UiPath

UiPath Automation

Walkthrough

# Walkthrough – Calculate Client Security Hash

- Start with the REFramework template.
  - We start off with a simple implementation to demonstrate the REFramework without using Orchestrator Queues.
  - Please have a quick glance at the input data (queue item list); we can extract the whole table with the Data Scraping wizard.
  - The transaction item is a data row from the list. The same approach applies when the input is a datatable extracted from Excel spreadsheets, CSV files and Databases.
- It's a good practice to keep the values that are prone to changes in the configuration file. Open the Data\Config.xlsx workbook and switch to the Settings sheet.
  - Add settings for **System1 URL** and **SHA1 Online URL**.
  - The System1 application requires authentication; we'll use Orchestrator Assets to store the credentials for System1. Add another setting, System1_Credential, to store the Credential name.
  - Add an Asset of the Credential type and write the username and password for System1. Make sure that the Asset name matches the value of the System1_Credential setting.
- Switch to the **Constants** sheet in the **Config** workbook and set the value of MaxRetryNumber to 2. This parameter controls how many times the framework tries to process a work item when it fails with an application exception, before moving on to next one.
- Make the following changes in the framework.
  - The **TransactionItem** variable in the **Main** file should be of the System.Data.DataRow type, as we are extracting the entire table to process it one row at a time. You should also change the argument type in the **GetTransactionData**, **Process** and **SetTransactionStatus** workflows to match the TransactionItem type.
  - Remove the three **SetTransactionStatus** activities from the SetTransactionStatus workflow as we are not using the transaction functionality provided by Orchestrator.
- We are using two applications in this exercise, **ACME System1** and **SHA1-Online.com**. Create two folders, "System1" and "SHA1Online", in the solution root directory, and use them for the workflows created for the two applications.

- Create a blank sequence in the System1 folder, for the System1 login process. We want to create a reusable component which can be used with many different credentials.
  - It's a good practice to start your new sequence with a short annotation meant to explain the purpose of the workflow. That's what we're going to do in all the future files. The annotation starts with the description that includes the arguments used, a Precondition, and a Post action.
  - Create two In Arguments - one for **System1 URL,** and the other for **System1 Credential**.
  - Invoke the Framework\GetAppCredential workflow file. Use the System1 Credential argument as input. Create two variables to store the two output values – username and password.
 - This is what the sequence should look like:
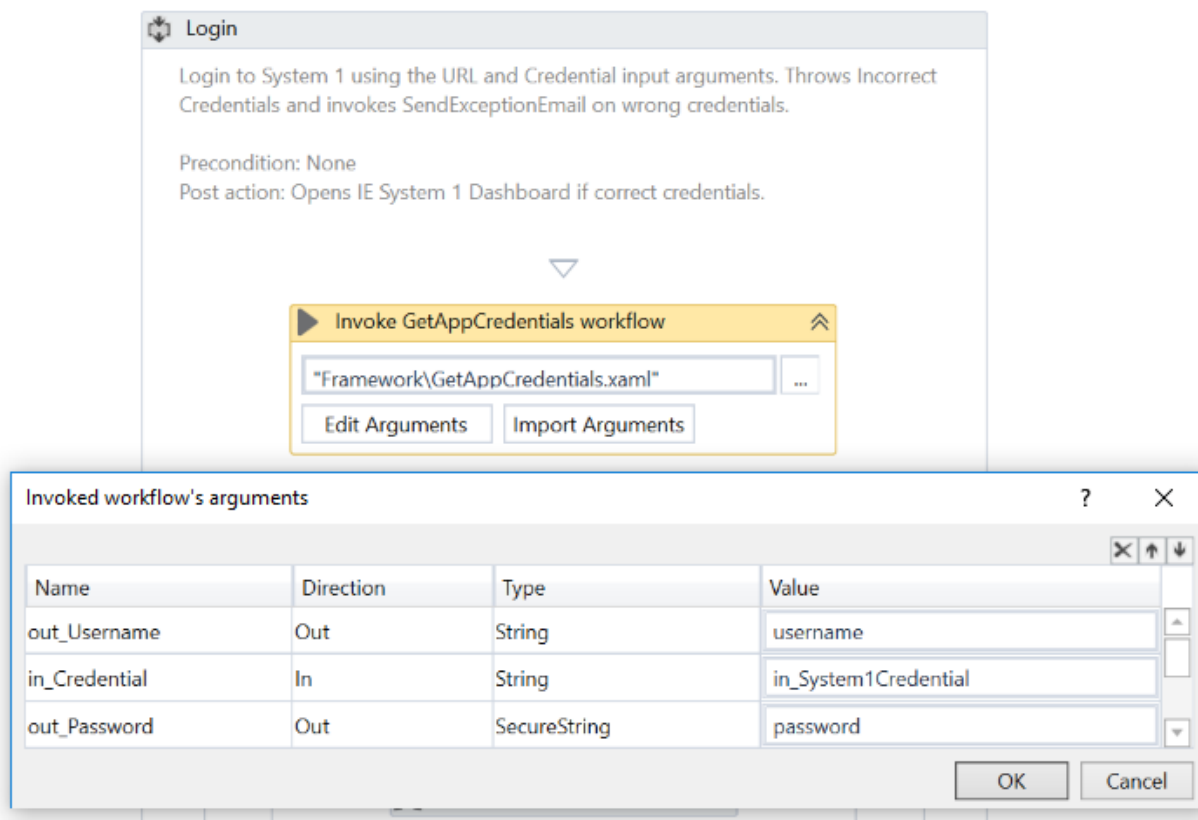
- Use an **Open Browser** activity to navigate to System1 URL. By default, Internet Explorer is used unless the **BrowserType** property is edited.
- Optionally, in the **Do** sequence in **Open Browser**, add a **Maximize Window** activity for better visibility when performing the next three activities.
- Use a **Type Into** activity to provide the username. Make sure the **SimulateType** checkbox in the **Properties** panel is selected.

o Use a **Type Secure Text** activity to enter the password. Make sure the **SimulateType** checkbox in the **Properties** panel is selected.
o Use a **Click** activity to select the Log In button. Make sure the **SimulateClick** checkbox in the **Properties** panel is selected.
o This is what the Open Browser activity should look like.
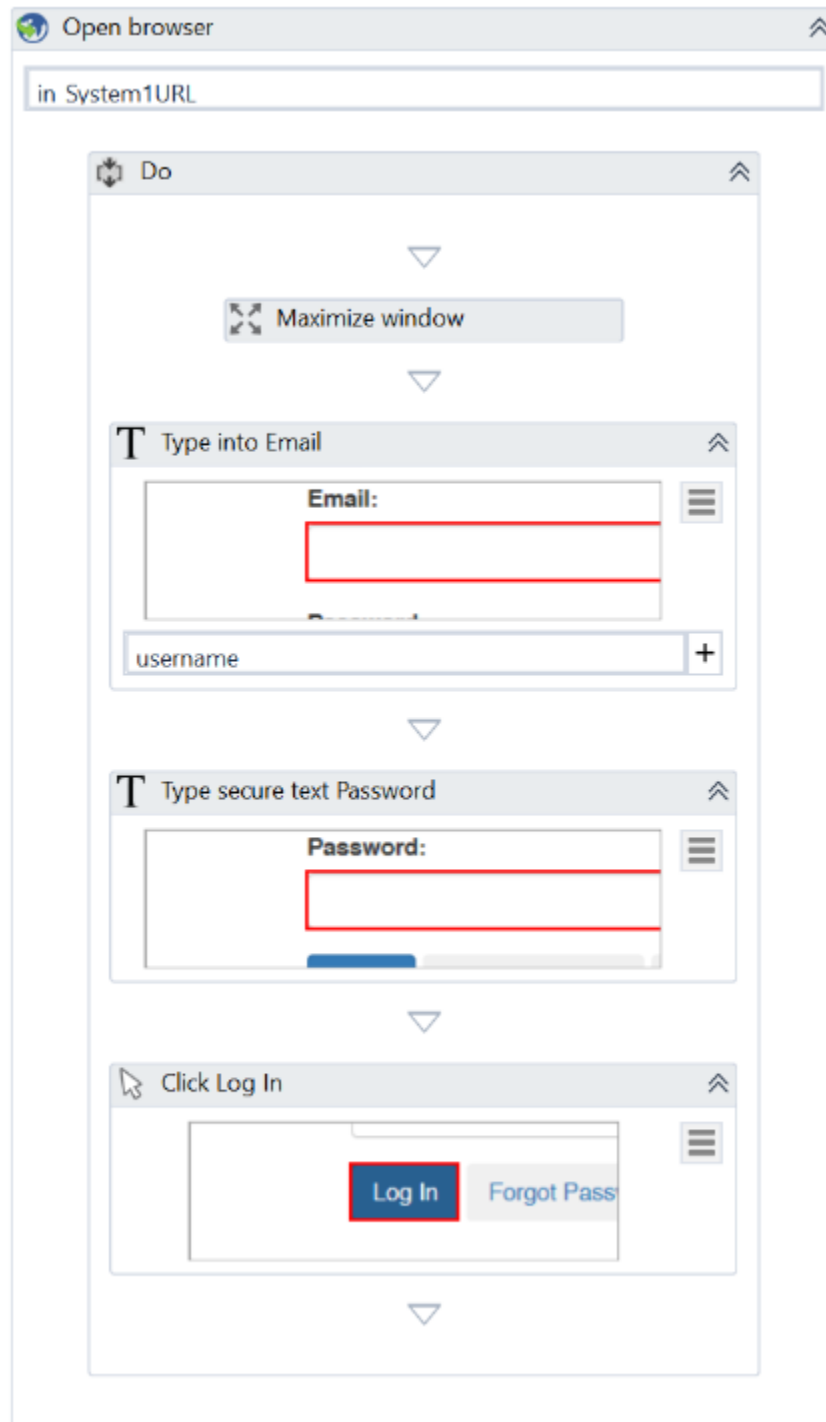
o   It's always a good practice to think about the exceptions that might occur during the execution of our process. In this case, we should check if the login was successful or not. This requirement is also presented in the PDD file. Try to log in with wrong credentials and notice the difference.

o   Use an **Element Exist** activity to check whether the login succeeded by searching for an element that is shown only in this case.
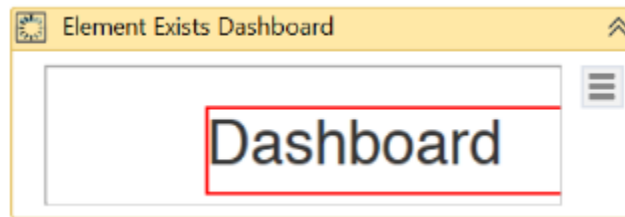
o   Use an **If** activity to check if the login attempt failed. If so, perform the actions below.
  ▪   Send an email with the exception. It is recommended to create a separate reusable workflow and invoke it in the **Then** section.
  ▪   Close the error message through a **Click** activity.
  ▪   Throw an exception for Incorrect Credentials supplied to System1 to stop the process.

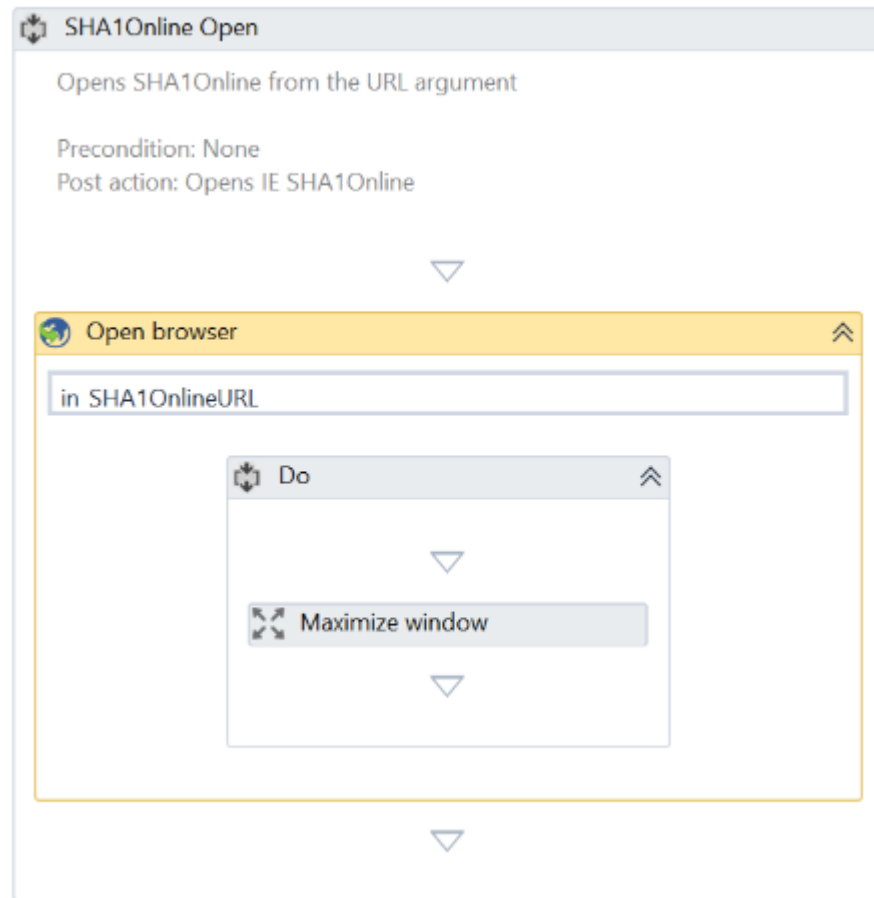o   This is what the **If** activity should look like:
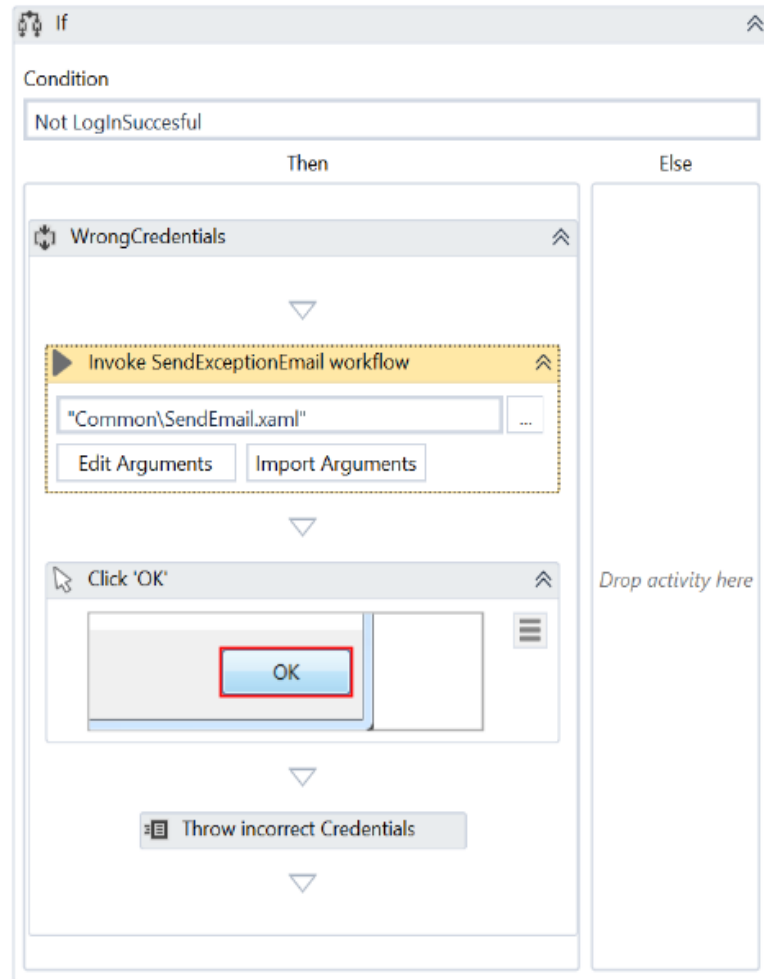
- o Unit test the created file using the default values of the arguments.
- Next, we'll create the workflow to log out and close System1. Create a blank sequence in the System1 folder.
    - o The new sequence should start with an annotation.
    - o Use the **Attach Browser** activity to attach to the browser window containing the System1 application. We can extract from here the first prerequisite for this workflow – System1 application is opened.
    - o Inside the attached browser activity, drag and drop a **Click** activity. Set the target to the Log Out button. This is the second prerequisite for this workflow – the Log Out button is accessible.
    - o Use a Close Tab activity to close the System1 browser window.
   - o This is how the sequence should look:
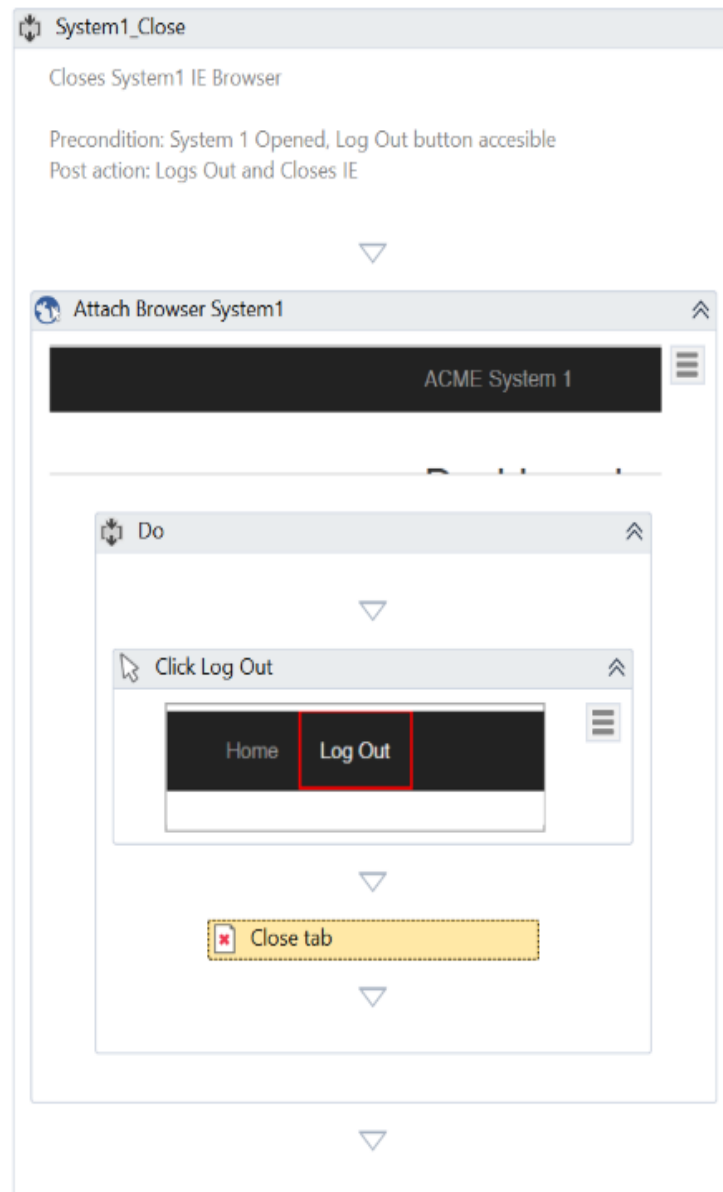
- Next, we'll do the same for the other application used in the SHA1 Online process.
- Create a blank sequence workflow in the SHA1Online folder to open the application. This sequence is simpler, as the application does not require authentication.
  - o Of course, we start with an annotation.
  - o The URL of the application should be passed to the workflow using an argument, to make the project easier to maintain. The URL value is stored in the process configuration file.
  - o Use an **Open Browser** activity with the created argument as input URL. By default, Internet Explorer is used unless the **BrowserType** property is edited.
  - o Optionally, you could use a **Maximize Window** activity on the browser window.
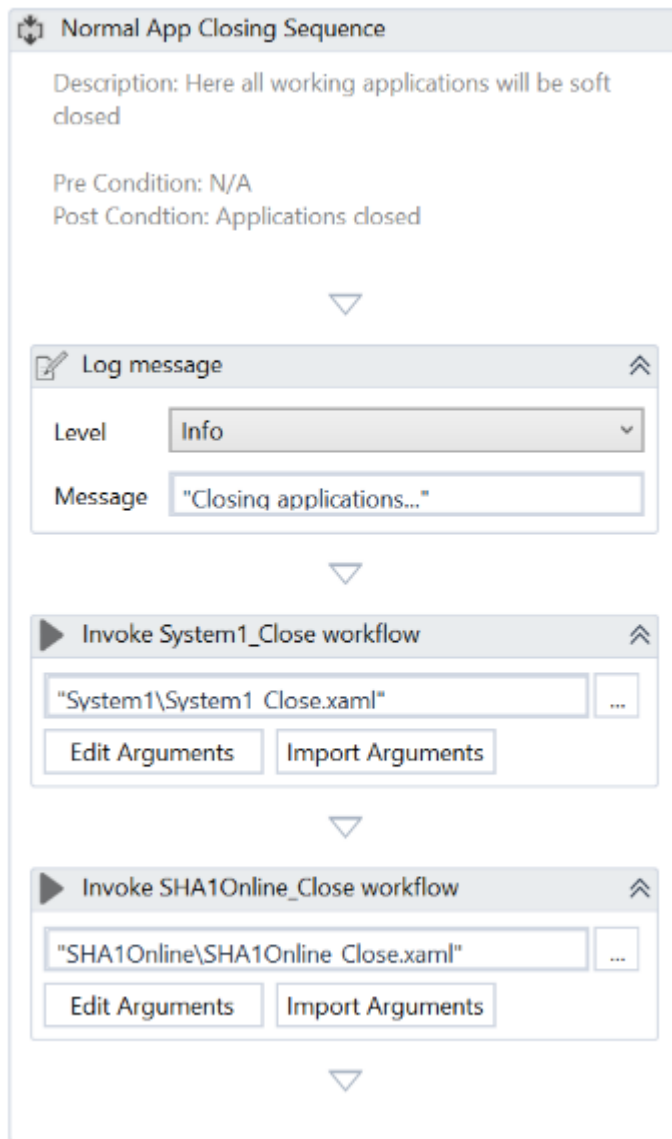
o   This is what the sequence should look like:

SHA1Online Open

Opens SHA1Online from the URL argument

Precondition: None
Post action: Opens IE SHA1Online

▽

Open browser   ≫

in SHA1OnlineURL

Do   ≫

▽

Maximize window

▽

▽

o

- Create a blank sequence to close the SHA1Online application. This one should be straightforward.
- Now that we have the workflows that open and close both applications, we can do the changes in the framework initialization and closing parts.
- Open the **Framework\InitAllApplications** workflow
  - o   Drag and drop the System1_Login file. An **Invoke Workflow File** activity is created automatically.
  - o   Click **Import Arguments** and bind the values to those in the Config file.
  - o   This is what **Invoke Workflow File** activity looks like:
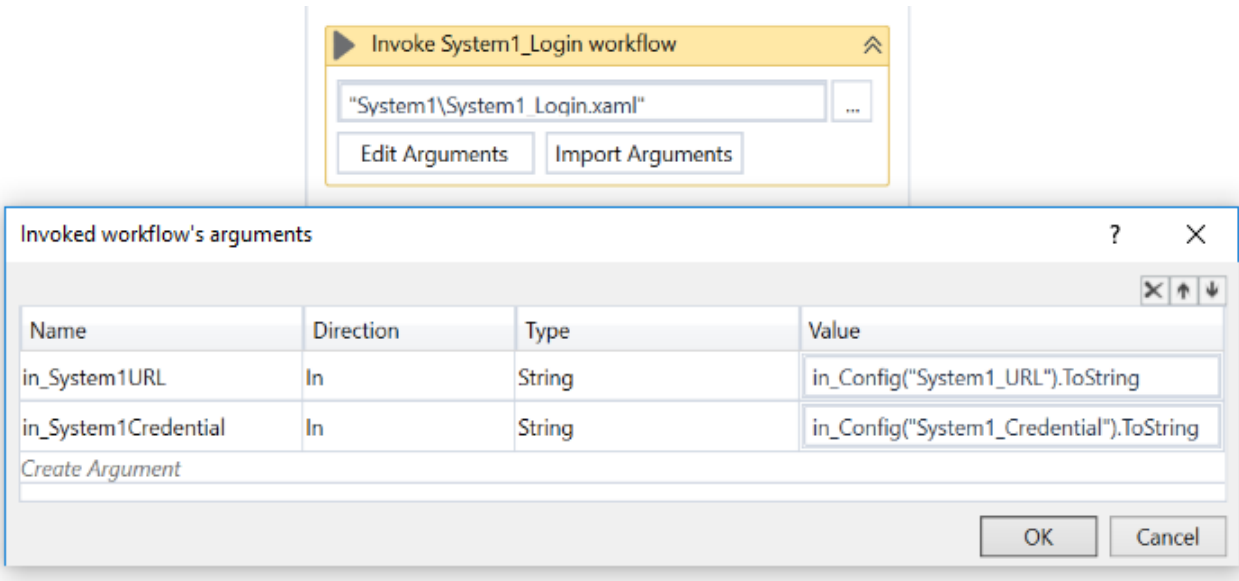
- o Drag and drop the SHA1Online_Login workflow.
- o Click **Import Arguments** and bind the URL argument to the value in the Config file.
- Open **Framework\CloseAllApplications**.
    - o Drag and drop the two workflows created to close the System1 and SHA1Online applications.
    - o This is what the resulting sequence looks like:

- Open **Framework\KillAllProcesses.**
  - Both our applications are hosted inside a web browser. If the default value of the **BrowserType** property in the **Open Browser** activities has not been changed, Internet Explorer is used.
  - Use a **Kill Process** activity and set the Process Name to "iexplore".
- Create a blank sequence workflow in the System1 folder to navigate to **Work Items** in the System1 application. We want this action to be included in a separate workflow, as we are going to navigate to Work Items in other projects as well.
 - Add the necessary annotation.
  - Use a **Click** activity on the Work Items button. Make sure to use a full selector, as this activity is not inside an Attach Browser scope.
 - This is what the project should look like:
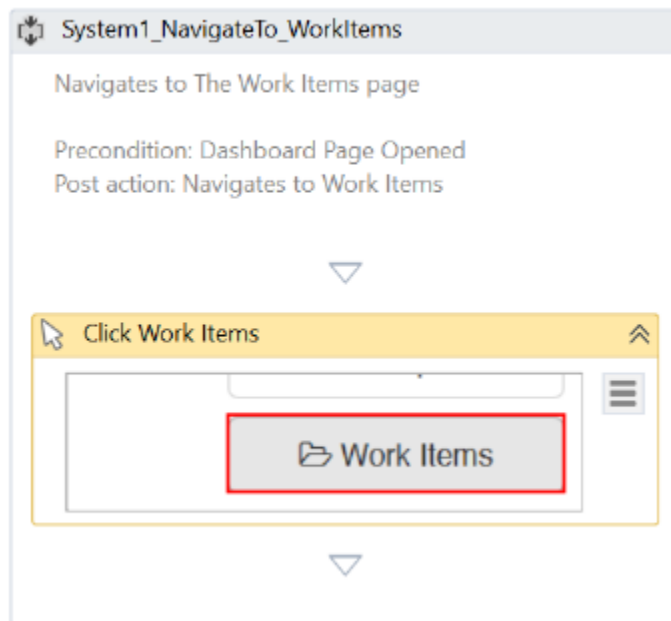
- Before starting to process transactions, we need to add the activities that are necessary to read the input data in the Init State. Usually, this is very simple - you can use either Read Range or Read CSV. In our process however, the input data is stored on a website, so more steps are required.
- Remember the purpose of this process, namely to retrieve the hash code for each item of the WI5 type. To do that, first we need a list of all the WI5 items.
- Create a blank sequence workflow in the System1 folder, to extract a Data Table variable that holds all the Work Items in the System1 application. We'll extract all the available work items and filter the WI5 type later.
  - Use the Data Scraping wizard to extract the entire HTML table. When asked if the data spans multiple pages, answer Yes, and point to the next page button.
  - Set the **Maximum number of results** option to 0, so that all the identified elements can be extracted as output.
  - Create an output argument and assign it the value of the the extracted data table.
  - If you added an annotation, you're done with this workflow! If you didn't, add the annotation now.
- Open the **Main** workflow and expand the **Init** state by double clicking it.
  - On the Entry region, locate where the KillAllProcesses workflow is invoked.
  - Add a new sequence after the **Invoke KillAllProcesses** activity to read the input transactions data table.
  - Inside this sequence, invoke four of the previously created workflows, as follows:
   - System1 Login

- System1 Navigate to Work Item
- System1 Extract Work Item Data Table
- System1 Close
  - o Import and bind arguments where necessary.
  - o From the list of work items, extract only the items needed in the current process - those of the WI5 type, with the status set to "Open".
    - Use the DataTable.Select method to filter out the elements that do not match the criteria above. Assign the result to a new variable.
   - This is what the **Assign** activity should look like:
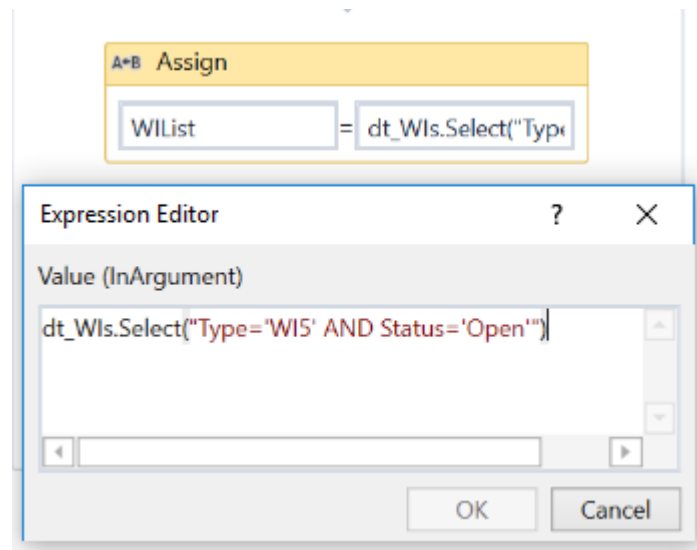
  - o We are going to process each item in the array at a time. This way, all DataRow objects become Transaction Items. The transaction index starts at 1, but the arrays are zero-based, hence the item index in the array is 1 less than the transaction number.
- Open the **Framework\GetTransactionData** workflow
  - o Update the annotation for the current process.
  - o Add an input argument for the array of Work Items.
  - o Remember that our Transaction Item is one object in the Work Item array. We can have a new transaction only if the transaction number is less than or equal to the index of the array element.
  - o Add an **If** activity to check if there is any new transaction to process.
    - In the **Then** section, use an **Assign** activity to set the value of the output Transaction Item argument according to its index.
    - In the **Else** section, set the Transaction Item value to Nothing.
  - o This is what the Get Transaction Data sequence looks like:
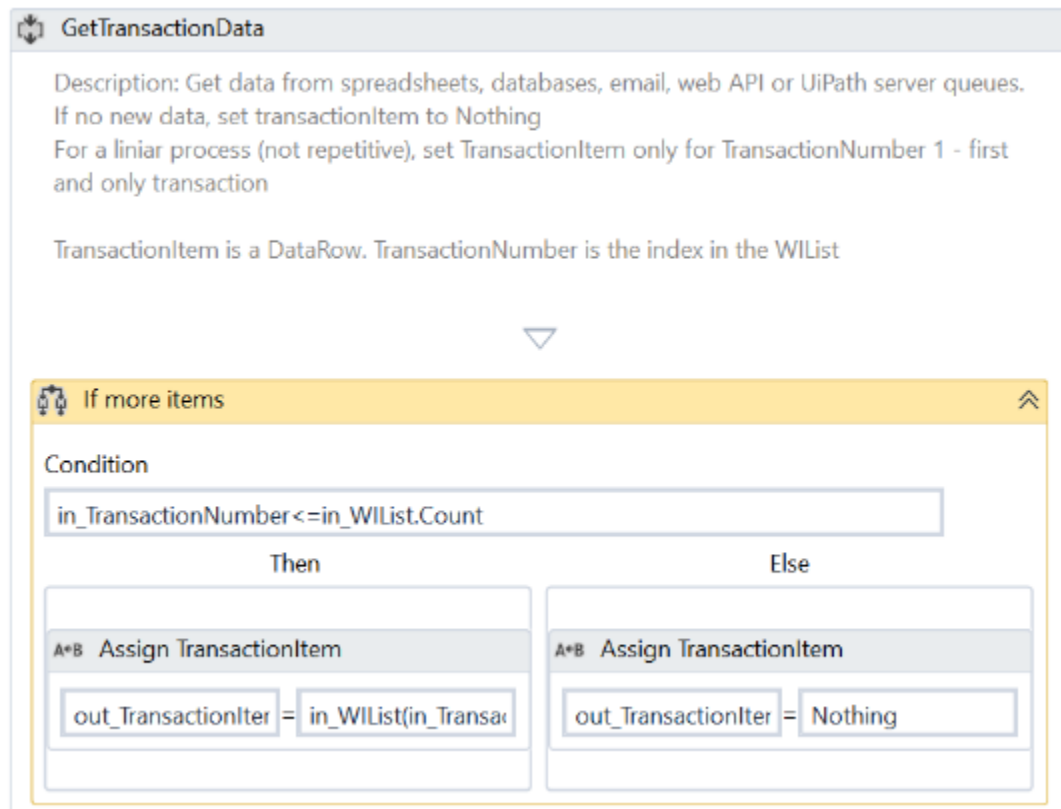
- o Also, when there are Transaction Items left to be processed, we need to set the Transaction ID to the value of the Work Item ID. The activities are already in place, so we only need to change the value of TransactionID in the **Assign** activity to "out_TransactionItem("WIID").ToString".
- At this point, the framework configuration is complete. We can test the **Main** workflow by running it, and then checking the extraction of input data. Use the Output log window to see if the data is correct.
- Now let us start to develop the workflows that process the Work Items.
- Create a blank sequence in the **System1** folder, to navigate to the Work Item Details page. Name it **System1_NavigateTo_WIDetails**. We can use the Work Item ID to navigate directly to the Item Details page. Open a work item and notice the format of the URL - it is composed of the System1 URL, the "/work-item/" string, and the Work Item ID.
  - o Work Item ID and System1 URL constitute the required input. Create two in arguments - one of the Int32 type, and the other, of the String type.
  - o Attach to System1 Dashboard and then use a **Navigate To** activity to go the Work Item Details page.
  - o This is what the sequence looks like:
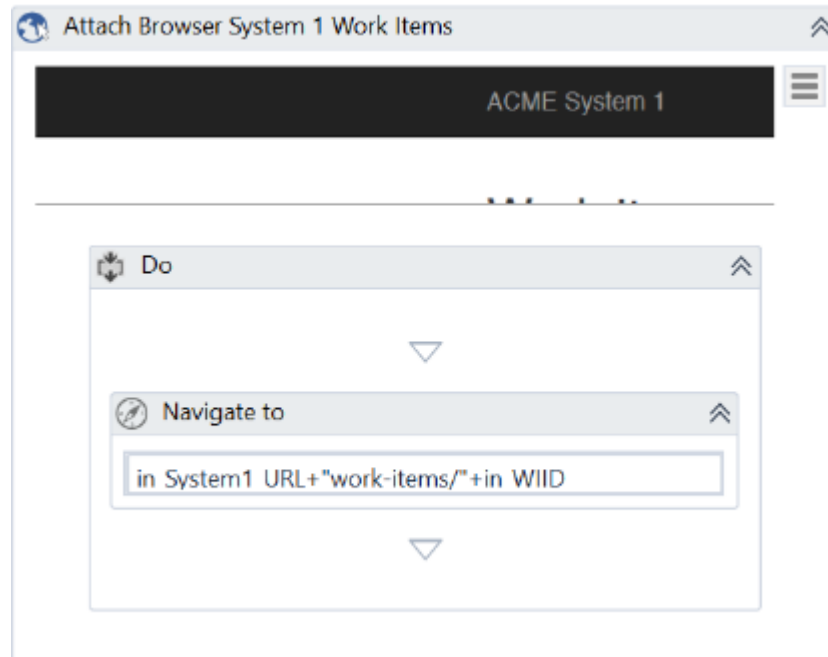
- Create a blank sequence workflow in the System1 folder called **System1_ExtractClientInformation**. We will use it to retrieve the details of an item.
  - There are no input arguments in this workflow. There is only a precondition - the Work Item Details page has to be already open.
  - We need 3 output arguments: Client ID, Client Name and Client Country.
  - Before getting the text from the web page, we have to make sure that the page is loaded, and all the elements are available. Add an **Attach Browser** activity. In the **Do** section, we'll use a **Get Text** activity with the **WaitForReady** property set to **Complete**.
  - This is what the workflow looks like:
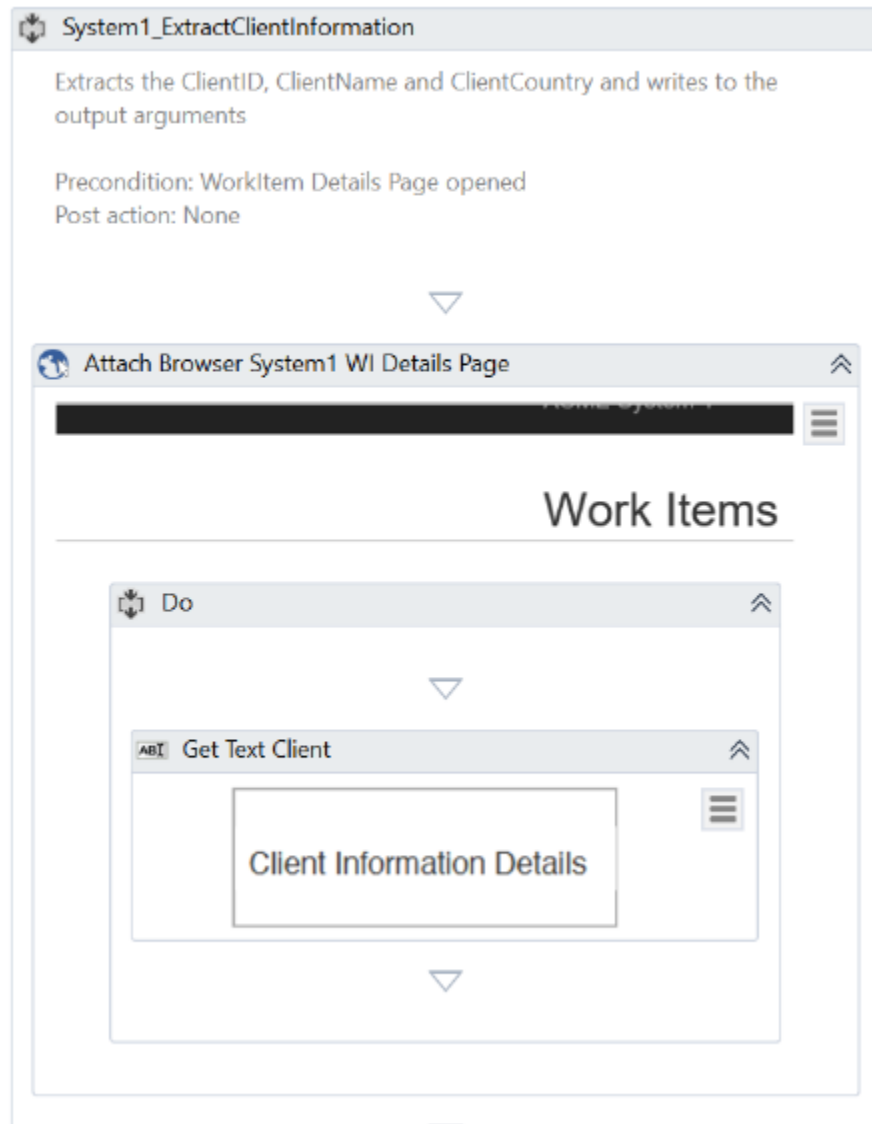
- In the same workflow, we need to extract the value of Client ID, Client Name and Client Country.
  - Add an **Assign** activity. Use the output argument for Client ID in the **To** field.
  - In the **Value** field, we are using the Substring and Split methods to extract the required information only. The resulting expression is ClientInformation.Substring(ClientInformation.IndexOf("Client ID: ")+"Client ID: ".Length).Split(Environment.NewLine.ToCharArray)(0)
  - Add two more **Assign** activities for Client Name and Client Country. The input in the **Value** field should be similar to the one we used for the Client ID argument.
  - After that, it's a good practice to write the extracted values to the Output panel, using the **Write Line** activity; that can help us to debug our workflow later on.
- Open the **Process** workflow

- o Edit the annotation.
- o Invoke the **System1_NavigateTo_WIDetails** workflow. Import and bind the arguments.
- o This is what it looks like:
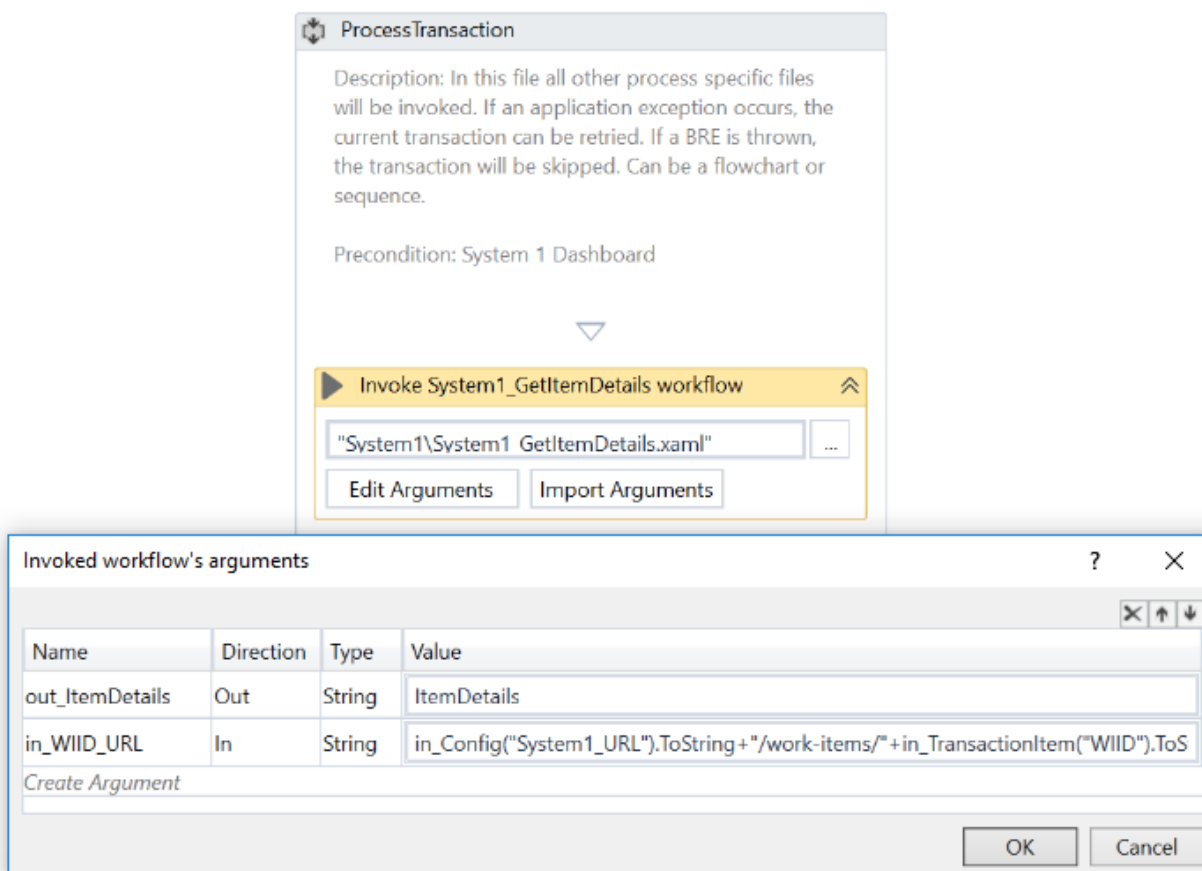
- o Invoke the **System1_ExtractClientInformation** workflow. Bind the three output arguments to the local variables.
- Now let's create a workflow to retrieve the hash value from the SHA1Online.com application. Create a blank sequence workflow the in SHA1Online folder. Name it SHA1Online_GetHashCode.
  - o We need an input argument of the String type. Its value is the one whose hash code we have to obtain.
  - o We also need an output argument of the String type for the computed hash code.
  - o Add a **Type Into** activity and configure it to type the input argument into the hash field in the System1 application. Optionally, in the **Properties** panel, select the checkbox in the **Simulate Type** field if you want this activity to work in the background accurately.

o Add a **Click** activity to select the **hash** button. Optionally, go to the **Properties** panel to select the checkbox in the **Simulate Click** field if you want this activity to work in the background accurately.

o Now we can get the result by using a Get Full Text activity; use the output argument to store the output text.

o Finally, we need to return to the initial page of the application, so that we can use the same sequence to process the next item. To do that, add a **Go Back** activity to navigate to the initial page.

- Remember that we must compute the hash code for ClientID-ClientName-ClientCountry, so we need compose that string. Let's use the output value of the **System1_ExtractClientInformation** workflow as input argument in the **SHA1Online_GetHashCode** workflow.

- Go back to the Process workflow.

o Add an **Invoke Workflow** activity and select SHA1Online_GetHashCode.

o Import the arguments. Use the hash formula as input argument. Create a new variable to store the hash result.

o This is what the workflow looks like:
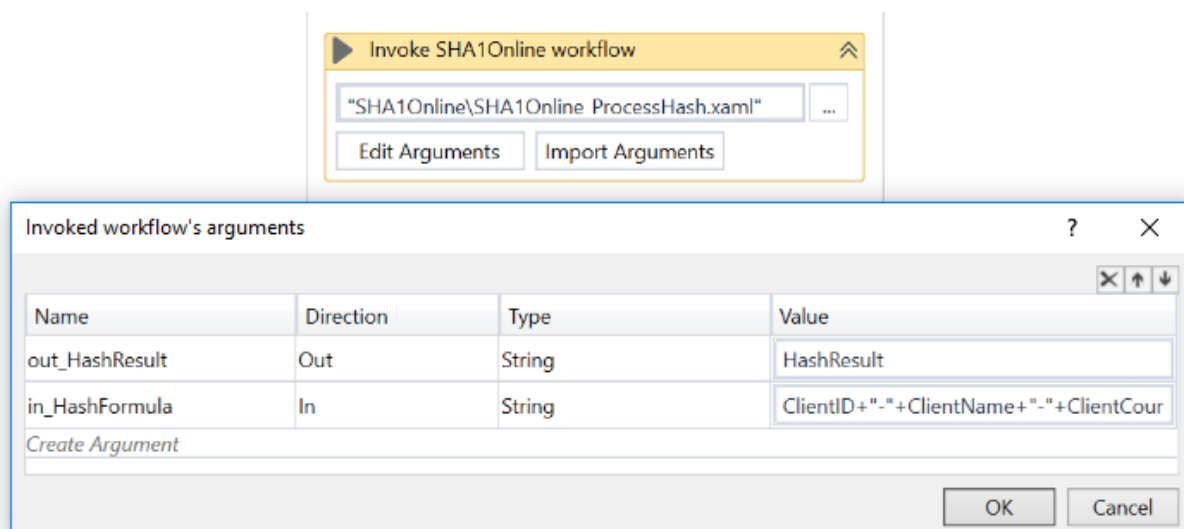
- Create a blank sequence workflow in the System1 folder, to update Work Items with the computed hash code. It's best to create a generic workflow that can be reused in future projects. To update the work items, we simply need to add a comment, set the new status, and then submit the changes.

o Start the new sequence by adding an annotation. The precondition is that the Work Item Details page is open, so we can update the Work Item.

o Add two input String arguments, one for the comment, and the other for the new status.

o   Add a **Click** activity and set the **Update Work Item** button as its target. In the **Properties** panel, select the checkbox in the Simulate Click field.

o   Use a **Type Into** activity to fill in the comment field on the **Update Work Item** page. Also, select the **Simulate Type** checkbox in the **Properties** panel.

o   Now we should update the status of this work item. To do that, drag and drop a **Select Item** activity. Click **Indicate on Screen** and then select the drop-down box in the **New Status** field. You're probably getting an error message stating that this control does not support select item. That is because we have other UI Elements on top of the Select input.

▪   Open **UiExplorer** and click Select Target Element. The returned element can be a Button or a Span UI Element, depending on where you clicked.

▪   Click the **New Status** field. Select element below the Button or Span, in Visual Tree panel and use the generated selector for the Select Item activity.

▪   This is how it looks in UiExplorer:
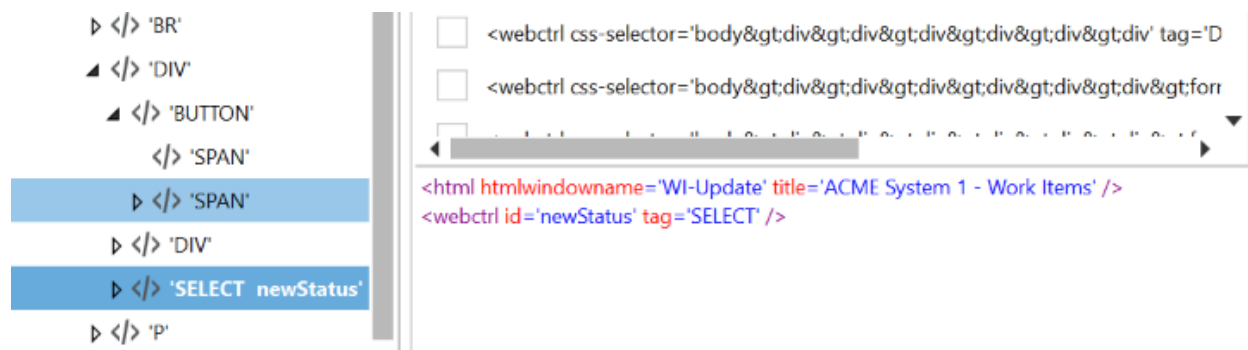
▪   Finally, set the Item property to the in_Status argument.

o   Add a **Click** activity for the Update Work Item button. Make sure the **Simulate Click** option is enabled.

o   A confirmation message pops up, so we need to use another **Click** activity to select the OK button. The **Simulate Click** option should be enabled here as well.

o   Now you can close the Update Work Item window by clicking on the close button in the upper-right corner.

•   Go back to the Process workflow and invoke the newly created workflow. Be sure to use the right variables as values of the input arguments.

•   Finally, we need to leave the application in its initial state, so that we can process the next item. To do that, invoke the workflow created to navigate to the Dashboard page.

- We are done with the process implementation. Next, we need to test the entire process. You should have already tested each individual workflow, right after development, using default values for the arguments.
  - o Run the Main workflow several times and see that every time it is executed correctly. If not, fix the issues and run again.
  - o Use the **Reset test data** option in the **User options** menu, to generate a fresh set of data for testing.
  - o Test the retry functionality. Set the MaxRetryNumber parameter in Config.xlsx to 1 and interfere with the robot by clicking on a different link menu, i.e. on Home link just after the robot opens current Work Item. This way, the Update Work Item button is not available and a System Exception is thrown because the UI element is not found. The process is re-initialized and the execution is resumed from failed work item. If you interfere again, the process will stop as the maximum number of retries is reached.

Process implementation notes

We started by retrieving the list of all the items that need to be processed, using the Data Scraping wizard. Ask yourself this: "What happens if you have a large set of transactions and the Extract Data activity in the Init state fails due to a browser timeout? How could we improve the design to increase error handling?"

- We processed one item at a time. All the items are independent of one another, so we could also process them in parallel, using multiple robots. In the next exercise, we will see how the Orchestrator Queues functionality can be used to implement a process in which the work items are distributed among the robots and processed only once, in parallel.

UiPath

Process Design Document
Calculate Client Security Hash

Process Design Document History

Date	Version	Role	Name	Organization	Function	Comments
28.09.2017	1.0	Draft	Olfa Ben Taarit	ACME Systems Inc.	SME	Creation v 1.0
28.09.2017	1.2	Reviewer	Vrabie Stefan	UiPath	BA	Approved v 1.0
09.01.2019	1.3	Reviewer	Silviu Predan	UiPath	RPA Dev	Updated

Table of Contents

1. Introduction

1.1 Purpose of the document

The Process Design Document describes the business processes chosen for automation using the UiPath Robotic Process Automation (RPA) technology.

This document describes the sequence of steps performed as part of the process, as well as the conditions and requirements prior to its automation. This design document serves as a base documentation for developers to collect the details required for robotic automation of the same business process.

1.2 Objectives

The process has been selected for RPA as part of the larger project initiative conducted within ACME Systems Inc., the Finance and Accounting department.

The objective of this process automation is linked to the project business case and is mainly intended to:

- ➢ Deliver faster processing
- ➢ Reduce redundant activities
- ➢ Improve overall performance and reliability

1.3 Process key contacts

The Design Document includes a brief, but comprehensive set of requirements for the process. Its structure is based on the input provided by the Subject Matter Expert (SME) in the process.

Role	Name	Date of action	Notes
Process SME	Aurel Vlaicu	TBD	Point of contact for questions related to business exceptions and passwords
Reviewer / Owner	Sergiu Celibidache	TBD	POC for process exceptions
Approval for production	Nicoale Herlea	TBD	Escalations, Delays

2. AS IS Process Description

2.1 Process overview

General information about the process selected for RPA implementation, prior to its automation:

AS IS process details	
Process full name	Calculate Client Security Hash
Function	Security
Department	Finance and Accounting
Process short description (operation, activity, outcome)	Generate the Security Hash for each Client based on their personal information.
Role required for performing the process	System 1 User
Process schedule	Daily
# of item processes / day	7 – 15 Clients
Average handling time per item	2 min / Client
Peak period (s)	No peak period
# of FTEs supporting this activity	1
Level of exception rate	No expected exceptions
Input data	Client Data
Output data	Client Security Hash

2.1.1 In scope for RPA

The activities and exceptions in this process that are in the scope for RPA, are listed below:
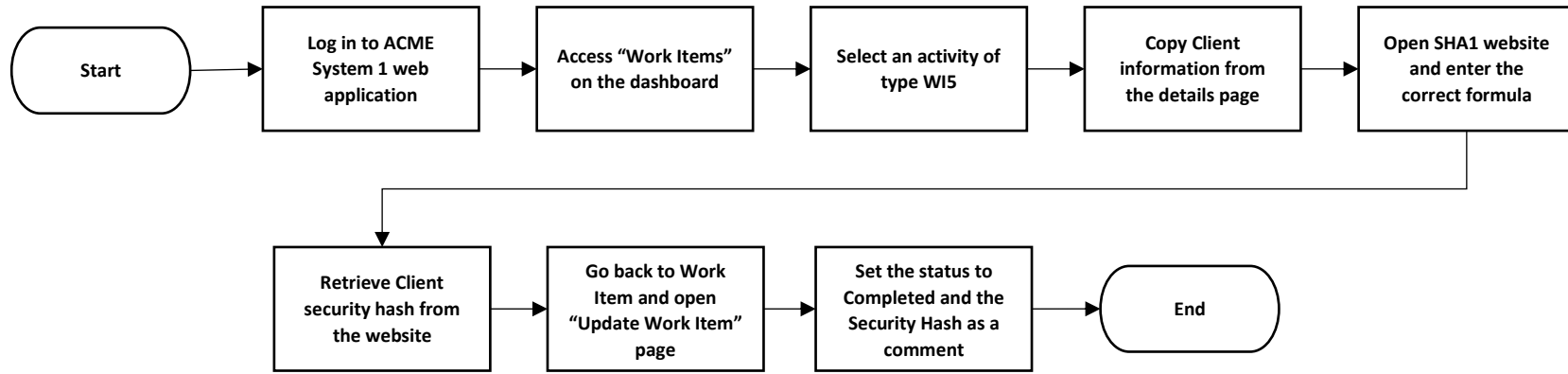
> ➢ Full Scope for RPA - the process is to be 100% automated.

2.1.2 Out of scope for RPA

There are no activities out of scope for RPA

2.2 Detailed Process map

This chapter presents the chosen process in detail, which enables the developer to build the automated process.
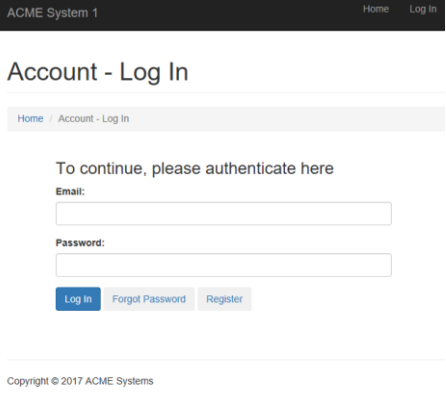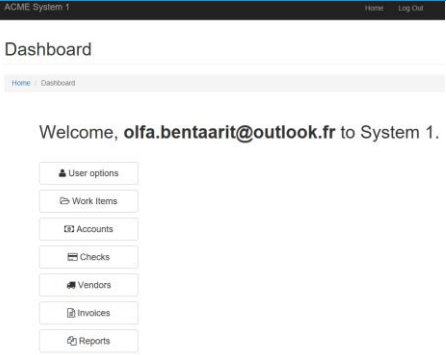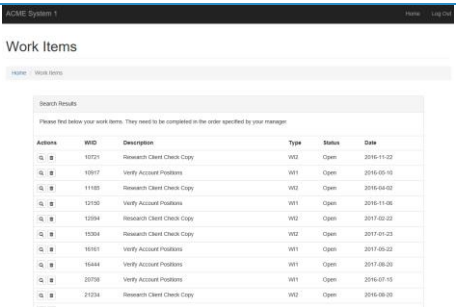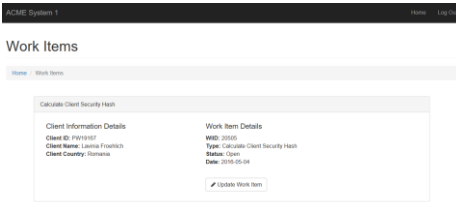
Step	Short Description
1.1	Open the ACME System 1 Web Application.
1.2	Log in to System 1. Required input data: email and password.
1.3	Access the Dashboard - the central location, where the user can pick a specific menu item.
1.4	Access the Work Items listing to view all the available tasks to be performed (Output data: Work Items).
1.5	**For each activity** of the WI5 type, perform the following steps:
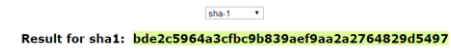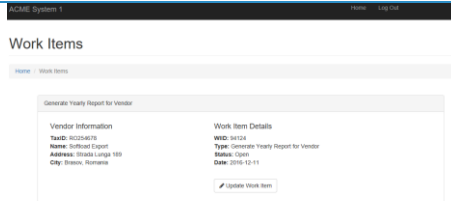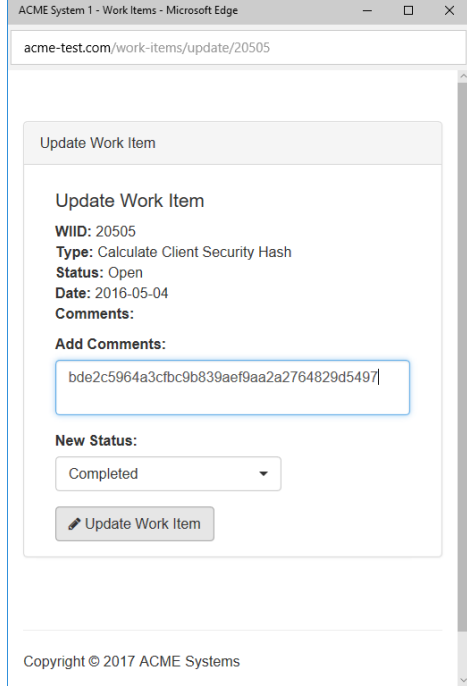
1.5.A	Open the Details page of the selected activity to retrieve the Client Information Details.
1.5.B	Open the SHA1 Online webpage - http://www.sha1-online.com/, and provide the following input: **[ClientID]-[ClientName]-[ClientCountry]**. Replace all the variables with the corresponding values. Use dashes between items, as shown above.
1.5.C	Retrieve the Client **Security Hash** value from the webpage.
1.5.D	Go back to Work Item Details and open Update Work Item.
1.5.E	Set the status to "Completed". Add a comment with the obtained **[SecurityHash]**.
1.6	Continue with the next WI5 Activity.

2.3 Detailed Process Steps

The complete set of steps in the process, including keystrokes and clicks, are to be defined with screenshots. If there are any data restrictions, mask the sensitive information, such as Policy Number, Customer ID, bank account number, etc).

#	Step action description	Screenshot	Expected result	Remarks
1.1	Open the ACME System 1 Web Application		The display of the System 1 Web App screen.	Possible exception: - Handle exception if Web app not available
1.2	Log in to System 1. Required input data: email and password.		Access to the dashboard	Possible exception: - Handle exception if Incorrect email or Password
1.3	Access the Dashboard - the central location, where the user can pick a specific menu item		The display of each item in the menu	

1.4	Access the Work Items listing to view all the available tasks to be performed		The display of the task list	
1.5	**For each activity** of the type WI5 perform the following steps:			Possible exception: Handle exception if no task of type 'Calculate Client Security Hash' exist
1.5.A	Open the Details page of the selected activity to retrieve the Client Details (Output data: Client Details)			
1.5.B	Open the SHA1 Online Webpage - http://www.sha1-online.com/ and provide the following input: **[ClientID]**-**[ClientName]**-**[ClientCountry]** Replace all the variables with the corresponding values. Use dashes between items, as shown above.			

1.5.C	Retrieve **Client Security Hash** from the webpage			
1.5.D	Go back to the Work Item Details and select Update Work Item			
1.5.E	Set the status to "Completed". Add a Comment with the obtained **[SecurityHash]**			
1.6	Continue with the next WI5 Activity			

2.4 Exceptions handling

The types of exceptions identifiable in the automation process can be classified according to the table below.

Area	Known	Unknown
Business	Previously encountered situation. A possible scenario is defined, and clear actions and workarounds are provided for each case.	A situation never encountered before. It can be caused by external factors.

Based on the above criteria, the table below should reflect all the known exceptions identified throughout the process and map the expected action the robot needs to take in each case.

Insert as many rows as required in the table, to capture all exceptions in a comprehensive list.

#	Exception name	Step where exception is encountered	Parameters	Action to be taken
1	Incorrect email or password	Step # **1.2**	If message for incorrect username or password exist	Send email to exceptions@acme-test.com "Hello, The username or the email is incorrect. Please check and restart Thank you"
2	No task of type WI5 exists	Step # **1.5**		Stop process

For any other unanticipated or unknown exceptions, the robot should send an email notification at exceptions@acme-test.com with the original email and error message screenshot attached.

2.5 Error mapping and handling

A comprehensive list of all the errors, warnings, or notifications should be consolidated here with the description and action to be taken by the Robot in each case.

The errors identified in the automation process can be classified according to the table below.

Area	Known	Unknown
Technology	Previously encountered situation - action plan or workaround available.	A situation never encountered before, or may happened independent of the applications used in the process.

Based on the above criteria, the table below should reflect all the identifiable errors in the process, and map the expected action of the Robot in each case.

Insert as many rows as required in the table, to capture all the errors in a comprehensive list.

#	Error Name	Step where error is encountered	Parameters	Action to be taken
1	Application unresponsive / page not loading	Any step	No response / blank page	Retry 2 times. Close application and run the sequence again

*Feel free to insert an additional error mapping table for more complete explanation.

2.6 In-Scope application details

The table below lists all the applications that are used as part of the automated process.

#	Application name & Version	Syst. Lang.	Login module	Interface	Environment/ Access method	Comments
1	ACME System 1	EN	Web	Web	Web Browser	
2	SHA1 Online	EN	n/a	Web	Web Browser	

3. Development details

3.1 Prerequisites for development

- Development or testing environment are to be provided for development purposes.
- The provided development and testing environments are exact replicas of the production environment.
- Dedicated system and application access are given to developers with the adequate permissions.

3.2 Password policies

Users manage their own passwords. There are no special policies in place.

3.3 Credentials and asset management

Login details (user IDs and passwords) should be stored under **Windows Credential Manager** or **UiPath Orchestrator Assets**.

4. Document Approval Flow

Version	Flow	Role	Name	Organization (Dept.)	Signature and Date:
1.0	Document prepared by	Business Analyst	Name Surname		
1.0	Document Approved by:	Business Process Owner	Name Surname		
1.0	Document Approved by:	Dev/RPA Solution Architect	Name Surname		

5. Appendix

5.1 UiPATH automated process details

Note: this step is to be filled in after automation process is complete

Automation overview: (time to dev, test, etc)

Robots type: Unattended

Level of human intervention required:

Use of Orchestrator:

Exceptions recorded in automation process:

Errors identified in the automation process:

Challenges identified in the automation process:

Lessons Learned:

Any adjustments made to facilitate the automation process and any steps taken to shift from the human way of working to the automatic one. Any activity performed to improve the As Is process and to enable higher rates of automation of the process:

- ➢ Process Assumption
- ➢ Input data assumption
- ➢ Number or types of input to be received
- ➢ Skipping the login interface and collecting backend details
- ➢ Extracting backend data without opening the file
- ➢ Data conversion / formatting

Reporting: The details and format of the logging mechanism available in the workflow have to be specified here, whether it is a local log report or the Orchestrator log).

The format should be specified by the business users.

Workflow and scripts: A brief overview of each workflow and the sequence in which it is executed should be provided here.