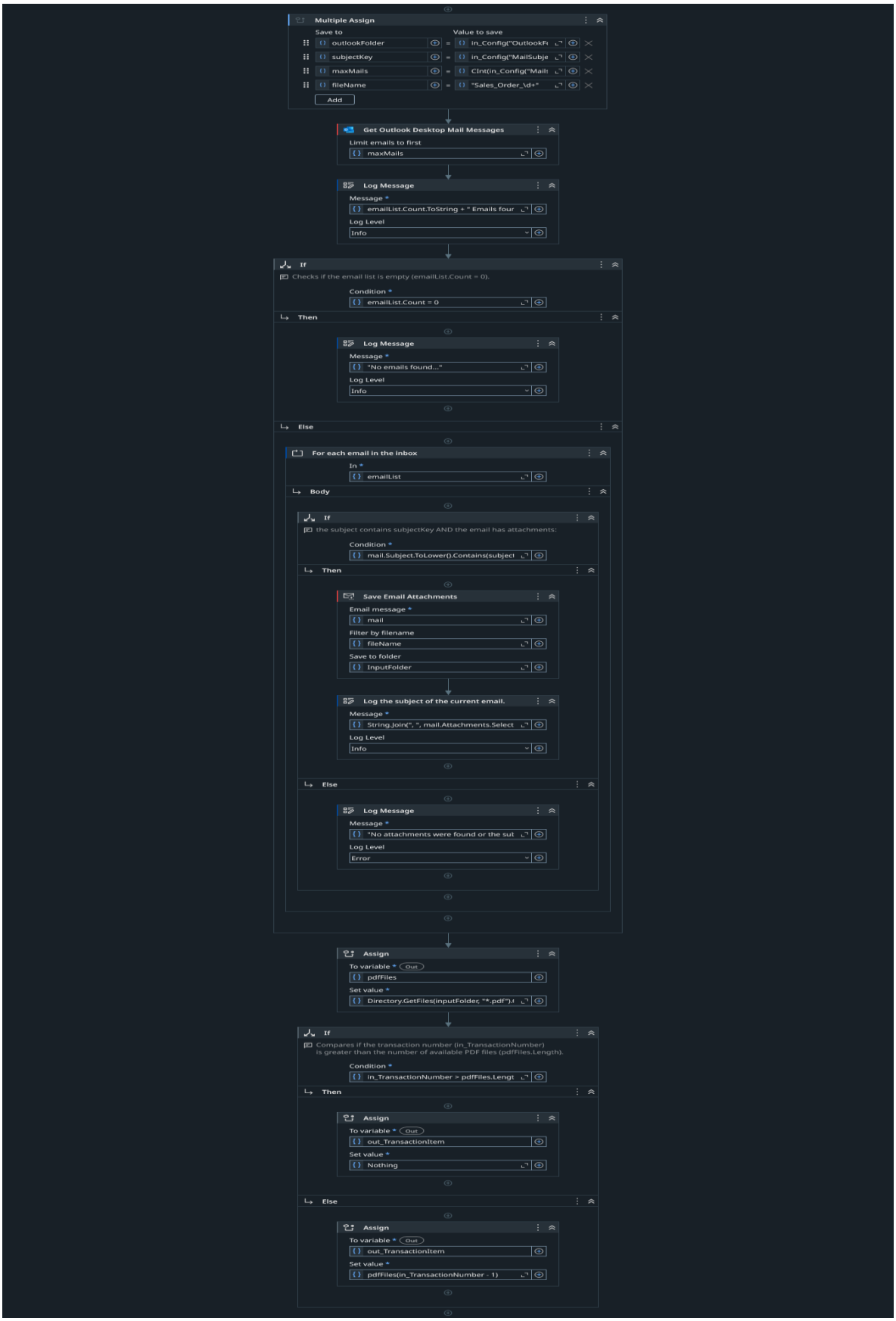
A. Initialization (InitAllApplications)

- Loads the configuration file Config.xlsx.
- **Environment Cleanup (Clean_Folders.xaml):** To ensure a clean execution, the robot removes residual files in the following folders:
  - Data\Input
  - Data\Output
  - Data\Temp
  - Data\Processed_Folder

B. Get Transaction Data (GetTransactionData)

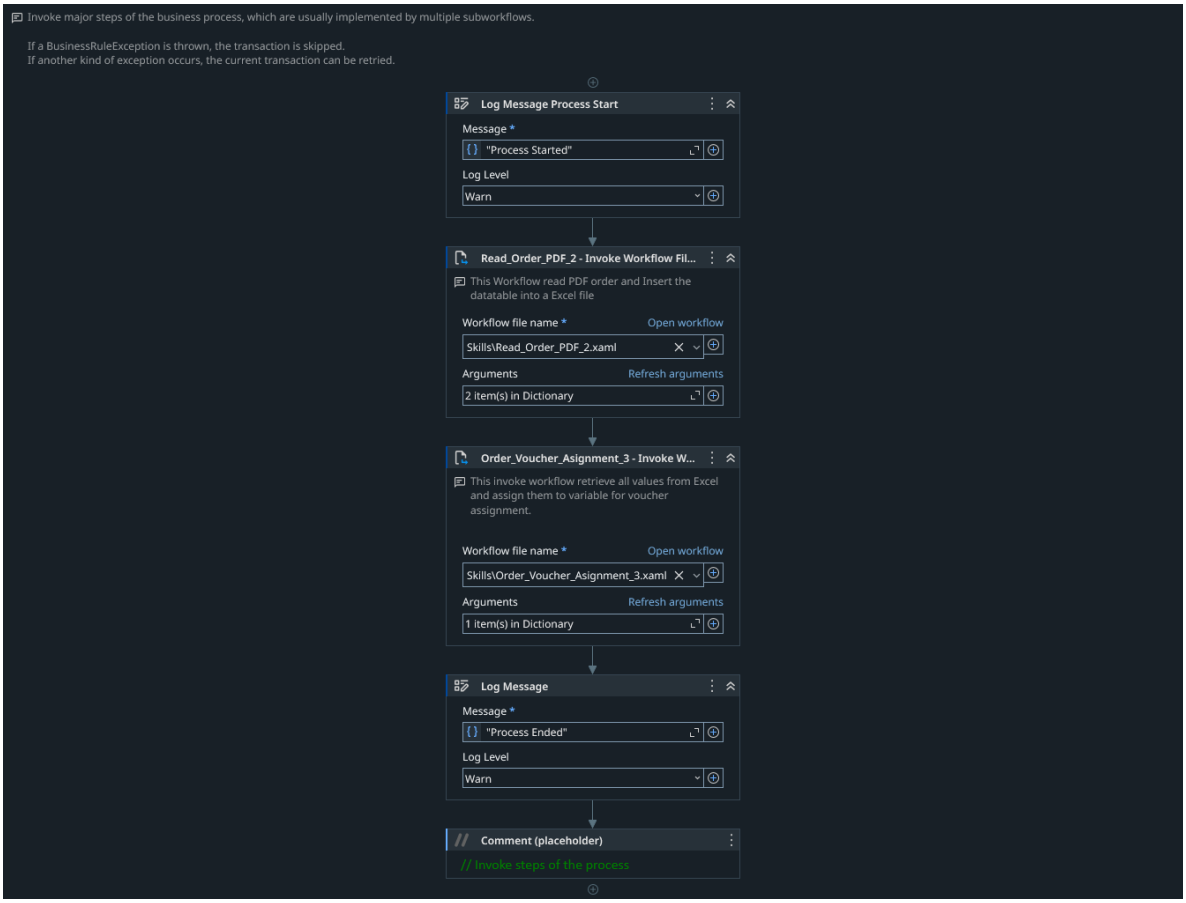
The robot operates using a **Mass Ingestion** logic:

1. **Download:** Connects to Outlook, filters unread emails (e.g., Subject: "Sales Order"), and downloads **all** PDF attachments simultaneously to the Data\Input folder.
2. **Transaction Definition:** The TransactionItem is defined by the existence of files in the Input folder.
3. **Trigger:** If the folder contains files, a single transaction starts to process the entire batch. If the folder is empty, the process ends.


Multiple Assign
Save to
Value to save
outlookFolder
in_Config("OutlookF
subjectKey
in_Config("MailSubje
maxMails
CInt(in_Config("Mails
fileName
"Sales_Order_\d+"
Add
Get Outlook Desktop Mail Messages
Limit emails to first
maxMails
Log Message
Message *
emailList.Count.ToString + " Emails four
Log Level
Info
If
Checks if the email list is empty (emailList.Count = 0).
Condition *
emailList.Count = 0
Then
Log Message
Message *
"No emails found..."
Log Level
Info
Else
For each email in the inbox
In *
emailList
Body
If
the subject contains subjectKey AND the email has attachments:
Condition *
mail.Subject.ToLower().Contains(subject
Then
Save Email Attachments
Email message *
mail
Filter by filename
fileName
Save to folder
InputFolder
Log the subject of the current email.
Message *
String.Join(", ", mail.Attachments.Select
Log Level
Info
Else
Log Message
Message *
"No attachments were found or the sut
Log Level
Error
Assign
To variable * Out
pdfFiles
Set value *
Directory.GetFiles(inputFolder, "*.pdf").
If
Compares if the transaction number (in_TransactionNumber)
is greater than the number of available PDF files (pdfFiles.Length).
Condition *
in_TransactionNumber > pdfFiles.Lengt
Then
Assign
To variable * Out
out_TransactionItem
Set value *
Nothing
Else
Assign
To variable * Out
out_TransactionItem
Set value *
pdfFiles(in_TransactionNumber - 1)

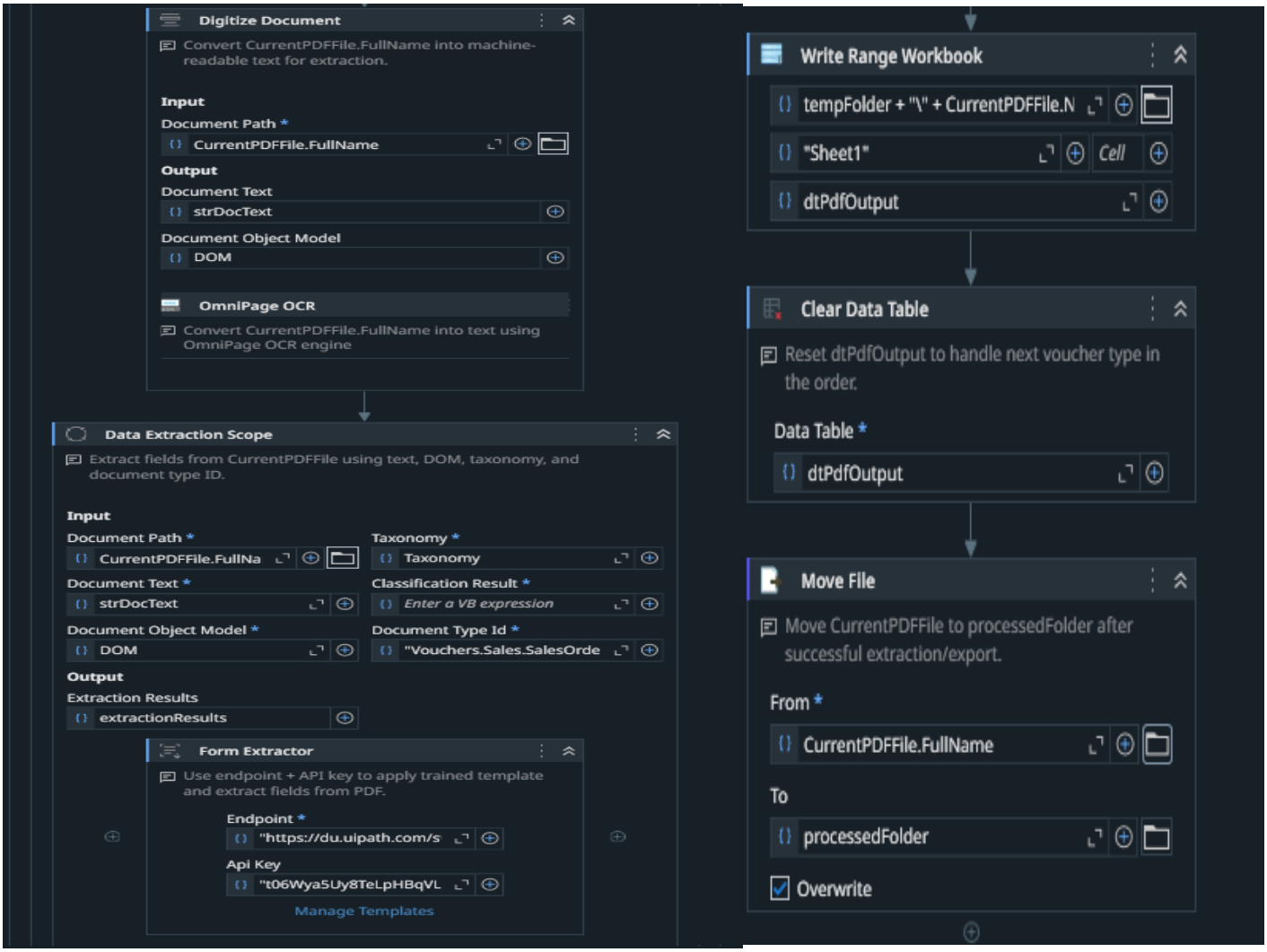
C. Processing (Process.xaml)

In this stage, the business logic is executed for the **entire batch of files** at once:



1. **Mass Read and Extraction (Read_Order_PDF_2.xaml):**
   - The robot iterates over **all** PDF files present in Data\Input.
   - **Document Understanding:** Digitizes each PDF and extracts: Client Name, Sales Order Number, Email, and Item Table.
   - **Excel Generation:** The extracted data is structured and saved temporarily in an Excel file in Data\Temp.
   - **File Movement:** Immediately after extraction, the original PDF is moved to Data\Processed_Folder.

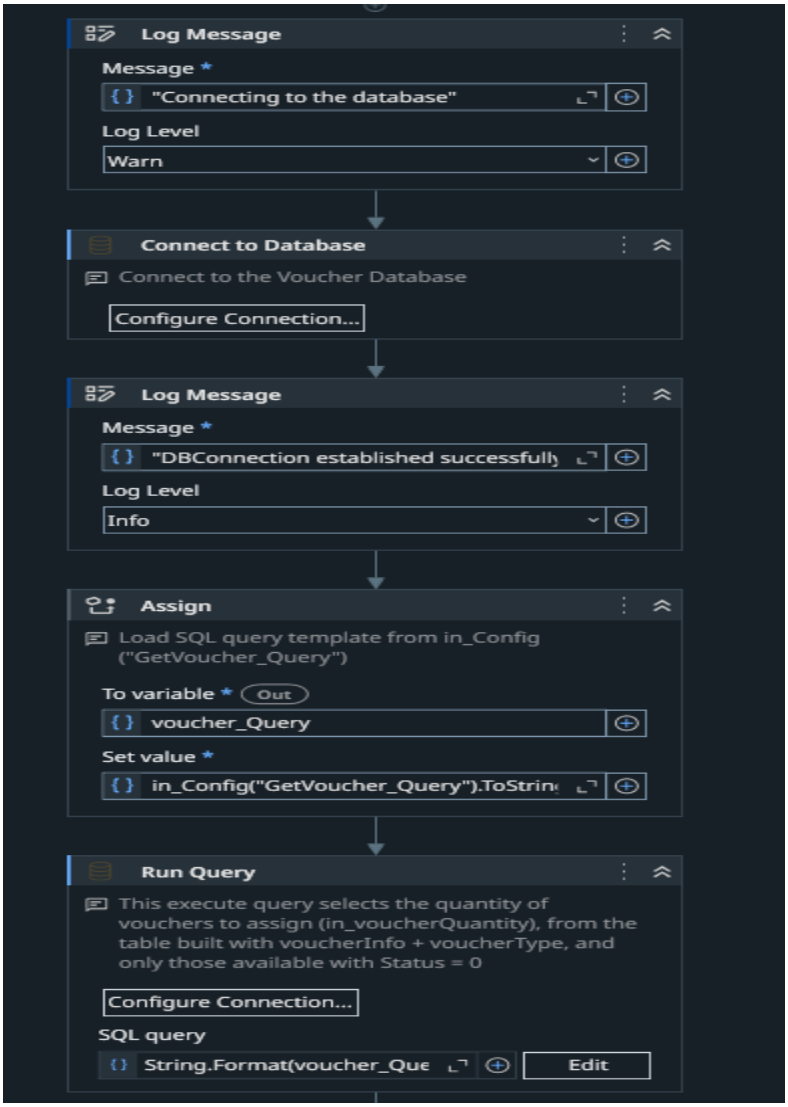
Note: This activity empties the Input folder in a single execution.



2. **Voucher Assignment (Order_Voucher_Asignment_3.xaml):**
   - The robot reads the generated temporary Excel files.
   - Identifies the requested **Voucher** type and maps the name to the corresponding SQL table using the Config file.

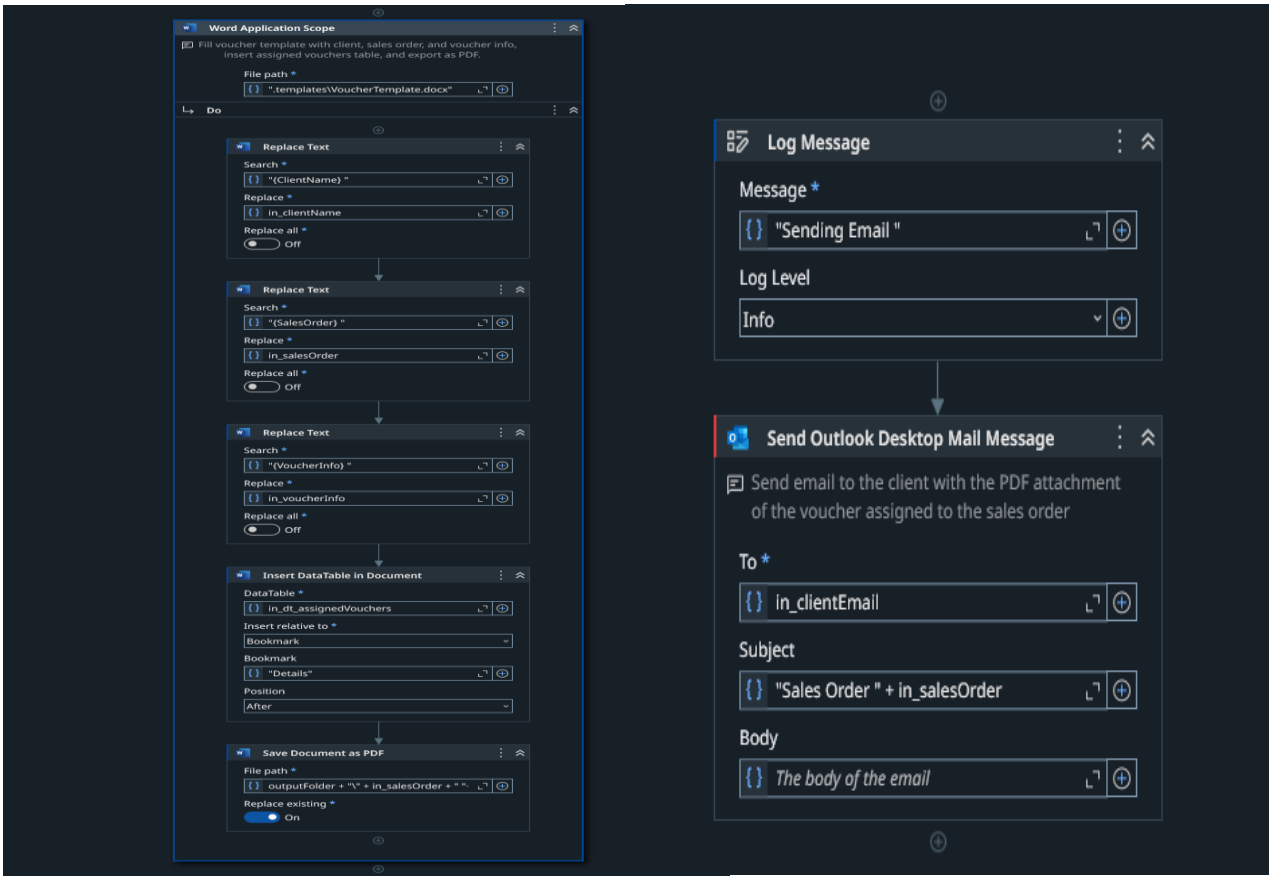
3. **SQL Interaction (Get_Voucher_DB_4.xaml):**
   - **Stock Query:** Executes the GetVoucher_Query to search for codes with Status = 0.
   - **Validation (Business Rule):**
     - *If stock exists:* Reserves the codes.
     - *If NO stock:* Throws a **Business Rule Exception** and sends an alert to the support team (Send_Email_Notification).

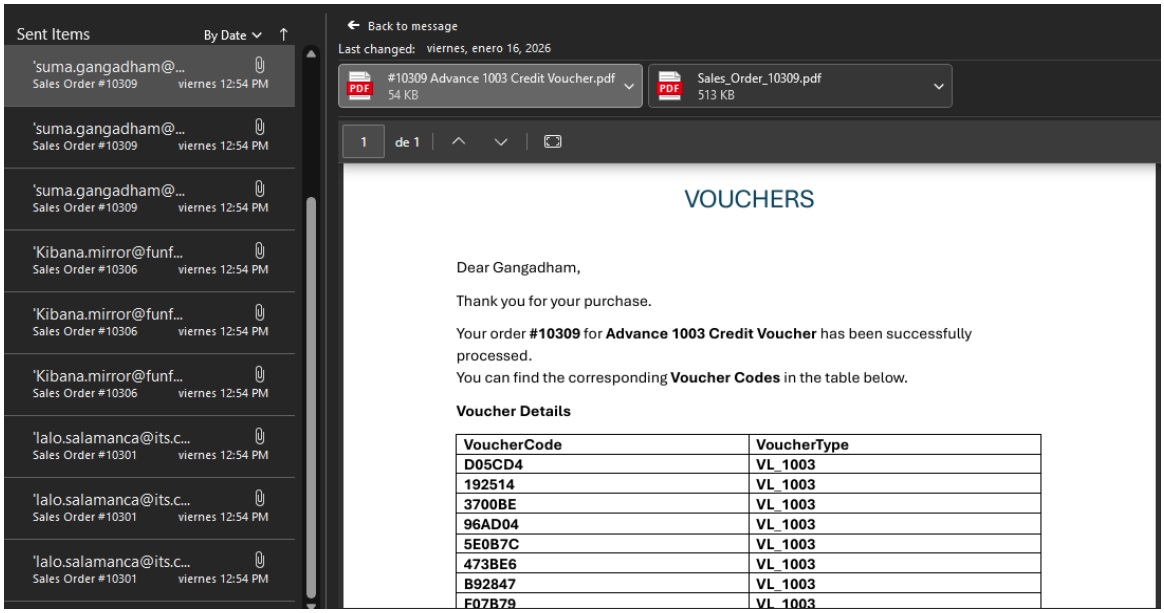
- **Update:** Executes an UPDATE to change the code status to 1 (Used).
- **Audit:** Inserts the sale record into the VoucherAssignments table.



4. **Document Generation (Create_Doc_PDF_5.xaml):**
   - Opens the VoucherTemplate.docx template.
   - Injects the client data and the table of assigned codes.
   - Exports the final document to PDF.

5. **Final Dispatch (Send_Email_Confirmation.xaml):**
    - Sends the email to the client with **two attachments**:
        1. The newly generated **Voucher PDF**.
        2. The original **Sales Order PDF** (for reference).





Database Structure (SQL Server)

The system uses the VoucherOrderDB database, organized into a 3-tier relational model: **Master**, **Inventory**, and **Audit**.

1. **Master Price Catalog (Master Data)**

Centralizes product definitions and current pricing. Serves as the source of truth for descriptions.

- **Table Name:** VoucherCatalog
- **Columns:** VoucherType (PK), Description, CurrentPrice.

	VoucherType	Description	CurrentPrice
1	VL_1001	Starter 1001 Credit Voucher	55.00
2	VL_1002	Starter 1002 Credit Voucher	70.00
3	VL_1003	Advance 1003 Credit Voucher	99.00

2. **Inventory Tables (Transactional)**

Manages the actual stock. Segregated by voucher type for performance and organization.

- **Real Table Names:**
  - Starter1001CreditVoucherVL_1001
  - Starter1002CreditVoucherVL_1002
  - Advance1003CreditVoucherVL_1003
- **Structure:** VoucherCode (PK), RefillDate, Status (0/1), VoucherType.

	Id	VoucherCode	Status	RefillDate	VoucherType
28	28	D05626	1	2026-01-14 18:09:14.843	VL_1001
29	29	E8A08E	1	2026-01-14 18:09:14.843	VL_1001
30	30	D91DAA	0	2026-01-14 18:09:14.843	VL_1001
31	31	C7F330	0	2026-01-14 18:09:14.853	VL_1001
32	32	E4E6C4	0	2026-01-14 18:09:14.853	VL_1001
33	33	8B9488	0	2026-01-14 18:09:14.853	VL_1001
34	34	C1F5CE	0	2026-01-14 18:09:14.853	VL_1001
35	35	76203D	0	2026-01-14 18:09:14.853	VL_1001
36	36	6FAF60	0	2026-01-14 18:09:14.857	VL_1001
37	37	FE7F08	0	2026-01-14 18:09:14.857	VL_1001
38	38	FF5FDD	0	2026-01-14 18:09:14.857	VL_1001
39	39	7E20FA	0	2026-01-14 18:09:14.857	VL_1001

3. **Audit Table (History)**

A centralized table that records every successful transaction for full traceability.

- **Table Name:** VoucherAssignments
- **Columns:** VoucherCode, SalesOrder, ClientName, ClientEmail, AssignmentDate, VoucherType, VoucherCost.

	Id	VoucherCode	SalesOrder	ClientName	ClientEmail	AssignmentDate	VoucherCost	VoucherType
9	9	534000	#10301	IT Services	lalo.salamanca@its.com	2026-01-16 12:53:53.270	99.00	VL_1003
10	10	70D8B1	#10301	IT Services	lalo.salamanca@its.com	2026-01-16 12:53:53.270	99.00	VL_1003
11	11	65394B	#10301	IT Services	lalo.salamanca@its.com	2026-01-16 12:53:53.270	99.00	VL_1003
12	12	45002A	#10301	IT Services	lalo.salamanca@its.com	2026-01-16 12:53:53.270	99.00	VL_1003
13	13	83F3F4	#10306	Mirror	Kibana.mirror@funfac...	2026-01-16 12:53:54.090	55.00	VL_1001
14	14	A592E2	#10306	Mirror	Kibana.mirror@funfac...	2026-01-16 12:53:54.090	55.00	VL_1001
15	15	9408AD	#10306	Mirror	Kibana.mirror@funfac...	2026-01-16 12:53:54.090	55.00	VL_1001
16	16	C9F048	#10306	Mirror	Kibana.mirror@funfac...	2026-01-16 12:53:54.090	55.00	VL_1001
17	17	CB527A	#10306	Mirror	Kibana.mirror@funfac...	2026-01-16 12:53:54.090	55.00	VL_1001
18	18	78FFFF	#10306	Mirror	Kibana.mirror@funfac...	2026-01-16 12:53:55.097	70.00	VL_1002
19	19	778897	#10306	Mirror	Kibana.mirror@funfac...	2026-01-16 12:53:55.097	70.00	VL_1002
20	20	7EC150	#10306	Mirror	Kibana.mirror@funfac...	2026-01-16 12:53:55.097	70.00	VL_1002

Configuration (Config.xlsx)

The robot's behavior is customizable through the configuration file, avoiding hardcoded values.

Here you can see some of the most important ones:

- **Settings Tab (General):**
  - **Paths:** InputFolder, OutputFolder, TempFolder, ProcessedFolder.
  - **Connection:** DBConnToStr (SQL Connection String).
  - **SQL Queries:** Parametrized queries for stock check and status updates.
- **Settings Tab (Error Notifications):**
  - ErrorMessage_Subject: Subject for critical alerts (Default: **"ERROR DETECTED - Action Required"**).
- **Assets Tab:**
  - Credentials and Outlook configuration.

Exception Handling

The robot implements an **Active Notification with Evidence** system to ensure operational continuity:

Error Type	Cause	Robot Action
System Exception	Critical technical failure (SQL crash, Outlook closed, I/O Error, Network Error).	**1. Logging:** Generates a "Fatal" log record. **2. Evidence:** Automatically takes a **Screenshot** of the desktop at the moment of the error. **3. Notification:** Sends an email to the Admin with the subject **"ERROR DETECTED - Action Required"**, attaching the screenshot. **4. Termination:** Stops processing to prevent data corruption.
Business Exception	**Insufficient Stock** (Status=0 exhausted).	**No retry.** Immediately sends an alert email to the support team indicating "Refill Required" and marks the transaction as a Business Exception.
Business Exception	Unknown Voucher or invalid format.	Sends an "Invalid Voucher" alert for manual review.

Deployment Instructions

1. **Database:** Create Master, Inventory, and Audit tables on the destination server.
2. **Configuration:** Update Config.xlsx with production values (DB Connection String and Admin Email).
3. **Templates:** Verify VoucherTemplate.docx is in the project root.
4. **Execution:** Publish to UiPath Orchestrator or run via UiPath Assistant.