How to use User Tasks

Project name: User Task – Cloud Studio

Package version: 1.0.0

THE BEST RUN SAP

Version of document	Date	Description
1	20 August 2021	Document created

TABLE OF CONTENTS

INTRODUCTION

This document describes the SAP Build Process Automation sample **User Task – 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.

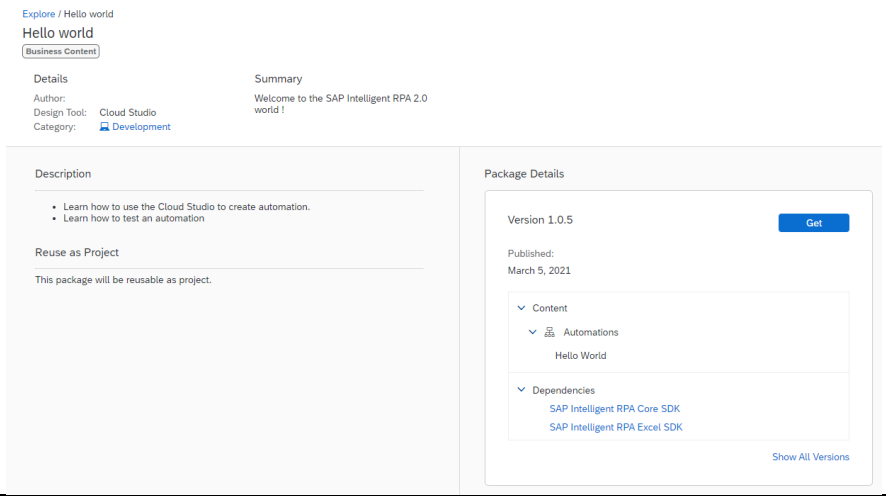
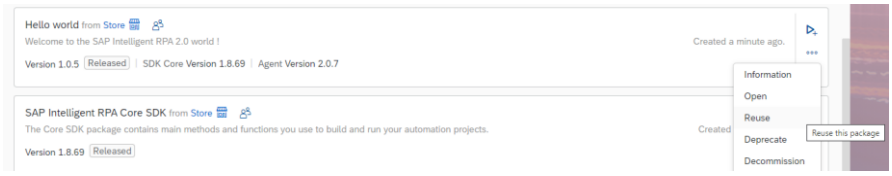
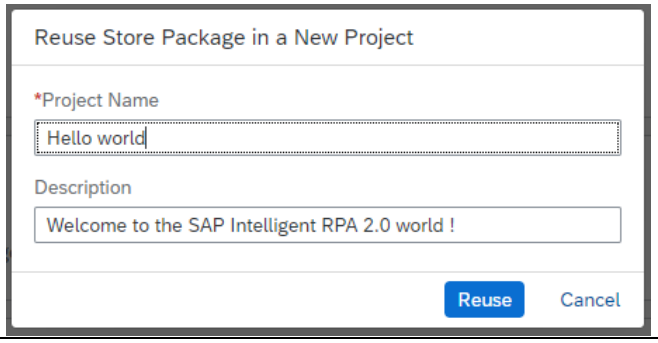
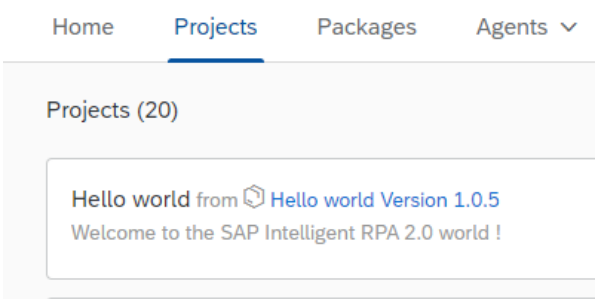
IMPORTANT RECOMMANDATION

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.

From the Cloud Factory, open the Store tab and select the sample you want to retrieve. Click on the **Get** button.	
Once the package is retrieved, open the Packages tab of the Cloud Factory. Click on the Options button of the package you just retrieved and select the option **Reuse**.	
Set a name for the project to be created.	
Open the project that has just been created.	
If needed, update the content of this project, and generate a new package from it.	

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
folderPath	Folder path to store configuration files	String

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.

Configuration Item

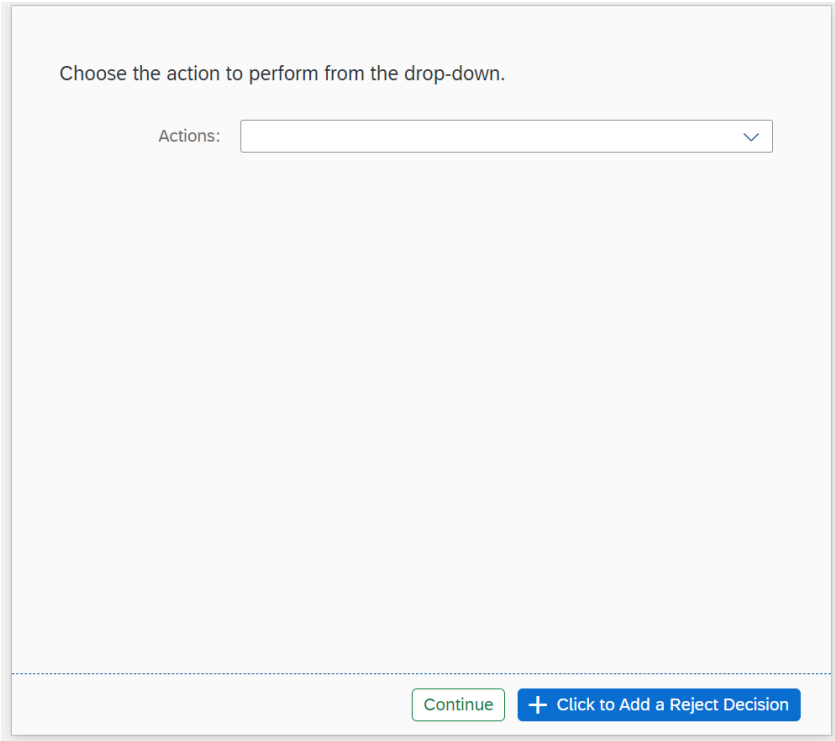
Name of attribute	Type	Description
Name	String	Name of the user
Country	String	Country from which user belongs to
Hobbies	String	List of user's hobbies
Agree	Boolean	If true, agree to policy

User Tasks

This section describes the user tasks used in this sample. It describes the structure of the user task.

Main Menu

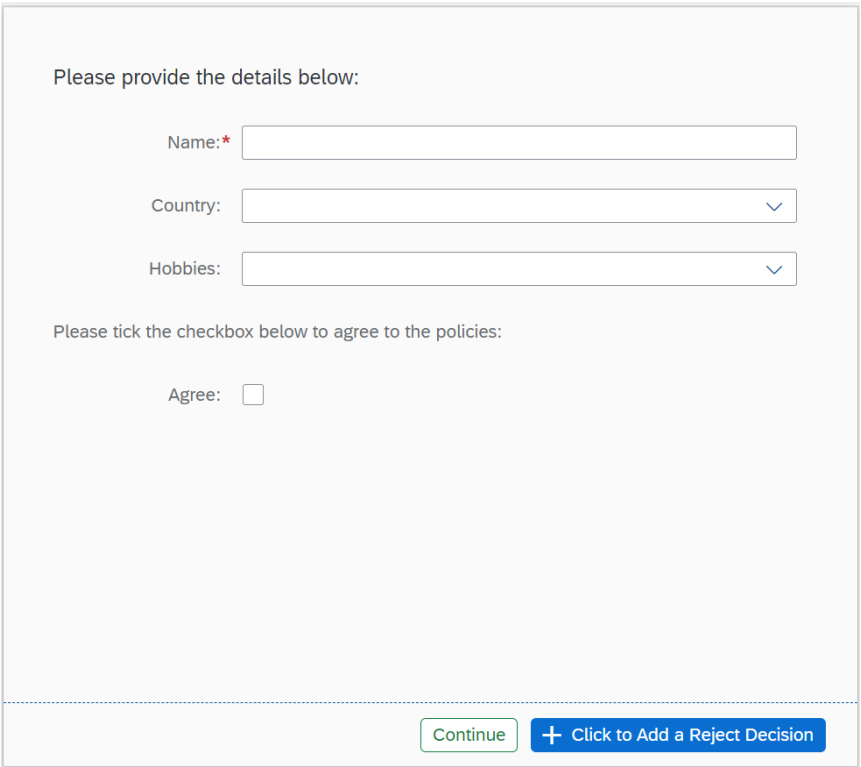
Input: options (String) [List]
Output: decision (String), processor (String), status (String), selectedOption (String)



This user task uses input **options** which is a list of actions that the user can perform – create, edit or delete configuration file.

Output is the user selected option which gets stored in the variable **selectedOption.**

Create File
Input: countryList (String) [List], hobbiesList (String) [List]
Output: decision (String), processor (String), status (String), selectedName (String), selectedCountry (String), selectedHobbies (String) [List], selectedAgreed (Boolean)

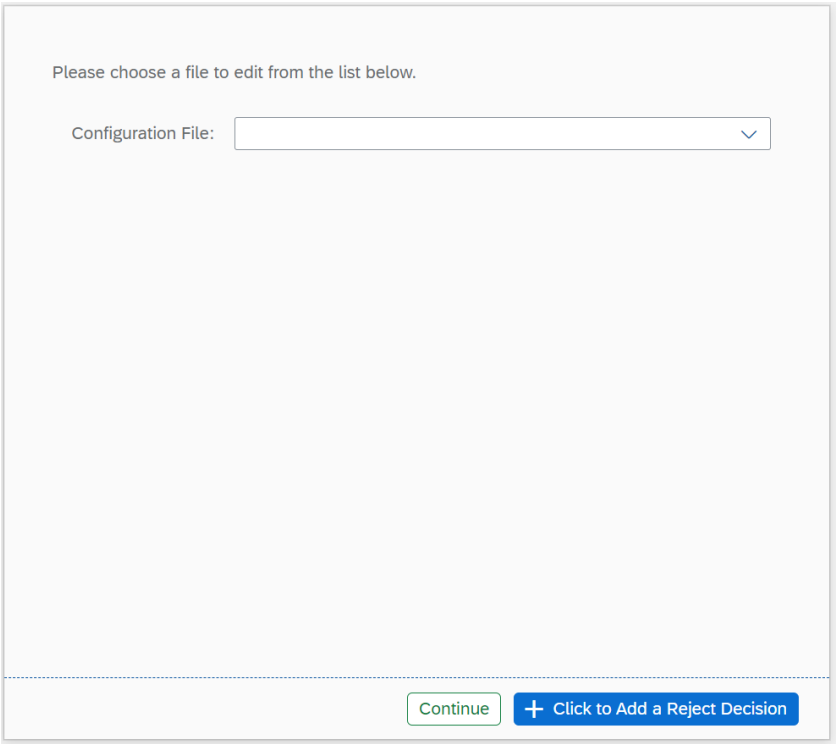


This user task is responsible for creating a configuration file with the information provided in different components.

Values in drop-down for Country and Hobbies come from list **countryList** and **hobbiesList** respectively.

Name gets stored in **selectedName**. Country chosen is stored in variable **selectedCountry**. Multiple hobbies can be selected and gets stored as a list in **selectedHobbies**. Agree value is saved in **selectedAgreed**.

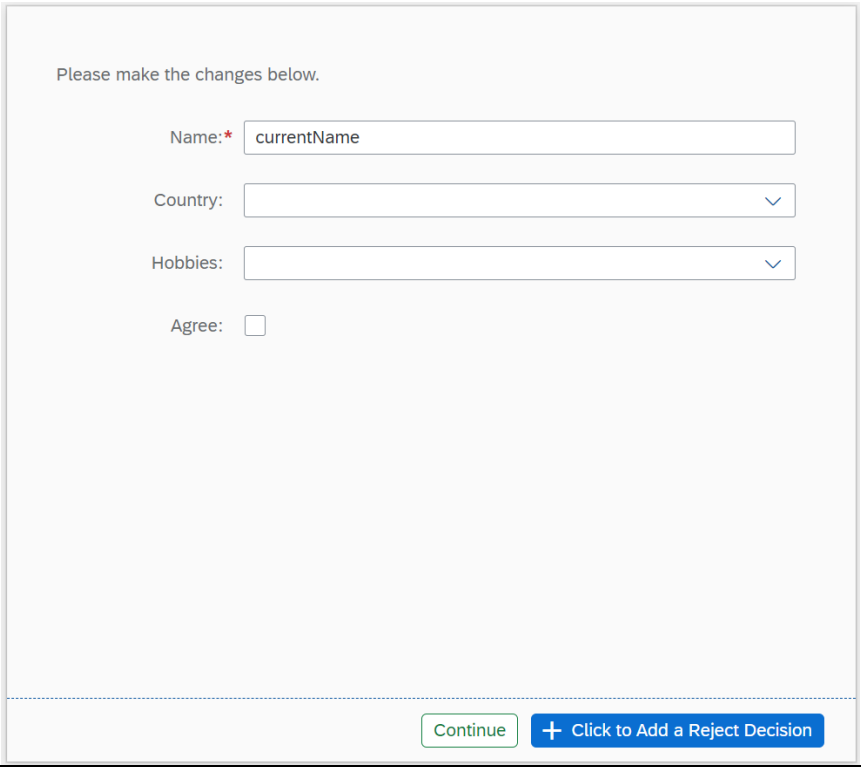
Edit File
Input: filesToEdit (String) [List]
Output: decision (String), processor (String), status (String), selectedFileToEdit (String)



This user task gets the list of all the configuration files created in the environment variable, **folderPath**. The list of files is stored in variable **filesToEdit**.

User can choose any file to edit and the output gets saved in **selectedFileToEdit.s**

Make Your Changes
Input: currentName (String), countryList (String) [List], hobbiesList (String) [List], currentAgreedStatus (Boolean), item {Configuration Item}
Output: decision (String), processor (String), status (String), updatedName (String), updatedCountry (String), updatedHobbies (String) [List], updatedAgreedStatus (Boolean)

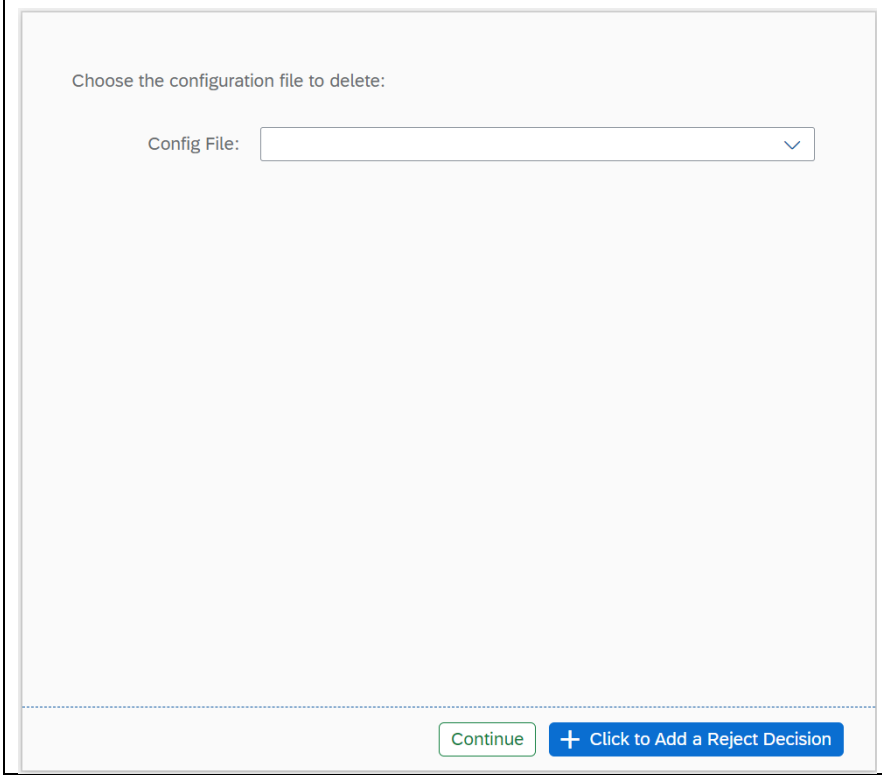


The user task is triggered from automation **editConfigFiles** and is responsible for making the changes.

The Name attribute saved in the configuration file gets fetched from **currentName** variable and is shown in the user task. Similarly, the initial value of Agree attribute is stored in **currentAgreedStatus**. The components Country and Hobbies are a static list of all the countries, **countryList** and hobbies, **hobbiesList**. User can choose from the respective lists.

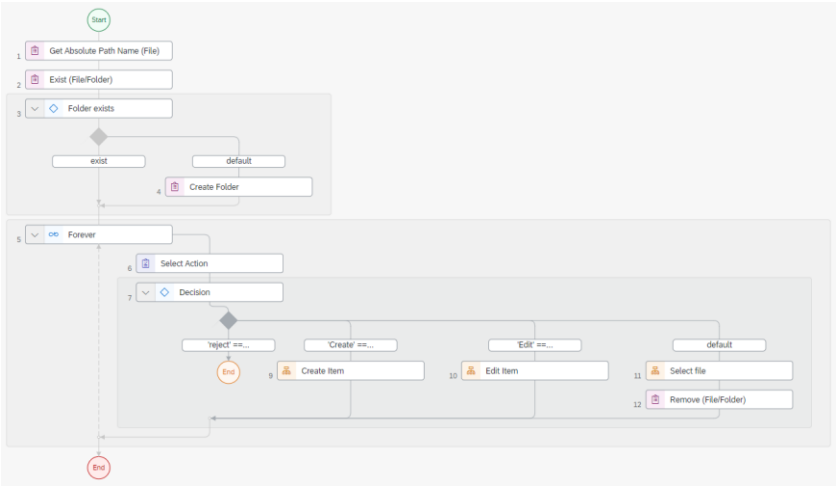
The chosen values for Name, Country, Hobbies and Agree are saved in variables **updatedName**, **updatedCountry**, **updatedHobbies** and **updatedAgreedStatus** respectively.

Delete File
Input: filesListforDeletion (String) [List]
Output: decision (String), processor (String), status (String), selectedFileToDelete (String)

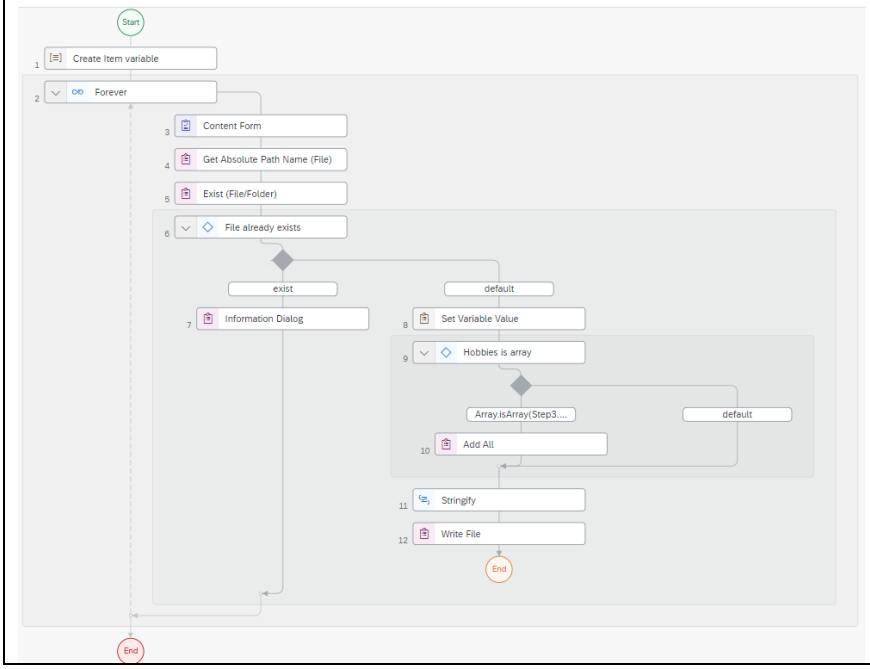
	The user task is responsible for deleting the chosen configuration file. The list of existing configuration files is saved in **filesListforDeletion**. The selected file name for deletion gets stored in variable **selectedFileToDelete** and the file is then, deleted.

Automations

manageConfigFiles
Type: Attended
Input: N/A
Output: N/A

	This automation is the main entry point. The agent will use the environment variable **folderPath** to check whether the folder exists or not and create/ edit/ delete the configuration files in that folder. The path existence is checked. If the path does not exist, then, folder is created. Else user is provided with the user task **Main Menu** to choose the action to perform.

createConfigFiles
Type: Attended
Input: N/A
Output: N/A

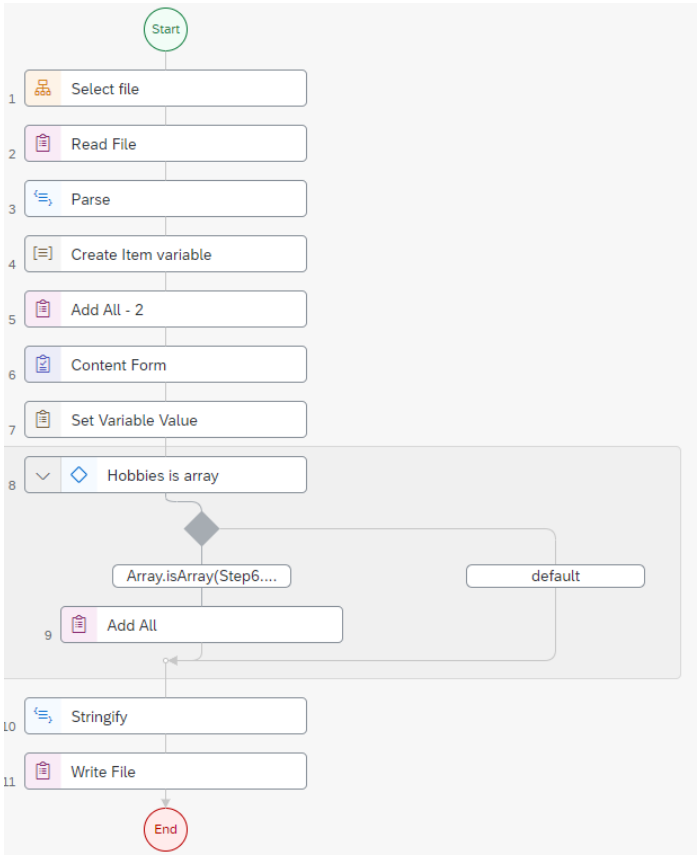


This automation allows user to create new configuration files in the path provided in environment variable **folderPath.**

A configuration file is created using the variable **selectedName** provided in user task **Create File**. File format is ".json".

A forever loop checks for the existence of the file with the same name. If file name already exist, then, user is prompted with a message dialog. If not, then, the configuration file is created.

editConfigFiles
Type: Attended
Input: N/A
Output: N/A



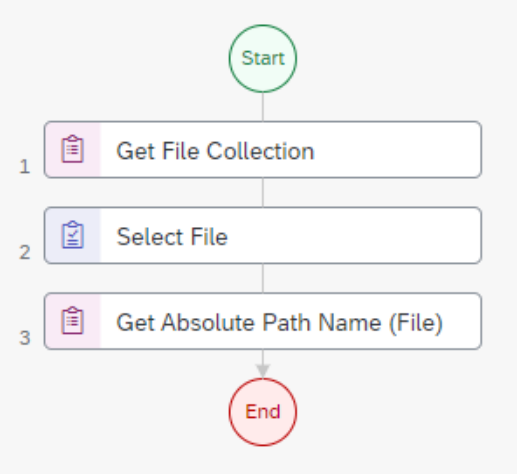
This automation takes the environment variable **folderPath** to get the list of files to edit and then, finally, write the changes at the same path.

The list of files present in the path provided are shown to the user in user task **Edit File**. The selected file is then, read and parsed. The respective values are mapped to different fields in data type **Configuration Item**.

The user task **Make Your Changes** allows user to make relevant changes to the file.

The custom script use stringify function to concatenate the changes and then, write the changes in the same file without changing the file name.

selectFile
Type: Attended
Input: N/A
Output: N/A

	It uses the environment variable **folderPath** to get the list of configuration files created. User task **Delete File** asks the user to select the file to work with.

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 **2.0.15** or higher.

The following SDK dependencies were used to generate this sample:

irpa_core	1.14.54
irpa_excel	1.14.54
irpa_outlook	N/A
irpa_pdf	N/A
irpa_ui5	N/A
irpa_word	N/A
irpa_powerpoint	N/A

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

- When you deploy your automation, set the environment variable **folderPath** with the location where configuration files need to be managed.

EXPECTED OUTPUT
N/A

www.sap.com/contactsap

THE BEST RUN SAP®