Library Documentation

Created by Stefan Oberholzer

Table of Contents

[Email Library 2](#_Toc126849365)

[Send Email 2](#_Toc126849366)

[Reading Emails 2](#_Toc126849367)

[Excel Library 3](#_Toc126849368)

[Auto Fit Columns 3](#_Toc126849369)

[Convert Excel File Type 3](#_Toc126849370)

[Index To Column Letter 3](#_Toc126849371)

[Index To Column Letter Activities 4](#_Toc126849372)

[Sum Table 4](#_Toc126849373)

[Miscellaneous Library 5](#_Toc126849374)

[Directories 5](#_Toc126849375)

[Local Report File 5](#_Toc126849376)

[Randomly Generate Password 6](#_Toc126849377)

# Email Library

A library was built that handles the credential retrieval for the email and do all the email sending and reading parts. The library uses Microsoft Outlook and Exchange emails.

## Send Email

The sending of emails is performed by using a queue and the RE-Framework. The JSON format of the queue item’s specific content is seen below.

|  |
| --- |
| Specific Data: Object {  **RecipientEmail**: *stefanus.oberholzer@gmail.com*  **Subject**: *Email Subject*  **Body**: *Storage Bucket Test Email*  **Sender**: *On behalf of Stefan*  **AttachmentLocation**: *StorageBucketFile.txt*  **RecipientName**: *Stefan*  **CC**: *Empty*  **BCC**: *Empty*  } |

Table 2: Example JSON format for send email queue item.

The configuration file of the RE-Framework must be configured with the orchestrator folder and queue name, as well as the storage bucket’s name which is stored as an asset.

No additional arguments are required for the library to function. The email template must be configured using text, html, or a word document. The variables extracted from the queue must be integrated into the body template. The template is currently structed in a html format.

The attachment is processed in the following order. Given a file or folder must be attached the library firstly tries to locate the file/folder locally. Only when the file/folder could not be found locally the library tries to locate the file in the storage bucket. If none of the locating methods were successful, an exception is thrown. The local file will always take precedence over the storage bucket file if both files have the same name.

## Reading Emails

The all the unread emails are retrieved and stored locally. The attachments and emails are stored in the different specified location. A summary of the email can also be created and stored in a text file, since it is difficult to read the email saved in the msg format. The email summary can be configured to store all the relevant information from the email for future processing.

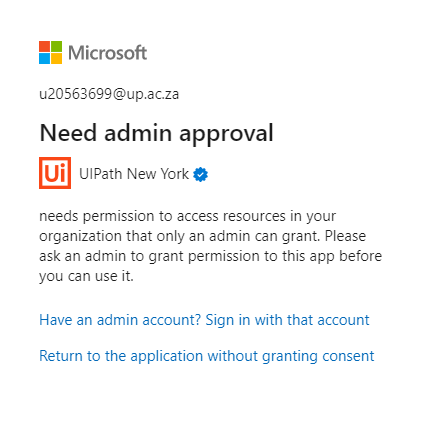
|  |  |  |  |
| --- | --- | --- | --- |
|  | **Name** | **Data Type** | **Description** |
| **In Arguments** | In\_AttachmentFilter | String | Save only files that match a specific pattern. For example, \*.xls\* will store most excel files. |
| In\_FolderPathMail | String | The folder path where the email or summary file must be stored. |
| In\_ FolderPathAttachment | String | The folder path where the attachment files must be stored. |

Inline attachments will be excluded and the attachment files will be overwritten if they already exist.

*The reading email activity was not tested extensively since my outlook account does not receive emails. No email can be processed since I am unable to get emails inside the account.*

# SharePoint Library

The SharePoint Library was not implemented. When using my student account as I did for the rest of the library development, I do not have the required permissions to perform SharePoint activities.



# App Logger

I attempted to implement the app logger with a background process instead of a parallel activity. The background process can only identify the start of a process and not individual workflows. This make tracing the exception much more difficult.

My app logger is still a working progress. I will most likely switch from a background process to a parallel activity.

# Excel Library

The Excel Library contains useful function which is often repeated in a project.

## Auto Fit Columns

Auto fit the columns of all the sheets or only the specified sheets of an Excel file.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Name** | **Data Type** | **Description** |
| **In Arguments** | In\_ExcelFile | String | The operation will be performed on the specified Excel file. The absolute path of the Excel file is needed with the file extension. The string must be quoted. |
| In\_AllSheets | Boolean | If checked, all the columns of all the sheets will be set to auto fit the content. The next argument is ignored if this value is set to true. |
| In\_SheetName | String[] | Only set the columns of the specified sheet(s) to autofit. The argument is be a list of string (in other words string[]). The in\_AllSheets must be set to false for this argument to be used. |

## Convert Excel File Type

Convert the file type of an Excel file to xlsx, xls, xlsm or xlsb.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Name** | **Data Type** | **Description** |
| **In Arguments** | In\_ExcelFile | String | The operation will be performed on the specified Excel file. The absolute path of the Excel file is needed with the file extension. The string must be quoted. |
| In\_Extension | String | The extension of the new Excel file. The in\_ExcelFile is converted to the file type specified in this string argument. This string must be quoted. |

## Index To Column Letter

Convert table indexes to Excel column letter, for example index 0 = column A. This operation is not limited to single letters, for example 26 = AA and 702 = AAA is supported. The library is not limited by the length of the resulting string.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Name** | **Data Type** | **Description** |
| **In Arguments** | In\_Index | Integer | The index of the column that needs to be converted to the column letter. |
| **Out Arguments** | Out\_Result | String | The resulting column letters. |

This operation uses the invoke code activities. The invoke code activity is significantly slower than implementing a similar algorithm with more fundamental activity like the assign activity. Rather use the “Index to Column Activities” operation for improved performance.

## Index To Column Letter Activities

Convert table indexes to Excel column letter, for example index 0 = column A. This operation is not limited to single letters, for example 26 = AA and 702 = AAA is supported. The library is not limited by the length of the resulting string.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Name** | **Data Type** | **Description** |
| **In Arguments** | In\_Index | Integer | The index of the column that needs to be converted to the column letter. |
| **Out Arguments** | Out\_Result | String | The resulting column letters. |

This operation mainly uses assign activities. The assign activities are significantly faster than implementing a similar algorithm with the invoke code activity. Use this (“Index To Column Activities”) operation for improved performance.

## Sum Table

Add a sum row to the bottom of a range or a sum column at the end of a range. If only a row or a column is specified only the row/column will be summed. The activity also has the capabilities to sum multiple rows and/or columns. Both the rows and columns can be summed in one activity.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Name** | **Data Type** | **Description** |
| **In Arguments** | In\_ExcelFile | Integer | The operation will be performed on the specified Excel file. The absolute path of the Excel file is needed with the file extension. The string must be quoted. |
| In\_Range | String | The range of the data to be summed. The data can be a single row/column or in table format. The string must be quoted. |
| In\_SumColumns | Boolean | If checked, the columns will be summed, and the result will be placed below the data. |
| In\_SumRows | Boolean | If checked, the rows will be summed, and the result will be placed to the right of the data. |
| In\_SheetName | String | The name of the sheet where the data can be found. The default value is “Sheet1”. The string must be quoted. |

This activity uses VBA to sum the table. The VBA instruction is stored in a module named Module5.bas which is used in the excel file to sum the data.

# Miscellaneous Library

There are many things which are reused across projects that do not fall under the above categories.

## Directories

This activity can create a list of local directories if they do not exist, or delete and recreate them if they already exist.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Name** | **Data Type** | **Description** |
| **In Arguments** | In\_Directories | String[] | The list of directories that must be created or overwritten. |

*On my machine UiPath cannot delete folders. I don’t know if it because of permission on my machine or another problem. I implemented a lot of error handling in this activity to combat this shortcoming. I observed that the delete folder activity deletes all the files in the folder, but not any sub folder or the intended folder. This approximately acts as if the folder is deleted and recreated, provided there is no subfolders.*

## Local Report File

Create a local report file, update a local report file, and read a local report file. This can be useful when tracking exceptions.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Name** | **Data Type** | **Description** |
| **In Arguments** | In\_ReportLocation | String | The directory where the report file will be stored. The string must be quoted. |
| In\_ReportName | String | The name of the report. The string must be quoted. |
| In\_LineMessage | String | The message to be stored in the report file. The message will be preceded by the date and time. |
| In\_CreateNew | Boolean | If checked, a new file is created in the specified location. If the file already exists, the file is overwritten. The file heading contains the date and time of the file creation. |
| In\_OpenReport | Boolean | If checked, the report file is opened to allow review of the file. |

## Randomly Generate Password

A password string is randomly generated to meet the specified length. The default characters used to create the password string are all the lower case letters. Options are available to make the password more secure by including uppercase letters, numbers, and symbols.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Name** | **Data Type** | **Description** |
| **In Arguments** | In\_PasswordLength | Integer | The index of the column that needs to be converted to the column letter. Default is 20 characters long. |
| In\_UpperCase | Boolean | If checked, the password string will contain uppercase letters. Default is false. |
| In\_Numbers | Boolean | If checked, the password string will contain numbers. Default is false. |
| In\_Symbols | Boolean | If checked, the password string will contain symbols. The symbols are limited to:  ! # $ % & \* + @. Default is false. |
| **Out Arguments** | Out\_Password | String | The resulting password string. |

The algorithm that was implemented in this activity is briefly explained here. The activity generates random numbers. If the number is within the allowed range, the number is converted to a letter according to the ASCII value table. If the number is not allowed a new number is generated. The allowed numbers depend on the allowed letters which is summarised in the table below.

Table 1: ASCII Values

|  |  |
| --- | --- |
| **Description** | **ASCII Numbers** |
| Symbols | 33!, 35#, 36$, 37%, 38&, 42\*, 43+, 64@ |
| Numbers | 48-57 |
| Upper Case | 65-90 |
| Lower Case | 97-122 *(Default)* |