



Excel best practices

Project name: Excel Best practices - Cloud Studio

Package version: 1.3.15

| Version | Date | Description |
|---------|-------------------|---|
| 1 | January 12, 2021 | Document created |
| 1.1 | March 08, 2021 | Change template Add information about SDK dependencies |
| 1.2 | May 03, 2021 | Add automation to manage the copy of very large table Add paragraph for automation Split large tables into multiple files Package version: 1.1.4 |
| 1.3 | June 14, 2021 | Add automation to format and export a table. Add paragraph for automation Format and Export Package version: 1.2.0 |
| 1.4 | August 12, 2021 | Changing version of the package |
| 1.5 | December 17, 2021 | Add automation to present how to sort data Package version 1.2.7 |
| 1.6 | January 17, 2022 | Add automation to present how to filter data |
| 1.7 | April 29, 2022 | Add automation to present how to split large files into a fixed number of files |
| 1.8 | May 24, 2022 | Add automation to present how to use formulas in Excel spreadsheet |
| 1.9 | October 11, 2022 | Add automation to present how to retrieve an already opened Excel file |

TABLE OF CONTENTS

| | |
|--|----|
| INTRODUCTION | 4 |
| IMPORTANT RECOMMANDATION | 5 |
| General | 5 |
| Reuse the sample as a new project | 5 |
| DESCRIPTION | 7 |
| Settings | 7 |
| <i>Environment variables</i> | 7 |
| <i>Dependent packages</i> | 7 |
| Captures | 7 |
| Datatypes | 7 |
| <i>Excel data</i> | 7 |
| Automations..... | 7 |
| <i>Get range definition</i> | 7 |
| <i>Copy dynamic table</i> | 8 |
| <i>Split large tables into multiple files</i> | 9 |
| <i>Format and Export</i> | 10 |
| <i>Sort data and export to CSV</i> | 11 |
| <i>Filter data and export to CSV</i> | 12 |
| <i>Split large tables into fixed number of files</i> | 13 |
| <i>Use formulas</i> | 14 |
| <i>Use already opened Excel workbook</i> | 15 |
| VERSION | 17 |
| SAP Build Process Automation | 17 |
| Target application | 17 |
| PREREQUISITES..... | 18 |
| Global setup | 18 |
| Specific steps to follow before launching the agent | 18 |
| EXPECTED OUTPUT | 19 |

INTRODUCTION

This document describes the SAP Build Process Automation sample **Excel Best practices - Cloud Studio** and provides the following information:

- Description (functional and technical)
- Version used to generate this sample

It also contains information on prerequisites, such as the steps to follow before launching the agent.

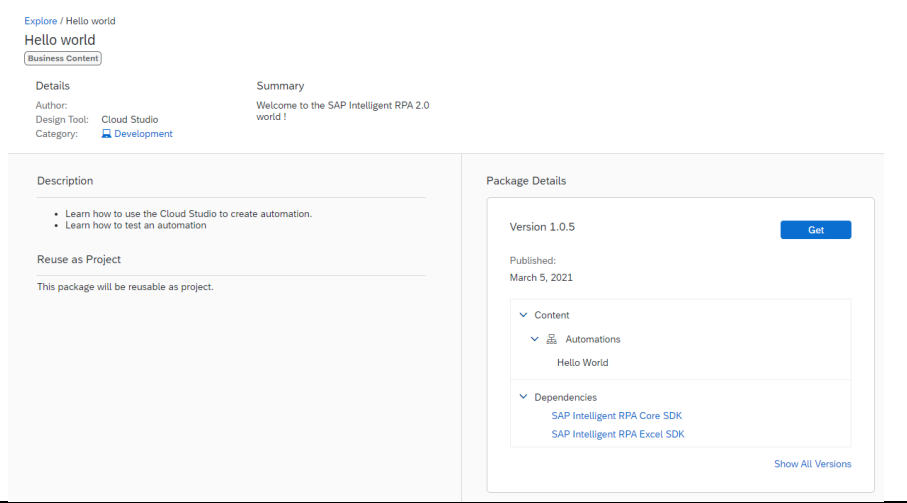
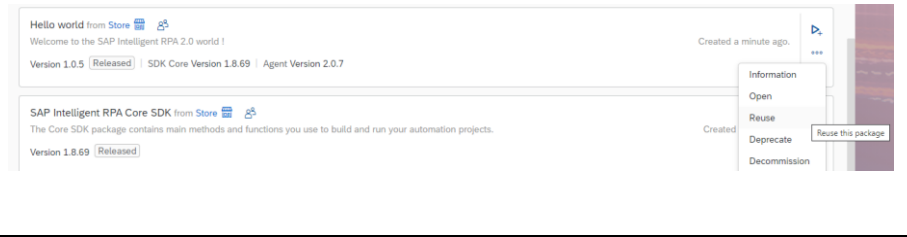
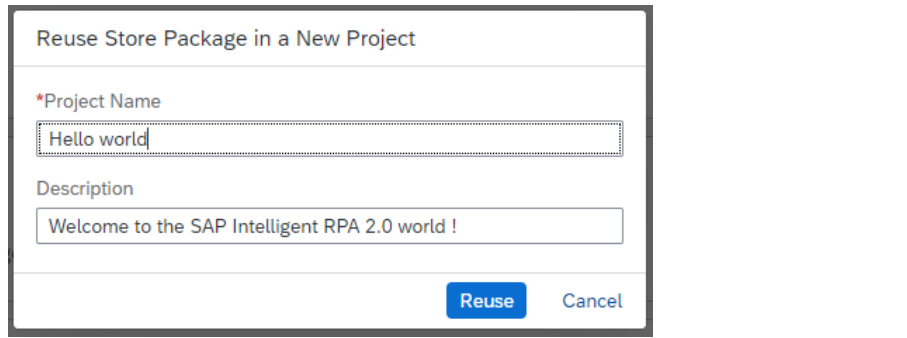
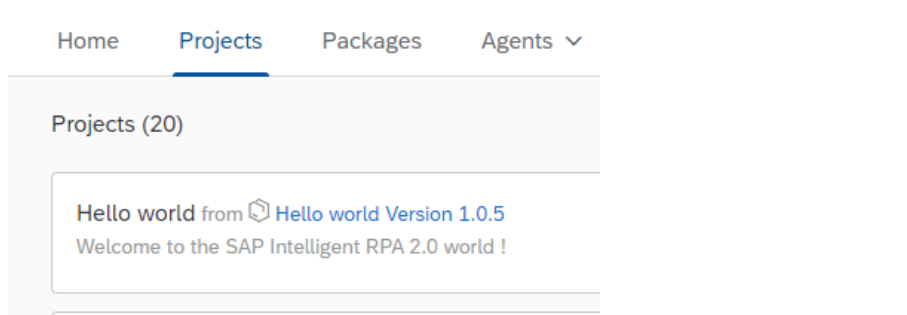
IMPORTANT RECOMMENDATION

General

To use this sample, you need to have a basic knowledge and understanding of SAP Build Process Automation tool. At the very least you need to know how to build an automation, add and modify activities and generate a package.

Reuse the sample as a new project

Note: screenshot might display a different name than the one of this sample.

| | |
|--|--|
| <p>From the Cloud Factory, open the Store tab and select the sample you want to retrieve.</p> <p>Click on the Get button.</p> |  <p>Explore / Hello world Hello world (Business Content)</p> <p>Details Author: Cloud Studio Design Tool: Cloud Studio Category: Development</p> <p>Summary Welcome to the SAP Intelligent RPA 2.0 world !</p> <p>Description</p> <ul style="list-style-type: none"> Learn how to use the Cloud Studio to create automation. Learn how to test an automation <p>Reuse as Project This package will be reusable as project.</p> <p>Package Details</p> <p>Version 1.0.5 Get</p> <p>Published: March 5, 2021</p> <p>Content</p> <ul style="list-style-type: none"> Automations <ul style="list-style-type: none"> Hello World <p>Dependencies</p> <ul style="list-style-type: none"> SAP Intelligent RPA Core SDK SAP Intelligent RPA Excel SDK <p>Show All Versions</p> |
| <p>Once the package is retrieved, open the Packages tab of the Cloud Factory.</p> <p>Click on the Options button of the package you just retrieved and select the option Reuse.</p> |  <p>Hello world from Store Welcome to the SAP Intelligent RPA 2.0 world ! Version 1.0.5 (Released) SDK Core Version 1.8.69 Agent Version 2.0.7 Created a minute ago.</p> <p>SAP Intelligent RPA Core SDK from Store The Core SDK package contains main methods and functions you use to build and run your automation projects. Version 1.8.69 (Released) Created</p> <p>Information Open Reuse Deprecate Decommission</p> <p>Reuse this package</p> |
| <p>Set a name for the project to be created.</p> |  <p>Reuse Store Package in a New Project</p> <p>*Project Name Hello world</p> <p>Description Welcome to the SAP Intelligent RPA 2.0 world !</p> <p>Reuse Cancel</p> |
| <p>Open the project that has just been created.</p> |  <p>Home Projects Packages Agents</p> <p>Projects (20)</p> <p>Hello world from Hello world Version 1.0.5 Welcome to the SAP Intelligent RPA 2.0 world !</p> |
| <p>If needed, update the content of this project, and generate a new package from it.</p> | |

You need to execute this procedure to be able to open the project and see all its content (the captured applications, the declared items, the automations, etc.).

DESCRIPTION

This package contains captures, datatype and automations that are described below. See chapter Version for more details about the version of the Desktop Agent and the SDK dependencies.

Settings

This section describes the settings of the project such as environment variables or dependent packages that are used in the automation.

Environment variables

| Name | Description | Type |
|--------------------|---|--------|
| SheetName | Default name of the worksheet created in Excel file | String |
| NumberOfRowsToCopy | Number of rows to copy when the large file is split into smaller ones | Number |
| NumberOfBlocks | Number of files to be created with splitting one large excel table | Number |

Dependent packages

N/A

Captures

This section describes the captures which were made to pilot the application in this sample. It will also describe the different methods which were used to capture the pages and declare the items.

N/A

Datatypes

This section describes the datatype used in this sample. It describes the structure of the datatype and where it is used in the automations.

Excel data

| Name of attribute | Type | Description |
|-------------------|--------|-------------|
| Attribute1 | String | |
| Attribute2 | String | |
| Attribute3 | String | |
| Attribute4 | String | |
| Attribute5 | String | |

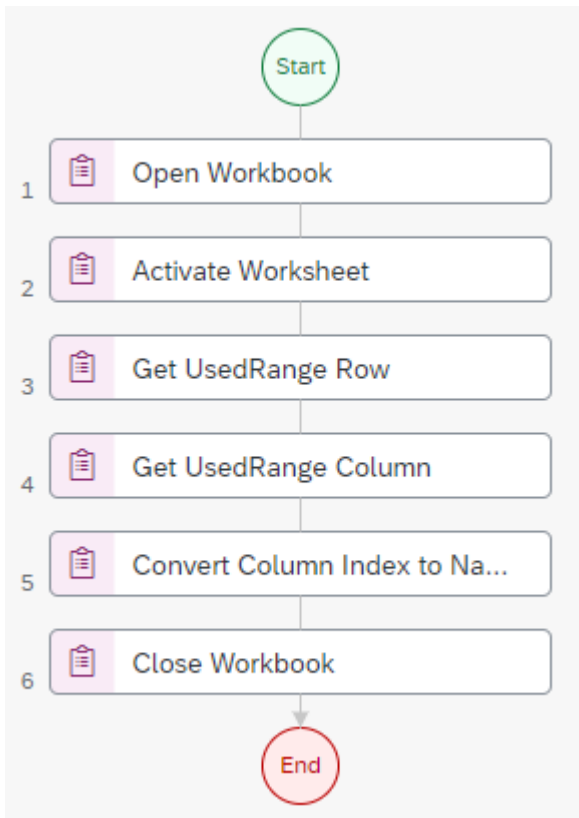
Automations

Get range definition

Triggerable: No

Input: Path of the input file

Output: Range definition of the table in Excel file (type = String)



The agent will use the variable **ExcelFilePath** to open the Excel file. Then, as we know we are working with a table, we get the number of columns first.

It will return the index of the column and row for the last cell in the range. Then it converts the column index into the column name, and return the complete range.

Important note: Let's assume we have the following data:

| Column1 | Column2 | Column3 | Column4 | Column5 | | |
|---------|---------|---------|---------|---------|--|--|
| Data 11 | Data 21 | Data 31 | Data 41 | Data 51 | | |
| Data 21 | Data 22 | Data 32 | | Data 52 | | |
| | Data 23 | Data 33 | Data 43 | Data 53 | | |
| Data 41 | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

When the agent executes the activity **Get Row From Data** in Column 1, with the reference cell begin A1 (first cell of the column), the result will be 3, as the last cell before the first empty cell is on row 3 (it's equivalent of the combination CTRL-KeyDown when Excel is opened).

Using the same pattern, the activity **Get Column From Data** in row 3 will return the value 4.

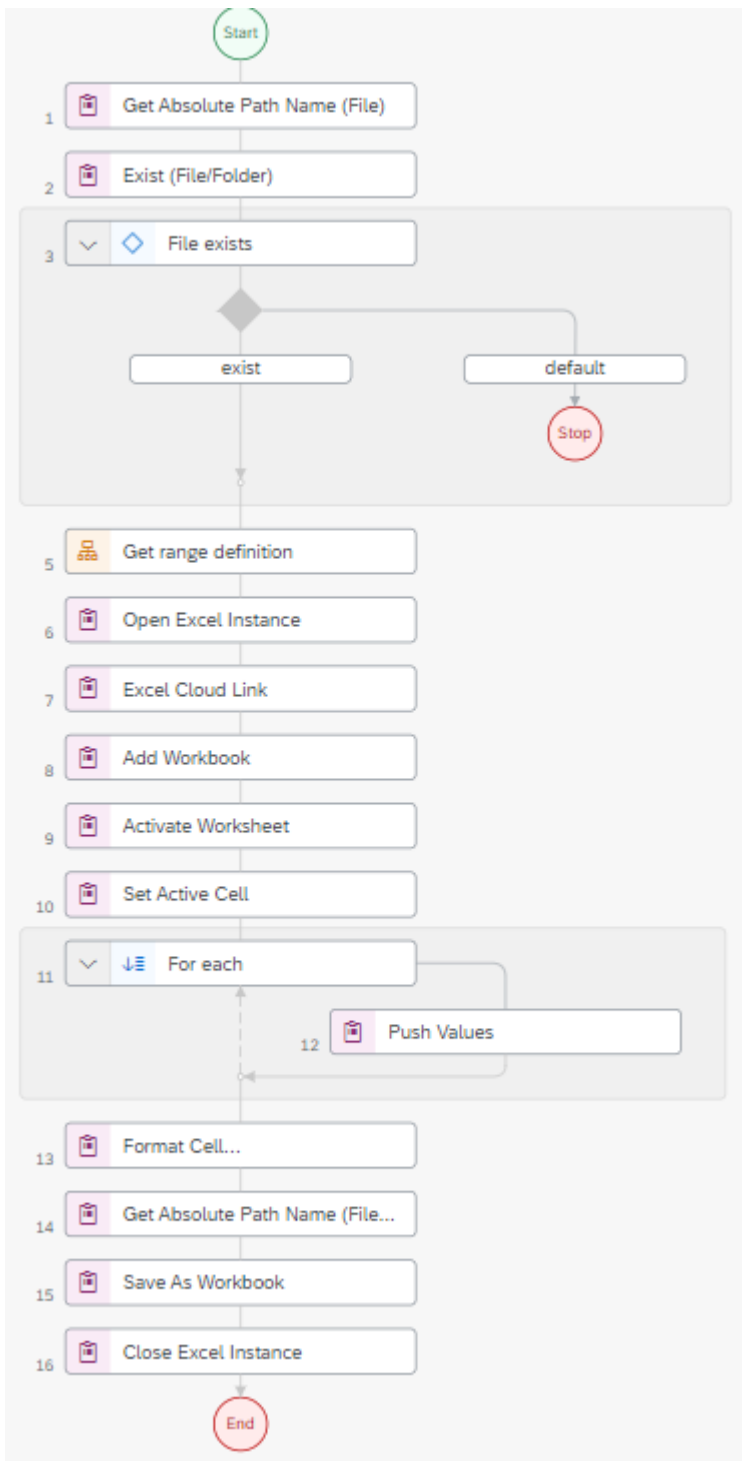
In this example, after the execution of the first loop, we would have a range definition **A1:E4**, as the cell A5 is never read. That's why we are using another loop to read data below the empty cell.

Copy dynamic table

Type: Attended

Input: None

Output: None



The agent executes the nested automation *Get range definition* to get the range to read in the Excel file and creates a collection of **Excel Data** objects using the Excel Cloud Link activity.

Then a new workflow is created where we insert the data we extracted from the previous file, using the **Push Values** activities. This activity takes an array as input parameter. So we need to specify all the data we want to write for each line (in that case, all the attributes of each **Excel Data** object).

At the end of the automation, all the cells in the new table are formatted and the Workbook is saved.

Important note: due to the way the Excel library is implemented, it is mandatory to use the *Save As Workbook* activity with a variable which the value contains \. Ex:

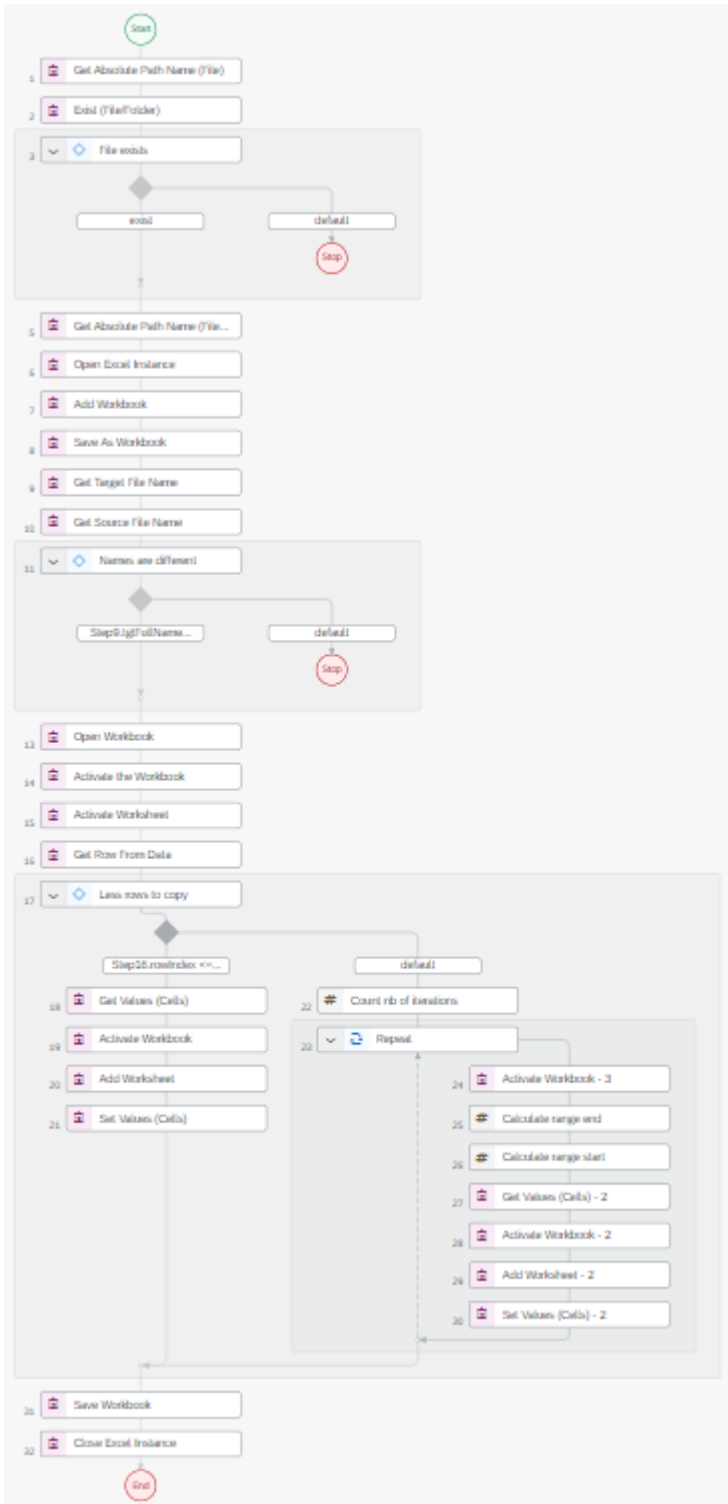
`C:\Templtest.xlsx`

Split large tables into multiple files

Type: Attended

Input: inputPath is the location of the file to be split

Output: outputPath is the location of the file to be created



In this automation, we split a table with a large number of rows into multiple smaller tables. Data are copied and pasted from one source file to another file with multiple spreadsheets.

The number of rows to be copied at once is given as a parameter using Environment variables.

There is some logic to determine the number of blocks we need to split the data into. Then the agent copies each block and paste it in a new spreadsheet into the target file.

(see [documentation](#))

Important note: due to the way the Excel library is implemented, it is mandatory to use the *Save As Workbook* activity with a variable which the value contains \. Ex:

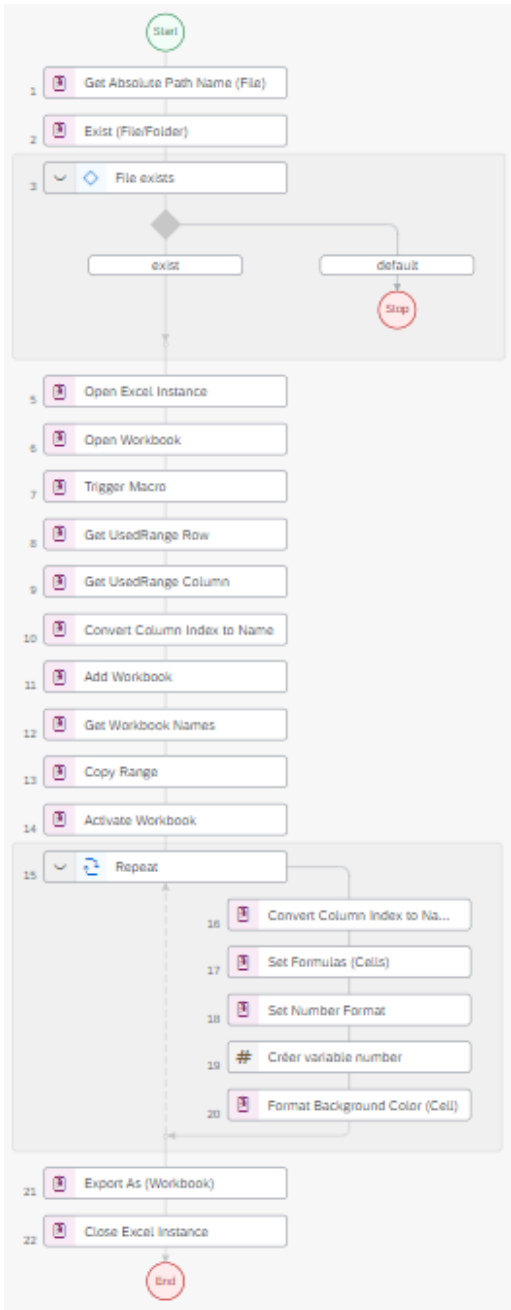
`C:\Temptest.xlsx`

Format and Export

Type: Attended

Input: inputFilePath

Output: exportPdfFilePath



The agent will use the input variable **inputFilePath** to open the Excel file.

We trigger the macro called “**class_data**” to sort the data decreasing in each column.

We get the number of rows as well as the number of columns, then convert the number of columns into its name to simplify the use of the excel range for the next steps.

Then we open a new workbook and get the names of the open workbooks to work with them. Next, we copy the data from the first excel to the second excel.

We activate the second workbook because we are going to work on this one from now on. We do a repeat loop of the number of columns. For each column we set the formula for the sum of the column to the row below the last row of data, then add some background colour.

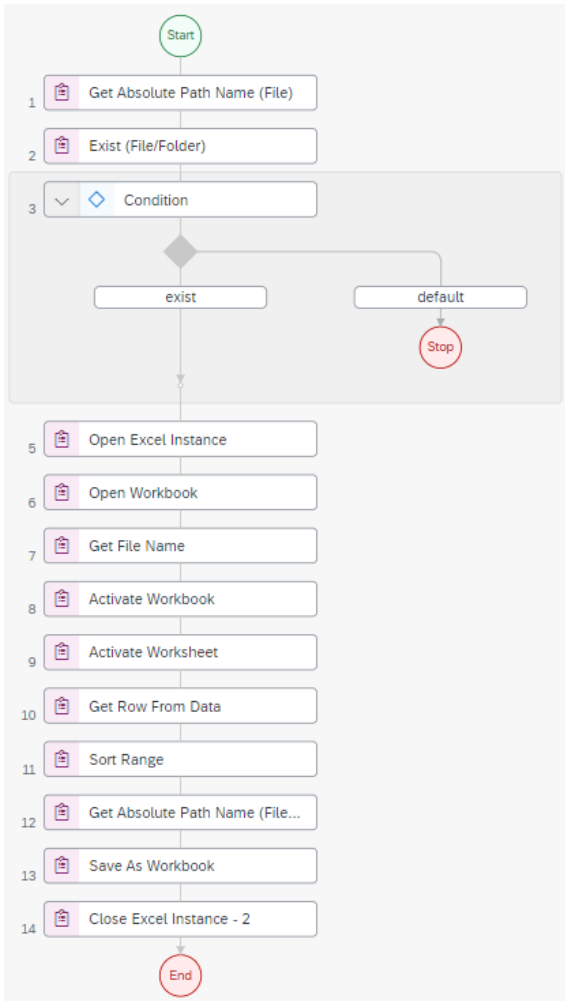
Lastly, we export the file as a pdf the path given and close excel.

Sort data and export to CSV

Type: Attended

Input: excelFilePath, outputCsvFile, columnName

Output: None



Agent opens the excel file provided as input parameter, and it will sort the data according to column name (which is also provided as parameter)

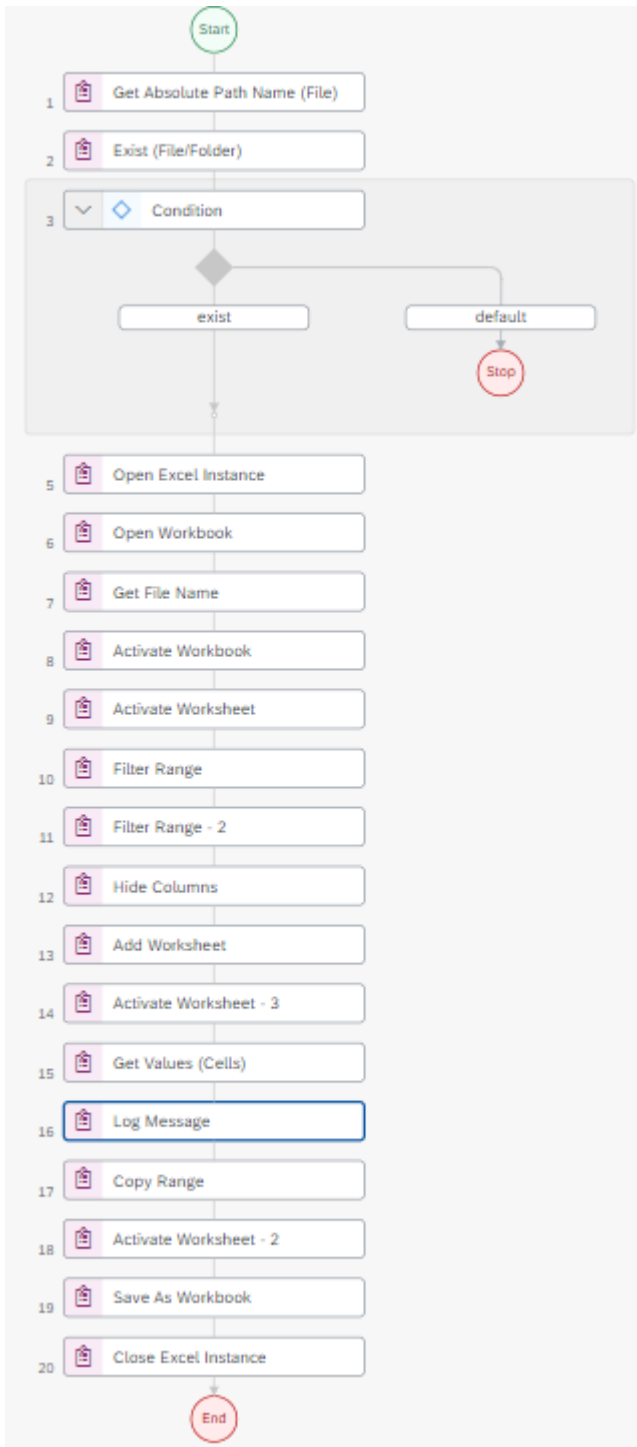
Once the data are sorted, the content of the worksheet is saved as CSV file.

Filter data and export to CSV

Type: Attended

Input: excelFilePath, outputCsvFile

Output: None



Agent opens the excel file provided as input parameter, and it will filter the data to display only rows where content in column A is greater than 100. It will apply another filter to display only rows where content in column B is lesser than 100 (first filter is still enabled)

Then the agent will copy the filtered ranged to another worksheet.

Once the data are filtered and copied/pasted, the content of the result worksheet is saved as CSV file.

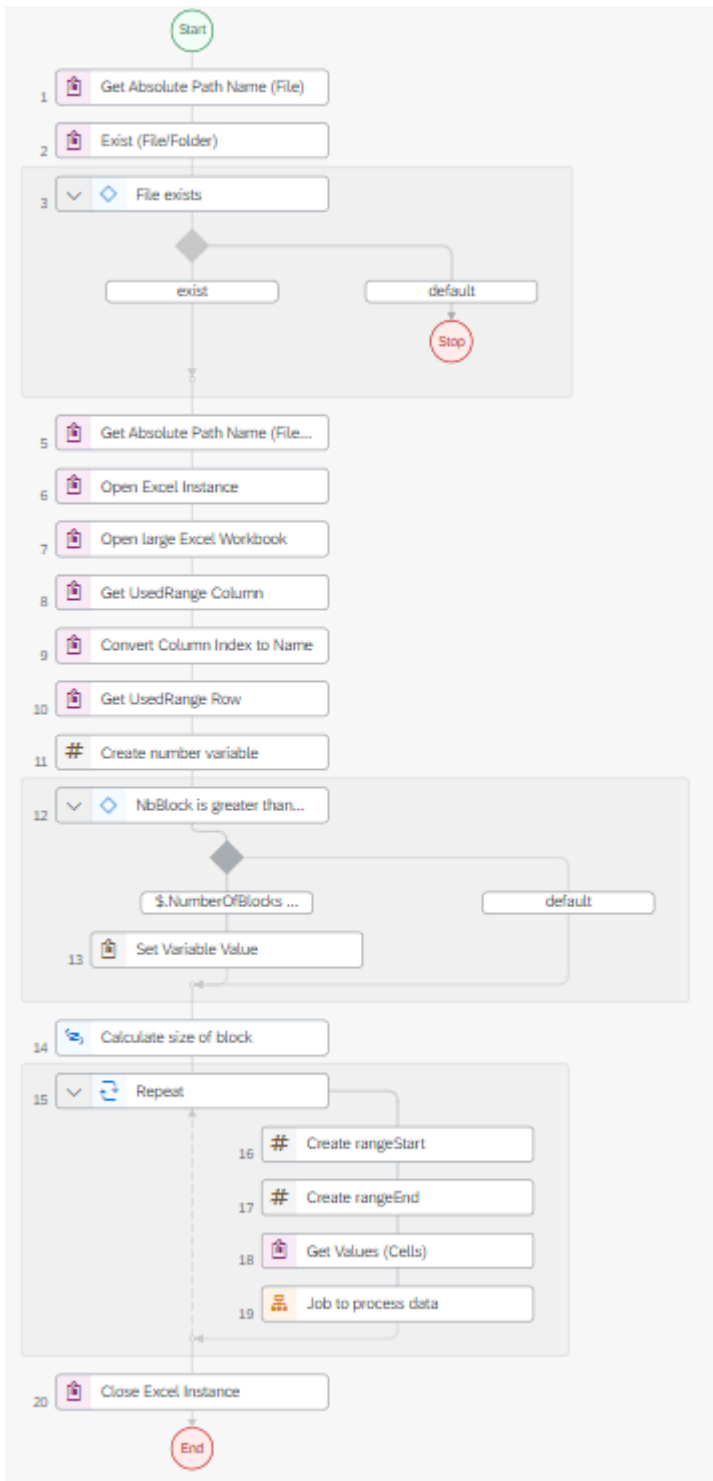
Note: The Get Values (Cells) activity is only used to demonstrate that we can also automatically retrieve data from filtered range, by setting the parameter *visibleOnly* to *true*

Split large tables into fixed number of files

Type: Attended

Input: inputPath is the location of the file to be split

Output: outputPath is the location of the folder where the files will be created



In this automation, we split a table with a large number of rows into multiple smaller tables. Data are copied and pasted from one source file to another file with multiple spreadsheets.

The number of blocks is given as a parameter using Environment variable. (one block is a part of the table)

There is some logic to determine the size of the blocks we need to split the data into. Then the agent copies each block and send these data to a sub-automation.

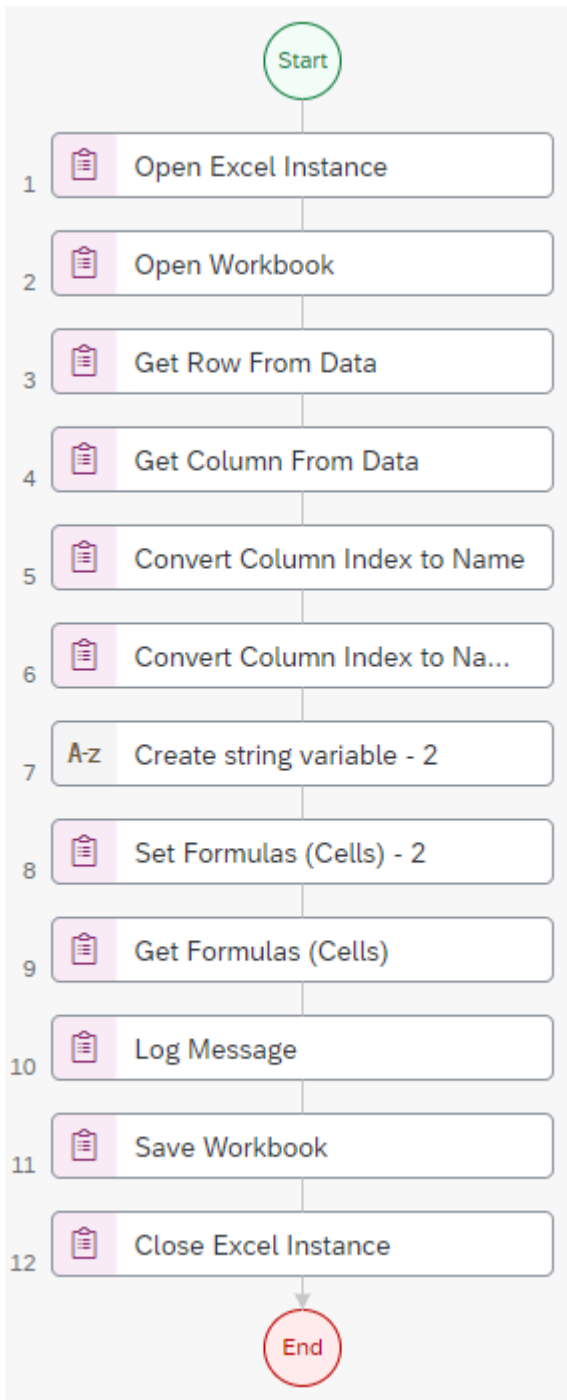
(see [documentation](#))

Use formulas

Type: Attended

Input: inputPath is the location of the file

Output:



In this automation, the bot will perform some calculations in an Excel spreadsheet using formulas.

For each line of the file, it will sum the values and insert the result in the last column.

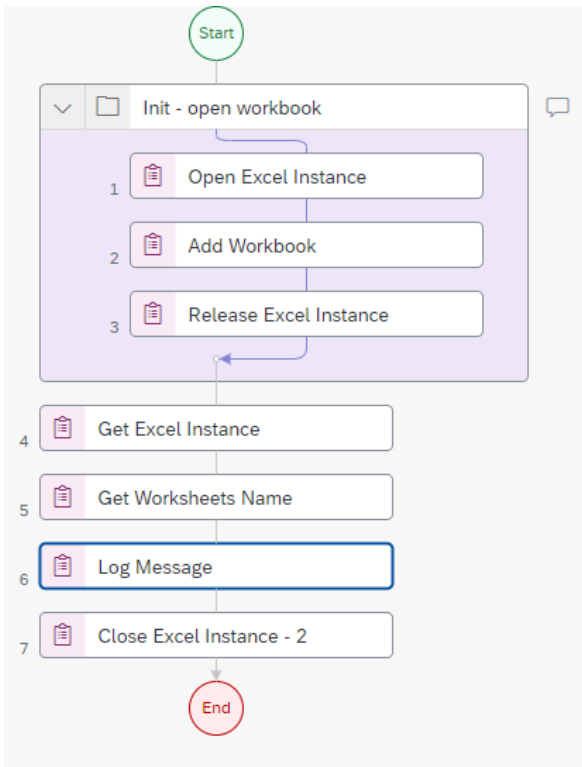
It also displays the formulas in the log console

Use already opened Excel workbook

Type: Attended

Input: N/A

Output: N/A



This automation is composed of 2 parts. The first one is the initialization and does not contain any learning content. It only opens a new Excel workbook and close the Excel instance.

The second part of the automation it the one where the bot will be able to connect to the Workbook which has been opened previously (note: this workbook can be opened manually, or by another application, such as SAP GUI).

It will then retrieve the list of worksheets and display them in the log console.

VERSION

The product versions used to generate this sample are detailed below. This sample is provided “as is”, with no warranty that it will work correctly with other versions. If some versions of your software are different (such as the tool version or the target application version), you may need to recapture the application and/or update the workflow activities.

SAP Build Process Automation

This sample targets the Desktop Agent **2.0.8** or higher.

The following SDK dependencies were used to generate this sample: 1.30.37

See [documentation](#) for more details about the compatibility between SDK version and Desktop Agent.

Target application

N/A

PREREQUISITES

Global setup

SAP Build Process Automation must be installed in accordance with the installation guide available [here](#). An SAP Build Process Automation Factory must be available with a suitable environment (containing an agent). All information can be found in the “Getting Started” section accessible via the above link.

Specific steps to follow before launching the agent

- Download the **sample.zip** archive from the sample and extract its content
- When you deploy your automation, set the environment variable **SheetName** with the name usually used by Excel when you create a new worksheet. Ex: *Sheet1*
- When you deploy your automation, set the environment variable **NumberOfRowsToCopy** with the number of rows to be copied from one spreadsheet to another. Ex: *10000*
- When you deploy your automation, set the environment variable **NumberOfBlocks** with the number of block you want to split your table into. Ex: *100*
- When executing the automation **Filter Data and Export to CSV**, the input Excel file should be the **excel_data.xlsx** that you extracted from the **sample.zip** archive

EXPECTED OUTPUT

N/A

www.sap.com/contactsap

© 2019 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company.

The information contained herein may be changed without prior notice. Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE's or its affiliated companies' strategy and possible future developments, products, and/or platform directions and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, and they should not be relied upon in making purchasing decisions.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies. See www.sap.com/copyright for additional trademark information and notices.

THE BEST RUN

