

**Robotic Process Automation**

Recruitment\_Task

*Process Definition Document (PDD)*

Table of Contents

[1. Document Information 4](#_Toc80381944)

[2. Purpose 4](#_Toc80381945)

[3. Definitions 4](#_Toc80381946)

[4. Master Project Details 5](#_Toc80381947)

[4.1 List of packages 5](#_Toc80381948)

[4.2 Runtime Details 6](#_Toc80381949)

[5. User Requirements, Functional or Use Case Specification Document Reference 7](#_Toc80381950)

[5.1 Process Diagram 7](#_Toc80381951)

[5.2 Requirements implementation 7](#_Toc80381952)

[5.3 Configuration files 8](#_Toc80381953)

[6. Solution Design 9](#_Toc80381954)

[6.1 Project folders structure 9](#_Toc80381955)

[6.2 Input Data 10](#_Toc80381956)

[6.3 Output Data 10](#_Toc80381957)

[7. Project Workflows 11](#_Toc80381958)

[7.1 Main.xaml 12](#_Toc80381959)

[7.2 ExtractNames.xaml 13](#_Toc80381960)

[7.3 Process.xaml 14](#_Toc80381961)

[7.4 SortAndWriteDataTableToExcel.xaml 15](#_Toc80381962)

|  |  |
| --- | --- |
| **System Name** | Collecting Most Popular Names in Poland 2018 |
| **Document Name** | Process Definition Document (PDD) |

## Document Information

| **Role** | **Name** | **Date** |
| --- | --- | --- |
| Developer | Konrad Bywald | 18.08.2021 |

## Purpose

This document describes the implementation of Recruitment\_Task. Its purpose is to specify and explain the way in which automation has been created. In further chapters are described sequences and workflows used in the project. The Process Definition Document (PDD) is created for every business process that is automated using the RPA technology and document is filled by the RPA Solution Architect and RPA Developer who automates the business process.

## Definitions

The following additional terms and abbreviations are used in this document:

| **Term** | **Definition** |
| --- | --- |
| Master project | The overall output of the development, containing one or multiple projects that together cover the scope of the robotic process automation. |
| Project | An UiPath Studio project containing one or multiple workflow files. A project can be converted to a package and run independently, covering a particular scope within the master project. |
| Package | The output of compiling a project. A package can be deployed on the robot machine and be executed by the robot service. Only one package can be executed at a given time by a robot. The package is used when defining the running phase of the automation. |
| Workflow | A component of the package, the workflow encapsulates a part of the project logic. The workflow can be of type: sequence, flowchart or state machine. a workflow is saved as an .xaml file inside the project folder. A workflow file can be invoked from another workflow and by default there is an initial workflow file that will run when executing the package. |
| Activity | An action that the robot executes. |
| Sequence | A workflow where activities are executed one after another, in a sequential order. |
| Flowchart | A workflow where activities are connected by arrows and the logic of the workflow can be easily followed in a visual manner. |
| State machine | A more advanced way of organizing a workflow, with higher capability of flow control. Similar to a flowchart. |
| State | A part of state machine representing state of application. |
| UiPath | Software provider for the automation. |
| ReFramework | Framework for building enterprise class automations in UiPath. |
| Robot | An execution host that runs processes built in UiPath Studio. |
| System exception | Exception caused by system failure. |
| Business rule exception | Exception caused by rules defined by the Business. |

## Master Project Details

Details filled in by the developer reflect the actual information for the master project.

| **Item** | **Details** |
| --- | --- |
| Master project name and version | Recruitment\_Task |
| Robot type | Unattended |
| Is Orchestrator used? (Yes/No) | No |
| Scalable? (Yes/No)  (can the process be run by multiple robots in parallel) | No |

### List of packages

List of packages used in the automation.

| **Package Name** | **Version** | **Details** |
| --- | --- | --- |
| UiPath.Excel.Activities | 2.11.0 | The Excel activities package allow users to automate all aspects of Microsoft Excel |
| UiPath.System.Activities | 21.6.1 | The System Activities Pack contains all the basic activities used for creating automation projects |
| UiPath.UIAutomation.Activities | 21.6.1 | The UIAutomation activities package contains all the basic activities used for creating automation projects. |

### Obraz zawierający tekst Opis wygenerowany automatycznieRuntime Details

|  |  |
| --- | --- |
|  |  |
| **Prerequisites to run** | * UiPath Robot installed * UiPath Studio installed * Before first start it is necessary to manually open Internet Explorer and check if popup window presented above appears. If yes it needs to be clicked with option recommended settings. If popup did not appear robot is ready to work. * ~~UiPath Chrome Extension has to be enabled in Google Chrome with allow incognito option enabled~~ * Access to the Internet |
| **Input Data** | Names of children extracted from the given websites. |
| **Expected Output (output data)** | Exported table to the excel file with top 10 names of boys and girls ordered descending by number of children with given name. Obraz zawierający stół  Opis wygenerowany automatycznie |
| **How to start the automated process?** | * Manually from the Robot’s tray icon * Manually from the UiPath Studio (recommended for this project) |
| **Reporting**  (Kibana or another platform) | N/A (Robot does not send any reports after processing) |
| **How to resume the process in case of error** | Automation contains a mechanism which repeat process if exception has been thrown. |
| **Stored Credentials:**  (Never hardcode credentials) | N/A (This process does not use any credentials from Orchestrator or credential manager) |
| **List of Asset Names:** | N/A (This process does not use any assets from Orchestrator) |
| **Recommended resolution** | 1920x1080 |

Before first execution of the process using Internet Explorer as a browser there is a possibility that popup window will appear. In this case it is necessary to manually open IE and click that popup next start the automation.

Details of the automated process.

## User Requirements, Functional or Use Case Specification Document Reference

|  |  |
| --- | --- |
| **User Requirements Specification Document Name:** | Zadanie rekrutacyjne.pdf |

Obraz zawierający tekst

Opis wygenerowany automatycznie

### Process Diagram

Diagram which represents flow of the process based on the requirements.



### Requirements implementation

The requirements are implemented in the following way:

Step 1 – Download from websites list with names of children

Robot navigates to the given websites using URLs which are stored in the Config.xlsx file in settings called govWebsites\_URL. Both URLs are assigned to one setting and they are separated with # character, later in the code Robot use split method to separate URLs and use them in the Open Browser activity in order to navigate to the website. Once website is opened Robot use Get Text activity which purpose is to extract whole list of names from given website, as a result Robot receive text value which is later processed during automation.

Step 2 – Select top 10 names on each website

Robot uses regex regular expression in order to find match in the extracted text. Next he takes specify number of names using Take() method. This number can be set in the Config.xlsx file by changing value for setting called numberOfNamesToTake.

Step 3 – Generate random month for each name

Robot generates random month by using list of months to which he refers with random index generated by Random() method. Extracted value from list is assigned to the variable and added to the data table.

Step 4 – Generate Unicorn Name based on names and assigned months

In order to generate Unicorn Name robot take two variables – name and generated month. Next he navigates to the given website and enter data. After generating result value is extracted using Get Text activity and assigned directly to the appropriate variable.

Step 5 – Export table to the excel file. Table should be sorted descending by number of children. Names should start with capital letter but rest should be lowercase.

In the End Process state if no error occurred during processing robot invokes workflow which is responsible for write prepared sorted data table to the excel. First data table is sorted descending by column “Number of children”.

### Configuration files

Following files are used as configuration sources for the project:

1. Project.json - this file contains the project metadata.

| **Variable name:** | **Description:** |
| --- | --- |
| name | The title of the automation project. |
| description | The description of the project. |
| main | The entry point of the automation project. It consists of an .xaml file. The default name is “Main.xaml”. |
| dependencies | The activity packages used to create the automation project and their versions. |
| studioVersion | The version of Studio used to create the automation project. |
| projectVersion | The version of the project |

1. Config.xlsx

Main source of configuration data for the project. The purpose of each configuration variable is described inside the file.

Spreadsheet is divided into 3 separate tabs:

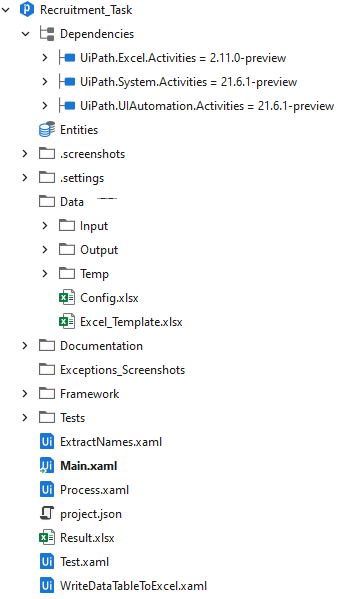
* Settings: contains configuration variables for the project.
* Constants: constants used around the project.
* Assets: contains mappings between configuration key names and assets names on the Orchestrator. Assets are downloaded from the Orchestrator and added to the Dictionary<string, object> Config object. Assets will overdrive the configuration variables with the same names.

## Solution Design

The Recruitment\_Task is an automation which main purpose is to automate short process of downloading most popular names from the given websites. Moreover during execution robot also generates random month and unicorn name for each children name. All of those values are added to the data table and moved to the excel file as a result of execution. Project has been created based on the UiPath ReFramework with one additional state added after Initialization. This state is responsible for extracting data for processing and passing them to the Transaction Data collection.

### Project folders structure

The following folder structure is used in the project:



### Input Data

Input for automation is extracted from websites specified in the requirements. Data are presented on the pages in very long list which robot needs to extract.

Obraz zawierający tekst

Opis wygenerowany automatycznie Obraz zawierający tekst

Opis wygenerowany automatycznie

### Output Data

Automation as the output produce an excel file which contains selected top 10 names for girls and boys. Table is sorted descending by column “Number of children”. Excel file which is produced as the output of the automation is always overwritten after each execution.

Obraz zawierający stół

Opis wygenerowany automatycznie

## Project Workflows

Define below all the workflow files (xaml files) used in the project, with the Input and Output data.

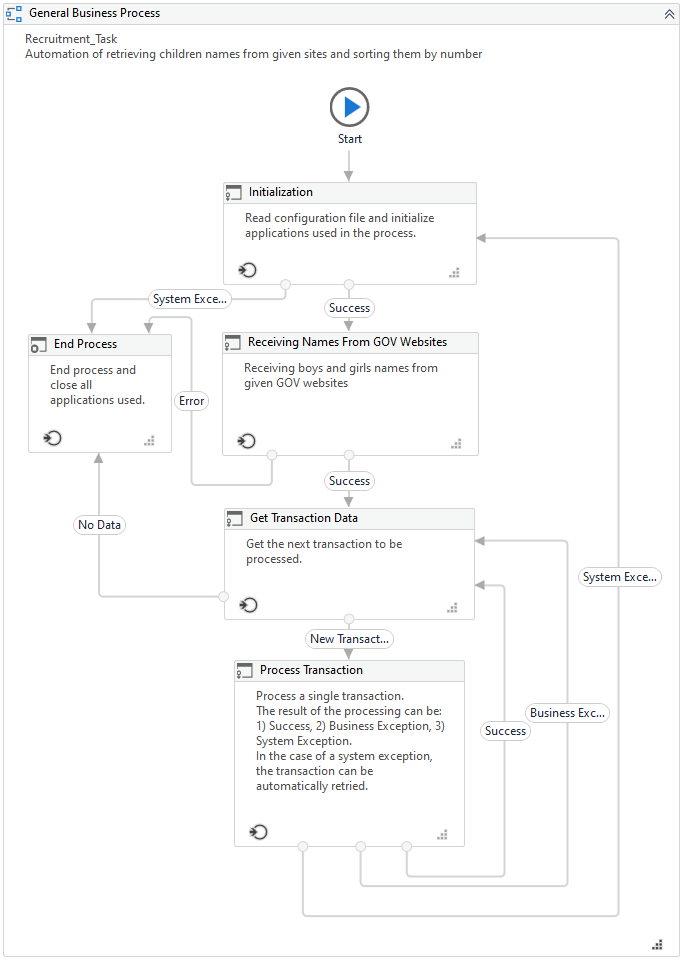
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Workflow file name** | **Description** | **Input Argument** | **Output Argument** |
| 1 | Main.xaml | Main file contains state machine which holds all states and is responsible for performing process. By default Main.xaml file is set as main in project.json | N/A | N/A |
| 2 | ExtractNames.xaml | This workflow is responsible for extracting names from given websites. In the for each loop robot navigates to the pages and on each of them he reads names using Get Text activity next result is checked with the regex expression and specified number of names is added to the TransactionData List<String> collection. | * io\_TransactionData (List<String>) * in\_WebsiteURLS   String[]   * in\_Config   Dictionary<String, Object> | * io\_TransactionData (List<String>) |
| 3 | SortAndWriteDataTableToExcel.xaml | This workflow is used in the End State and it is responsible for sorting DataTable descending by number of children. Next sorted DataTable is moved to the excel file using Excel\_Template.xlsx file which is stored in the Data folder | * in\_ExcelDataTable   Datatable   * in\_Config   Dictionary<String, Object> | N/A |
| 4 | Process.xaml | Invoke major steps of the business process, which are usually implemented by multiple subworkflows. | * in\_TransactionItem   String   * in\_Config   Dictionary<String, Object>   * in\_UnicornNameBrowser   Browser   * io\_excelDataTable   DataTable | * io\_excelDataTable   DataTable |
| 5 | GetTransactionData.xaml | Part of REFramework. Gets data from TransactionData collection. If no new data, set out\_TransactionItem to nothing. | * in\_TransactionNumber   Int32   * in\_Config   Dictionary<String, Object>   * out\_TransactionItem   String   * io\_TransactionData   List<String> | * io\_TransactionData   List<String> |
| 6 | InitAllSettings.xaml | Part of REFramework. Outputs a settings Dictionary with key/value pairs to be used in the project. Settings are read from local config file then fetched from Orchestrator assets. Assets will overwrite the configuration file settings | * in\_ConfigFile   String   * in\_ConfigSheets   String[] | * out\_Config   Dictionary<String, Object> |
| 7 | KillAllProcesses.xaml | Part of REFramework. Kills the working processes used in the automation. Processes killed by the workflow: saplogon.exe, chrome.exe | N/A | N/A |
| 8 | SetTransactionStatus.xaml | Part of REFramework. This workflow sets the TransactionStatus and Logs that status and details in extra Logging Fields.  The flowchart branches out into the three possible Transaction Statuses: Success, Business Exception and Application Exception. | * in\_Config   Dictionary<String, Object>   * in\_TransactionItem   String   * in\_TransactionData   List<String>   * in\_BusinessException   BusinessRuleException   * in\_Exception   Exception | * io\_TransactionNumber   Int32   * io\_RetryNumber   Int32 |
| 9 | TakeScreenshot.xaml | Part of REFramework. This workflow captures a screenshot and logs its name and location. | * in\_Folder   String | N/A |
| 10 | CloseAllApplications.xaml | Part of REFramework. Closes all working applications. | N/A | N/A |

### Main.xaml

Main.xaml file holds all states which belongs to the automation. In this case one additional state has been added which is responsible for collecting data for processing. At the beginning in the Initialization robot is reading Config file and preparing data table which will be filled later with all data and move to the result excel file. Below the structure of the data table is presented and Main.xaml file as well. Result file is saved in the project folder. We can specify his name by change value in config file for setting called excel\_ResultFileName.

Obraz zawierający tekst

Opis wygenerowany automatycznie

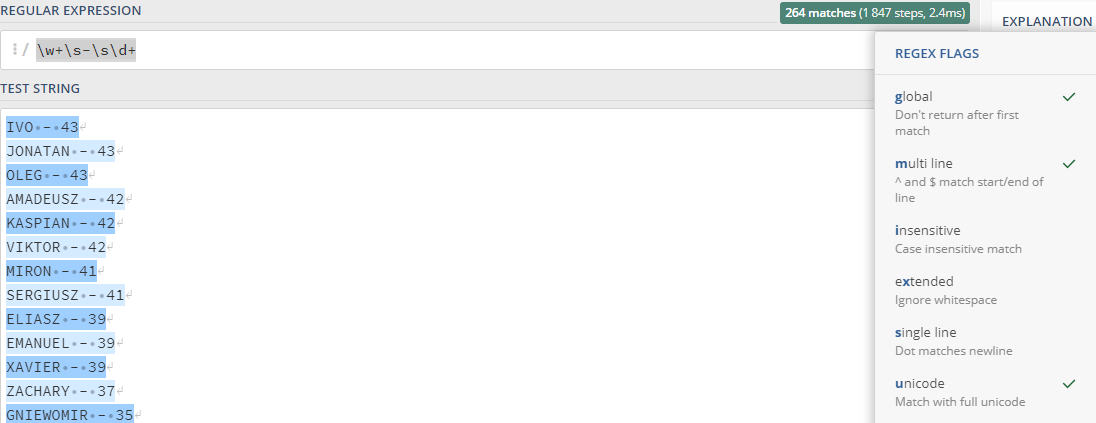


### ExtractNames.xaml

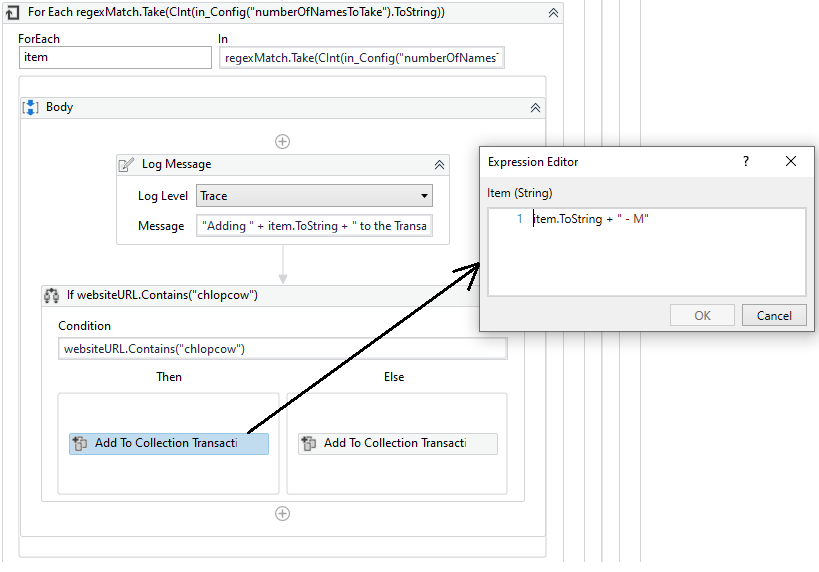
This workflow is invoked in state called Receiving Names From FOV Websites. It is responsible for navigate to the given websites using value passed via argument in\_WebsiteURLS this value is splitted by # and passed to the For Each loop. Robot uses Google Chrome in incognito mode in order to open website and it cause that always after opening the page at the bottom of the page cookie popup appears, but it does not have any impact on the automation and it is not necessary to focus on it. At the beginning browse window is maximized and next Robot uses Get Text activity to extract whole list with the names.

Extracted value is check if regex found matches using pattern: \w+\s-\s\d+

Below example screenshot from regex101 website with flags sets as: global, multi-line and unicode.



Once matches were found Robot takes specified number of names by using value stored in the Config and adds them to the TransactionData collection in meantime checking if given name was received from boys page or no and based on that information Robot makes decision which gender should be assigned for this name.



### Process.xaml

In this workflow robot performing actions according to the requirement document. Here the random month and Unicorn Name are generated.

In order to generate random month robot is using list with months with the given structure:

New List(Of String) From {"January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"}

Random month is selected by assigning value from the list to the variable called randomMonth using Random().Next() method which is invoked as index of the list.

randomMonth = monthsList(cInt(new Random().Next(0,11)))

Unicorn Name is generated on the page specified in the requirements. Robot uses name and generated month next enters those values to the form in order to receive Unicorn Name which is extracted from the website using Get Text activity and result is assigned to the variable called extractedUnicornName.

Obraz zawierający tekst

Opis wygenerowany automatycznie

At the end of Process.xaml all received and generated data are added to the Data Table which is set as in/out argument and in the End Process state will be sorted and saved to the excel file.

Obraz zawierający tekst

Opis wygenerowany automatycznie

### SortAndWriteDataTableToExcel.xaml

This workflow is invoked in the End Process state. The purpose of this workflow is to sort data table which is passed as input argument. Sorting is done descending by column “Number of children”. After that sorted data table is saved in the result excel file which name can be specified in the Config file by change value for setting excel\_ResultFileName value. Name of the sheet used in the file can be specified in the Config as well. Excel file as a template is saved in the Data folder and it is always copied to the main project folder with option overwrite set to true.

Obraz zawierający tekst, zrzut ekranu, wewnątrz

Opis wygenerowany automatycznie