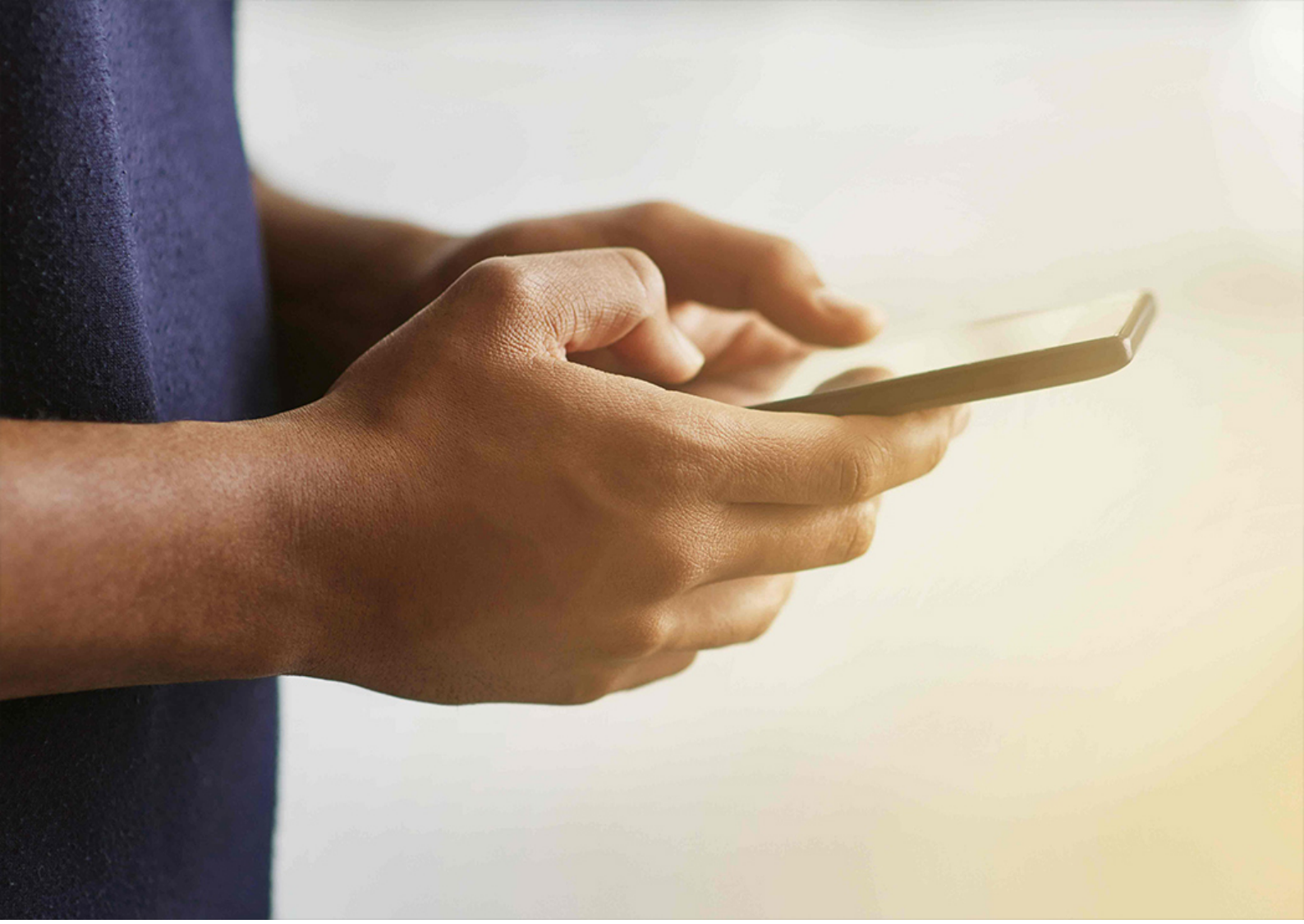


PUBLIC
2021-09-30

SAP Intelligent RPA Cloud Studio User Guide

© 2021 SAP SE or an SAP affiliate company. All rights reserved.

THE BEST RUN SAP

Content

1 Overview . . . 5

2 Getting Started with Cloud Studio. . . . 6

2.1 Web Search Automation Tutorial. . . . 6

Prerequisites. . . . 6

Create a Project. . . . 6

Capture the Search Engine. . . . 7

Declare Elements. . . . 11

Capture the Search Results Screen. . . . 14

Create your Automation. . . . 23

Test your Automation. . . . 36

3 Design Automations Quickly with the Recorder. . . . 38

3.1 Overview. . . . 38

3.2 Prerequisites. . . . 42

3.3 Automatically Capture Applications with the Recorder. . . . 45

Implement Collection in Recorded Elements. . . . 56

Handle Multiple Screens with the Same Criteria in SAP WinGUI Recorder. . . . 61

Edit a Custom Activity in Generated Automation. . . . 64

3.4 Enable Logs for Recorder in TraceViewer. . . . 65

3.5 Recorder for Web GUI. . . . 66

Best Practices. . . . 68

Limitations. . . . 69

3.6 Recorder for Win GUI. . . . 69

Best Practices. . . . 71

Limitations. . . . 71

3.7 Error Management. . . . 72

3.8 Best Practices. . . . 73

4 Capture Applications and Design Automations. . . . 75

4.1 Capture and Declare Applications. . . . 75

Overview. . . . 76

Capture an Application. . . . 77

Declaration Phase. . . . 101

Search Option. . . . 126

Test an Application. . . . 129

Technology Connectors. . . . 132

4.2 Design Automations 153

Create an Automation. 156
Automation Tools Panel Filters. 159
Manage Data within an Automation. 160
Add an Activity to an Automation. 208
Add an Application Screen to an Automation. 214
Add a Control to an Automation. 216
Add a User Task to an Automation. 262
Generate a Sub-Automation. 264
Trigger an Automation as a New Job. 265
Monitor an Automation Using the Console. 267
Document Information Extraction. 269
Support Multi-level Collections in an Automation. 280
Test an Automation. 282
Manage Errors within an Automation. 309
Tips and Tricks. 342
4.3 Trigger Automations in Attended Mode using the Project Pane 352
4.4 Alerts. 357
4.5 Manage Files. 360
Create a File. 360
Import and Edit Files. 362
Integrate Files in Automation. 364
4.6 Import Files. 367
4.7 Business Activity Monitoring (BAM). 368

5 Automation Activities provided by SDK Packages. 371
5.1 SDK Packages. 372
SDK Core Package. 373
SDK SAPUI5 Package. 395
SDK SAP WebGUI Package. 401
SDK Excel Package. 403
SDK Word Package. 418
SDK Outlook Package. 424
SDK PowerPoint Package. 447
SDK PDF Package. 448
SDK BAPI Package. 458
SDK ARIBA Package. 472
5.2 Lifecycle of SDK Activities. 474

6 Reuse Desktop Packages and Build Processes. 476
6.1 Reuse Desktop Scenarios. 476
Data Types Created When a Scenario is Imported. 477
6.2 Processes. 482

Create a Process. . . . 483
Provide Links to User Inboxes for User Tasks. . . . 488
6.3 User Tasks. . . . 489
Define the Settings of the User Task. . . . 491
Insert Components and Define Parameters to Build a User Task. . . . 493
Define Input and Output Parameters. . . . 505

7 Reuse, Deploy Projects and Packages 507
7.1 Reuse a Package from the Store. . . . 507
7.2 Generate Packages. . . . 508
7.3 Environment Variables. . . . 509
Best Practices for Environment Variables of Type Password. . . . 512
7.4 Manage Compatibility of Projects with Agents. . . . 513
Configure an Agent Version. . . . 513
7.5 Export a Project. . . . 519
7.6 Restore a Project. . . . 520
7.7 Add Agent Attributes to a Project. . . . 521

8 Dependencies. . . . 525
8.1 Manage Project Dependencies. . . . 526
Add a New Dependency. . . . 528
Update a Dependency. . . . 530
Delete a Dependency. . . . 532
Decommission a Dependency. . . . 533
8.2 Project Dependency Removal and Re-creation. . . . 534

9 Miscellaneous. . . . 536
9.1 Artifact Integration Within a Project. . . . 536
9.2 Collaborative Editing. . . . 537
9.3 Duplicate Artifact. . . . 539
9.4 Deactivate an Artifact. . . . 541
9.5 Glossary. . . . 547

1 Overview

The SAP Intelligent RPA Cloud Studio web application offers a Low-code and No-code approach enabling you to design bots and automate your applications.

With a Low-Code approach, developers and business users can build and run their application bots with their Javascript hand-coding knowledge.

With a No-Code approach, users are guided through an intuitive interface with a simple workflow design. The simple-to-use bot building capabilities offered in the cloud studio allow expert developers as well as business users to build and edit cloud projects without writing a line of code.

To build your cloud projects, you can import a number of artifacts in the Cloud Studio:

- Projects extracted from the Desktop Studio as packages, called *Desktop packages*. You then reuse the *scenarios* from these *Desktop packages* as *skills* to build *processes*, or directly trigger the *scenarios*.
- Dependencies [page 525] to your project, containing SDK packages with activities [page 371] you can use to build your automation. The activities are displayed as modules and data types.
- Desktop project archives or other documentation files.

In addition, you can create the following artifacts to build your automation in the Cloud Studio:

- Capture and Declare Applications [page 75]: For capturing applications running on your local machine, and recognizing application screens and UI elements to use in your automation.
- Design Automations [page 153]: Succession of steps to orchestrate multiple activities and applications on a local machine.
- Data Types [page 168]: For managing the data structure.
- Processes [page 482]: Sequence of skills (imported scenarios, user tasks, or other processes) designed to automate one or more tasks.
- Trigger Automations in Attended Mode using the Project Pane [page 352]: For designing the agent systray directly from the Cloud Studio.
- User Tasks [page 489]: For enabling user interaction while the automation is running (Workflow inbox).

Once you have finished designing your cloud project, you can generate a package from it. The generated package will automatically be available in the Cloud Factory for deployment.

2 Getting Started with Cloud Studio

This section groups easy-to-follow tutorials that help you create your first projects.

2.1 Web Search Automation Tutorial

In this tutorial, you will go through the different steps you need to follow to create your first basic project with an application and an automation in the Cloud Studio.

You will learn how to capture and declare a web application, and how to add it to an automation. The goal of the automation will be to automatically search for the key word SAP Intelligent RPA in a web search engine and to gather the result information.

> **i** Note
>
> To learn how to perform a query on a web search engine, you can download the Pilot a web search engine sample on the SAP Intelligent RPA Store.

2.1.1 Prerequisites

Before you work through this tutorial, make sure an agent has been properly connected to your tenant. Otherwise, you will not be able to complete the tutorial. For more information, see the Desktop Agent Tenant Registration in the Installation Guide.

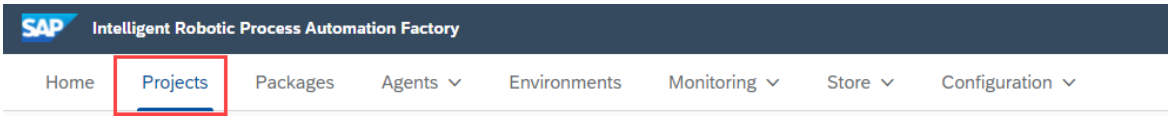
Make sure you have created an environment and added your agent to this environment. For more information, see Create and Manage Environments in the Factory User Guide.

In the following tutorial, you will capture a web application: launch Internet Explorer and enter the URL of your favorite search engine. Here we will use https://duckduckgo.com as an example.

2.1.2 Create a Project

Procedure

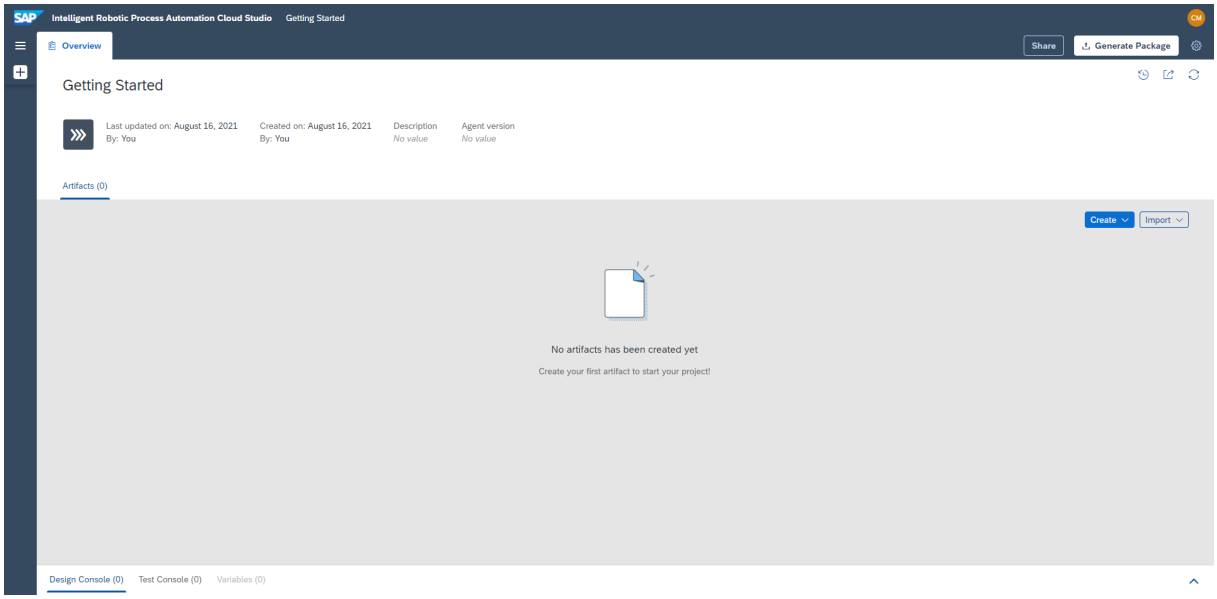
1. In SAP Intelligent RPA Factory, click the *Projects* tab in the top navigation bar.



2. Click the *New Project* button. A *New Project* window pops up.
3. Enter a project name and a description (optional).
4. Click *Create*.

Results

The Cloud Studio opens in another window.



2.1.3 Capture the Search Engine

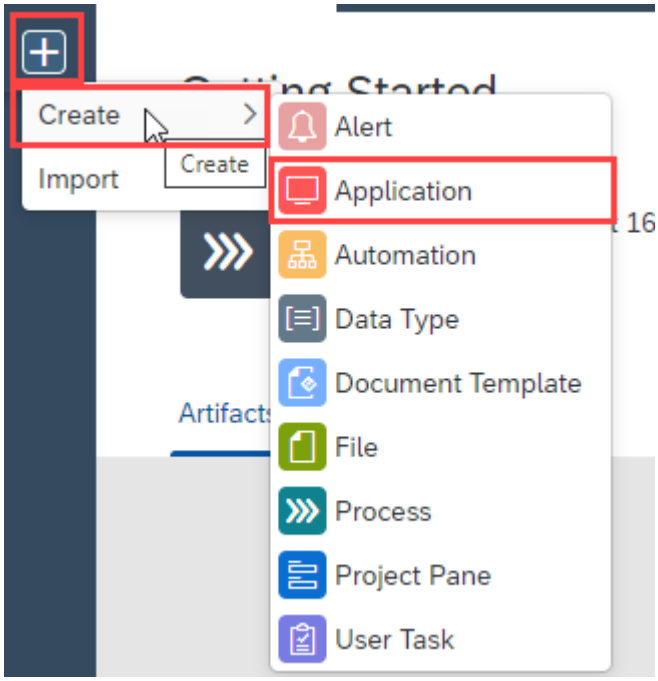
Prerequisites

You have kept the window zoom factor same on every screen while capturing with Chrome and Firefox in Cloud Studio as described in Best Practice [page 81].

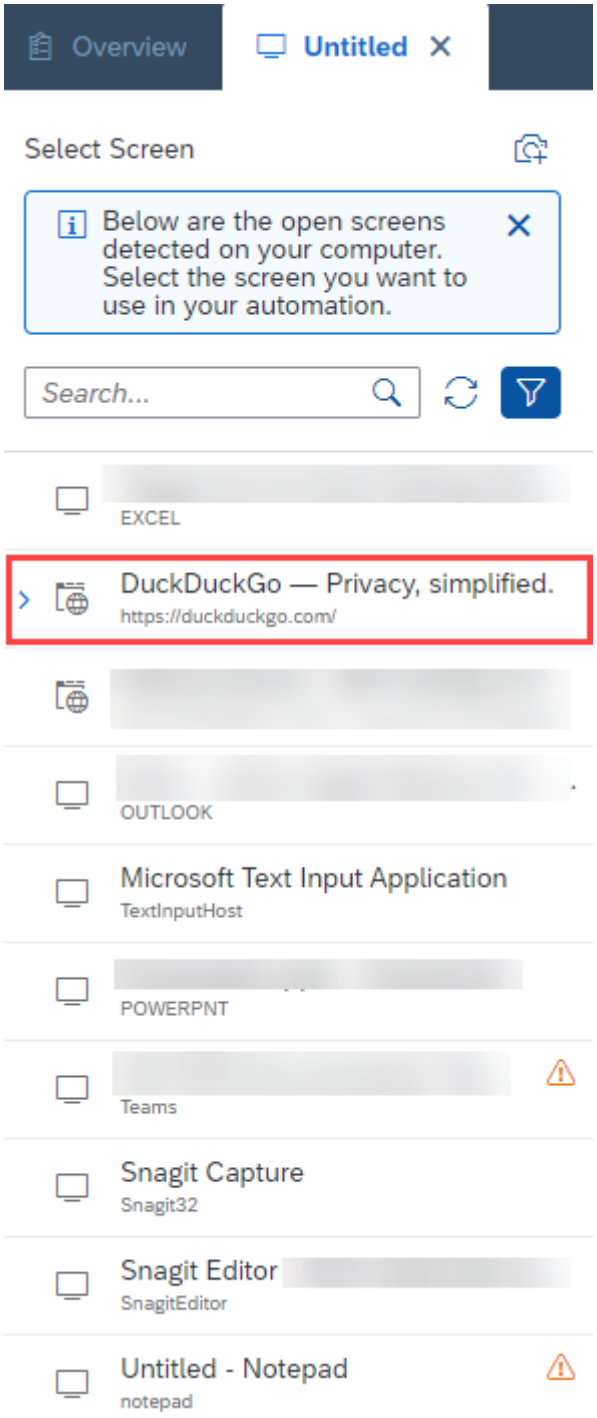
Context

Procedure

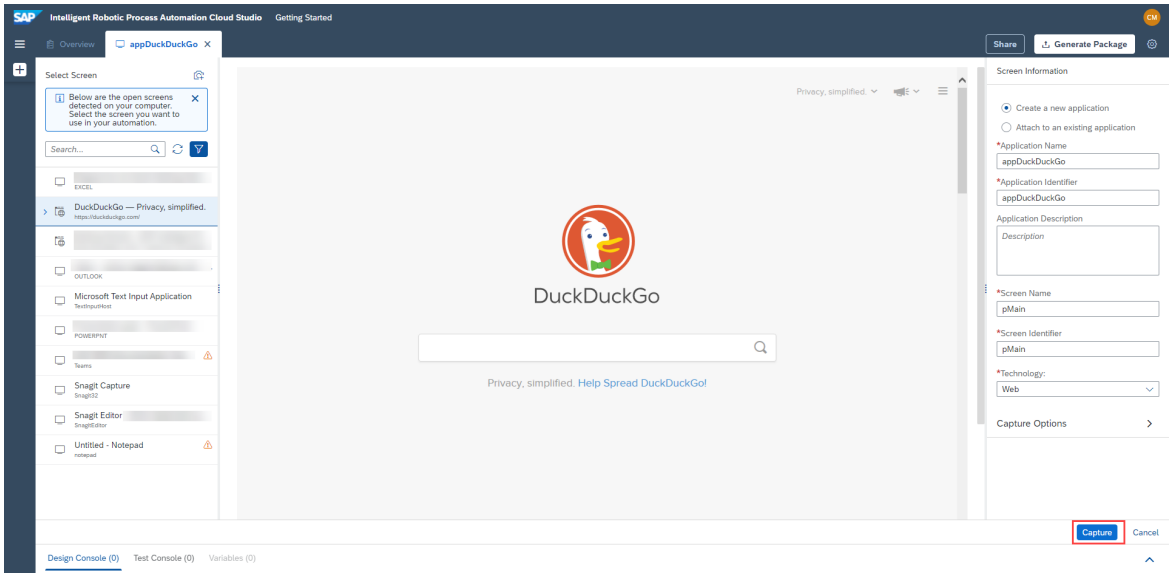
1. In SAP Intelligent RPA Cloud Studio, click the + icon and then the *Create* button on the left-side panel. A dropdown menu opens.
2. Select *Application*.



3. In the *Select Screen* side panel, select the DuckDuckGo application from the list. A preview image of the screen is displayed in the capture area.



4. In the *Screen Information* side panel, enter an *Application Name* and a *Screen Name*.
   We enter `appDuckDuckGo` as the *Application Name* and `pMain` as the *Screen Name*.
5. Click *Capture*.

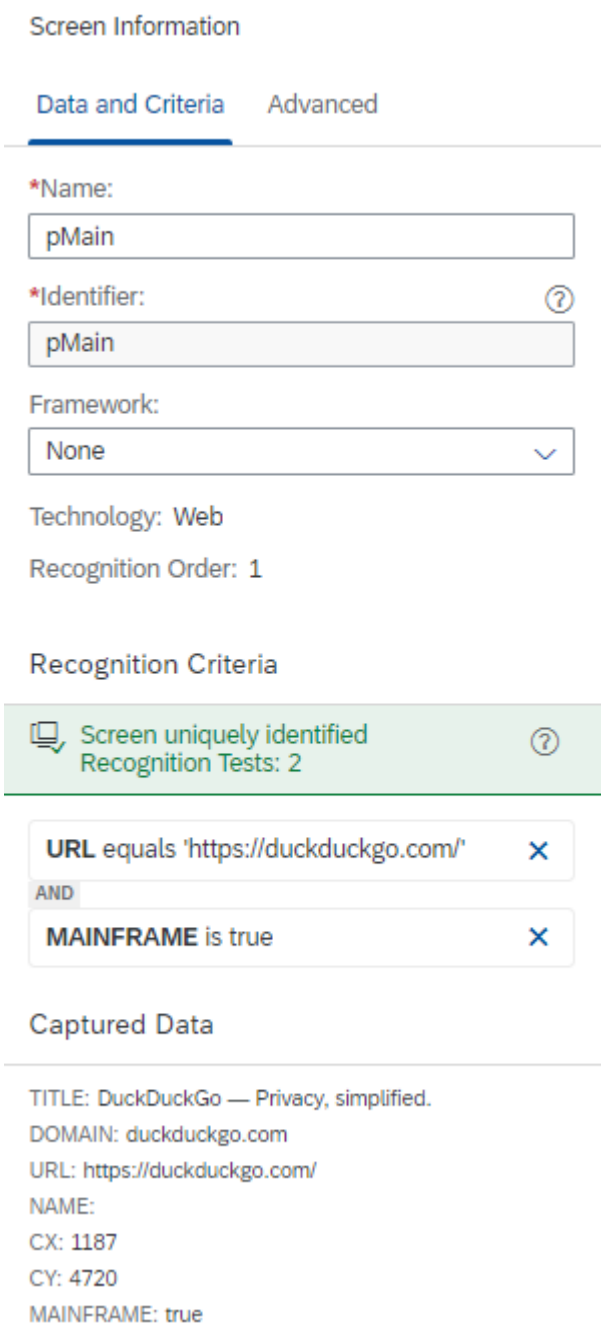


6. In the *Screen Information* panel, set the criteria of the screen. You can add the criteria by clicking on them in the *Captured Data* section.

   In the *Captured Data* section, check that the following criteria are automatically selected:

   - *MAINFRAME* is 'true'.
   - *URL* equals to 'https://duckduckgo.com/'.

> **i Note**
>
> You must select *MAINFRAME* is 'true' to interact with the elements you will declare on the screen.



7. Once you defined your criteria, click *Save*.

2.1.4 Declare Elements

You must declare the elements from the captured application that you want to use in your automation.

In this tutorial, you will declare two elements: the search bar and the search button.

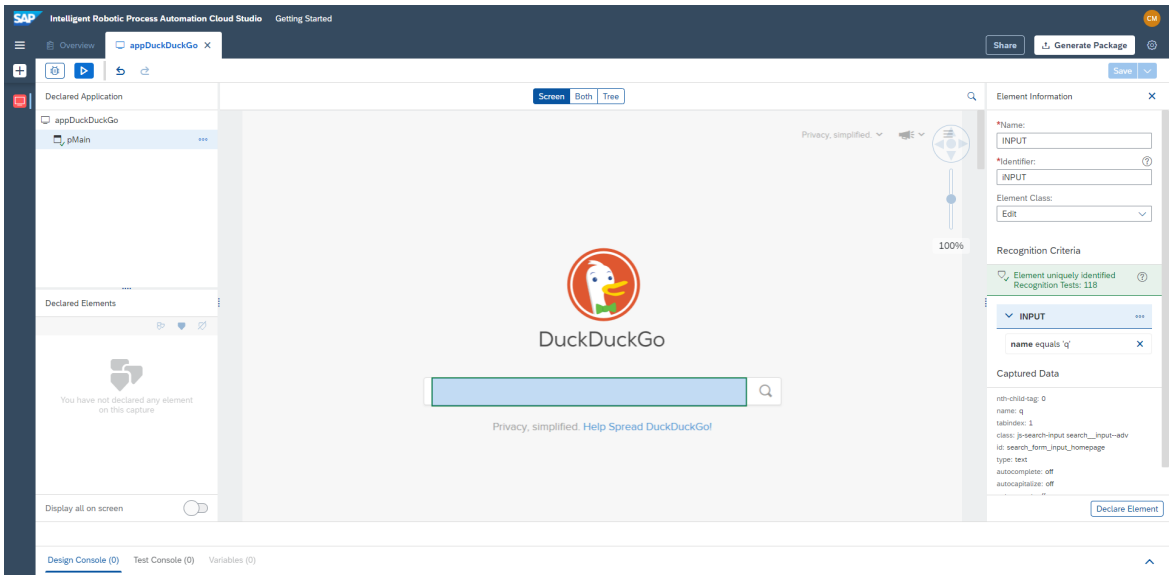
2.1.4.1 Declare the Search Bar

Context

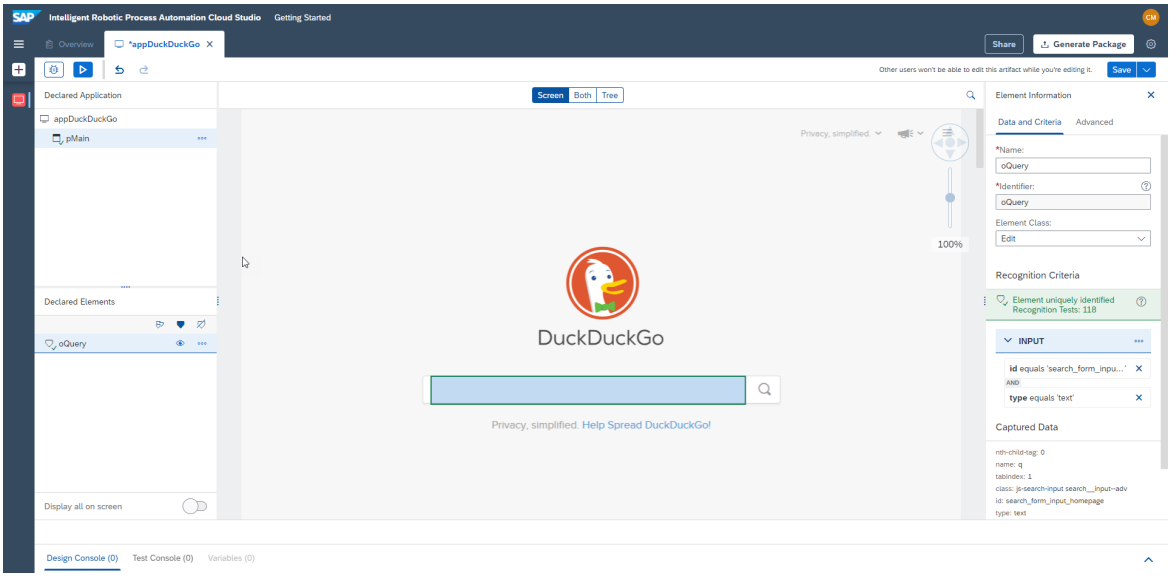
You will first declare the search bar.

Procedure

1. Click the search bar to select it.



2. In the *Element Information* side panel, enter a name for the element.
3. In *Criteria*, remove all criteria by clicking the ✕ icon and select:
   - *id* equals to 'search_form_input_homepage' and
   - *type* equals to 'text'
4. Click *Declare Element*.
   You declared the search bar.



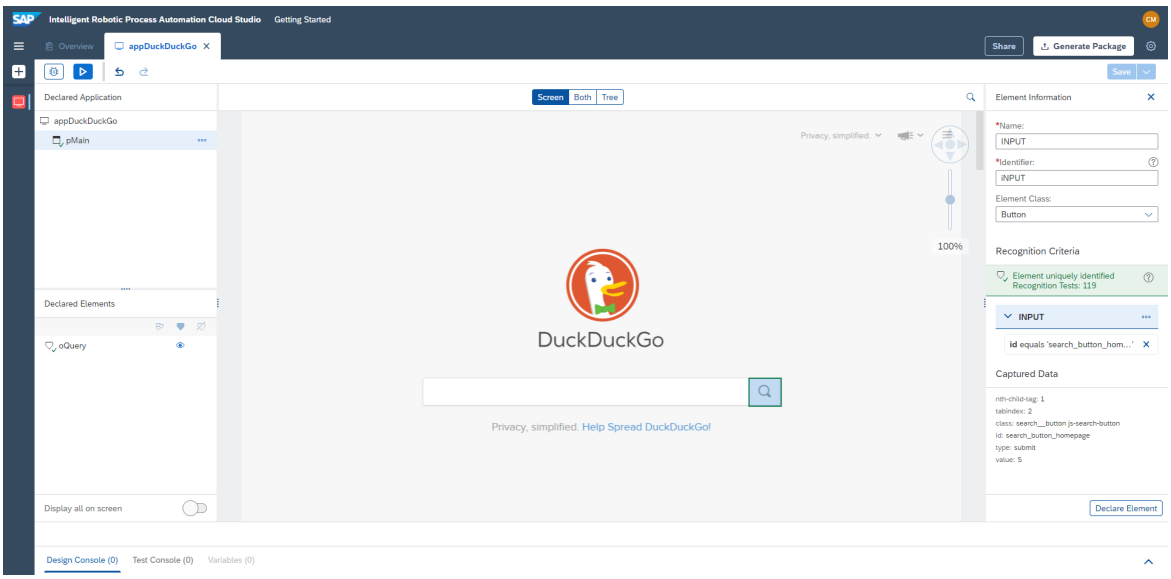
2.1.4.2 Declare the Search Button

Context

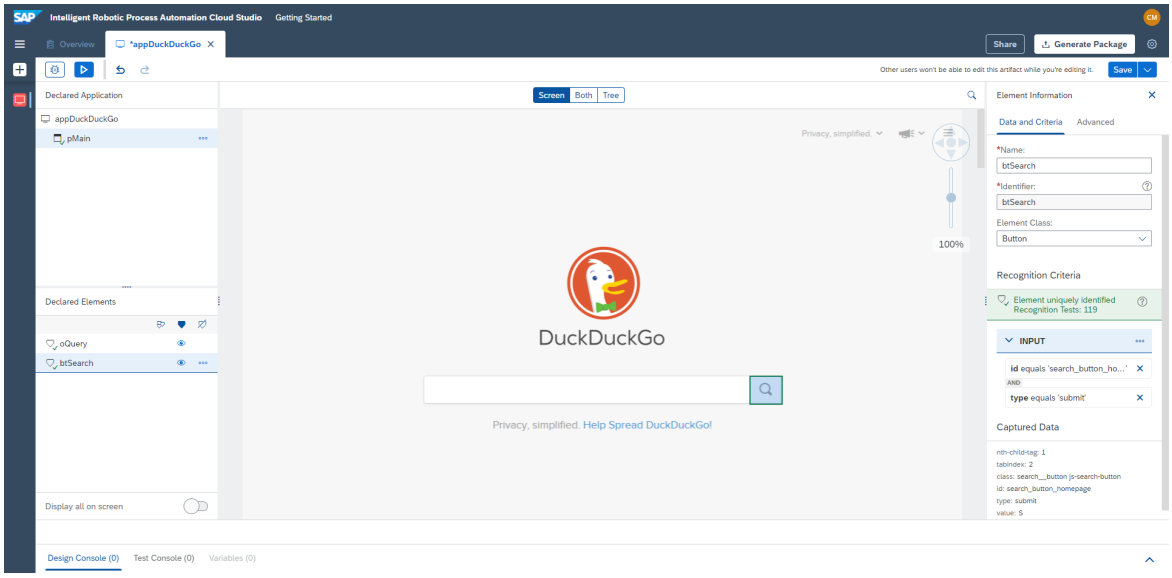
In this section, you will declare the search button.

Procedure

1. Click the search button to select it.



2. In the *Element Information* side panel, enter a name for the element.
3. In *Criteria*,remove all criteria by clicking the ✕ icon and select:
   - *id* equals to 'search_button _homepage' and
   - *type* equals to 'submit'
4. Click *Declare Element*.
   You declared the search button.



2.1.5 Capture the Search Results Screen

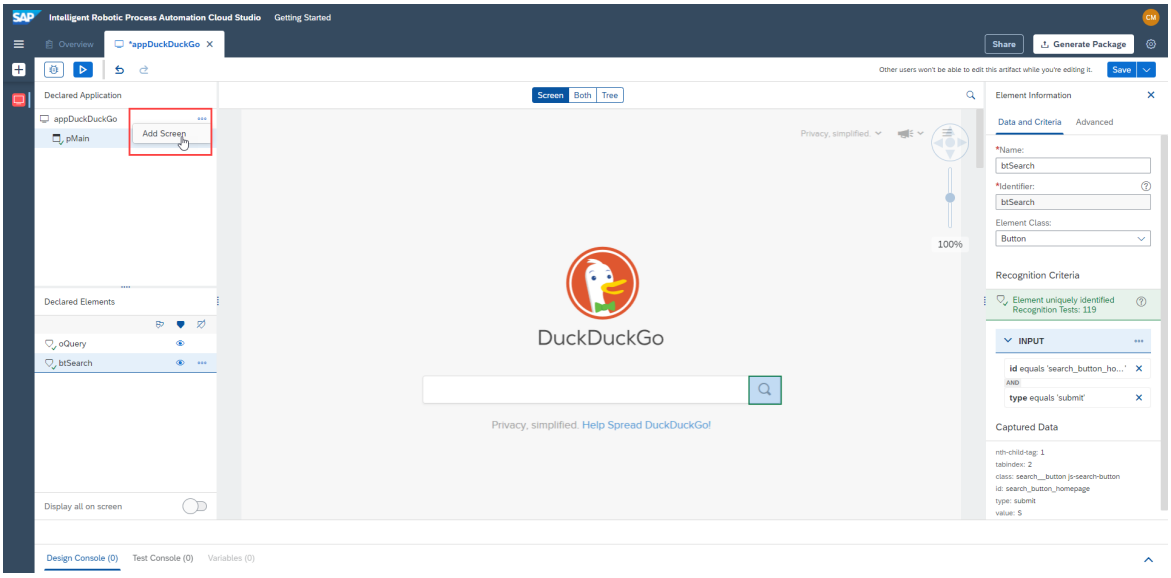
Context

You have captured an application and at least one screen. In this tutorial, you will capture another screen.

Procedure

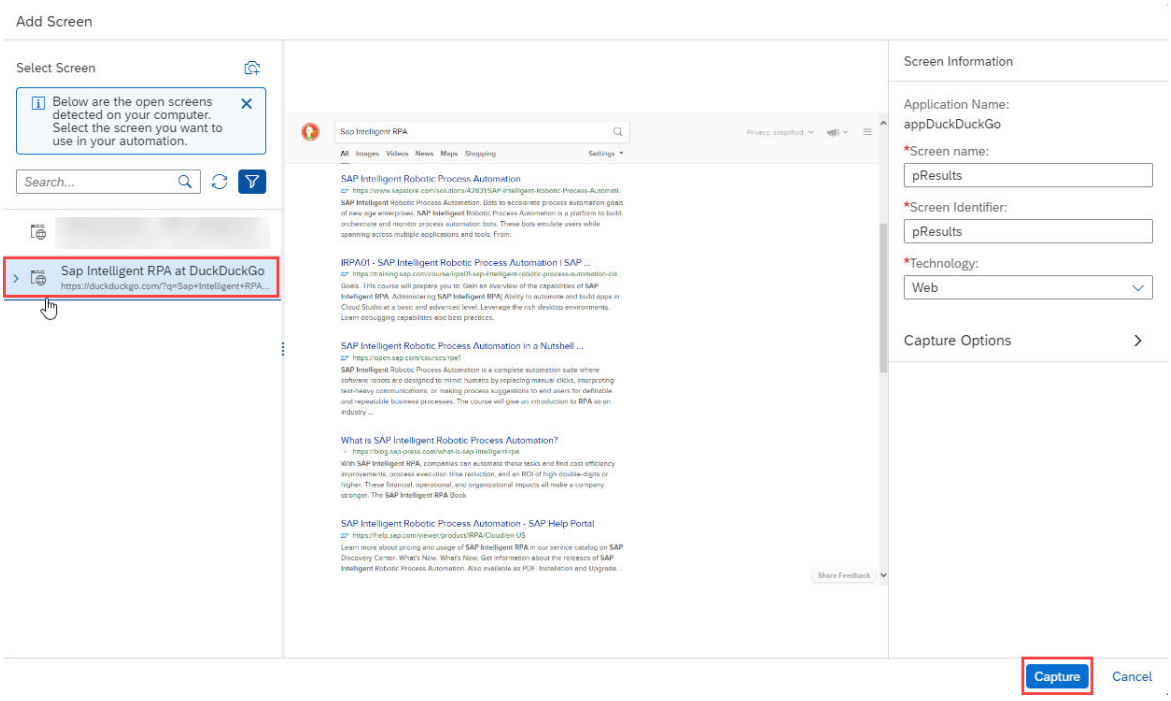
1. On the DuckDuckGo web page, enter `SAP Intelligent RPA` in the search bar and click the search button. The search results are displayed.

   In SAP Intelligent RPA Cloud Studio, go back to the DuckDuckGo screen you captured at the beginning of this tutorial.
2. On the *Declared Application* panel, click ••• next to the application your declared before to capture another screen and click *Add Screen*.



3. In *Select Screen*, select the DuckDuckGo screen you have opened from the list.

   A preview image of the screen is displayed in the capture area.



4. Enter a *Screen name* in the *Screen Information*.

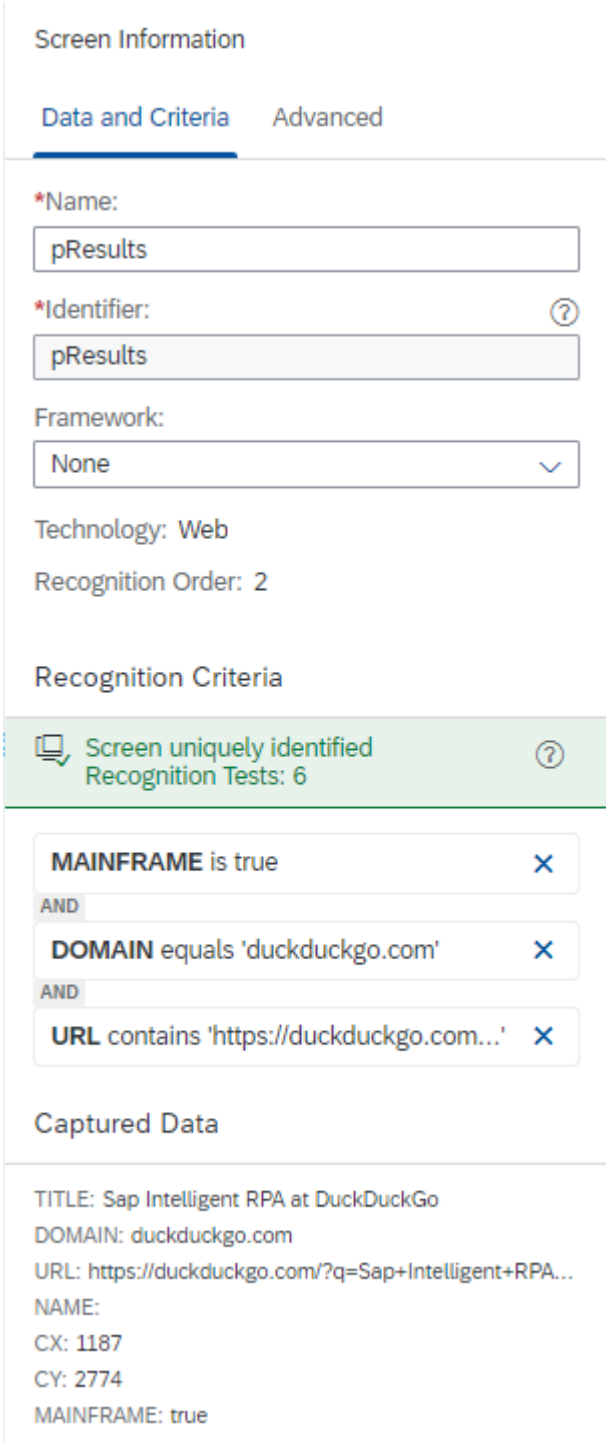
   We enter `pResults` as a name.

5. Click *Capture*.
6. In *Captured Data*, select:

   - *MAINFRAME* is 'true'
   - *DOMAIN* equals to 'duckduckgo.com'

> **Note**
>
> You must select *MAINFRAME* is 'true' to interact with the elements you will declare on the screen.

7. In *Criteria*, select the *URL* criterion. Change its properties by choosing *contains* instead of *equals to* and set the value as 'https://duckduckgo.com/?q='.



8. Once you defined your criteria, click *Save*.

Next Steps

Next, you need to declare several elements on the search results screen: titles, links and short descriptions.

2.1.5.1 Declare the Titles

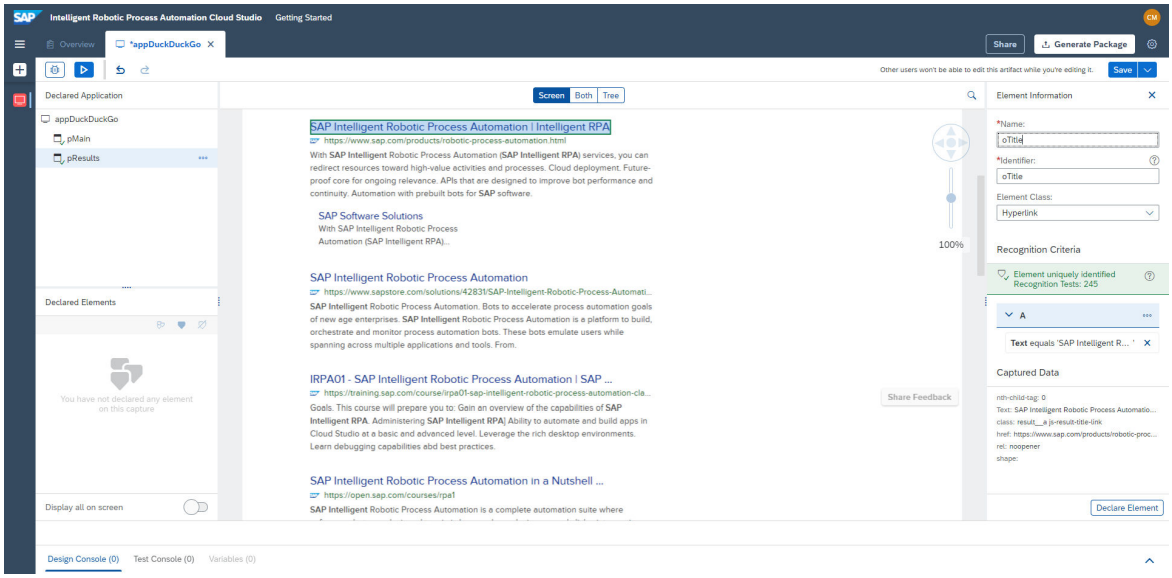
Context

You need to declare the titles on the results screen.

Procedure

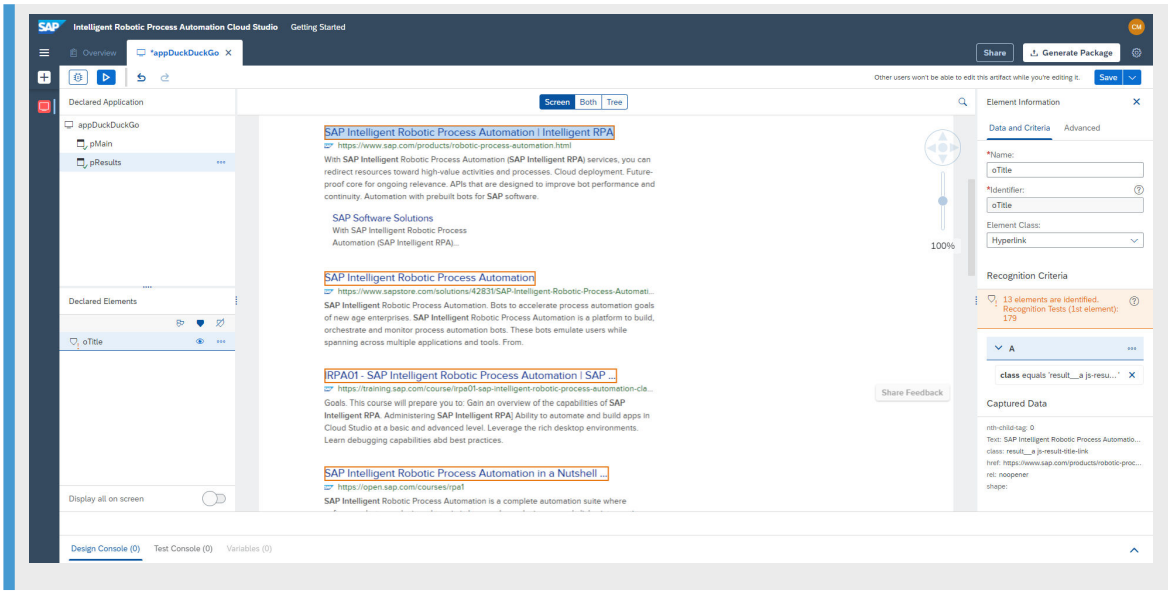
1. Click the first title to declare it.

   You can change its name. We choose to enter `oTitle` as a name.



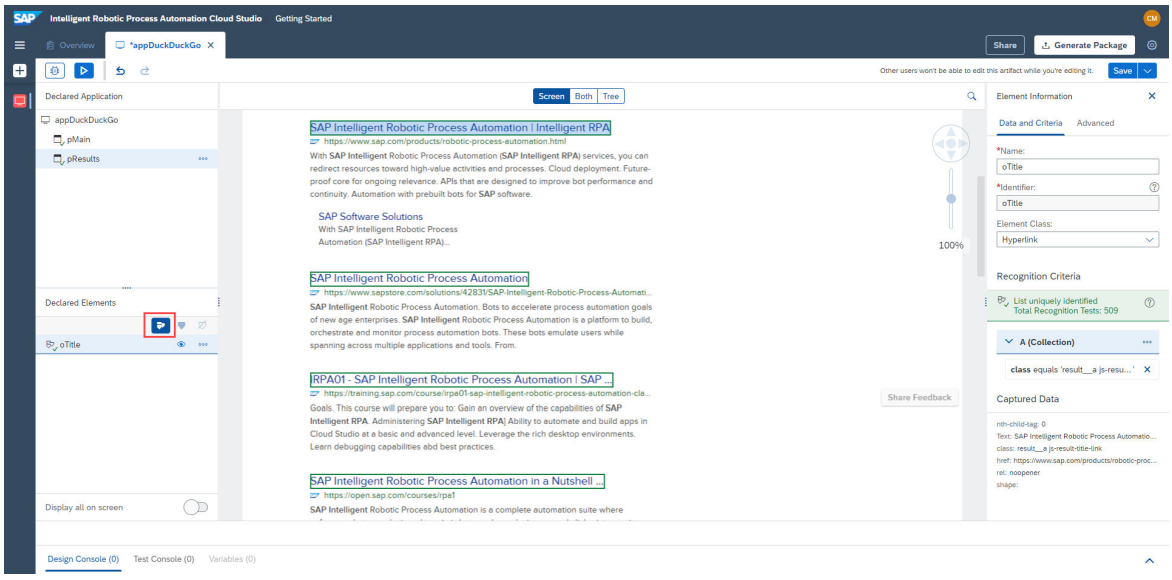
2. In *Captured Data*, click 'result__a js-result-title-link' to add it as a criterion. Click the cross ✕ to remove the criterion 'text equals to SAP Intelligent Robotic Process Automation | Intelligent RPA'.
3. Click *Declare Element*.

> **i Note**
>
> You have declared the first title. The declared element is circled orange until it is declared as a collection.



4. Under *Declared Elements* on the left-hand side panel, select the name of the element you declared and then click the *Is a collection* button.

> **Note**
>
> All elements on the screen that have the same criteria are added as a collection. A green border is shown around them.



5. Click *Save*.

2.1.5.2 Declare the Links

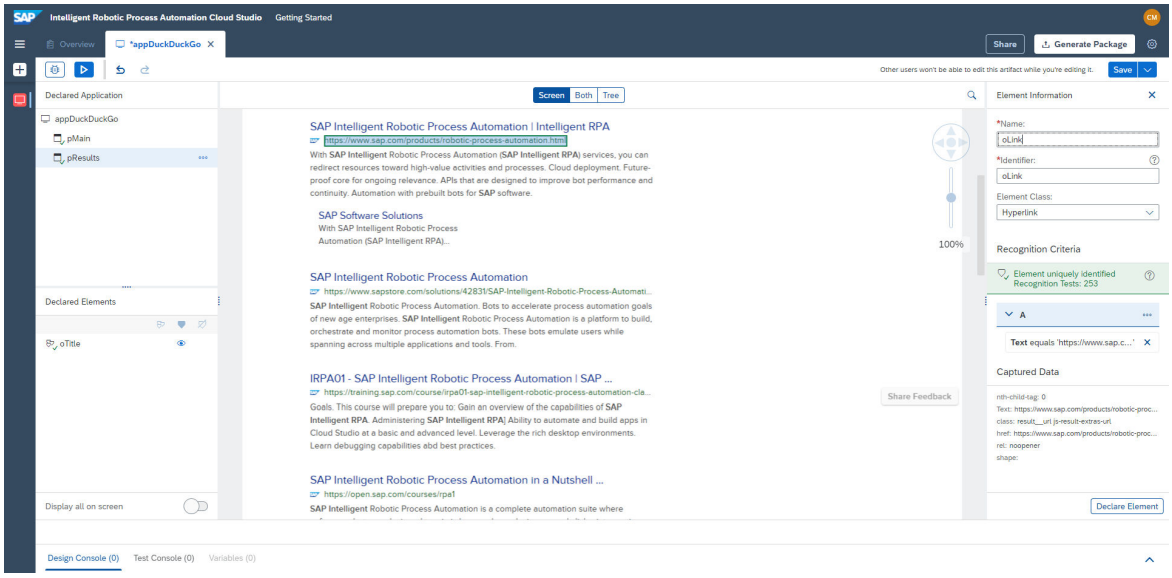
Context

You need to declare the links on the results screen.

Procedure

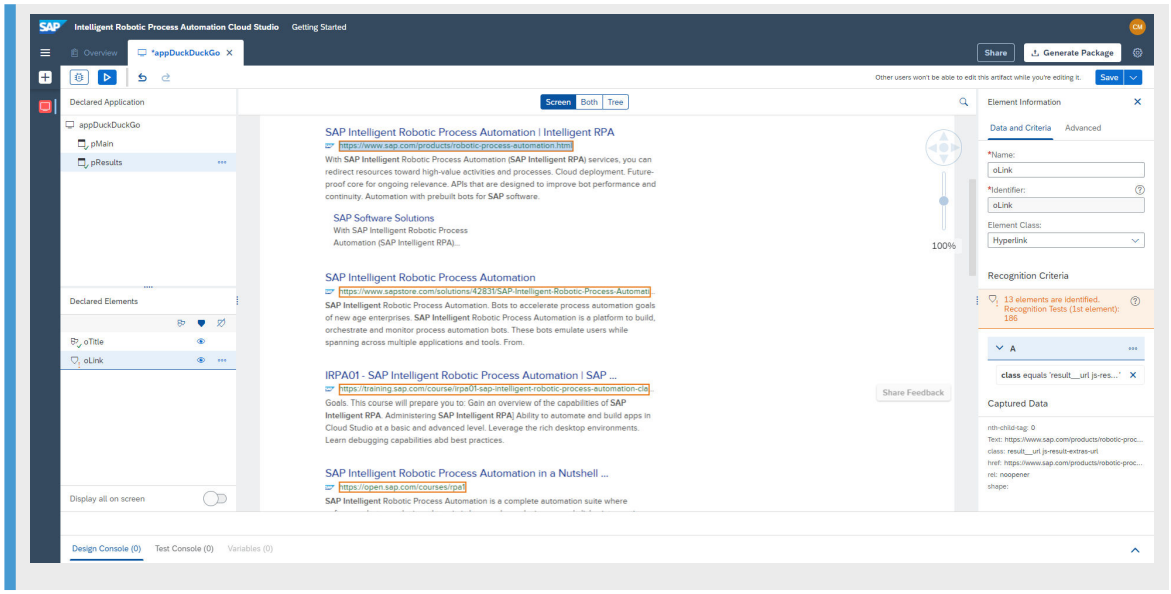
1. Click the first link to declare it.

   You can change its name. We choose to enter `oLink` as a name.



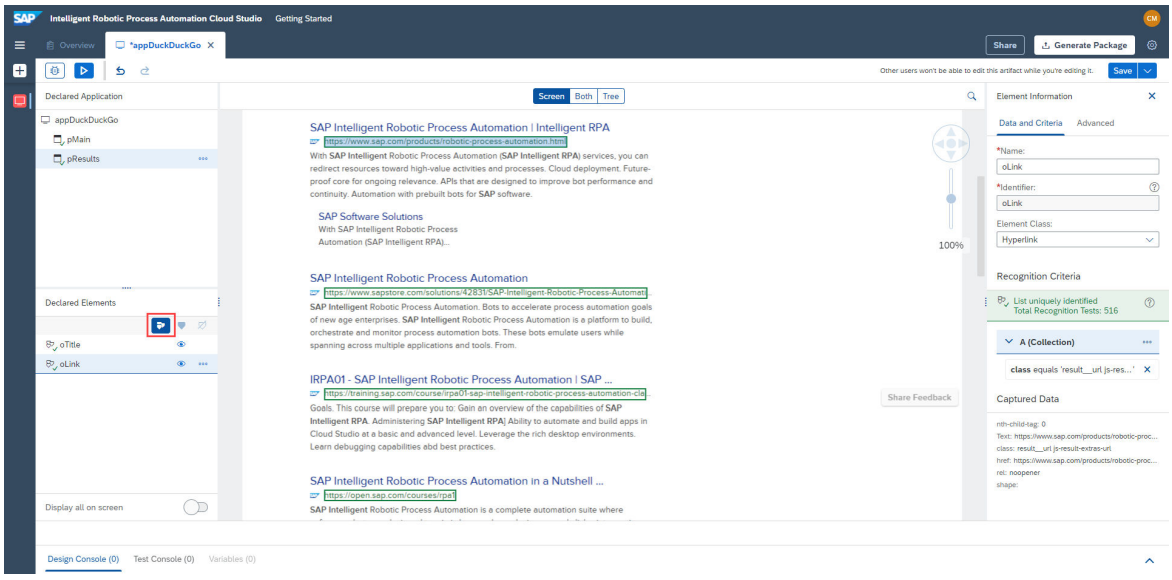
2. In *Captured Data*, click 'class: result__url js-result-extras-url' to add it as a criterion. Click the cross ✕ to remove the criterion 'text equals to https://www.sap.com/products/robotic-process-automation.html'.
3. Click *Declare Element*.

> **i Note**
>
> You have declared the first link. The declared element is circled orange until it is declared as a collection.



4. Under *Declared Elements* on the left-hand side panel, select the name of the element you declared and then click the *Is a collection* button.

> **Note**
>
> All elements on the screen that have the same criteria are added as a collection. A green border is shown around them.



5. Click *Save*.

2.1.5.3 Declare the Short Descriptions

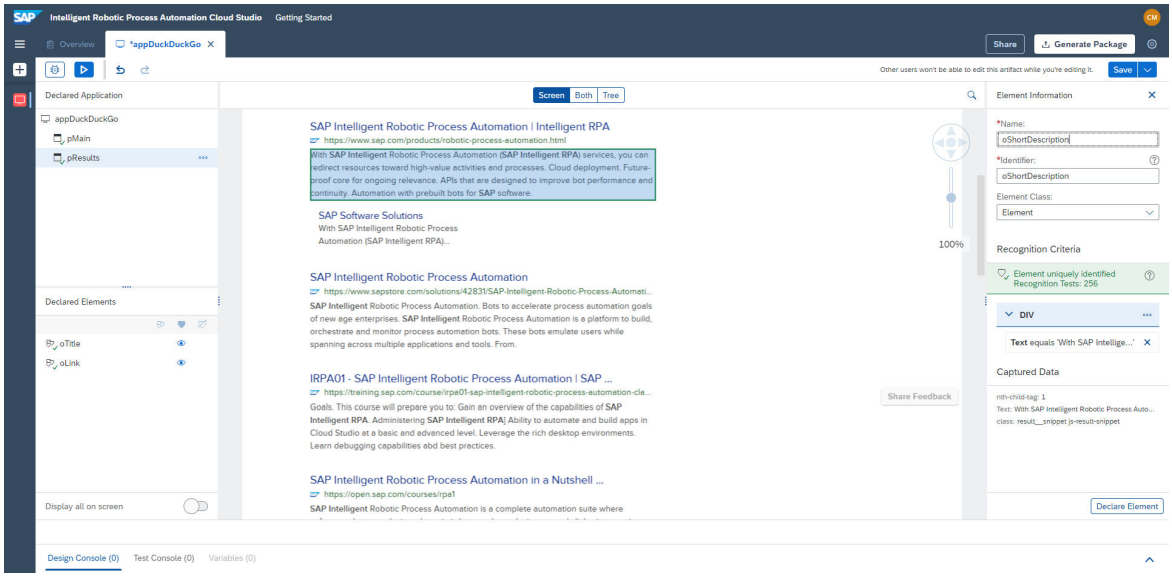
Context

You need to declare the short descriptions on the results screen.

Procedure

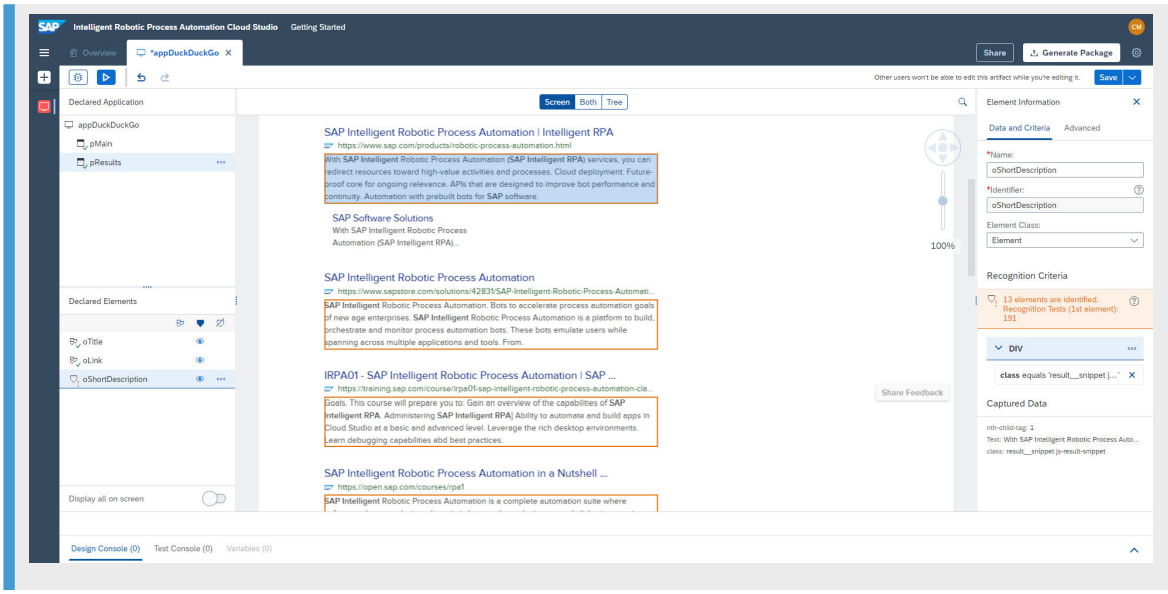
1. Click the first short description to declare it.

   You can change its name. We choose to enter `oShortDescription` as a name.



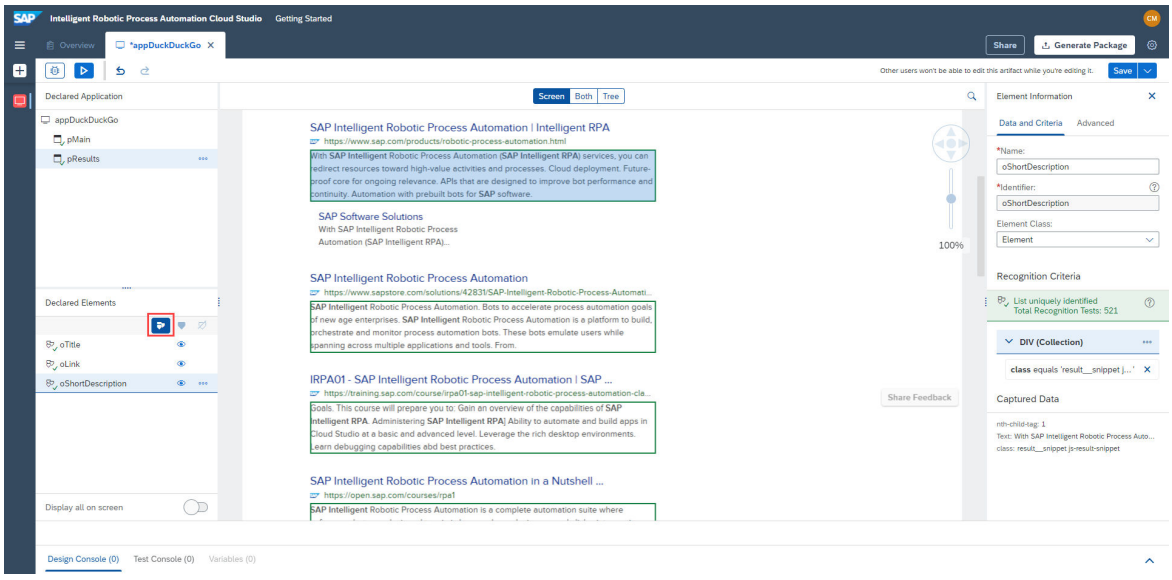
2. In *Captured Data*, click 'class: result__snippet js-result-snippet' to add it as a criterion. Click the cross ✕ to remove the criterion 'text equals to With SAP Intelligent Robotic Process Automation (SAP Intelligent RPA) ...'
3. Click *Declare Element*.

> **i** Note
>
> You have declared the first short description. The declared element is circled orange until it is declared as a collection.



4. Under *Declared Elements* on the left-hand side panel, select the name of the element you declared and then click the *Is a collection* button.

> **Note**
>
> All elements on the screen that have the same criteria are added as a collection. A green border is shown around them.



5. Click *Save*.

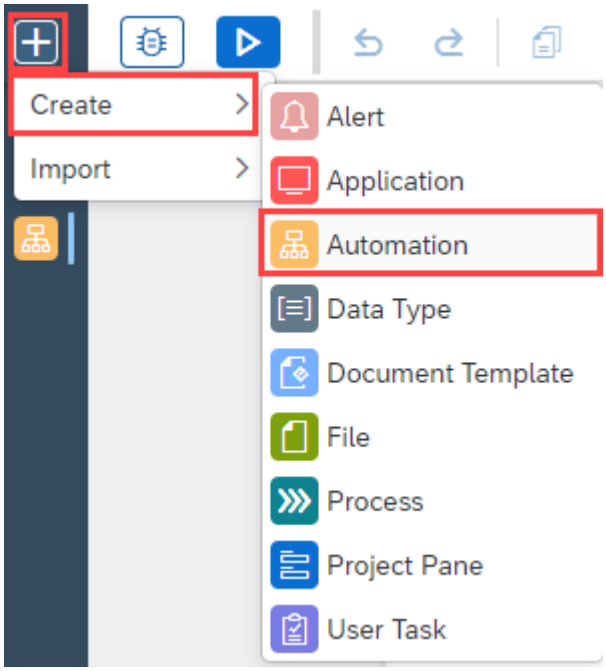
2.1.6 Create your Automation

Context

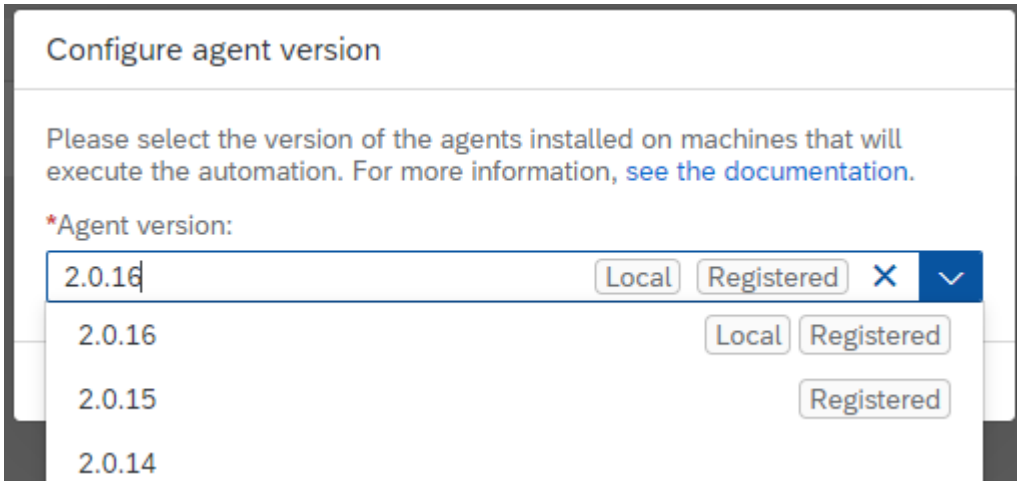
You captured and declared different screens and elements. You will now create an automation.

Procedure

1. Click the + icon and then the *Create* button in the Project Explorer left-hand side panel and select *Automation*.

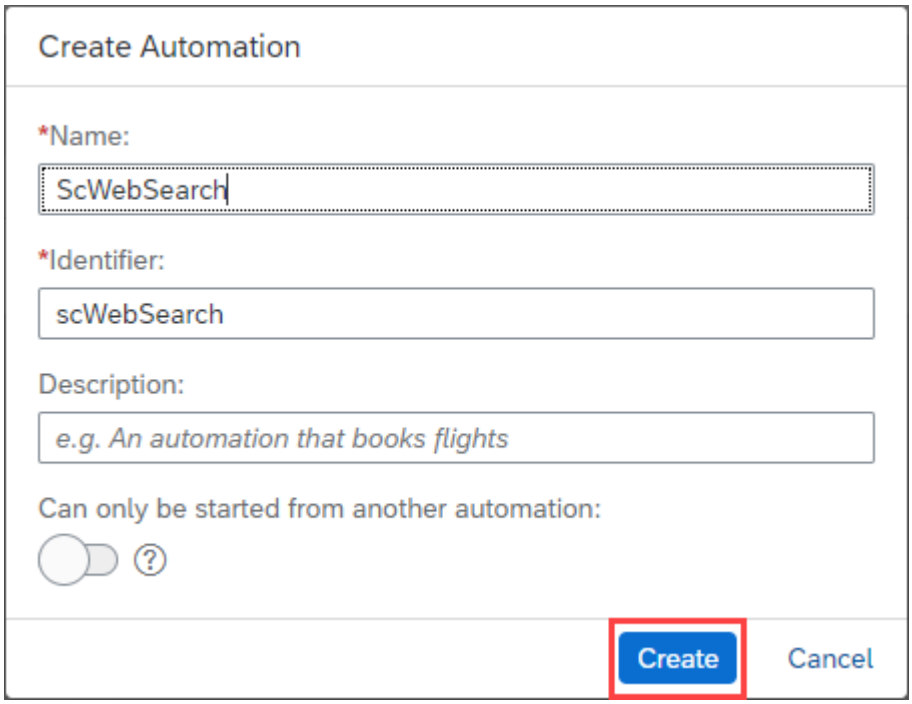


2. Configure the version of the agent that will execute the automation.



For more information, see: Configure an Agent Version [page 513]

3. Enter a name in the *Name* field and click *Create*.



Next Steps

You can now drag and drop activities to build your automation in the Cloud Studio.

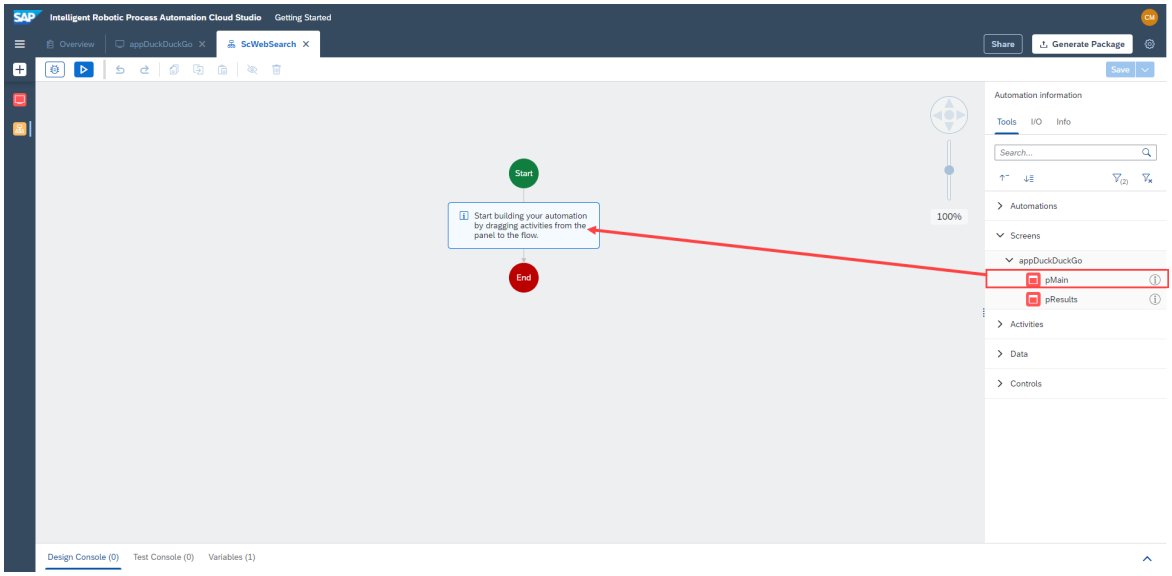
2.1.6.1 Set the Search Engine Screen Activities

Context

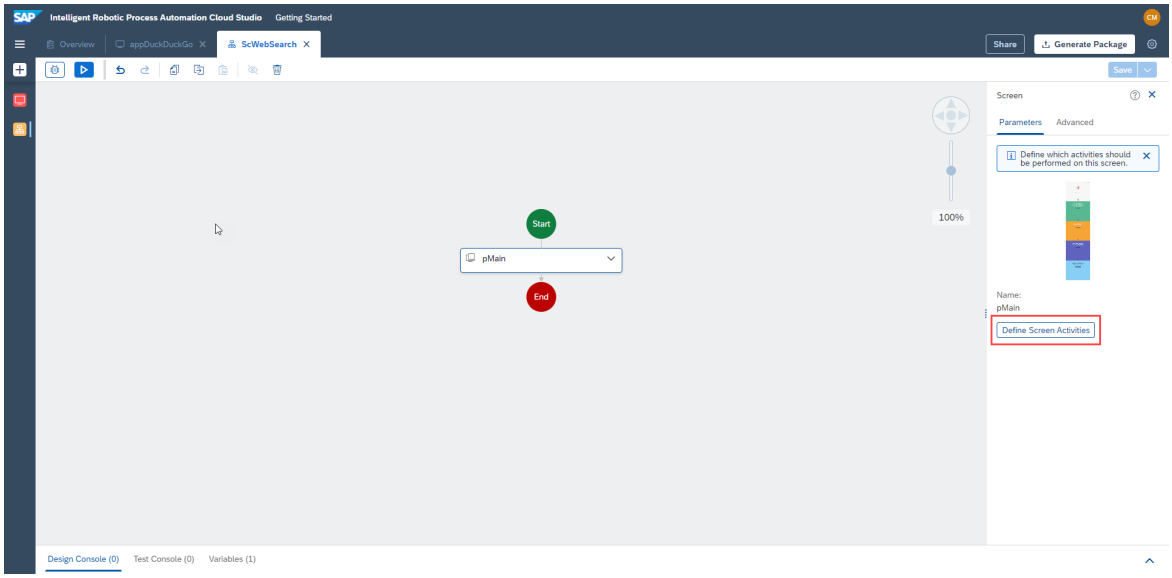
You have created an automation. You will first set activities of the search engine screen that we named `pMain` in Capture the Search Engine [page 7].

Procedure

1. In *Automation information*, open the *Screens* list in the dropdown menu on the right-hand side panel. Drag and drop the screen `pMain` into the workflow.



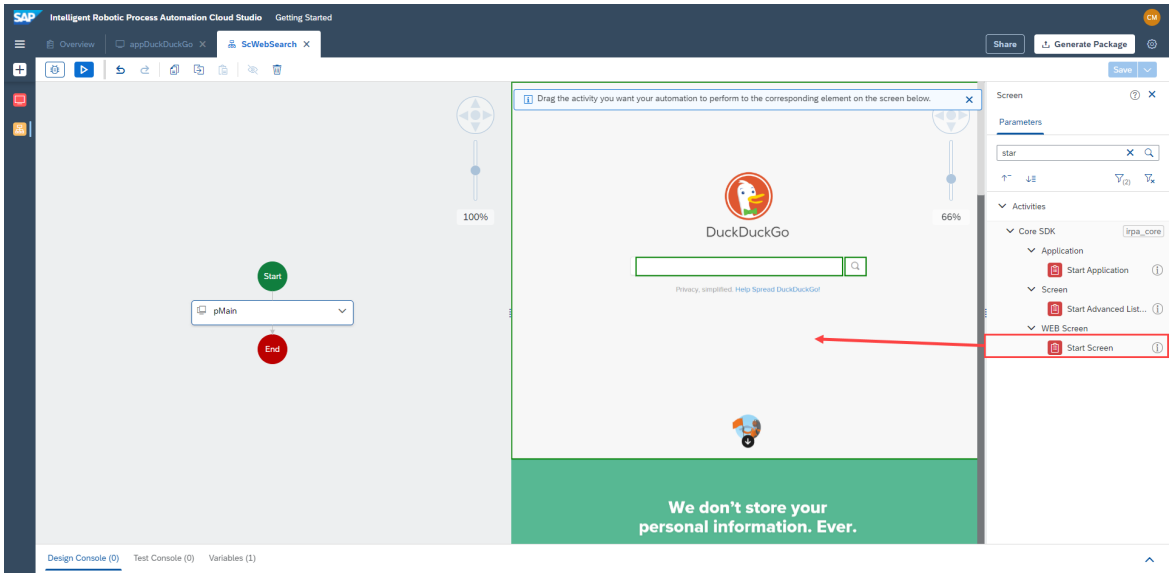
2. Select `pMain` in the workflow and click the *Define Screen Activities* button on the *Screen* panel.



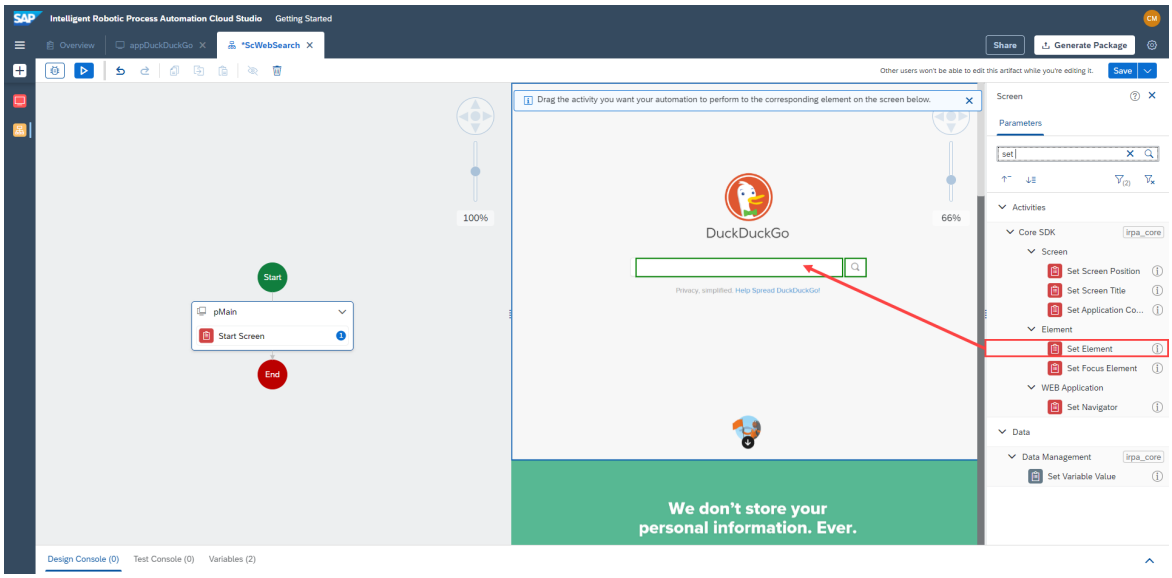
3. Enter `Start` in the search bar of the *Screen* panel and choose the *Start Screen* activity, in the *WEB Screen* submenu.

   Drag and drop the *Start Screen* activity anywhere on the screen preview in the main panel.

> **i Note**
>
> Do not drag and drop the activity on the elements you declared before.



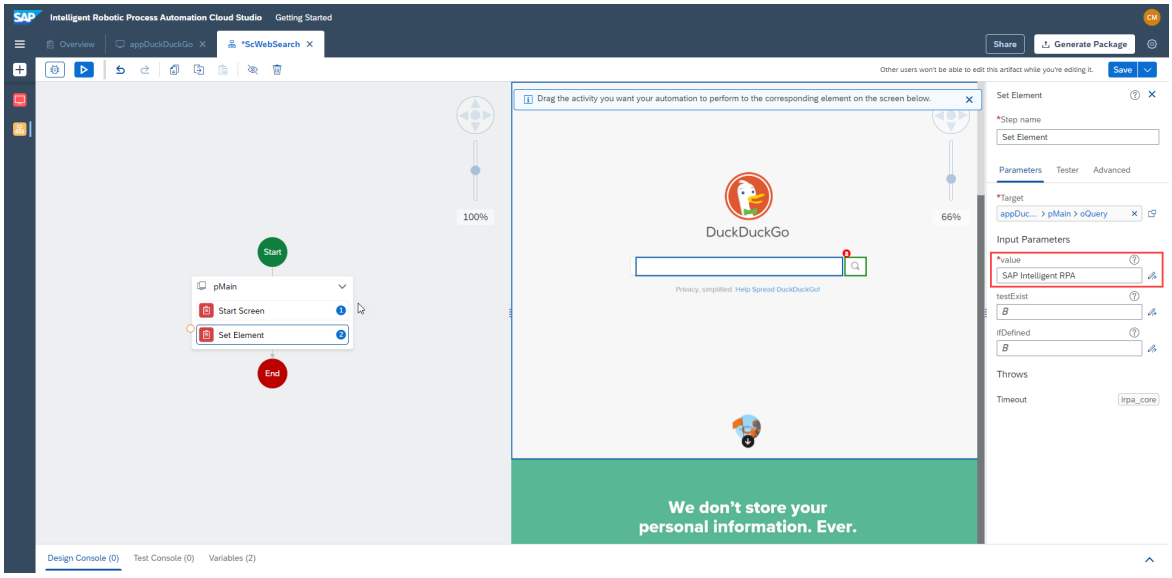
4. Save your automation.
5. Enter `Set` in the search bar of the *Screen* panel and choose the *Set Element* activity, in the *Element* submenu. Drag and drop the activity onto the search bar.



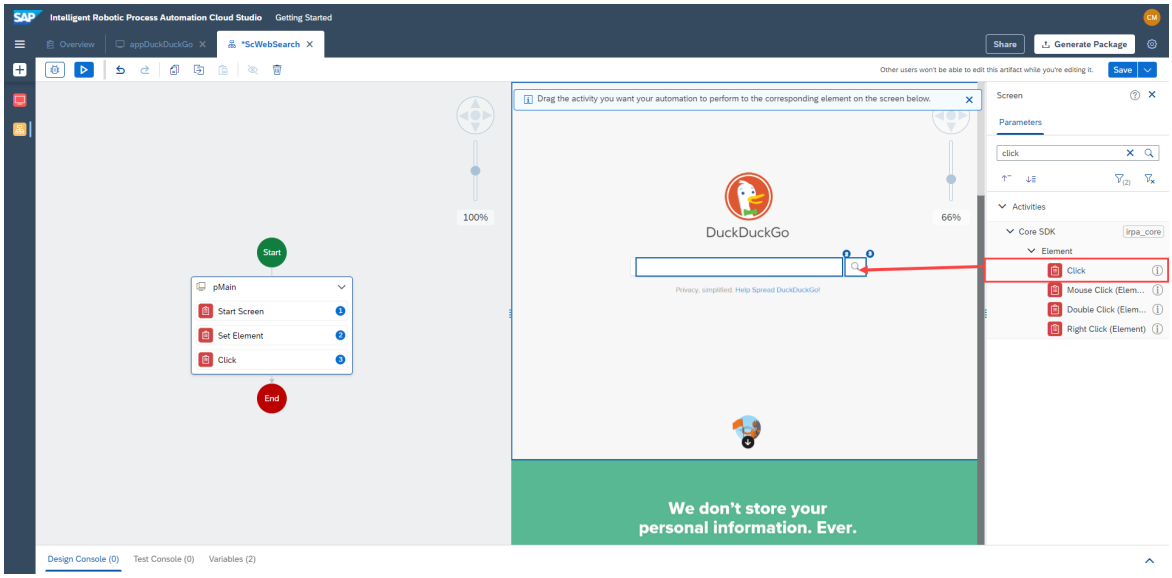
6. Select the *Set Element* activity in the workflow and enter `SAP Intelligent RPA` as *value* in *Input Parameters*.

   Click *Save*.

> **i Note**
>
> When you add the activity to your automation workflow, you can notice that it is labeled with a number. It means that the activity is linked to the search bar on the captured screen.



7. Click `pMain` on your workflow to reopen the side panel.
8. Drag and drop the *Click* activity from the *Element* submenu onto the search button.



> **i Note**
>
> You can notice that the activity is labeled with a number.

9. Save your automation.

Next Steps

You need to set the activities of the search results screen.

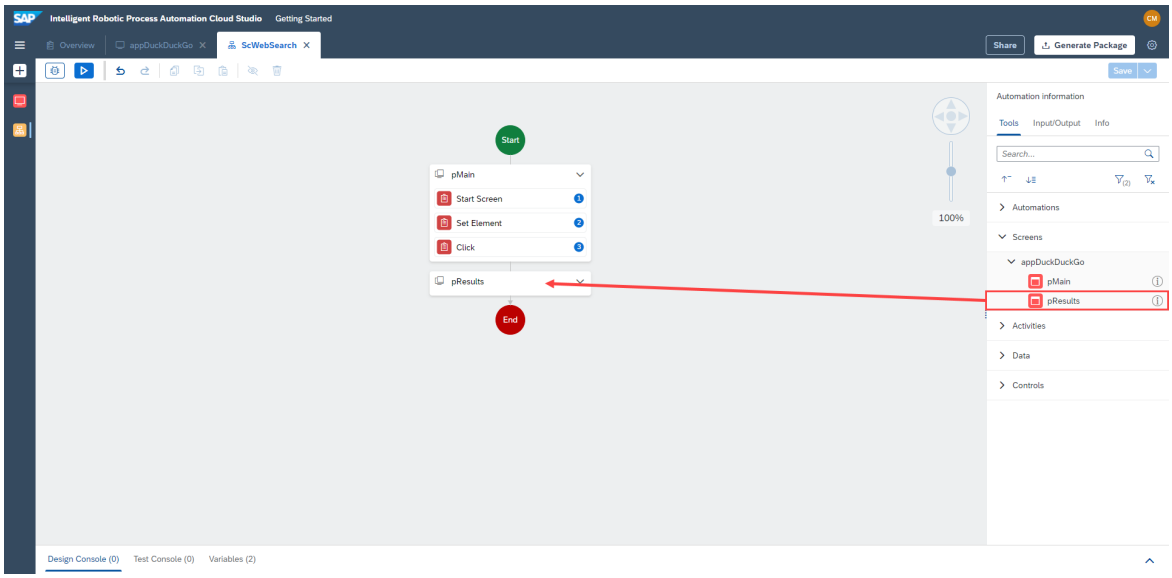
2.1.6.2 Set the Search Results Screen Activities

Context

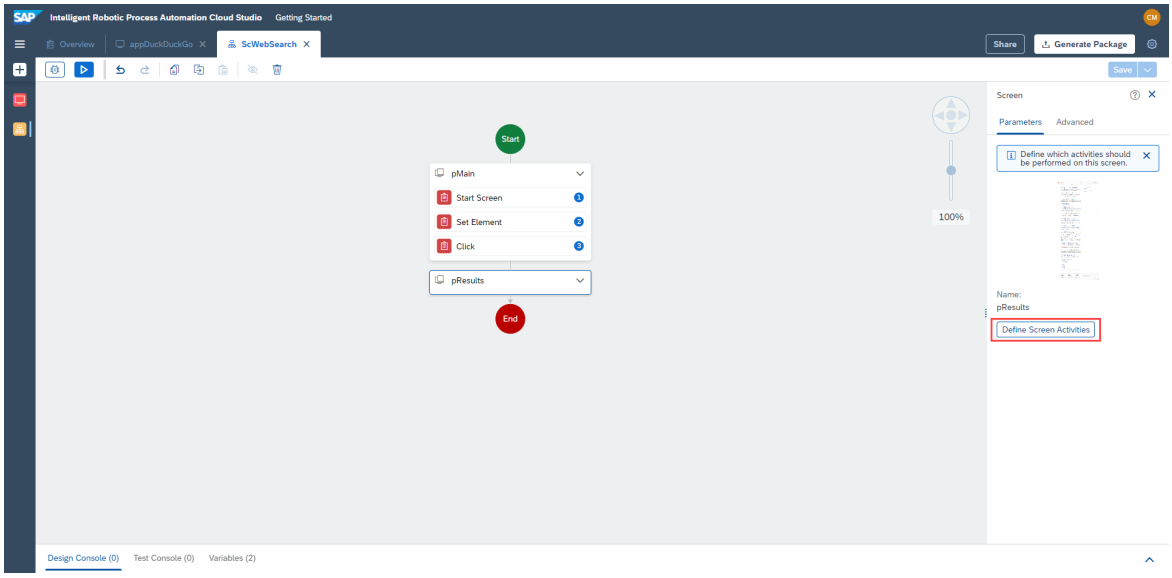
You have created an automation. You have set activities of the search engine screen. You will now set the activities of the search results screen that we named `pResults` in Capture the Search Results Screen [page 14].

Procedure

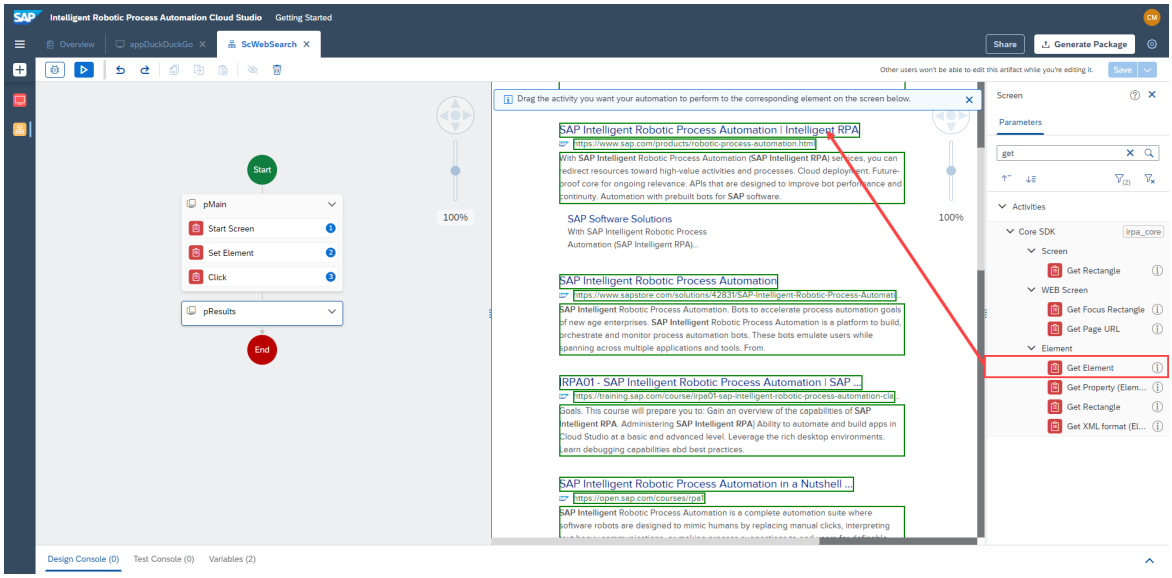
1. In *Automation information*, open the *Screens* list in the dropdown menu on the right panel. Drag and drop the screen `pResults` into the workflow and after `pMain`.



2. Select `pResults` in the workflow and click the *Define Screen Activities* button in the *Screen* panel.



3. Enter `Get` in the search bar at the top of the *Screen* panel. Drag and drop the *Get Element* activity from the *Element* submenu onto the first result title.



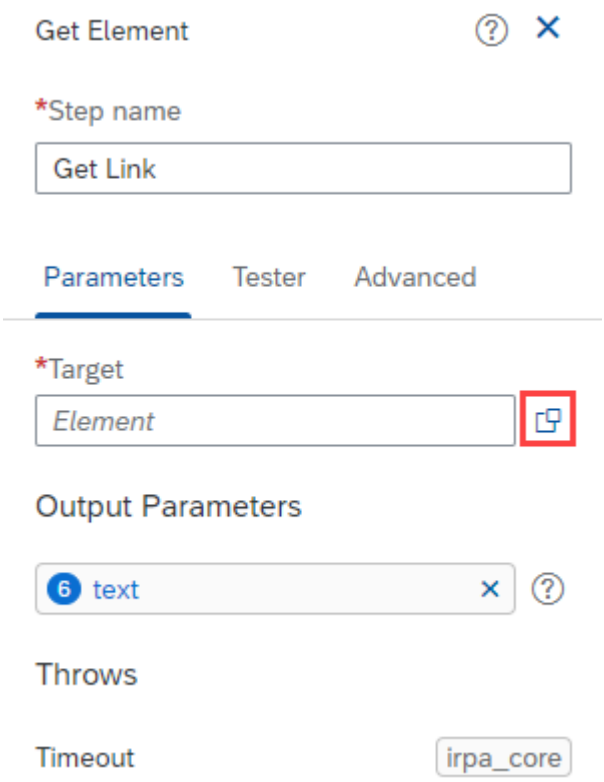
Select *Get Element* in the workflow and rename it in the *Step Details* panel. We name it `Get Title`.

> **Note**
>
> After adding the activity into `pResults`, you can see that there is a *For each* block in the workflow. It means that the *Get Element* activity will be repeated for each result title of the list.

4. Drag and drop the *Get Element* activity into the *For each* block after the `Get Title` activity.

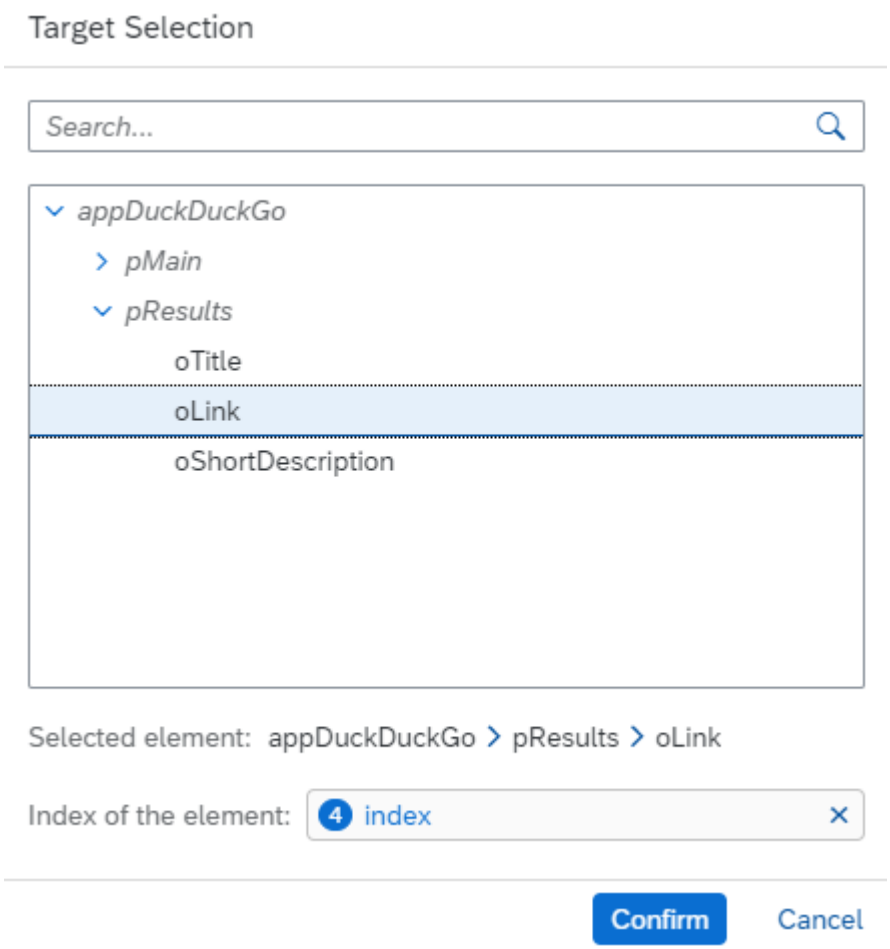
   You can select the activity you have added in the workflow and rename it on the right-hand side panel. We name it `Get Link`.

5. In the *Step Details* panel, click the icon to open the *Target Editor*.



You can know set the targeted element for this activity.

6. Expand:
   - *appDuckDuckGo* (or the name you have given to your application in Capture the Search Engine [page 7]).
   - *pResults* (or the name you have given to the search results screen in Capture the Search Results Screen [page 14]).
   - and select *oLink* (or the name you have given to the search results links in Declare the Links [page 19]).



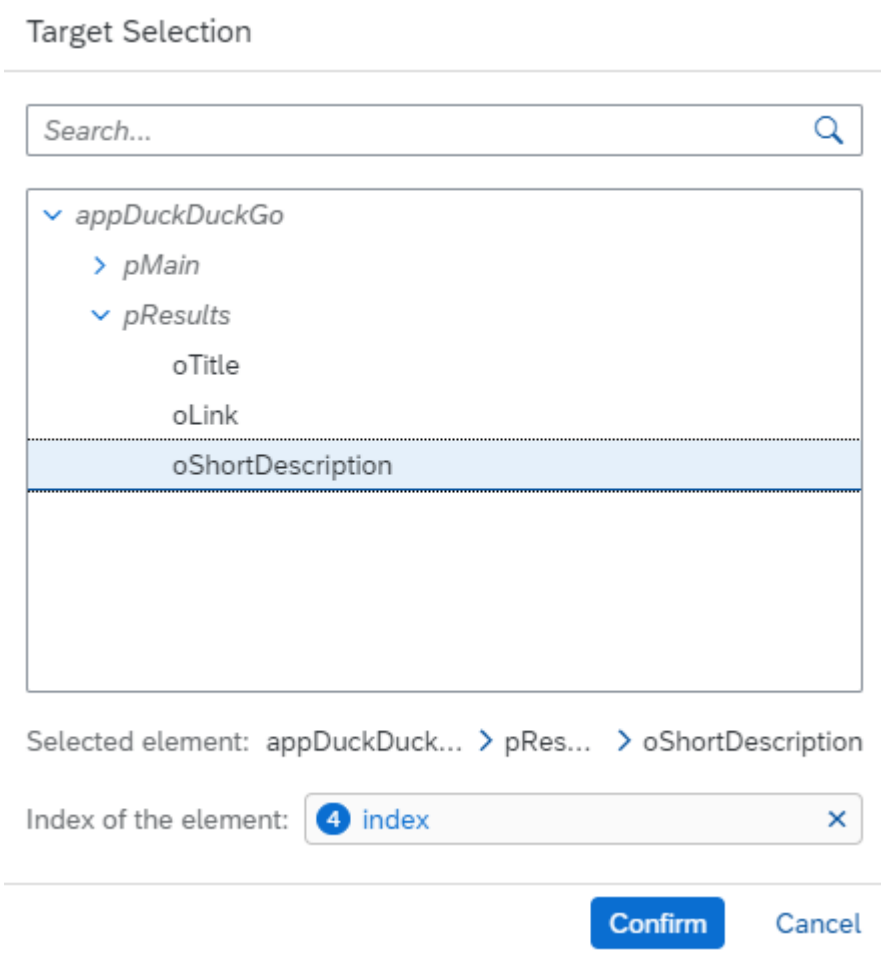
Select *4 Index* as the *Index of the element* and click *Confirm*.

7. Drag and drop the *Get Element* activity into the *For each* block after the *Get Link* activity.

   Select the activity you have added in the workflow. Rename it in the right-hand side panel. We name it `Get Short Description`.

8. In the *Step Details* panel, click the icon to open the *Target Editor*, expand:

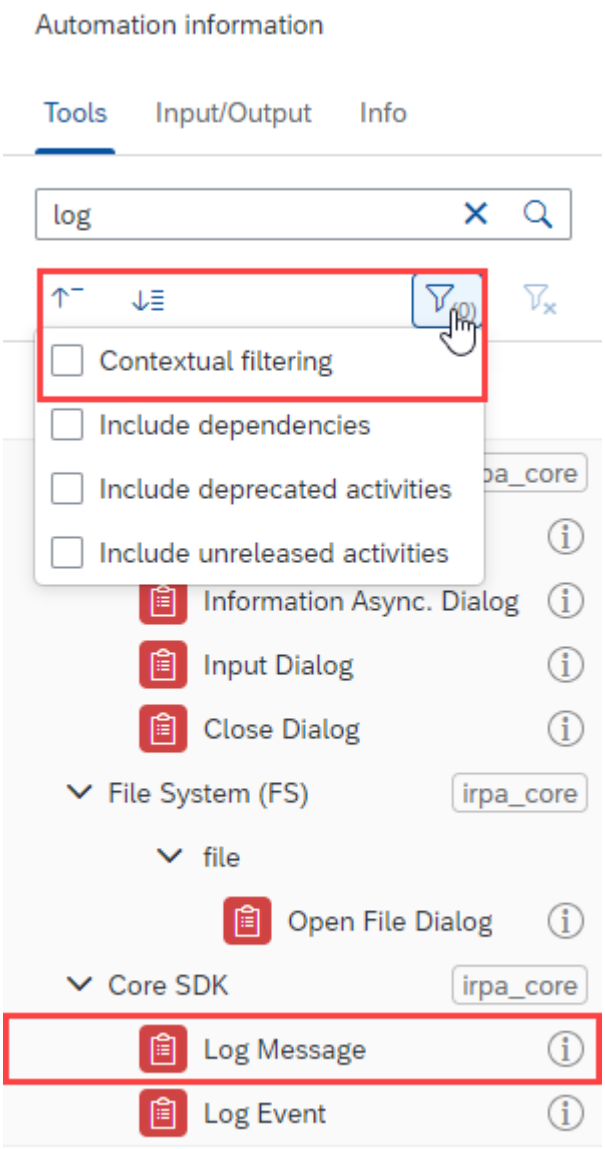
   - *appDuckDuckGo* (or the name you have given to your application in Capture the Search Engine [page 7]).
   - *pResults* (or the name you have given to the search results screen in Capture the Search Results Screen [page 14]).
   - and select *oShortDescription* (or the name you have given to the search results links in Declare the Short Descriptions [page 21]).



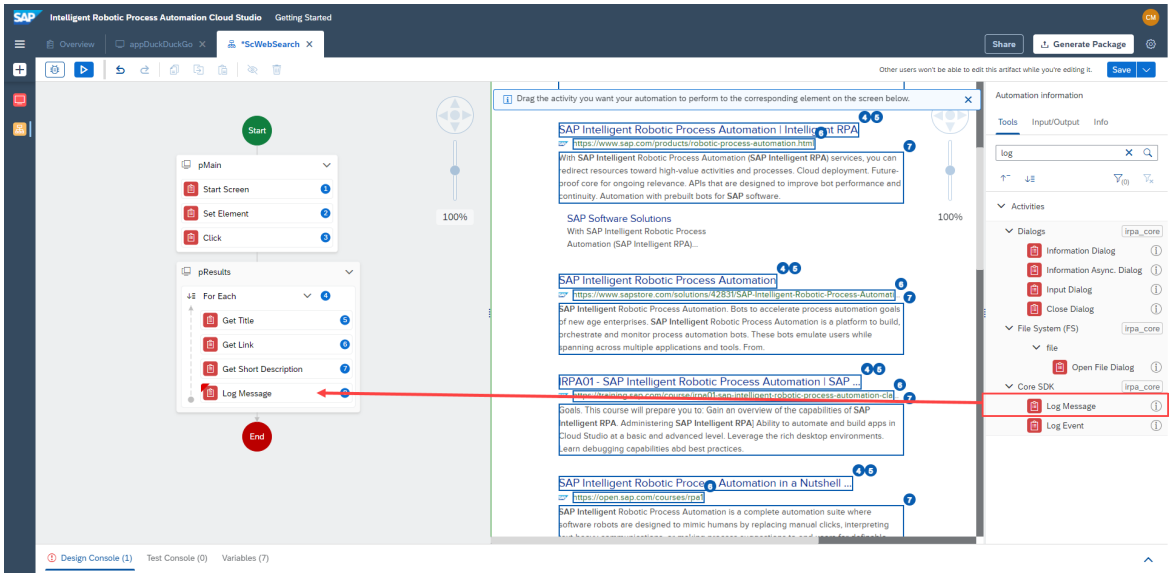
Select *4 Index* as the *Index of the element* and click *Confirm*.

9. Click *pResults* in the workflow. In the search bar of the *Screen* panel, enter Log.

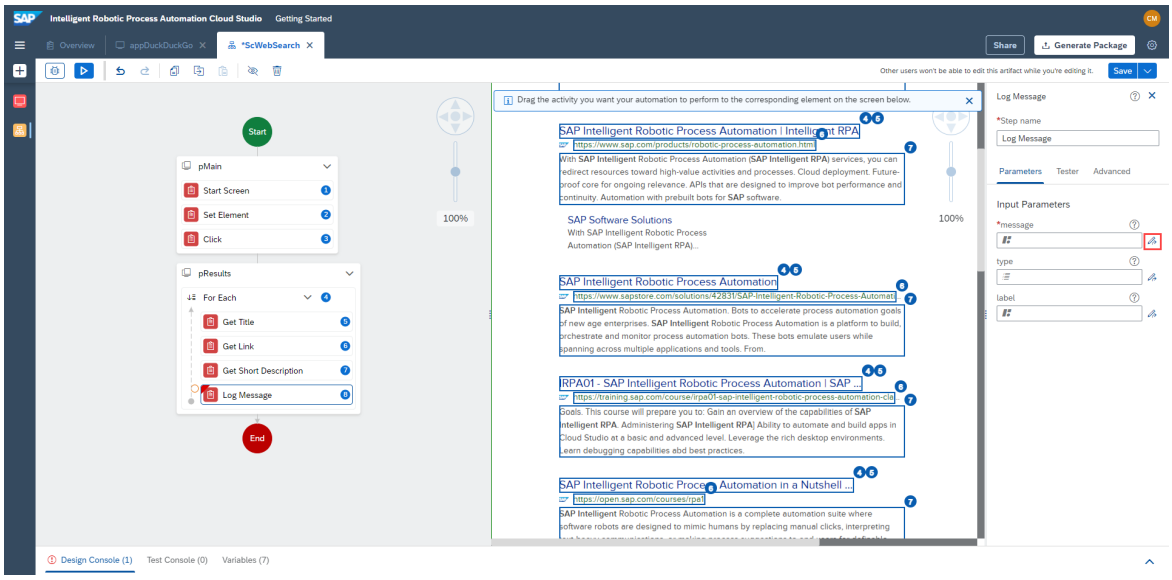
> **i Note**
>
> You will not find the *Log Message* activity because this activity cannot be added to a screen.
>
> To add a *Log Message* activity in the *For each* block, you must click ▽ and unselect *Contextual filtering*.



You can now drag and drop the *Log Message* activity into the workflow, under the *Get Short Description* activity.

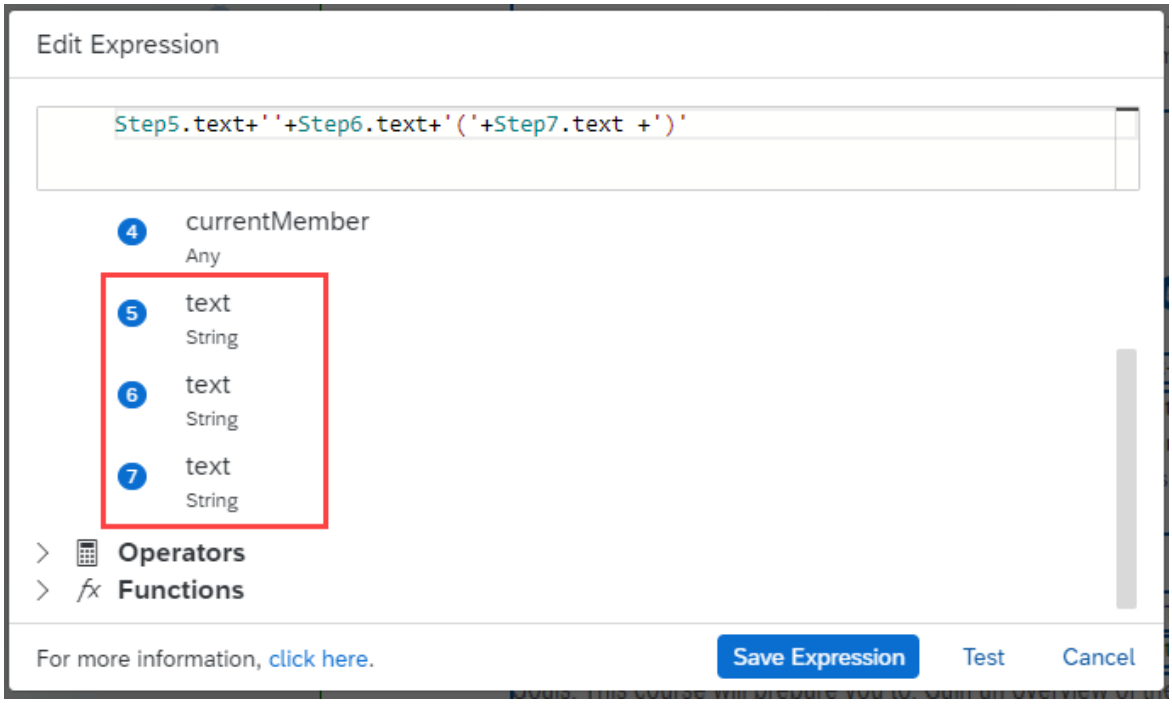


10. Select the *Log Message* activity to open the side panel and click next to the *message* field to open the *Expression Editor*.

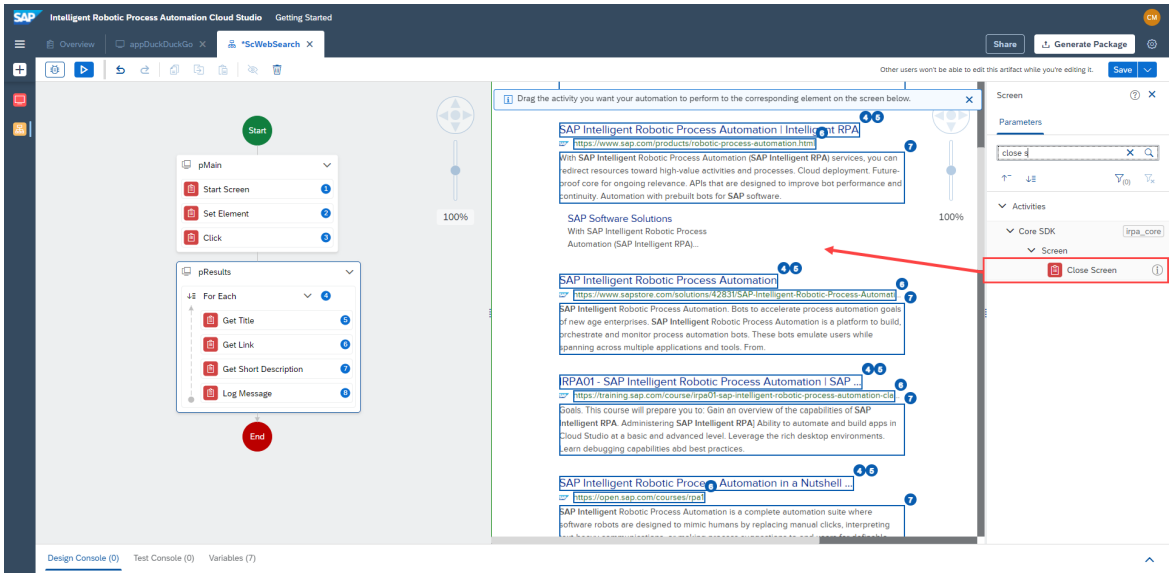


11. Enter `Step5.text+''+Step6.text+'('+Step7.text +')'` in the *Expression Editor* and save your expression.

> **i Note**
>
> You can expand the *Variables* section to select the steps.



12. In the workflow of your automation, double-click on *pResults* to open the *Screen* panel. Enter `Close` in the search bar and drag and drop the *Close Screen* activity from the *Screen* submenu anywhere on the screen preview.

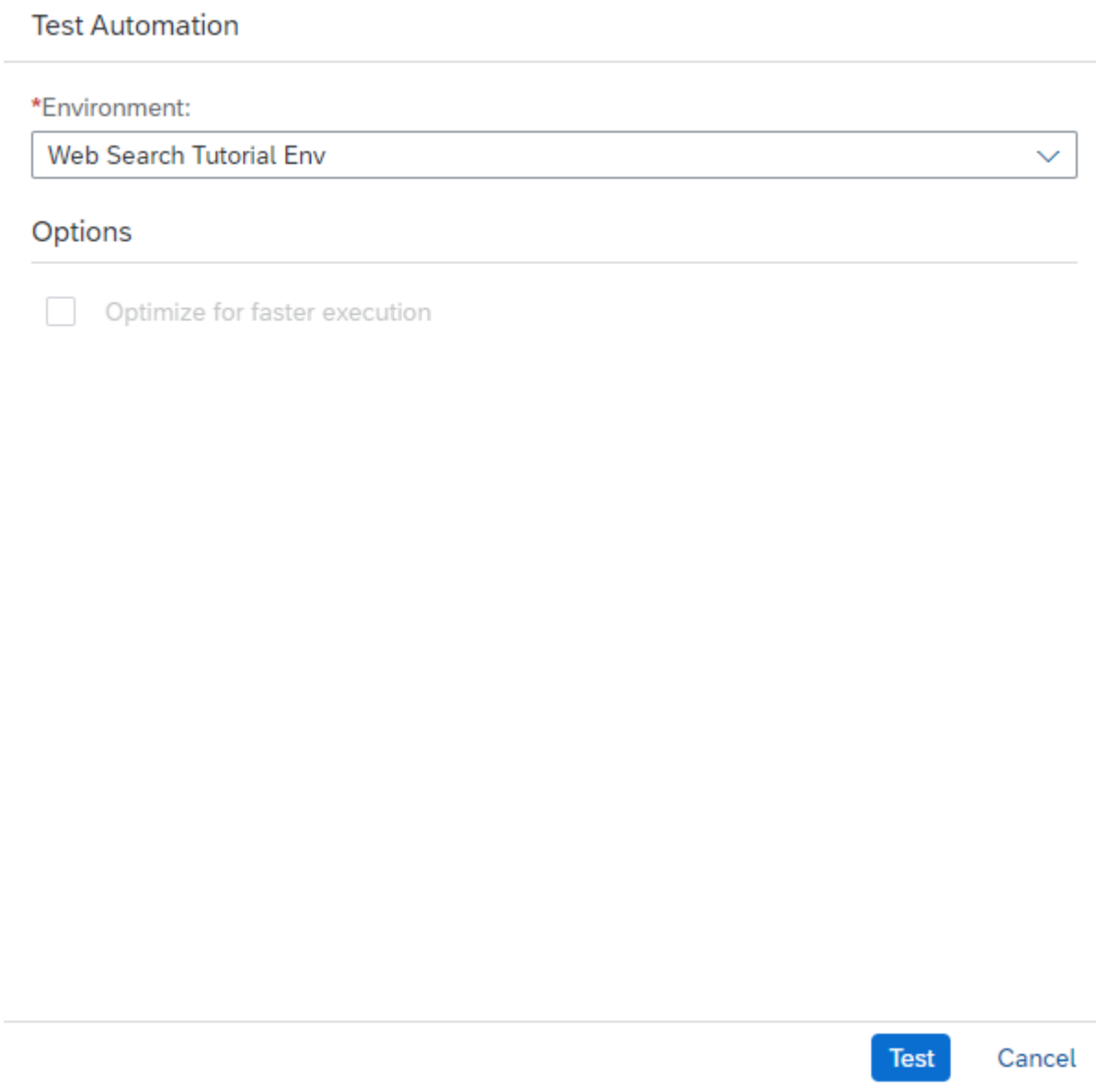


Next Steps

You can now save your automation and test it.

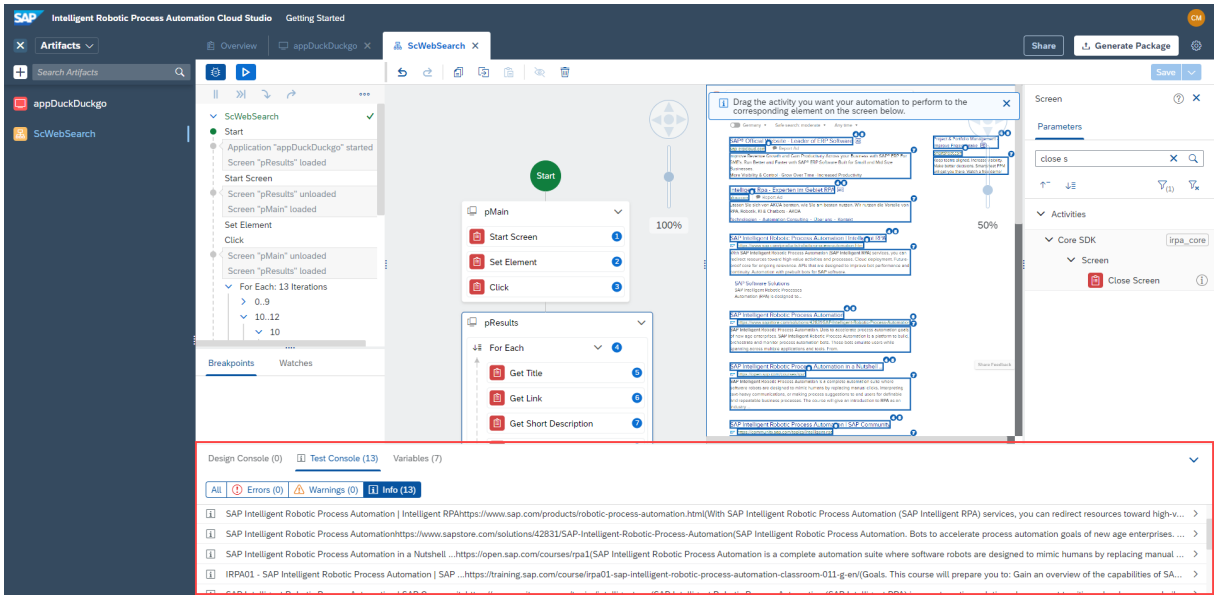
2.1.7 Test your Automation

You can now select your environment and test your automation.



> **Note**
>
> Before testing your automation, you must create an environment in the SAP Intelligent Robotic Process Automation Factory.
>
> For more information, see Create and Manage Environments.

If you have closely followed the steps, you can find the results of your search in the *Info* log at the bottom of the screen.


Intelligent Robotic Process Automation Cloud Studio
Getting Started
Artifacts
Share
Generate Package
Save
appDuckDuckgo
ScWebSearch
Start
pMain
Start Screen
Set Element
Click
pResults
For Each
Get Title
Get Link
Get Short Description
Drag the activity you want your automation to perform to the corresponding element on the screen below.
100%
50%
Screen
Parameters
Activities
Core SDK
Screen
Close Screen
Breakpoints
Watches
Design Console (0)
Test Console (13)
Variables (7)
All
Errors (0)
Warnings (0)
Info (13)
SAP Intelligent Robotic Process Automation | Intelligent RPAhttps://www.sap.com/products/robotic-process-automation.html(With SAP Intelligent Robotic Process Automation (SAP Intelligent RPA) services, you can redirect resources toward high-v...
SAP Intelligent Robotic Process Automationhttps://www.sapstore.com/solutions/42831/SAP-Intelligent-Robotic-Process-Automation(SAP Intelligent Robotic Process Automation. Bots to accelerate process automation goals of new age enterprises. ...
SAP Intelligent Robotic Process Automation in a Nutshell ...https://open.sap.com/courses/rpa1(SAP Intelligent Robotic Process Automation is a complete automation suite where software robots are designed to mimic humans by replacing manual ...
IRPA01 - SAP Intelligent Robotic Process Automation | SAP ...https://training.sap.com/course/irpa01-sap-intelligent-robotic-process-automation-classroom-011-g-en/(Goals. This course will prepare you to: Gain an overview of the capabilities of SA...

3 Design Automations Quickly with the Recorder

You can automate complex workflows easily using the Recorder in Cloud Studio. It automatically captures applications and designs automations accurately at the same time.

> **Note**
>
> The Recorder supports SAP WebGUI since the 2103 release. Recorder support for SAP WinGUI applications is available only for 2105 Beta program subscribers. Support for other technologies is planned for future releases.

The Recorder records the steps you perform across the screens of an application. Screens, elements, and their underlying metadata are automatically captured and properly declared. Then you can export the recording in the automation designer of the Cloud Studio where a workflow is built. The exported recording steps include automatically recorded application screens, elements, metadata, and the automation with a workflow made up of activities. If required, you can manually edit the captured screens and elements or edit the automation to update the *Step Details*. For more details about designing automations and capturing applications in the cloud studio, refer to the Capture an Application [page 77] section.

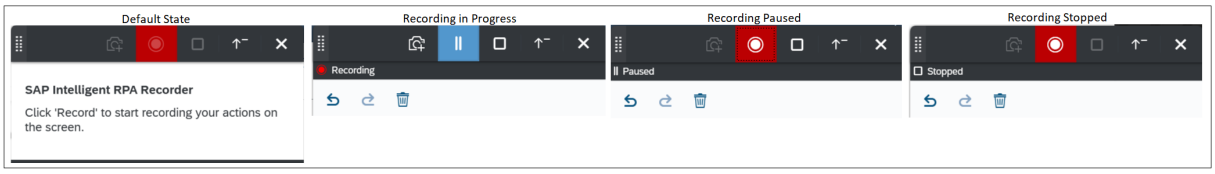
Based on your action as a bot developer, the Recorder records a series of interactions consisting of various steps across the screens such as:

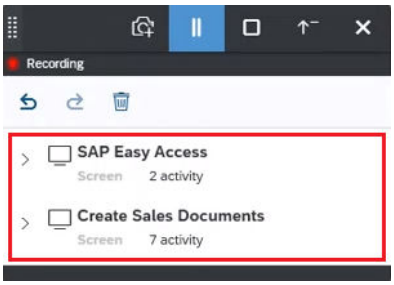
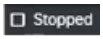
- Entering data,
- Mouse clicks and movements
- Keystrokes with object controls including input fields, tables, menus, buttons, check boxes, and screen tabs.

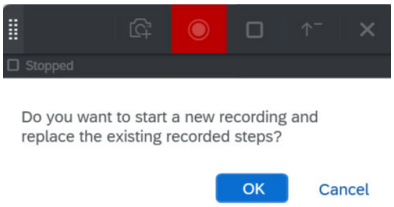
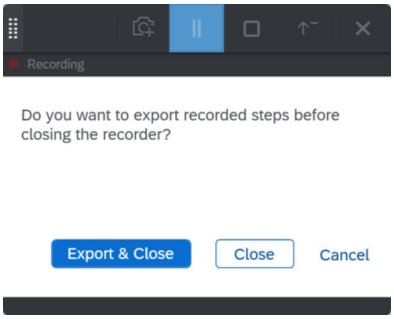
3.1 Overview

The following screenshot shows the various status of the Recorder.

> **Note**
>
> The functionality and UI of the recorder is the same for SAP Web GUI and Win GUI systems.



Name	Button/Icon	Description
Drag/Move		To move the recorder to a desired location in an application window.
Record		To start the recording. Once you click this button, by default, the active screen will be captured.
Capture		To capture the new screen of an application.
Pause		To pause the recording.
Resume		To resume the recording.
Expand/Collapse	 	To view the **Activities List** of recorded screens.  To collapse the **Activities List**.
Recording (Status)		This indicates that the recording is in progress.
Stopped (Status)		This indicates that the recording is stopped.
Undo		To remove unnecessary steps while recording. You can undo only from last recorded step.
Redo		To restore any actions that were previously undone using an Undo function. You can do this only from last recorded step.
Delete All		To delete entire recording. All the steps in the recording widget will be deleted. But WebGUI application being recorded is not rolled back.
Stop		To stop the recording. Once you stop the recording, by default, the *Export* button is displayed.

Name	Button/Icon	Description
		Rerecord - If you want to replace the existing recorded steps and start with new recording, then click  . The start new recording message is displayed.  • *OK* - If you want to start a new recording, then click this button. **i Note** If you click *OK*, steps will be cleared, and recording will start from the beginning. • *Cancel* - If you want to continue with the existing recording, then click this button. All recorded steps will be retained, then the *Export* button is displayed.
Close		To close the Recorder. Once you click this button, the closing the recorder confirmation message is displayed.  • *Export and Close* - If you want export the recorded steps to Cloud Studio and close the **Recording** widget, then click this button.

Name	Button/Icon	Description
		• *Close* – If you want to exit the **Recording** widget without exporting the recorded steps to Cloud Studio, then click this button. • *Cancel* – If you want to continue with the recording, then click this button. Retains the same state as before.
Expand/Collapse		To view list of activities of a specific screen. To collapse the activities list of a screen.
Export		To export the recording in the automation designer of the Cloud Studio.

3.2 Prerequisites

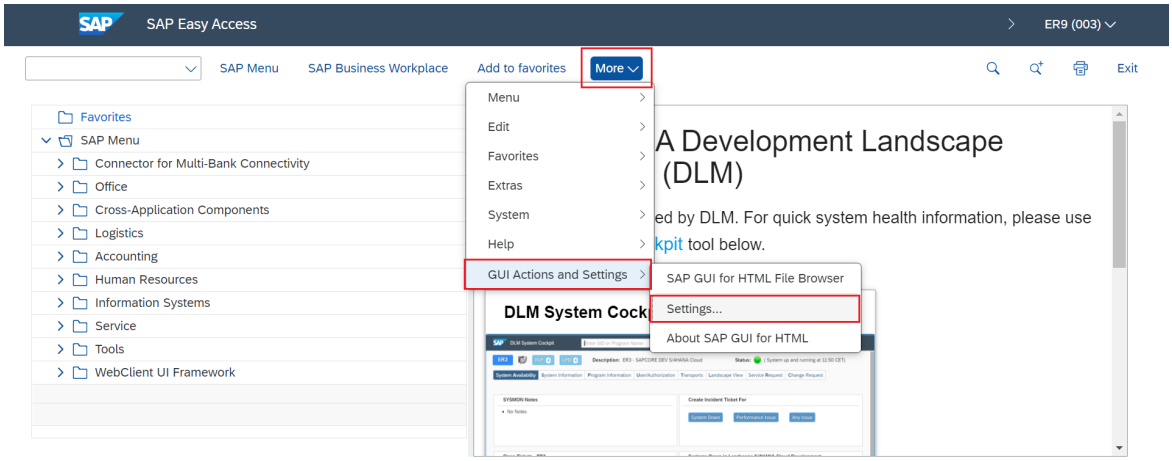
This section describes the necessary prerequisites to use the Recorder in the Cloud Studio with the SAP WebGUI and Win GUI systems.

Configuring SAP GUI for HTML (WebGUI)

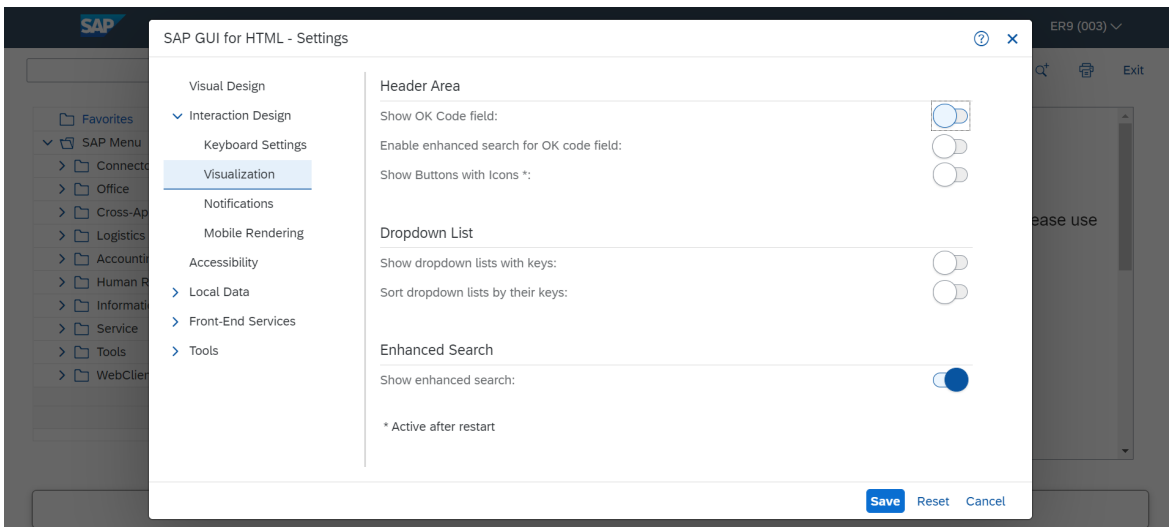
Check the following SAP Note to learn how to activate the WebGUI connector interface of SAP WebGUI: 2853735.

Enabling Show OK Code Field

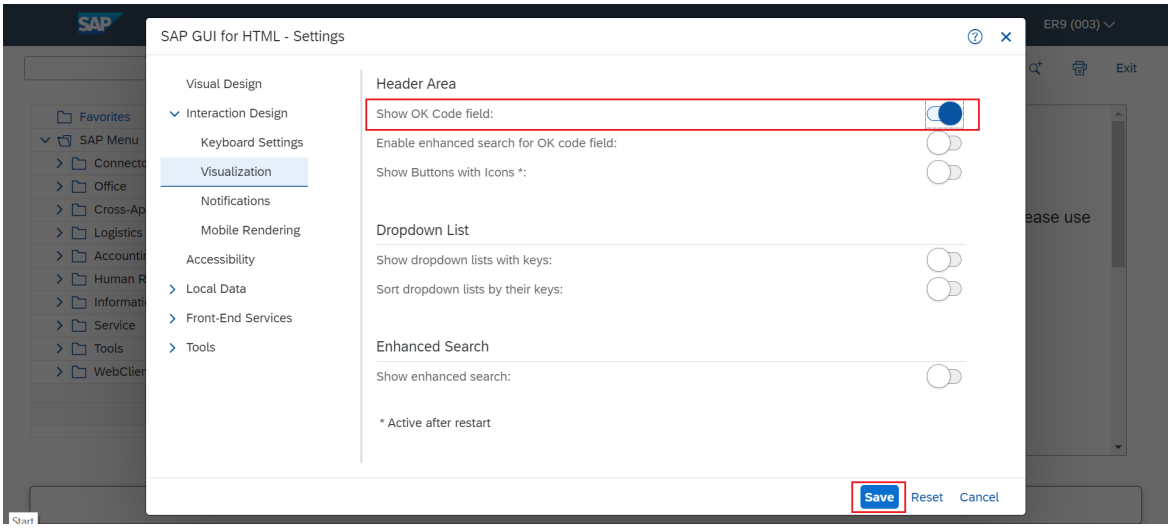
1. In the SAP WebGUI application, Click *More*. The *More* menu is displayed.
2. Click *GUI Actions and Setting* and then click *Settings*.



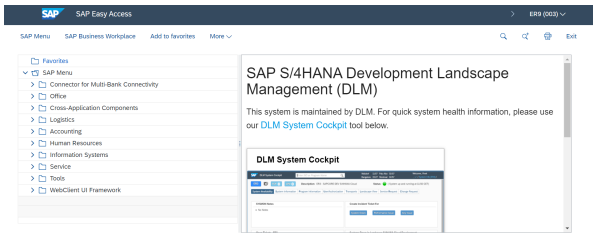
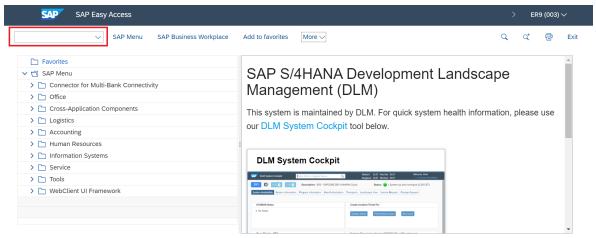
The SAP GUI for HTML – Settings (by default, *Visualization* is highlighted) screen is displayed.



3. Enable the *Show OK Code field*.



4. Click *Save*. The *Show OK Code field* is enabled.

SAP WebGUI before enabling OK Code field.	SAP WebGUI after enabling OK Code field.
	

Configuring SAP GUI for Windows (WinGUI)

To use recorder with SAP WIN GUI, SAP scripting and Recording must be enabled in the SAP WIN GUI system.

Enabling Scripting on the Client

To enable SAP scripting on the client, refer to the Enabling Scripting on the Client Side section.

Enabling Scripting on the Server

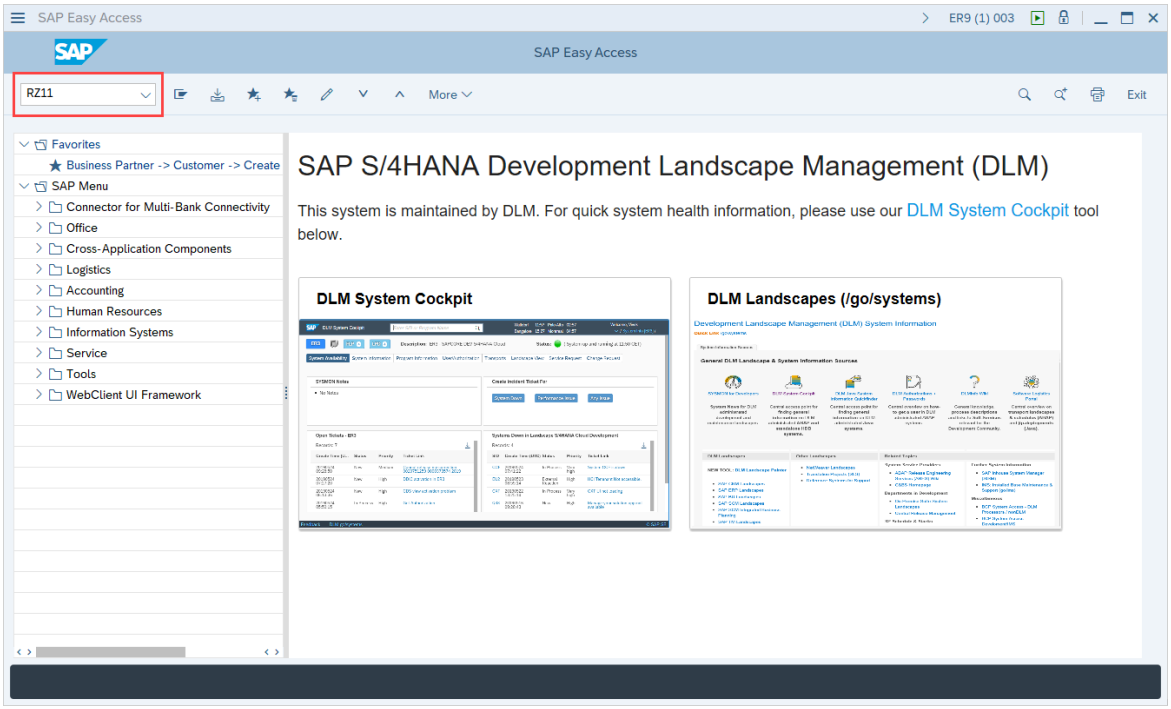
To enable SAP scripting on the server, refer to the Enabling Scripting on the Server Side section.

Enabling Recording

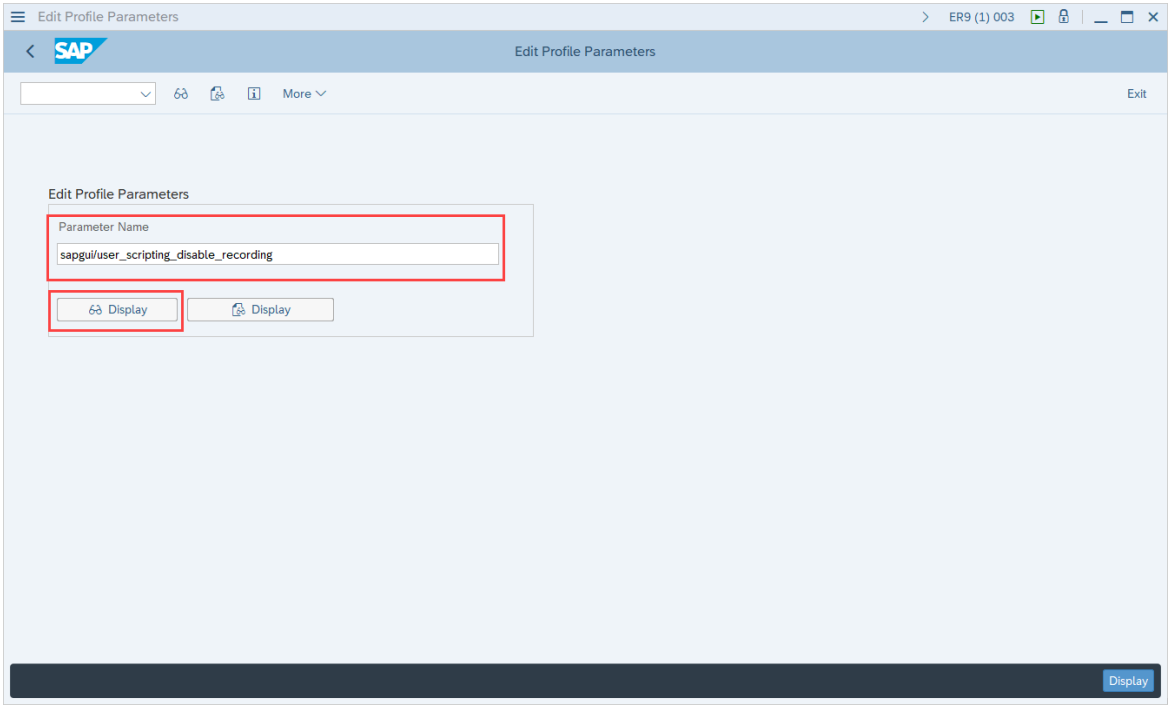
If you are using an SAP Win GUI application for the first time, then you must enable Recording. Else, the Recorder button will not be enabled in the Cloud Studio.

To enable Recording in the SAP Win GUI application, follow the below procedure:

1. Launch SAP Logon and connect to the server.
2. In the *Enter Transaction Code* search field of the *SAP Easy Access* screen, enter the transaction code, **RZ11**.

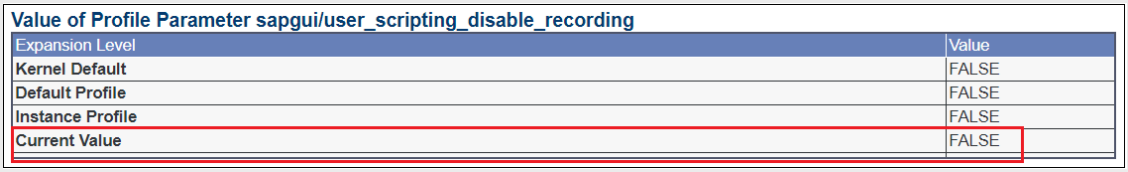


3. Press **Enter**. The *Edit Profile Parameters* screen is displayed.
4. Enter the parameter name, **sapgui/user_scripting_disable_recording**, in the *Parameter Name* field.



5. Click *Display*. The parameter details screen is displayed.
6. In the *Value of Profile Parameter sapgui/user_scripting_disable_recording* table, check the value of *Current Value*. If the value is **TRUE**, then click *Change Value*.

> **i Note**
>
> If the *Current Value* is set to **FALSE**, the recording is enabled already and you can exit the screen.

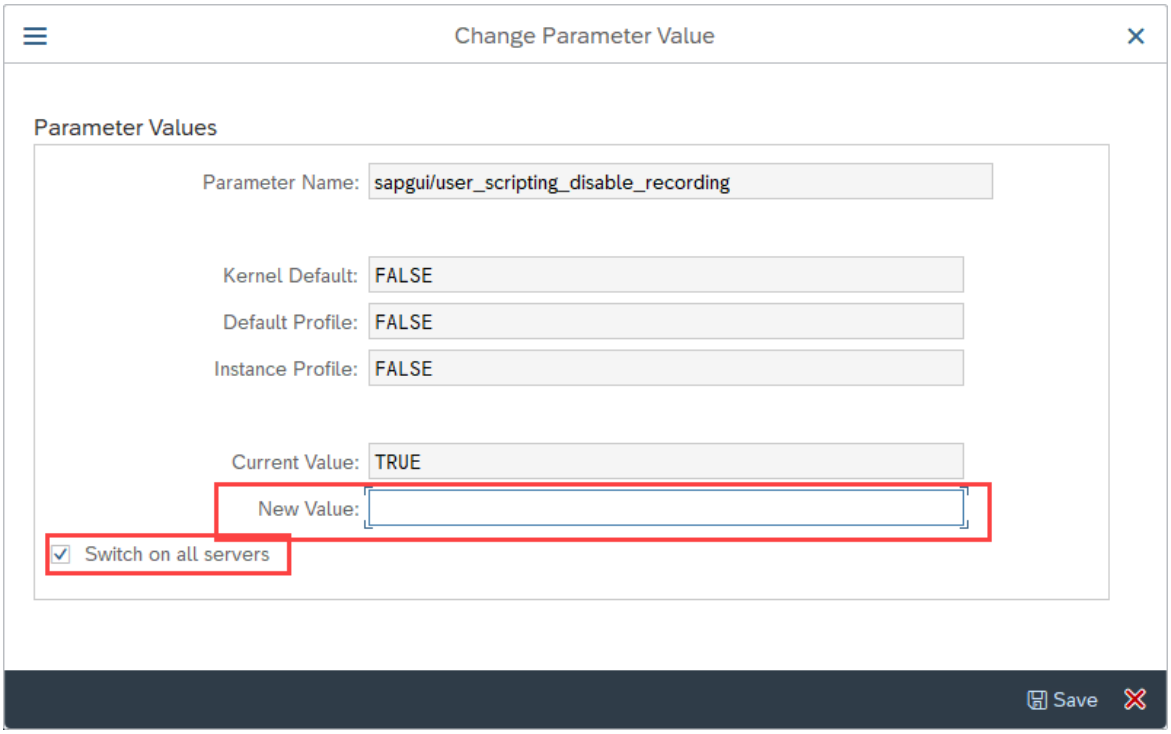


Value of Profile Parameter sapgui/user_scripting_disable_recording

Expansion Level	Value
Kernel Default	FALSE
Default Profile	FALSE
Instance Profile	FALSE
Current Value	FALSE

The *Change Parameter Value* screen is displayed.

7. Enter **FALSE** in the *New Value* field. If required, un-check the *Switch on all servers*.



8. Click *Save*. The recording is enabled in the SAP Win GUI system.

3.3 Automatically Capture Applications with the Recorder

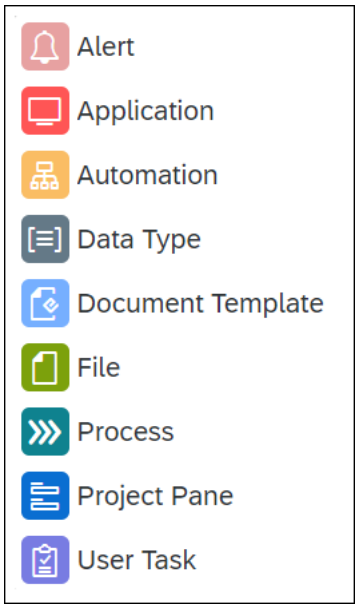
This section describes how to use the Recorder for an SAP WebGUI or WinGUI application.

Before recording the steps of a workflow, ensure that the WebGUI application and the Cloud Studio are running in a Chrome browser in two separate windows. The Desktop Agent must be configured to use with the target tenant.

> **i Note**
>
> The procedure for recording the steps of a workflow in the Cloud Studio with the SAP WebGUI and SAP WinGUI systems is same.
>
> The below procedure is performed using the SAP WebGUI system.
>
> The screenshots and data used in this section are only an example for understanding.

Getting Started

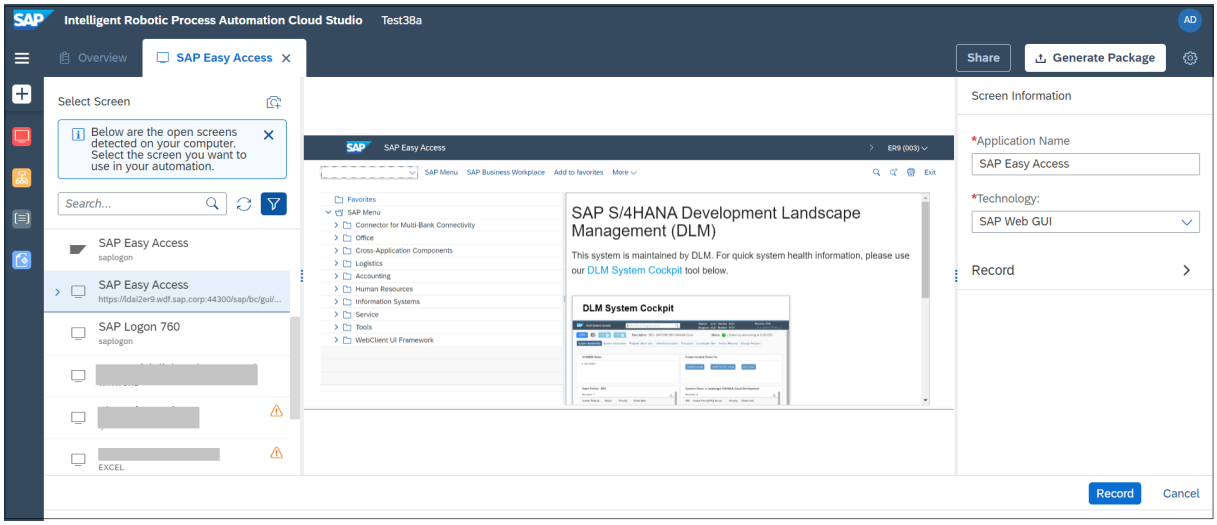
To start recoding, you must create a project in the Cloud Studio. In the Project Explorer, click + and then click *Create* in the left panel or click *Create* in the Project Explorer main panel. The artifact menu is displayed.



In the artifact menu click *Application*. A new tab labeled **Untitled** opens. Later, this label will change to the name of the captured application. The system starts detecting the applications and their screens currently running on your local machine. When it's done, you'll see a list of screens in the picker panel on the left.

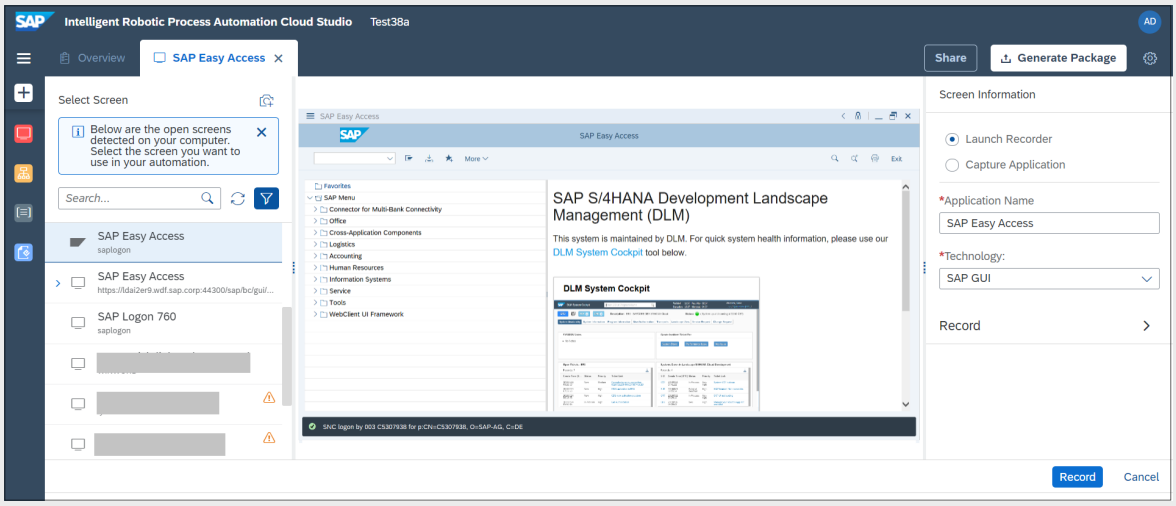
To know more details about capturing an application, refer to the Capture an Application [page 77] section.

Select the WebGUI application you want to record from the list of screens. A preview of the screen is displayed in the capture area.



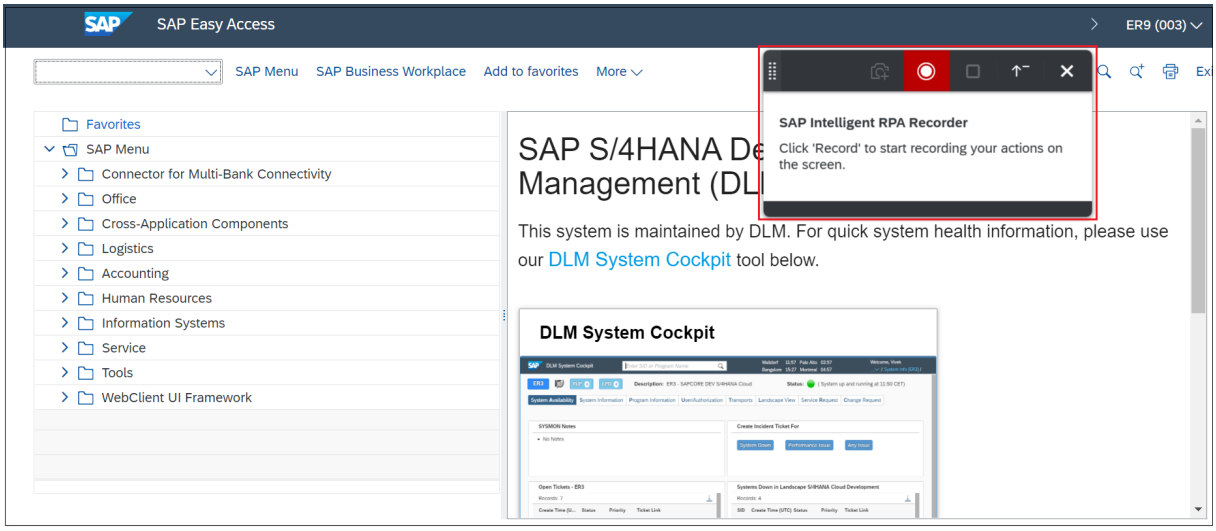
> **Note**
>
> The recorder button is only enabled if the detected application is an SAP WebGUI or SAP WinGUI application.

> **Note**
>
> Below is the WinGUI preview in the Cloud Studio for reference:

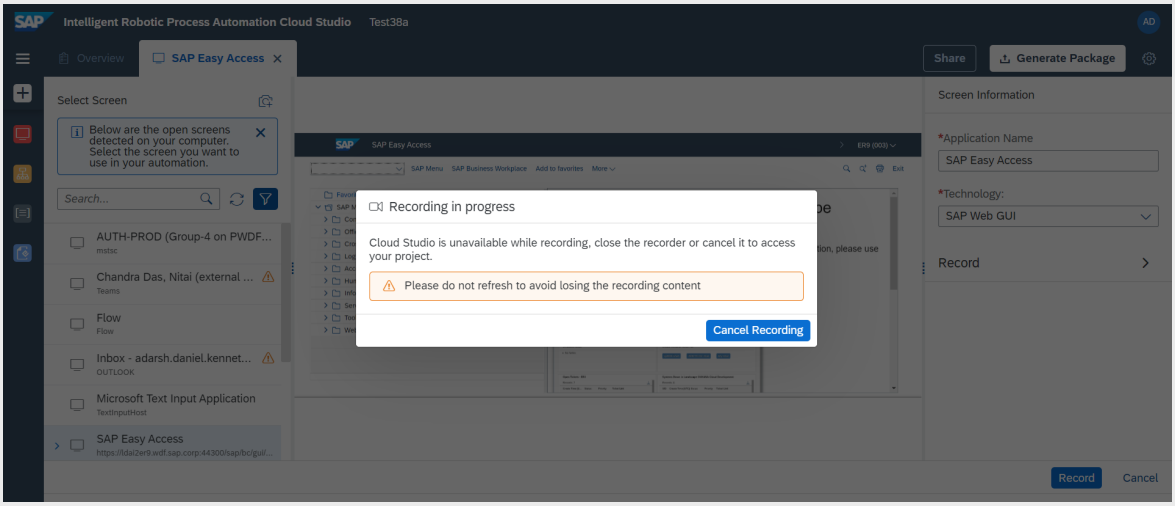


Click *Record*. After you click the *Record* button, you will be directed to the WebGUI application you want to record.

In the WebGUI or WinGUI application, by default, the **Recorder** is enabled.



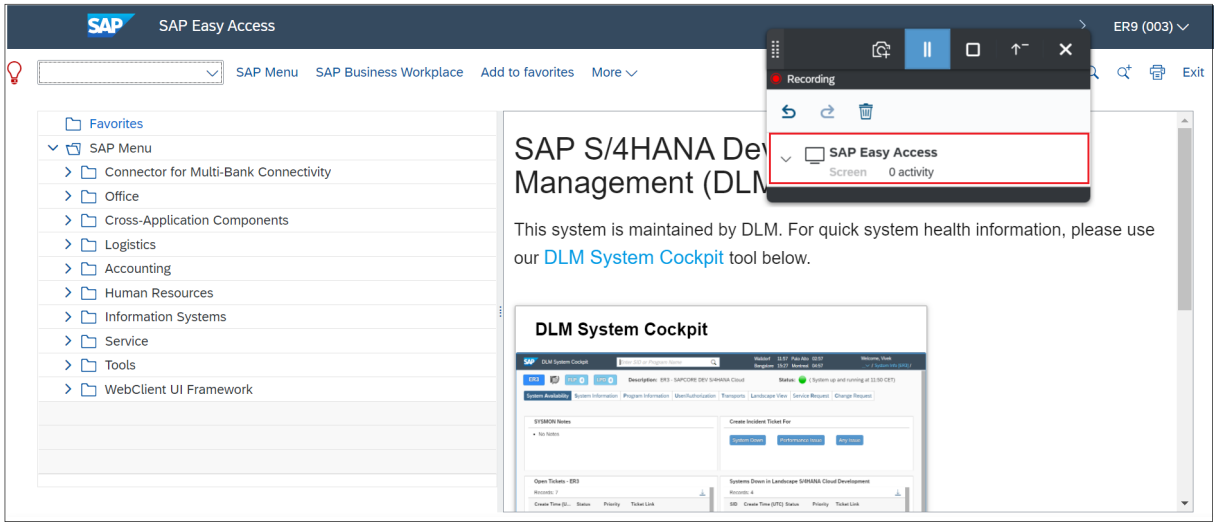
> **Note**
>
> When you click the *Record* button the Cloud Studio is locked and the *Recording in progress* pop-up message is displayed.



If you want to cancel the recording, click *Cancel Recording*. The *Cancel recording and dismiss all created steps?* warning pop-up is displayed.

Click *OK*. The recording will be cancelled and then Cloud Studio window is displayed.

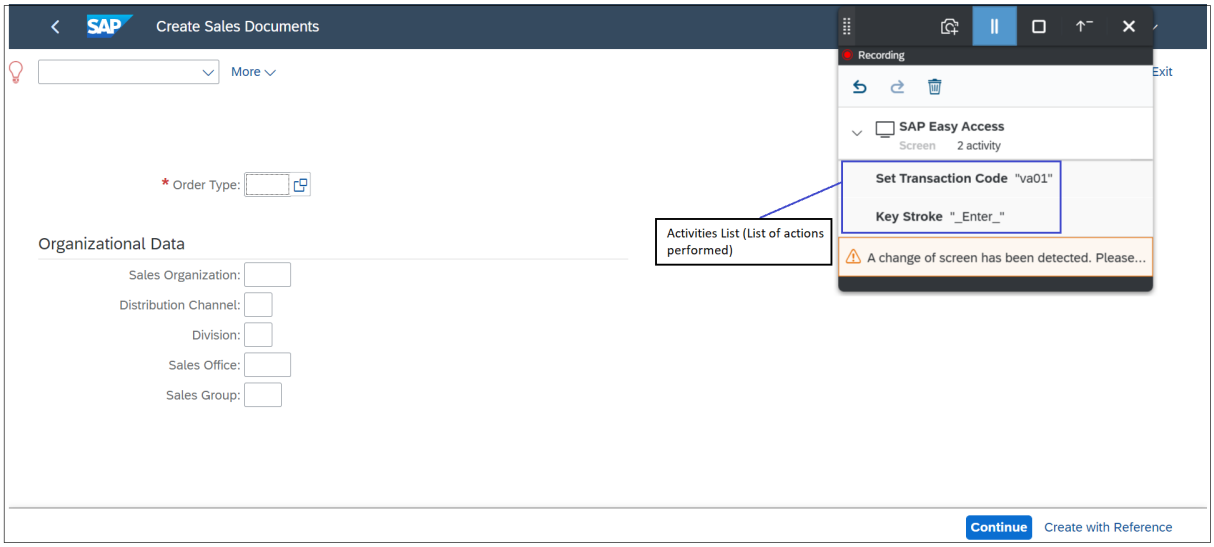
To initiate the recording, you must click  button. The recorder captures the first screen of an application automatically. You can see it in the **Activities** list.



The recorder captures the first screen of an application automatically and subsequent screens must be captured manually by clicking the  button after which the actions you perform on that screen will be recorded automatically.

The steps are recorded even if you do not click the  button manually to capture new screens. However, the generated automation will be incorrect. This is applicable across the workflow recording.

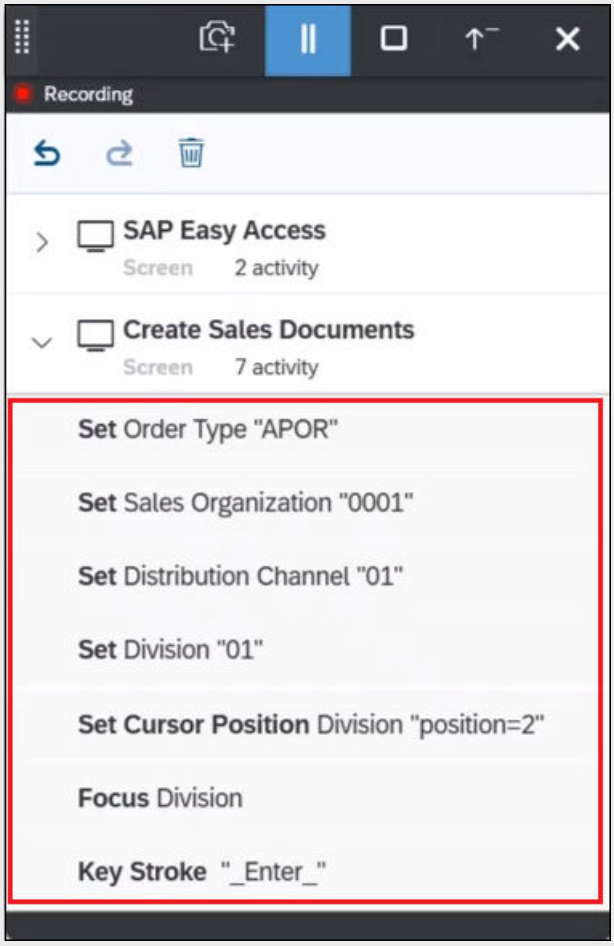
For example, refer to the below screenshot. The recorder automatically records all actions performed on the screen.



Note

The actions performed on the screen are updated in the **Activities** list only after you press **Enter**.

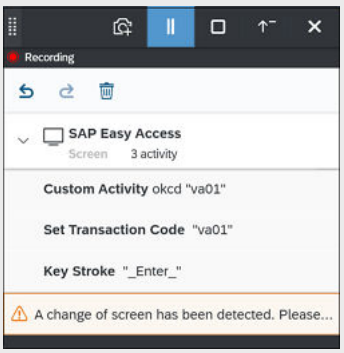
This is because, when you press **Enter**, a server round trip happens and the recording widget is refreshed with data from the server.



While recording, if a new screen appears or if there is a change in UI, then you must first click the button manually and then perform the actions in the application screens. This is applicable across the workflow recording.

i Note

Whenever new screen appears or if there is a change in UI, the recorder detects the screen change in most of the cases and displays a hint message. This gives you a hint to capture a screen manually by clicking the button.

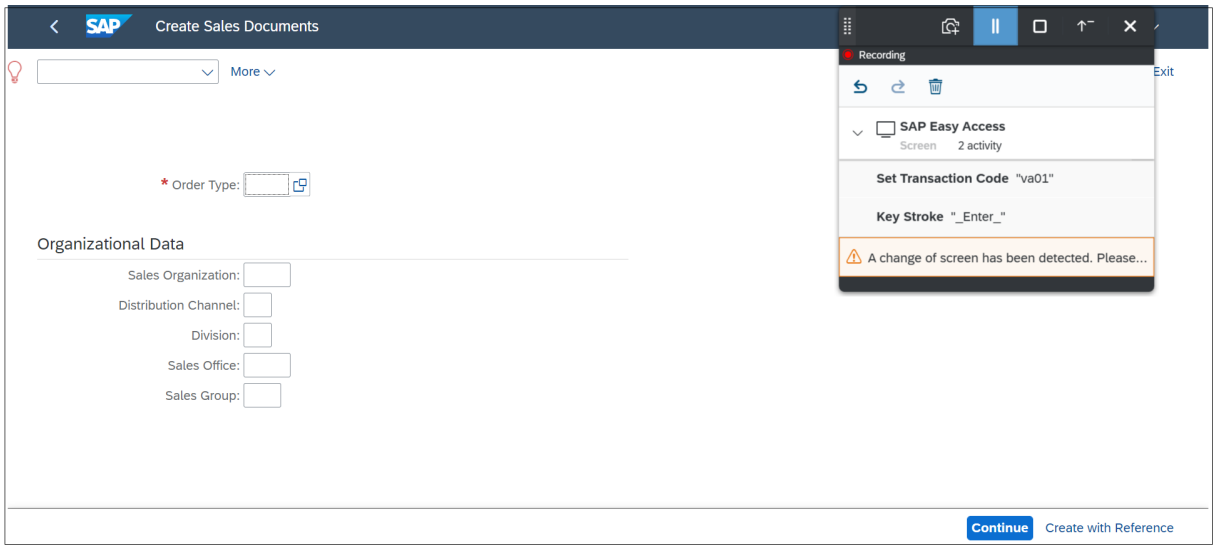


> The Recorder may not detect screen changes in all cases. You must click the button whenever there is a change in the screen or UI.
>
> The hint message disappears when you click the button or perform any action in the screen.

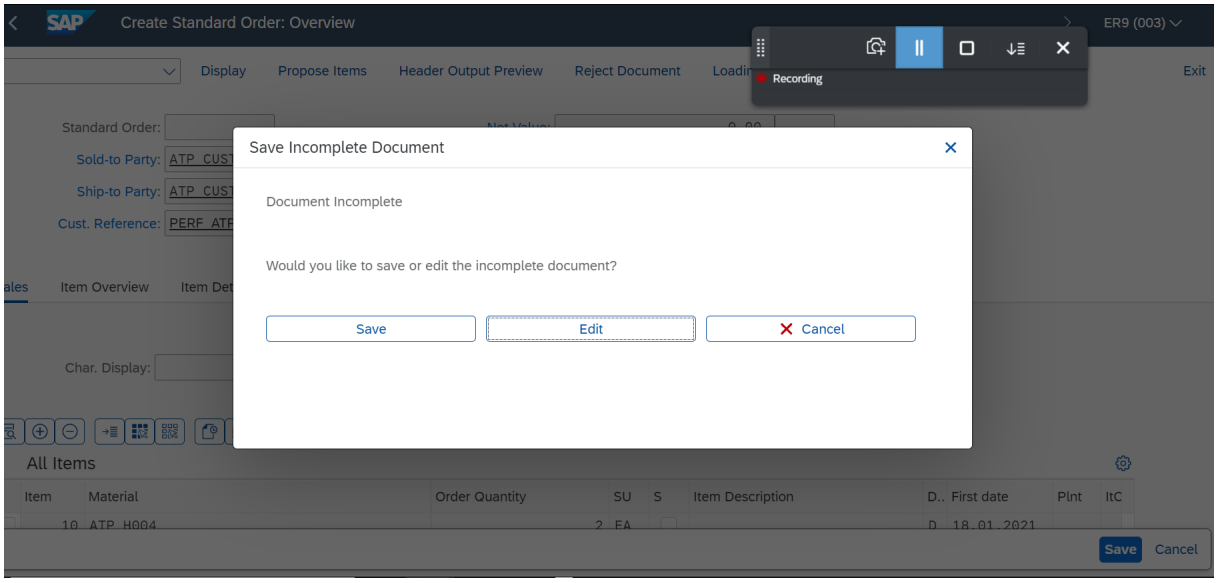
A new screen or change in UI may include application screen, modal dialogs, pop-ups, value helps (F4), UI elements (control, table, and hierarchical list) and many more.

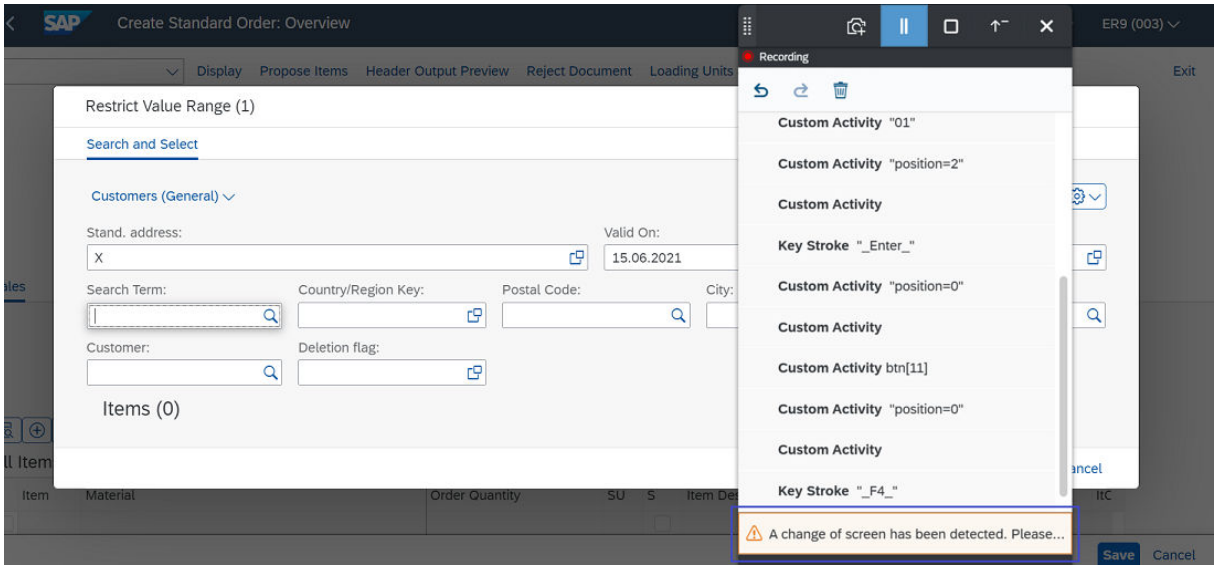
For example, refer to the below screenshot.

New Screen



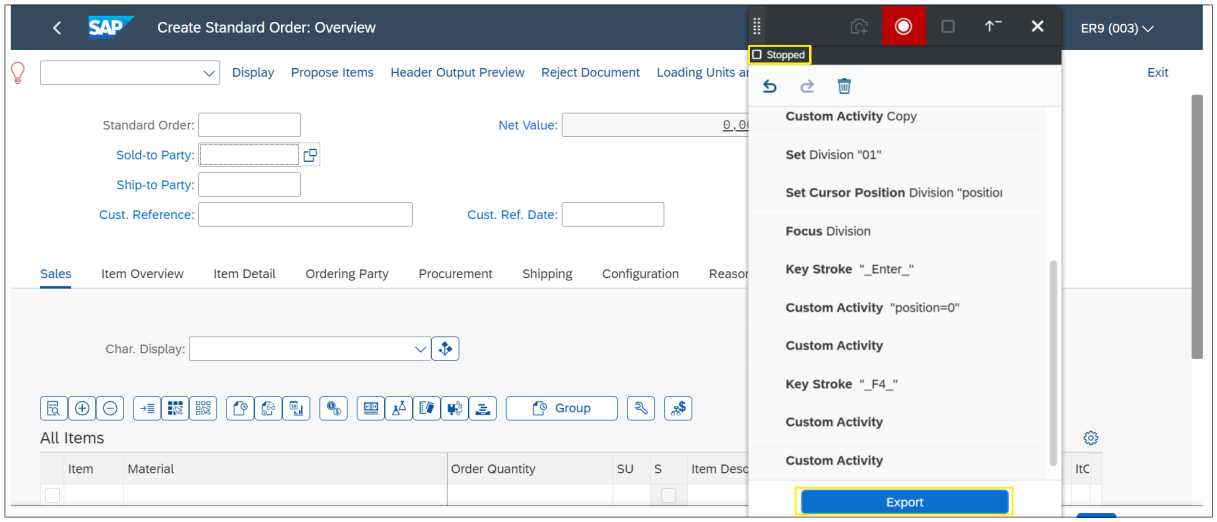
Pop-up Screen





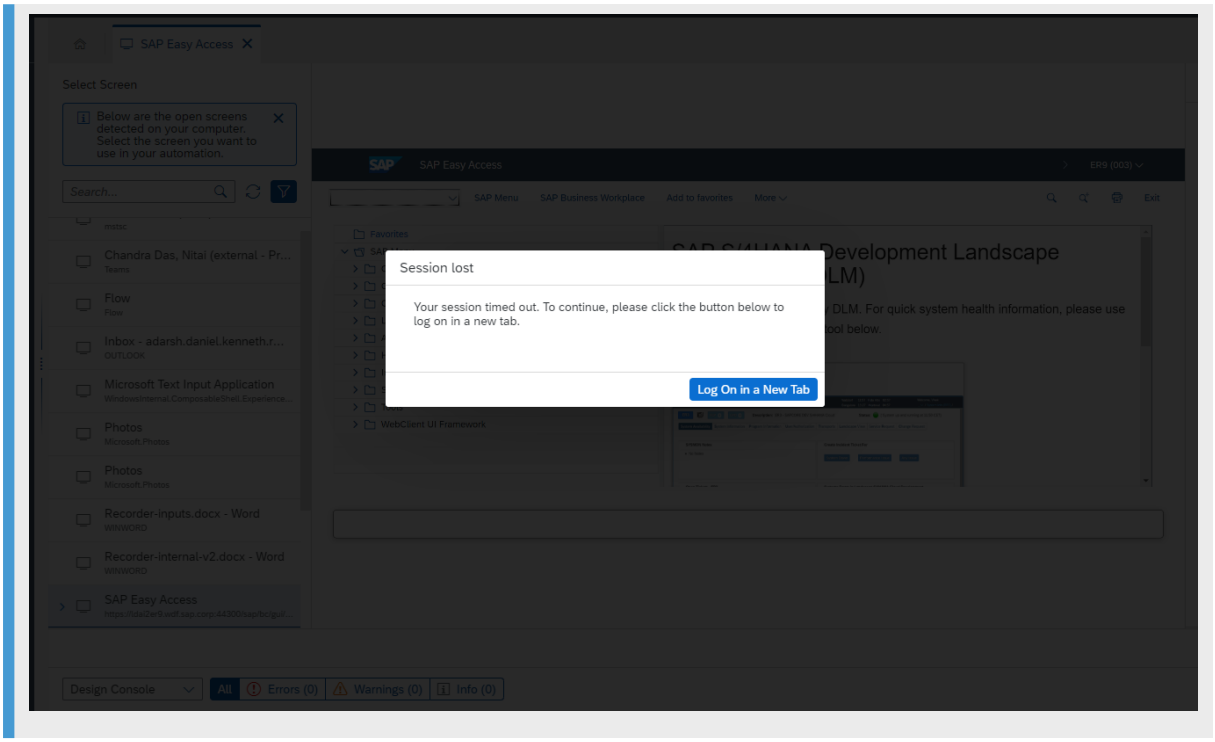
Once you have completed entering the data in the application screens, you can stop recording. Click the ⏹ button to stop the recording.

After you stop the recording, by default, the *Export* button is displayed.

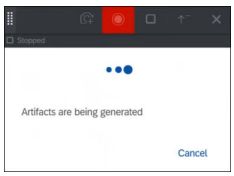
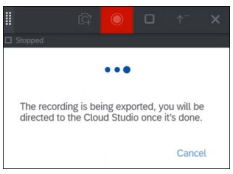
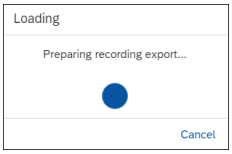
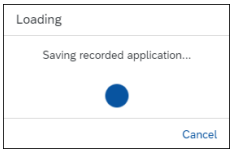
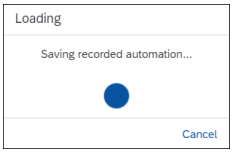


ℹ Note

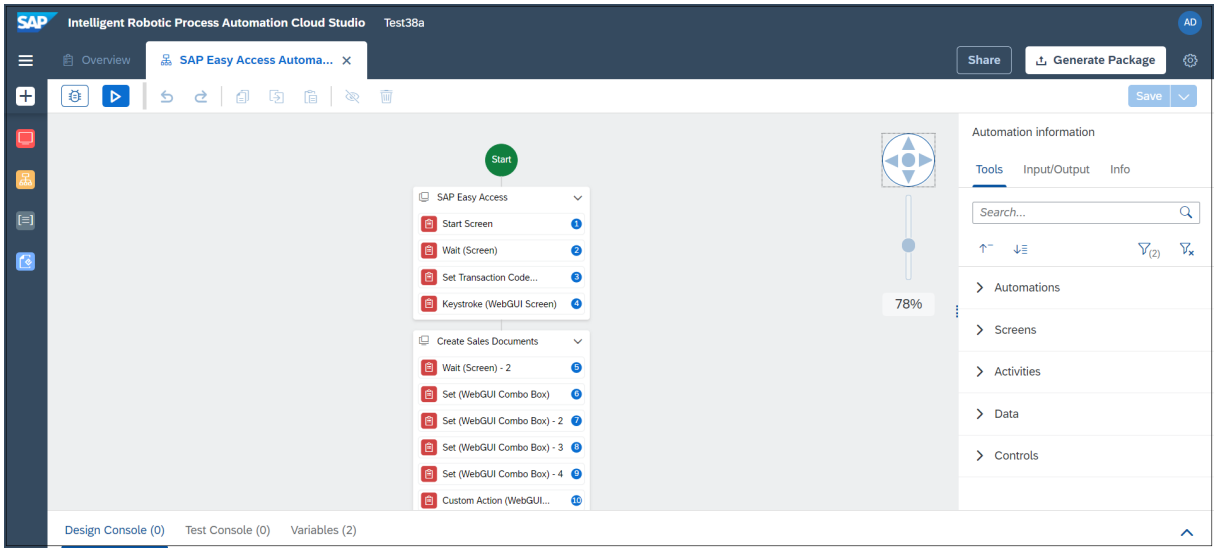
Before you start exporting the recording to the Cloud Studio, check if the Cloud Studio has not timed out. If it is timed out, you must re-login (do not refresh) to the Cloud Studio and then export the recording.



To export the recorded steps to the Cloud Studio, click **Export**. While exporting is in progress, the system displays the following pop-ups.

Recorder Widget	Recorder Widget	Cloud Studio	Cloud Studio	Cloud Studio
				
Cancel If you do not want to export the steps, then click. The Recorder widget retains the same state as before.	The *Cancel* is disabled.	If you do not want to export the steps, then click *Cancel*. The Recorder widget will be closed and *Recording successfully cancelled* message is displayed.	If you do not want to export the steps, then click *Cancel*. The Recorder widget will be closed and *Recording cancelled* message is displayed.	If you do not want to export the steps, then click *Cancel*. The Recorder widget will be closed and *Recording cancelled* message is displayed.

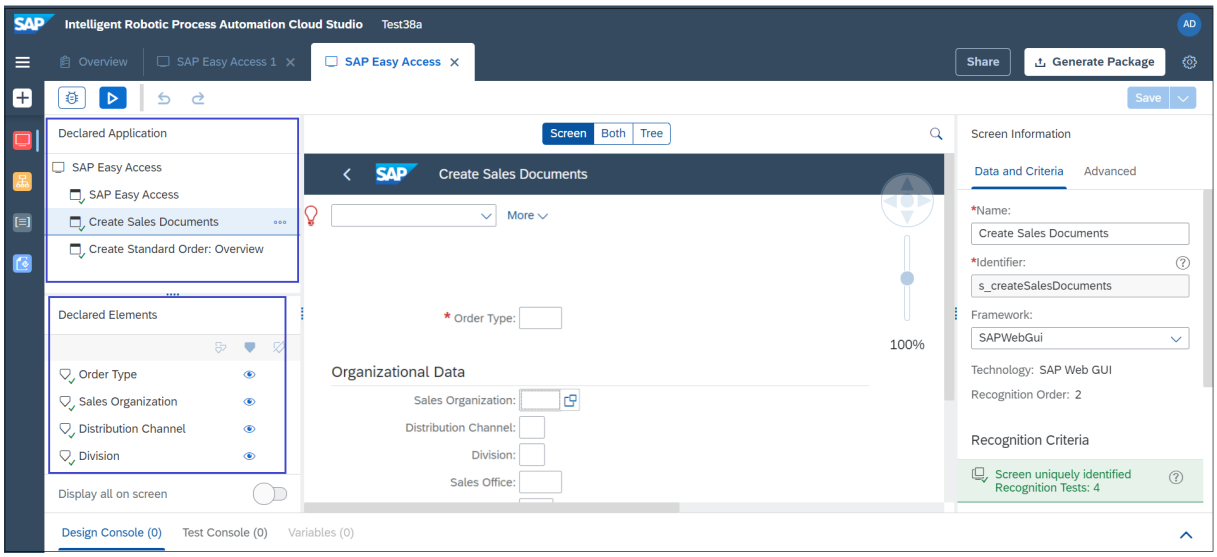
After successful processing, **Recorder** widget will be closed. You will be redirected to the Cloud Studio and the confirmation message, *Recording successfully exported* is displayed. All the recorded steps will be displayed in the automation designer of the Cloud Studio.



You can then edit the automation to update the *Step Details*. For more details about designing automations in the cloud studio refer to the Create an Automation [page 156] and Manage Data within an Automation [page 160] sections.

View Recorded Screens

You can view the recorded screens under *Declared Application* and the recorded elements under *Declared Elements*.



If required, you can manually edit the screens to add or remove objects. For more details about editing the screens, refer to the Declare an Application [page 101] and Declare Elements [page 107] sections.

Test an Automation

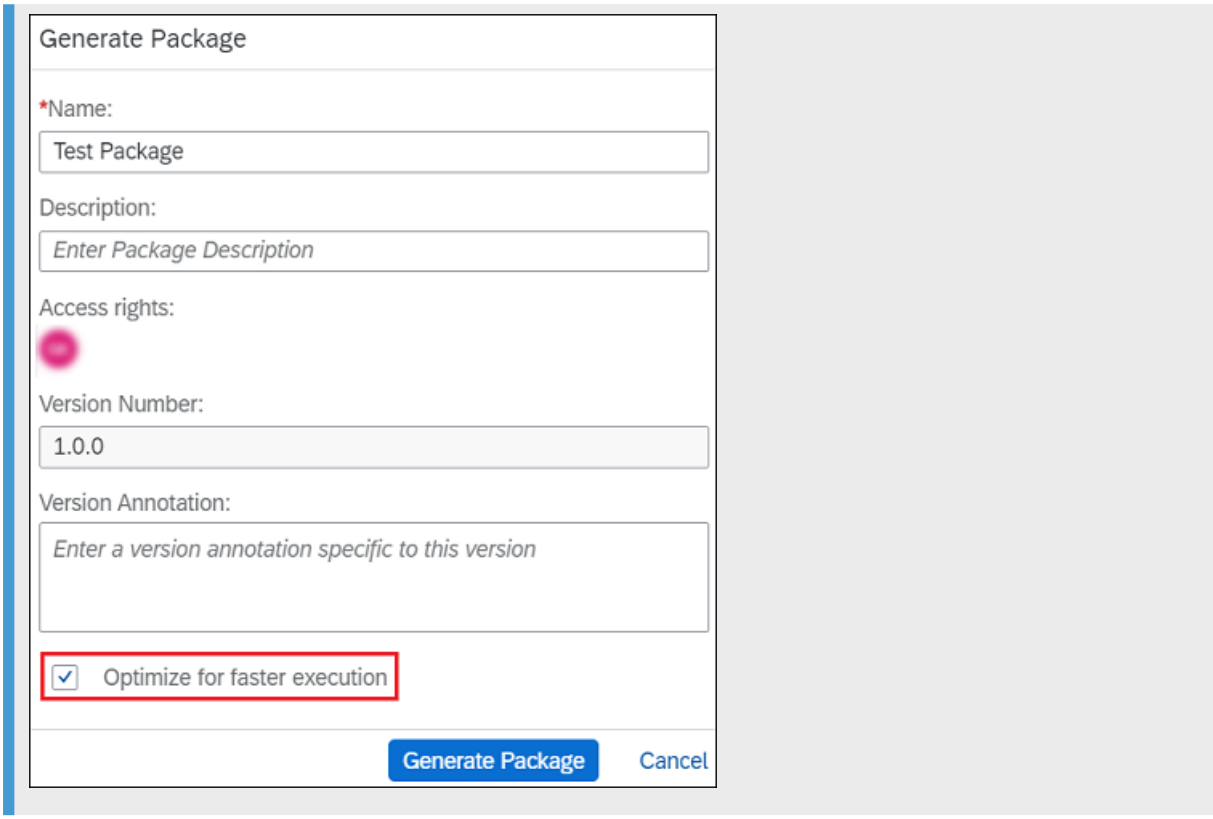
After you export the recorded steps to the Cloud Studio and finalize the automation, screens, and its elements, you can test the execution flow of your automation. For more details about testing an automation, refer to the Test an Automation [page 282] section.

> **i Note**
>
> **Batch Execution of Recorded Actions**
>
> In SAP WebGUI Recorder, you can batch a set of recorded actions that are performed on the same screen and send those actions together to the REST GUI. It allows you to execute your automation faster.
>
> While testing the recorded automation, you must enable the **Optimize for faster execution** option by selecting the following check box on the **Test Automation** window.
>
> 
>
>
> You must not use this option for debugging purposes as the actions are executed in a batch mode, and the data in the tester window might not be valid. Therefore, it is recommended to enable this feature once your automation is finalized.
>
> The batch execution is enabled for a project that depends on SAP WebGUI SDK.

Generate Package

Once you have designed your project, you can generate a package in the Cloud Factory. To know more about generating packages, refer to the Generate Packages [page 508] section.

> **i Note**
>
> While generating a package, you can also enable the **Optimize for faster execution** option by selecting the following check box on the **Generate Package** window. It will generate a package containing the optimized version of the automation.



3.3.1 Implement Collection in Recorded Elements

You can create an element and declare it as a collection or modify an existing recorded element and declare it as a collection and use them in the automation.

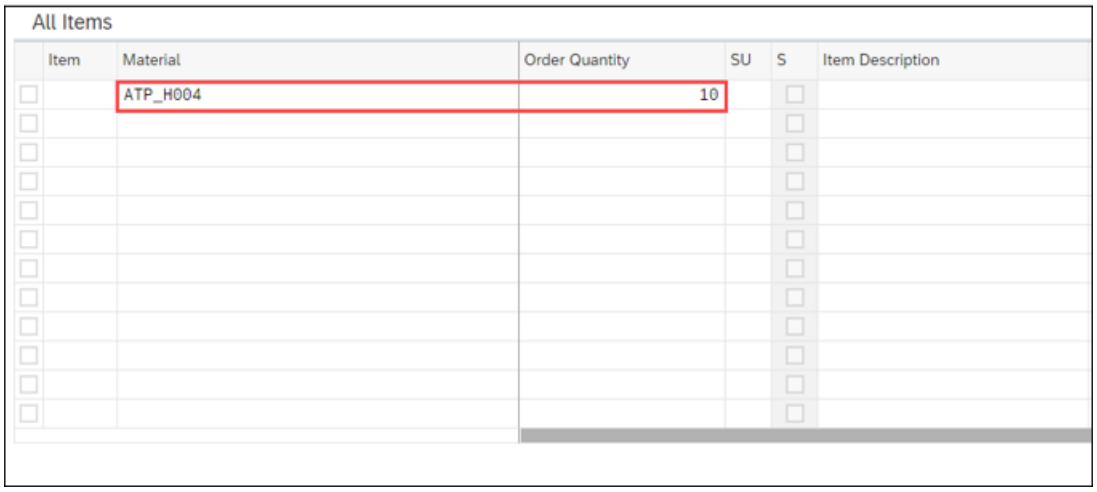
Prerequisites

You've captured at least one screen of the SAP WebGUI or WinGUI application using the Recorder. To know more details about capturing the SAP WebGUI or WinGUI application using the Recorder, see Automatically Capture Applications with the Recorder [page 45]. You've manually edited the screens to add or remove objects. For more details about editing the screens, refer to the Declare Elements [page 107] section.

Example

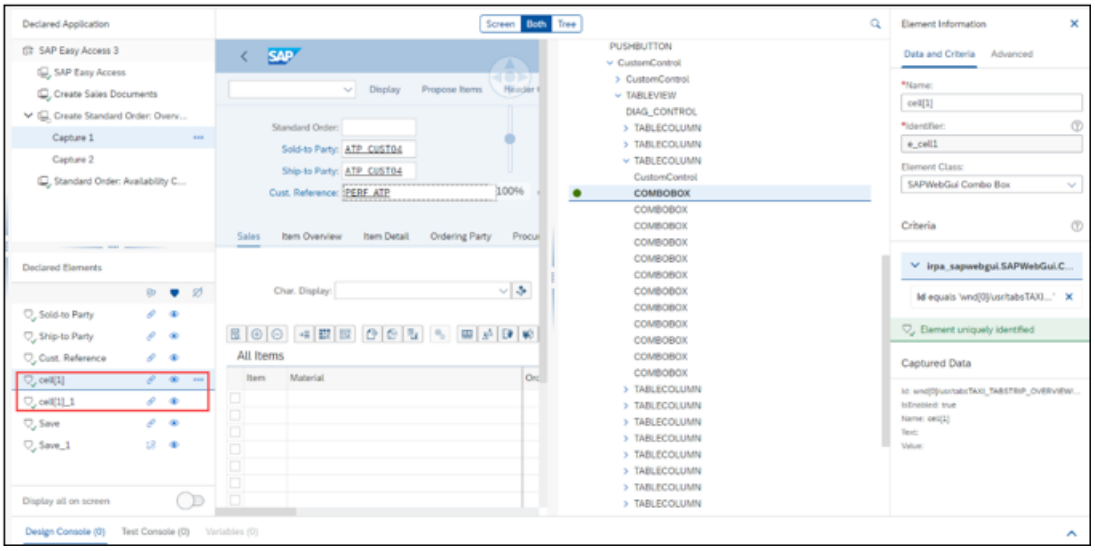
The following scenario describes the usage of collection element while recording a sales order creation in SAP WebGUI application.

In the sales order creation, you will encounter a table where you need to enter the items such as *Material*, *Order Quantity* and so on.



While recording you can just enter one *Material* and *Order Quantity* value in the table like the image mentioned above and finish creating the sales order. An element will be created for both the first cells of the *Material* and the *Order Quantity* columns of the table.

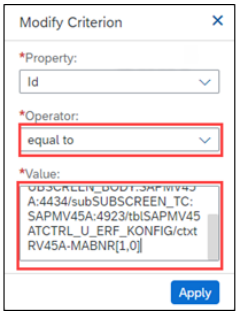
In the recorded application of the sales order creation, two elements *cell[1]* and *cell[1]_1* are created under *Declared Elements*. The *cell[1]* is for the first cell in the *Material* column and the *cell[1]_1* is for the first cell in the *Order Quantity* column.



By default, the criteria for *cell[1]* element is *Id* equals

```
wnd[0]/usr/tabsTAXI_TABSTRIP_OVERVIEW/tabpT\01/ssubSUBSCREEN_BODY:SAPMV45A:4434/
subSUBSCREEN_TC:SAPMV45A:4923/tblSAPMV45ATCTRL_U_ERF_KONFIG/ctxtRV45A-MABNR[1,0]
```

Here the ‘0’ in the Id represents the first cell in the column.



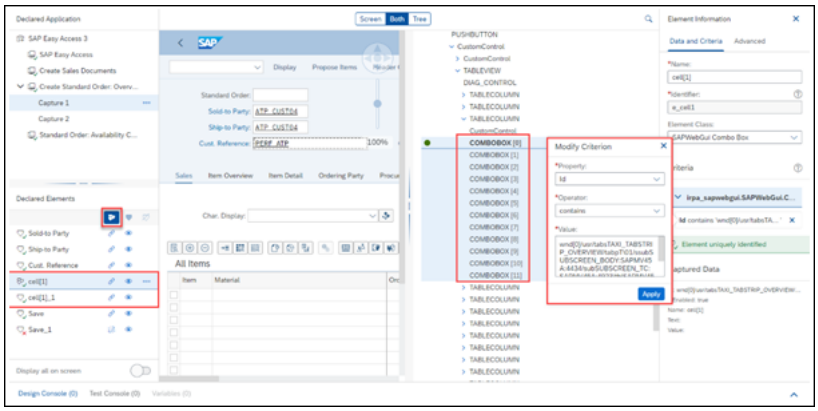
Context

You can record a dynamic number of rows using the collection control feature in the SAP WebGUI application. This means you can record one entry on a row and export it to the Cloud Studio. In the Cloud Studio, you need to make few modifications to the recorded application (such as modifying the *Criteria*, marking the item as occurs) and the automation (such as adding the *For each* loop) to create a dynamic number of rows. When the bot is executed, all defined number of rows will be filled in automatically, like in any other technology.

To implement collection in recorded elements, follow the below procedure:

Procedure

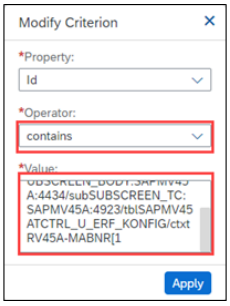
1. Under *Declared Elements*, select the element to be declared as collection, modify the criteria of the element, and then click *Is a collection* ( ) button.
   In the sales order creation, select *cell[1]* and *cell[1]_1* element respectively, and make these two elements into a collection by changing the criteria in such a way that it can recognize all the cells in that column.



Modify the criteria for *cell[1]* to *Id* contains

```
wnd[0]/usr/tabsTAXI_TABSTRIP_OVERVIEW/tabpT\01/ssubSUBSCREEN_BODY:SAPMV45A:
4434/subSUBSCREEN_TC:SAPMV45A:4923/tblSAPMV45ATCTRL_U_ERF_KONFIG/ctxtRV45A-
MABNR[1
```

Here you can see that the Id is trimmed till the first index so that it is unique for all the cells in that column.



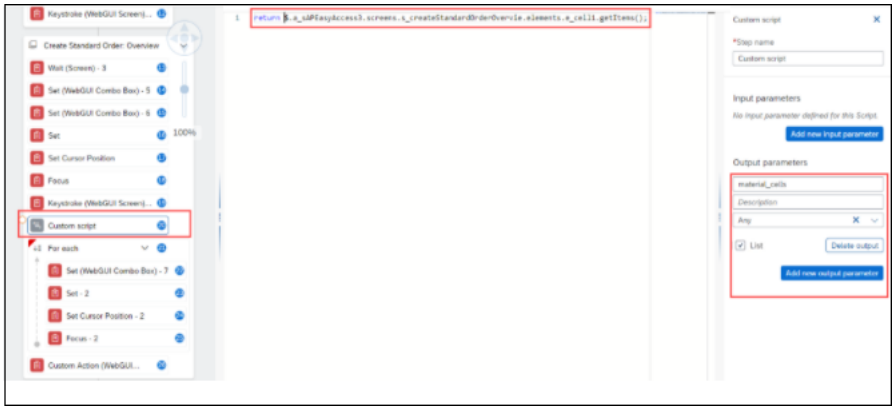
2. Identify all the steps that have collection element as target in the automation and then embed them inside a *For each* loop.
   In the sales order creation, identify all the steps that have *cell[1]* and *cell[1]_1* as target in the automation and then embed them inside a *For each* loop.



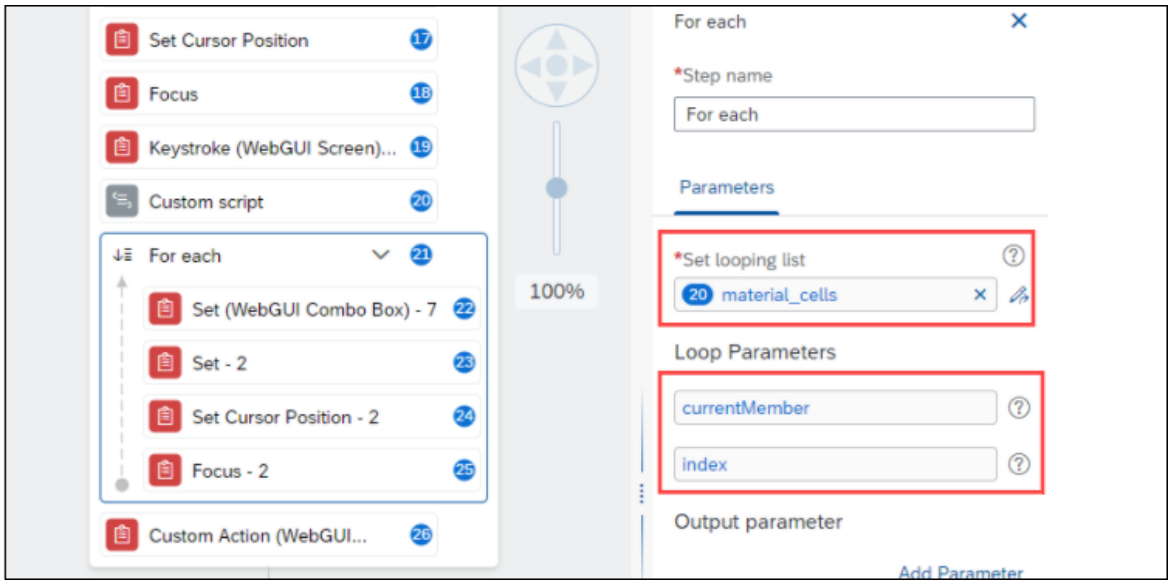
3. Modify the *For each* loop in such a way that it will enter the same item values (such as *Material* and *Order Quantity* values in the sales order creation) in all the visible rows of the table.
4. Add a *Custom script* before the *For each* loop and add the relevant code inside it. This code will return all the occurrences of an element in a list during runtime.
   In the sales order creation:
   - Add the following code inside the *Custom script*.

     ```
     return $.[app].screens.[screen].elements.[element].getItems();
     ```

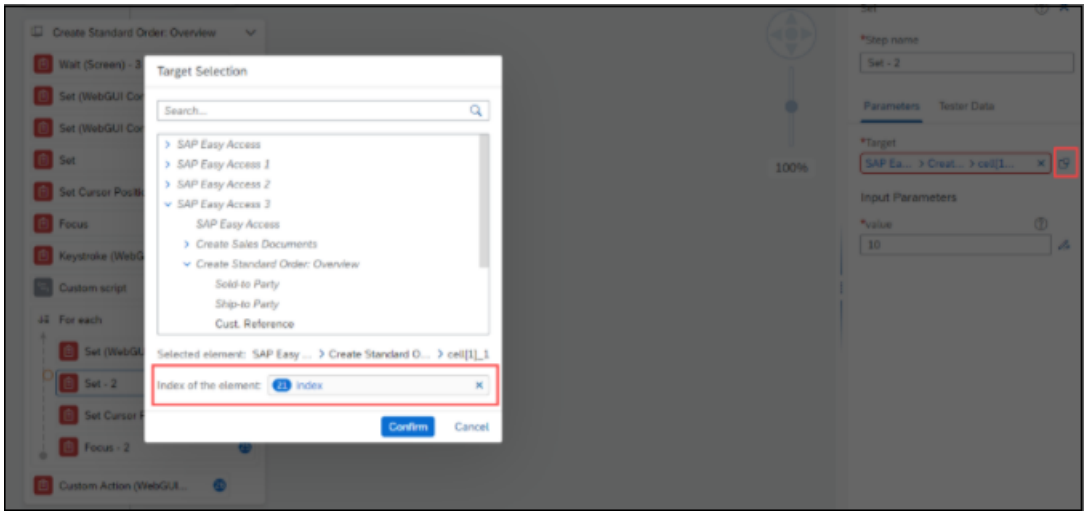
   - Replace the placeholders with the appropriate application, screen, and element name.
   - The element used in the code mentioned above can be any one of the collection elements you have created (*cell[1]* or *cell[1]_1*).
   - Here, the *cell[1]* element represents the material column in the *Custom script*.
5. Create an output parameter to store the list of elements returned by the *Custom script*.



6. Assign the output parameter from the *Custom script* to the *For each* loop as a looping list.
   In the sales order creation, you must assign the output parameter *material_cells* as a looping list.



   You can see that the *For each* loop provides two *Loop Parameters* which you can use in the steps inside the *For each* loop.
   In the sales order creation, *currentMember* loop parameter contains the element from the materials_list for that iteration and *index* loop parameter contains the index for the iteration (starts from 0).
7. Use the *index* parameter to update the target index for the steps inside the *For each* loop.



8. Click *Test*. For more details about testing an automation, refer to the Test an Automation [page 282] section.

Results

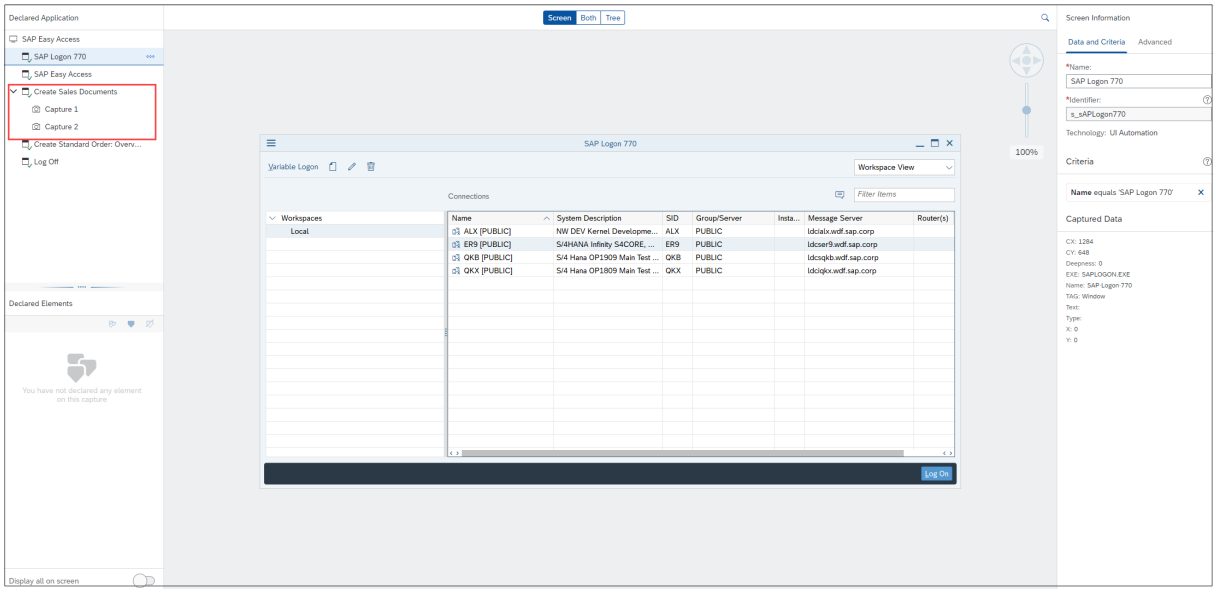
The bot will fill all the visible rows with the same set of values even though the count of the visible rows can vary from system to system. Therefore, you have executed your automation successfully.

3.3.2 Handle Multiple Screens with the Same Criteria in SAP WinGUI Recorder

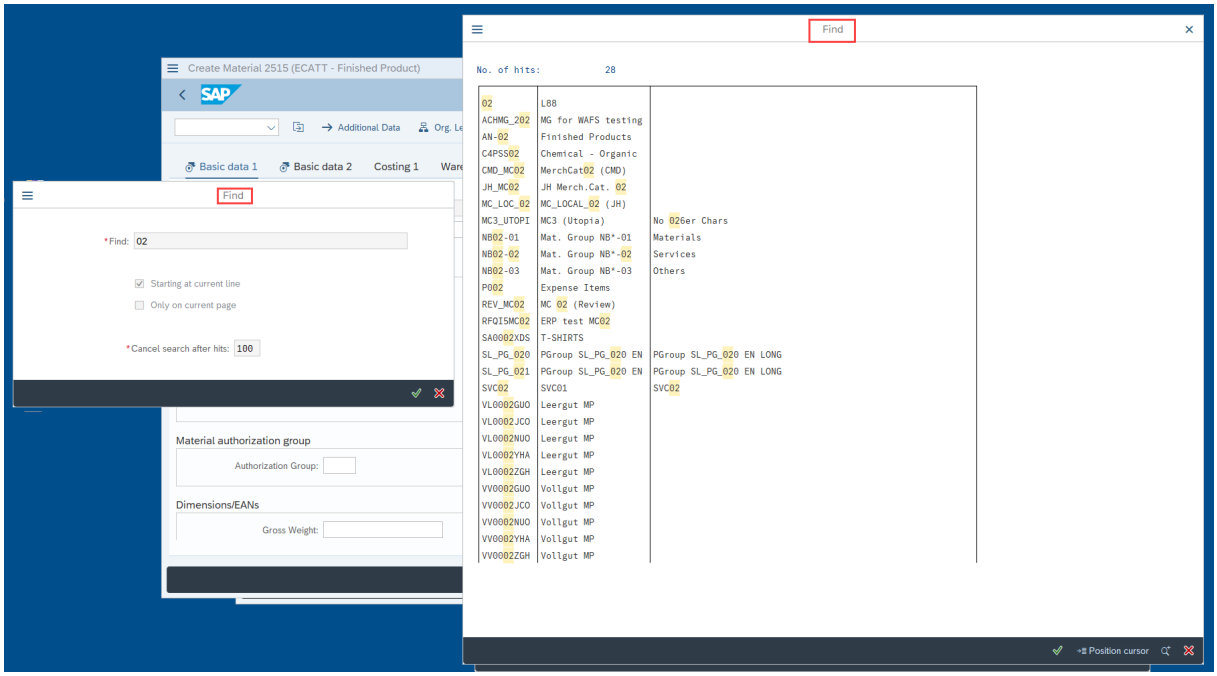
This section describes how to handle multiple screens with the same Criteria in SAP WinGUI recorder.

> **i Note**
>
> To know more details about declaring elements, refer to the Declaration Phase [page 101] section.

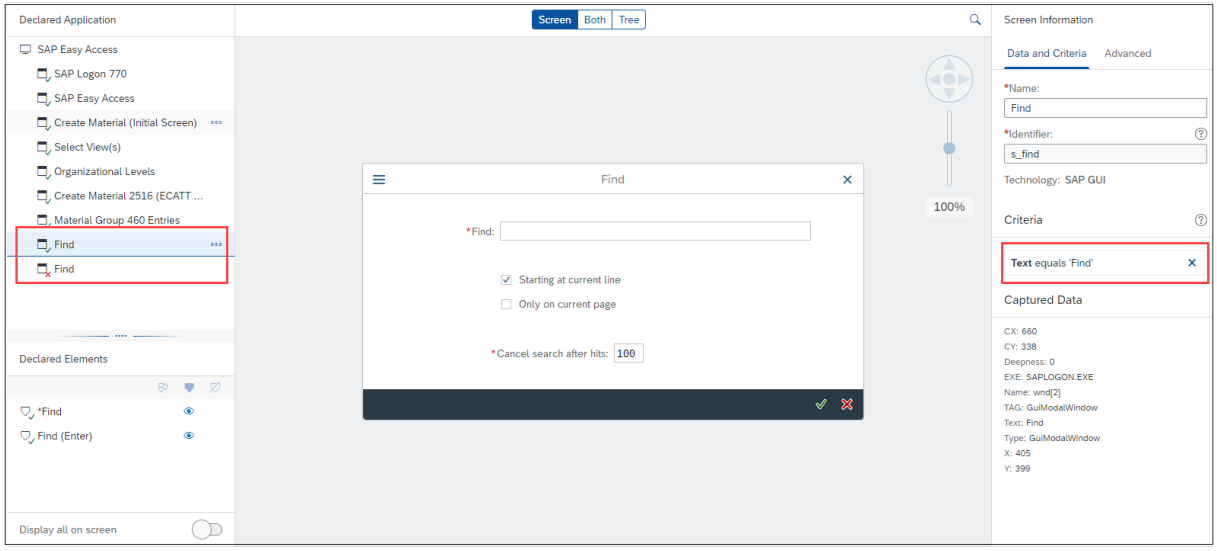
In most of the cases, all the screens with the same title (criteria) will be grouped together as multi-capture by the SAPWinGUI Recorder.



In some cases, two or more screens of the same title may appear simultaneously during the recording process. For example, refer to the below screen.

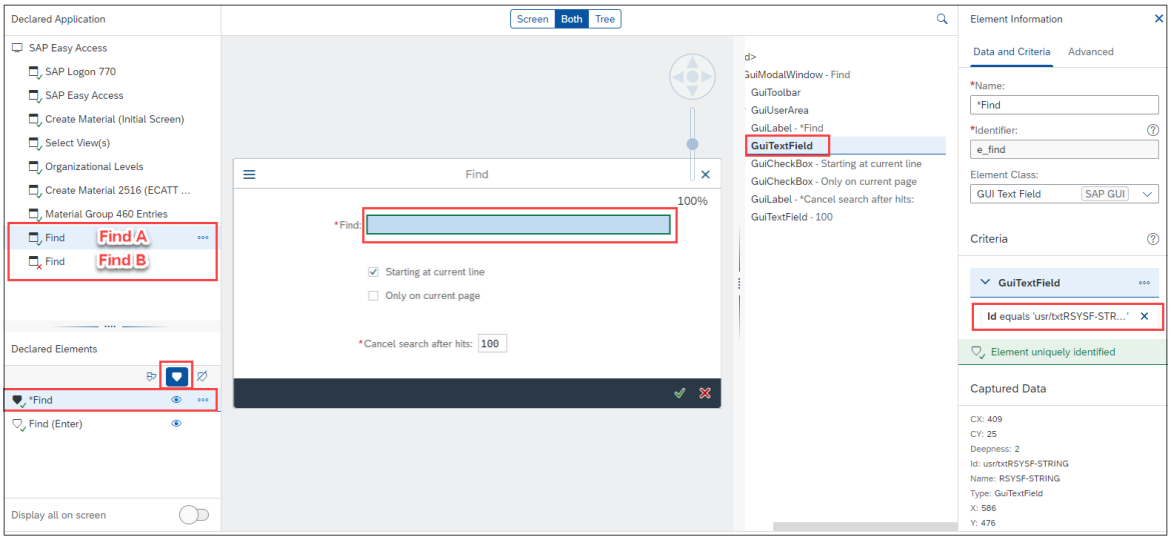


To recognize (LOAD) all screens simultaneously during runtime, the recorder will choose to declare them as separate captures.



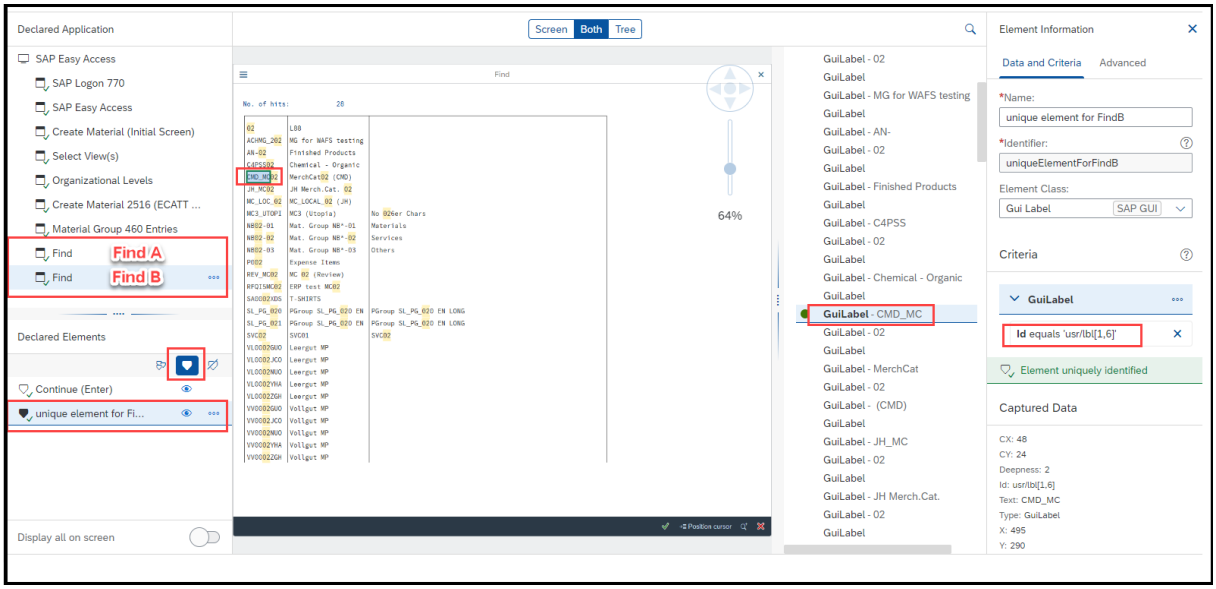
Now, the bot works fine as the captures are declared in a proper sequence in the Automation. To make recognition of the screens more robust, the following steps are recommended:

1. In the above scenario, we have two pages with **Find** as the criteria (Title). Let's call these two pages, **Find A** and **Find B**.
2. First, take the screen **Find A** and try to identify a unique element in the Tree (DOM) by comparing it with **Find B**'s Tree and add that as a **Must Exist** element.



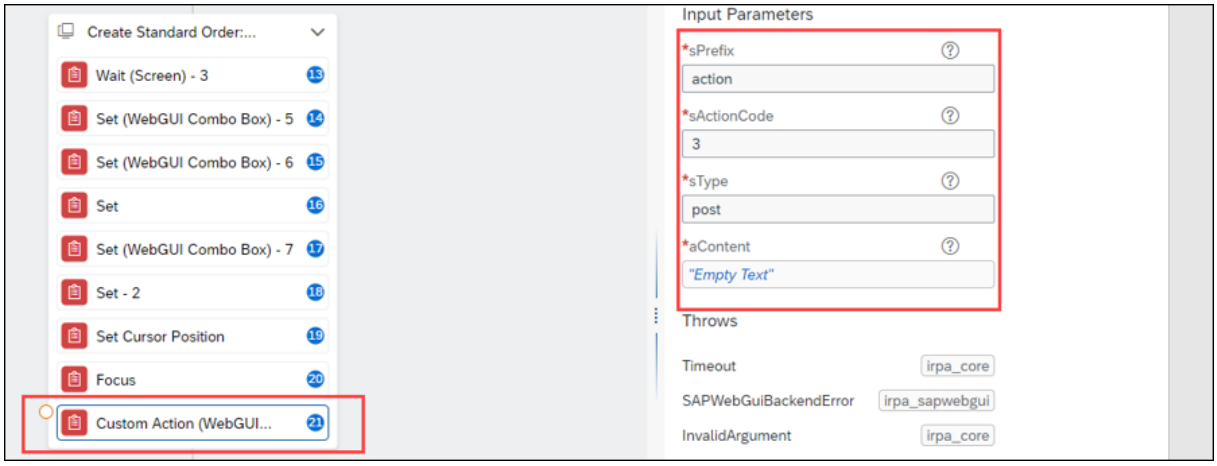
In the above image, in **Find A**, the text field declared as a **Must Exist** element is unique to that screen as the element with the same ID is not found in **Find B** after the tree comparison.

Follow the same procedure and identify a unique **Must Exist** element for **Find B** in comparison to **Find A**.



3.3.3 Edit a Custom Activity in Generated Automation

While recording application of any technology, if you perform a certain action that is not available in the SDK, a *Custom Action* appears in the automation.



The custom activity helps you to execute the automation without missing out on any recorded action.

By default, the *Input Parameters* of the custom activity is not editable as these parameters are mostly technical and therefore, it won't be of much use to you.

However, you must change the parameter of the custom activity in some scenarios. For example, if the *Custom Action* step is for setting a value in a text field, you must change the value to be set in that text field.

To edit the parameters in the custom activity, you must perform the following steps:

- Drag and drop a new empty custom activity on the same *Element/Screen* target and move the activity next to the existing custom activity.

> **Note**
>
> In any technology, the *Element* or *Screen* type will have its own custom activity.

- Copy the parameters from the existing custom activity to the new custom activity and edit the parameter as per your need.
- Delete the existing custom activity.

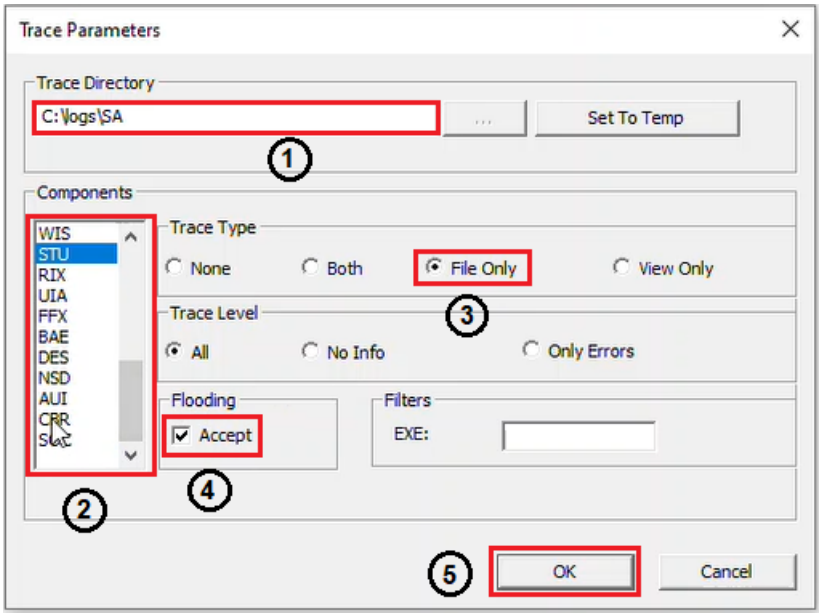
3.4 Enable Logs for Recorder in TraceViewer

Context

You need to open the TraceViewer tool and enable the *STU* and *CBR* components to capture logs for the recorder. Once you enable the TraceViewer, you will be able to see the logs for the recorder. For more information on TraceViewer, see Using TraceViewer.

Procedure

1. Enter the trace directory path in the *Trace Directory* (1) input box.
2. Select the *STU* component trigram from the *Components* (2) list box.
3. Check the *File Only* (3) radio button under *Trace Type*.
4. Check the *Accept* (4) checkbox under *Flooding*.
5. Select the *CBR* component trigram from the *Components* list box.
6. Check the *File Only* radio button under *Trace Type*.
7. Check the *Accept* checkbox under *Flooding*.
8. Click *OK* (5) to close the *Trace Parameters* dialog box.



> **i** Note
>
> To collect the logs, make sure that the Trace Viewer program is open during your trace session in the recorder. However, you can minimize the program.

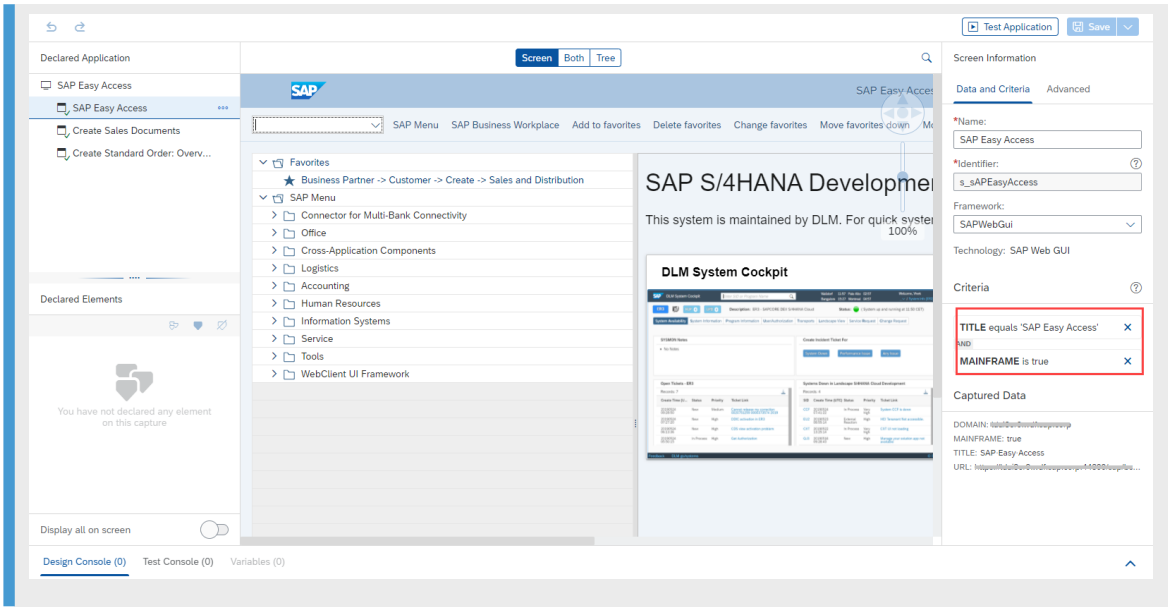
3.5 Recorder for Web GUI

You can use the Recorder to automate complex workflows easily with SAP Web GUI system.

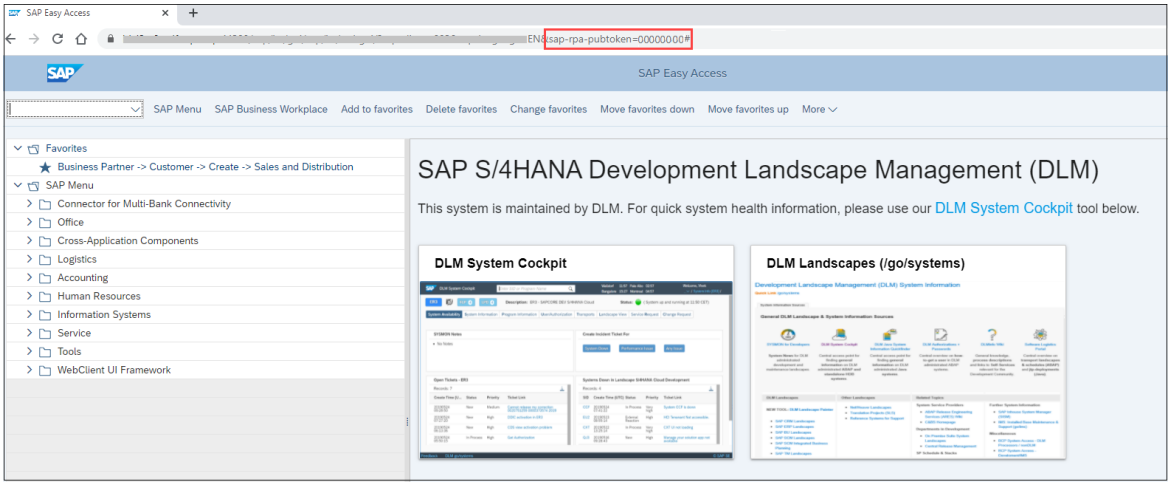
More Information

1. **Manual Capture**
   You can only add new screens to the exported recording of a WebGUI application. Currently, capturing WebGUI application screens manually from scratch is not supported.

   > **i** Note
   >
   > By default, *TITLE* and *MAINFRAME* are added as criteria for recorded and manually captured screens.



2. **URL Criterion**
   Do not add URL as a criterion for a screen. The screens captured manually across multiple SAP WebGUI systems have their own unique URLs.
3. **SAP Pub-token**
   The recording and manual capture is not possible for an SAP WebGUI application if the URL of the application contains a pub-token.



> **i Note**
>
> Remove a pub-token parameter from a URL and refresh an SAP WebGUI application to record or manually capture the screens.

Related Topics

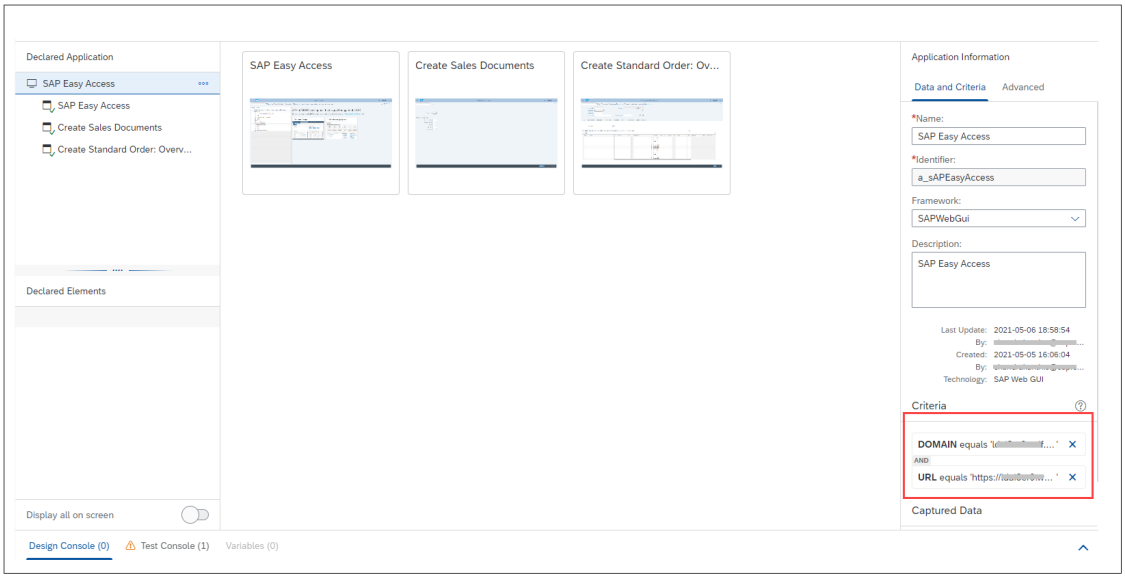
- Prerequisites [page 42]
- Best Practices [page 68]

- Limitations [page 69]
- Overview [page 38]
- Automatically Capture Applications with the Recorder [page 45]
- Enable Logs for Recorder in TraceViewer [page 65]

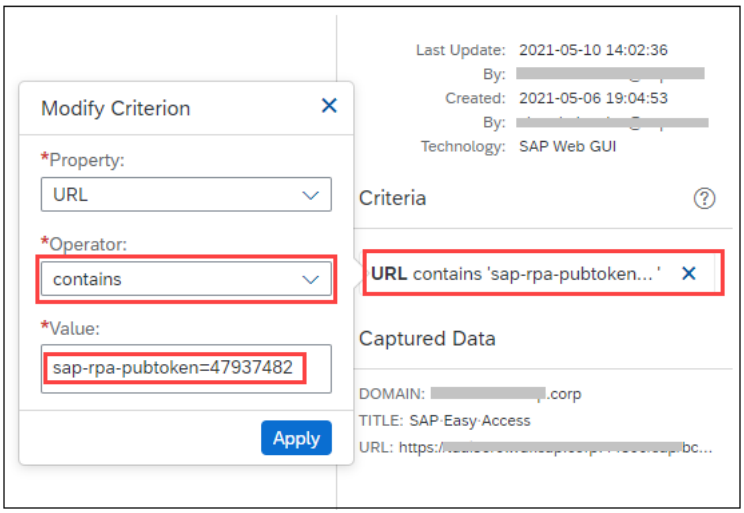
3.5.1 Best Practices

The following is the best practice we recommend in managing recording of workflow using the Recorder.

- You can record a workflow in one system and then you can run an automation in another system. To achieve this, you must change the URL in the Cloud Studio.
  To change a URL, follow the below procedure:
  Application
  1. In the Cloud Studio, select the to be updated application. By default, *DOMAIN* and *URL* are added as criteria.



  2. Remove *DOMAIN* from the *Criteria*.
  3. Update the URL and retain only **sap-rpa-pubtoken** and change the operator *equal* to *contains*.



Automation (Start Web Page Activity)
The Start Web Page activity opens the appropriate SAP WebGUI system in which the automation is performed in the web browser. This activity fetches the URL as the input parameter: you can then use variables to dynamically provide an URL during the execution of the automation.

1. Add sap-rpa-pubtoken to a *Start Web Page* activity URL.

   > **i** Note
   >
   > Ensure to add the same **sap-rpa-pubtoken** as the application.

   For example, consider a URL used in runtime as,
   https://abcd.abc.sap.corp:00000/sap/ab/gui/sap/its/webgui/?sap-client=000&sap-language=EN#
   Add the **sap-rpa-pubtoken** to the URL
   https://abcd.abc.sap.corp:00000/sap/ab/gui/sap/its/webgui/?sap-client=000&sap-language=EN&**sap-rpa-pubtoken=47937482#**

- It is recommended to start recording on an SAP WebGUI window opened in a new Chrome window. Do not use the SAP WebGUI window that is opened when the BOT/automation is running because its URL has a pub-token which prevents the recording.
- After the automation is completed, you can close the SAP WebGUI page that was opened by the automation.

3.5.2 Limitations

- **Manual Capture of WebGUI Application**
  You can manually add new screens only to the application generated from the recording of a WebGUI application. Capturing WebGUI application screens manually from scratch is not supported.
- Login to SAP WebGUI via SSO is supported, but the login via user and password is not.
- A headless browser (a browser without a graphical UI) is not supported for a bot execution.

3.6 Recorder for Win GUI

You can use the Recorder to automate complex workflows easily with the SAP WinGUI system.

This section provides tips and details to help you to understand and use the recorder appropriately with the SAP WinGUI system.

More Information

1. **Default Capture Parameters**

The default capture parameters in Win GUI recorder are:

1. Capture Mode = Print Window
2. Max Capture Height = 16000 Pixels
3. Max Deepness = 0
4. Max Siblings = 0

These values are fixed for recording scenario and cannot be changed after recording.

2. **Win GUI Recorder Activities**
   The activities introduced for Win GUI Recorder are:
   1. Open Connection
      SAP WinGUI recorder uses Open Connection activity to open connection to SAP GUI system when running an automation. This activity is generated automatically when recording is completed. If you want to connect to another SAP GUI system, then change the connection name parameter of this activity with connection name as shown in the SAP Log On page. Example, Q43 [PUBLIC].

      > **i** Note
      >
      > The connection name must be the same as what you see on your SAP Logon pad.

   2. Close Connection
      Closes connection to SAP GUI system that was opened by Open Connection and Open Connection By String.
   3. Click Sync
      Recorder generates "Click Sync" activity on GUI Button to perform click synchronously. If Click Sync operation initiates a time-consuming action, then replace with Click.
3. **Capture Meta-data**
   To capture meta-data faster, fewer properties are captured during recording.
   Recorder captures meta-data such as Id, Text, Type, Name, ScreenLeft, ScreenTop, Width, and Height properties for a page or item depending on the control type.
   If other properties like IconName, Tooltip, LeftLabel, RightLabel, Changeable, and ContainerType are required for recognition, then the page must be recaptured in cloud studio after recording artifacts are exported to Cloud Studio.
4. **Two or More Screens of the Same Name**
   When two or more screens of the same name are added to the SAP GUI Application after exporting the recording, add a unique component from the DOM for each page as a Must Exist element to aid proper recognition.
   Initially, with the generated automation, each page sharing the same title will work and not have issues in runtime because of the sequence of the screen appearance in the bot. When the automation is scaled, there could be recognition issues if handling multiple screens with the same criteria process is not in place.
   To know more details about handling multiple screens with the same criteria, refer to the Handle Multiple Screens with the Same Criteria in SAP WinGUI Recorder [page 61] section.
5. **Password Security**
   In SAP GUI recorder, if any password field is encountered then its value will be recorded as '********'.
   For the automation to work, the bot developer needs to select the respective step in the automation and change the value of the input parameter from '*******' to your actual password. You can set the value as a parameter and provide the password at runtime.
   Refer to the section Best Practices for Environment Variables of Type Password for best practices for password security.
6. **Session Busy**
   While running the automation, If Current Session is Busy... exception is encountered, user must increase the busyTimeOut value for Set Busy Wait Time activity in the automation.

The Set Busy Wait Time activity helps the SAP WinGUI bot to check if the session is busy before sending each action to the actual SAP GUI application.

Related Topics

- Prerequisites [page 42]
- Best Practices [page 71]
- Limitations [page 71]
- Overview [page 38]
- Automatically Capture Applications with the Recorder [page 45]
- Enable Logs for Recorder in TraceViewer [page 65]
- Handle Multiple Screens with the Same Criteria in SAP WinGUI Recorder [page 61]

3.6.1 Best Practices

The following is the best practice we recommend in managing recording of workflow using the Recorder.

- Until capture of a page is completed, do not perform any action on SAP GUI page that causes a round trip. If any round-trip action is performed on a page before the capture is completed, the capture fails and the actions are associated with the previously captured page.
  Undo actions associated with previous page and recapture the page.
- While Recording, do not initiate a screen capture when the Session is busy. The screen capture action might not be successful.
- Cloud Studio Timeout
  While the recording is in progress, the Cloud Studio may timeout for various reasons. So, before you start exporting the recording to Cloud Studio, check if the Cloud Studio has not timed out. If it is timed out, you must re-login (do not refresh) to the Cloud Studio and then export the recording.

3.6.2 Limitations

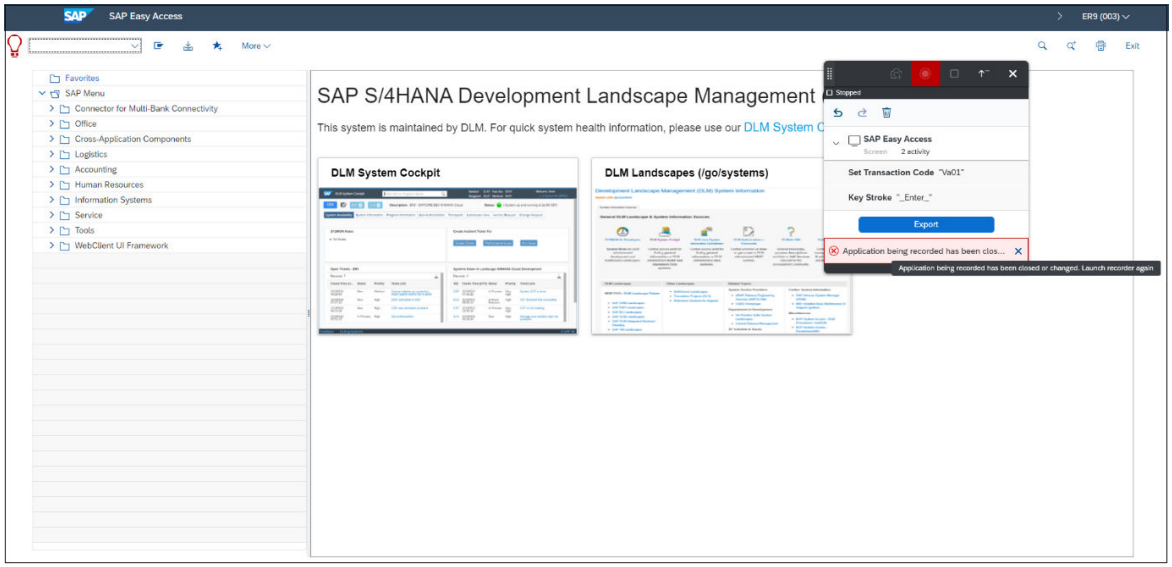
- While capturing a page, if SAP scripting does not respond within 7 seconds, a timeout error is generated internally and "wnd[0]" is displayed as title in the recording widget page.
  If such captures are seen in the widget, undo the capture and recapture the page.
- The SAP WinGUI recorder will not record UIAutomation pages if these pages appear during recording. For example, file upload.
  In this situation, use Win GUI file uploader. Alternatively, after artifacts are exported to cloud studio, capture such UIAutomation pages from Cloud Studio and modify automation by adding activities.
  This limitation exist as SAP Intelligent RPA does not support UIAutomation recorder.

3.7 Error Management

This section provides details about errors and recommendations to address the errors that may occur during recording.

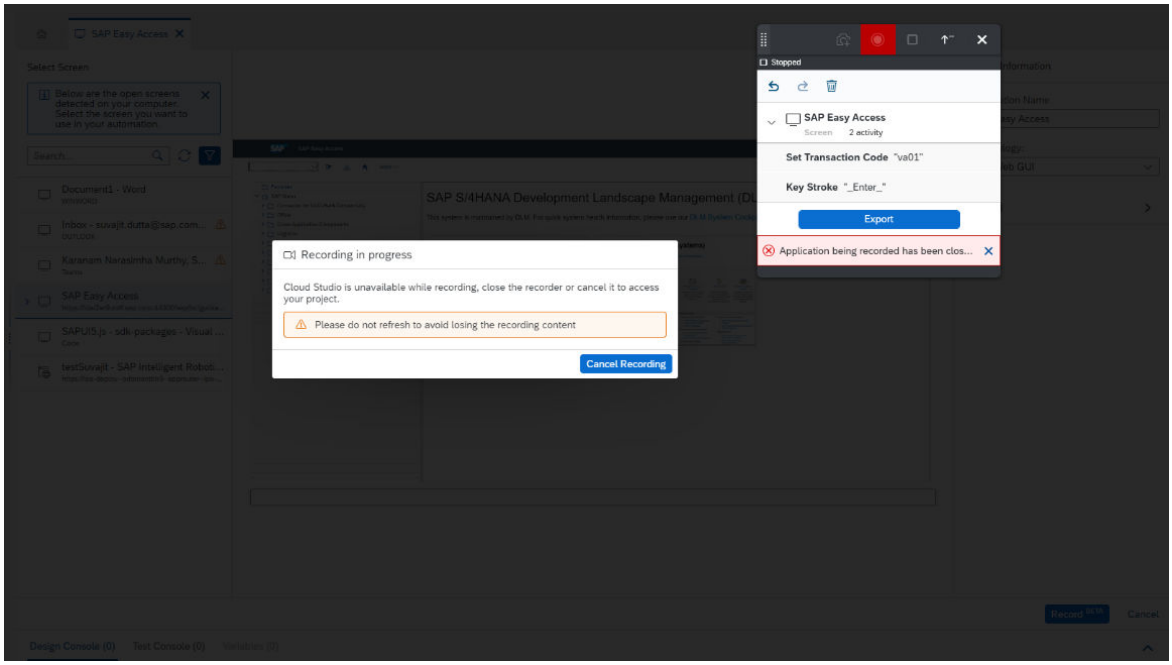
The error messages are same for all technologies. The error message examples shown below are captured using SAP WebGUI technology.

- When you launch the recorder and start the recording and if you refresh the SAP WebGUI screen in between the recording, then the Recorder shows an exception message.



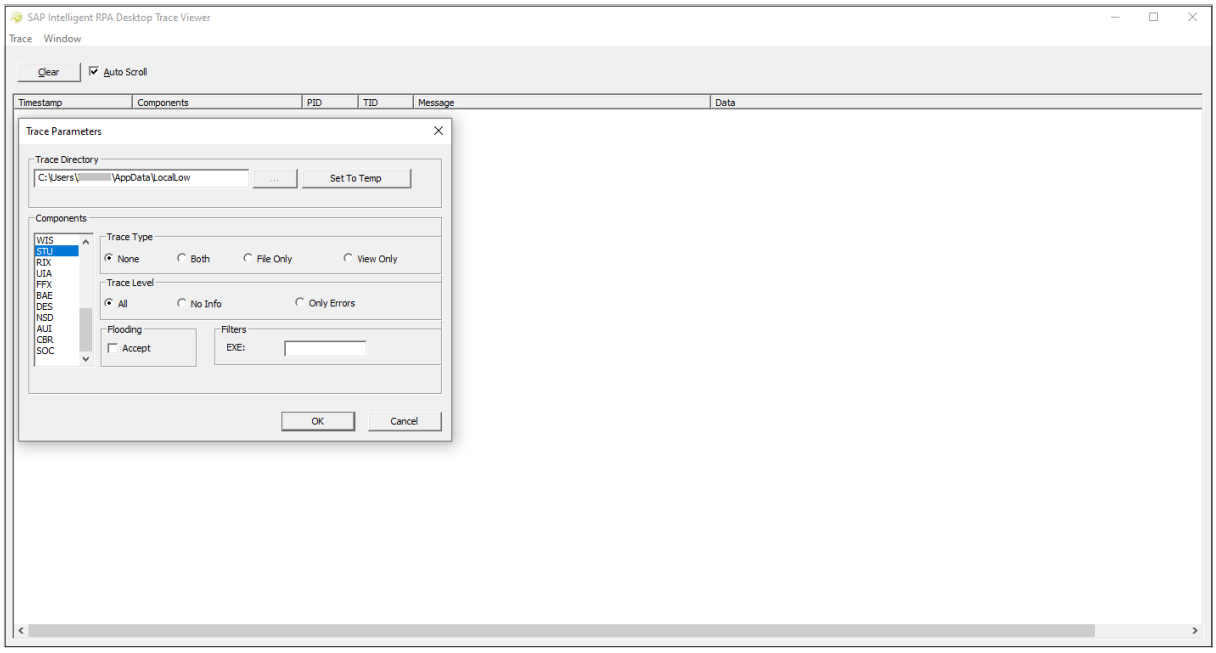
You must click the *X* button to exit the error pop-up.

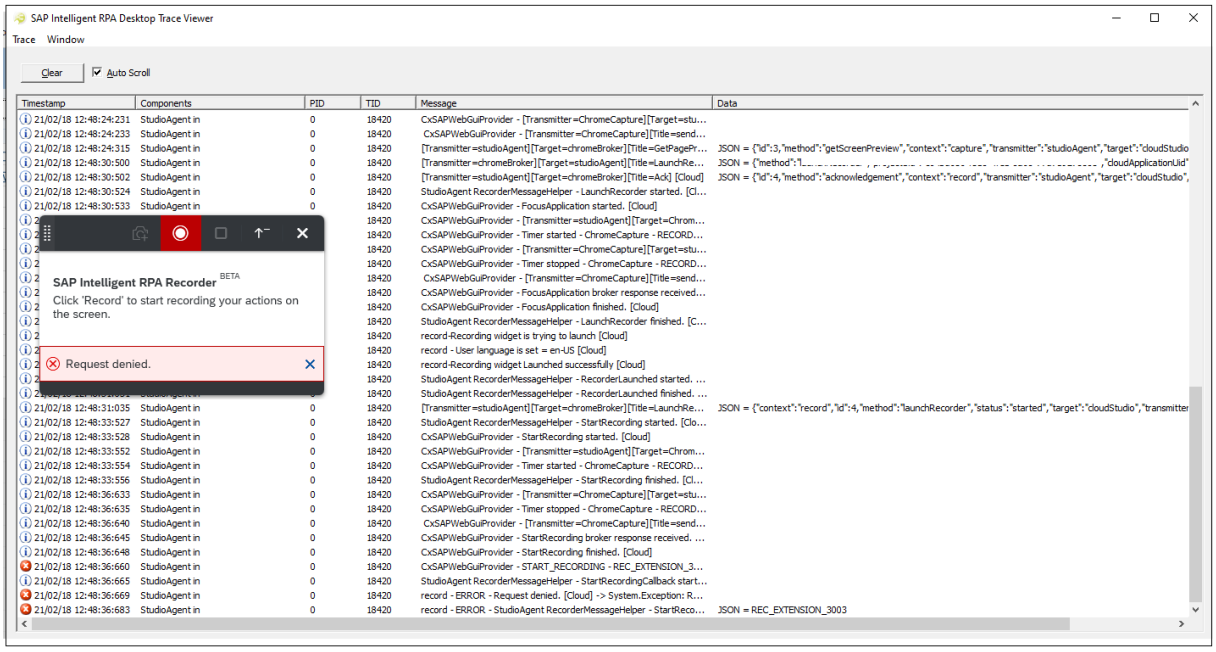
- When you launch the recorder and start the recording and if you close the SAP WebGUI screen in between the recording, then the Recorder shows an exception message.



You must click the *X* button to exit the error pop-up.

To get more details on the error, you can keep CxTraceViewer.exe by selecting STU option.





3.8 Best Practices

This section provides the best practices applicable for all technologies.

- Cloud Studio Timeout

While the recording is in progress, the Cloud Studio may timeout for various reasons. So, before you start exporting the recording to Cloud Studio, check if the Cloud Studio has not timed out. If it is timed out, you must re-login (do not refresh) to the Cloud Studio and then export the recording.

4 Capture Applications and Design Automations

Capture and Declare Applications [page 75]
The Cloud Studio artifact of type *Application* lets you capture an application running on your local machine, select and capture screens, and recognize UI elements on those screens for use in automations.

Design Automations [page 153]
Automations are composed of a succession of steps you build in the Cloud Studio with the automation designer. An *automation* can orchestrate multiple activities on different applications and screens available on a specific computer.

Trigger Automations in Attended Mode using the Project Pane [page 352]
The project pane allows you to design the agent systray menu directly from the Cloud Studio.

Alerts [page 357]
Alerts artifact allows you to define business events. Once an alert is created, it can be raised from an automation. You can use the *Raise Alert* activity from the SDK Core to raise an alert from your automation.

Manage Files [page 360]
In Cloud Studio, you can create, import, and edit a file.

Import Files [page 367]

Business Activity Monitoring (BAM) [page 368]

4.1 Capture and Declare Applications

The Cloud Studio artifact of type *Application* lets you capture an application running on your local machine, select and capture screens, and recognize UI elements on those screens for use in automations.

Capturing and declaring applications means identifying the applications you want to control in an automation. Even though applications are based on different technologies, they share many features in common.

You start by locating the screens that you want to use. Within a screen, you then look at the UI elements you want to control - such as input fields, output fields, and buttons.

Another key task is declaration, which means choosing the recognition criteria for the application, screens and UI elements that will be used at run time to correctly identify them.

When you have captured all screens and declared the application, screens and UI elements, you can then test the application before including it in an automation.

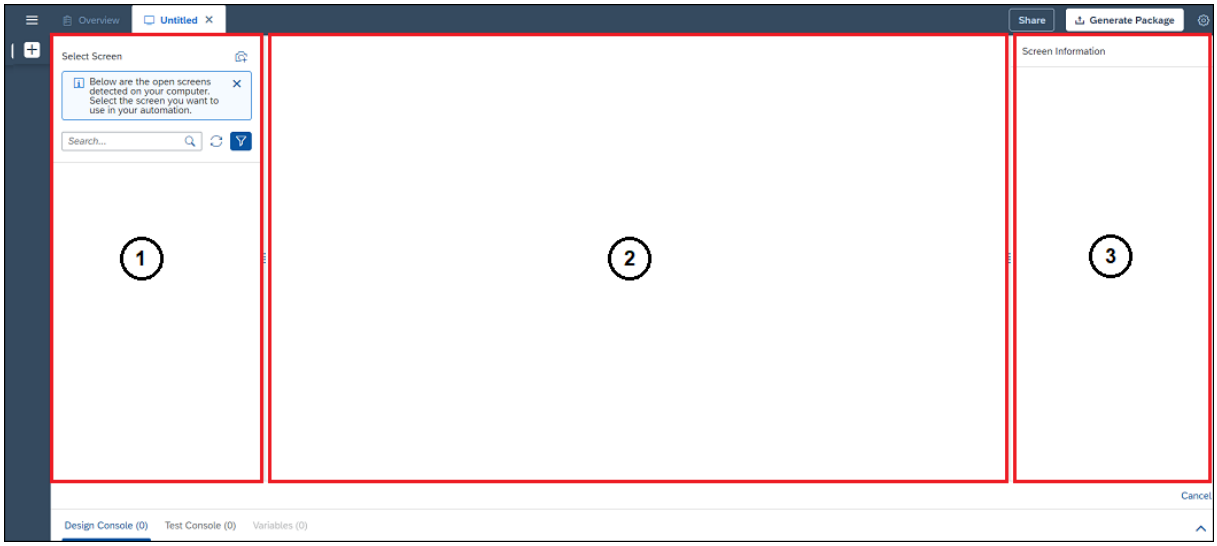
> **i Note**
>
> You can't create an application in a project if the project already contains a Desktop Package.

4.1.1 Overview

The application editor allows you to capture an application running on your local machine, select and capture screens, and recognize UI elements on those screens for use in automations.

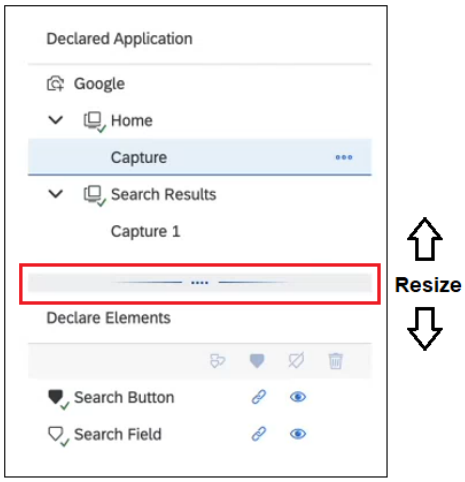
The application editor consists of three parts:

- picker panel on the left side (1)
- main area in the center (2)
- screen information on the right side where criteria are defined (3).



Resizing the Picker Panel

The application editor's picker panel splits into two sections - *Declared Application* and *Declare Elements*. These two sections can be resized so that you can give more space to any section. To resize, move the splitter up or down accordingly as shown in the following screenshot.



4.1.2 Capture an Application

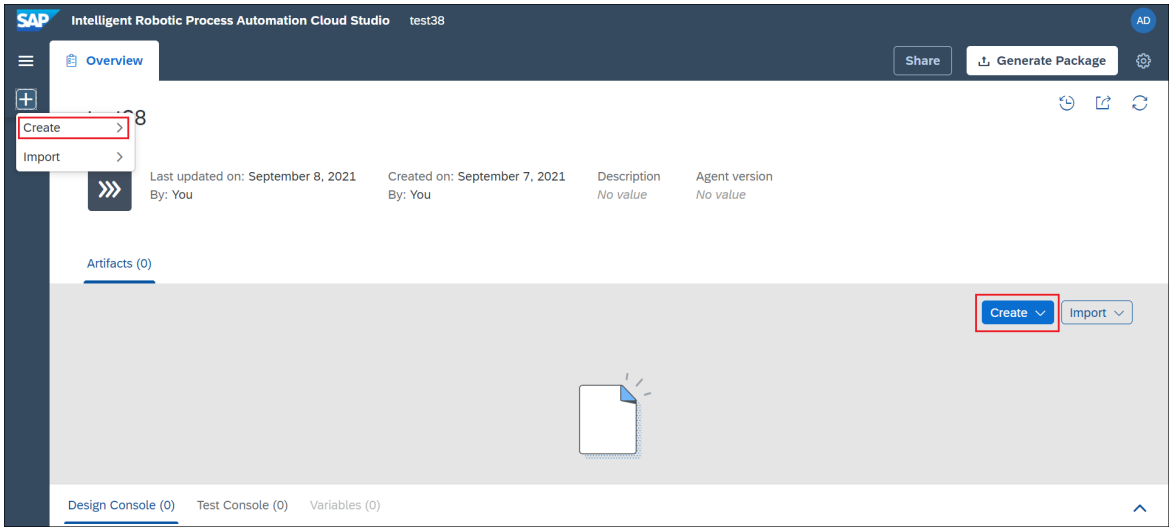
Prerequisites

Capturing requires SAP Intelligent Robotic Process Automation Factory and the Cloud Studio to be running in a Chrome browser. It also requires the browser extension installed and activated, and the Desktop Agent configured to use with the target tenant.

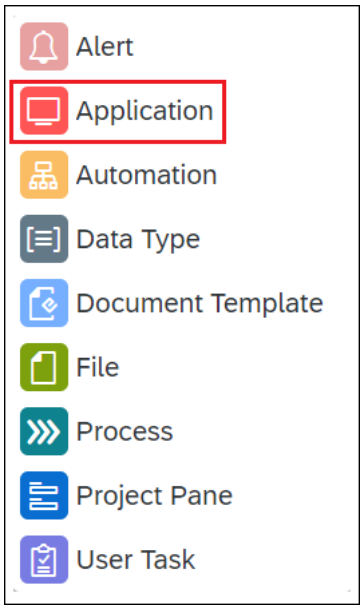
To capture an application, you start by capturing one of its screens. An application can be a browser app, a Windows application, an SAP application, and so on.

Procedure

1. In the Project Explorer, click + and then click *Create* in the left panel or click *Create* in the Project Explorer main panel.

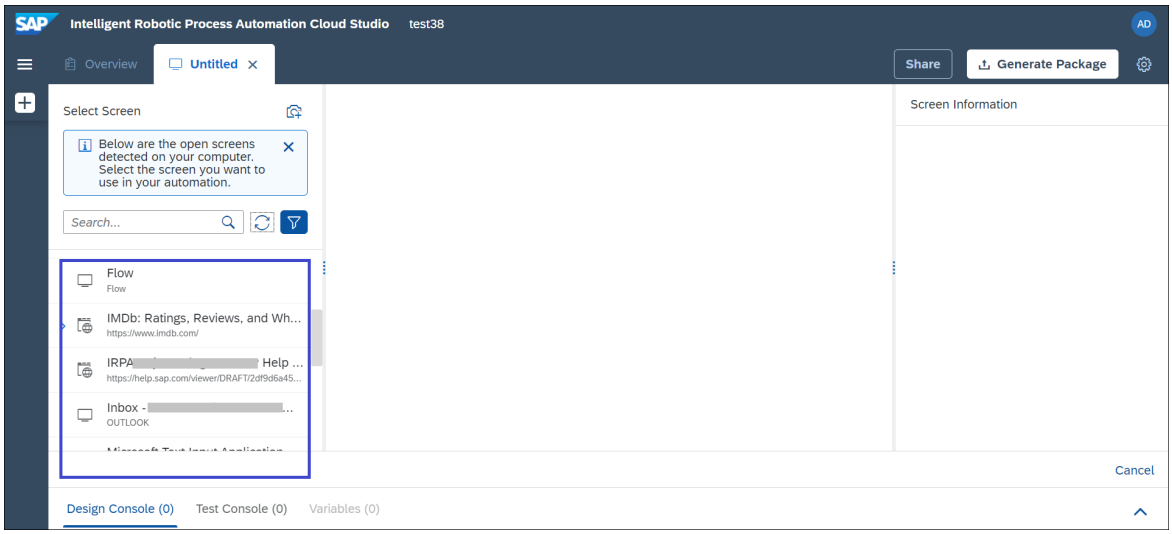


The artifact menu is displayed.



2. In the artifact menu click *Application*.

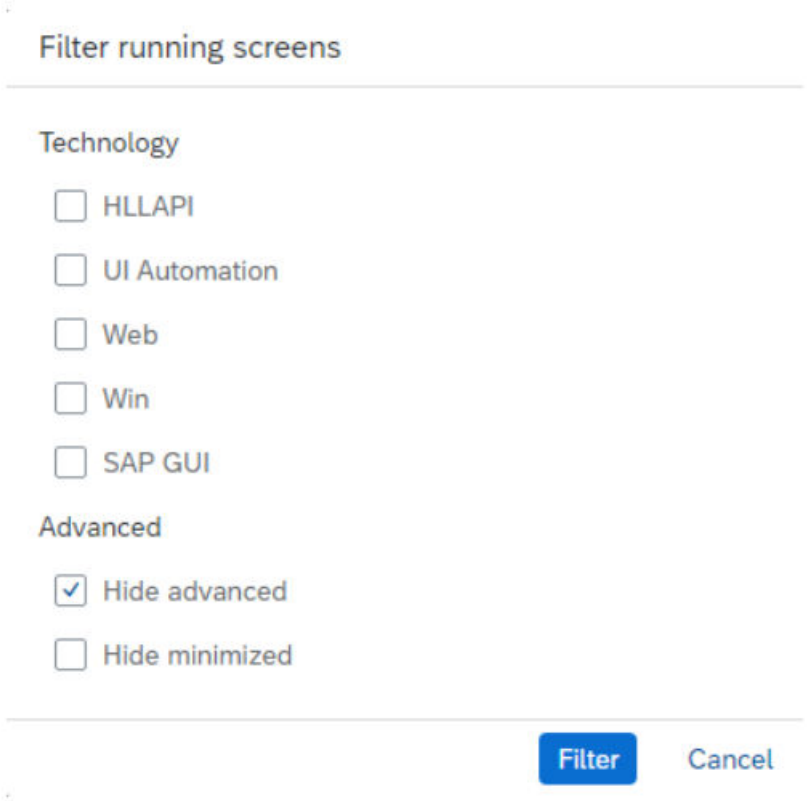
A new tab labeled **Untitled** opens. Later, this label will change to the name of the captured application. The system starts detecting the applications and their screens currently running on your local machine. When it's done, you'll see a list of screens in the picker panel on the left.



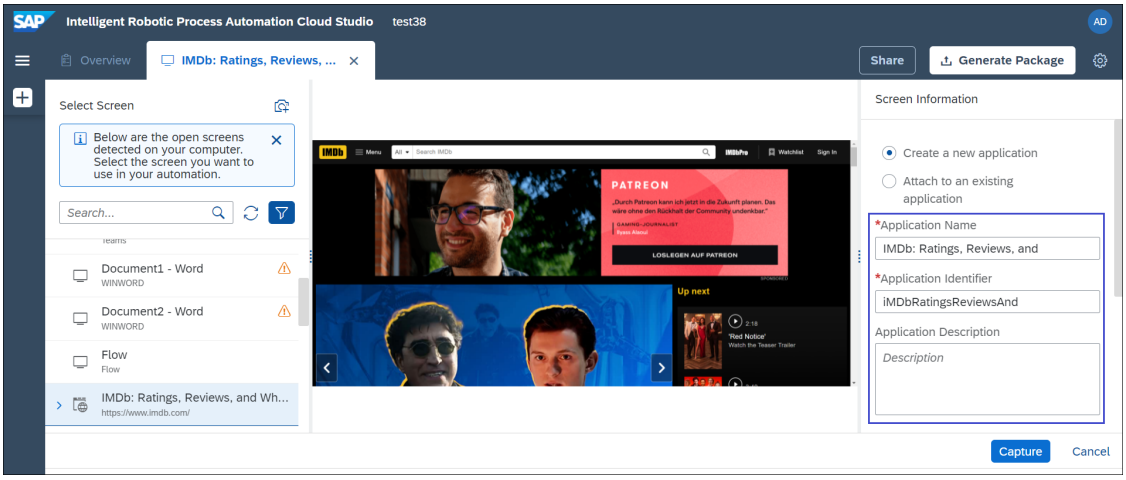
The icons in front of the screen names indicate the technology, such as Web.

You can refresh the list any time via ↻ - for instance, when the applications currently running on your desktop have changed.

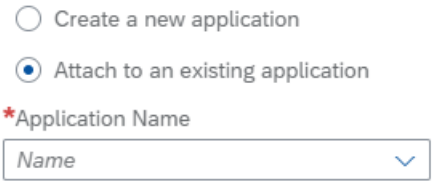
3. Choose the application you want to capture from the list of screens.
    1. Enter a term in the search field to find your application faster.
       Your search term is compared with the screen names, executable names, and where relevant, URLs of the detected open screens.
    2. If the list is very long, filter it by technology or other criteria via ▽ .



A preview image of the screen is displayed in the capture area. The application's name is automatically entered in the field in the right-hand panel. You can then manually change the name and the identifier, and add a description.
The tab's label changes to the application name you entered.
Optionally, you can also change the name and the identifier of the screen.
If an error appears, refresh the running application screen you want to capture.



4. **Optional**: If your project already contains one or more applications, you can attach the screen to an existing one by choosing its name from the dropdown list.



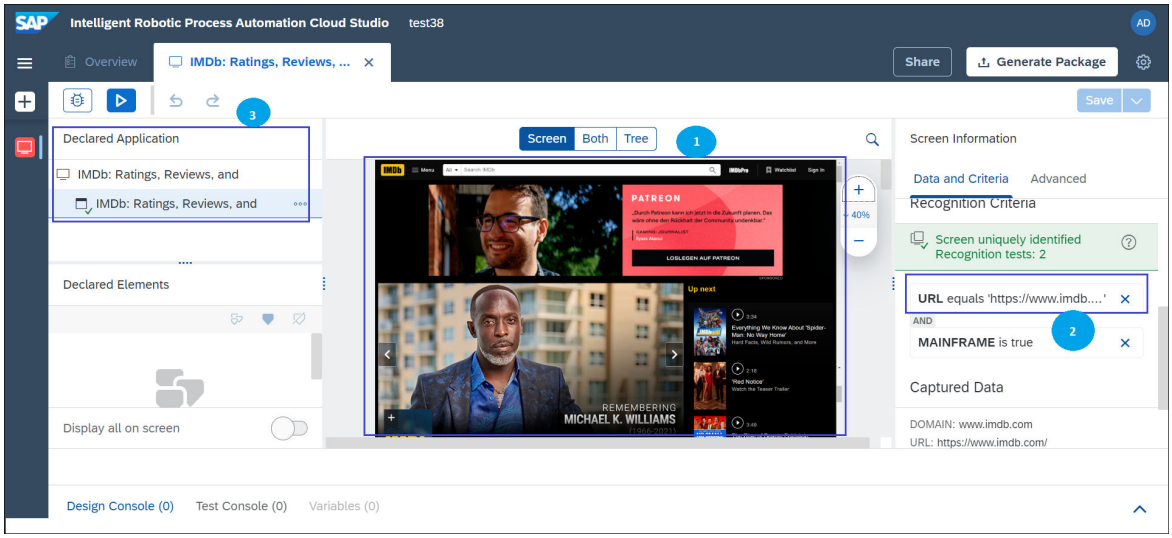
5. **Optional**: Select the technology by choosing from the dropdown list.
   Cloud Studio offers various technologies for capturing applications. For example, Web is used to capture a browser application, Win for Windows applications. UI Automation technology can be used to capture a wide range of applications.

   > ⚠ Caution
   >
   > We recommend keeping to the detected technology in most cases.

6. Define Capture Options.
   For details, see Capture Options [page 87].
7. Click *Capture*.

   > i Note
   >
   > If you haven't imported SDK packages in your project yet, the mandatory SDK packages are automatically downloaded.

   You've now captured an application by capturing one of its screens. Once the screen has been captured, the image (1) in the capture area changes and becomes interactive. You can zoom in and out. The application recognition criteria (2) - used to identify the application, screens and elements correctly when your automation runs - are automatically selected. You can change them when you declare the application. In the left-hand navigation panel you see a cascading list (3) starting with the declared application. As you capture and declare screens and other entities, they appear in this list.



8. Save your work.

   > i Note
   >
   > Delete an application in the Project Explorer panel via 🗑.

Next Steps

You can now add as many screen captures as you want to your declared application. You also need to declare your application, its screens, and the elements on the screens (like buttons and input fields) that you want to use in your automation project.

Related Information

Capture another Screen [page 81]

Declare an Application [page 101]

Declare a Screen [page 104]

Declare an Element [page 110]

4.1.2.1 Best Practice

When you start a capture with Chrome and Firefox in Cloud Studio, make sure the window zoom factor is the same on every screen (whatever the browser zoom factor).

If you have some issues capturing screen elements, you may be able to solve the problems by setting the zoom factor to 100%. Go to the Scale and layout settings of your laptop and make sure that your zoom factor is set to 100%, then restart your session to apply the change.

In the Capture window, a warning message lets you know if the zoom level of the target page is not set to 100%.

The following issues can be avoided if the zoom factor is set to 100% in the runtime phase:

- Incorrect position of highlighting of elements.
- Incorrect coordinates during a mouse click event and its failure.

4.1.2.2 Capture another Screen

Prerequisites

You've captured an application and at least one screen as described in Capture an Application [page 77].

Context

You might need to capture the same screen multiple times, for example to capture different popup areas.

Procedure

1. From a project in Cloud Studio, click on *Applications* in the *Project Explorer* if necessary, and choose the application for which you want to capture another screen.
2. Click ••• next to the declared application and click *Add Screen* to add a screen capture.
3. Choose the screen you want to capture from the list of open screens detected.

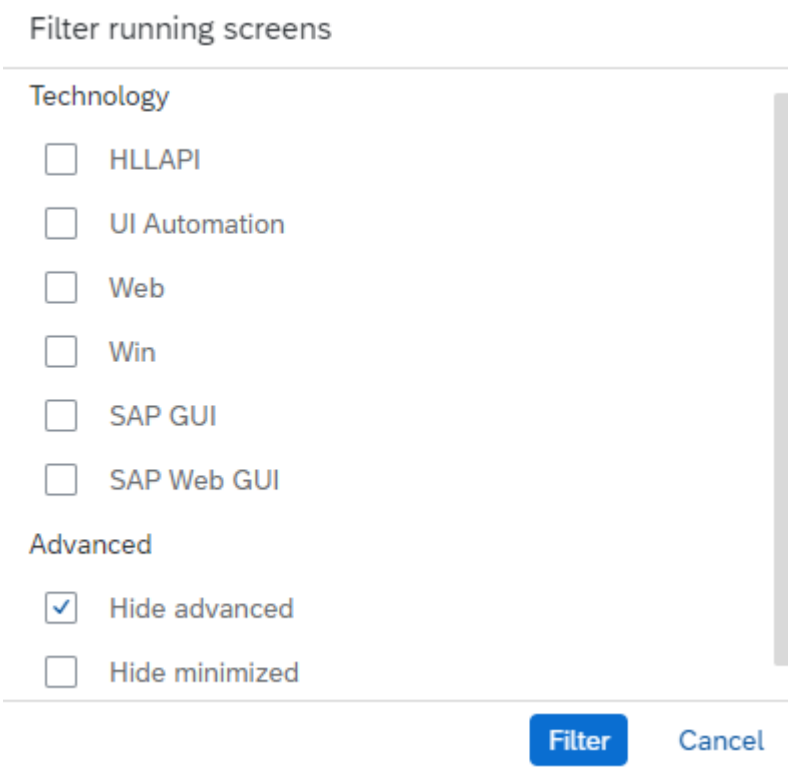
   The icons in front of the screen names indicate the technology, such as Web.

   You can refresh the list any time via ↻ - for instance, when the applications currently running on your desktop have changed.

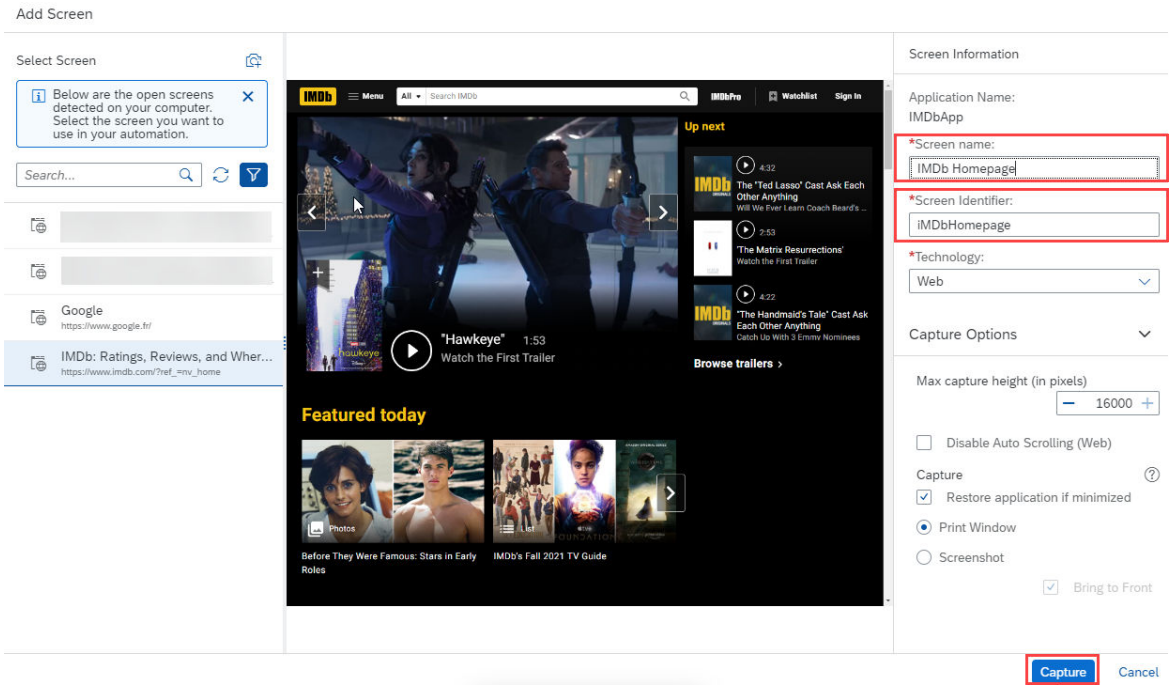
   a. **Optional:** Enter a term in the search field to find your screen faster.

      Your search term is compared with the screen names, executable names, and where relevant, URIs of the detected open screens.

   b. **Optional:** If the list is very long, filter it via ▽ before choosing a screen.



A preview image of the screen is displayed in the capture area. The screen name and identifier are automatically entered in the field in the right-hand panel. You can change them manually.



4. **Optional:** Select the technology by choosing from the dropdown list.

   Cloud Studio offers various technologies for capturing applications. For example, Web is used to capture a browser application, Win for Microsoft Windows applications. UI Automation technology can be used to capture a wide range of applications.

   > ⚠ Caution
   >
   > We recommend keeping to the detected technology in most cases.

5. **Optional:** Define one or more *Capture Options*.

   For details, see Capture Options [page 87].

6. Choose *Capture*.

   You've now captured an additional screen for your application. Once the screen has been captured, the preview in the capture area changes and becomes interactive. You can zoom in and out.

   The new screen appears in the picker list on the left.

   If you have captured the same screen multiple times, a number starting with 1 is automatically appended to the names and identifiers of all but the first capture.

   Click the application in the picker list and you're shown tiles representing each captured screen. Click a tile to view the corresponding screen.

7. Save your work.

   > i Note
   >
   > Delete a screen by choosing the three dots after the screen name in the picker panel and selecting *Delete*.

Next Steps

Next, you need to declare the screen.

Related Information

Declare a Screen [page 104]

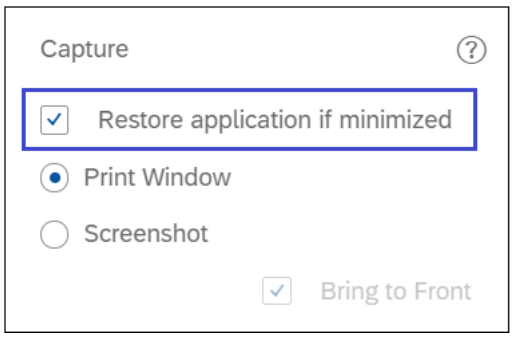
4.1.2.2.1 Warning Messages when Capturing a Screen

Context

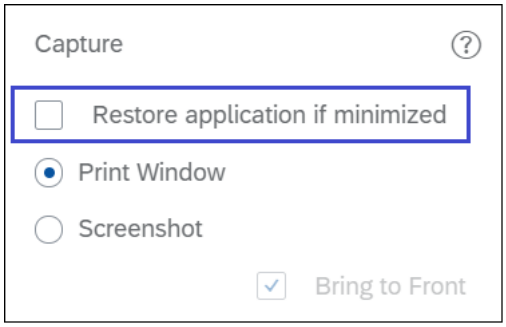
When browsing available screens in the *Search screen* panel, you select entries from the list. If the preview screen is empty during the application capture, several warning messages can be displayed. These warning messages explain why the screen cannot be captured.

List of Warning Messages

- **The screen is minimized and will be restored during capture.**
  This warning message is displayed when the target screen is minimized and the *Restore application if minimized* option is selected.

  If the *Restore application if minimized* option is selected, then the target application screen that is minimized will be restored while capturing.
- **The screen is minimized. Check the option to restore it on capture.**
  This warning message is displayed when the target screen is minimized and the *Restore application if minimized* option is **not** selected.



If the *Restore application if minimized* option is **not** selected, then the target application screen that is minimized will **not** be restored while capturing.

- **This screen is a frame of a web application and does not have a user interface to preview.**
  This warning message means that the screen is a frame of a web application that is not visible in the viewport.

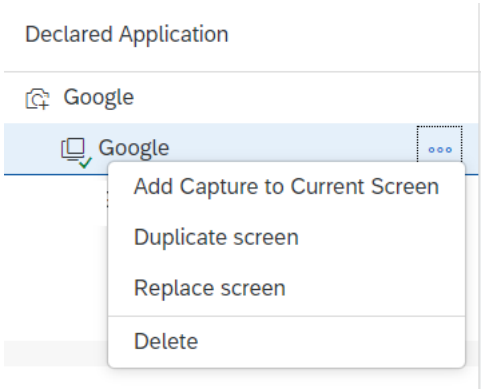
4.1.2.3 Duplicate Captured Application Screen

In the Cloud Studio, you can duplicate the captured application screens one or more times. The duplicate screen represents the contents of the original screen.

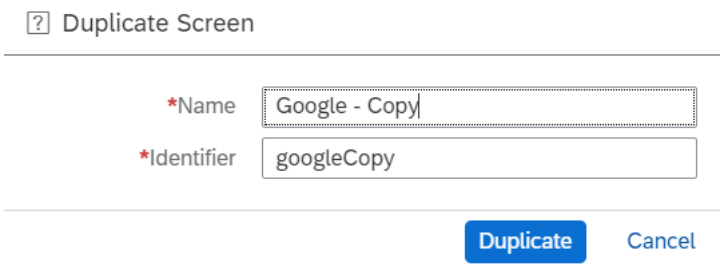
In the duplicated screen, you can declare new elements, or you can modify the existing declared elements of the screen.

To duplicate a screen, follow the below procedure.

1. In the IRPA Cloud Studio, under *Declared Application*, click the ••• icon of a captured screen.
   More Options are displayed.

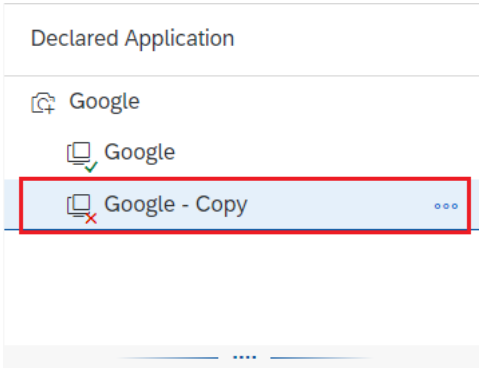


2. Click *Duplicate screen*.
   The *Duplicate screen* pop-up screen is displayed.



> **Note**
>
> If required, you can update the name of a screen.

3. Click *Duplicate*.
   The duplicate screen is created and displayed under *Declared Application*.

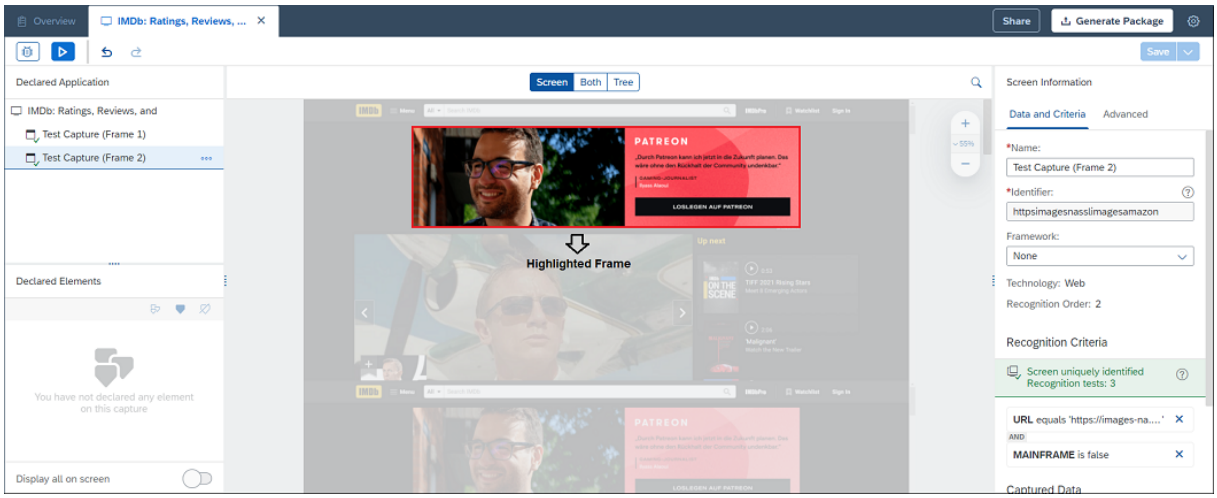


4.1.2.4 List the Application Frames

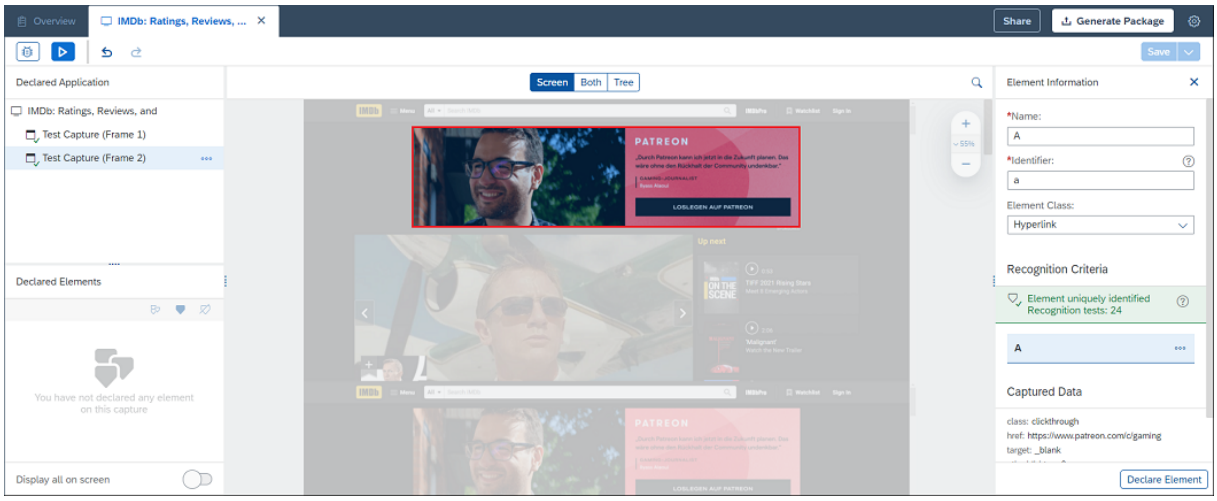
You can list all the application frames and subframes that appear on the screen. You can select the relevant frame of the captured screen. During an element declaration, the corresponding area of the selected frame is highlighted with respect to the rest of the screen. Therefore, you can easily distinguish between the different frames and find the different components that are selected on the screen.



In the following screenshot, you can observe that only the area of the selected frame is highlighted and the rest of the screen which is not part of the selected frame is faded. Thus, you can easily find the element available on the highlighted screen and select it accordingly.



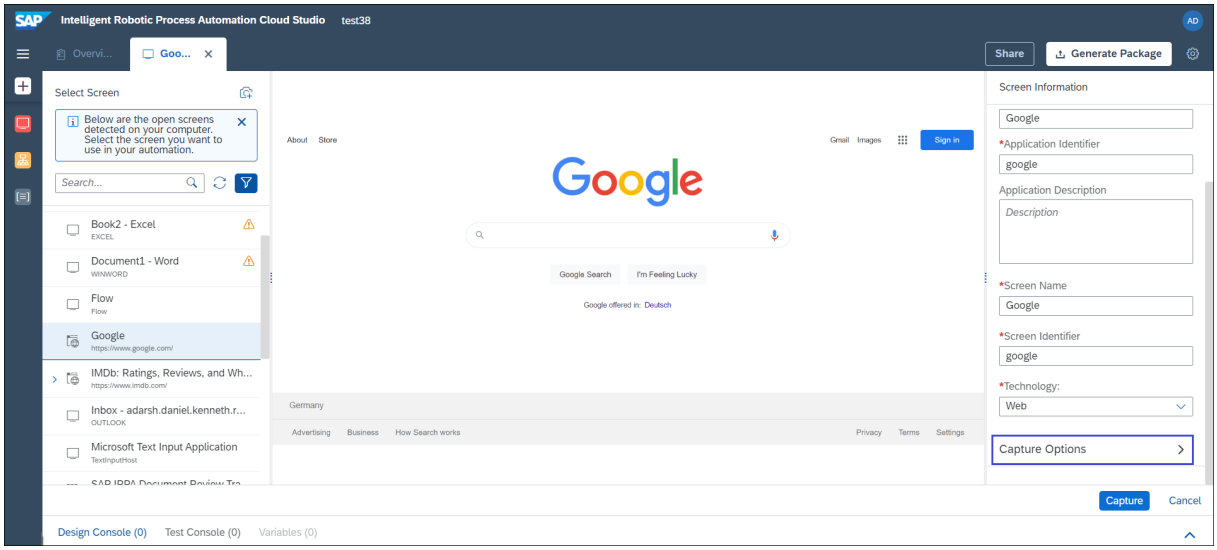
You can also select the elements within the frame without any offsets as shown in the following screenshot.



4.1.2.5 Capture Options

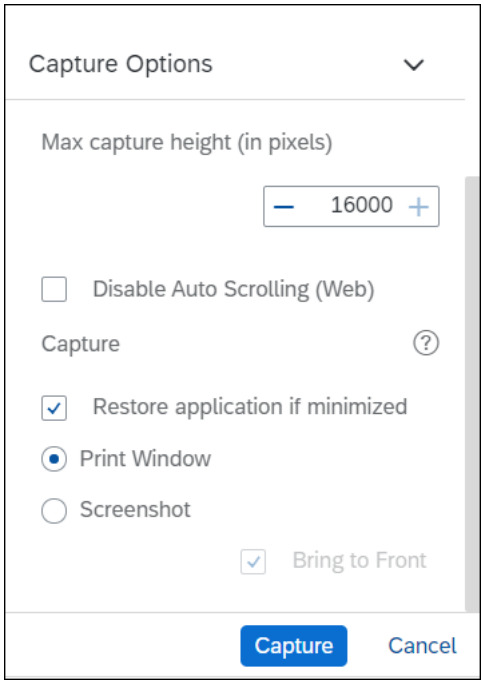
SAP Intelligent RPA Cloud Studio supports multiple application capture options such as *Max capture height (in pixels)*, *Disable Auto Scrolling (Web) Print Window*, *Screenshot (Bring to Front)*, and *Screenshot*. These options help the user to capture various scenarios of the application screens.

You can find the capture options at the bottom of the left panel in the Cloud Studio.



To see the application capture options click *Capture Options*. The capture options are displayed.

Before capturing a screen, you can define one or more of the following capture options, if necessary.



Capture Option	Description
Max capture height (in pixels)	This option limits the height of the screen captured. It's useful when capturing very large web pages. Please note that the entire tree structure of the screen's elements (fields, buttons, and so on) is captured. After capturing a screen, you can see this tree structure by switching the view in the center pane from *Screen* to *Tree*.
Disable Auto Scrolling (Web)	It disables auto scrolling of web application screen.
Restore application if minimized	If the Restore application if minimized option is selected, then the target application screen that is minimized will be restored while capturing. If the Restore application if minimized option is not selected, then the target application screen that is minimized will not be restored while capturing.
Print Window	By default, the target application screen is captured using the *Print Window* mode. **i Note** If the resulting capture is a black bitmap, the Cloud Studio automatically switch to the *Screenshot* mode.
Screenshot	This mode is the same as the *Screenshot* mode, except that it is up to you to bring the target screen to front. This can be useful in some specific cases where an application has a child window part (pop-up screen).

Screenshot with Bring to Front	If the Cloud Studio do not detect that the resulting capture is black, you can manually switch to the Screenshot (Bring to Front) mode by selecting *Screenshot (Bring to Front)* option. **i Note** In this case, the Cloud Studio automatically brings the target screen to front.

Related Topics

- Capture an Application [page 77]
- Warning Messages when Capturing a Screen [page 84]

4.1.2.6 Application Multi-Captures

You can capture multiple screens of the same application.

The captures represent variations of the same screen.

At any point in time you can switch between the captures and delete a capture. You can define the Capture Name and Description (Optional).

Elements defined in one of the captures will be highlighted on all the other captures based on the defined criteria.

4.1.2.6.1 Multi-Captures

You can capture multiple screens of the same application. The captures represent variations of the same screen.

Prerequisites

You've captured an application and at least one screen as described in the Capture an Application [page 77] section.

You must define the required elements for a screen capture. For more details about declaring an element refer to the Declare Element Icons [page 107] and Declare an Element [page 110] sections.

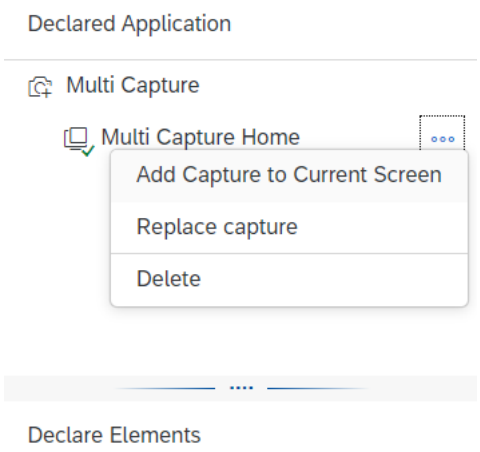
Context

To capture multiple screens of the same application, follow the below procedure:

Procedure

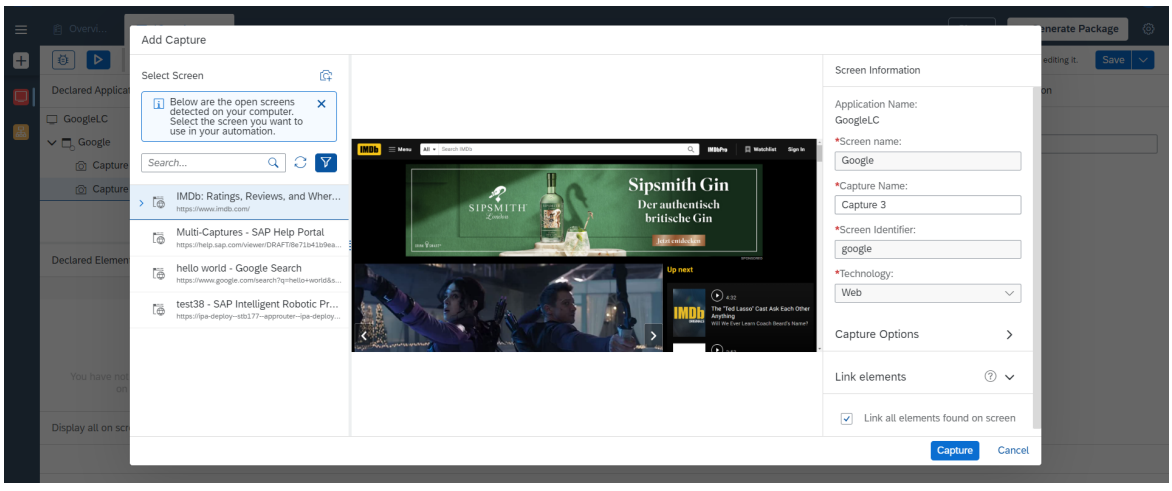
1. In the IRPA Cloud Studio, under *Declared Application*, click the ••• icon of a captured screen.

   More Options are displayed.



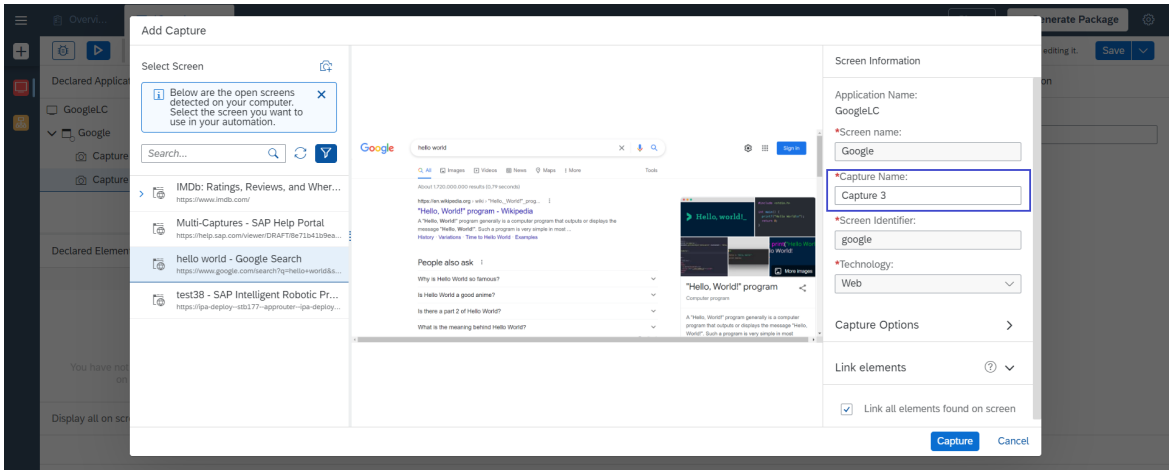
2. Click *Add Capture to Current Screen*.

   The *Add Capture*Add Capture screen is displayed.

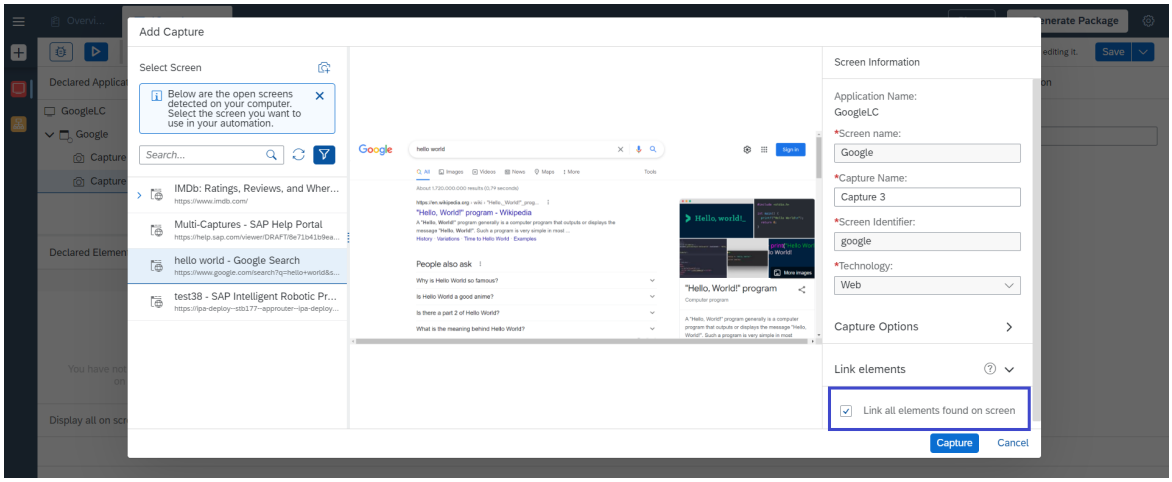


3. In the open screens list detected from your agent, select the desired screen you want to add.

   The selected screen is displayed in the preview area.

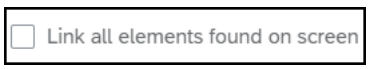


4. Enter the *Capture Name*, if required.
5. By default, the *Link all elements found on screen* check box is seleted.



This will fetch all declared elements from the screen and link to current capture, only those elements that can be found based on its criteria in the current capture.

If you do not want to link all elements found on the screen then deselect the *Link all elements found on screen* check box.



6. Click *Capture*.

   After successful capture of the application screen, a preview screen is displayed in the capture area.

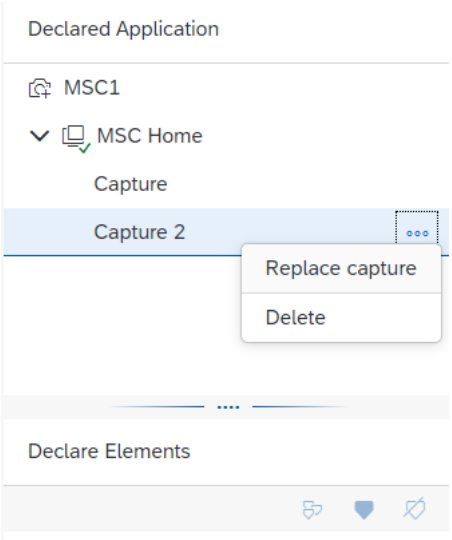
   All declared and linked lements are listed under *Declare Elements*.

4.1.2.6.2 Delete a Capture

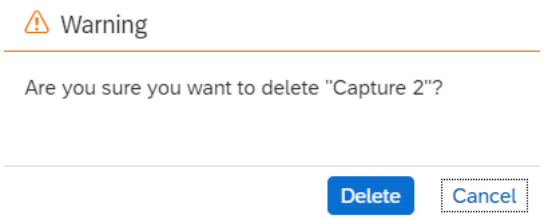
At any point in time you can delete a capture.

To delete a capture, follow the below procedure:

1. In the IRPA Cloud Studio, under *Declared Application*, select the desired capture and click the ••• icon.
   The more options are displayed.



2. Click *Delete*
   The warning pop-up message is displayed.

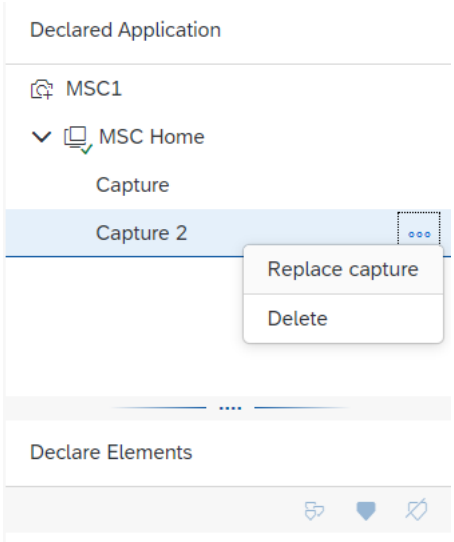


3. Click *Delete*
   The capture will be deleted.

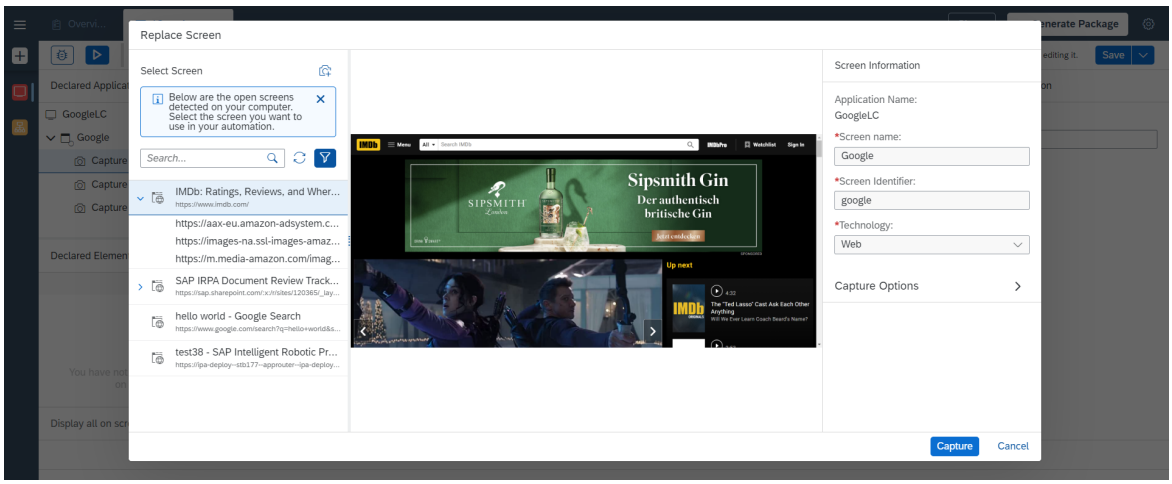
Replace a Capture

To replace a capture, follow the below procedure:

1. In the IRPA Cloud Studio, under *Declared Application*, select the desired capture and click the ••• icon.
   The more options are displayed.



2. Click *Replace capture*.
   The *Replace Screen* is displayed.
3. In the open screens list detected from your agent, select the desired screen you want to replace with the previous capture.
   The selected screen is displayed in the preview area.



4. Click *Capture*.
   After successful capture of the application screen, a preview screen is displayed in the capture area.

4.1.2.7 Capture on Hover

You can capture an application screen using Capture on Hover.

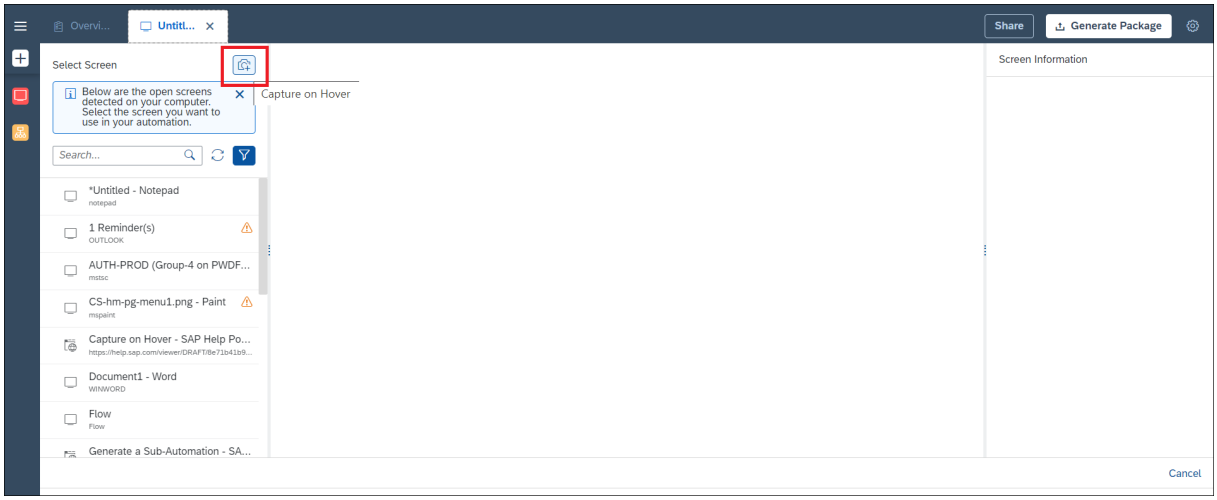
> **i Note**
>
> The capture on hover is only available for Desktop Agents version 2.0.12 and above.

Sometimes, elements can disappear from the screen when you lose focus on the screen to be captured. In such cases, you can still capture the element such as drop-down menus using the *Capture on Hover* option. To

achieve this, you must navigate to the application you want to capture and hover over the screen, part of the screen, or drop-down menu to capture.

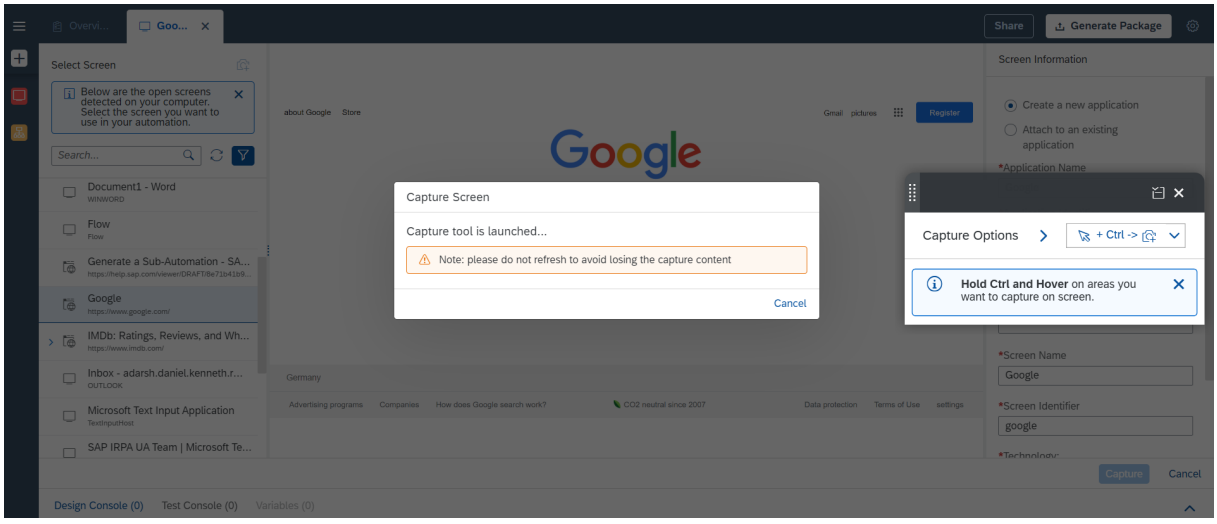
Using Capture on Hover

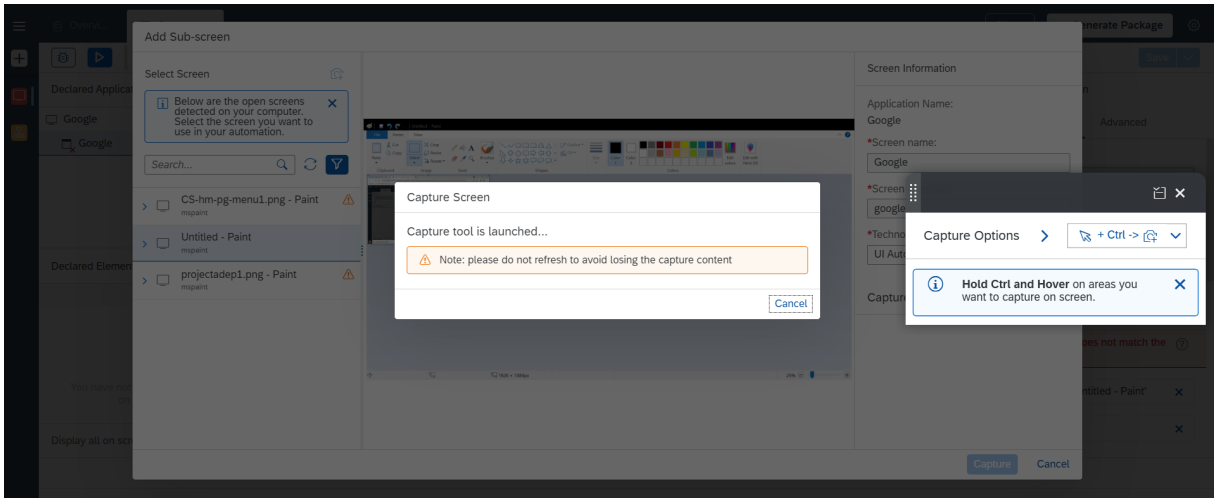
In the Cloud Studio, you can find the *Capture on Hover* option at the right side of the *Select Screen* panel.



> ### Note
>
> In the Cloud Studio, the *Capture on Hover* option is available when capturing the first screen of an application, adding capture, and replacing a screen.

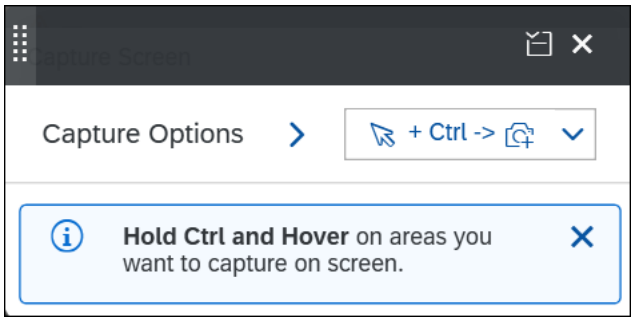
You can capture an application by selecting a screen in the Cloud Studio or without selecting a screen. When you click the *Capture on Hover* button, the Cloud Studio is locked and the Capture tool is launched. For example, refer to the following screenshot.





> ### Note
>
> If you cancel the capturing or Cloud Studio cancels the capturing, then the Cloud Studio is unlocked.
>
> The capture tool will be closed if you refresh or close the Cloud Studio.

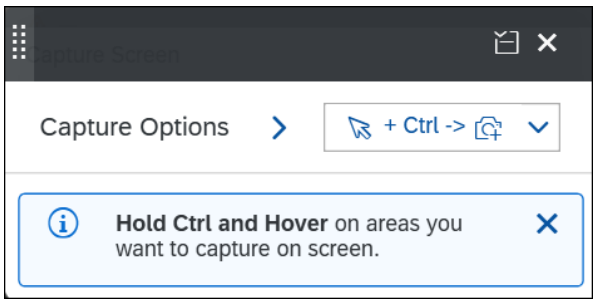
Capture Tool



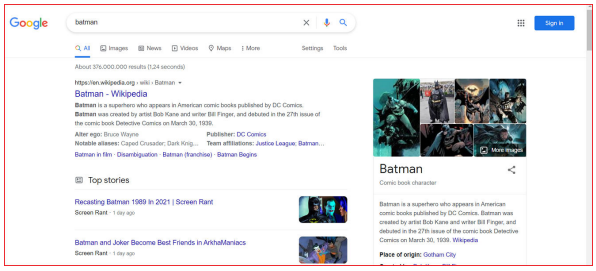
The capture tool includes the two capture options **ctrl + hover** and **shift + hover**. By default, the **ctrl + hover** option is enabled.

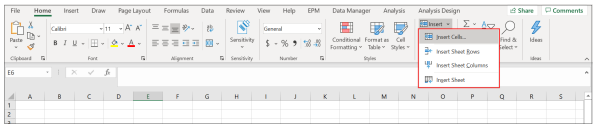
The function of these two options are the same. The only difference is the keys (ctrl and shift) you use to capture an application screen.

ctrl + hover



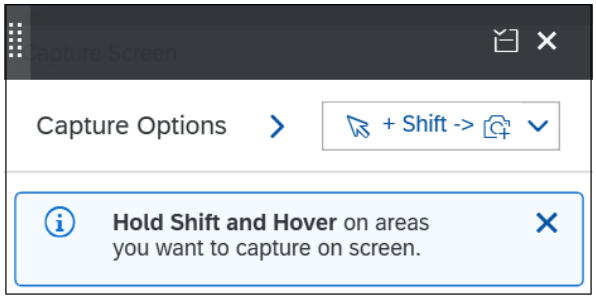
Press and hold the **ctrl** key and then hover the mouse over the entire screen,part of the screen, or drop-down menu. The hovered area is highlighted with a red border and then starts capturing the highlighted area.





After successful capture, the capture tool is closed automatically and you will be redirected to the Cloud Studio.

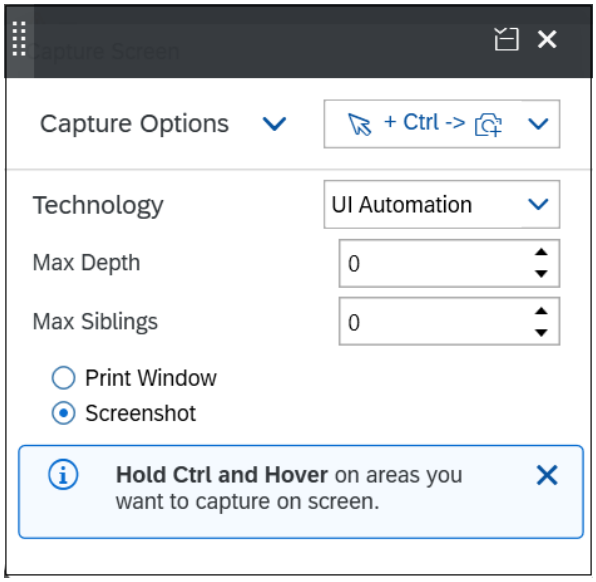
shift + hover



Press and hold the **shift** key and then hover the mouse over the entire screen, part of the screen, or drop-down menu. The hovered area is highlighted with a red border and then starts capturing the highlighted area.

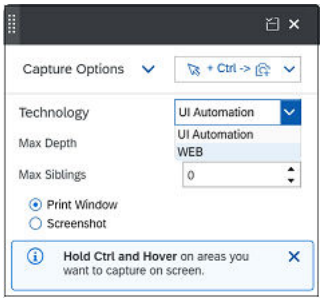
After successful capture, the capture tool is closed automatically and you will be redirected to the Cloud Studio.

More options



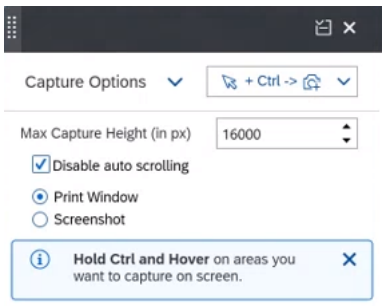
Technology:

This option allows you to change the technology. You can change technology only when you are on the UIAutomation screen. The web can be captured in UIAutomation or in a Web screen. But when you are on a web screen, you cannot change the technology and must capture with the Web technology only.



Disable auto scrolling:

This option disables auto scrolling. It is useful when capturing a pop-up menu.

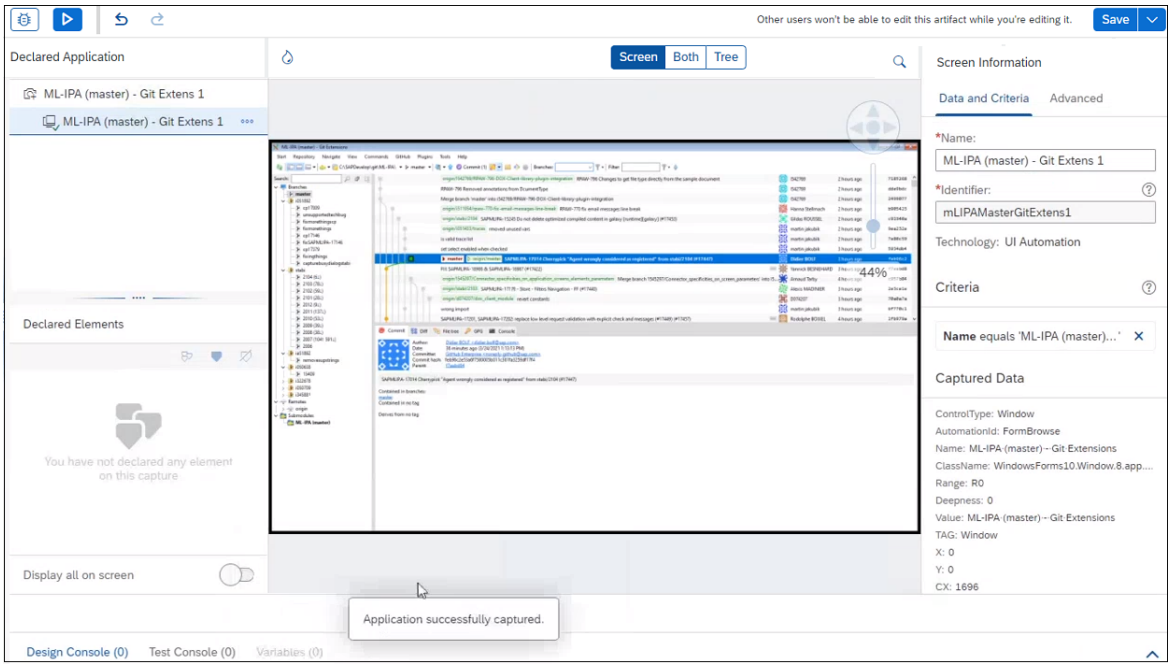


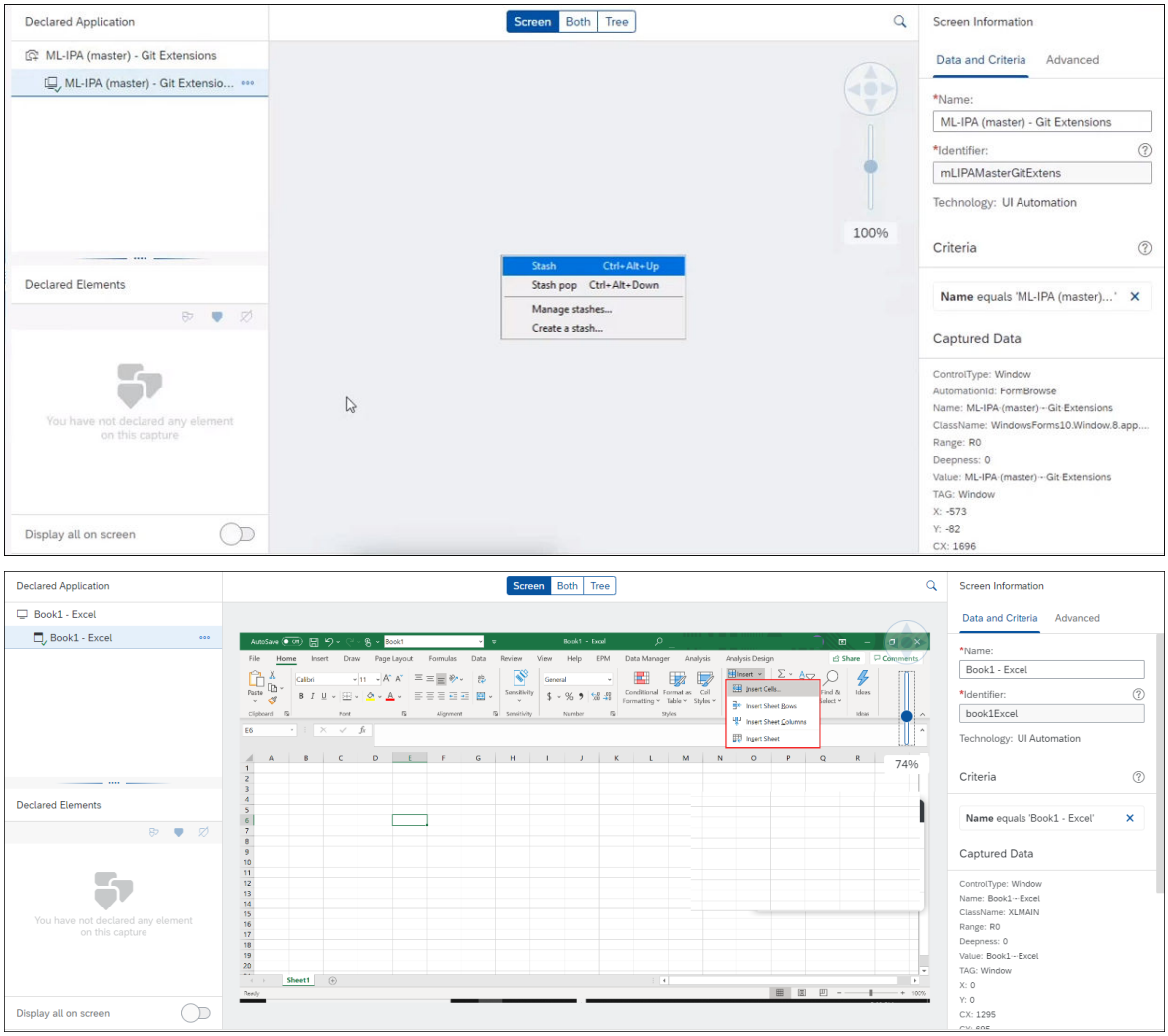
Sometimes, when you scroll, you may lose focus on the to be captured area. In such cases, use the *Disable auto scrolling* option and then capture the required part or menu of the web page.

To know more details about the capture options, refer to the Capture Options [page 87] section.

Procedure

1. Click *Capture on Hover*. The Cloud Studio is locked and the capture tool is launched.
2. Go to the application you want to capture.
3. Press ctrl and then hover on the application screen you want to capture. On the capturing tool, the *Capturing Screen to Studio* message is displayed.
4. Go back to the Cloud Studio. You can see the captured screen. For example, refer to the following screenshots.





If required, you can add a new screen or replace the current captured screen using *Capture on Hover*.

4.1.2.8 Capture Sub Screen

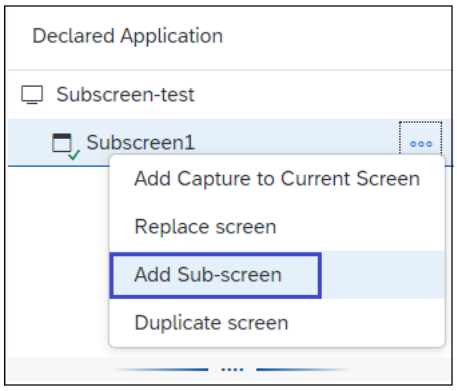
In some automation processes, it is required to declare a sub screen. Cloud Studio supports sub screen capture for UI Automation only.

This sub screen relationship ensures that the required controls are recognized. However, if not declared correctly, it prevents the required controls from being recognized. A sub screen is declared in the same way as an application screen. The only difference is that it must be declared with the corresponding parent screen.

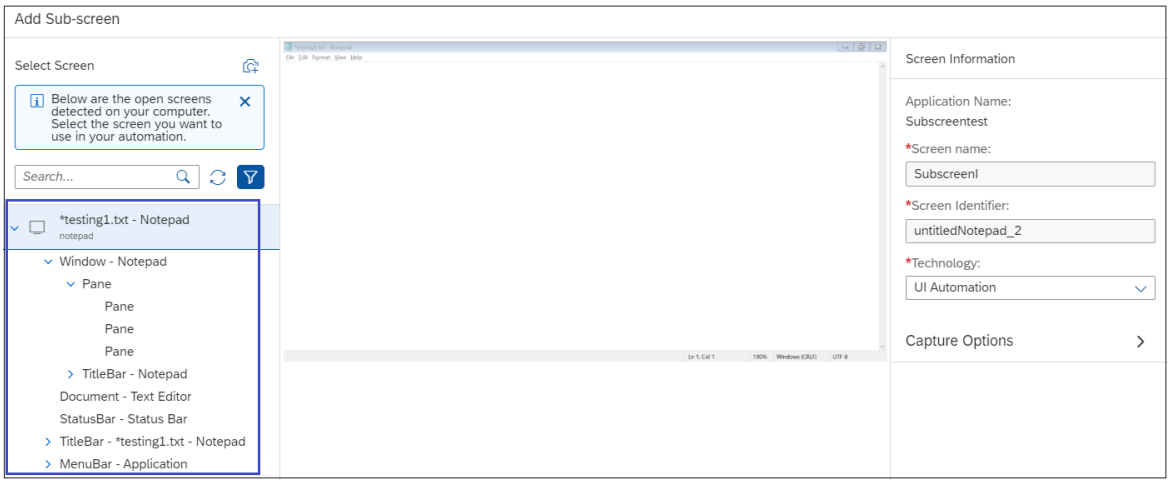
> **i Note**
>
> The SDK version 1.12.0 or higher and the Desktop Agent Version 2.0.14 or higher are supported for sub screens.

Capturing a Sub Screen

1. In Cloud Studio, under *Declared Application*, click the ••• icon of an application screen for which you want to capture the sub screen.
   More Options are displayed.



2. Click *Add Sub-screen*.
   The *Add Sub-screen* pop-up screen is displayed.
3. In the recognized screens list from your agent, select the appropriate parent screen.
   The screen structure is displayed. For example, refer to the below screenshot.



> **i Note**
>
> In the screen structure, select the screen or an item to preview in the main pane.

4. Select the to be captured screen or screen item.
5. Update the screen name in the *Screen name* field, if required.
6. Click *Capture*.
   After successful capture, a preview is displayed in the main pane of Cloud Studio.
   All declared screens are listed under *Declared Applications*.

> **i Note**
>
> You can capture multiple sub screens.
>
> You can capture a sub screen of a sub screen.

You can capture child sub screen of a sub screen.

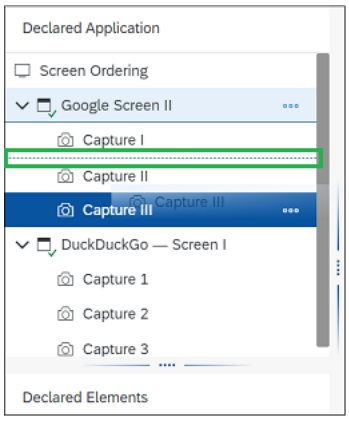
4.1.2.9 Manage Screen Ordering

In Cloud Studio, you can manage the order of applications, screens, and sub-screens. This helps the connectors to recognize the screens in the right order during runtime.

Prerequisites

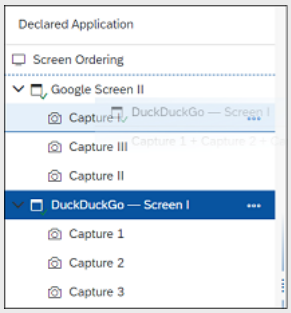
Capture at least one application and two screens. For more details about capturing an application and screens, refer to the Capture an Application [page 77] section.

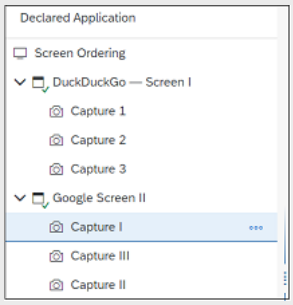
In the left panel, under the *Declared Application*, when you mouse hover with an item (application, screen, or sub-screen) at the targeted place, the straight line is highlighted with a dotted line. This helps you to drop an item at an appropriate place.



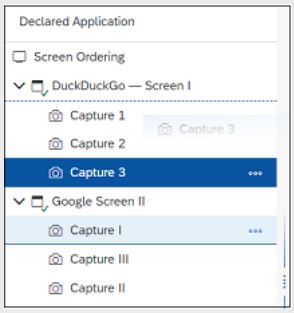
Note

When you reorder a screen, the associated captures and sub screens will also be moved.

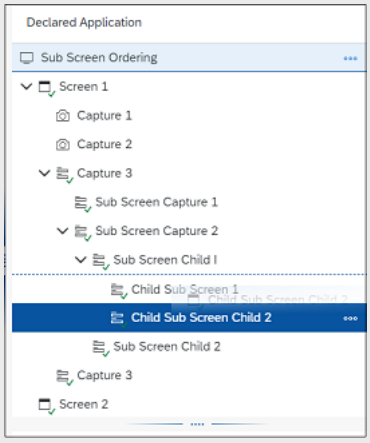




You can reorder captures in the same level only under its screen.



You can reorder sub screen only in the same level. Moving to a parent or its child screen is not allowed.



4.1.3 Declaration Phase

To use applications and its screens and elements in your automation project, you first need to capture and declare them. Declaring means selecting the criteria that the system will apply in order to correctly identify an application, screen or element when running your automation.

An application is composed of one or several screens. As for the screens, they are made of elements. Elements are buttons, input fields, and other controls on a screen.

4.1.3.1 Declare an Application

Prerequisites

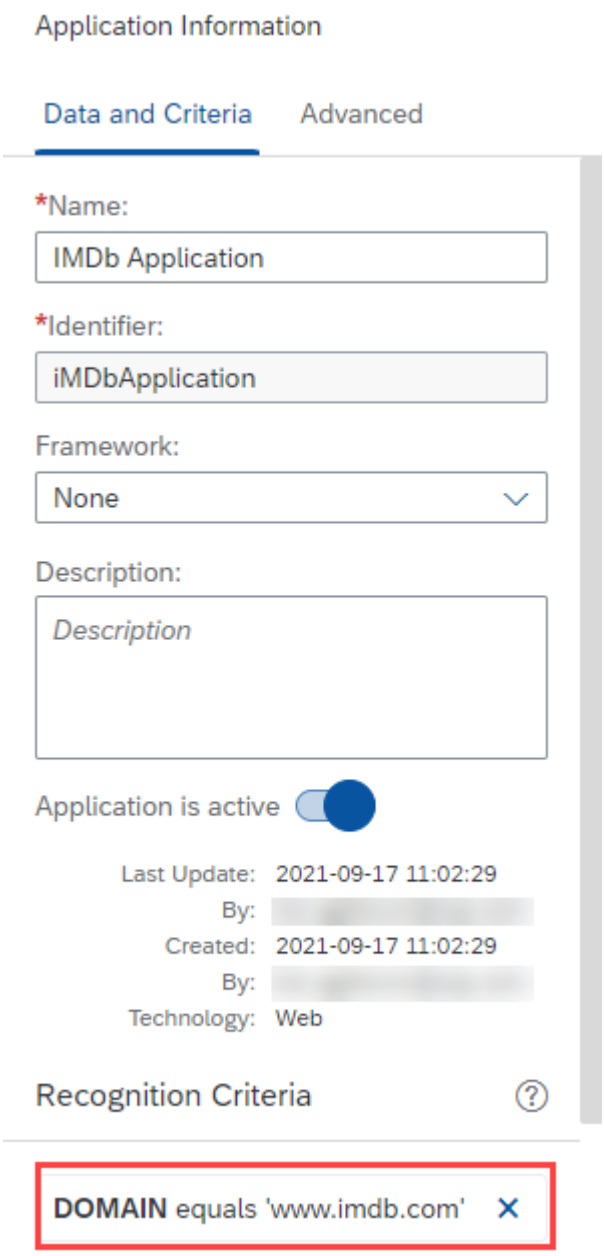
You've captured an application and at least one screen as described in Capture an Application [page 77].

Context

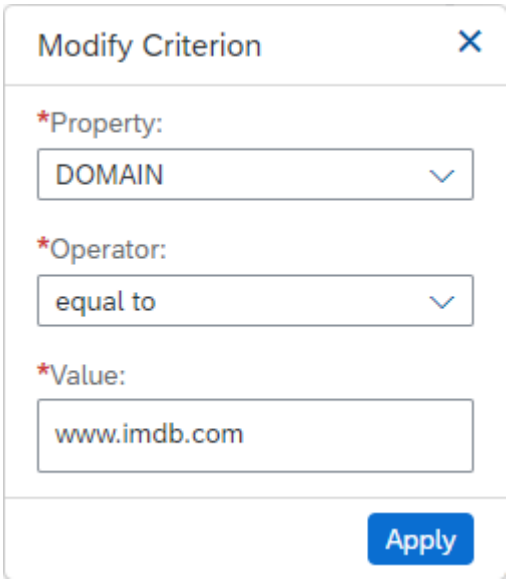
To declare an application, you need to set recognition criteria - used to identify the application correctly when your automation runs.

Procedure

1. The application recognition criteria - used to identify the application correctly when your automation runs - are automatically selected during the capture.



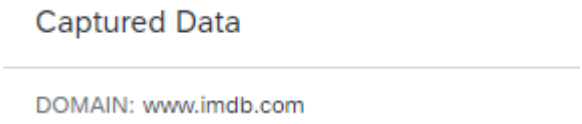
If necessary, click a criterion in the *Recognition Criteria* panel to change its properties.



→ Tip

You cannot declare two different applications with the same criteria (same domain for instance), otherwise only the first one will be recognized, and the second application will be hidden.

2. **Optional:** To add other criteria, click any data displayed in *Captured Data*: the selected data is then automatically added as a criterion.



3. Once you defined your recognition criteria, click *Save*.

4.1.3.1.1 Application Criteria

When capturing an application using UIAutomation technology, two properties are provided in the captured data of the application:

- EXE: name of the process
- TITLE: title of the main associated window

If the TITLE property is set as a recognition criterion in the Application criteria, the UIAutomation connector ignores it. This criterion is only used if you declare a web screen inside a UIAutomation application. The TITLE property is not available in the recognition criteria of the different screens captured for an application.

4.1.3.2 Declare a Screen

Prerequisites

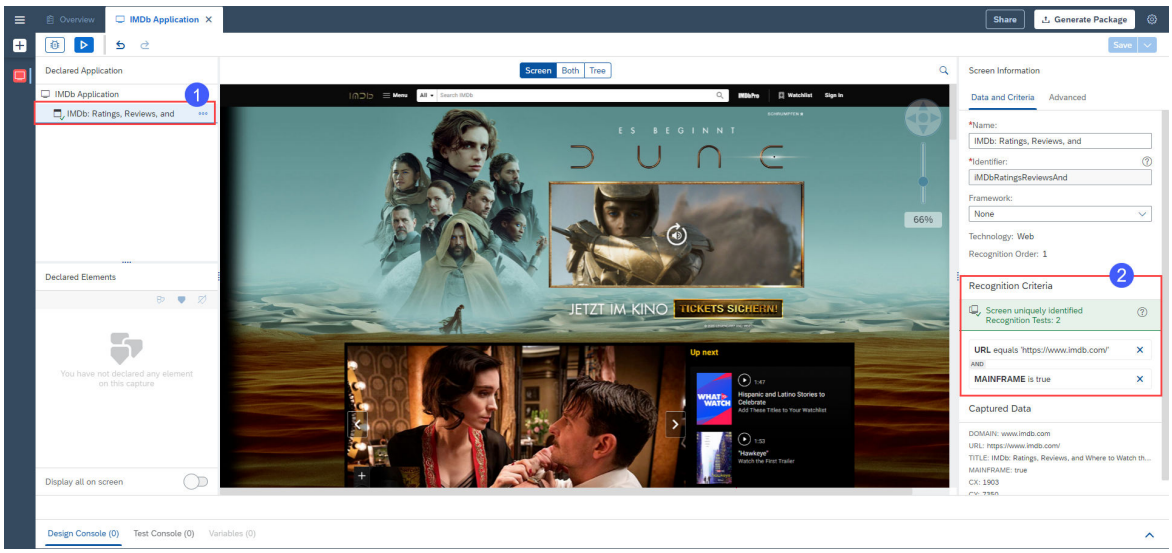
You've captured an application and at least one screen as described in Capture an Application [page 77].

Context

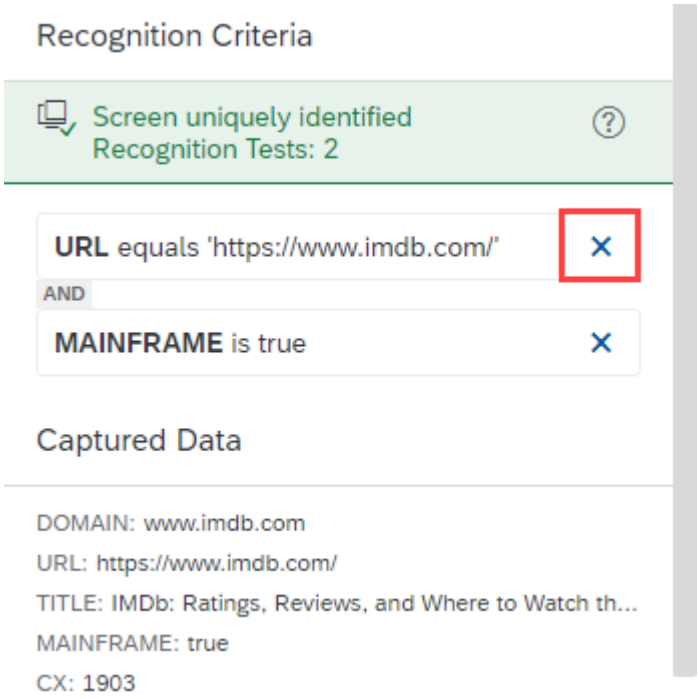
To declare a screen, you need to set recognition criteria - used to identify the screen correctly when your automation runs.

Procedure

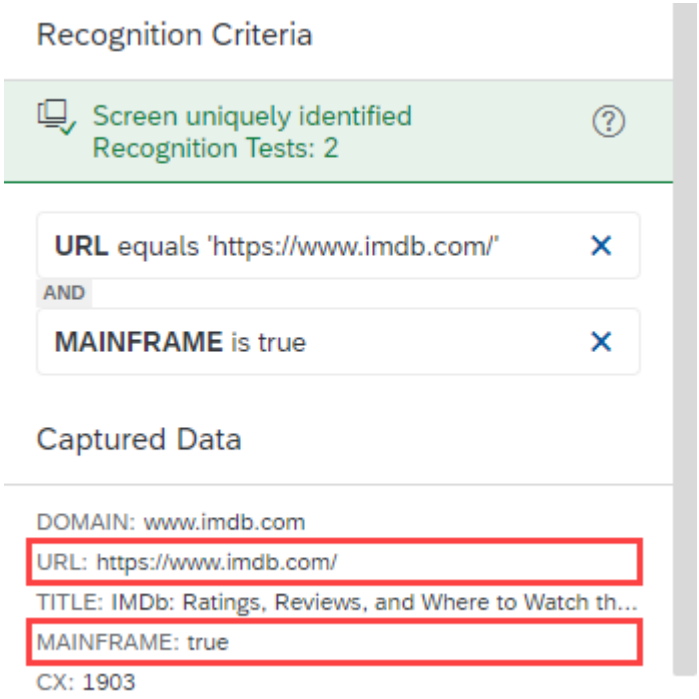
1. Once a screen has been captured, select it in the left-hand navigation panel (1). Some criteria have been automatically selected during the capture (2).



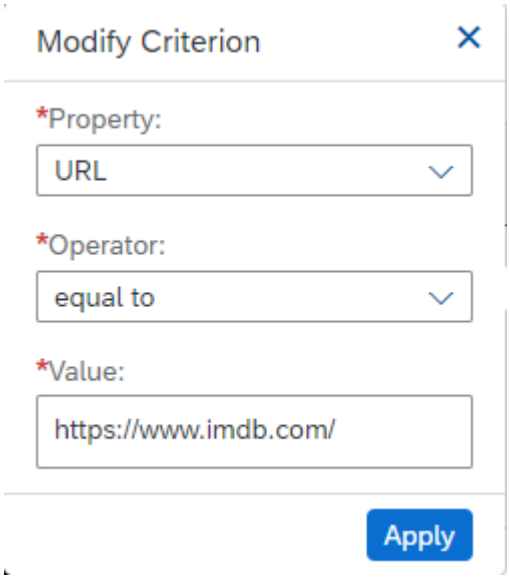
2. **Optional:** Select ✕ to remove a criterion.



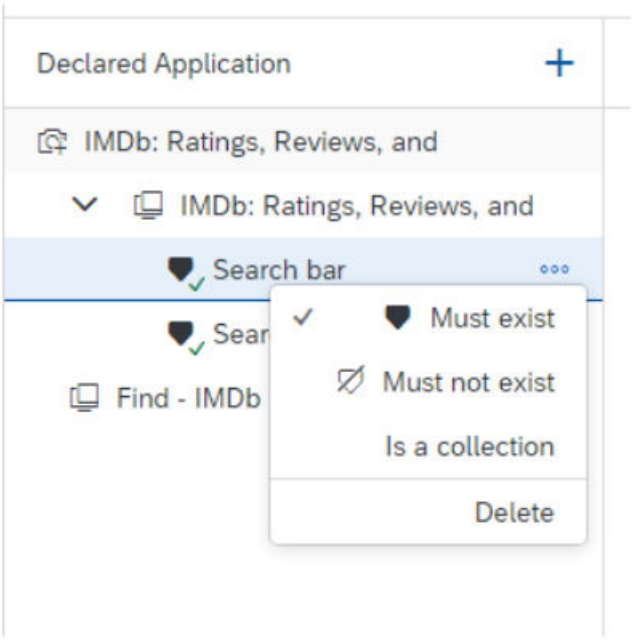
3. **Optional:** To add other criteria, click any data displayed in *Captured Data*: the selected data is then automatically added as a criterion. In the example below, the captured data "MAINFRAME" and "TITLE" (1) have been assigned as criteria (2).



4. **Optional:** Click a criterion in the *Criteria* panel to change its properties.



5. Once you've defined your recognition criteria, click *Save*.
6. **Optional:** After you've declared one or more elements of the screen, you can define an element as `must exist`, `must not exist` or as a `collection`. Select the three dots after the element name in the picker panel.



Must exist means that the screen is not recognized unless the element or elements defined as must exist are detected.

Must not exist means that the screen is not recognized if an element defined as must not exist is detected.

Is a collection means that the element consists of multiple parts - for example, a list with multiple entries.

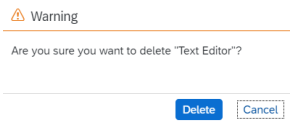
4.1.3.3 Declare Elements

An application is composed of one or several screens. As for the screens, they are made of elements. Elements are buttons, input fields, and other controls on a screen.

4.1.3.3.1 Declare Element Icons

Before defining the elements for a capture, go through the declare element icons and its descriptions. This will help you to define element for a capture appropriately.

Icon	Icon Name	Description	Procedure
	Is a collection.	You can define an element as a collection. For more details about declaring an element as a collection refer to the Declare an Element as a Collection [page 120] section.	Select the desired element and click the  icon to define the element as a collection.
	Must exist.	The element is mandatory in executing an automation.	Select the desired element and click the  icon to define an element as **Must exist**.
	Must not exist.	The element is not mandatory in executing an automation.	Select the desired element and click the  icon to define element as **Must not exist**.
	Link.	This includes various options such as *Link to selected element*, *Unlink*, *Learn more*, and *Appears in*.	Select the desired element and click the  icon to view available link options.
Link to selected element	Link to selected element.	This will remove the existing link and link to the selected element in the current capture or previous capture.	Select the desired element and click the  icon and then click *Link to selected element* to remove the existing link and link to the selected element.
Unlink	Unlink.	At any point in time you can manually unlink the element in the capture.	Select the desired element and click the  icon and then click *Unlink* to unlink the element in the capture.
Appears in	Appears in.	Displays the captures in which the element appears.	Select the desired element and click the  icon and

			then click *Appears in* to view the captures in which the element appears.
Learn more	Learn more.	Navigates to the help topic.	Select the desired element and click the icon and then click *Learn more* to go to the help topic.
	Ignore Element on this capture.	The element will be ignored in the capture and the element will not be linked when you select the option *Link all elements found on screen*.	Select the desired element and click the icon to ignore the element in the capture.
	Do not ignore element on this capture.	The element will not be ignored in the capture.	Select the desired element and click the icon to ensure that the element is not ignored in the capture.
	More options.	This includes more options such as *Add to criteria* and *Delete*.	Select the desired element and click the icon to view more options.
Add to criteria	Add to criteria.	The element is added in the side panel hierarchy as a parent/child element to the criteria of the target element. You can also modify the criteria for this parent/child element.	Select the desired element and click the icon and then click *Add to criteria*.
Delete	Delete.	This allows you to delete an element.	Select the desired element and click the icon and then click *Delete*. The warning pop-up message is displayed.  Click *Delete*. The element is deleted from the capture.
Duplicate	Duplicate.		

4.1.3.3.2 Capture Operators

Describes the capture operators.

Operators for String

Cloud Studio	Description
equals to	The property value is strictly equal to the argument.
not equals to	The property value is strictly NOT equal to the argument.
contains	The property value (an array) contains the argument.
doesn't contain	The property value (an array) does not contain the argument.
starts with	The property value starts with the argument. For Web-based only (*).
doesn't start with	The property value does not start with the argument. For Web-based only (*)
ends with	The property value ends with the argument. For Web-based only (*)
doesn't end with	The property value does not end with the argument. For Web-based only (*)
is empty	The property is empty. No argument.
is not empty	The property is NOT empty. No argument.
match	The property value (an array) matches the argument.
doesn't match	The property value (an array) does not match the argument.

```
<Property> <operator> <argument>: AutomationID equals to 'num4ID'
<Property> <operator> <argument>: AutomationID is empty
```

Operators for Numbers

Cloud Studio	Description
=	The property value is strictly equal to (=) the argument.
≠	The property value is strictly NOT equal to the argument.
>	The property value is strictly greater than the argument.
<	The property value is strictly less than the argument.
≥	The property value is greater or equal than the argument.
≤	The property value is less or equal than the argument.
≤	The property value is less or equal than the argument.
≯	The property value is not strictly greater than the argument.
≮	The property value is not strictly less than the argument.
≱	The property value is NOT greater or equal than the argument.

Cloud Studio	Description
≮	The property value is NOT less or equal than the argument.

```
<Property> <operator> <argument>: CX = 42
```

Other Operators (Boolean and List)

- A Boolean property has one value that can be selected or unselected.
- An enumeration is a collection (of Boolean).

Cloud Studio	Description
Selected or Unselected **Checkbox**	The property value (a Boolean) is selected or not.
Selected or Unselected **Checkboxes**	The property values (Boolean) are selected or not.

```
<Property> <operator> <argument>: isFourtyTwoTheAnswer = true
```

4.1.3.3.3 Declare an Element

Prerequisites

You've captured an application and at least one screen as described in Capture an Application [page 77]. You've also declared a screen as explained in Declare a Screen [page 104].

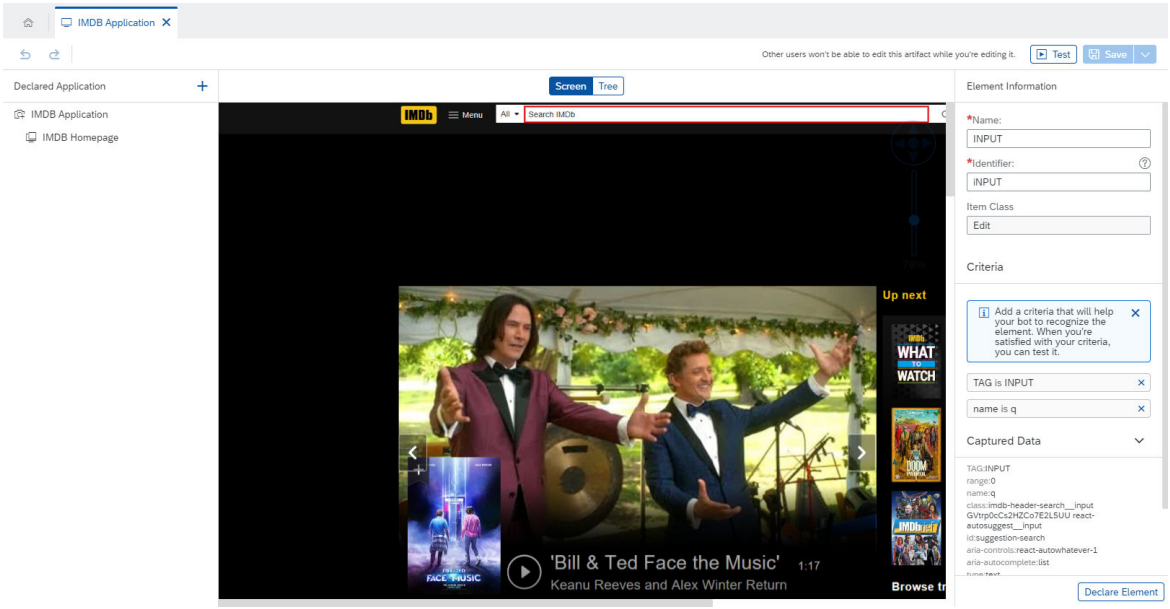
Context

To declare elements on a screen, you need to set recognition criteria - used to identify the elements correctly when your automation runs.

Procedure

1. Once a screen has been captured and declared, select it in the left-hand navigation panel.
2. Click an element to select it.

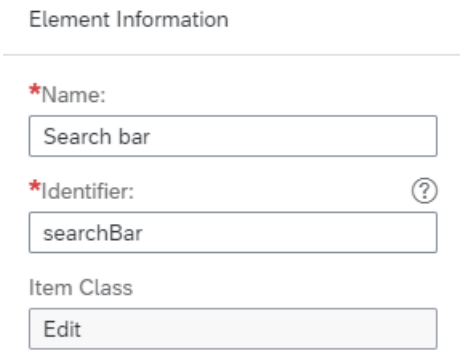
A green border is shown around the selected element - in this example, it's the search bar.



> ### → Tip
>
> If you find it difficult to highlight the right screen element in the image view, choose *Tree* and select it in the tree view instead.

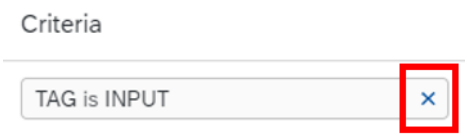
3. In the *Element Information* side panel, enter a name and an identifier for the element.

   The *Element Class* is automatically detected. *Edit* means the element is editable - it's an input field.



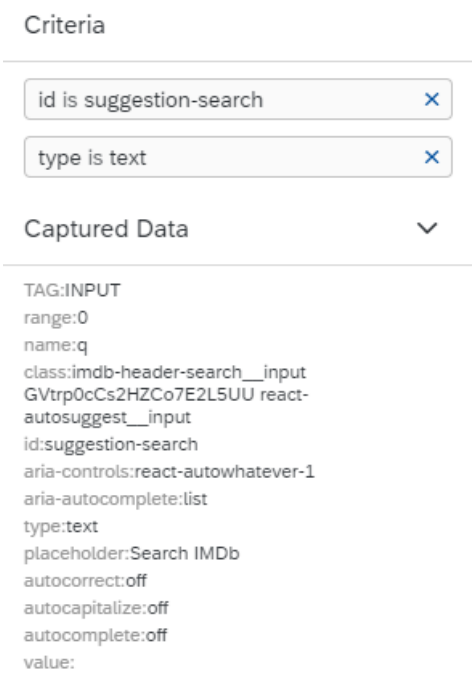
4. **Optional:** Choose the criteria you want to assign to the element.

   a. Remove a criterion via ✕.

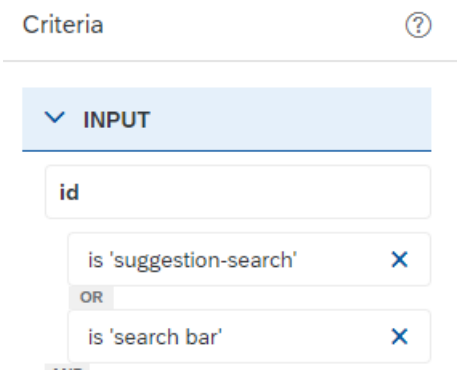


   b. To add further criteria, click any data displayed in *Captured Data*.

The selected data is then automatically added as a criterion. In the example below, the captured data "id" and "type" have been assigned as criteria.



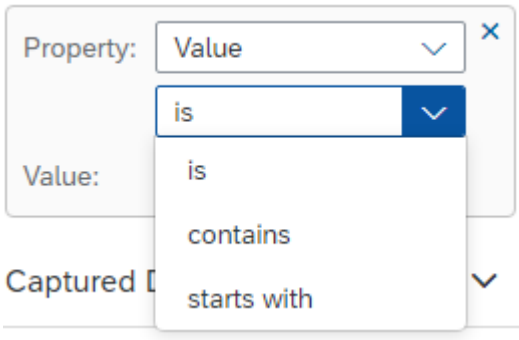
c. **Optional:** To add another value to a criterion, click again in *Captured Data* the data you already assigned as a criterion. This other value is displayed in the input panel below the operator *or*.



Then click the criterion to change its value (see next step).

5. **Optional:** Click a criterion in the *Recognition Criteria* panel to change its properties.

   A field value property, for example can be changed by choosing a different comparator instead of "equal to" and altering the value.

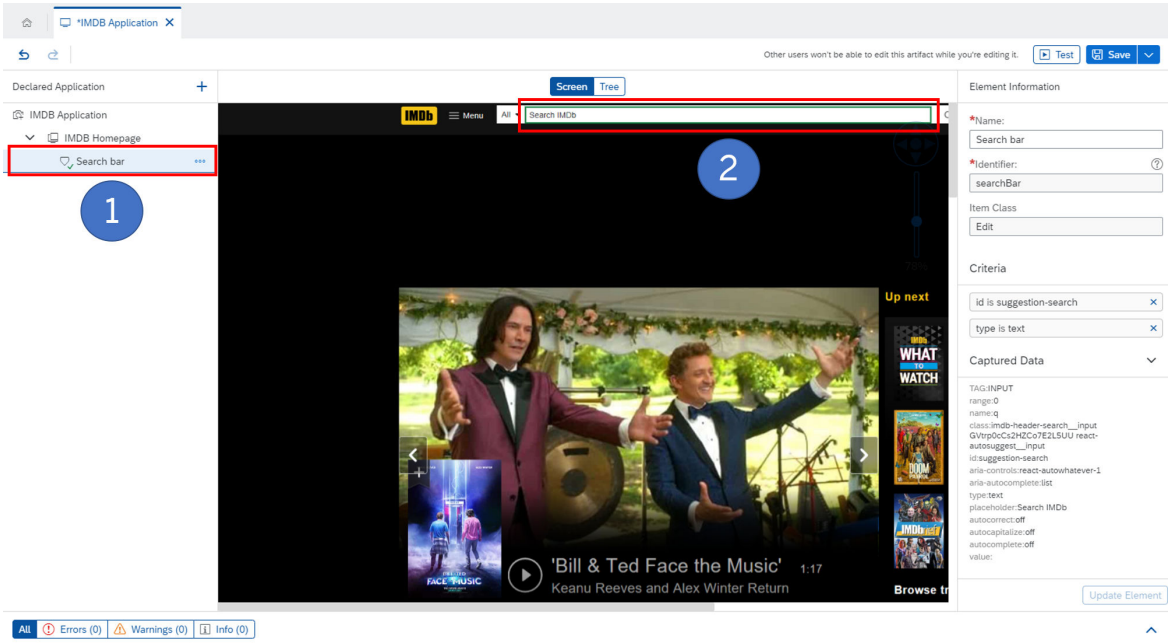


Example

Change "Value equal to Search" to "Value starts with Search".



6. Once you've added and defined your recognition criteria, click *Declare Element*.



Note

By default, if you create a new capture, all declared element of the screen are linked to the new capture. But, if you declare a new element, it is linked only to the currently displayed capture. So, after declaring new element, you must link it to other captures, if required.

> **Note**
>
> You can drag and drop the declared elements within the left-side panel. This functionality allows you to reorganize your declared elements.

When an element has been captured, it is displayed in the *Declared Elements* panel (1). When you select the element on the left-side hand panel, it is automatically highlighted on the screen (2).

7. Save your work.

> **Note**
>
> Delete an element by choosing the three dots after the element's name in the picker panel and selecting *Delete*.

4.1.3.3.3.1 Duplicate Declared Element

In the Cloud Studio, you can duplicate the declared elements. The duplicate element represents the characteristics of the original element.

Prerequisites

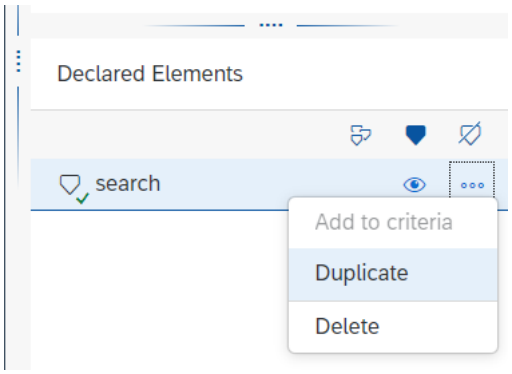
You've declared at least one element as described in Declare an Element [page 110].

Context

Procedure

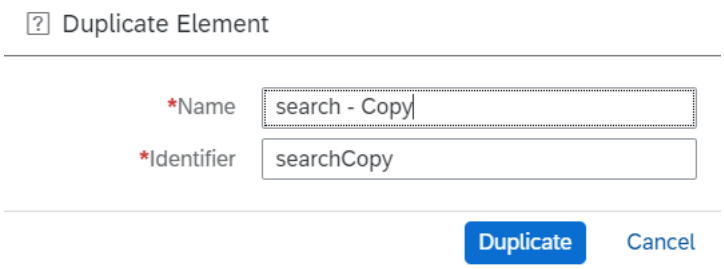
1. In the IRPA Cloud Studio, under *Declared Application*, select the application screen for which you want to declare a duplicate element.
2. Under *Declared Elements*, click the ••• icon of a element.

   More Options are displayed.



3. Click *Duplicate*.

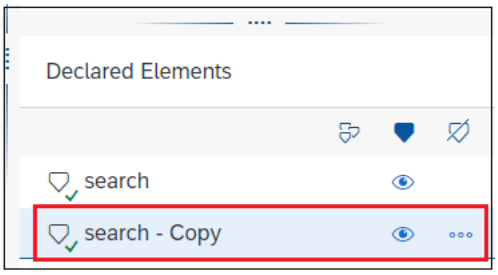
   The *Duplicate Element* pop-up screen is displayed.



> **i Note**
>
> If required, you can update the name of an element.

4. Click *Duplicate*.

   The duplicated element is created and displayed under *Declared Elements*.



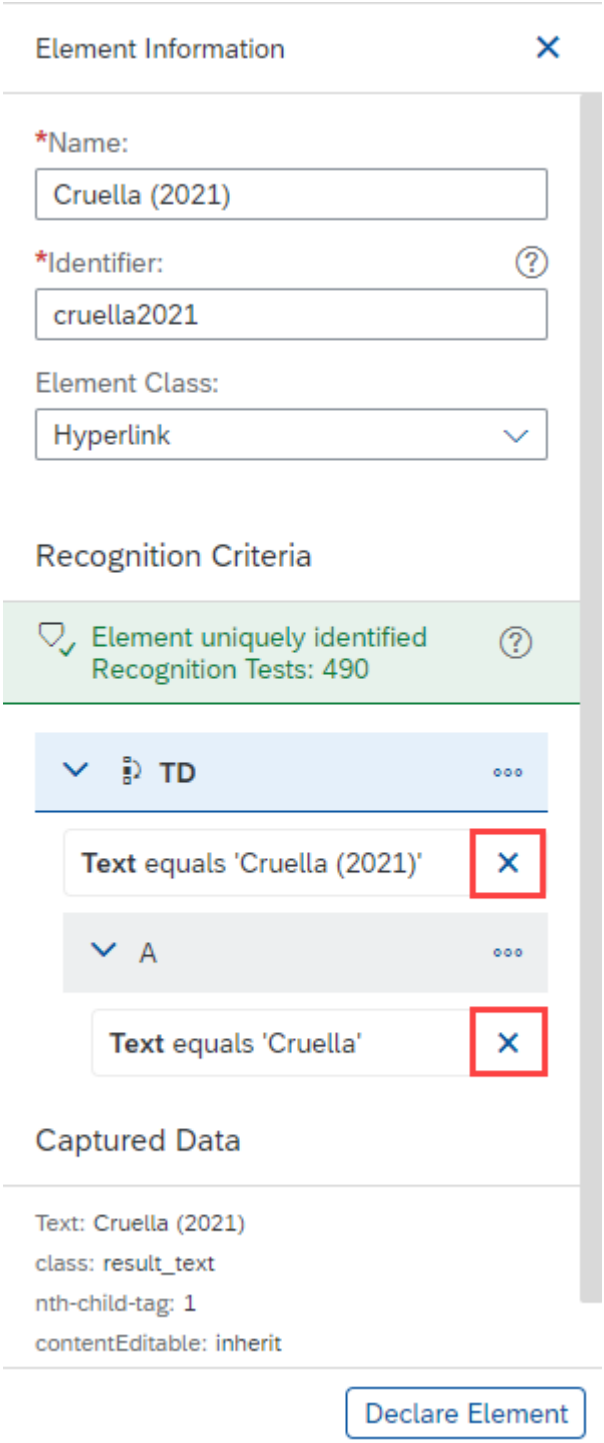
4.1.3.3.3.2 Declare a Parent as a Collection

Context

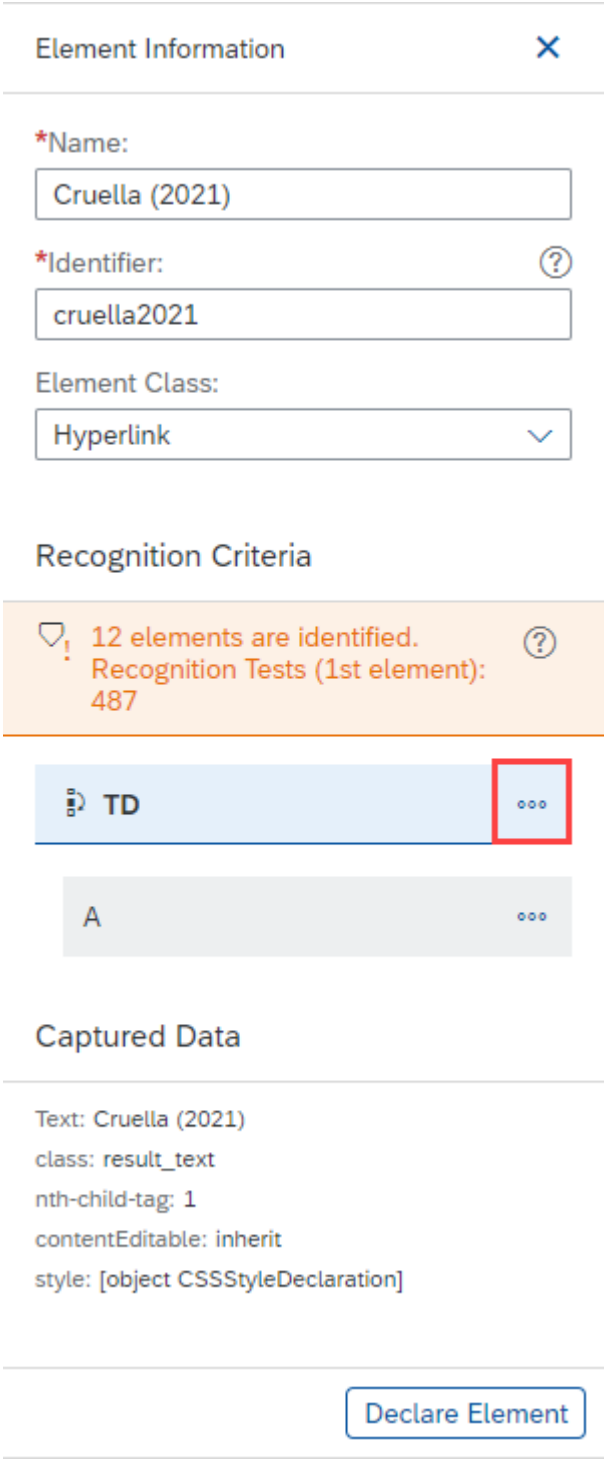
You can set the parent element of a declared element as a collection for an easier pattern identification.

Procedure

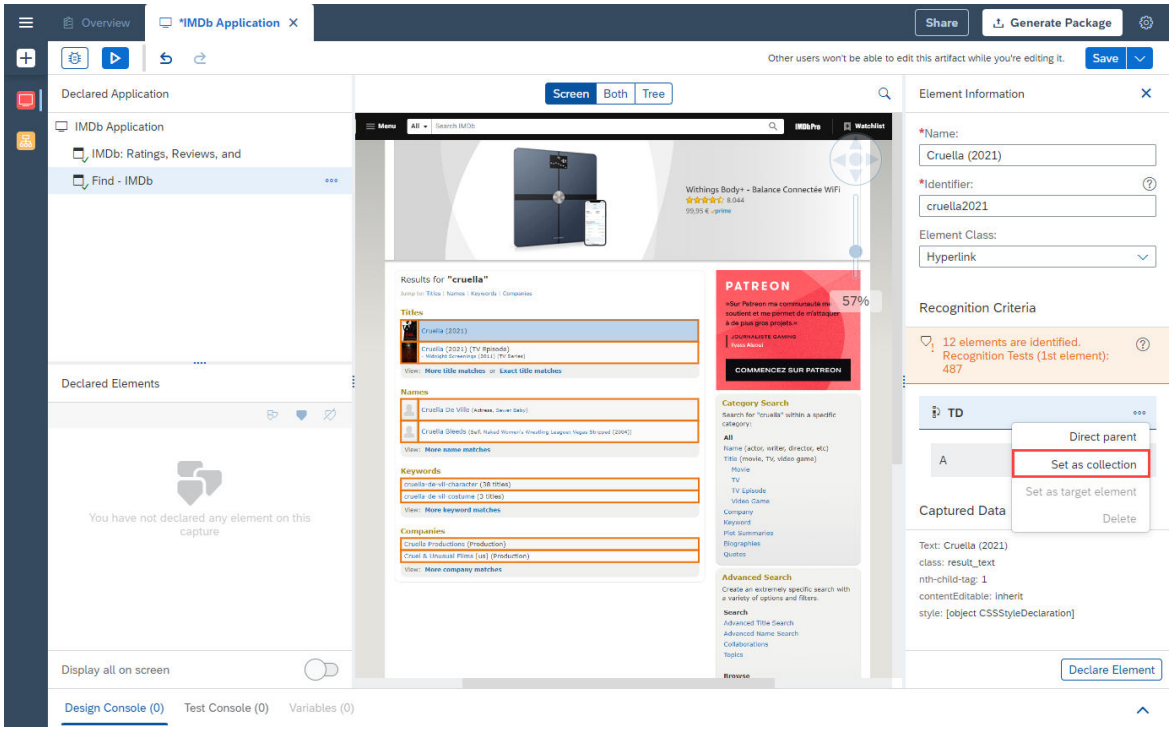
1. Once you have selected the parent element and its child, remove all criteria that have been automatically selected by clicking the ✕ icon.



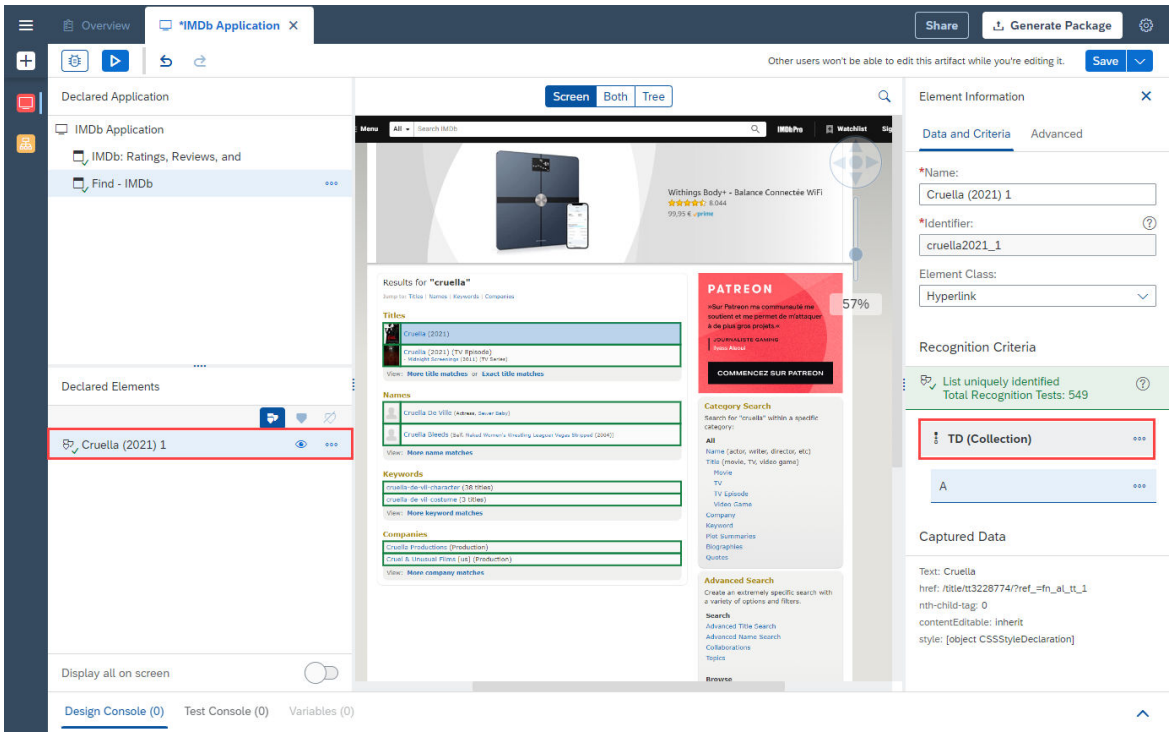
2. On the *Criteria* panel, next to the parent element, click the ••• icon.



3. Click *Set as collection*.

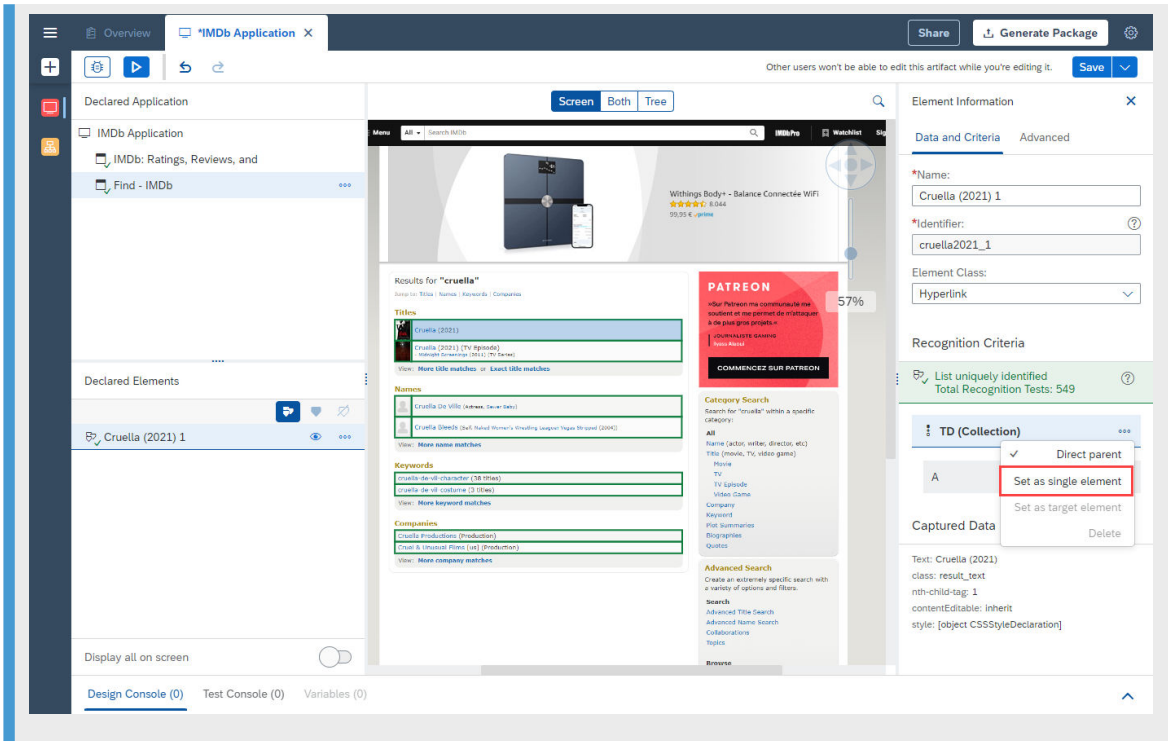


The parent element of the declared element is now set as a collection. As a consequence, the declared element has also became a collection.



Note

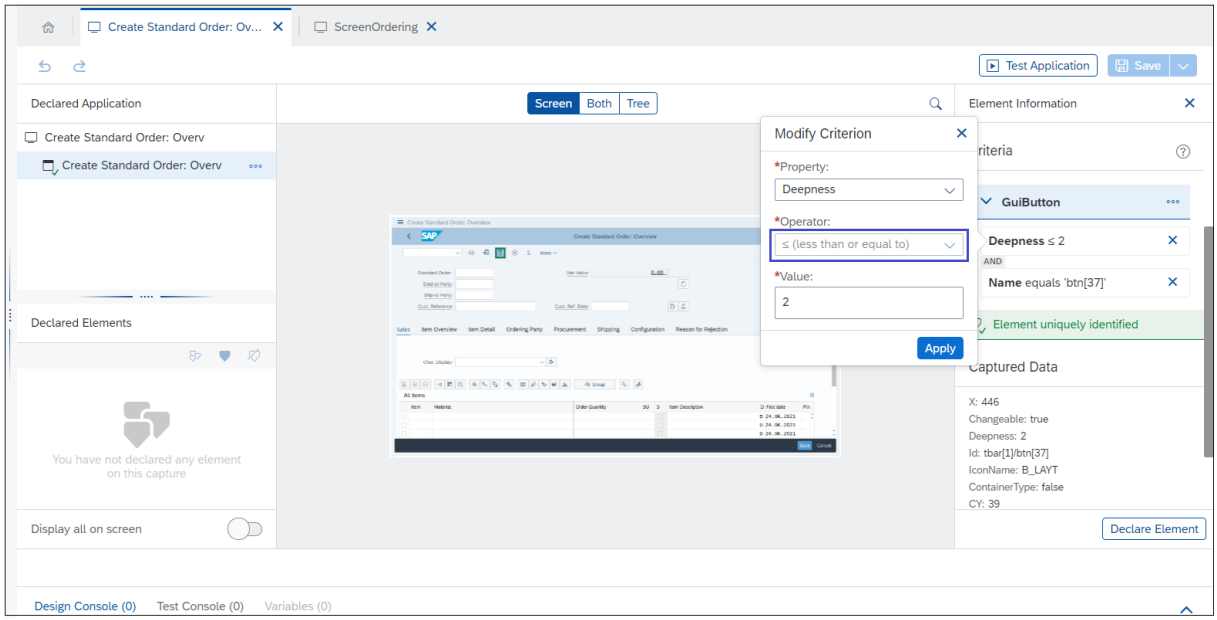
Once you have set an element as a collection, you can undo this action by clicking the ••• icon and selecting *Set as single element*.



4.1.3.3.3.3 Deepness Criteria

After you have captured the application and screens, identify the relevant UI elements. Elements are buttons, input fields, and other controls on a screen. For more details on declaring an element, refer to the Declare an Element [page 110] section.

By default, the *Deepness* criteria is added when you select an element. This is applicable for UI Automation and SAP GUI as these technologies use depth capture data. Deepness criteria can only be used with the less than or equals to (<=) operator.



> **Note**
>
> You cannot edit the *Operator* of *Deepness* criteria.

4.1.3.3.4 Declare an Element as a Collection

A collection consists of multiple or list of elements of same type. For example, a list of titles, URLs, summary, and many more.

Prerequisites

You must capture an application and at least one screen. To know more details about capturing an application screen refer to the Capture an Application [page 77].

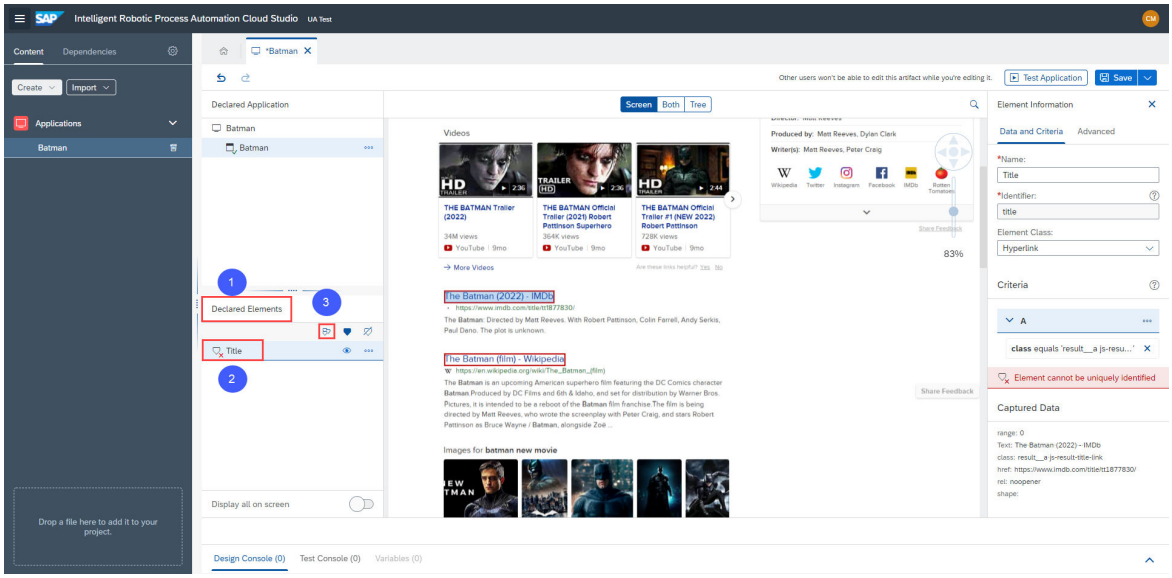
Context

You can declare an element as a collection using the *Is a Collection* button.

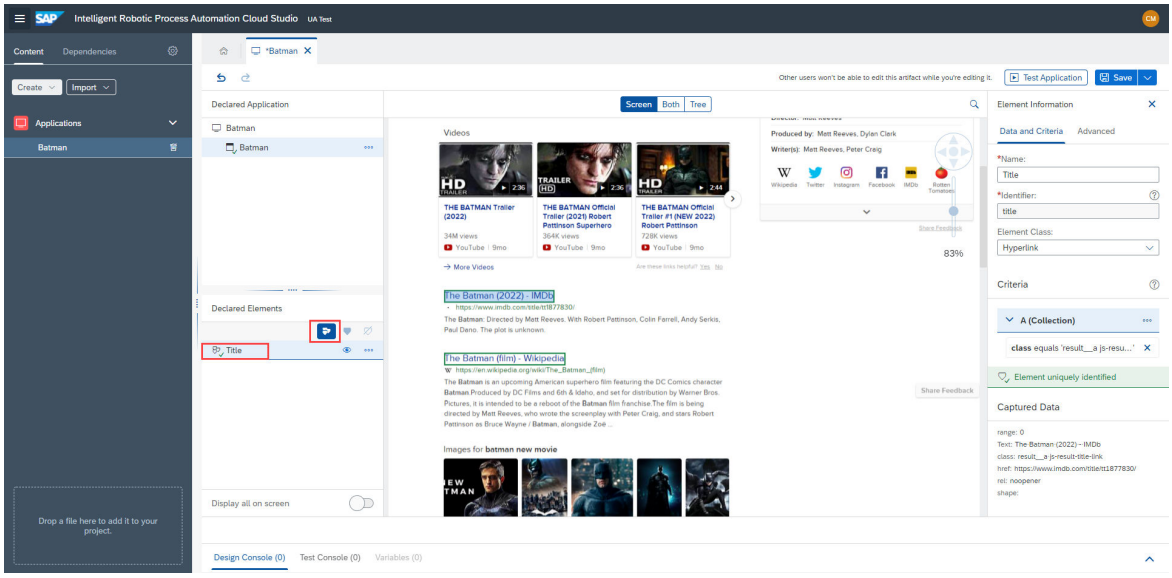
Declare an Element as a Collection Using the Is a Collection Button

1. Declare an element as described in Declare an Element [page 110].

2. Under *Declared Elements* (1) select the element (2) to be declared as collection and then click *Is a collection* button (3).

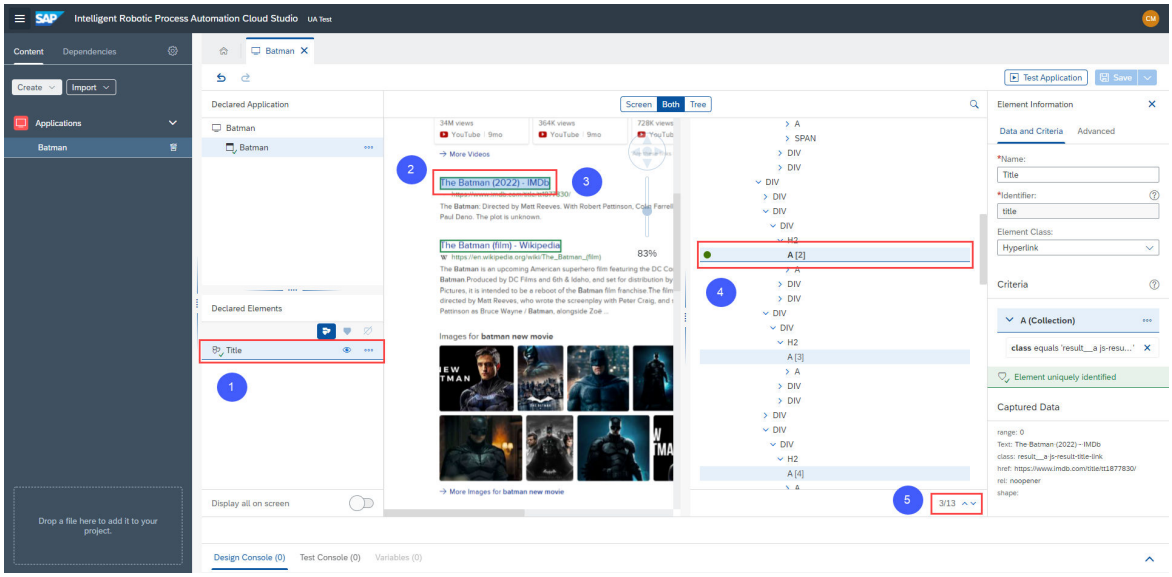


After the element is declared as a collection, the symbol is dislayed next to the declared element and the *Is a collection* button turns blue. By default, all related or similar elements of a screen are added as a collection.



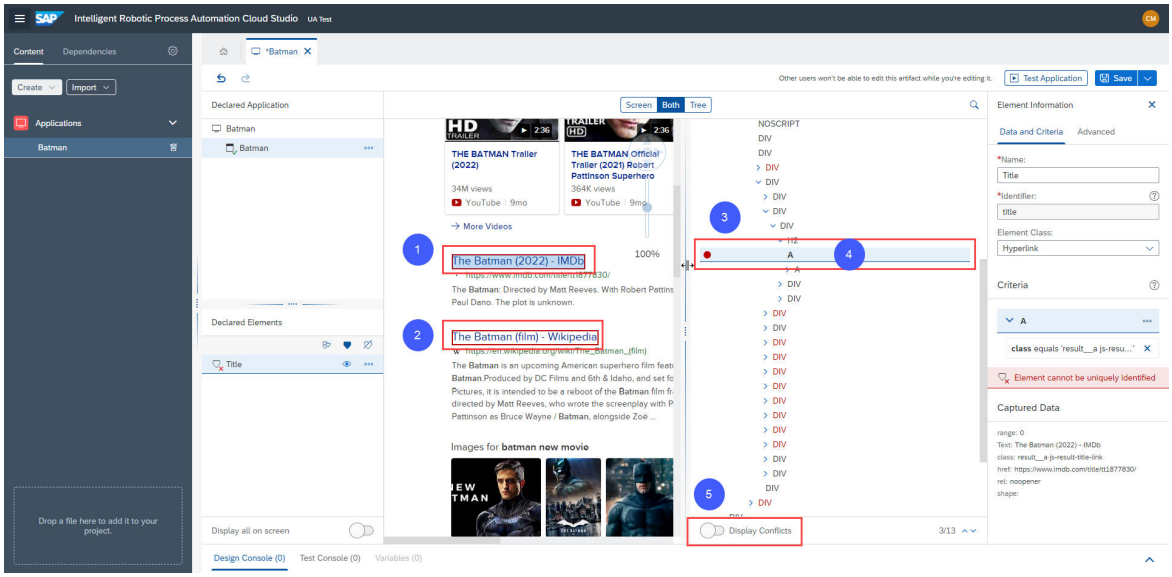
When an element is selected under *Declared Elements*, you can observe that the matched element gets highlighted with green border (2) and the selected element gets highlighted with blue background (3) in the *Screen* central panel. In the *Tree* view, the selected element gets highlighted with a blue background and a blue line at the bottom (4) of the element. You can also find one green dot (4) which refers to the corresponding initial element that you select under *Declared Elements* and then build all *Criteria* accordingly.

> **i Note**
>
> In the *Tree* view, you can move from one matching element to another the arrows (5) at the bottom of the screen.

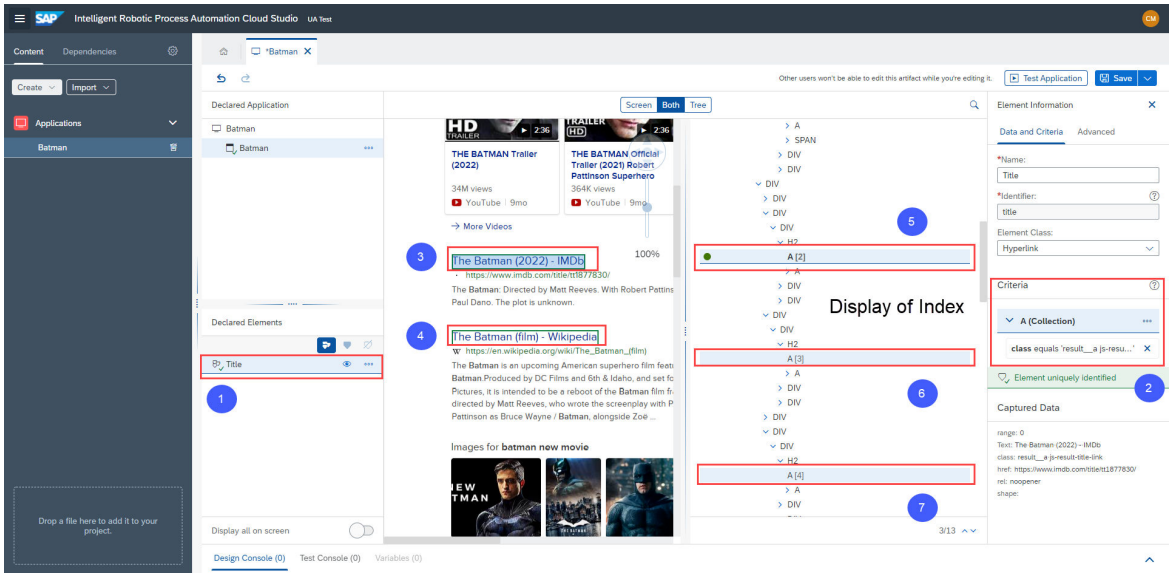


If the selected element under *Declare Elements* is not declared as a collection and it matches multiple elements, the selected element gets highlighted with a blue background (1). All other matched elements that have conflict with the selected element get highlighted with red border (2) in the *Screen* central panel. In the *Tree* view, the selected element gets highlighted with a blue background and a blue line at the bottom (3) of the element and all other elements that have conflict with the selected element become red font (4).

> **i Note**
>
> The *Display Conflicts* toggle button appears, and you can turn it on to expand the whole tree. If you turn this toggle button off, all child elements of the tree collapse.



When an element (1) is declared under *Declared Elements* and a class (2) of the element is selected as a *Criteria*, you can observe that the selected element gets highlighted with a blue background and a green border (3) and the index of all matched elements gets highlighted with a green border (4) in the *Screen* central panel. In the DOM *Tree* view, , the selected element gets highlighted with a blue background and a blue line at the bottom (5) and the index of all matched element gets highlighted with a blue background (6)(7).



> **i Note**
>
> By default, if you create a new capture, all declared element of the screen are linked to the new capture. But, if you declare a new element, it is linked only to the currently displayed capture. So, after declaring new element, you must link it to other captures, if required.

4.1.3.3.5 Advanced Declaration

During the declaration phase, you can't always identify elements with a simple declaration of criteria. You identify these elements with the structure of the page: you add hierarchical elements (parent/child) to their recognition criteria.

Prerequisites

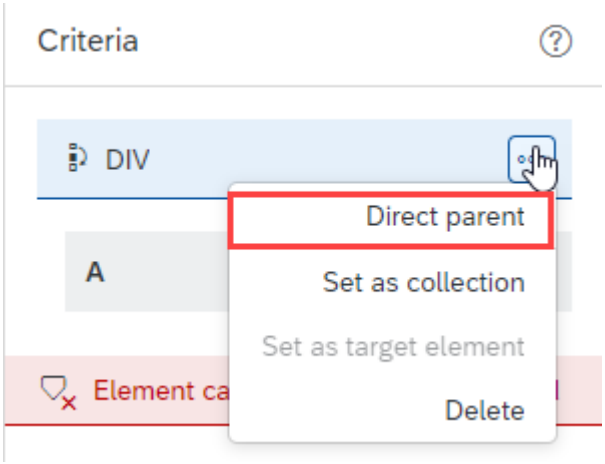
You've captured an application and at least one screen as described in Capture an Application [page 77]. You've also declared a screen and an element as explained in Declare a Screen [page 104] and Declare an Element [page 110].

1. In the *Screen* central panel, click the target element of the captured screen.
2. Switch to the *Both* or *Tree* view.
3. In the tree, hover over the element you want to add as a parent/child element and click ⋯ .

> **i Note**
>
> Similar to the concept of *ancestor* from the V1 release in the *Desktop Studio*, you can also select an element that has already been declared and add this element as parent/child element.
>
> In the *Declared Elements* section, hover over the element you want to add as a parent/child and click ⋯ .

> **Note**
>
> You can't add as parent/child an element that is not a parent/child in the hierarchy tree.

4. Click *Add to criteria*.
   The element is added in the side panel hierarchy as a parent/child element to the criteria of the target element. You can now also modify the criteria for this parent/child element.
5. To specify that a hierarchical element must be a direct parent of the target element, click ••• on this hierarchical element in the *Criteria* panel and select *Direct parent*.

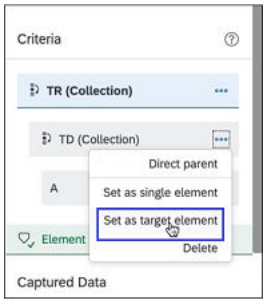


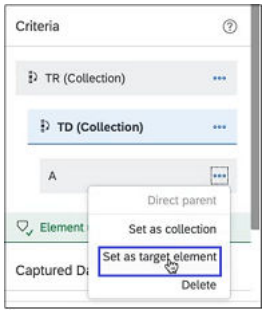
Set as Target Element

You can set a target element from the screen *Criteria*.

You can identify elements in the complex structure of a screen and then define hierarchy for the elements (parent/child) to their recognition criteria. This involves one or more elements to be recognised, where one of these elements is the element to be targeted.

You can edit the defined hierarchy for the elements (parent/child).





Related Topics

- Declaration Phase [page 101]
- Declare Elements [page 107]
- Declare an Element as a Collection [page 120]

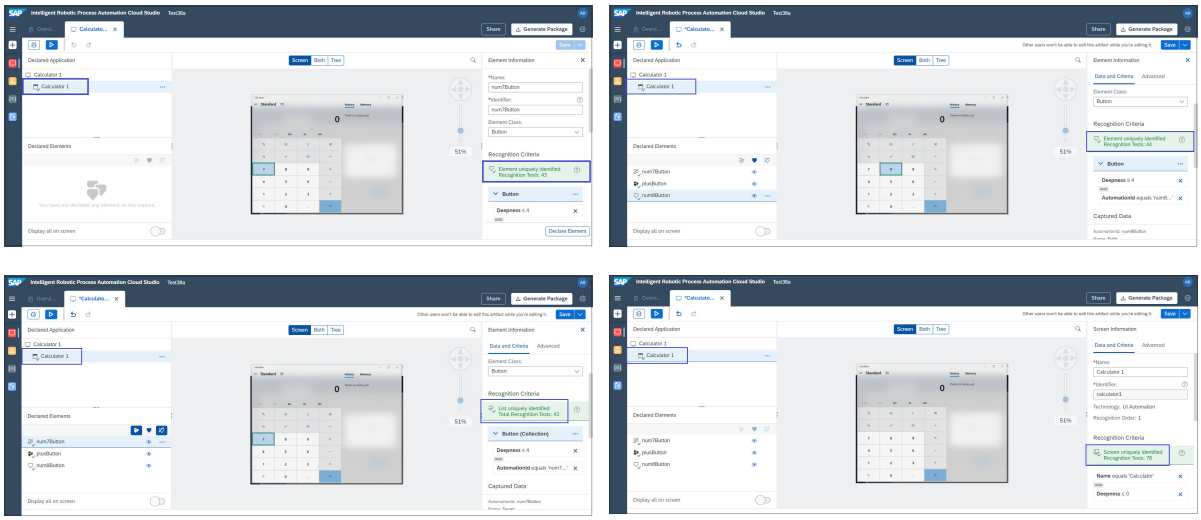
4.1.3.4 Recognition Tests

In the Cloud Studio, you can view the possible recognition tests associated with a captured application screen and declared elements.

Recognition tests are tests performed to identify elements, screens, and sub screens in the Cloud Studio.

To view possible recognition tests, you must select a captured screen or declared element. The possible recognition tests are displayed with a numeric value.

For example, refer to the following screenshots.



> **Note**
>
> The possible recognition tests may vary based on the declared element and its recognition criteria.

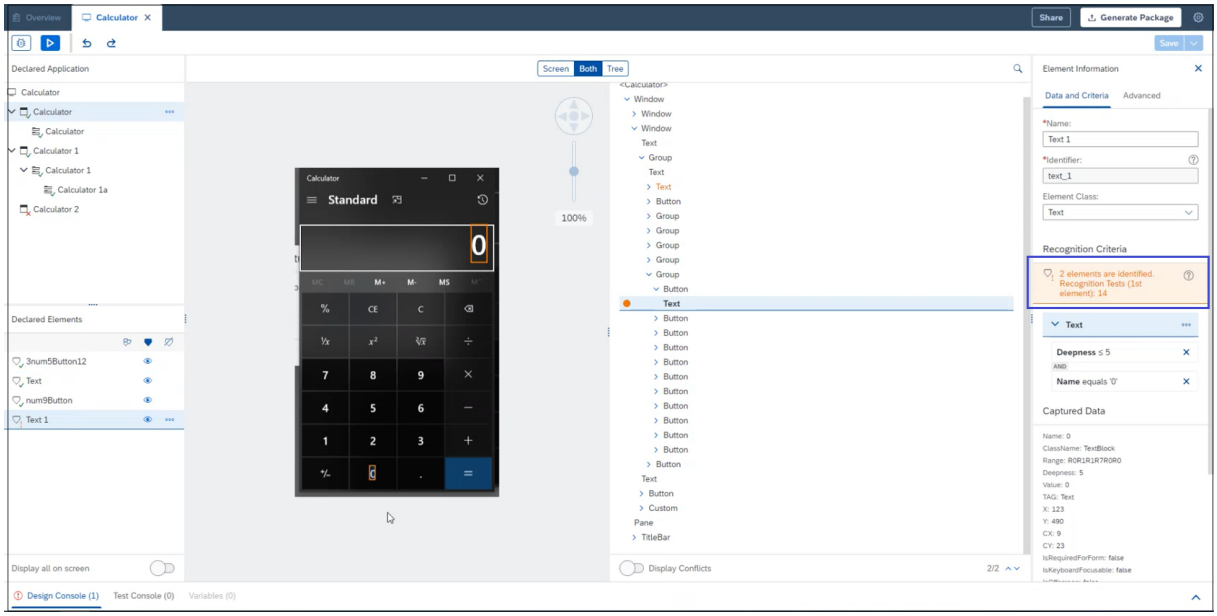
Recognition Details

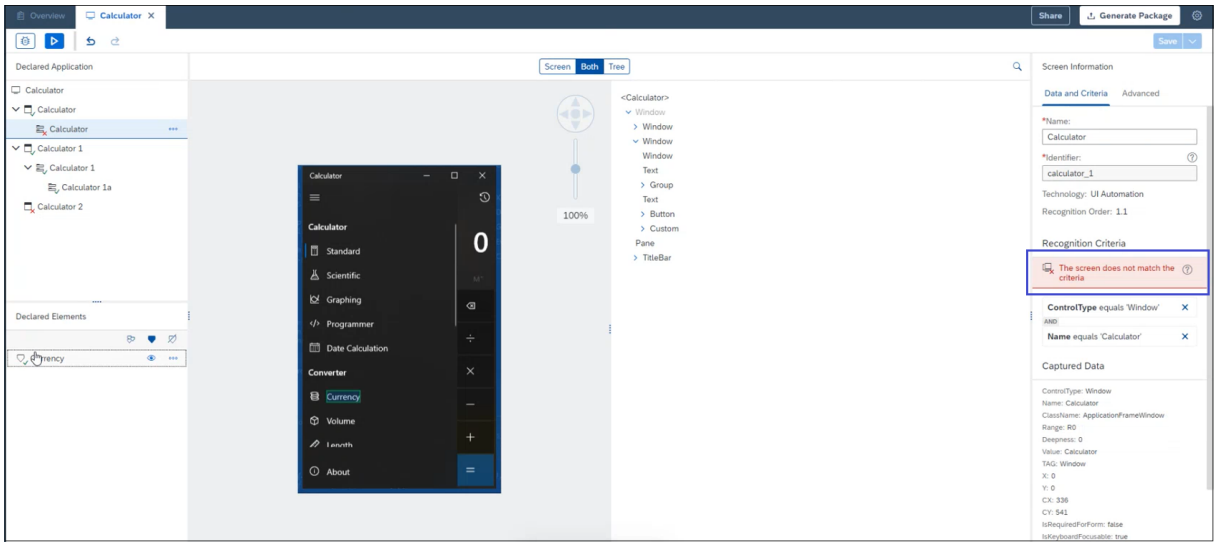
In the Cloud Studio, you can view information about the recognition such as warning type indication for relevant elements, elements that are identified using the same criteria, and list of all identified elements from the criteria.

Recognition of an element depends on the position of the element in the DOM tree. If an element is not uniquely identified in a screen, only the number of tests performed to detect the first element will be displayed.

If you declare a collection, the sum of tests required to identify all instances of the collection will be displayed.

For example, refer to the following screenshots.





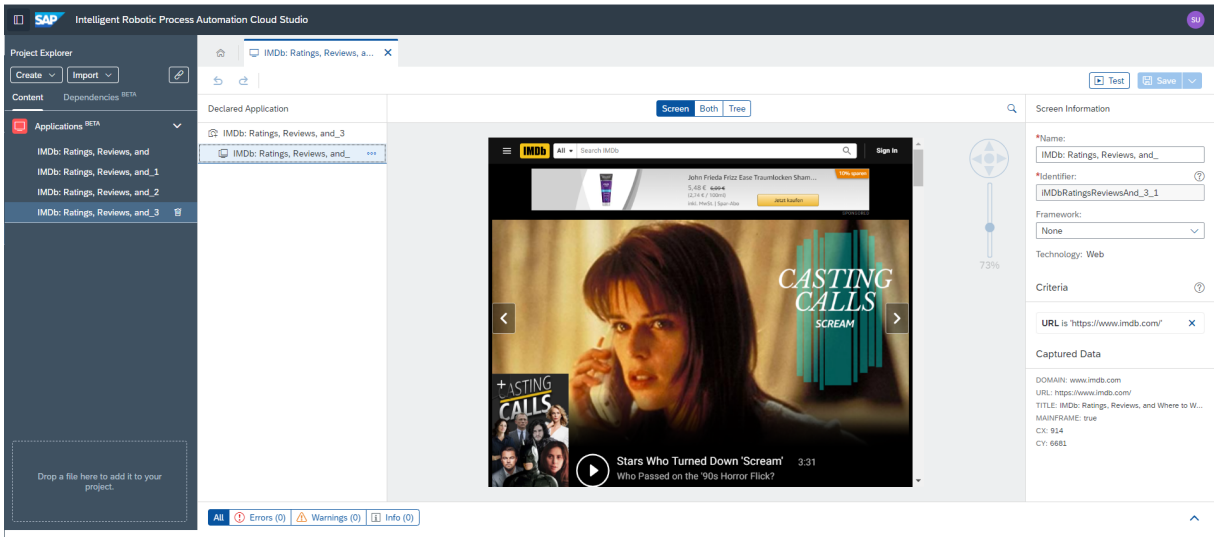
4.1.4 Search Option

4.1.4.1 Search

The search option allows you to find the terms (text, buttons, and input fields) in the screens and tree control types.

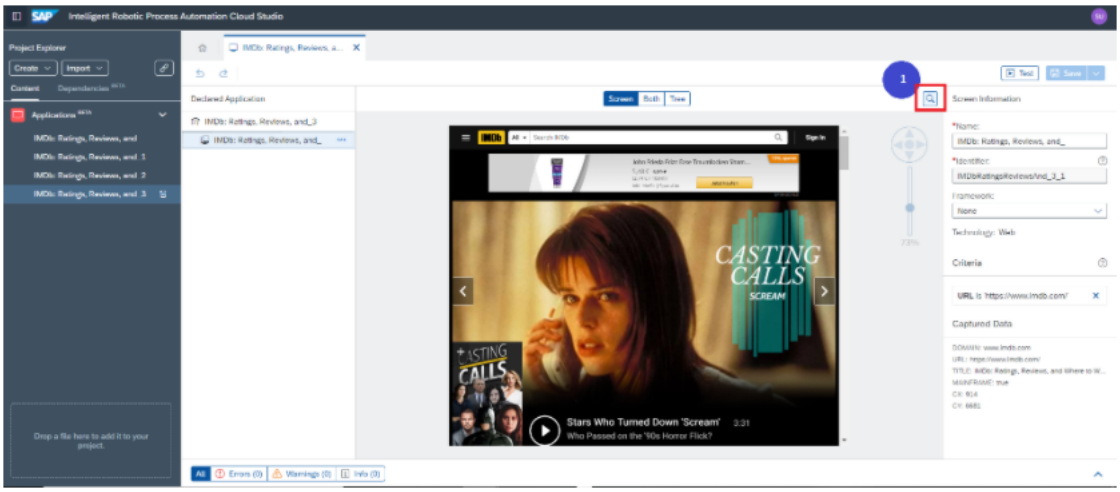
Context

After successful capture of the application screen, a preview screen is displayed in the capture area. For more information about capturing the screen, see the Capture an Application [page 77] section.

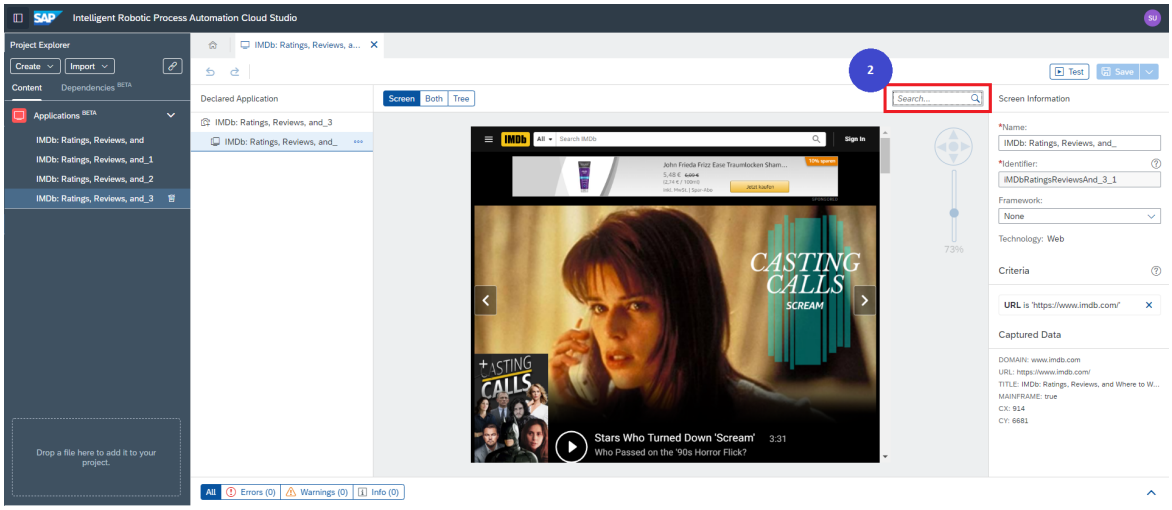


Procedure

1. Click the *Search* icon (1).



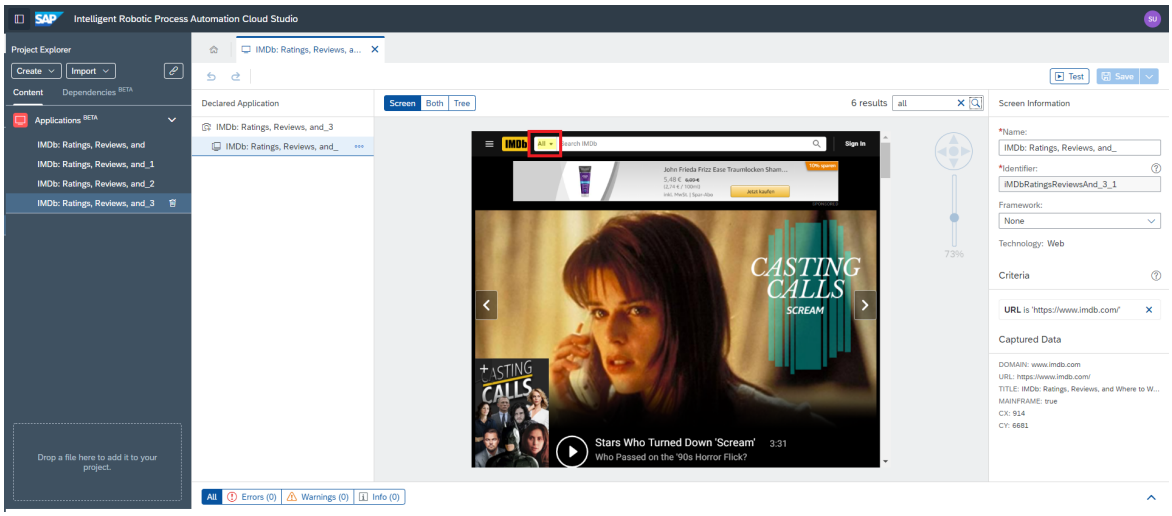
The search field is expanded (2).



2. Enter a term in the *Search* field.
3. Click *Search* or press **Enter**. The search result count is displayed to the left side of the *Search* field.

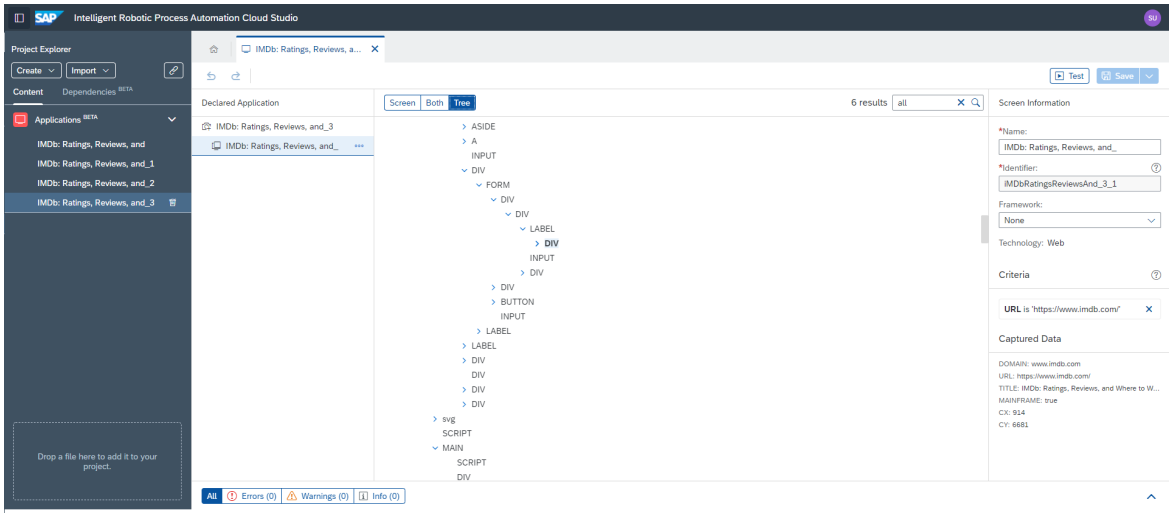
Screen

In the **Screen** view, the search results are highlighted in yellow colour.



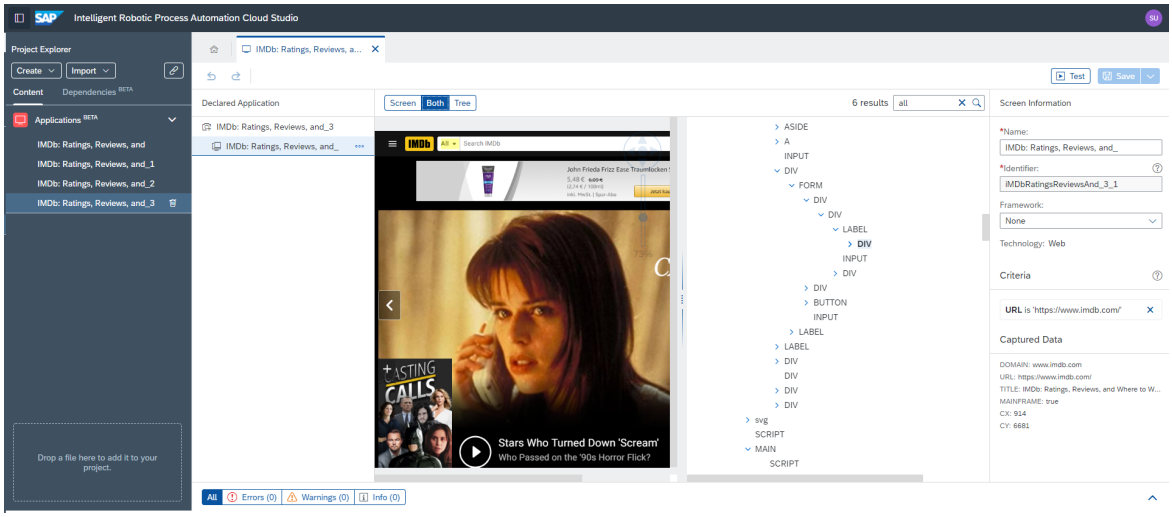
Tree

In the **Tree** view, the search results are highlighted in blue colour.



Both

This view displays both the **Screen** and **Tree** views in split screen mode.



4.1.5 Test an Application

Before you include an application and/or its screens in an automation, you should test it.

Prerequisites

You've captured an application as described in Capture an Application [page 77], and you've declared all the screens and UI elements you want to use in an automation.

You've created an environment in Cloud Factory. For details, see Create an Environment.

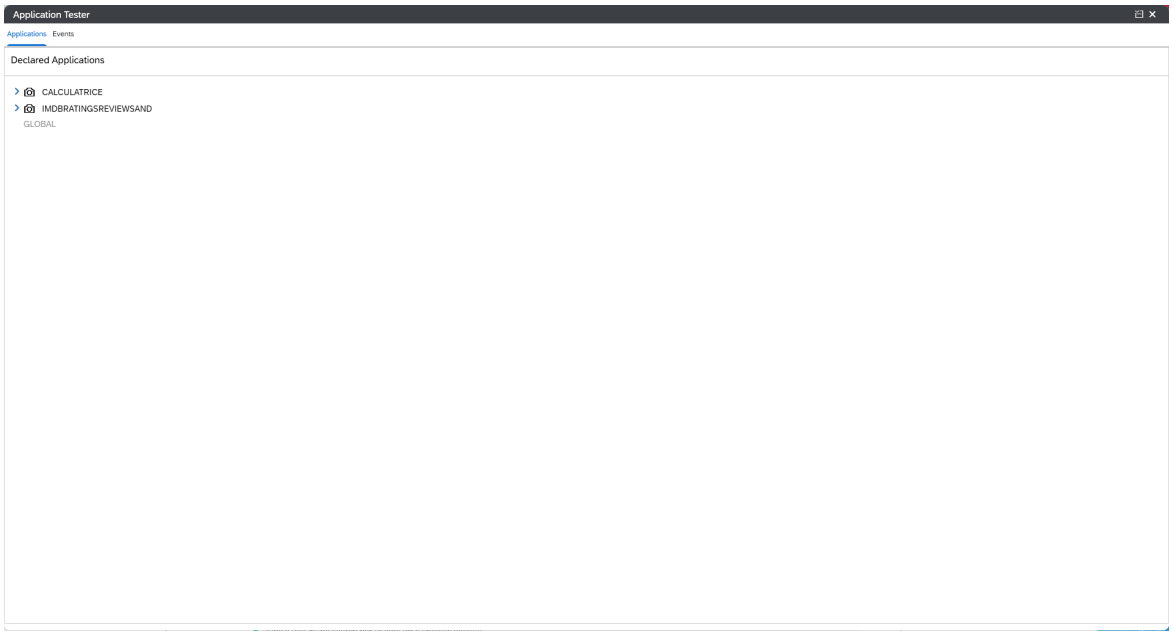
Context

The purpose of testing an application before using it in an automation is to ensure that you have declared all of the elements correctly so that at run time, they are correctly identified and can be controlled as intended.

Procedure

1. In Cloud Studio, open the application and select *Test Application*.
2. Select an environment in the Test Application dialog box.
3. Select *Test*.

   The system generates and deploys a test package. Then it launches the *Application Tester*.



   The applications declared in your project are displayed in a *Declared Applications* section.
4. Click the arrow next to the declared application to unfold the screen(s) to test.

   In case multiple instances of one declared application are open at once, the tester displays the instances as shown in the following screenshot.



   If you have entered different values for each instance, the tester differentiate the instance you select by displaying the corresponding values as input in the *Declared Element* section.
5. Select the screen to test.

The declared elements found on the selected screen are listed in a *Declared Elements* section. If applicable, click on the arrow next to a declared element to display its collection. For more information, see Declare an Element as a Collection [page 120].

If you have created a parent/child hierarchy and added a collection as direct parent to another collection, the elements of the child collection are displayed in sublists on deeper levels, as shown in the following screenshot.



> **i Note**
>
> This beahviour does not apply to web declared applications. For a web declared application, the elements of a child collection are dsiplayed on a single line.

On the right side of each declared element is an activity matching its type (e.g. a *Click* activity for a button), or an input field.

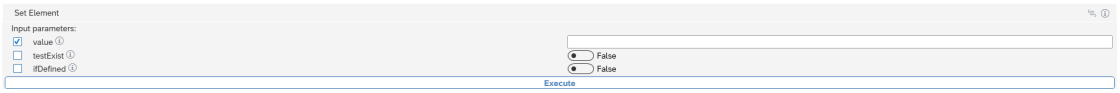
6. Test each element of the screen.
   a. Select an element from the list.

      That element is briefly highlighted with a red border on the corresponding screen of the running application. In addition, a list of activities you can execute on the element is displayed at the bottom of the application tester.

      > **i Note**
      >
      > Type a text in the search bar to look for a specific activity.

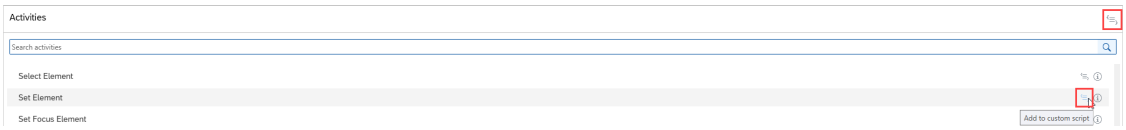
   b. Select an appropriate activity.

      The activities have dynamic inputs with optional parameters you can add to your activity.

      Additionally, if you select an activity that isn't compatible with your element, an error is displayed as an output at the bottom of the application tester.

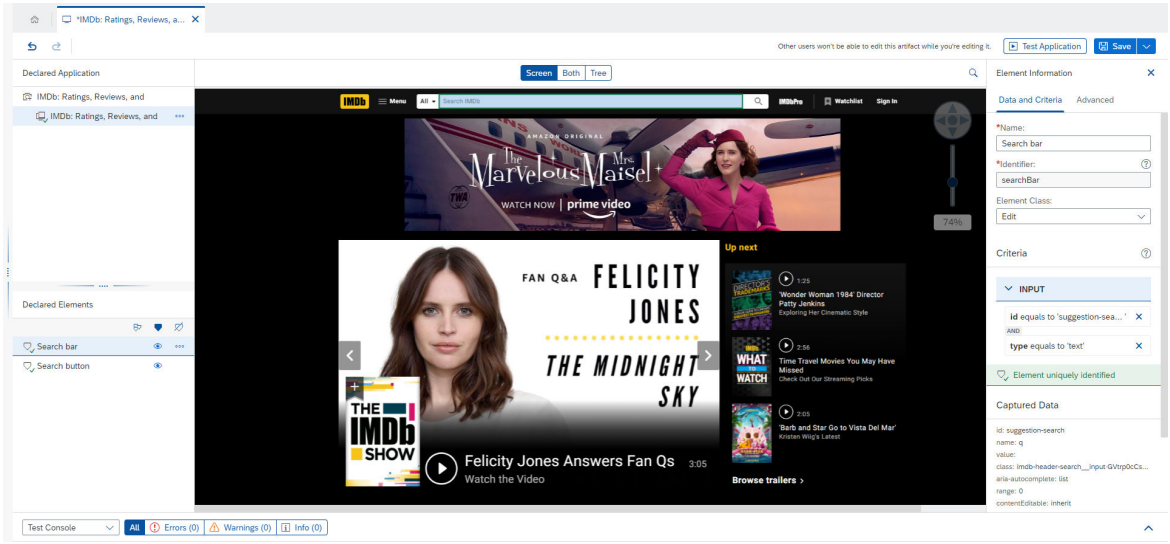
   c. **Optional:** Click the ‹≡› button on the right of an activity to add it to the custom java script.

      You can execute a workflow of activities on a single element by adding the appropriate activities to the custom script. Once you have added the activities to your custom script, click the ‹≡› button on the upper left corner to display the custom script, and click *Execute*.

   d. **Optional:** Switch to the corresponding application to check whether your action was successful.

You've defined an element named *searchBar* in the application's screen. It's an input field:



Make sure that the right screen is open on your desktop. On the application tester, select *searchBar*. In the activities section, select *Set Element*. Enter a value for the inputs and click *Execute*. The value appears in the text bar of the declared element. Switch to the application to check whether your value appears in the search bar.

On the application screen, there is also a button named *searchButton*. Select the *Click* activity on the right side of the search button. If your action is successful, both the application and the application tester will change to a new screen.

7. **Optional**: Select the next screen and perform Steps 5 and 6 until you have tested every screen.
8. Stop testing by closing the application tester window or by returning to the Cloud Studio and selecting *Stop* in the *Test Application* window.

Results

If it works correctly, your tested application is ready to be used in an automation.

Related Information

Add an Application Screen to an Automation [page 214]

4.1.6 Technology Connectors

Capturing and declaring applications means identifying the applications you want to control in an automation. To do this, the Cloud Studio and the Desktop Agent provide a set of technology connectors for controlling purposes.

The Web Connector

For more information, see The Web Connector [page 133].

The UI Automation Connector

For more information, see The UI Automation Connector [page 141].

The SAP GUI Connector

For more information, see The SAP GUI Connector [page 146].

The SAPUI5 Connector

SAPUI5 is an extension of the Web connector. It's a framework that includes a collection of libraries you can use to build applications which run in a desktop or mobile browser – while only maintaining one code base.

4.1.6.1 The Web Connector

Using the Web connector, you can capture HTML pages displayed in web browser tabs. You can use the following web browsers:

- Internet Explorer
- Google Chrome
- Firefox
- Microsoft Edge

Screen Recognition

When the Web connector detects the opening of a Web screen:

- It looks for the first declared Web application with matching criteria.
- If it finds one, it looks for the first declared screen of the application with matching criteria.
- Otherwise, it ignores the Web screen.

Application Instance Management

An application instance is a running instance of an application. The Web connector handles applications instances as follows:

- A Web application loaded in one Web browser tab is considered as one instance of the Web application.
- If a Web application is loaded twice in two different Web browser tabs, the Web connector regards them as two instances of the same Web application.

> **i** Note
>
> If a Web application instance navigates to another Web application in the same tab, the Web application instance is considered to be ended.

4.1.6.1.1 Declaring an Application Using the Web Connector

For general information on this topic, see Declare an Application [page 101].

Set Application Criteria

Application criteria are used by the Web connector as follows:

When the Web connector detects the opening of a Web screen:

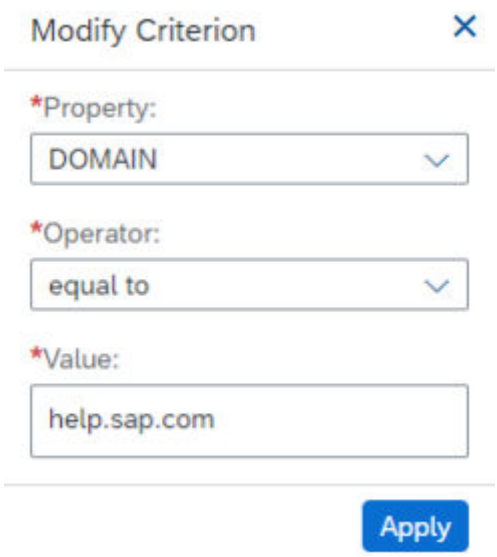
- It looks for the first declared Web application whose criteria match.
- If it finds one, it looks for the first declared screen of that application whose criteria match. If it finds a screen, it associates the new Web screen with that page.
- Otherwise, it ignores the Web screen.

That means that application criteria must be:

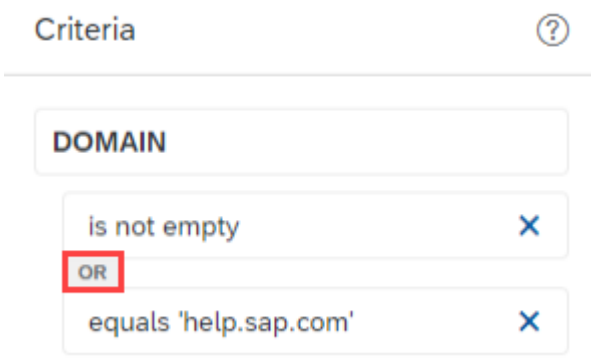
- Broad enough to match all the Web screens of the Web application.
- But not too broad, to ensure they do not match the Web screens of other Web applications.

You set recognition criteria using the Cloud Studio:

- You can use operators to define the value of properties.
  For example, you can use operators to define: **DOMAIN** (*Property*) **equal to** (*Operator*) **help.sap.com** (*Value*).



- You must declare **at least one** recognition criterion.
- You can set multiple criteria on the same property, which are then connected by a logical *OR*.

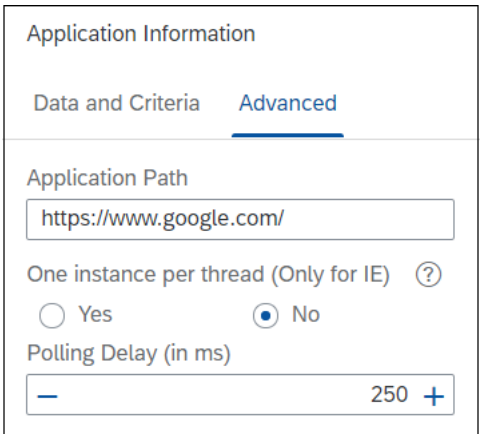


Usually, you set a recognition criterion for the *DOMAIN* property.

Advanced Parameters

You can use one of the following advanced parameters when declaring an application using the Web connector:

One instance per thread (Only for IE)



The Cloud Studio communicates with the connector which instance ID to select.

- Select *No* to use the **Process ID**.
- Select *Yes* to use the **Thread ID** to distinguish different instances of an application.

> **i** Note
>
> This option is used only with the Internet Explorer (IE).

Polling Delay (in ms)

The Cloud Studio waits for an application to check for its readiness or state. This option sets the polling timer frequency in milliseconds (polling mechanism cannot be stopped). By default, the polling delay is 250 milliseconds.

- Click + to increase the polling timer frequency.
- Click - to decrease the polling timer frequency.

4.1.6.1.2 Declaring a Screen Using the Web Connector

For general information on this topic, see Declare a Screen [page 104].

Set Screen Criteria

The Web connector applies screen criteria as follows:

When the Web connector detects the opening of a Web screen:

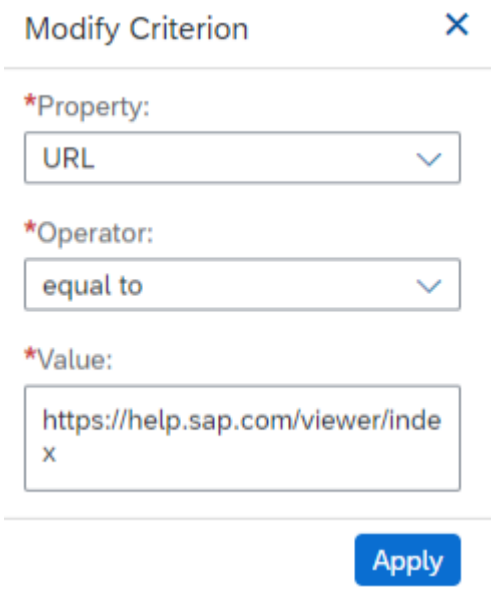
- It looks for the first declared Web application whose criteria match.
- If it finds one, it then looks for the first declared screen of the application whose criteria match. If such a screen is found, the connector associates the new Web screen with that screen.
- And so on.

That means that screen criteria must be:

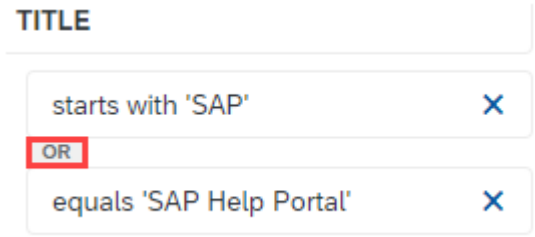
- Broad enough to match the desired Web screen of the Web application.
- But not too broad, to make sure not to match the other Web screens of the Web application.

When you set recognition criteria on a screen using the Web connector:

- You can use operators to define the values of properties. For example, you can use operators to define: **URL** (*Property*) **equal to** (*Operator*) **https://help.sap.com/viewer/index** (*Value*).



- You must declare **at least one** recognition criterion.
- You can use *OR* to define a combination of criteria for each property.

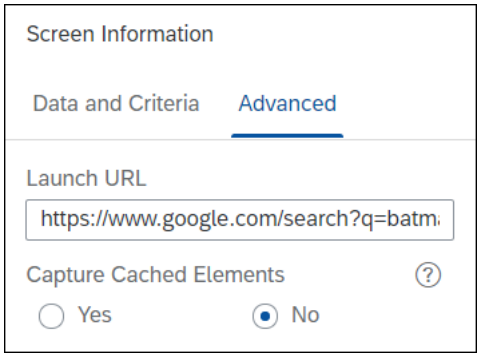


Usually, you will set a recognition criterion on the screen part of the *URL* property.

Advanced Parameters

You can use of the following advanced parameter when declaring a screen using the Web connector:

Capture Cached Elements



This option allows to capture screen elements automatically and their latest value being available for every event. By default, this option is set to *No*.

- Select *Yes*, if screen elements must be captured automatically and their latest value being available for every event.
- Select *No* if you do not want to capture screen elements automatically by the system.

4.1.6.1.3 Declaring an Element Using the Web Connector

For general information on this topic, see Declare an Element [page 110].

Set Element Criteria

Element criteria are used by the Web connector in the following way:

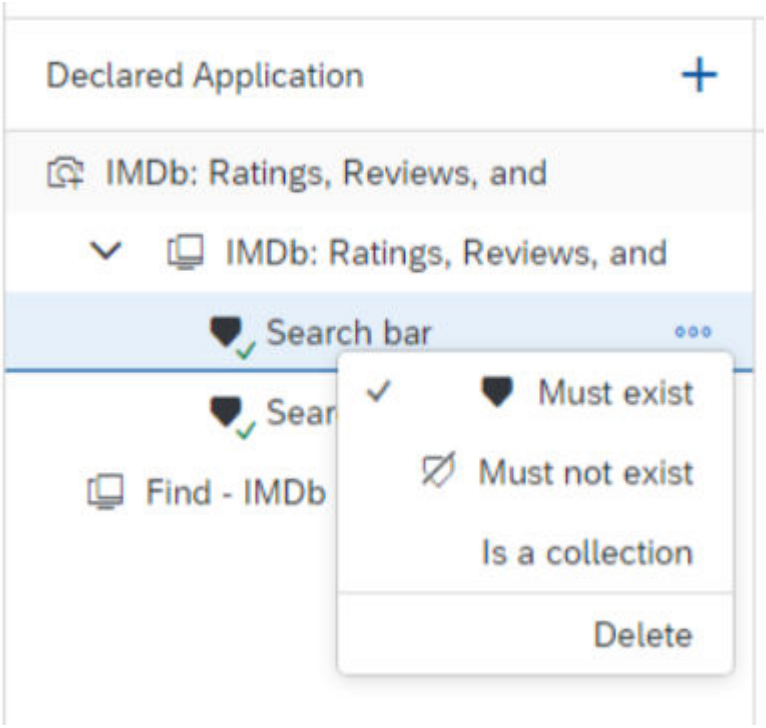
Once the Web connector recognizes a Web screen as a declared screen, it can search the screen DOM to find the target of each element:

- An element targets the first DOM component that matches its criteria.
- If an element is multiple, it will target all of the DOM components that match its criteria.

That means that element criteria must be:

- Precise enough to match the desired component of the Web screen.
- Broad enough to be able to handle as far as possible Web screen evolutions.

- *Must exist* means that the screen is not recognized unless the element or elements defined as must exist are detected.
- *Must not exist* means that the screen is not recognized if an element defined as must not exist is detected.
- *Is a collection* means that the element consists of multiple parts - for example, a list with multiple entries.



When you set recognition criteria on an element using the Web connector:

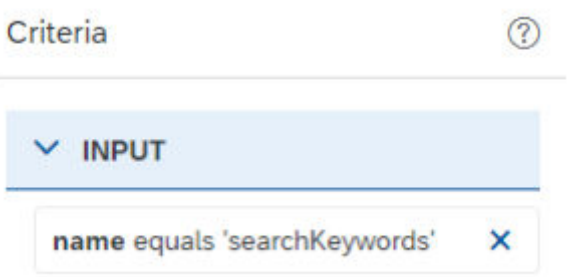
- You can use operators to define the values of properties.
- You must declare **at least one** recognition criterion.Elements are usually declared for controlling purposes. You can also use elements to help screen recognition by defining elements with specific options:
- You can use *OR* to define a combination of criteria for each property.

Usually, you will set recognition criteria for the following properties:

TAG

Elements are usually declared for controlling purposes. The Web connector requires a criterion for the HTML name (**TAG**) of the component: *INPUT*, *DIV*, *SPAN*, and so on. When you create an element from a Web component, the Cloud Studio automatically sets this criterion. It cannot be edited or removed.

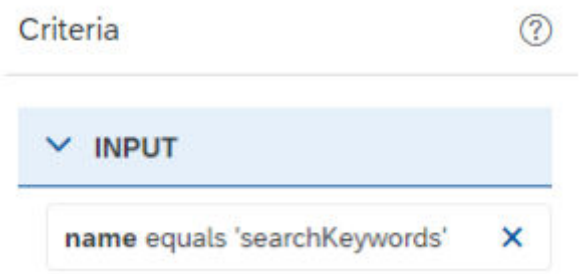
Example:



id or **name**

It is advisable to set a criterion for the properties *id* or *name* of the component, if present and unique. These properties are usually set by Web developers and do not change. When an element is created, the Cloud Studio automatically sets a criterion for one of these two properties if the criterion is suitable.

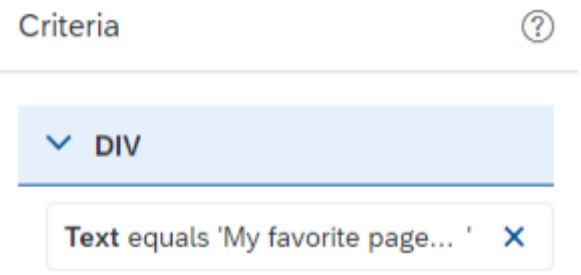
Example:



Text

With some components (such as *SPAN* or *TH*), if the *id* or *name* properties are not usable, you can set a criterion for the *Text* property.

Example:



> **i** Note
>
> This property corresponds to the HTML property, InnerText. So, if a component has a *Text* property equal to **My Text**, all ancestor components also have a text property containing **My Text**.

Advanced Declaration

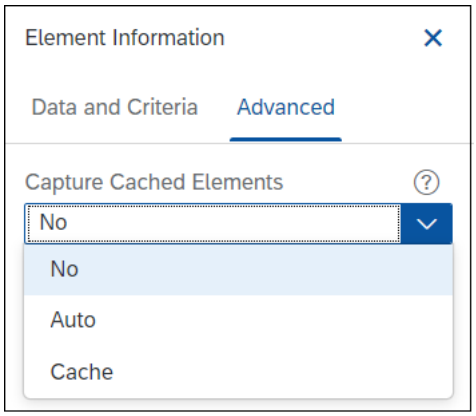
During the declaration phase, you cannot always identify elements with a simple declaration of criteria. You identify these elements with the structure of the page: you add hierarchical elements (parent/child) to their recognition criteria. For example, when declaring an element with a criterion on the nth-child-tag property, you should also declare a direct parent for this element. For more information, see Advanced Declaration [page 123].

> **i** Note
>
> The Range property is replaced with nth-child-tag property in the web connectors.
>
> The Range property is available in old captures.

In an old capture, when you select the range criteria, the *Is a collection* property is disabled.

Advanced Parameters

Capture Cached Elements



By default, the Capture Cached Elements values is *No*.

- *No* - Select *No* if you do not want to retrieve the element values automatically.
- *Auto* - Select *Auto* to retrieve the element values automatically for every event.
- *Cache* - Select *Cache* to retrieve all cached element values in the memory for every event.

4.1.6.2 The UI Automation Connector

UI Automation is a standard protocol developed by Microsoft. It enables applications developed with the following technologies to be controlled:

- Microsoft technologies (such as Win32, WinForm, WPF, or Silverlight).
- Any technologies that support MSAA (such as Qt)

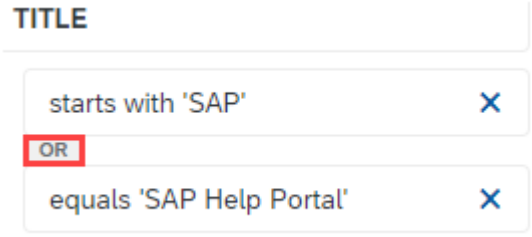
4.1.6.2.1 Declaring an Application Using the UI Automation Connector

For general information on this topic, see Declare an Application [page 101].

Set Application Criteria

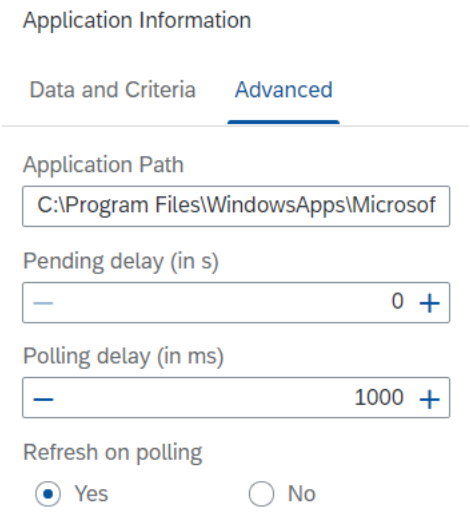
Using the UI Automation connector:

- You can use operators to define the value of properties.
- You must declare **at least one** recognition criterion.
- You can set multiple criteria on the same property, which are then connected by a logical *OR*.



Usually, you set recognition criteria on the *EXE* property (process name of the application). This criterion is automatically set by Cloud Studio when you declare an UI Automation Application.

Advanced Parameters



Pending delay (in s)

This option is used to add delay in seconds while the driver tries recognition on a newly opening window, waiting for its tree being completely built.

Polling delay (in ms)

This option sets the polling timer frequency in milliseconds (polling mechanism cannot be stopped). By default, the polling delay is 1000 milliseconds.

Refresh on polling

This option allows to suspend pages recognition when polling occurs.

- Click *No* if you do not want to suspend pages recognition when polling occurs.
- Click *Yes* if you want to suspend pages recognition when polling occurs.

4.1.6.2.2 Declaring a Screen Using the UI Automation Connector

For general information on this topic, see Declare a Screen [page 104].

Set Screen Criteria

Screen criteria are set on the properties of the root component of the targeted screen. They are used by the UI Automation connector as follows:

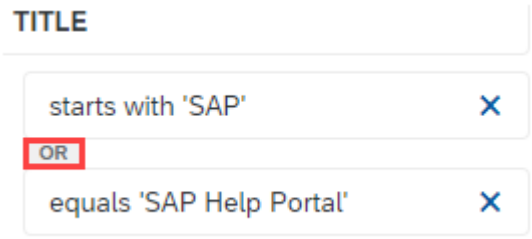
The UI Automation connector searches the DOM tree to find all DOM components matching the screen's criteria. If a component matches, the connector does not search its subtree to find other instances of the screen. The connector creates one instance of the screen for each matching component.

That means that screen criteria must be:

- Precise enough to match the desired root component, and no other.
- Robust enough to still work should the screens change.

When you set recognition criteria on a screen using the UI Automation connector:

- You can use operators to define the values of properties.
- You must declare **at least one** recognition criterion.
- You can set multiple criteria on the same property, which are then connected by a logical *OR*.

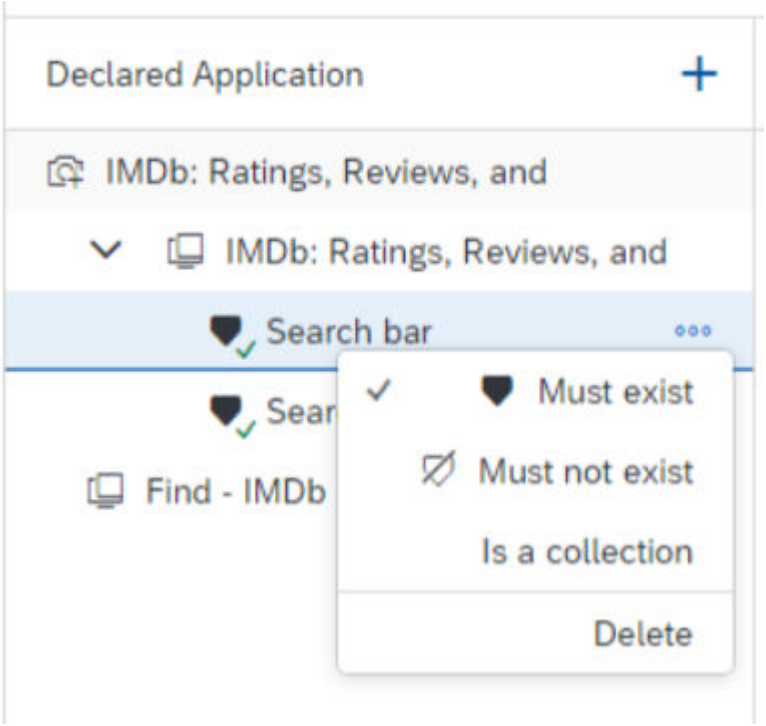


Advanced Recognition Methods for Screens

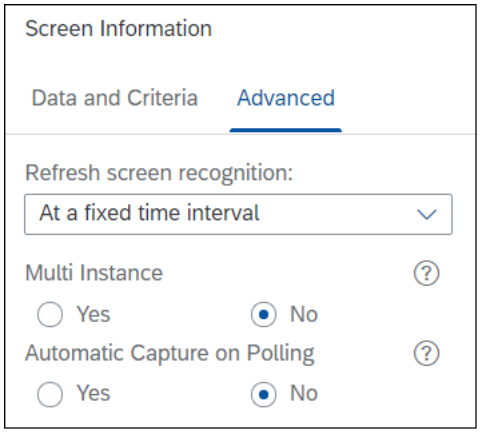
If the targeted root component has no unique distinguishing properties, you can use one of the following advanced declaration methods:

- *Must exist* means that the screen is not recognized unless the element or elements defined as must exist are detected.
- *Must not exist* means that the screen is not recognized if an element defined as must not exist is detected.

- *Is a collection* means that the element consists of multiple parts - for example, a list with multiple entries.



Advanced Parameters



Multi Instance

Indicates if this screen can have multiple running instances. If *No*, only the first running instance will be recognized. Set this parameter as *No* to optimize the recognition.

Automatic capture on polling

This option allows screen elements to be automatically captured on polling, their latest value being available for every event. By default, this option is set to *No*.

Refresh Mode

Defines when screen recognition should be refreshed. The default value can be computed based on the screen root control type, leaving the setting for advanced users.

This is a screen setting that can take the following values:

- *No*: No refresh is needed.
- *Window*: Refresh is triggered by the *WINDOWOPENED* event.
- *Polling*: Refresh is done at a fixed time interval.

4.1.6.2.3 Declaring an Element Using the UI Automation Connector

For general information on this topic, see Declare an Element [page 110].

Set Element Criteria

Element criteria are used by the UI Automation connector as follows:

Once the parent screen connector recognizes a UI component as a declared screen, it can search the component's subtree to find each element's target:

- An element will target the first DOM component that matches its criteria.
- If an element is multiple, it will target all of the DOM components that match its criteria.

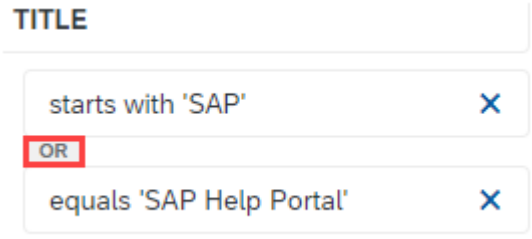
That means that element criteria must be:

- Precise enough to match the desired component.
- Broad enough to still work if the DOM changes.

Elements are usually declared for controlling purposes. You can also use elements to aid page recognition by setting *Must exist* or *Must not exist* parameters.

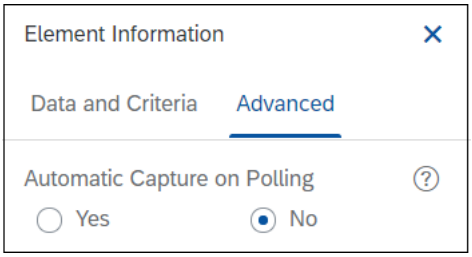
You set recognition criteria using Cloud Studio.

- You can use operators to define the values of properties.
- You must declare **at least one** recognition criterion.
- You can use *OR* to define a combination of criteria on each property.



Advanced Parameters

Automatic capture on polling



This option allows elements to be automatically captured on polling, their latest value being available for every event. By default, this option is set to *No*.

4.1.6.3 The SAP GUI Connector

You can capture SAP applications using the SAP GUI connector. SAP GUI for Windows is a front-end application that you can use to access SAP applications such as SAP ERP, SAP Business Suite, and so on. It is designed for the Windows operating system and provides a Windows-like user experience and integration with other applications based on OLE interfaces or ActiveX controls.

> **i** Note
>
> In some cases the SAP GUI technology of a SAP GUI popup isn't automatically detected. If so, and if the screen belongs to the SAPLOGON.EXE process, the UI Automation is set as technology, and SAP GUI as optional technology. This way, you can force the capture using SAP GUI.
>
> SAP GUI is also available as optional technology for the main SAPLOGON screen, which is not a SAP GUI screen. Please note that if you manually change the technology to select the SAP GUI technology to capture this screen, it will fail.

4.1.6.3.1 Enabling Scripting

For automation purposes, you need to enable and use the SAP GUI Scripting API. You need to enable both client and server scripting. A JavaScript library (SAPScripting.js) is available to implement specific behaviors.

For more information on how to enable the SAP GUI Scripting, see Enabling Scripting on the Client Side and Enabling Scripting on the Server Side.

4.1.6.3.2 Declaring an Application Using the SAP GUI Connector

For general information on this topic, see Declare an Application [page 101].

Declaring the SAPLogon Application

SAPLogon.exe is a multi-technology application.

It includes:

- Windows technology
- SAP GUI technology
- Web technology

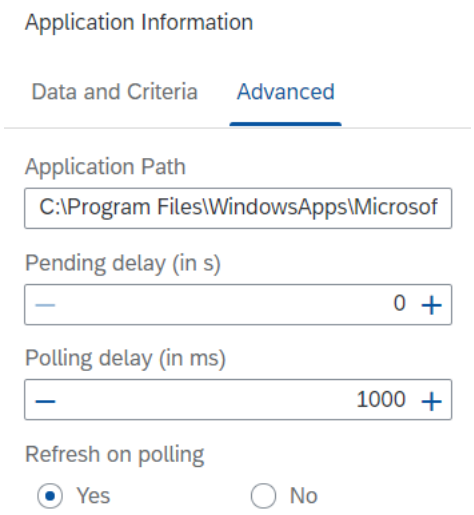
All SAP transactions based on SAP GUI are accessed through the SAPLogon application. This application must be declared using UI Automation technology, by targeting the SAPLogon window (see Declaring an Application Using the UI Automation Connector [page 141]). UI Automation technology lets you declare UI Automation screens, SAP GUI screens, and Web screens.

Once you have declared the SAPLogon application, you need to do the same with the SAPLogon screen.

This declaration activity also requires the following tasks to be performed:

- Declaring a Screen Using the SAP GUI Connector [page 150]
- Declaring an Element Using the SAP GUI Connector [page 152]

Advanced Parameters



Pending delay (in s)

This option is used to add delay in seconds while the driver tries recognition on a newly opening window, waiting for its tree being completely built.

Polling delay (in ms)

This option sets the polling timer frequency in milliseconds (polling mechanism cannot be stopped). By default, the polling delay is 1000 milliseconds.

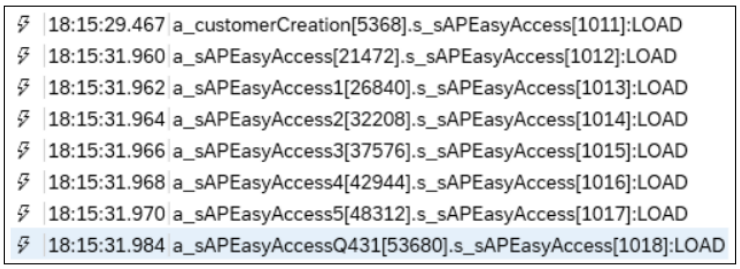
Refresh on polling

This option allows to suspend pages recognition when polling occurs.

- Click *No* if you do not want to suspend pages recognition when polling occurs.
- Click *Yes* if you want to suspend pages recognition when polling occurs.

4.1.6.3.2.1 Best Practices while Declaring Multiple Applications within Same Project

You can declare multiple SAP GUI applications within the same project. When you start the SAP GUI application, each of these declared applications will get a LOAD event as shown in the following screenshot:

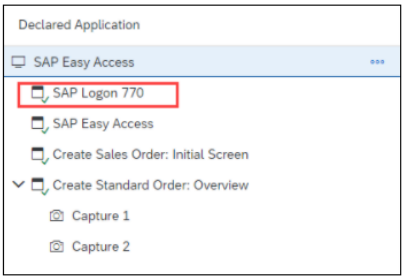


Therefore, you can use these screens declared across different application declarations in single automation.

The following is the best practice we recommend while declaring multiple applications within the same project.

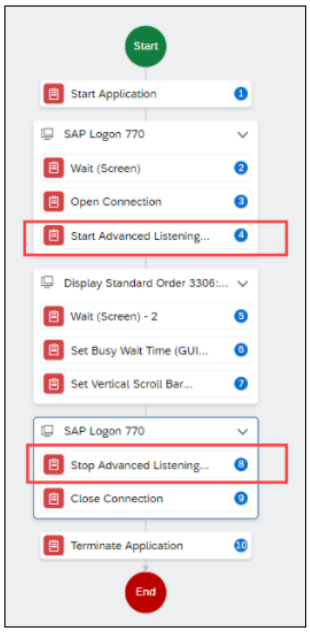
SAP Logon Capture

The *SAP Logon Capture* must exist inside all the declared applications even though they are mutually exclusive sets of screens. If *SAP Logon Capture* does not exist inside any one of the declared applications, there will be issues while generating the UNLOAD event of that application.



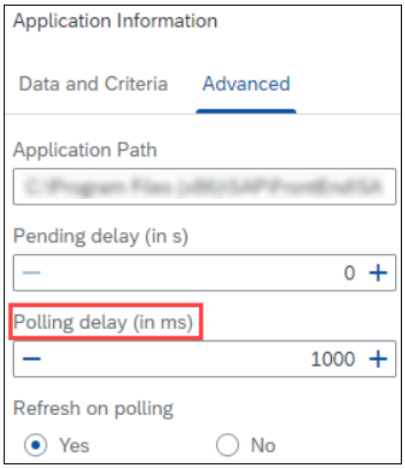
Refresh on Polling

- If *Refresh on Polling* is set to “No” in at least one of the declared applications, the polling of SAP GUI will not start when the project gets downloaded into the agent.
- When the automation starts, the polling can be started or stopped by using the *Start Advanced Listening Screens* and *Stop Advanced Listening Screens* activities on an SAP Logon screen of any declared application.
- You must call the *Start Advanced Listening Screens* and *Stop Advanced Listening Screens* activities only in the beginning and the end of the main automation workflow. It is not recommended to call these activities in a sub-automation or a reusable package from the store since calling the *Stop Advanced Listening Screens* activity in the middle of the automation will UNLOAD all the SAP GUI pages and stop the polling thread.



Polling Delay

The minimum value of the polling delay will be considered across all the declared applications.



4.1.6.3.3 Declaring a Screen Using the SAP GUI Connector

For general information on this topic, see Declare a Screen [page 104].

Set Screen Criteria

Screen criteria are set on the properties of the root component of the targeted screen. They are used by the SAP GUI connector as follows:

The SAP GUI connector searches the DOM tree to find all DOM components matching the screen's criteria. If a component matches, the connector does not search its subtree to find other instances of the screen. The connector creates one instance of the screen for each matching component.

That means that screen criteria must be:

- Precise enough to match the desired root component, and no other.
- Robust enough to still work should the screens change.

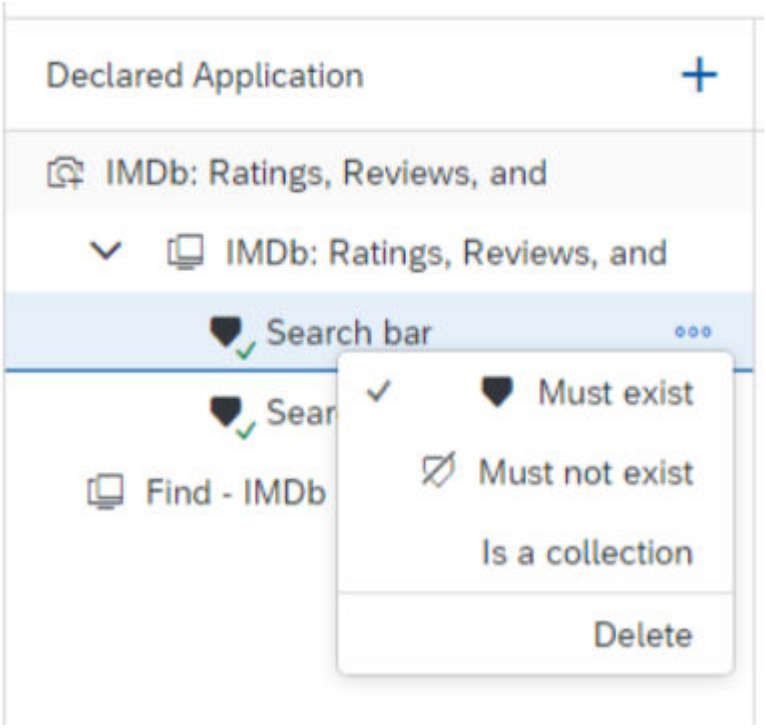
You set recognition criteria using the Cloud Studio:

- You can use operators to define the value of properties.
- You must declare **at least one** recognition criterion.
- You can set multiple criteria on the same property, which are then connected by a logical *OR*.

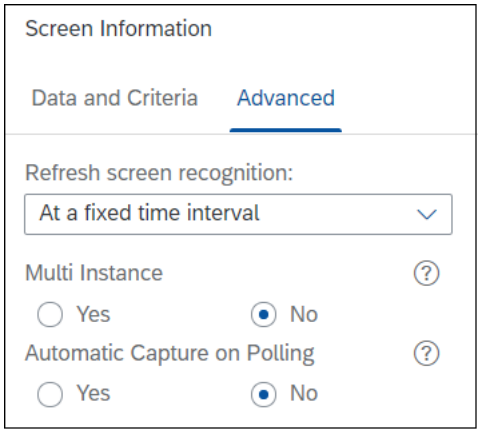
Advanced Recognition Options

If the targeted root component has no unique distinguishing properties, you can use one of the following advanced declaration options:

- *Must exist* means that the screen is not recognized unless the element or elements defined as must exist are detected.
- *Must not exist* means that the screen is not recognized if an element defined as must not exist is detected.
- *Is a collection* means that the element consists of multiple parts - for example, a list with multiple entries.



Advanced Parameters



Multi Instance

Indicates if this screen can have multiple running instances. If *No*, only the first running instance will be recognized. Set this parameter as *No* to optimize the recognition.

Automatic capture on polling

This option allows screen elements to be automatically captured on polling, their latest value being available for every event. By default, this option is set to *No*.

4.1.6.3.4 Declaring an Element Using the SAP GUI Connector

For more general information on this topic, see Declare an Element [page 110].

Set Element Criteria

Element criteria are used by the SAP GUI connector as follows:

Once the SAP GUI connector recognizes a UI component as a declared screen, it can search the component's subtree to find each element's target:

- An element will target the first DOM component that matches its criteria.
- If an element is indexed, it will target all of the DOM components that match its criteria.

That means that element criteria must be:

- Accurate enough to match the desired component.
- Broad enough to still work if the DOM changes.

Elements are usually declared for controlling purposes. You can also use items to aid page recognition by setting *Must exist* or *Must not exist* parameters.

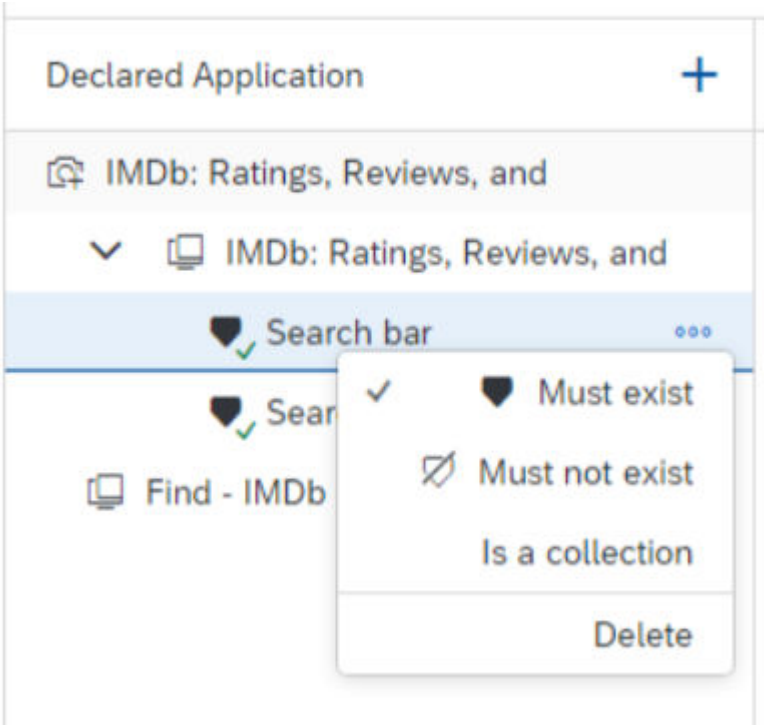
You set recognition criteria using Cloud Studio:

- You can use operators to define the value of properties.
- You must declare **at least one** recognition criterion.
- You can use *OR* to define a combination of criteria on each property.

Advanced Recognition Options

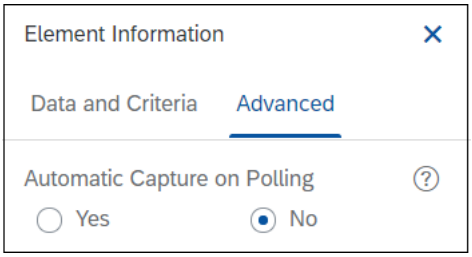
In some cases, it is not possible to recognize a targeted component merely by setting criteria on its properties. Another component is recognized by mistake. To ensure your declaration is correct, you can use advanced recognition options.

- *Must exist* means that the screen is not recognized unless the element or elements defined as must exist are detected.
- *Must not exist* means that the screen is not recognized if an element defined as must not exist is detected.
- *Is a collection* means that the element consists of multiple parts - for example, a list with multiple entries.



Advanced Parameters

Automatic capture on polling



This option allows elements to be automatically captured on polling, their latest value being available for every event. By default, this option is set to *No*.

4.2 Design Automations

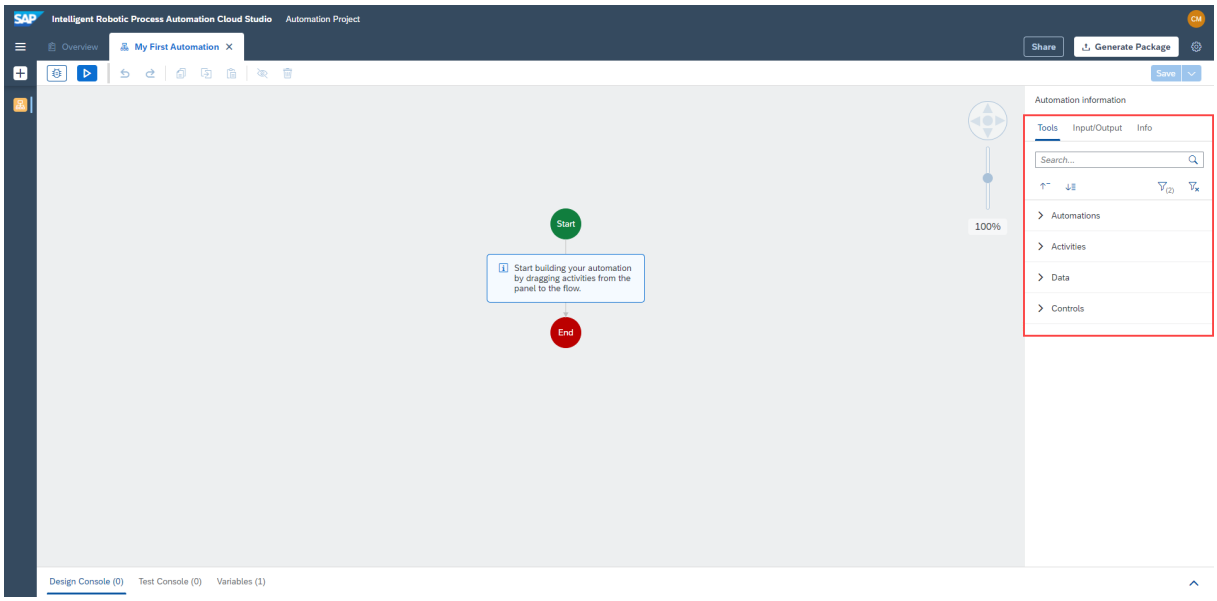
Automations are composed of a succession of steps you build in the Cloud Studio with the automation designer. An *automation* can orchestrate multiple activities on different applications and screens available on a specific computer.

An *automation* is executed on the Desktop Agent of your local machine. Creating an *automation* is similar to designing a *scenario* in the Desktop Studio.

> **Note**
>
> Automations are not available in projects with a Desktop Package.

Tools

To build an automation, you have access to a list of tools from the side panel.



- *Automations*
  You can use a previously created automation from this cloud project, as a tool to include in the flow of another automation.
- *Screens*
  An Application Screen is a capture of any application. For more information, see the section about Capture and Declare Applications [page 75].
- *Activities*
  Activities are used to build the workflow of your automation. They come from the SDK packages imported in the Cloud Studio the first time you create an automation. For more information, see the section about Automation Activities provided by SDK Packages [page 371]
- *Data Types*
  A data type is a complex data used to describe a data structure. It can be defined as an input or an output and used throughout your automation. For more information, see Data Types [page 168].
- *Controls*
  Controls allow you to add tools such as conditions, loops, and scripts to your automation. For more information, see Add a Control to an Automation [page 216].

◇	*Condition*	Inserts multiple situations c

	*End*	Stops the automation.
	*For Each*	Inserts a sequence of actio
	*Forever*	Repeats the step in a loop u
	*Repeat*	Repeats the step for a defin
	*Loop End*	Stops a step set in a loop.
	*Screen switch*	Inserts multiple situations c
	*Custom script*	Inserts a step defined by cu
	*Try*	Checks if errors occur in a s
	*Stop automation in error*	Stops the automation in an

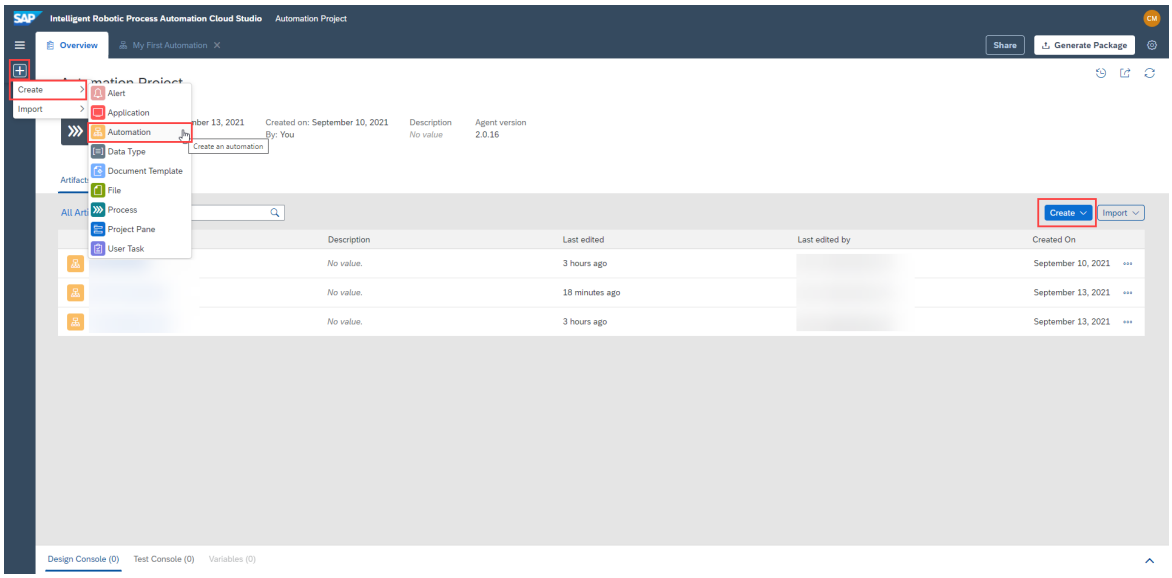
4.2.1 Create an Automation

Context

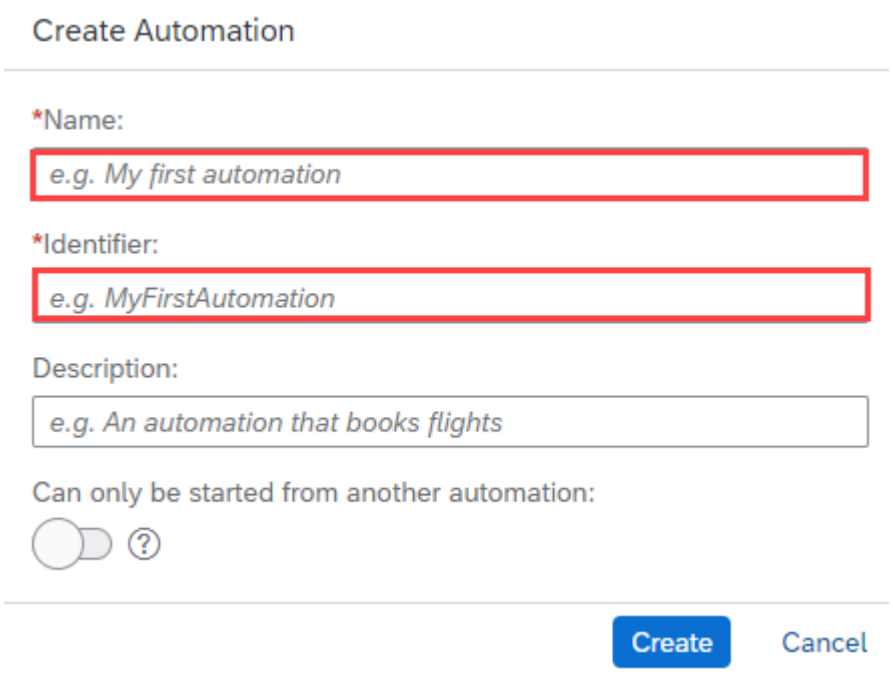
Perform the following steps to create an automation:

Procedure

1. In the Project Explorer, click the + icon, select *Create* and then, *Automation* in the left-hand panel. You can also click the *Create* button in the Project Explorer main panel and select *Automation*.



2. **Optional:** When adding an automation in the project for the first time, you must define the target agent version. For more information, see Configure an Agent Version [page 513].
3. Enter a name in the *Name* field.



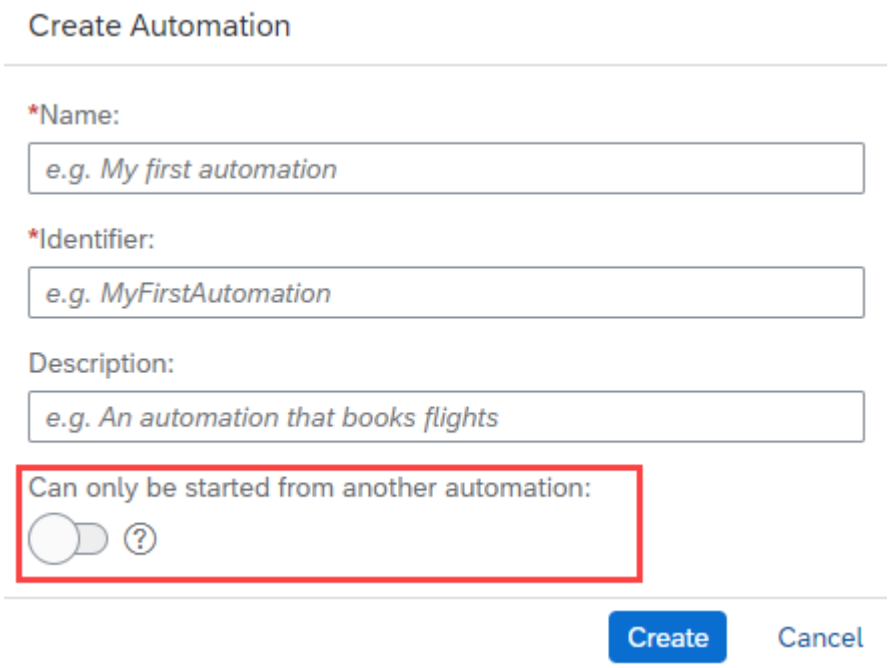
> **i Note**
>
> By default, the identifier is the same as the name. An identifier is mandatory for every artifact in the Cloud Studio, so you need to manually enter an identifier if it's not already pre-filled.

4. **Optional:** Edit the *Identifier* field (without space).

> **i Note**
>
> The identifier is the name of the automation that can be referenced in custom scripting.

5. **Optional:** Enter a short description about the automation in the *Description* field.
6. **Optional:** Enable the *Can only be started from another automation* option. When active, this option prevents you from starting this automation independently. Attended and unattended triggers will not be available. For more information about triggers, see the dedicated section in the Factory User Guide.

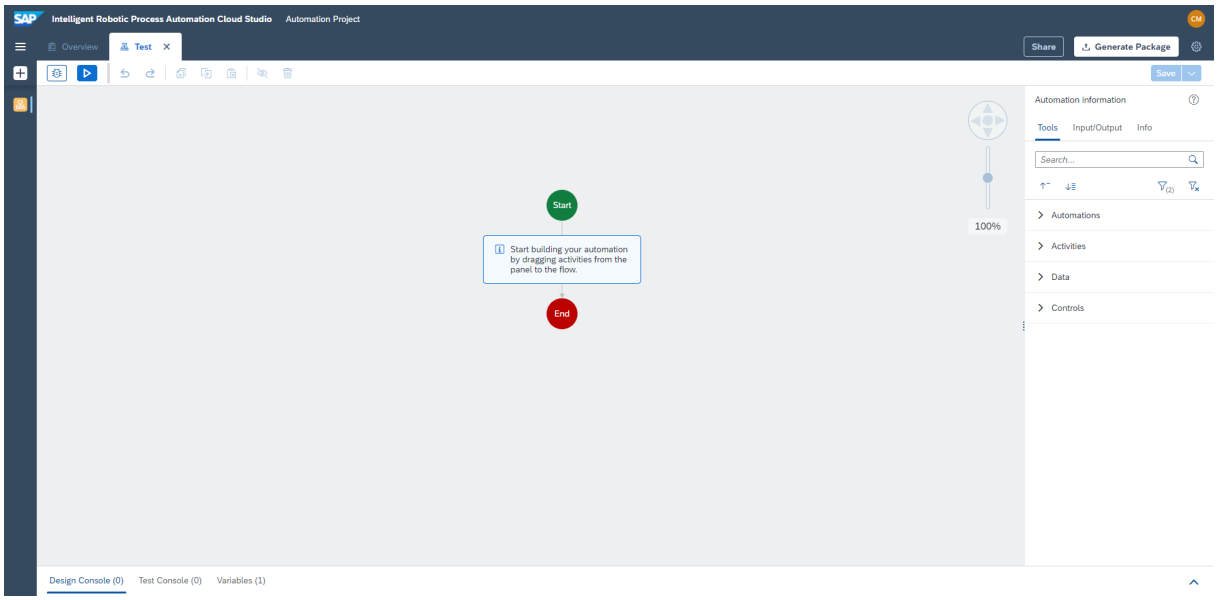
   If you want the automation to run only through another automation inside your project, the *Can only be started from another automation* option must be enabled.



7. Click *Create*. A new tab opens in the main panel of the Cloud Studio.

Results

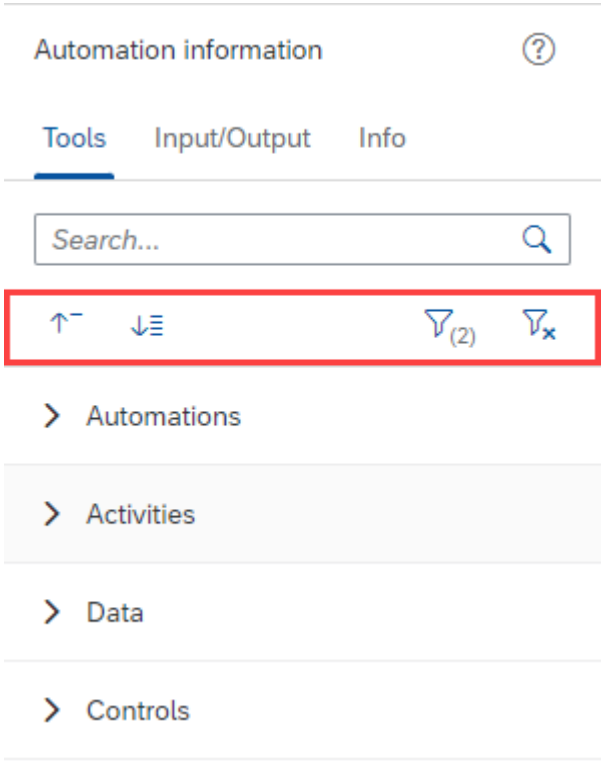
The automation designer opens and you can now drag and drop activities [page 371], data types [page 168], controls [page 216], applications [page 214], and even other automations from the side panel into the workflow to build your automation in the Cloud Studio.



4.2.2 Automation Tools Panel Filters

Automation tools panel filters allow you to find required items like **Automations**, **Screens**, **Activities**, **Data**, and **Controls**.

Automation Tools Filters



Symbol	Name	Description
↑⁻	*Collapse all*	This is used to collapse all items.
↓≡	*Expand*	This is used to expand all items. **i Note** Consider you keep some items open and some close. When you do *Collapse all* and then *Expand*, only the items you kept open earlier are expanded. All items will not be expanded. This option expands all items only at the first instance.
▽	*Filter result*	By default, the filter result is applied.

		This includes two options: • *Contextual filtering* - Filters the result depending on the selection. • *Include dependencies* - This option includes the dependencies.
	Clear filter	This is used to clear *Filter result* filter.

4.2.3 Manage Data within an Automation

1. Input/Output Parameters [page 160]
   In the Cloud Studio, input and output parameters allow you to exchange data in the workflow of your automation between activities, screens, and scripts.
2. Define Complex Input Parameters [page 166]
   You can use input complex parameters by entering the data that defines an object directly in the *Input Parameters* of an automation.
3. Data Types [page 168]
   A *data type* is an artifact describing a data structure that can be used as an input and/or output parameter in automations or processes.
4. Create and Update Variables [page 179]
   The variables you use in the Cloud Studio to build your automation are data storage that has a name, a type (example: string, list of string or data type), and a value. A variable in the automation is also associated to a step represented by its number.
5. Different Types of Data Fields [page 197]
6. Expression Editor [page 199]

4.2.3.1 Input/Output Parameters

In the Cloud Studio, input and output parameters allow you to exchange data in the workflow of your automation between activities, screens, and scripts.

What are Input/Output Parameters?

An input or output parameter is a variable that is passed, received, or sent from one automation, SDK activity or control (for example the condition) to another. This variable allows you to manipulate data that you can use in your workflow. Input or output parameters have a name (optionally a description) and data that complies to a type. Its type can be defined as:

- A simple type variable such as a string, a boolean, a number.

- A more complex type variable such as a *Data Type*.
  For more information on *Data Types*, see Data Types [page 168].
- A variable of type "any".

Input and output parameters can be defined for later use in your project, or for direct use to set and get data. In the Cloud Studio, you can:

- Define the input or output parameters of the automation or custom script your are building.

  > **i Note**
  >
  > When you define an input or output parameter, you create an empty variable with a type and no value.

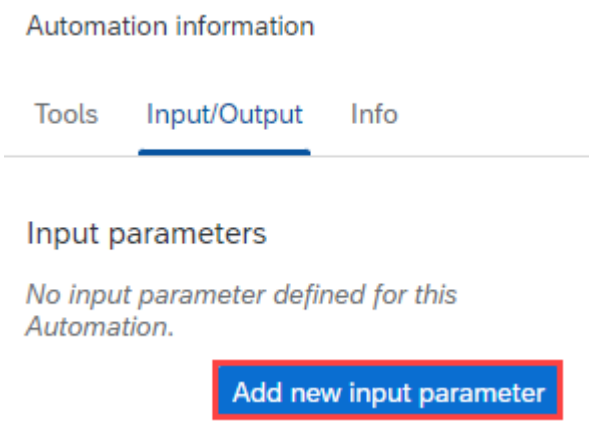
- Set the value of the input parameter, or get the value of the output parameter of an activity, an automation or a control.

  > **i Note**
  >
  > Defining input and output parameters in your automation is optional.

Creating Input/Output Parameters in an Automation

Create an Input Parameter

1. In your automation, on the right-hand side panel, go the *Input/Output* section and click *Add new input parameter*.



2. Enter a name.

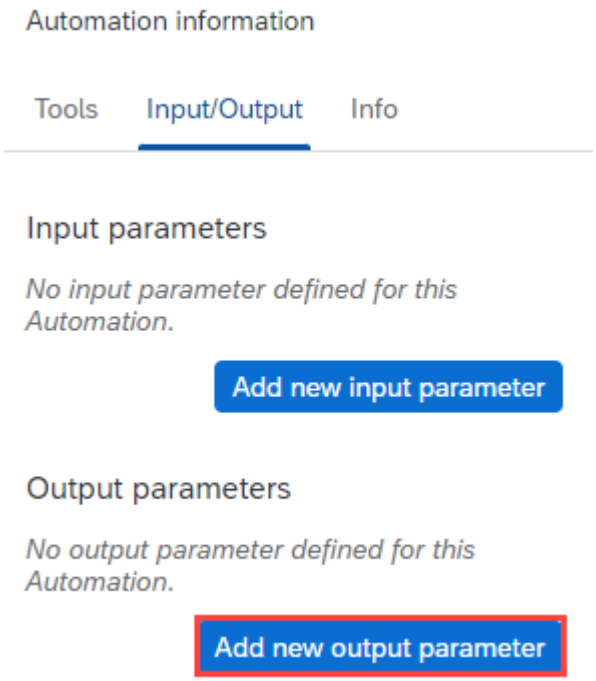
   > **⚠ Caution**
   >
   > You can enter alphabetical and numerical characters:
   > - the first character must be alphabetical
   > - except for the underscore, special characters are not allowed (including blank space characters)

3. **Optional**: Enter a description.
4. Click on the dropdown menu to select a type (simple type or data type).
5. **Optional**: Select the checkbox *List* to have the type presented as an array (for example a list).

6. Click *Save*.

Create an Output Parameter

1. In your automation, on the right-hand side panel, go the *Input/Output* section and click *Add new output parameter*.



2. Enter a name.

> ⚠ Caution
>
> You can enter alphabetical and numerical characters:
> - the first character must be alphabetical
> - except for the underscore, special characters are not allowed (including blank space characters)

3. **Optional**: Enter a description.
4. Click on the dropdown menu to select a type (simple type or data type).
5. **Optional**: Select the checkbox *List*, to have the type presented as an array (for example a list).
6. Click *Save*.

Use Case Example

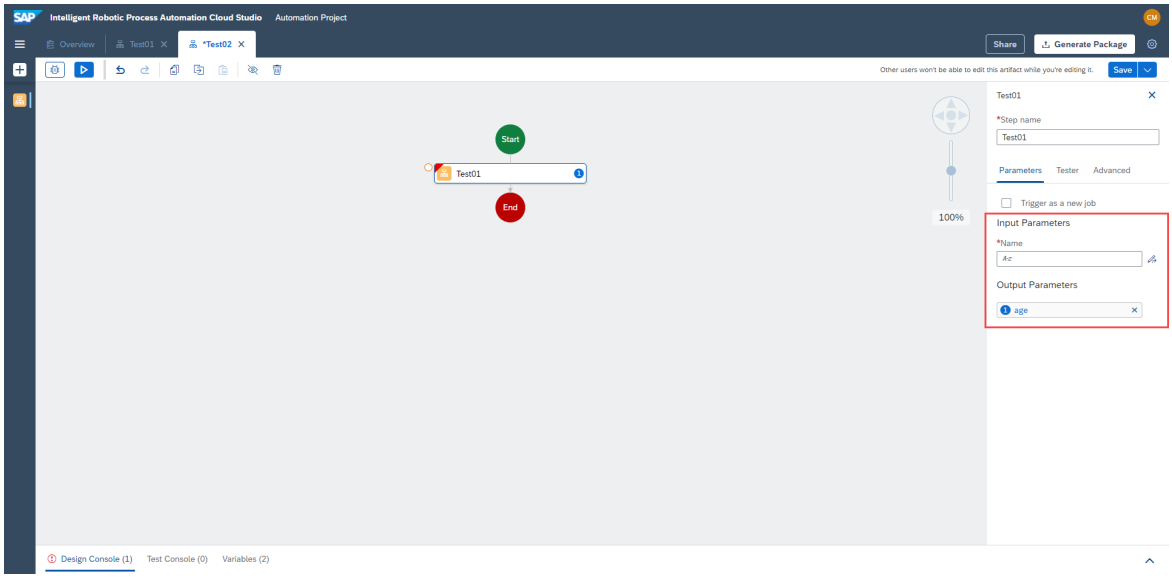
The following use case demonstrates how the data in your automation is passed from one step to another with the input and output parameters.

Prerequisite

You have created the automation Test01.

Procedure

1. In your automation, on the right-hand side panel, go to the *Input/Output* section.
2. Define the input and output parameters of the automation.
    1. Add the input parameter **name** of type string.
    2. Add the output parameter **age** of type number.
3. Create a second automation Test02 and drag and drop Test01 in the workflow of the automation.

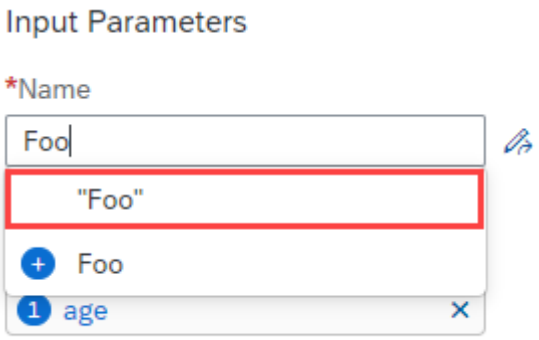


As a result:

- the input parameter of the first step is a name and calls for an input of type string
- the first step returns the output parameter **age**

> **i Note**
>
> You can only set a value of a defined type. For example, in the case of a type string, you cannot set a numeric value.
>
> To set your input parameter to the correct type, you can edit it in the expression editor. For more details, see Expression Editor [page 199].

4. Set a constant value in the input parameters and click on the second value.



Note that the first text suggestion next to the blue mark creates the input parameter for Step0, which is the input parameter of the automation.

> **Note**
>
> In the workflow of the automation, each step has a number. This number indicates from which step an input or output parameter comes from.
>
> 
>

5. Drag and drop the activity *Log*.
6. Set a constant value in the input parameters.

Parent topic: Manage Data within an Automation [page 160]

Next: Define Complex Input Parameters [page 166]

4.2.3.1.1 Reserved Words

You cannot use reserved words as input or output parameter in the Cloud Studio (such as in automation, user task, process, or custom script).

For example, if you use “function” as an input parameter in an automation, the following message is displayed as it is a JavaScript reserved word:

> **⚠ Caution**
>
> Reserved words are not allowed. You need to use a prefix to make the name valid.

Reserved words are categorized into three parts:

1. JavaScript Reserved Words
2. Old JavaScript Reserved Words
3. Other Reserved Words

JavaScript Reserved Words

In IRPA you cannot use the following list of JavaScript reserved words:

abstract	arguments	await	boolean
break	byte	case	catch
char	class	const	continue
debugger	default	delete	do
double	else	enum	eval
export	extends	false	final
finally	float	for	function

goto	if	implements	import
in	instanceof	int	interface
let	long	native	new
null	package	private	protected
public	return	short	static
super	switch	synchronized	this
throw	throws	transient	true
try	typeof	var	void
volatile	while	with	yield

Old JavaScript Reserved Words

In IRPA you cannot use the following list of old JavaScript reserved words:

abstract	boolean	byte	char
double	final	float	goto
int	long	native	short
synchronized	throws	transient	volatile

Other Reserved Words

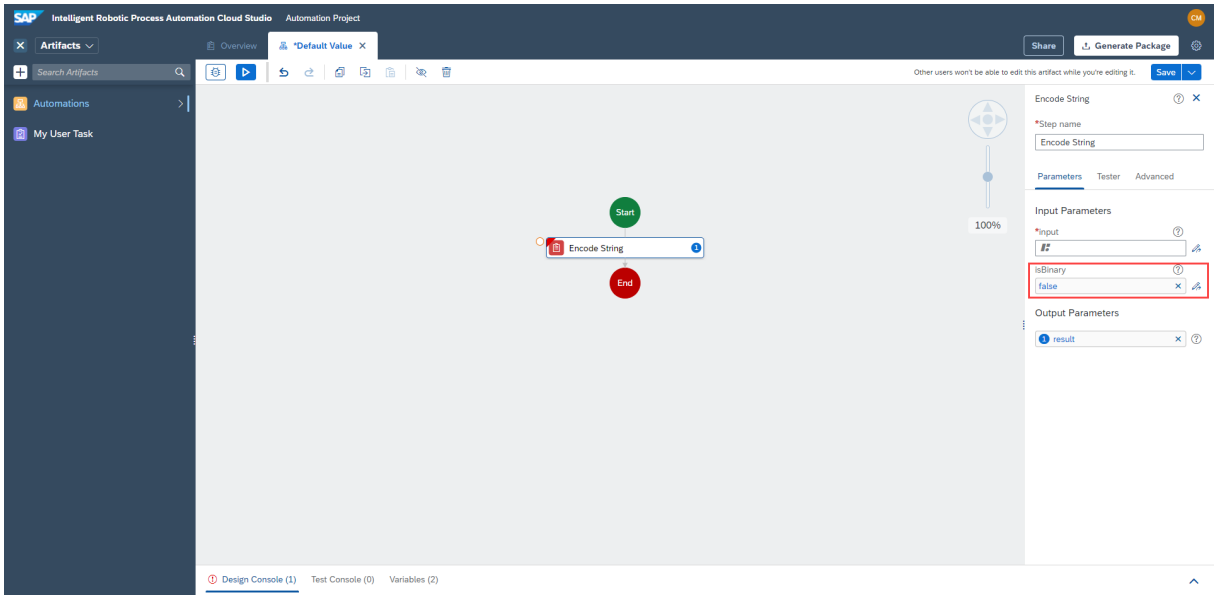
You must avoid using the following list of reserved words that are comprised of JavaScript and SDK built-in objects, properties, and methods:

Array	Date	eval	function
hasOwnProperty	Infinity	isFinite	isNaN
isPrototypeOf	length	Math	NaN
name	Number	Object	prototype
String	toString	undefined	valueOf
irpa	require		

4.2.3.1.2 Activity Default Value

For some activities in the Cloud Studio, input comes with a default value. It means that the default value will be used if the user does not provide any other value.

You can see a default value of an activity at design level. Refer to the below screenshot.



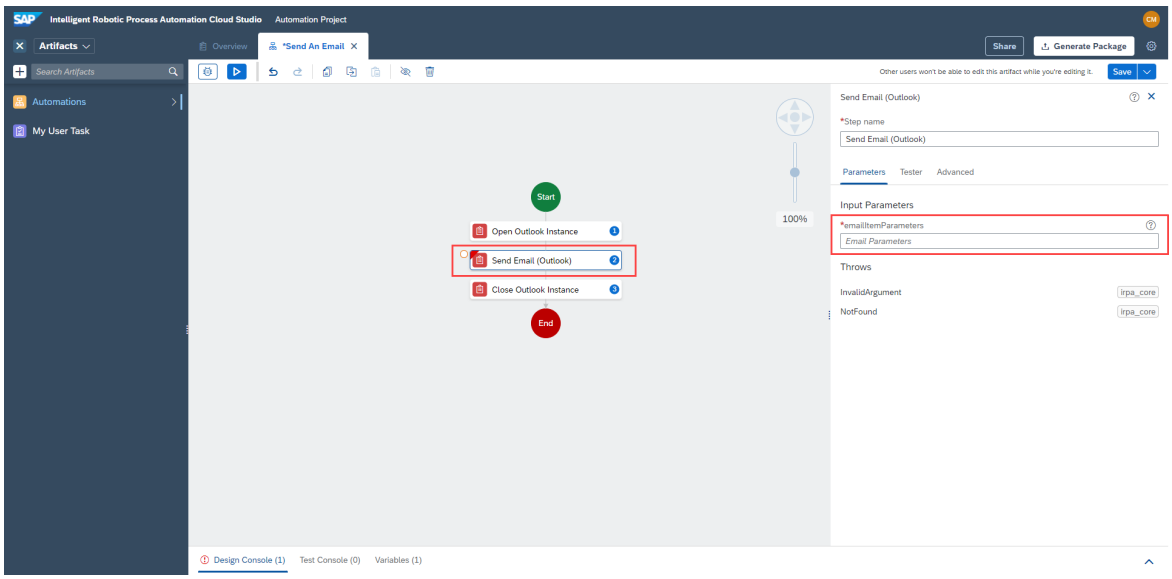
At any point in time, you can update the default value with an appropriate value.

4.2.3.2 Define Complex Input Parameters

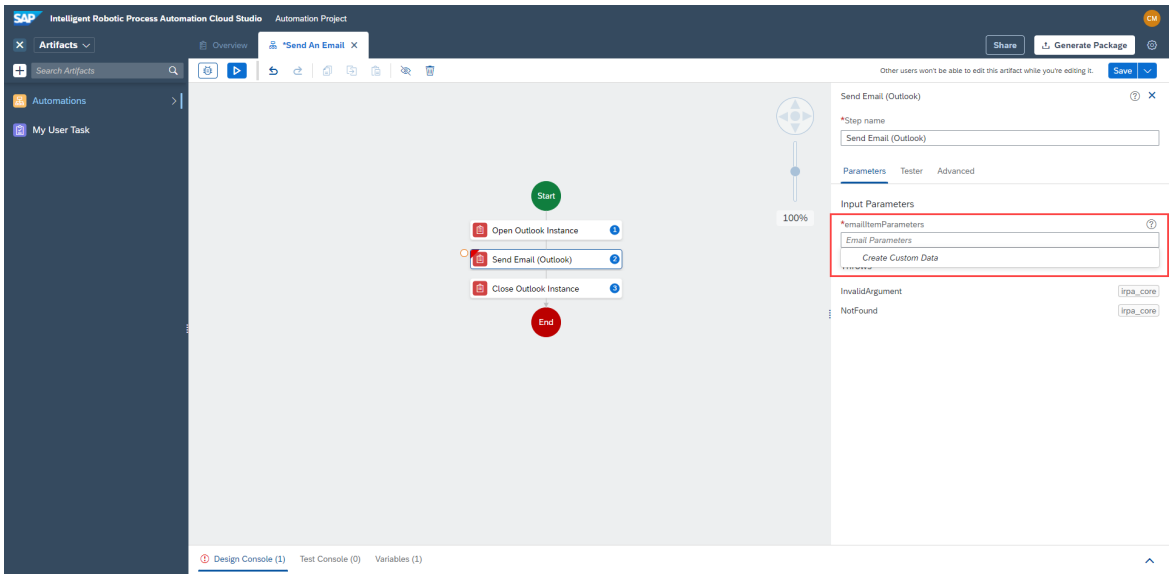
You can use input complex parameters by entering the data that defines an object directly in the *Input Parameters* of an automation.

Procedure

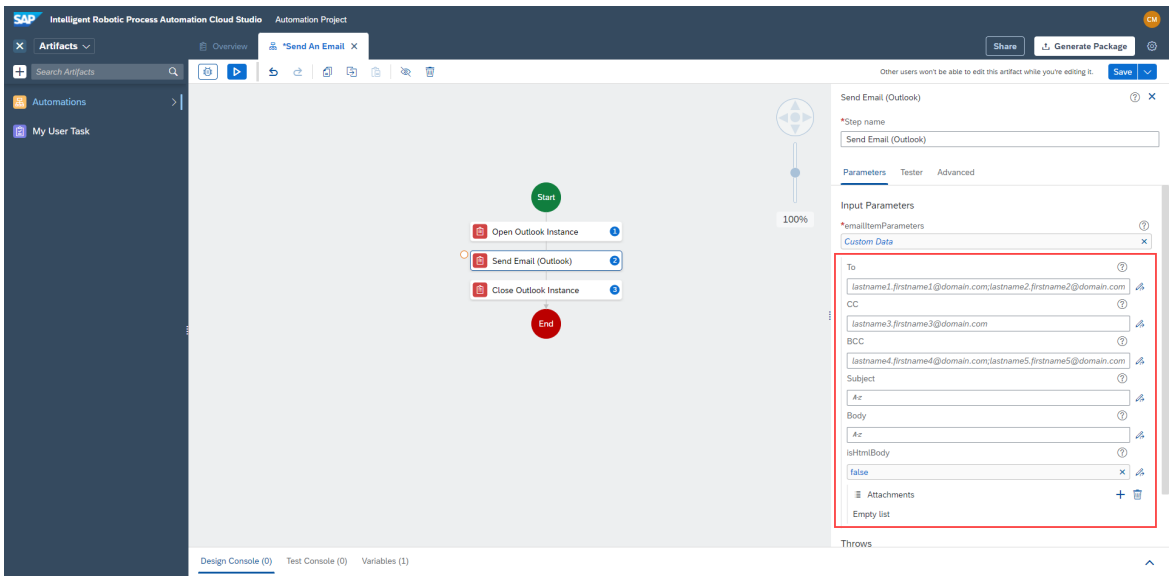
1. In your Automation, select the activity which requires complex input parameters. The activity input parameter field is displayed.



2. Click in the activity *Input Parameters* field and then select *Create Custom Data*.



The Activity Data Panel is displayed.



3. Provide the required data directly in Activity Data Panel.

> **i Note**
>
> You can define a complex input parameters using the expression editor. For more details about the expression editor refer to the Expression Editor [page 199] section. Referring to this section helps you to define complex input parameters easily.

If you want to provide the required data or edit the provided data using expression editor, follow the below procedure:

1. Click the ✎ at the right side of the required field.
   The *Edit Expression* dialog opens.
   You can manually type your expression.
2. Click *Test* to check your expression is valid.

3. Click *Save Expression*. The data type is now defined and the Activity Data Panel is displayed with the provided or updated data.

Parent topic: Manage Data within an Automation [page 160]

Previous: Input/Output Parameters [page 160]

Next: Data Types [page 168]

4.2.3.3 Data Types

A *data type* is an artifact describing a data structure that can be used as an input and/or output parameter in automations or processes.

By default, three types of data are available in the Cloud Studio: string, number, and boolean. For more complex data, you need to use *data types*.

Data types enable you to better formalize and describe the data used as input/output parameters for steps, activities, skills, processes, scenarios, triggers, or notifiers. Data types facilitate the manipulation and validation of data.

You can create a data type manually in the Cloud Studio, but some data types can also be created automatically when SDK packages or Desktop Studio scenarios are imported.

Parent topic: Manage Data within an Automation [page 160]

Previous: Define Complex Input Parameters [page 166]

Next: Create and Update Variables [page 179]

Related Information

Manage Data within an Automation [page 160]

4.2.3.3.1 Create a Data Type

Context

You can manually create a data type in the Cloud Studio.

Procedure

1. In the Home view or in the Project Explorer, click the *Create* button and select *Data Type*.
2. Enter a name and description, and click *Create*.

   > **i** Note
   >
   > A unique identifier is automatically created from the name you entered. If not, you must manually create an identifier.

3. A new tab opens in the main panel of the Cloud Studio. You can now add the fields that will be included in your data type: Click the *New Field* button.
4. In the *Field Properties* on the right, enter the name of the field you just added, and select its type.

   > **i** Note
   >
   > You can insert several types of field in a data type:
   > - Simple fields of type string, number or boolean.
   > - More complex fields of type object, or other existing data types (imported from SDK packages or from Desktop Studio scenarios).

5. **Optional:** Click the *New Child* button to add children to the field.

   > **i** Note
   >
   > Once you add children to a simple field it automatically becomes a field of type object.

6. **Optional:** If you want to manage a field as an array containing multiple values, check the *List* box in the *Field Properties*.
7. Save your changes.

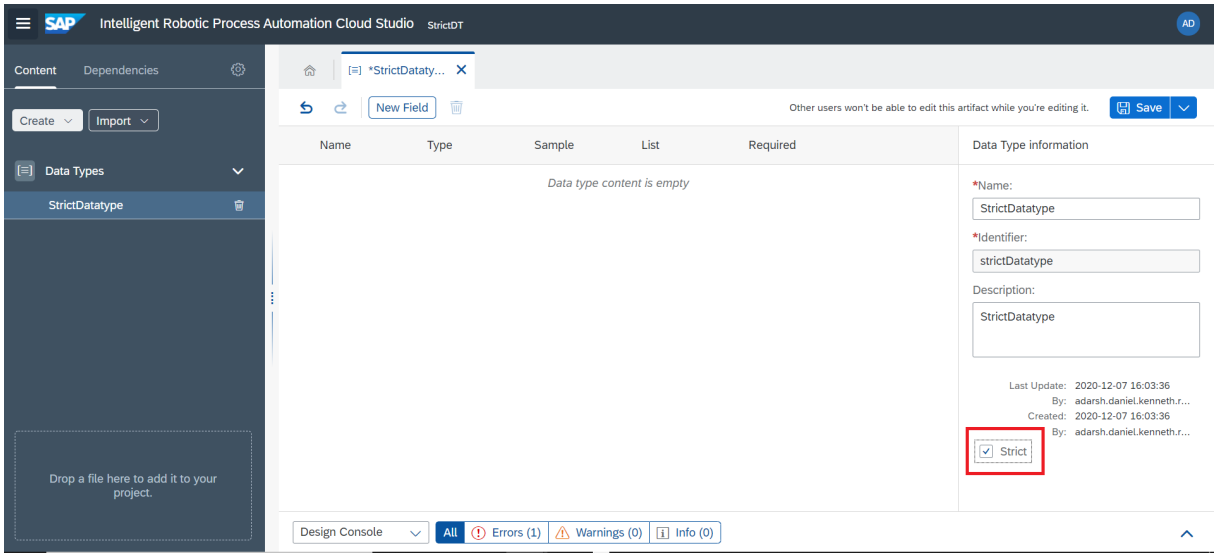
4.2.3.3.1.1 Create a Strict Data Type

At any point in time you can manually create a **Strict** data type in the Cloud Studio.

If an object is defined using a strict data type, you cannot add other properties to this object (using the Custom Scripting [page 251] for instance) other than the ones defined in the data type. If you provide any additional input properties, at runtime, the automation will fail.

In the Create a Data Type [page 168] topic, you have learned how to create a data type.

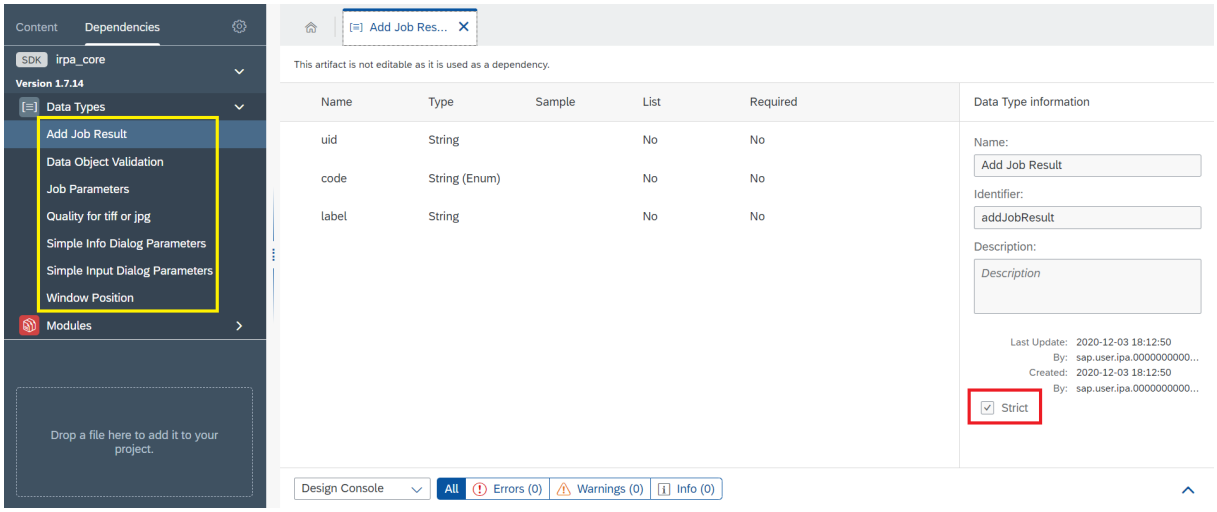
To create a strict data type, you must select the *Strict* option.



Save your changes. The **Strict** data type is created.

Data Types from SDK Packages

By default, some pre-defined data types exist in the SDK packages and the data types are listed when you import the SDK packages in the Cloud Studio. Every data type coming from SDK are strict and you cannot change the strict data type. You can create variables using the pre-defined data types from the SDK package.

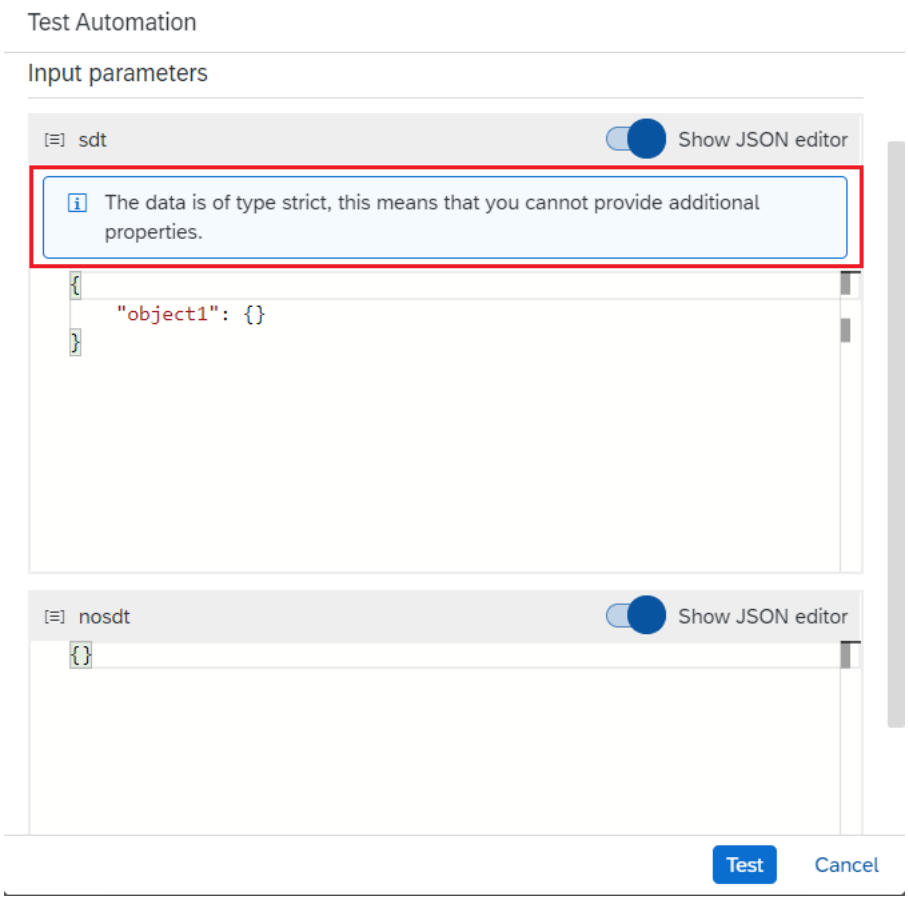


Strict Data Type Property

If a data type is strict, then you must not provide any additional input properties.

If you are working with a strict data type, then you must define the data exactly as the data type.

In the following screenshot shows the information message, not to provide any additional input properties for a strict data type.

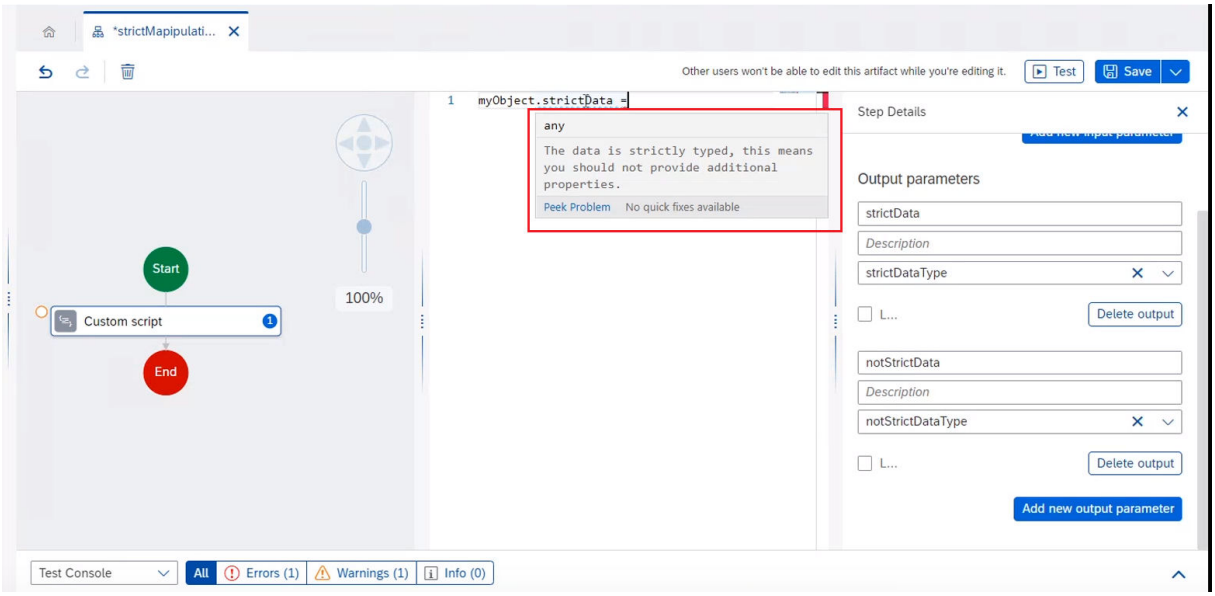


> **Note**
>
> If you still provide the additional input properties, then the automation will fail.

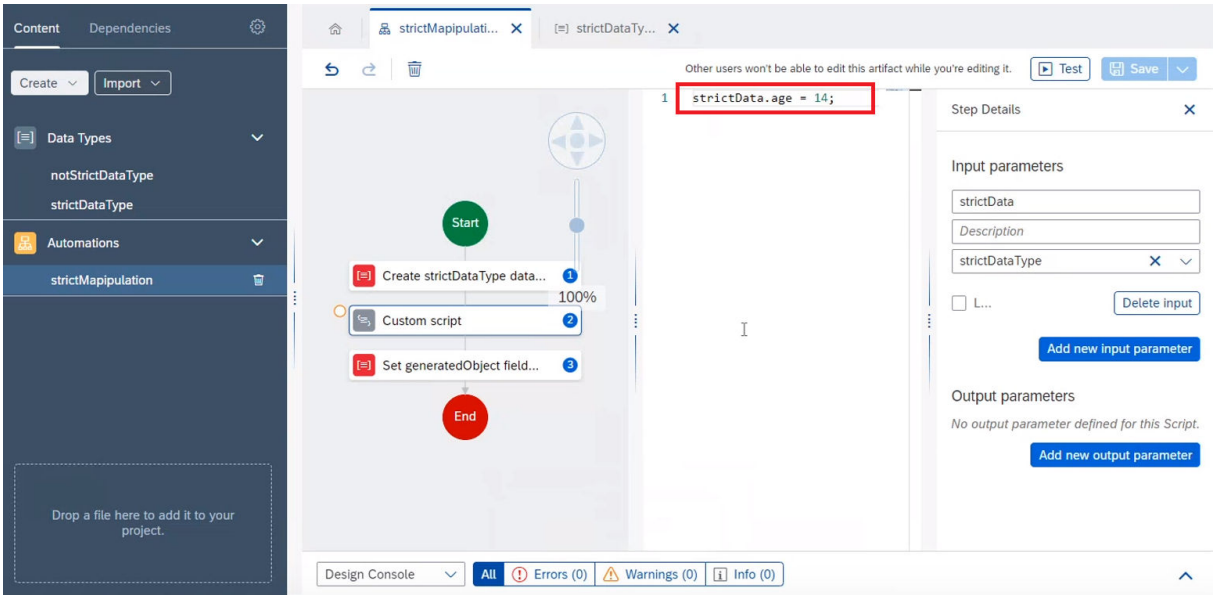
Also, you can observe that there is no information message is displayed for a data type which is not a strict data type. So, you can provide the appropriate additional input properties.

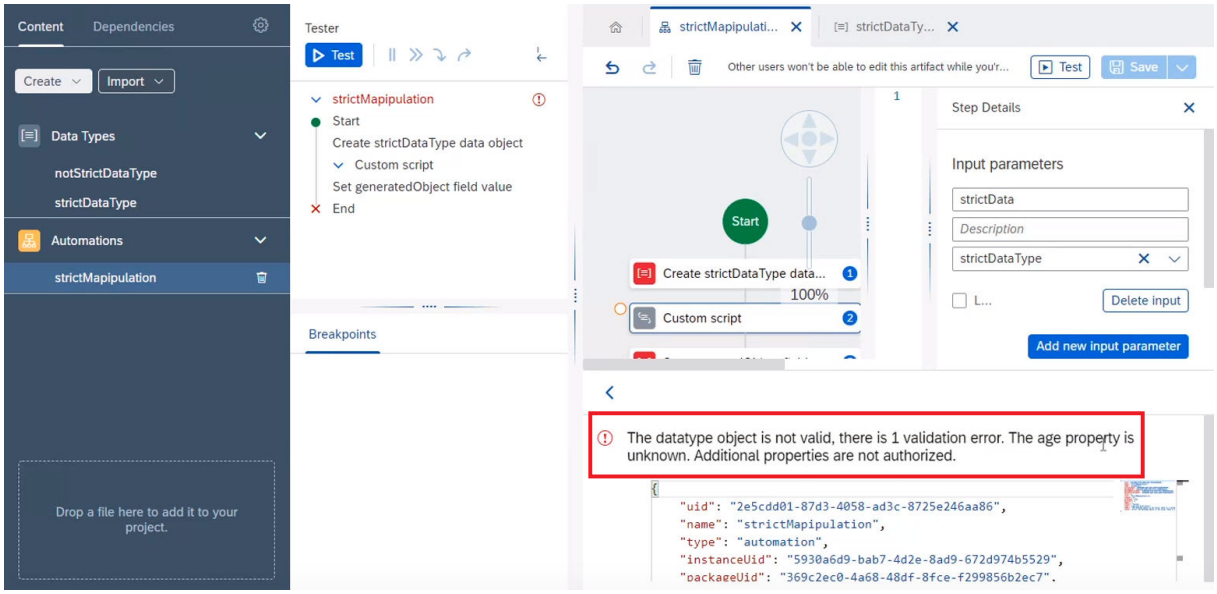
Custom Script

If a data type is strict and you try to provide the additional input properties through Custom Scripting [page 251] then the system shows the information message, not to provide any additional input properties.



The following example shows the automation failed due to an additional input property provided.





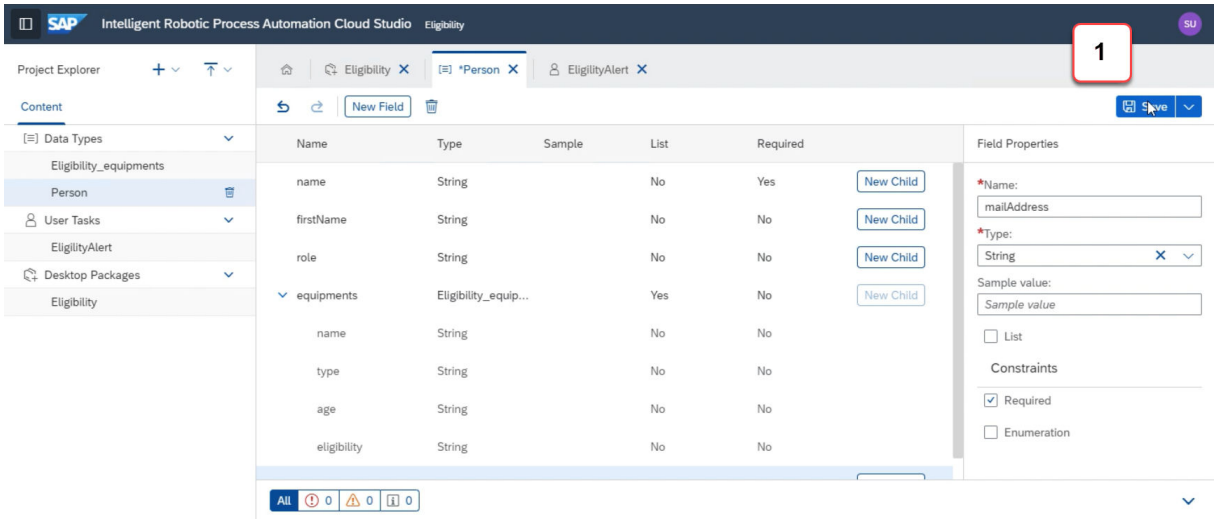
4.2.3.3.1.2 Constraints in Data Types

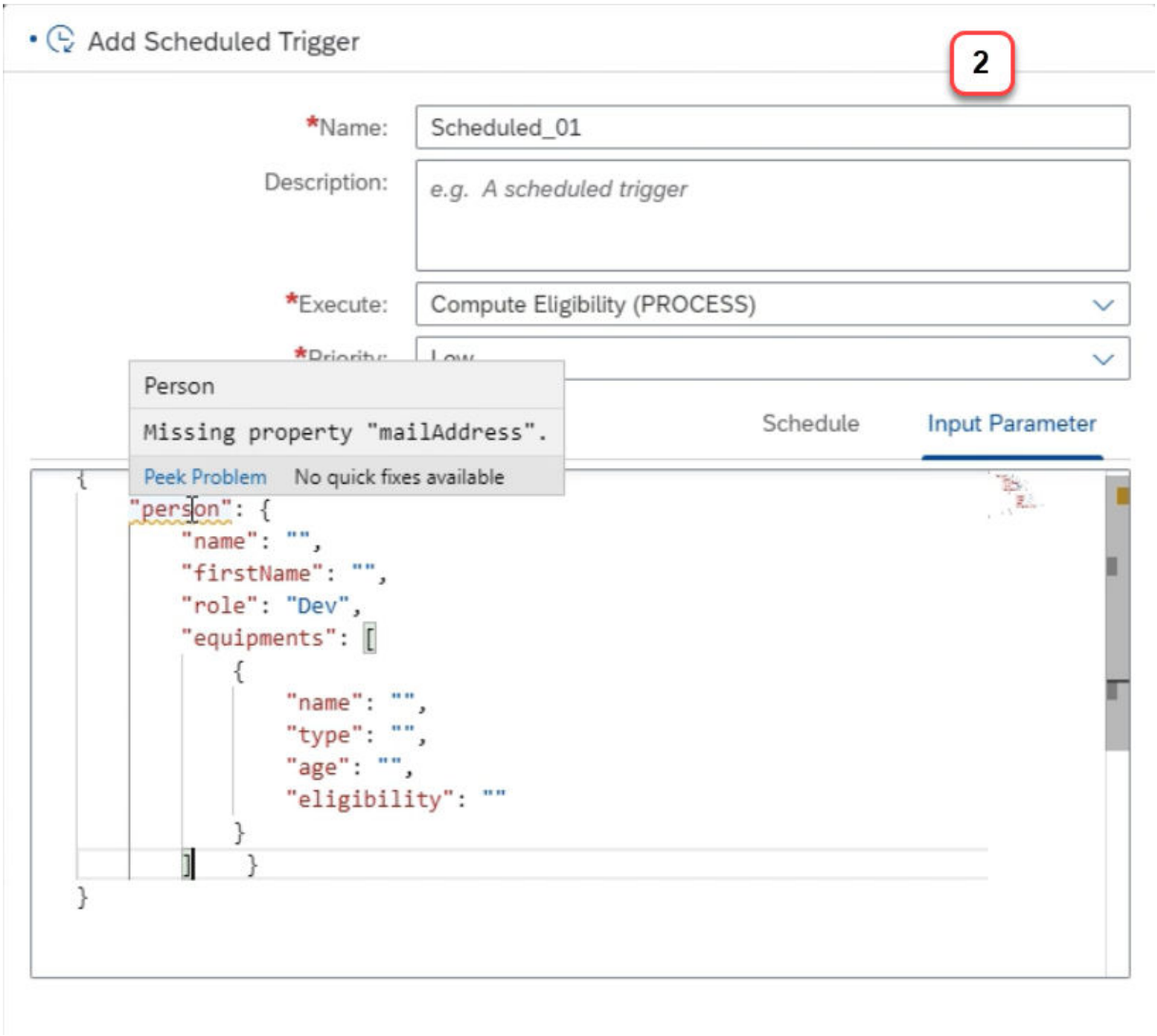
You can set two types of *Constraints* when you create a data type in the Cloud Studio. The *Required* constraint and the *Enumeration* constraint.

Required

The *Required* constraint allows you to set a data field that must be provided later on in the *Input Parameter* of a trigger.

For example, if you add the *New Field mailAddress* with the *Required* constraint (1), you must provide this data to be able to create the trigger (2).

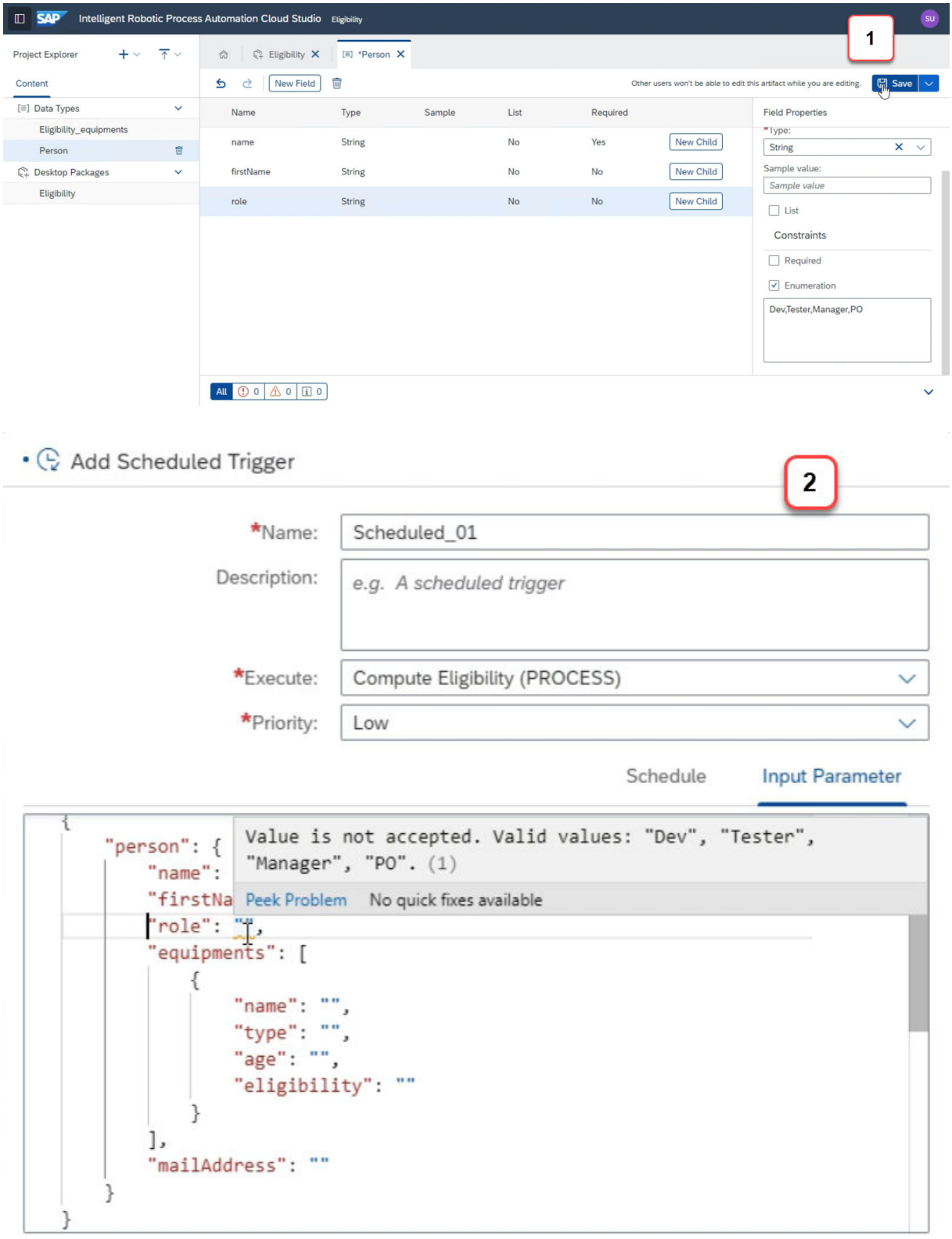




Enumeration

The *Enumeration* constraint allows you to define a set of several values for a data field. Meaning that you can limit the choice of values to provide for a data field, in the *Input Parameter* of a trigger.

For example, if you add the *New Field role* with the *Enumeration* constraint and set several values (1), you must provide one of the values you previously set, to be able to create the trigger (2).


Intelligent Robotic Process Automation Cloud Studio
Eligibility
1
Project Explorer
Content
Data Types
Eligibility_equipments
Person
Desktop Packages
Eligibility
Eligibility
*Person
New Field
Other users won't be able to edit this artifact while you are editing.
Save
Name
Type
Sample
List
Required
name
String
No
Yes
New Child
firstName
String
No
No
New Child
role
String
No
No
New Child
Field Properties
Type:
String
Sample value:
Sample value
List
Constraints
Required
Enumeration
Dev,Tester,Manager,PO
All
0
0
0
Add Scheduled Trigger
2
*Name:
Scheduled_01
Description:
e.g. A scheduled trigger
*Execute:
Compute Eligibility (PROCESS)
*Priority:
Low
Schedule
Input Parameter
{
"person": {
"name":
"firstNa
"role": "",
"equipments": [
{
"name": "",
"type": "",
"age": "",
"eligibility": ""
}
],
"mailAddress": ""
}
}
Value is not accepted. Valid values: "Dev", "Tester", "Manager", "PO". (1)
Peek Problem
No quick fixes available

> **Note**
>
> The *Enumeration* constraint is case sensitive: the values must be spelled identically in both the *Field Properties* of the Cloud Studio and the *Input Parameter* of the Cloud Factory.

4.2.3.3.2 Data Types from SDK Packages

Some data types can be automatically created when SDK packages are imported.

When you first create an automation or an application, you select SDK packages that will be imported as dependencies to your project. You can also manually import SDK packages from the store.

The SDK packages contain activities, and some of the activities are displayed as data types. If you go to the *Dependencies* tab in the *Project Explorer* panel, you can see all the dependencies of your project, including the data types that are part of imported SDK packages.

> **Note**
>
> To maintain compatibility between versions, data types coming from SDK packages are **strict** data types. If an object is defined using an SDK data type, you cannot add other properties to this object (using the Custom Scripting [page 251] for instance) other than the ones defined in the data type.

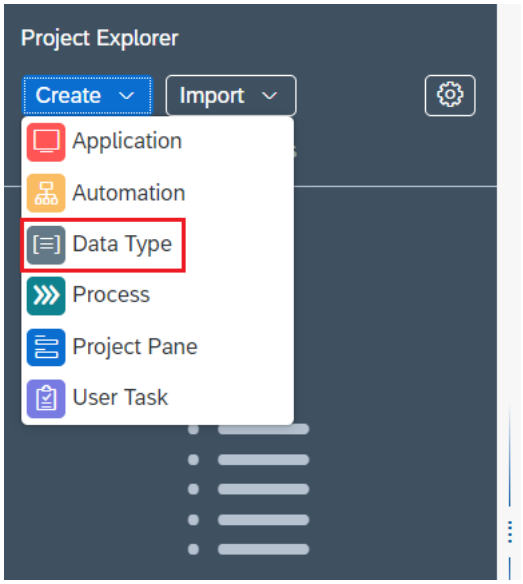
4.2.3.3.3 Reuse Data Type from another Package

The data type can use data types coming from an SDK package. This helps in reusing existing data type inside the data type and this can be used to automate many processes.

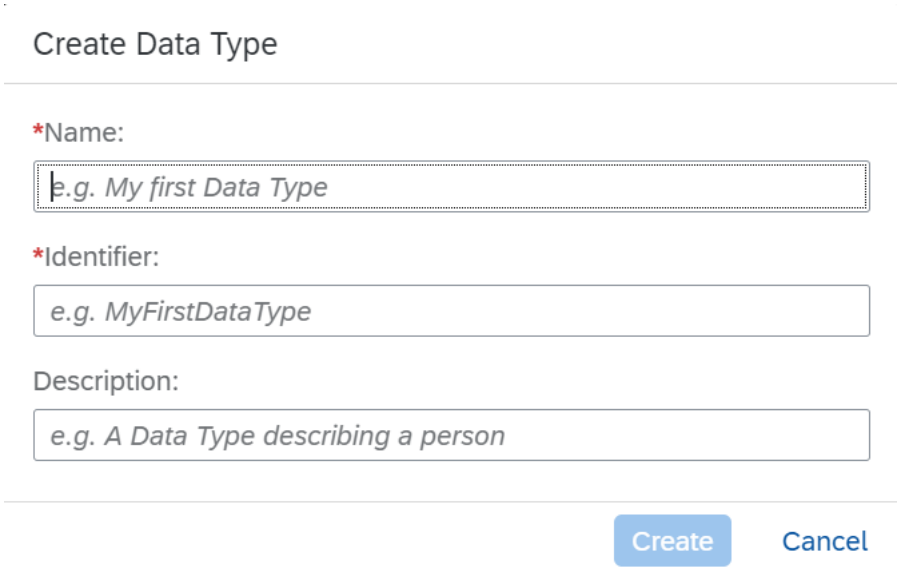
To achieve this, you must create a data type and then reuse existing data type inside the data type. For more information about data type and creating a data type refer to the Data Types [page 168] section.

To reuse the data type inside the data type, follow the below procedure:

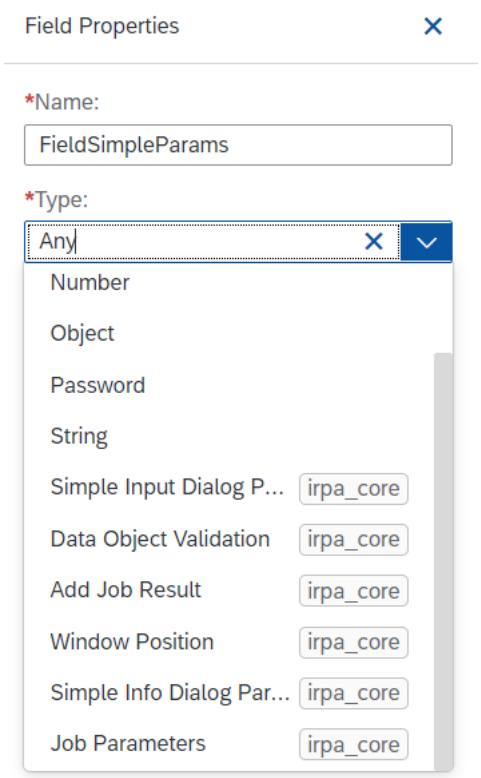
1. In the SAP IRPA Cloud Studio, Project Explorer, under *Content* click *Create* and then click *Data Type*.



The *Create Data Type* pop-up screen is displayed.

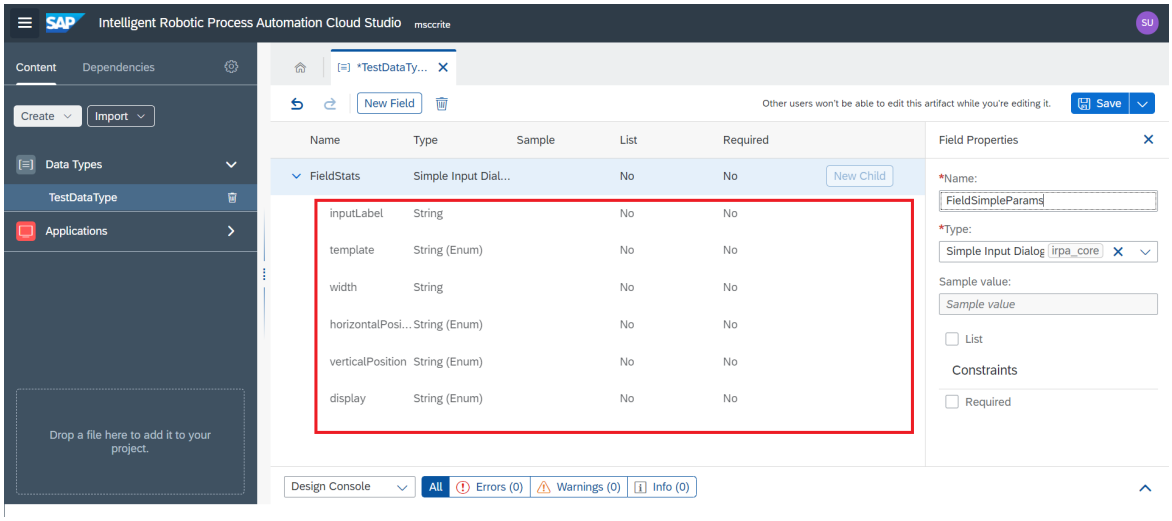


2. Enter a name and description, and click *Create*.
3. A new tab opens in the main panel of the Cloud Studio. You can now add the fields that will be included in your data type: Click the *New Field* button.
4. In the *Field Properties* on the right hand side, in the *Name* field enter the name of the field.
5. Click **Type*. The drop-down list is displayed.



> **i Note**
>
> You can define your field type by:
>
> - Choosing predefined simple types (String, Number, Boolean etc).
> - Choosing a Data Type defined inside the same project.
> - Choosing a Data Type coming from SDK packages (core or excel etc).

6. Select the required data type coming from an SDK package.
   The Data Type details are displayed.



7. Click *Save* to save your changes.

> **i Note**
>
> Data types coming from SDK packages are **strict** data types: to maintain compatibility between versions, you cannot manually modify the data type properties using the Custom Scripting [page 251].

Related Information

Data Types [page 168]

4.2.3.4 Create and Update Variables

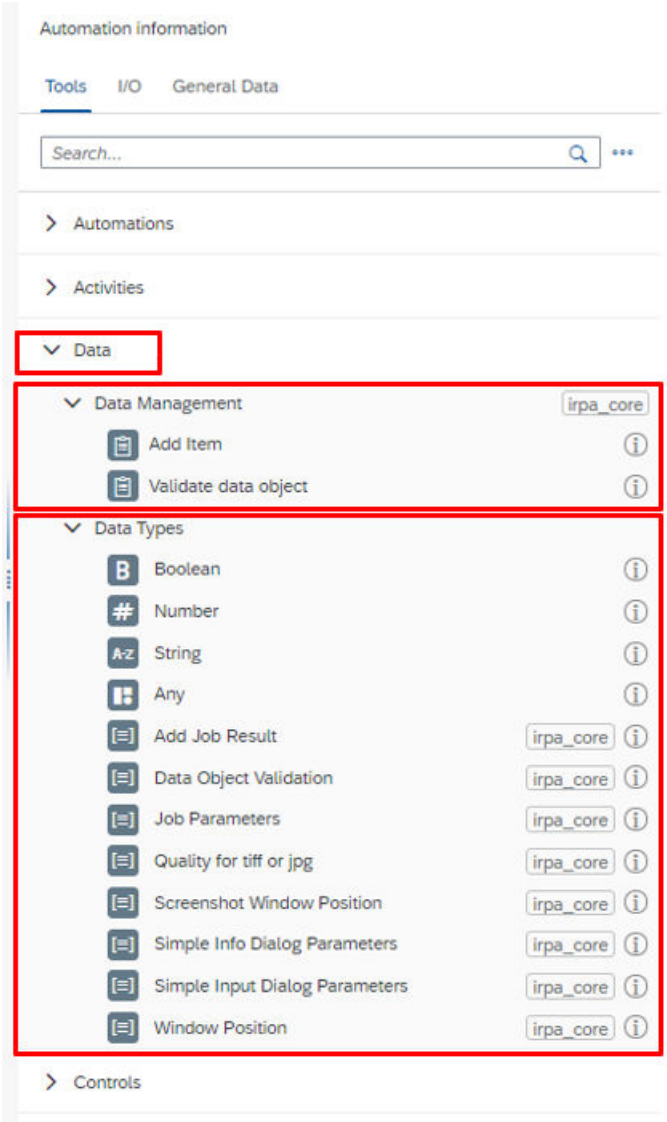
The variables you use in the Cloud Studio to build your automation are data storage that has a name, a type (example: string, list of string or data type), and a value. A variable in the automation is also associated to a step represented by its number.

Variables allow you to:

- store content that can be reused inside the automation flow.
- set or modify the content of a variable that contains value in the automation flow.

To view *Data Management* and *Data Types* activities, in the *Automation information* panel, go to the *Tools* section and click *Data*. The dropdown list displays:

- The dropdown *Data Management* list where you can find all activities related to data management.
- The dropdown *Data Types* list where you can find all data types.



Visual Representation of Data Type Variables

There are two types of data type variables:

- Primitive data types
- Complex data types

These primitive and complex data type variables allow you to model your data to define how the automation intends to use the data.

The different types of variables are described as follows:

Primitive data type: It can be a simple data such as a string, a number, and a boolean.

A variable of type "any" can provide any type of data such as a string, a number, a boolean, a complex subject and so on. A variable of type "any" allows you to get rid of the data validation.

Complex data type: You need to use complex data types for more complex data. A data type allows you to compose primitive types into a compound type representing your data structure. It contains several property fields which can be either primitive types defined that could be arranged in a hierarchy form, or other project's data types. A data type enables you to organize and manage your data which can impact the access and modification efficiency. For more information on data types, see Data Types [page 168].

You can define a primitive or a complex data type as a list that represents a collection of those elements.



Parent topic: Manage Data within an Automation [page 160]

Previous: Data Types [page 168]

Next: Different Types of Data Fields [page 197]

4.2.3.4.1 Using Primitive Data Type Variables

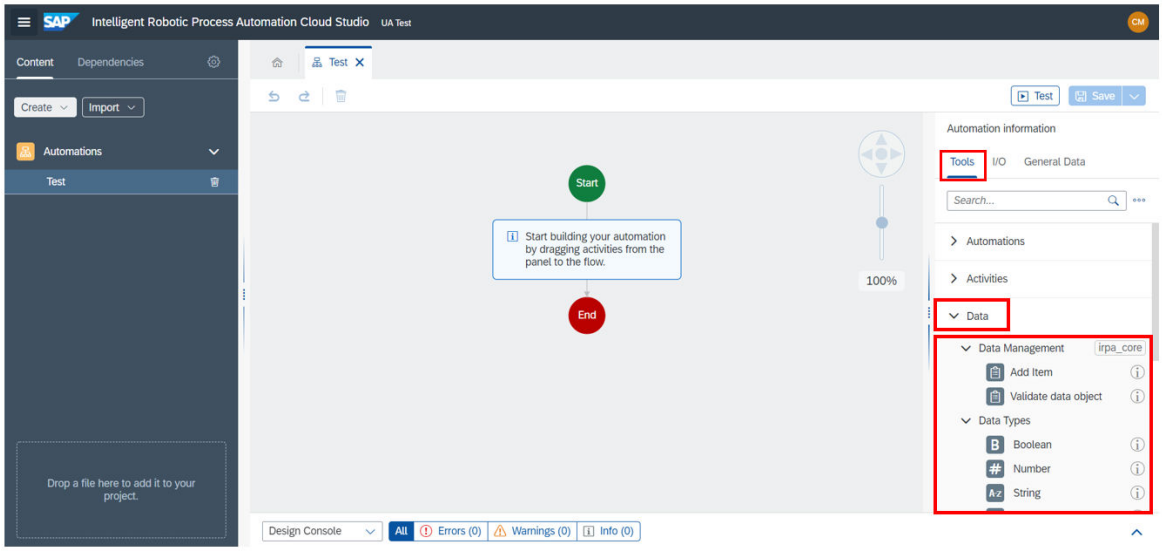
Prerequisites

You have created one or several automations in the Cloud Studio as described in Create an Automation [page 156].

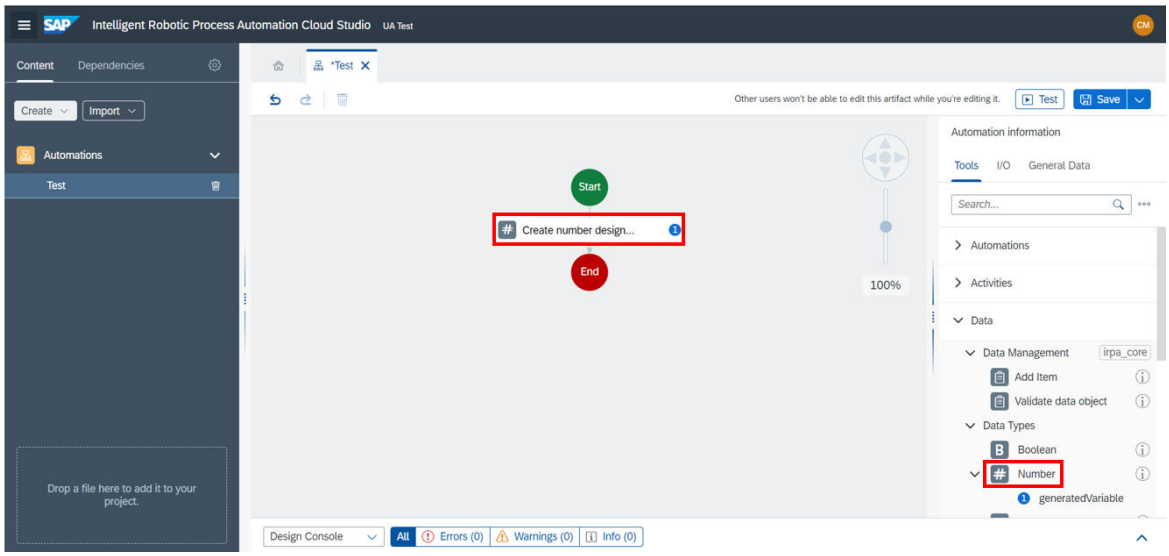
Example

The following use case demonstrates how to create a variable of type "number". You can follow a similar procedure for "boolean", "string" and "any" types of variables.

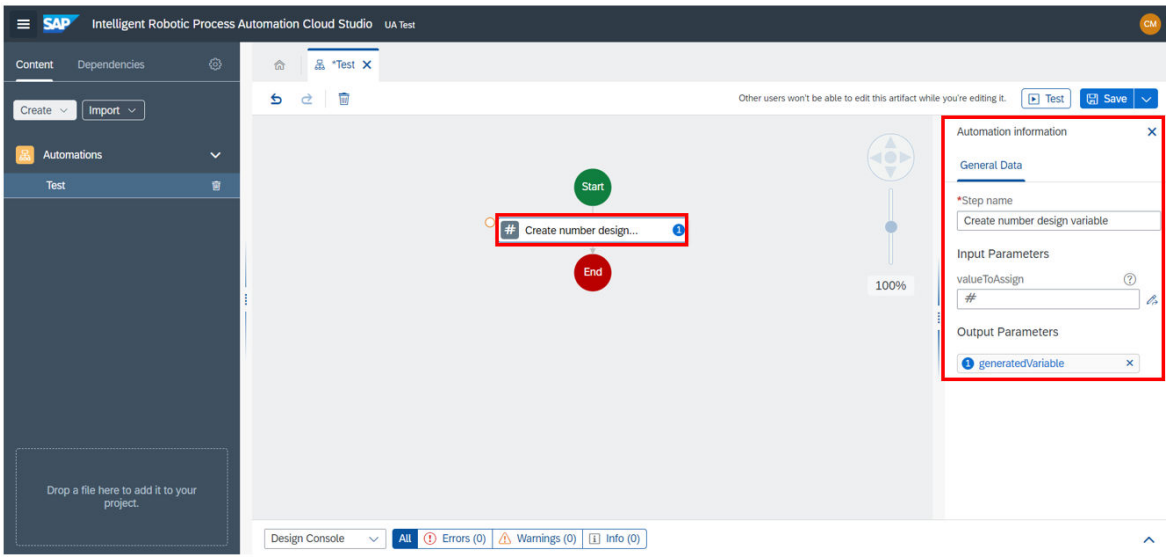
1. In the automation, on the right-hand side panel, go to the *Tools* section and click *Data*. The dropdown list of different kinds of data is displayed.



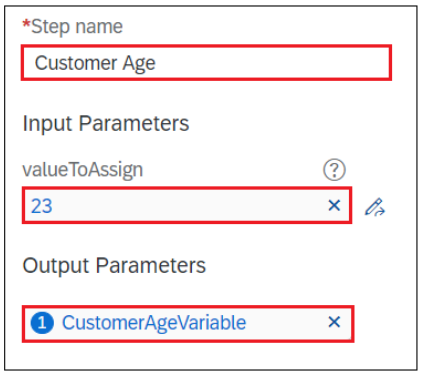
2. Drag and drop *Number* data into the workflow of the automation.



3. Select the step *Create number design variable*. The *Automation information* panel appears.

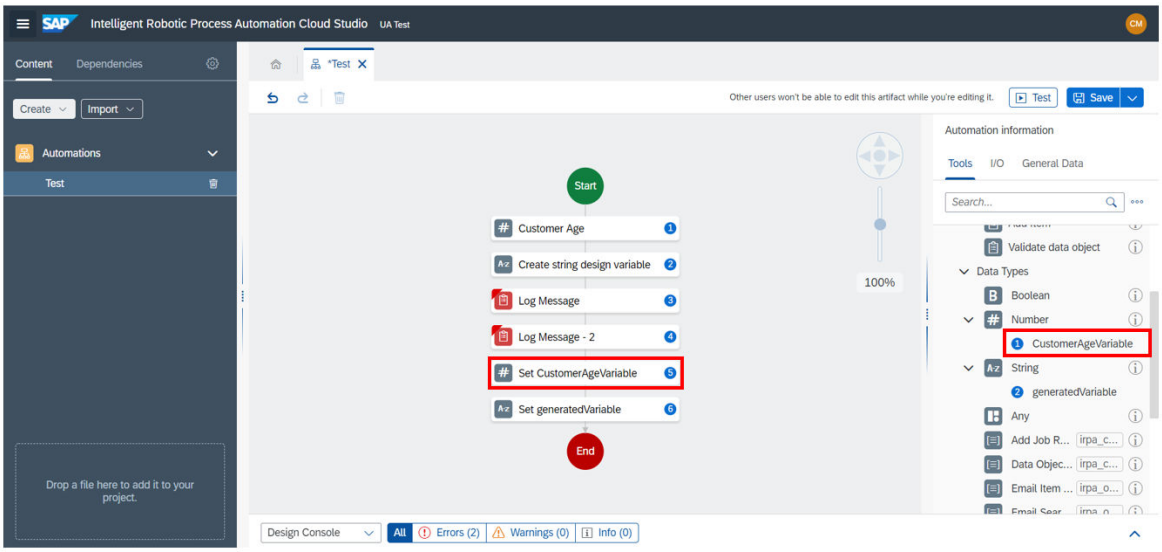


4. Modify the *Step name* as per your requirement. Set an input parameter in the *valueToAssign* field. The value must be a number for a variable of type number. You can also modify the value of output parameter as per your requirememt.

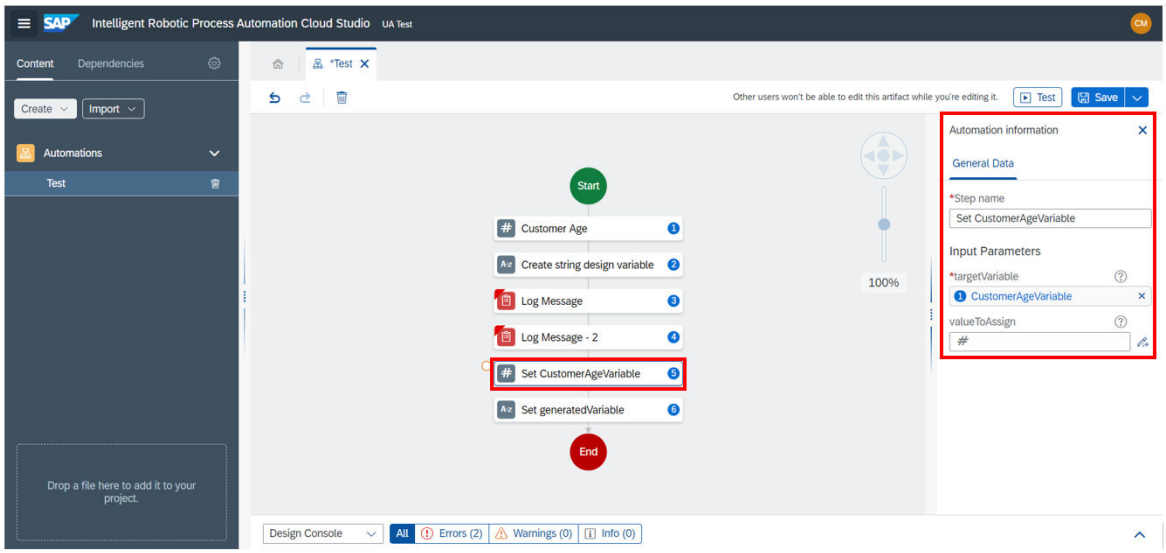


> **Note**
>
> The value must be the same type as the generated variable for any variable such as number for number, string for string and so on.

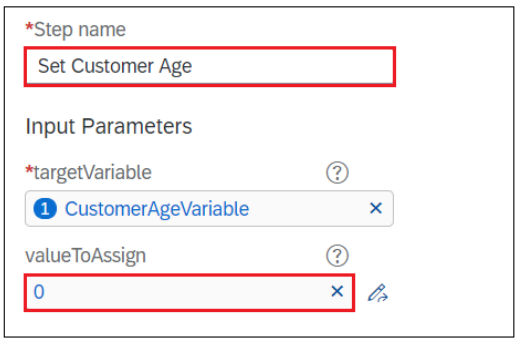
5. Drag and drop *CustomerAgeVariable* under *Number* data into the workflow to update its value.



6. Select the step *Set CustomerAgeVariable*. The *Automation information* panel appears.



7. Modify the *Step name* as per your requirement. Set an input parameter in the *valueToAssign* field. You can modify the value of *targetVariable* field.



> **Note**
>
> The value of *valueToAssign* field can be a formula or another previously computed variable.

8. You can use the *Log* activity if you want to display the variable value at runtime.

4.2.3.4.2 Using Complex Data Type Variables

Complex Type Data Variables

Complex type data variables allow you to:

- initialize data depending on the type of object. It allows you to parameterize data and store data in different data field.
- modify the content of an object.
- validate the type of data variable. It allows you to know whether the data is valid or not.

User Interface (UI) of Complex Data Variables

Create Variable: You must drag and drop the variable to create a complex variable.

Set Variable: You must drag and drop the variable for the set variable.

> **Note**
>
> Create and set variables both have almost similar UI. The representation of the fields of the variables is similar. The only difference is that you can map only one data field in set variable.

Validation Variable Activity: Validation variable activity is accessible from right hand side panel of the *Activities*. It allows you to validate the data explicitly. This activity is no longer required as data validation is now done automatically at each step of automation. For more details on data validation, refer to the following note.

> **Note**
>
> When a data type is used as input of automation or parameter in an activity or a custom script, it gets validated automatically. An exception will be thrown if:
>
> - any field does not have the correct data type. For example, string field is expected however the value is a number.
> - some mandatory fields are missing.

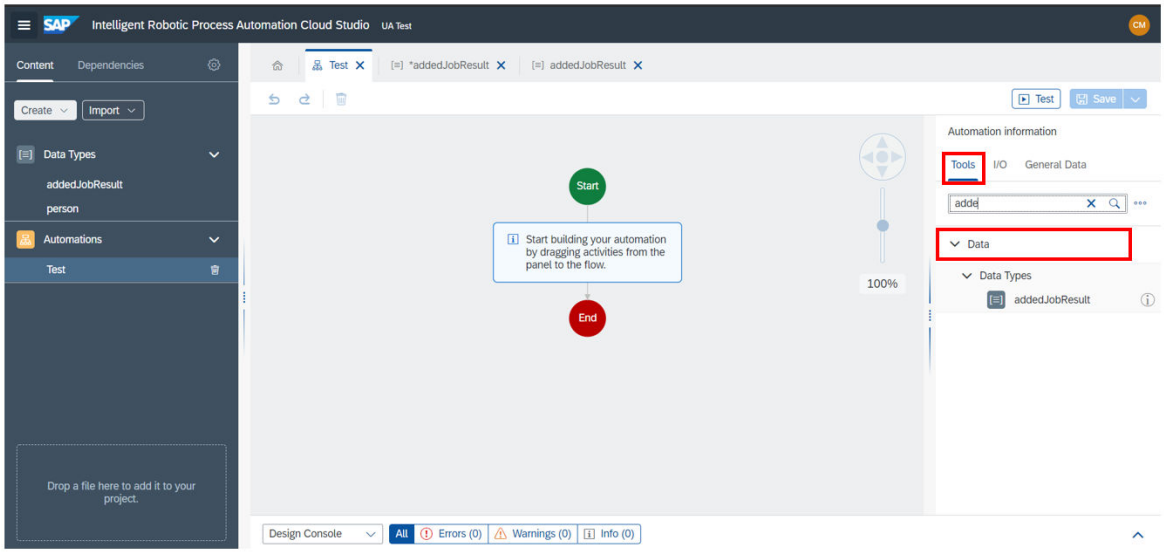
Using Primitive Data Type Variables

As a prerequisite, you must have created one or several automations in the Cloud Studio as described in Create an Automation [page 156].

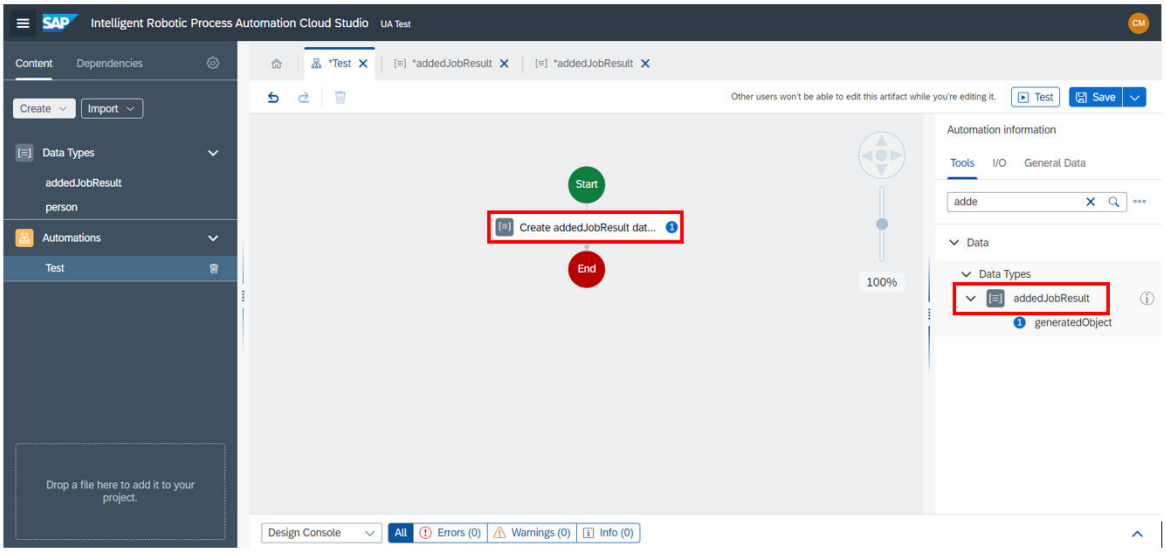
Example

The following use case demonstrates how to create, set, and validate complex type data variables.

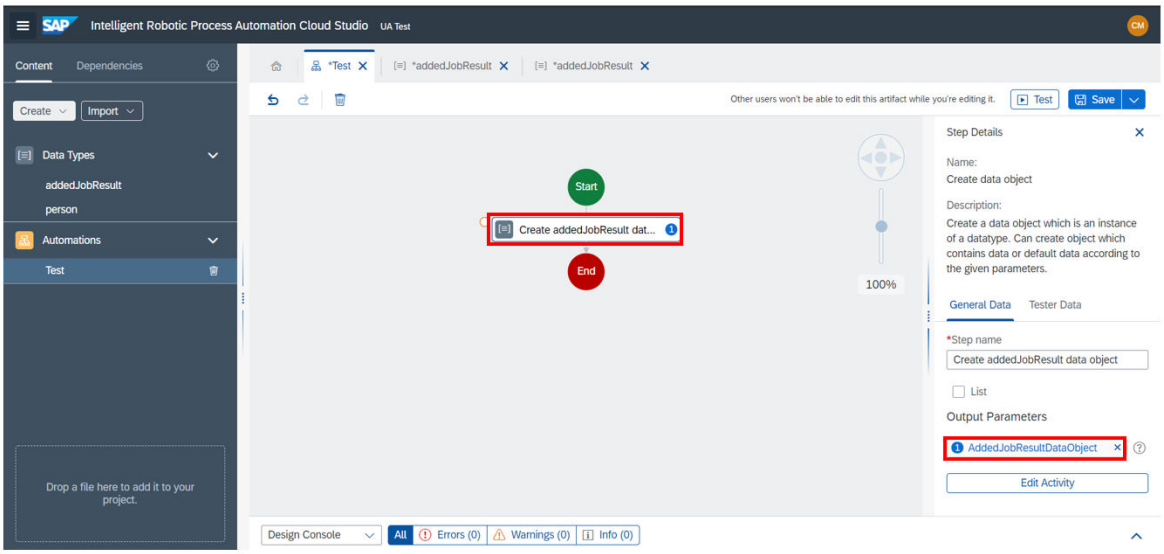
1. In the automation, on the right-hand side panel, go to the *Tools* section and click *Data*. You must know the type of the object that need to create in the automation.



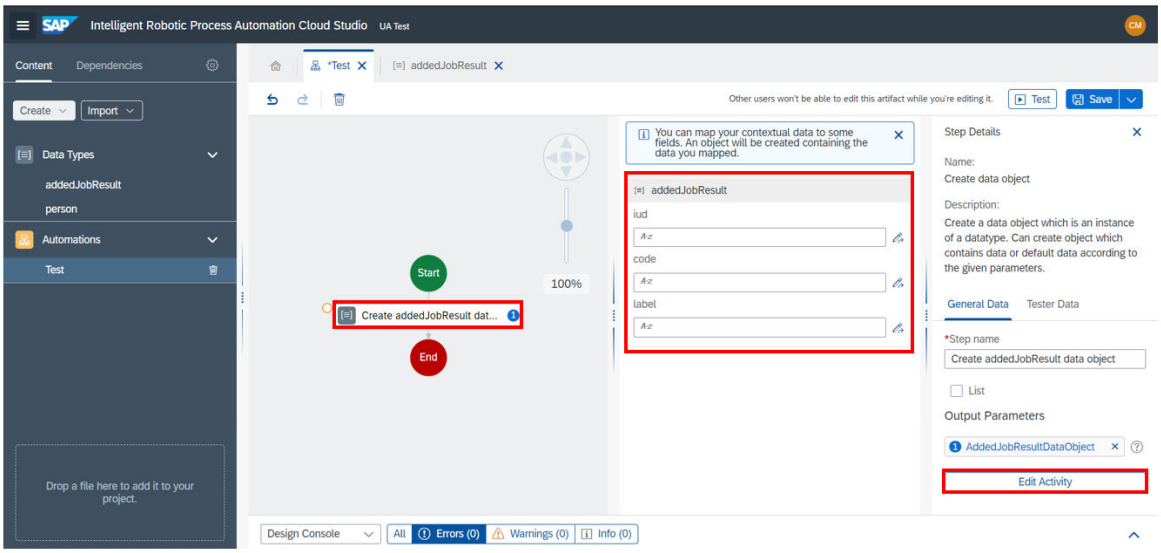
2. Drag and drop *addedJobResult* data into the workflow of the automation.



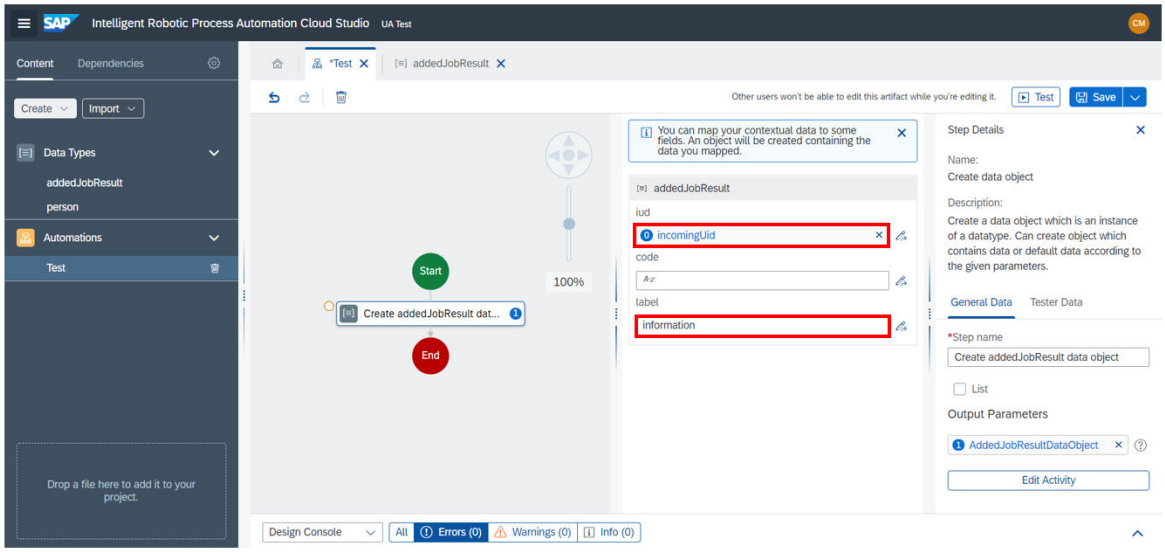
3. Select the step *Create addedJobResult data object*. The *Step details* panel appears. You can modify the value of the *Output Parameters* field as per your requirememt. The *Output Parameters* field has the correct structure with empty fields.



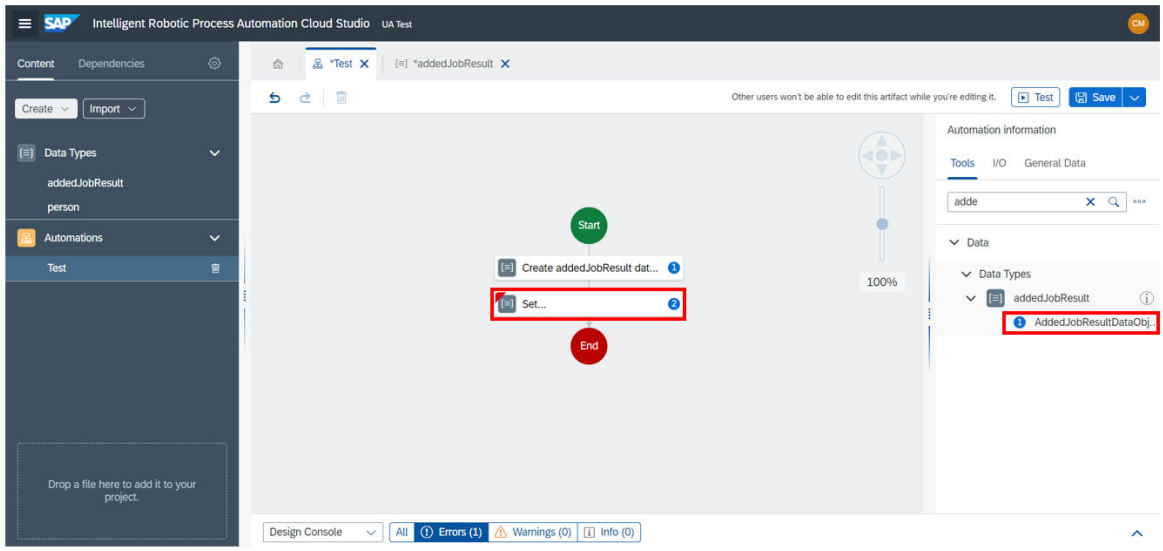
4. Double click on the step *Create addedJobResult data object* or click *Edit Activity* in the right pane. Different data fields appear.



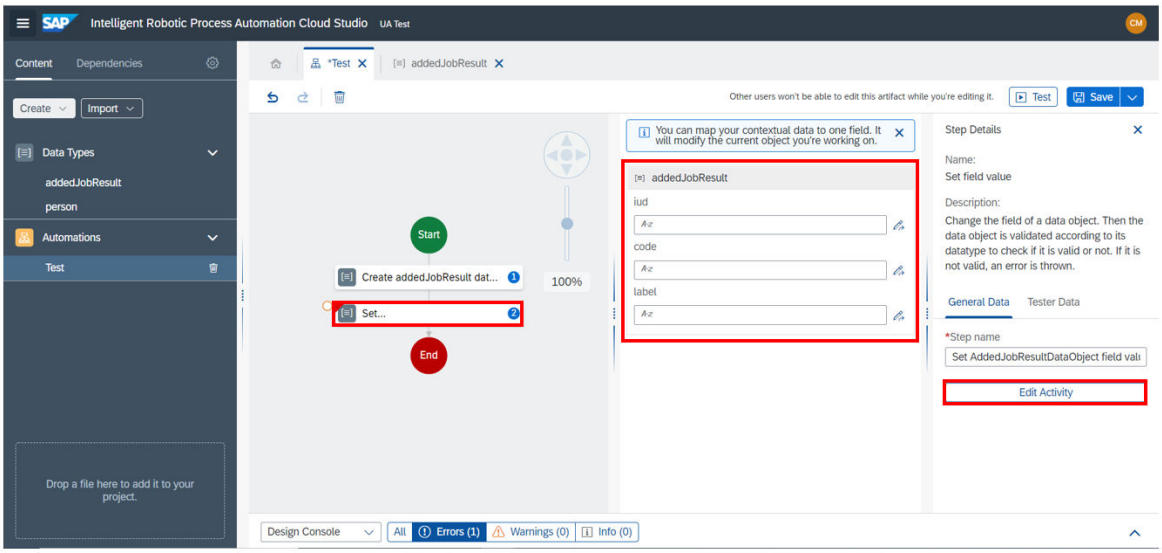
5. Provide the value into the complex fields. You can also edit formula with expression editor.



6. Drag and drop *AddedJobResultDataObject* under *addedJobResult* data into the workflow of the automation to update the value of complex type data variable.

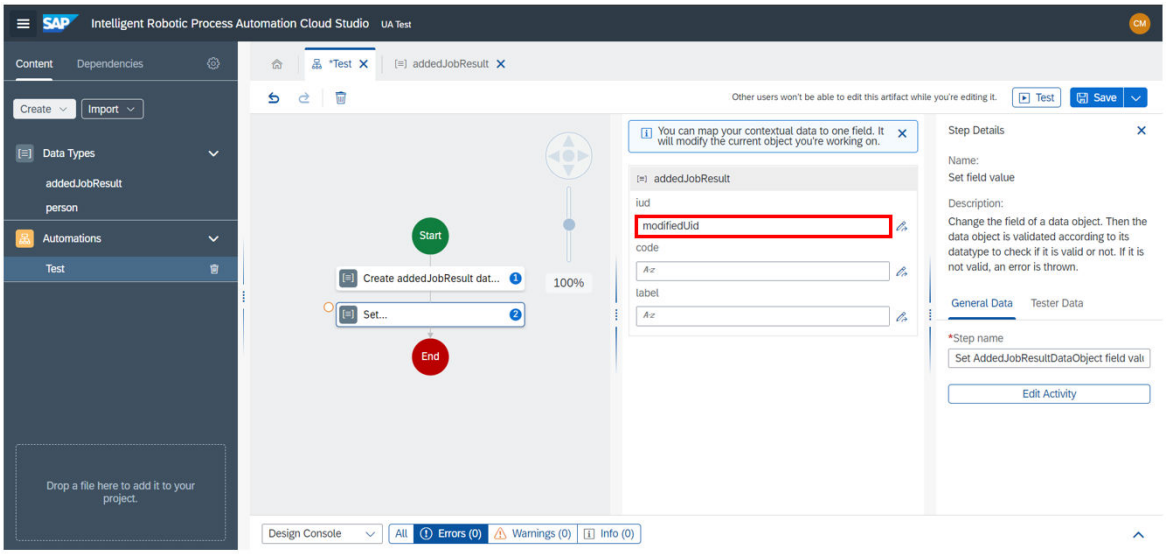


7. Double click on the step *Set AddedJobResultDataObject field value* or click *Edit Activity* in the right pane. A similar view of different data fields like the create variable activity appears. However, the only difference is that you can provide only one data field value in the set variable activity.



8. Modify the field value of a data variable.

> **i Note**
>
> You need to use:
>
> - text to enter static data.
> - suggest capability to bind previously defined variables.
> - expression editor to enter formula with Expression Editor [page 199].



9. You can use the *Log* activity if you want to display the data variable value at runtime.

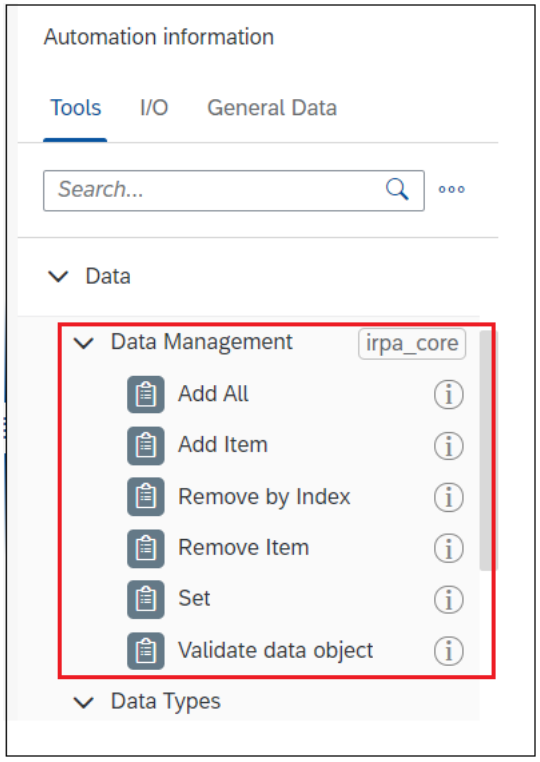
> **i Note**
>
> You can not use the expression editor to set or modify complex data variable since you can do it through the complex data variable.

4.2.3.4.3 List Management

In the Cloud Studio, a collection of activities provide enhanced support to manage the lists of your automation.

> **i** Note
>
> To learn how to use lists, you can download the How to use list of objects sample on the SAP Intelligent RPA Store.

You can now have a powerful way of handling list items using data management activities.



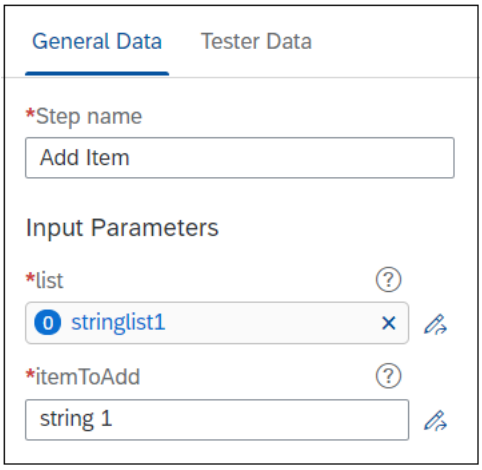
Using Data Management Activities to Manage Lists

Add Item

Use this activity to add an item to the existing list.

In the *Input Parameters*, you must provide the name of a list in the *list* field and to be added value of an item in the *itemToAdd* field.

For example,

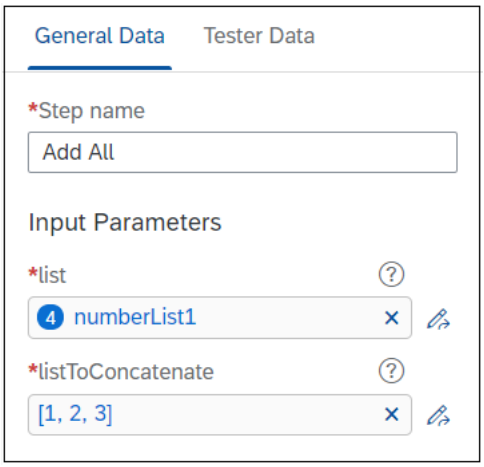


Add All

Use this activity to concatenate a list of items to another list.

In the *Input Parameters*, you must provide the name of an existing list in the *list* field and the items to be added or the name of another list in the *listToConcatenate* field.

For example,

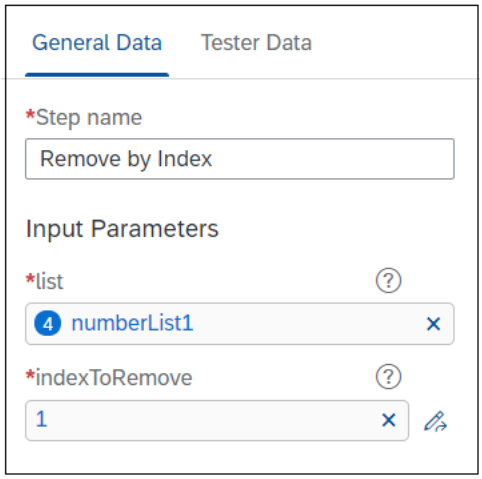


Remove by Index

Use this activity to remove an item from a list based on its index in the list.

In the *Input Parameters*, you must provide the name of an existing list in the *list* field and the index value to be removed from the list in the *indexToRemove* field.

For example,



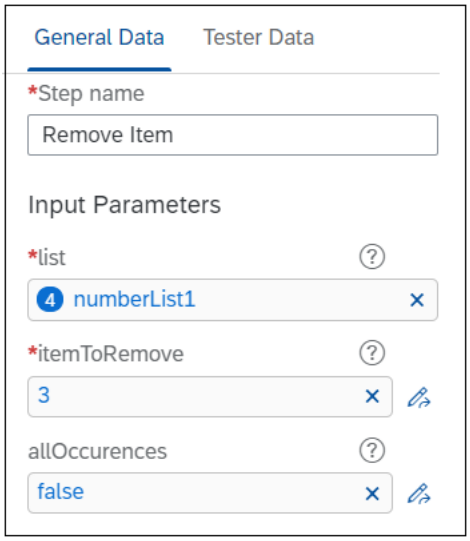
> **Note**
>
> If the index is higher than the length of the list, no item is removed.

Remove Item

Use this activity remove an item from a list based.

In the *Input Parameters*, you must provide the name of an existing list in the *list* field and the item or name of the item to be removed from the list in the *itemToRemove* field.

For example,



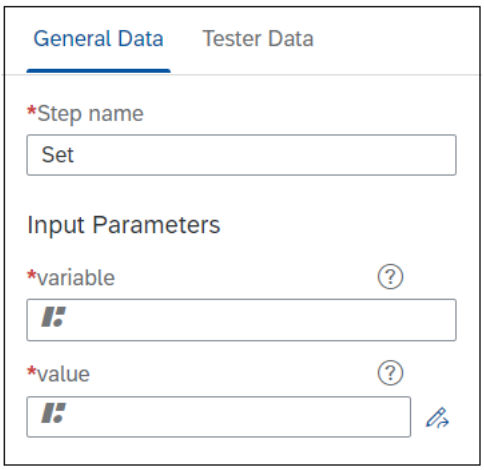
> **Note**
>
> *allOccurences*
>
> If false, only the first item will be removed. In case of **dataObject**, the remove is performed by reference but not by the value, that means only the repetition of the same instance corresponding to **itemToRemove** will be removed but not the other instances with the same property values.

If true, removes all occurrences of the item if it is found several times in the list.

Set

Use this activity to update the new value or reference of an object.

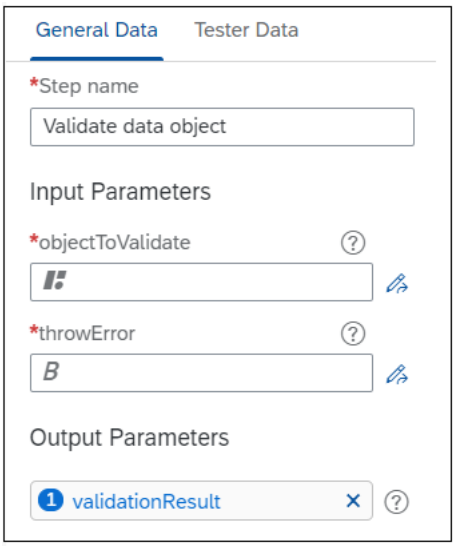
In the *Input Parameters*, you must provide the variable in the *variable* field and the new value or reference an object in the *value* field.



Validate data object

Use this activity to validate an object which is datatype instance..

In the *Input Parameters*, you must provide the data type object to be validated in the *objectToValidate* field.



> **i Note**
>
> *throwError*
>
> Throws an error if the object is not validated. If this error is not caught later, the automation will stop.

4.2.3.4.4 Date and Time Data Type Variables

You can now handle dates and times using data types.

Date and Time Data Types Format

You can now handle date and time using three data types:

- *Date*
- *Time*
- *DateTime*

The representations of these data types are based on the **ISO 8601** standard. According to this standard,

- A date is written in the **2018-11-13** format.
- A time is written in the **20:20:39+00:00** format.
- A date and a time are written in the **2108-11-13T20:39+00:00** format.

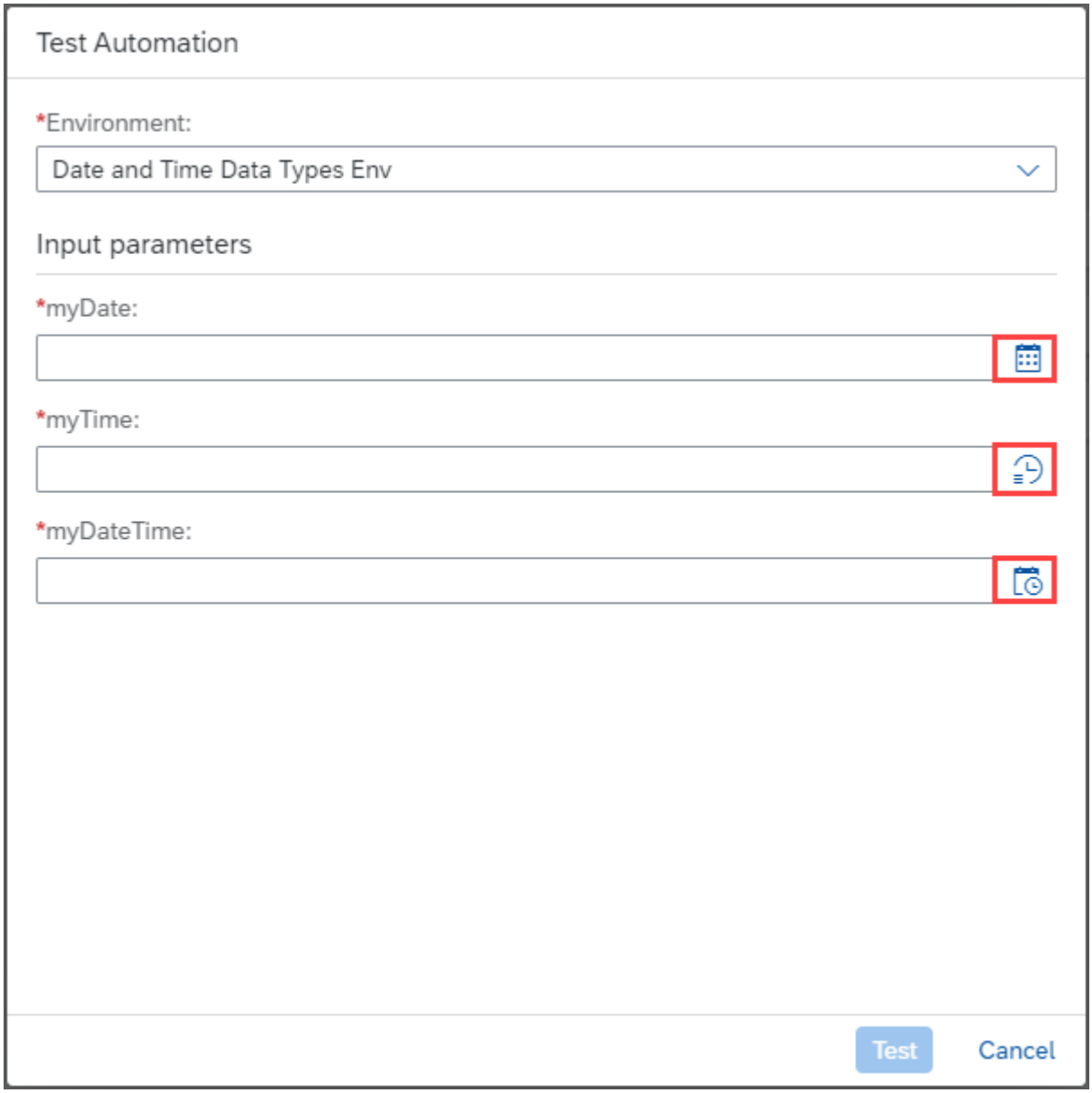
If you want to use a date or a time, you must follow the ISO 8601 format.

Set Input Parameters in Picker Panels

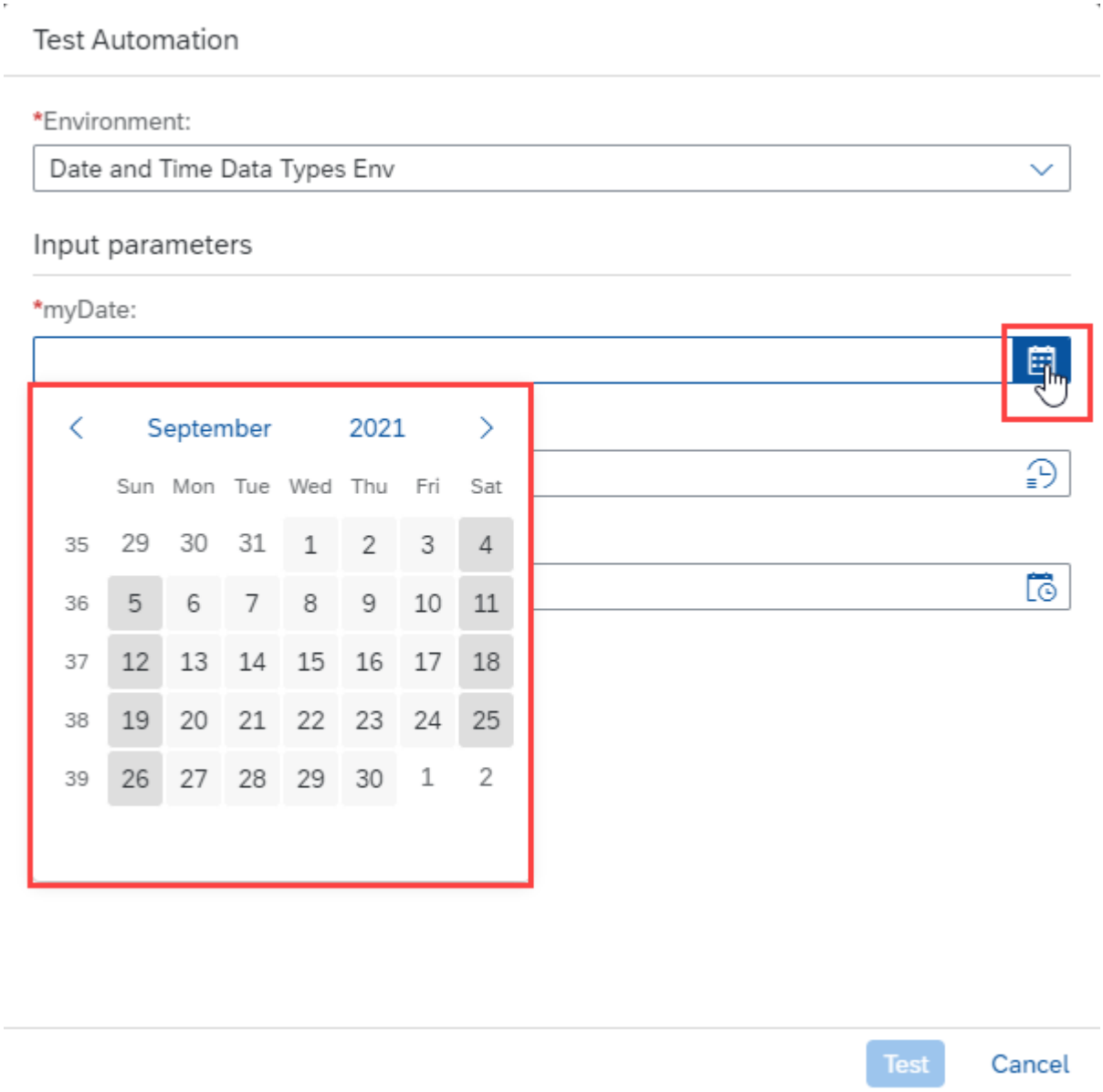
When you define an input parameter which is expecting a date or a time, a picker panel appears and you can select dates and/or times. This picker panel appears in three cases:

- When you want to test your automation.
- When you want to provide environment variables.
- When you want to input parameters from triggers.

> **Note**
>
> You cannot open a picker panel for dates and times in the Automation Editor. However, you can still enter dates using the keyboard or by binding a variable.



Click the icons to open the corresponding picker panels:



Manipulate a Date in JavaScript

If you want to manipulate a date with all the APIs available in JavaScript, you must convert this date into a JavaScript one:

- Using a *Custom Script*.
  You can do the conversion with: `new Date(myDate)`
- Using the *Expression Editor* and entering `new Date(myDate)` .
  For more information, see Dates [page 207].

4.2.3.5 Different Types of Data Fields

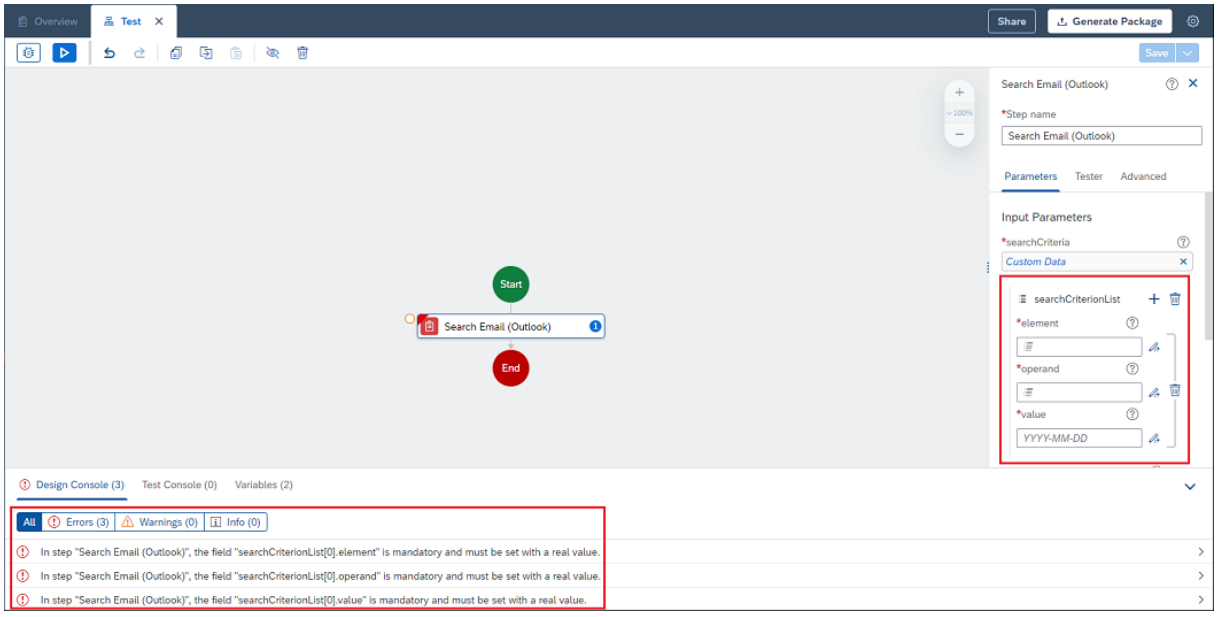
When you create an automation in the Cloud Studio, the data fields that you set can be one of the following types:

- Mandatory Data Field
- Optional Data Field
- Default Data Field
- Sample Data Field

Mandatory Data Field

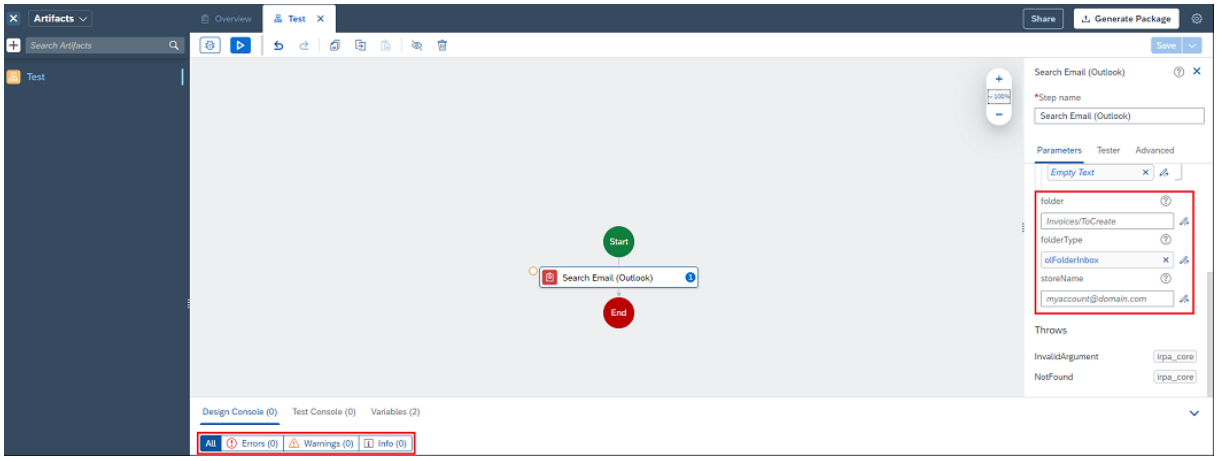
You must set values to the data fields that are marked with a red asterisk (*). These fields are called the mandatory data fields. If you forget to fill the mandatory data fields, you will see the errors displayed in the *Design Console*.

For example, when you use the *Search Email (Outlook)* activity in your automation, if you add a search criterion and keep the mandatory fields empty, errors are displayed in the *Design Console* and you won't be able to launch the test session.



Optional Data Field

You can either set values to the data fields or leave the data fields blank if the data fields are not marked with a red asterisk (*). These data fields are called the optional data fields. If you keep the optional fields empty, you won't see any error displayed in the *Design Console*.

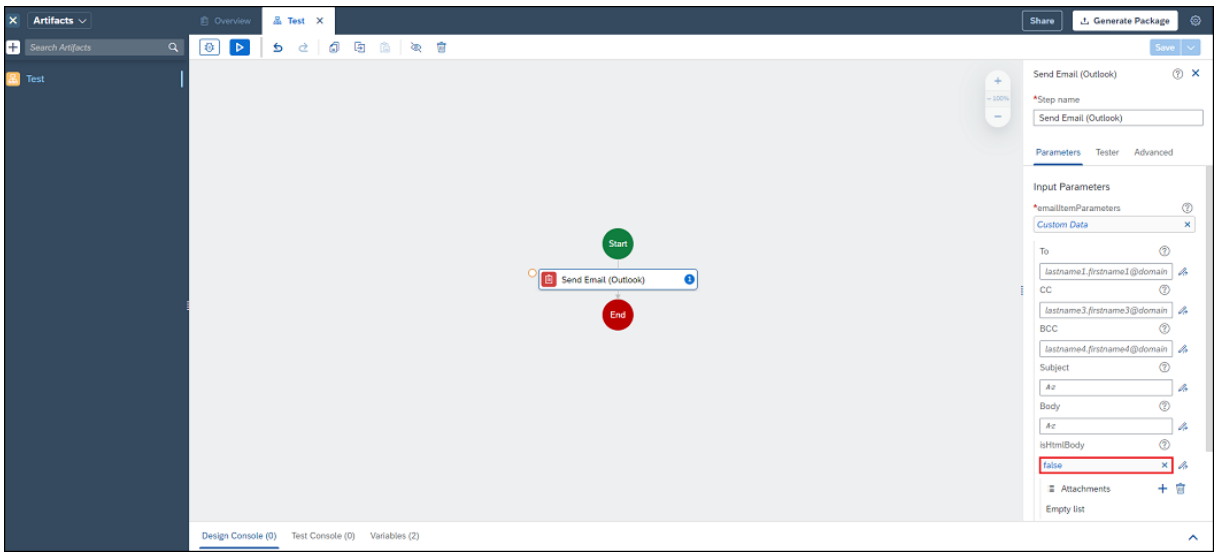


Default Data Field

In the Cloud Studio, the default value is displayed with priority on the default data fields.

For example, when you use the *Send Email (Outlook)* activity in your automation, you can see that *false* is selected as default value for *isHtmlBody* default data field.

You can change the default value by clicking the *X* button and selecting *true*. You can return to the default value by clicking the  button.

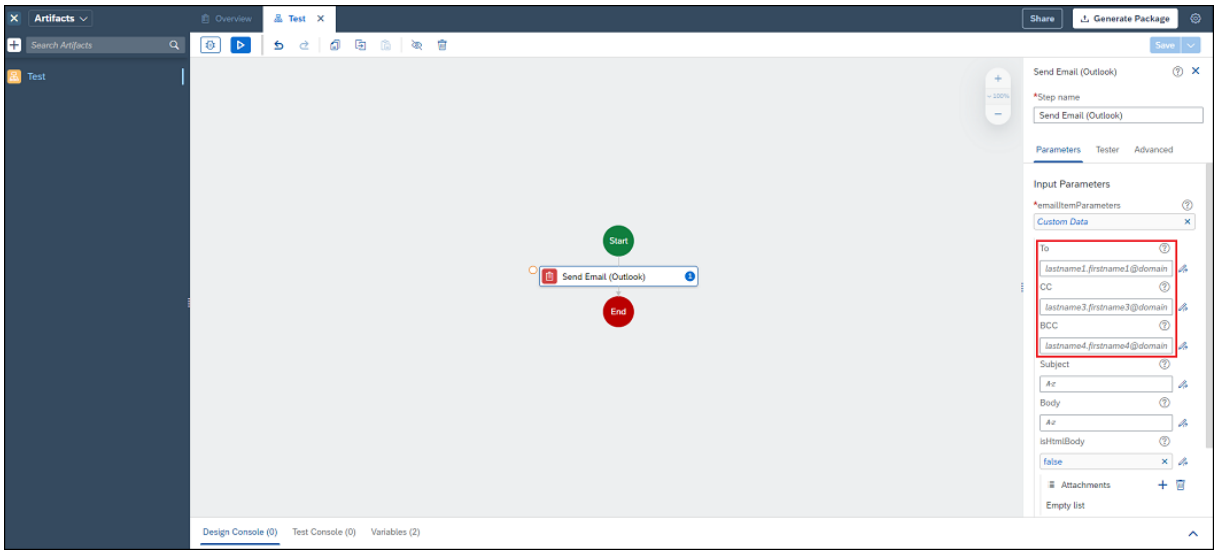


Sample Data Field

In the Cloud Studio, the sample values are the displayed values that help you while setting values to the sample data fields. These values are not considered as the real values because these values only give you the hint of the correct data values.

For example, when you use the *Send Email (Outlook)* activity in your automation, you can see that sample values are displayed on the sample data fields (*To*, *CC* and *BCC*). By seeing the sample values, you can easily understand what values you must set to the sample data fields.

If you enter email address into the sample data fields, the sample values will disappear from the sample data fields. If you remove the defined email address from the sample data fields, the sample values will appear again.



Parent topic: Manage Data within an Automation [page 160]

Previous: Create and Update Variables [page 179]

Next: Expression Editor [page 199]

4.2.3.6 Expression Editor

What is the Expression Editor?

The expression editor allows you to create and edit expressions in your automation.

> **i** Note
>
> An expression is a single code line returning a typed value.

In the interface of the Cloud Studio, expressions can be:

- A constant

    - as a string
    - as a number
    - as a boolean
- A variable: coming from the input of the current automation you are editing, or from the output of a specific step used in the automation

  > **Example**
  >
  > ```
  > Step0.name
  > ```

- A complex expression (for example to express a condition). Unlike constants or variables, you can define a complex expression only by using the expression editor.

  > **Example**
  >
  > ```
  > Step0.name.trim()==="jane.smith"
  > ```

When building an automation in the Cloud Studio, you use expressions in:

- Input and output parameters of steps
  For more information on input and output parameters, see Input/Output Parameters [page 160].
- Conditions: to express the different cases, or in the outputs of a condition

  > **Example**
  >
  > ```
  > Step0.name.trim()==="jane.smith"
  > ```

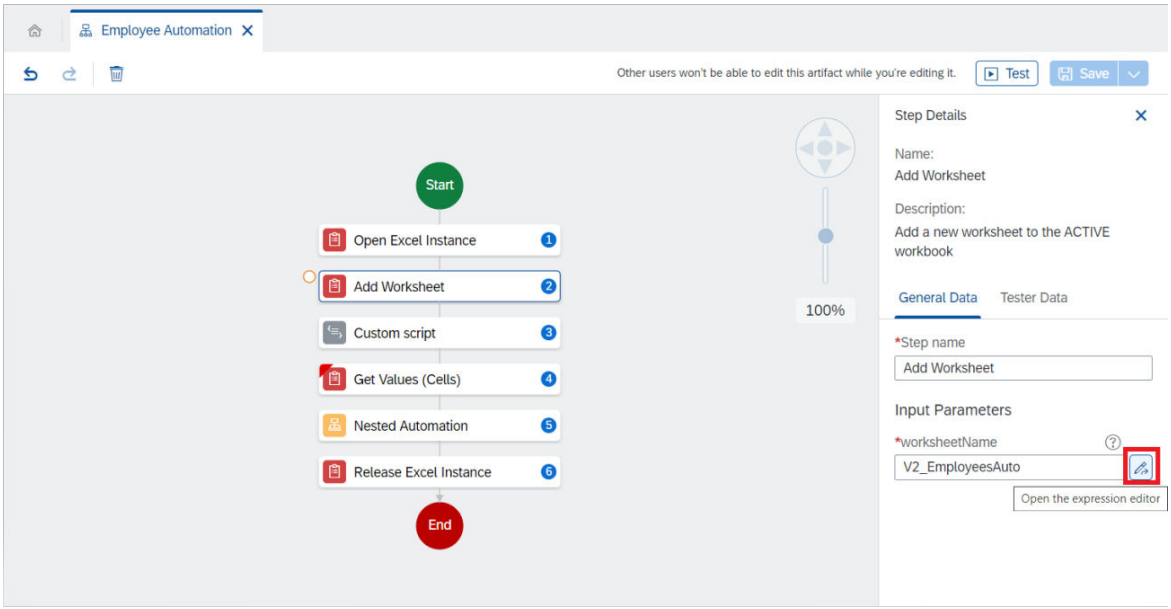
- Loops (for iterators)

  > **Example**
  >
  > ```
  > Step1.nameList
  > ```

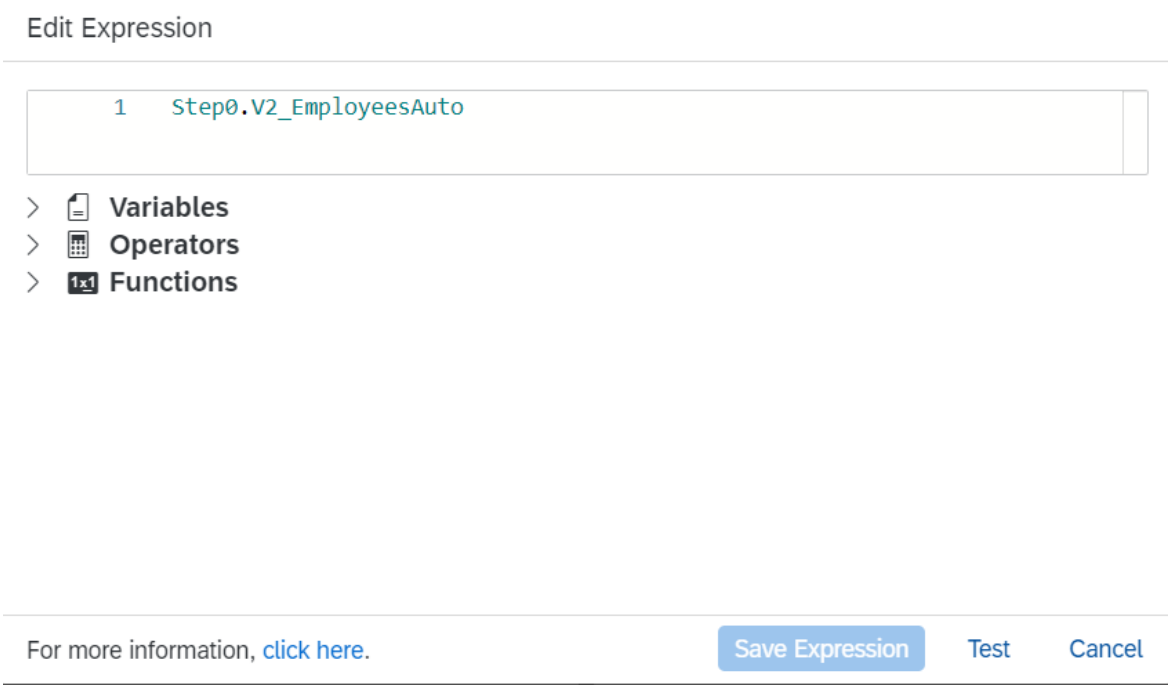
- Output parameters of the automation

Using the Expression Editor

1. In your automation, select a step and click the button on the side panel. The following example shows the expression editor button that you can use to define the input parameter of an activity.



2. The expression editor dialog opens. It's an editor of type JavaScript in which you can manually type your expression, or use the predefined references (variables, operators and functions).



> **i Note**
>
> The variables you use to create an expression with the expression editor must be prefixed with the name of the step they come from. In the example above, `Step0` represents the input of the current automation.

> **i Note**
>
> Auto-completion is available when typing variables and functions (except for functions of type *any*).

3. Click *Test* to check that your expression is valid.
   The test of your expression checks if:
   - the syntax is correct
   - the variables used in the expression exist
   - the functions and operators used in the expression are allowed in the expression editor grammar.
   - the type of the expression is compatible: an expression of type *any* can be of any type, but the other types of expressions must respect the expected type.
4. Click *Save*.

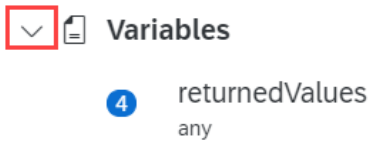
> **i Note**
>
> Don't forget to save your automation after you saved your expression in the formula editor.

References in the Expression Editor

Variables

A variable represents a data and is used to store information. Once you name a variable and define its value and type, you can use and reference it throughout your project.

When you edit the expression of a step, the expression editor shows you the variables and their type, that are available for this step.



In the example above, the variable `returnedValues` comes from the step 4 of the automation and is of type *any*.

Operators

An operator performs a specific operation on a set of expressions. You can use three types of operators:

- arithmetic
- logic
- comparison

In the expression editor, when you click on the dropdown menu next to *Operators*, you can see a list of all the operators available, with a definition. You can clik on any operator to add it to your expression.

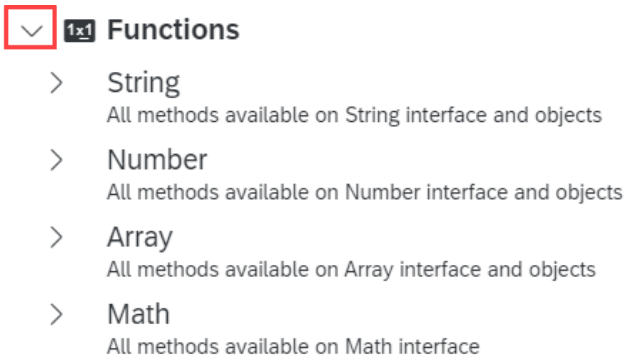


Functions

A function performs a specific operation on the expression (for example remove the space at the beginning and end of a string of characters). Four types of functions are available:

- string
- number
- array
- math

In the expression editor, when you click on the dropdown menu next to *Functions*, you can see a list of all the functions available, with a definition. You can click on any function to add it to your expression.



Parent topic: Manage Data within an Automation [page 160]

Previous: Different Types of Data Fields [page 197]

4.2.3.6.1 Enumerations

4.2.3.6.1.1 Combination of Enumerations

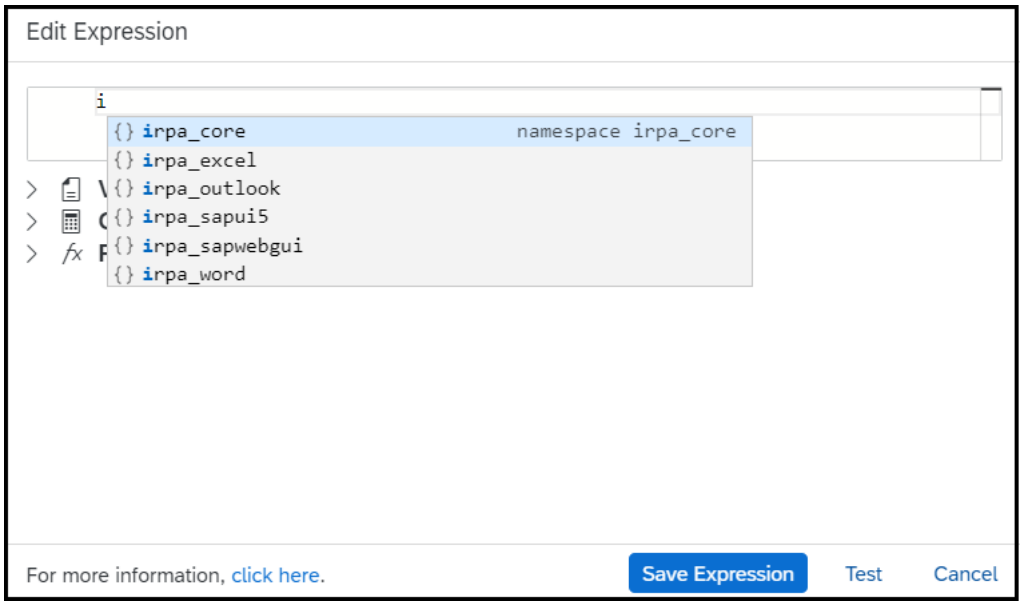
You can now have a powerful way of using enumerations to handle expressions of your automation.

You can use the combination of enumerations inside the expression of an Automation Activities provided by SDK Packages [page 371] to get the desired result from an automation.

The enumerations values are considered as a string and you can use the enumerations on every string input parameter. The enumerations must be handled in the Expression Editor [page 199].

Using Combination of Enumerations

You can use the enumerations coming from SDK Packages [page 372]. However, you must ensure that the enumeration values from various sources and models must be included in the Project.



For more details about SDK packages and their enumerations, refer to the Expression Editor [page 199] section.

The following example shows the combination of enumerations using *irpa_core*.





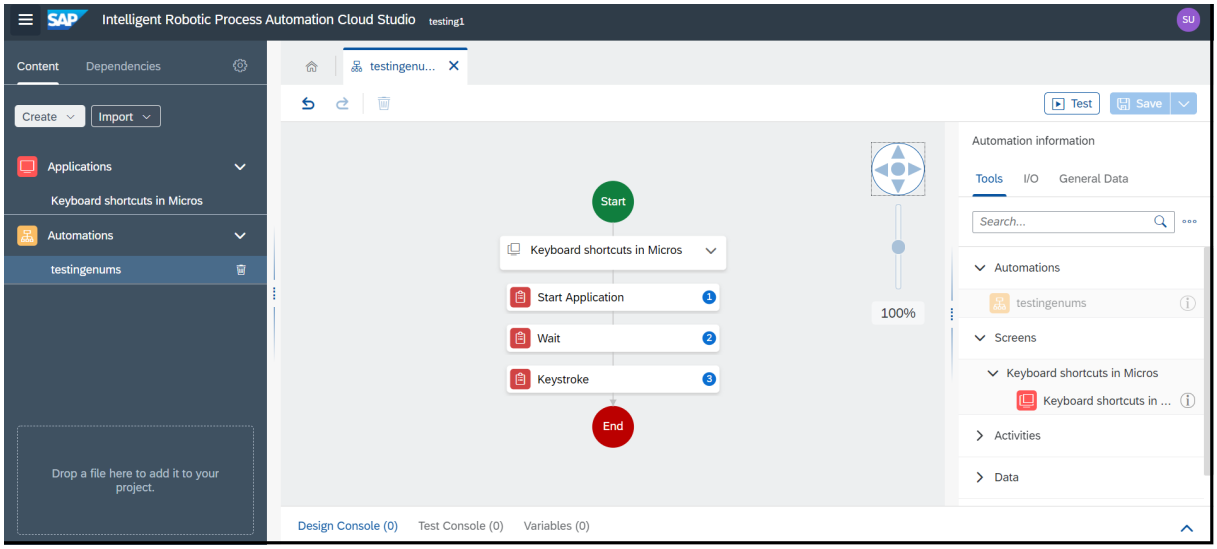
Use Case Example

The following example shows the keyboard shortcuts used to get the expected result.

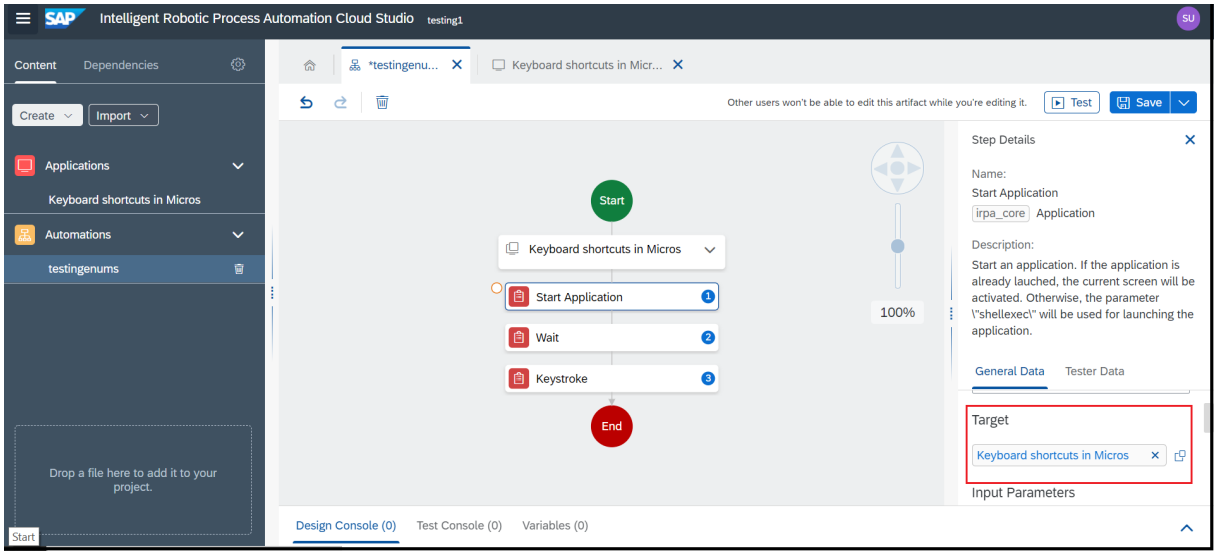
Prerequisites

- You have captured the Microsoft Edge browser application and declared the required elements.

The following screenshot shows the activities used to execute the automation.

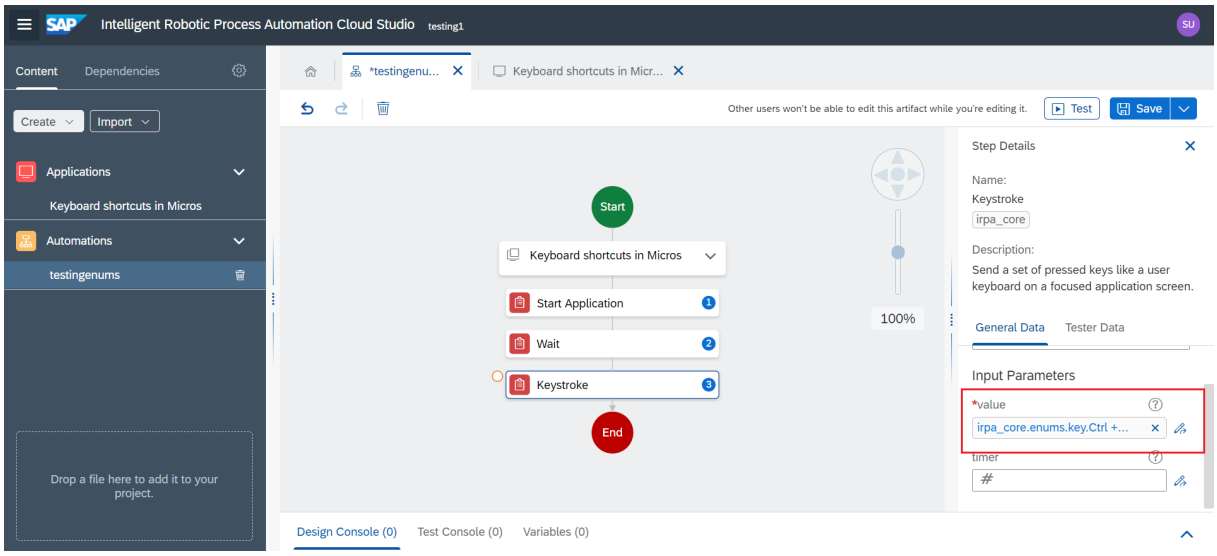


In the *Start Application* activity, you must define the Activity Target Inputs [page 215].



and in the *Keystroke* activity Expression Editor [page 199], you must define the required combination of enumerations value.





Result

Opens Microsoft Edge and uses the keyboard shortcut, Ctrl + F which enables the *Find on page* in Microsoft Edge.

4.2.3.6.2 Dates

> **i Note**
>
> Dates are expressed according to the *ISO 8601* standard. For more information, see Date and Time Data Type Variables [page 194].

You can use dedicated data types to set dates in the Cloud Studio. However, if you want to manipulate dates with all the APIs available in JavaScript, you must convert them into JavaScript dates. To perform this conversion, you can use the *Expression Editor*.

To convert dates into JavaScript ones, use the following syntax: `new Date(yourDate)`



Thus, you will be able to access the APIs available in JavaScript, as shown in the screenshot above.

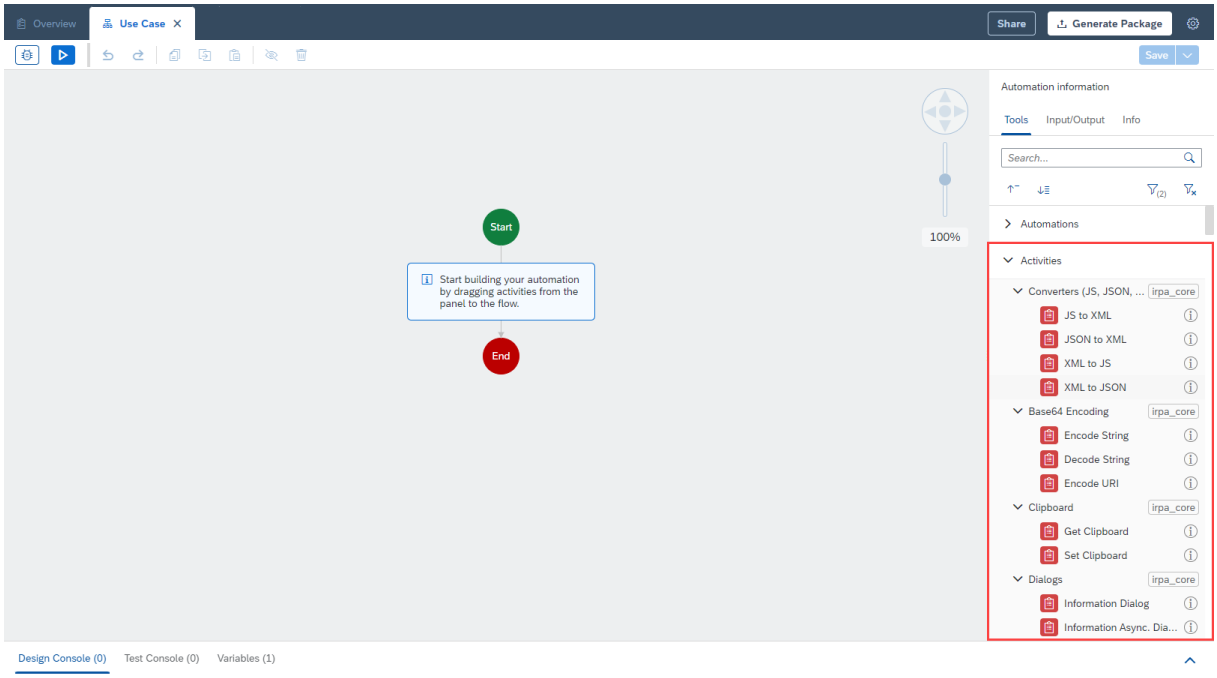
Related Information

Expression Editor [page 199]

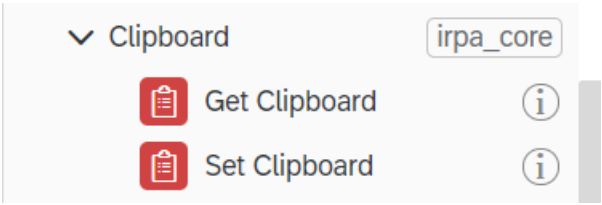
4.2.4 Add an Activity to an Automation

In the Cloud Studio, activities are provided with the SDK packages you import the first time you create an automation, to build the workflow of your automation.

Activities are organized in classes and subclasses in the tools section on the side panel of the automation.



Each activity executes an action according to the class it belongs to. The following example displays the activities in the **irpa-core** subclass activity **Clipboard**.



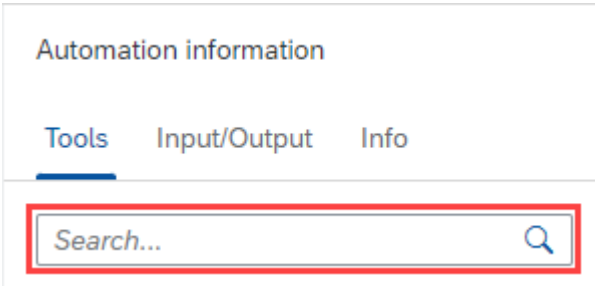
The activities in this subclass perform actions linked to the clipboard:

- **Get Clipboard**: Retrive the textual content of the clipboard.
- **Set Clipboard**: Clear the clipboard and then set text on clipboard.

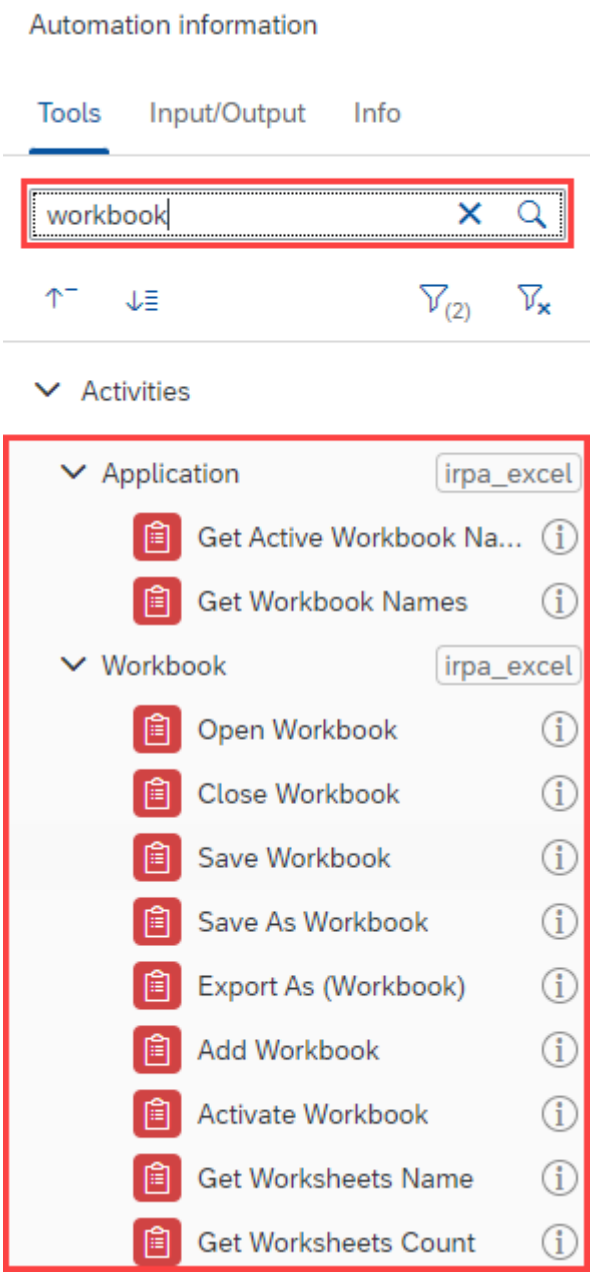
To build your automation you can drag and drop the activities in your workflow where they become the steps of your automation. Each activitiy has different parameters that you can define such as the name of the step, and the input and output parameters.

Filtering Feature

To help you look for an activity, the Cloud Studio provides a filtering feature.

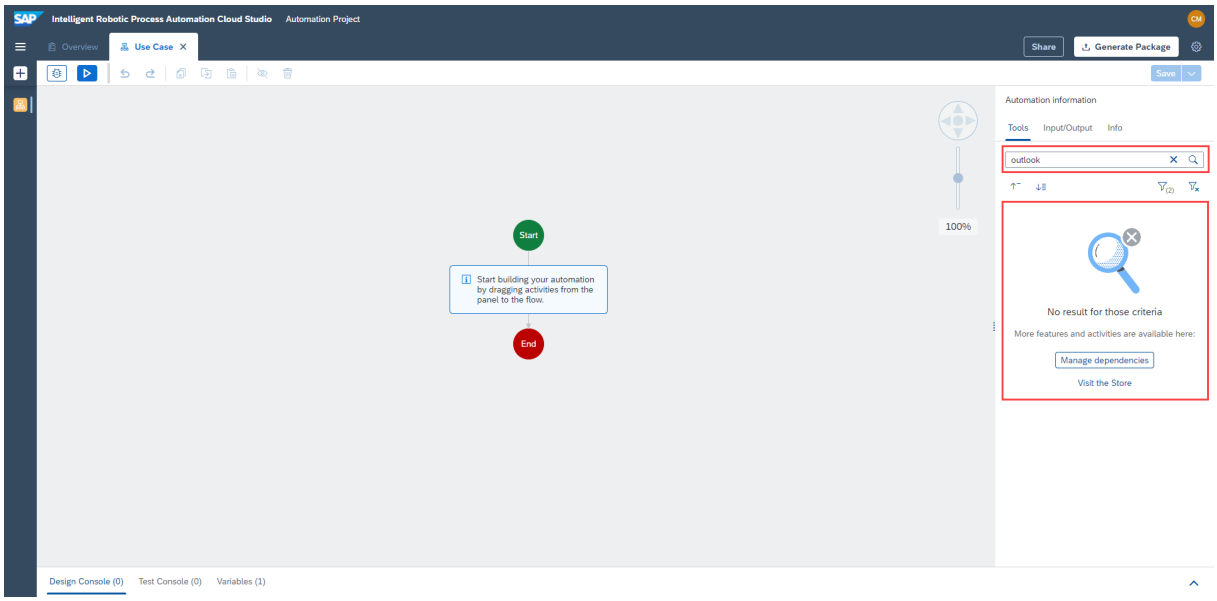


The filtering feature displays the tools from the *Tools* side panel that match your search text. The following example displays the filtering feature when you search for a workbook activity.



4.2.4.1 More Activities

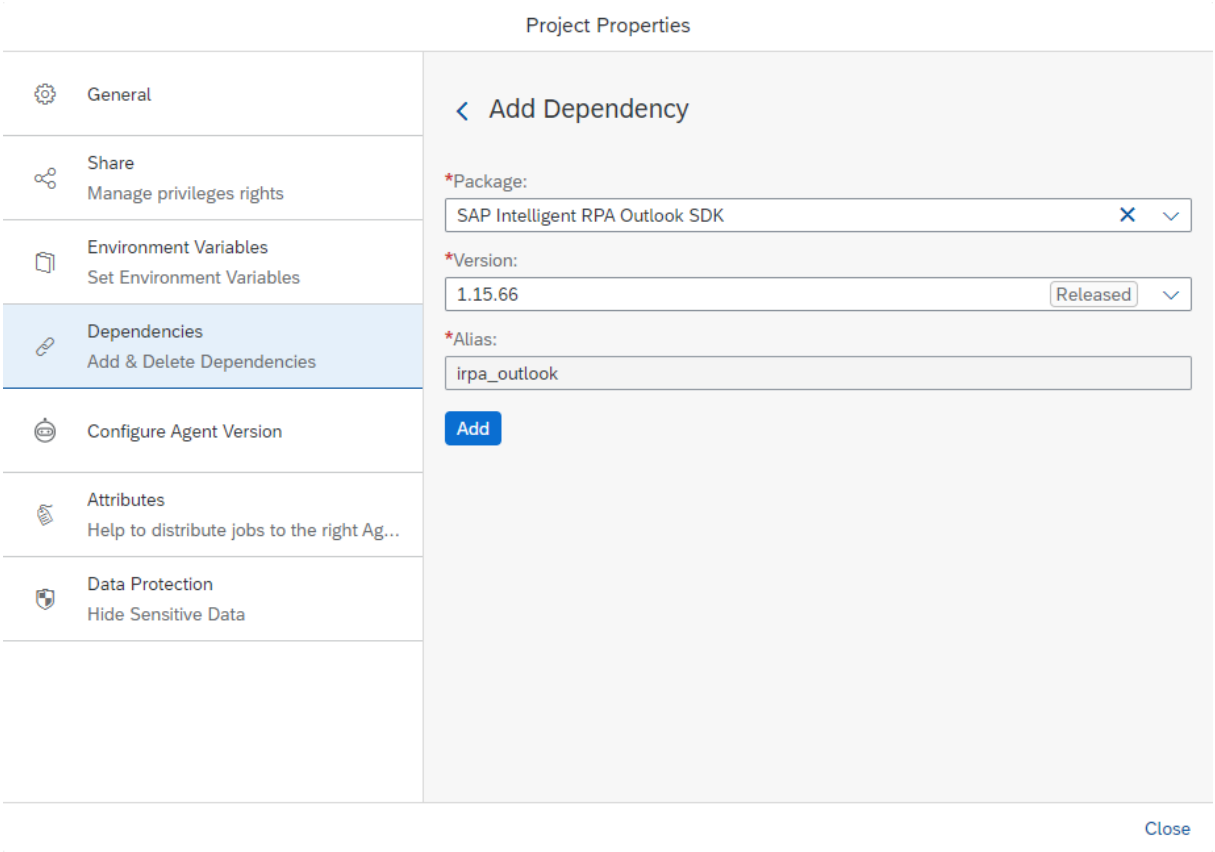
When you search for activities, controls, or data in the Search bar of *Tools* and if you find no result for your search criteria, then the Cloud Studio suggests more options to find appropriate items.



Manage Dependencies

Use this option to add appropriate dependencies.

To add appropriate dependencies, click *Manage dependencies* and you will be redirected to the *Add Dependency Screen*.

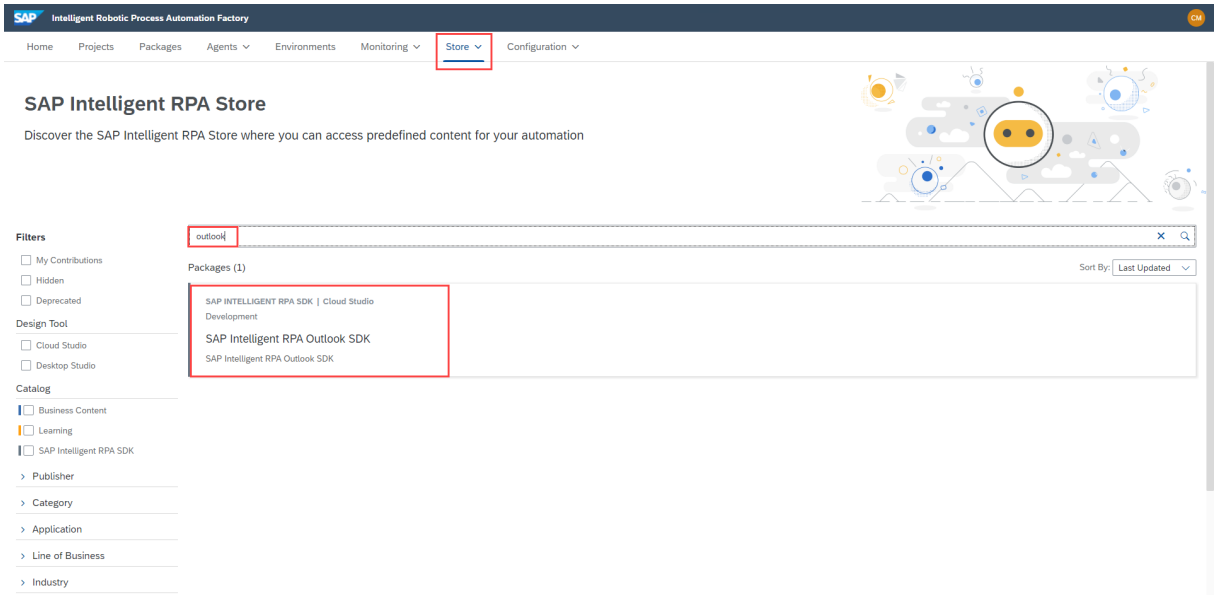


To know more details about the dependencies and adding appropriate dependencies, refer to the Manage Project Dependencies [page 526] section.

Visit the Store

Use this option to search for appropriate predefined content for your automation in Store.

To visit the Store, click *Visit the Store*. You will be redirected to Store, where you can access predefined content for your automation.



To know more details about the Store, refer to the Store section.

4.2.4.1.1 Display an Activity Status

You can view the status (Alpha, Beta, Active, Deprecated, and Decommissioned) of an activity in an automation workflow.



If an activity status is Alpha, Beta, or Deprecated then you cannot use that activity in the production.

In an automation workflow, right-click on a deprecated activity to view the substitute activity suggested by the system. If the suggested activity has the same output and input as the deprecated activity, then you can keep the value of input and output parameters.

For more details about the activity status, refer to the Lifecycle of SDK Activities [page 474] section.

4.2.5 Add an Application Screen to an Automation

Prerequisites

You've captured and declared an application, at least one screen, and the screen elements you want to automate, as described in Capture an Application [page 77].

You've tested the application whose screen or screens you want to automate as described in Test an Application [page 129].

Procedure

1. Open your automation.
2. Open the *Screens* list in the right-hand panel.
3. Find the screen you want to automate under the associated application, and drag and drop it into the workflow.
4. Click the screen instance in the workflow to open the *Screen* panel.
5. Click the *Define Screen Activities* button.

   A screen preview is shown.

   > → Tip
   >
   > Zoom into the screen preview to drop activities onto UI elements more easily.

6. Open the *Activities* list.
7. **Optional:** Type an activity name or part of the name in the search bar to quickly find an activity.

   Only activities matching the name you typed are shown.
8. Drag and drop each activity you need - either anywhere on the screen preview, or onto the corresponding UI element, depending on the activity.

   > Example
   >
   > 1. Type 'Start' in the search bar.
   > 2. Drop the *Start Page* activity from the *Page* class anywhere on the screen preview.

9. Where relevant, set values for the activity, such as for an input parameter.

   > Example
   >
   > 1. Type 'Set' in the search bar.
   > 2. Drag and drop the *Set* activity from the *Item* class onto the search bar in the screen preview.
   > 3. Click the activity step in the workflow, and on the right, set the required value under *Input Parameters*.

> **Note**
>
> When you use a *Get* activity, an output parameter with the appropriate type for the activity - such as text - is set automatically.

10. Save your work.

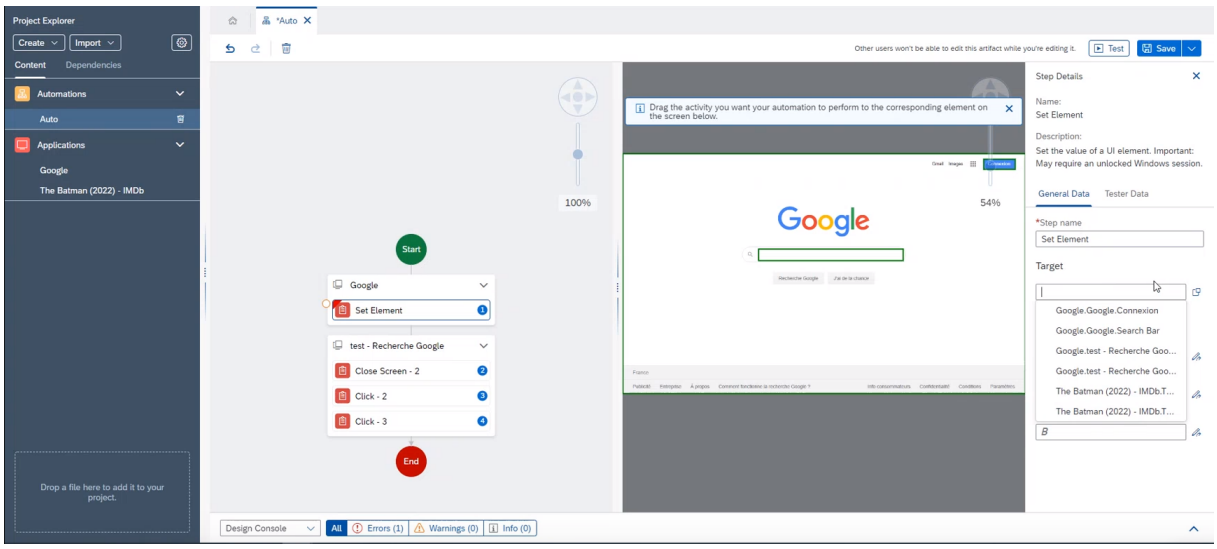
> **Note**
>
> Remove a screen or an activity step from a workflow by clicking it and choosing 🗑.

4.2.5.1 Activity Target Inputs

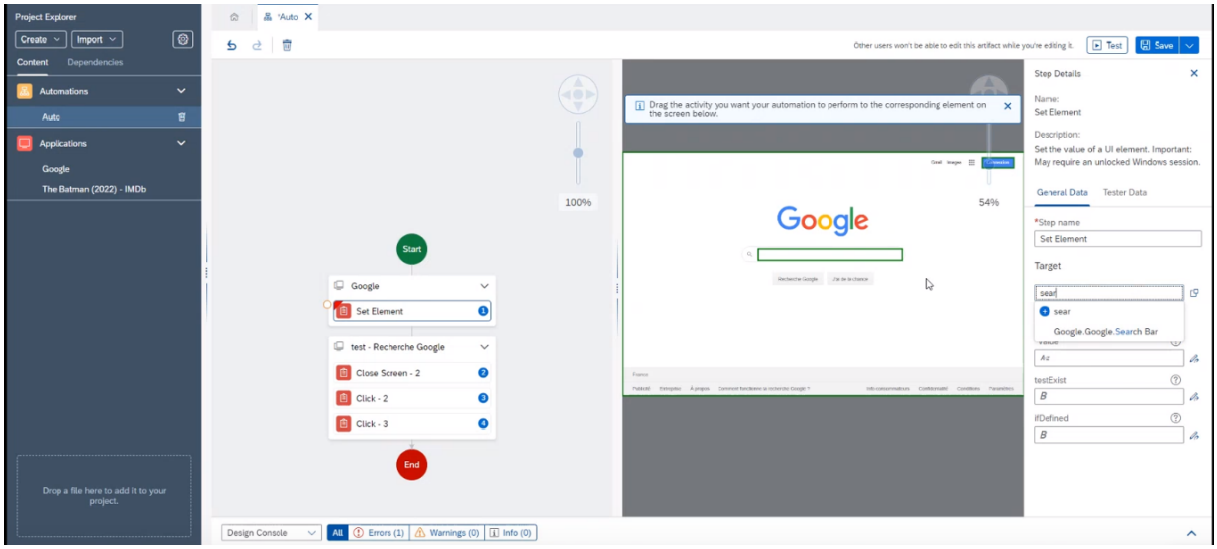
You can search for screens and elements in the suggest input (*Target* field).

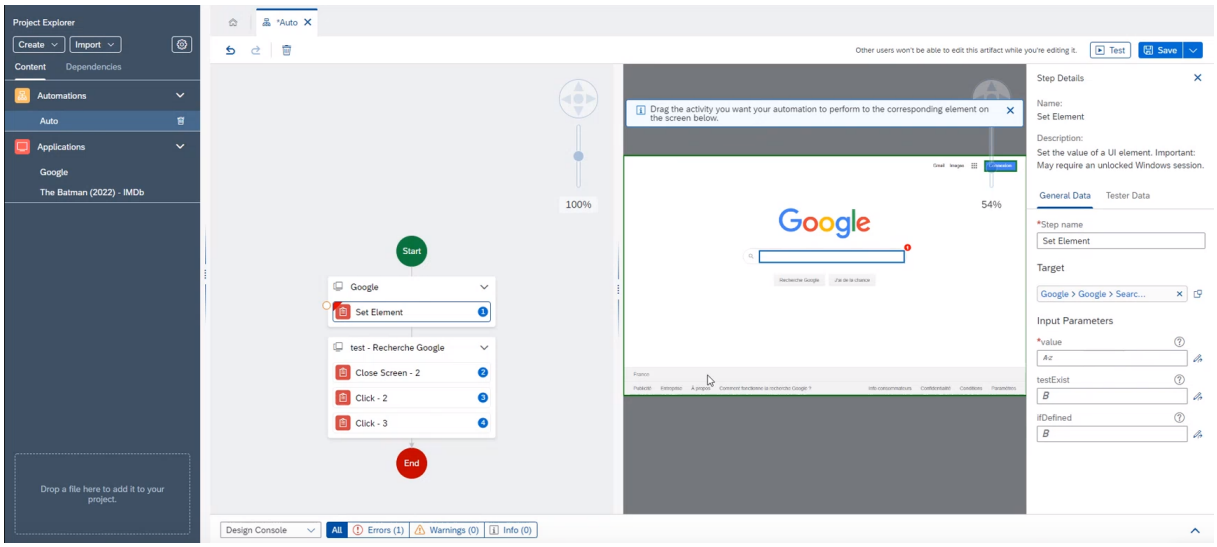
In a workflow, if you drag and drop an application screen and a required activity, (for more details about activities, refer to the Automation Activities provided by SDK Packages [page 371] section.) then you can find the target inputs (screens and elements) corresponding to the activity displayed in the *Target* field.

In the following example, the target inputs are displayed for the Set Element activity.



You can use the target inputs to build the workflow of your automation.





4.2.6 Add a Control to an Automation

In the Cloud Studio, controls allow you to add tools such as conditions and loops in the workflow of your automation. The following topics explain the purpose of each control.

Loop Controls [page 217]

Condition Controls [page 231]

Other Controls [page 246]

Reuse Variable Defined in Any Block [page 257]

4.2.6.1 Loop Controls

For Each [page 217]
This control allows you to go through a list of members provided as input to your automation, and execute an action for each member in that list.

Forever [page 224]
This control allows you to repeat a step in a loop until it meets the required conditions.

Repeat [page 226]
This control allows you to define a loop that will repeat a step for a defined number of times.

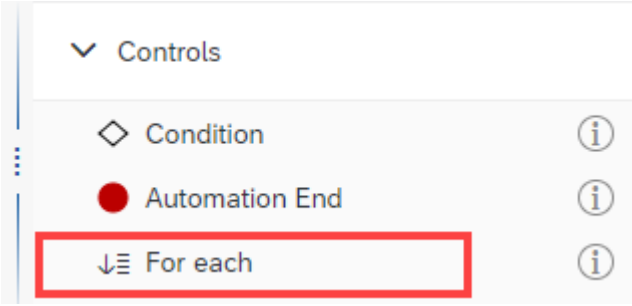
Loop End [page 230]
This control allows you to exit a loop before its original end.

4.2.6.1.1 For Each

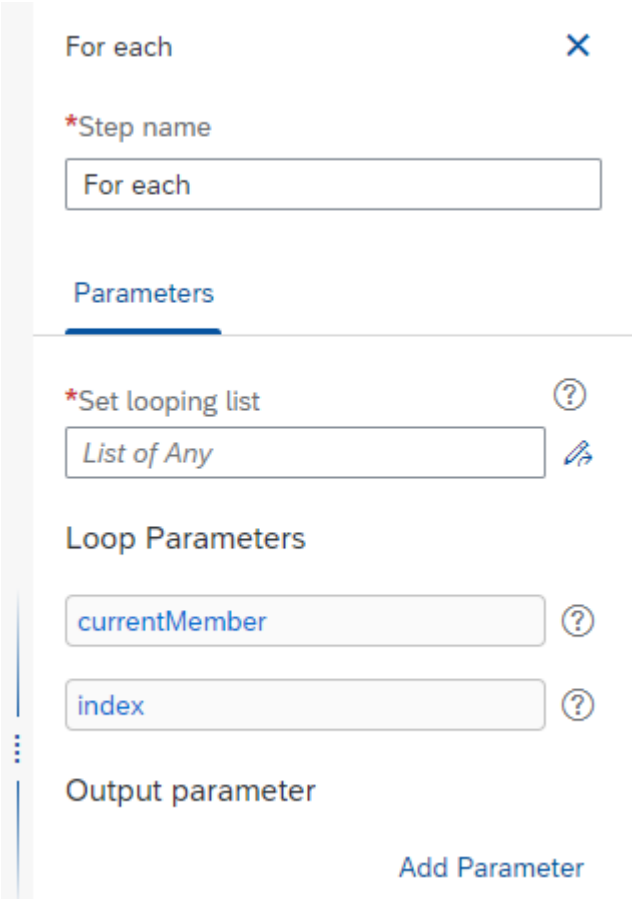
This control allows you to go through a list of members provided as input to your automation, and execute an action for each member in that list.

Using the For Each Control

1. In your automation, on the right-hand side panel, go to the *Controls* section.



2. Drag and drop *For Each* in a workflow of your automation.
3. Select *For Each*. The *For each* side panel appears.



For Each control has the following loop parameters:

- *currentMember* - The member of the list for the current loop iteration.
- *index* - An integer that is the index of the current loop iteration, starts at 0.

4. Provide an appropriate input in the **Set looping list*.
   The input must be a list of elements you want to loop.



5. Drag and drop an automation, activity, or other controls such as condition in a sequence within the *For Each* control in a workflow of your automation.
6. Select the automation, activity, or screen. The *Step Details* side panel appears, where you can set the currentMember or index as an input of the activity.
7. Click *Save*. The automation is saved.

Use Case Example

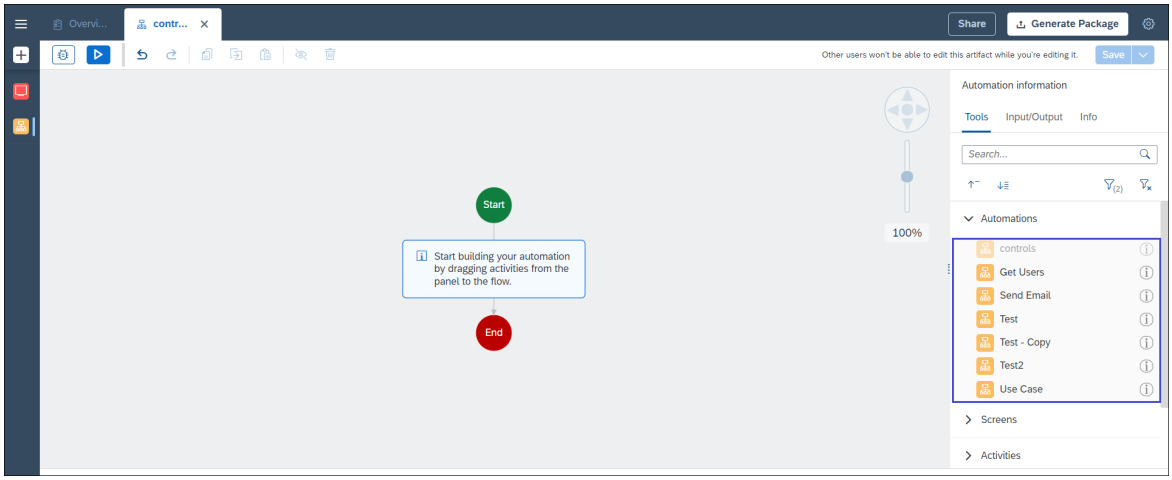
The following use case retrieves the list of users from another automation and sends an email with a message to all users.

Prerequisites

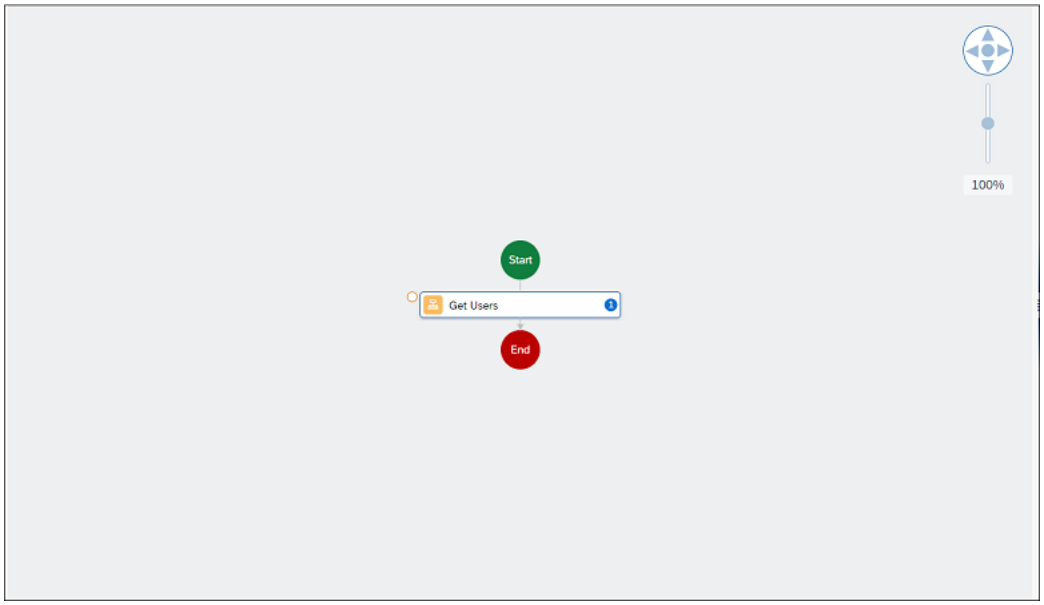
- You have created the automations **Use Case** and **Send Email**.
- You have created the automation **Get Users** with a list of users as an output parameter.

Procedure

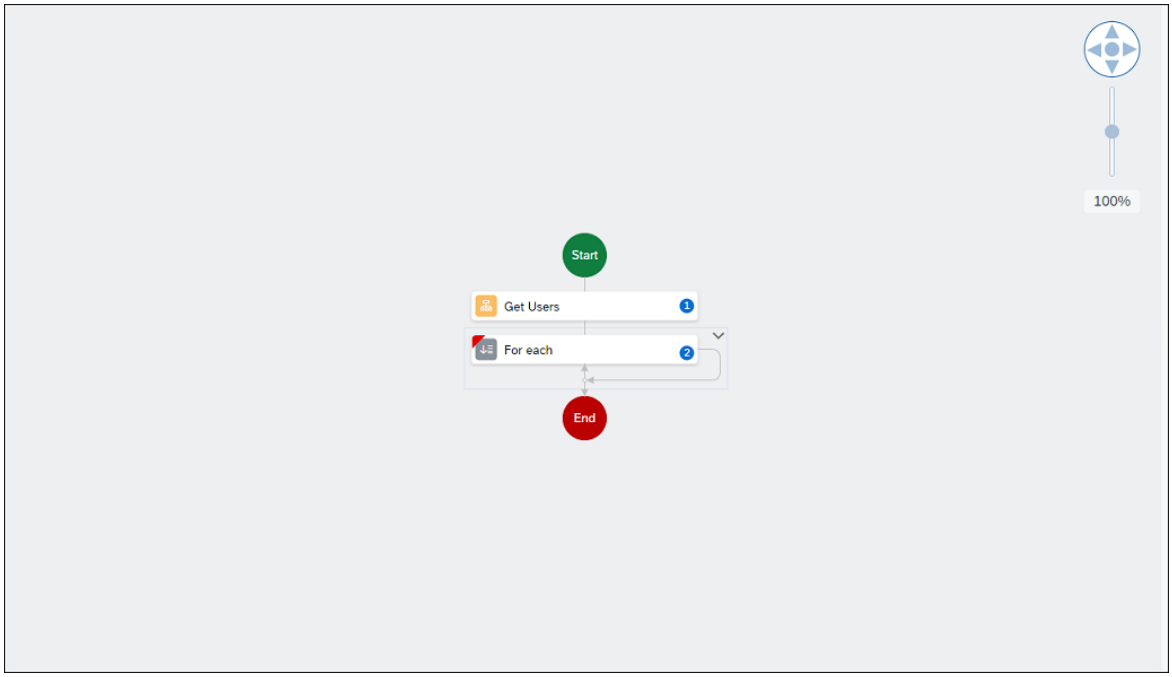
1. In your automation, on the right-hand side panel, under *Tools*, click *Automations*.
   The automations list is displayed.



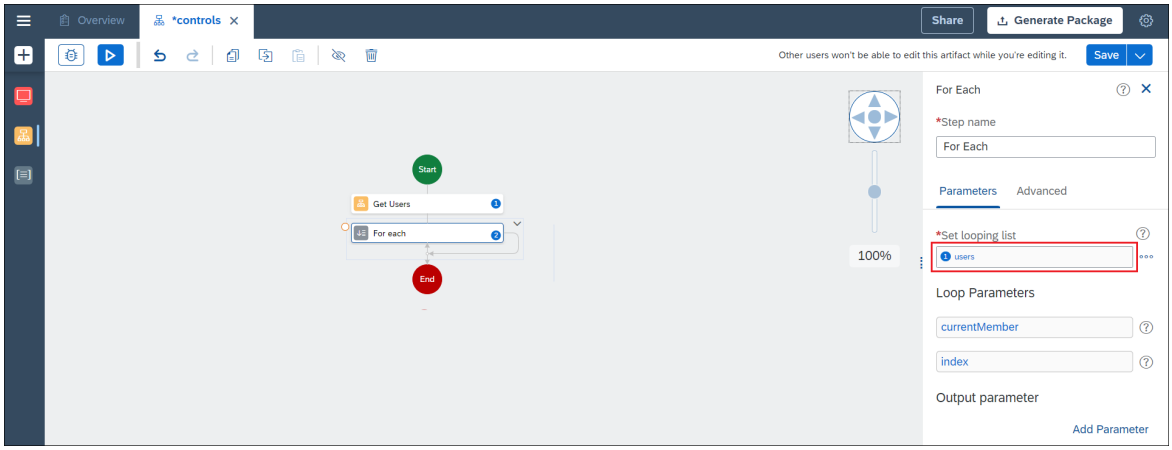
2. Drag and drop the Get Users automation in your workflow.



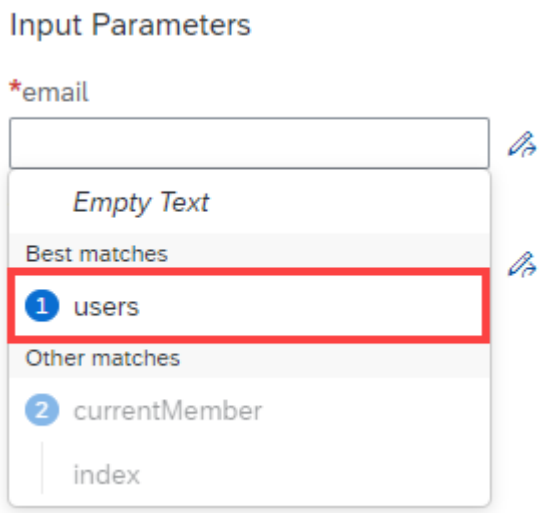
3. Drag and drop the *For Each* control in your workflow.



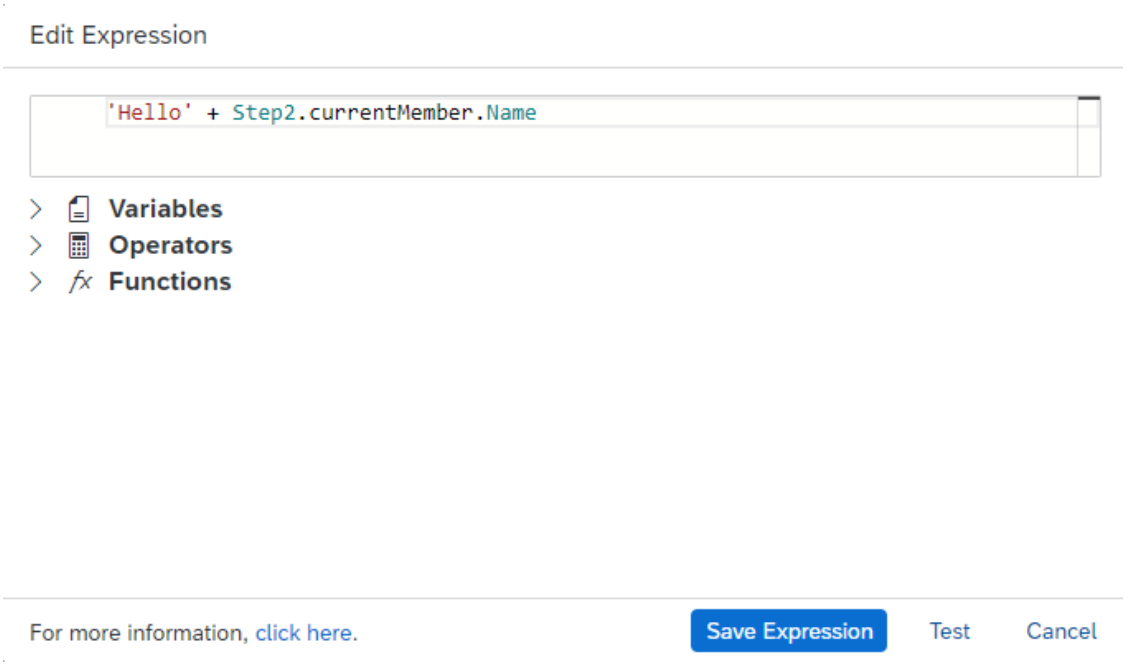
4. Select *For Each*. On the side panel, provide an appropriate input in the *Set looping list* field.
   The input must be a list of elements you want to loop.



5. Drag and drop the Send Email automation in your workflow.
6. Select the *Send Email* automation.
7. Provide an appropriate input in the *email* and *message* fields:
   1. Click in the *email* field and select an appropriate option from the drop-down list.



2. To edit the message expression, click the button on the right of the *message* field. The Edit Expression dialog opens.
   You can manually type your expression.



   3. Click *Test* to check your expression is valid.
   4. Click *Save Expression*.

8. Save your automation and test it.

Result

For each loop iteration, an email is sent to a member of the list.

4.2.6.1.1.1 Assign List of Screen Elements for For Each

You can assign a list of screen elements from Set Looping List or Target Selection in a For Each loop.

> **i Note**
>
> You can assign only a screen element which is a collection.
>
> In the *Set looping list*, only a list of compatible screens is displayed.
>
> In the *Target Selection*, you can select only the compatible screens. The incompatible screens are greyed out and the selection option is disabled.

Prerequisites

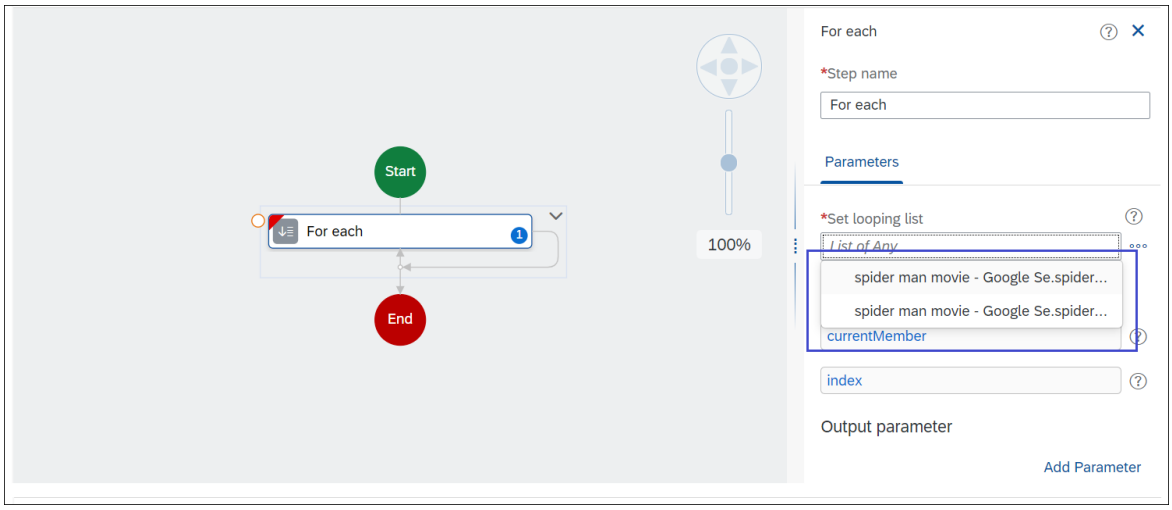
You must capture an application and at least one screen. To know more details about capturing an application screen, refer to the Capture an Application [page 77] section.

You must declare an element as a collection. To know more details about declare an element as a collection, refer to the Declare an Element as a Collection [page 120] section.

Procedure

Follow the below procedure to assign a list of screen elements from *Set looping list*:

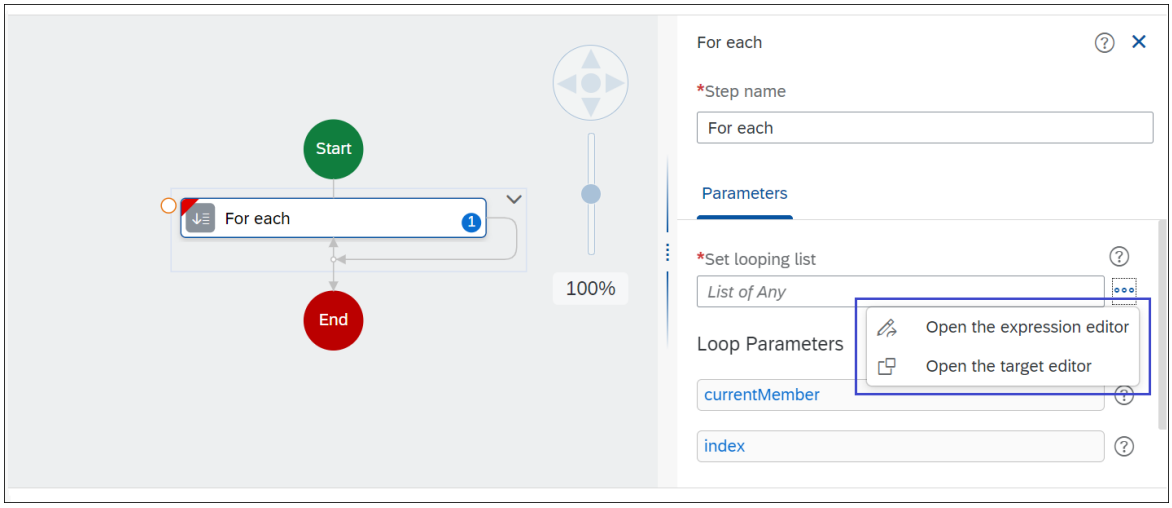
1. In an automation, select *For Each* and then click in the *List of Any* field. The list of compatible screens is displayed.



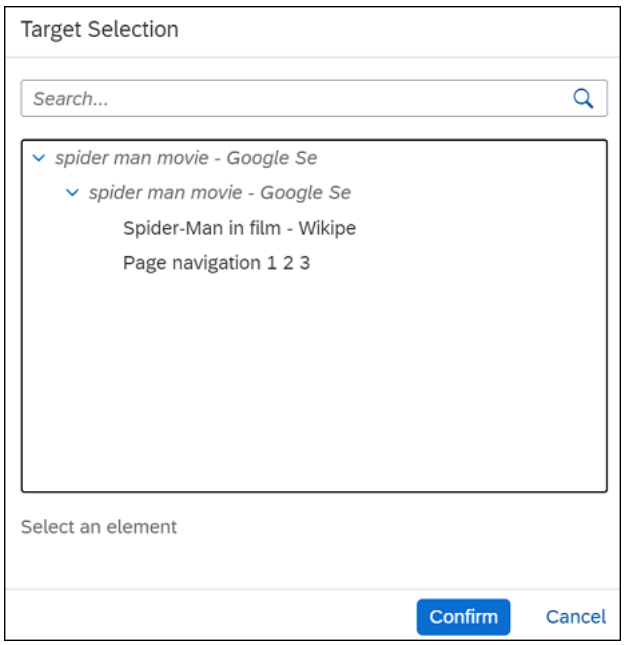
2. Select the required screen from the list and then click *Save*. The screen element which is collection is now assigned in a For Each loop.

Follow the below procedure to assign a list of screen elements from *Target Selection*:

1. In an automation, select For Each and then click ⋯. The more options list is displayed.



2. Click *Open the target editor*. The *Target Selection* dialog is displayed.



3. In the *Target Selection* dialog, select the required screen and then click *Confirm*. The automation screen is displayed.
   The screen element which is collection is now assigned in a For Each loop.

Tips

You can view screen item datatype in a *Variable* panel (for currentmember when you loop on a collection of screen items).

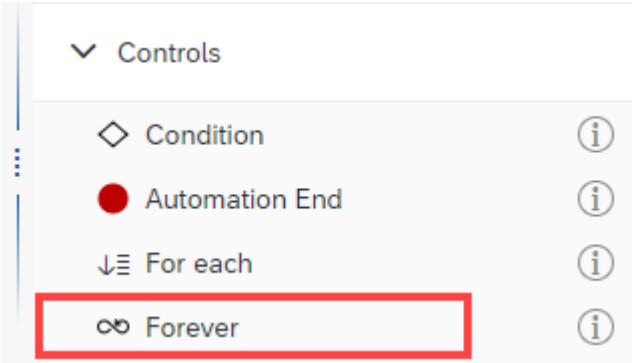
You can use the expression (list of screen elements) of a For Each in a new For Each loop by copying an expression of a For Each and pasting it in an Expression Editor of a new For Each.

4.2.6.1.2 Forever

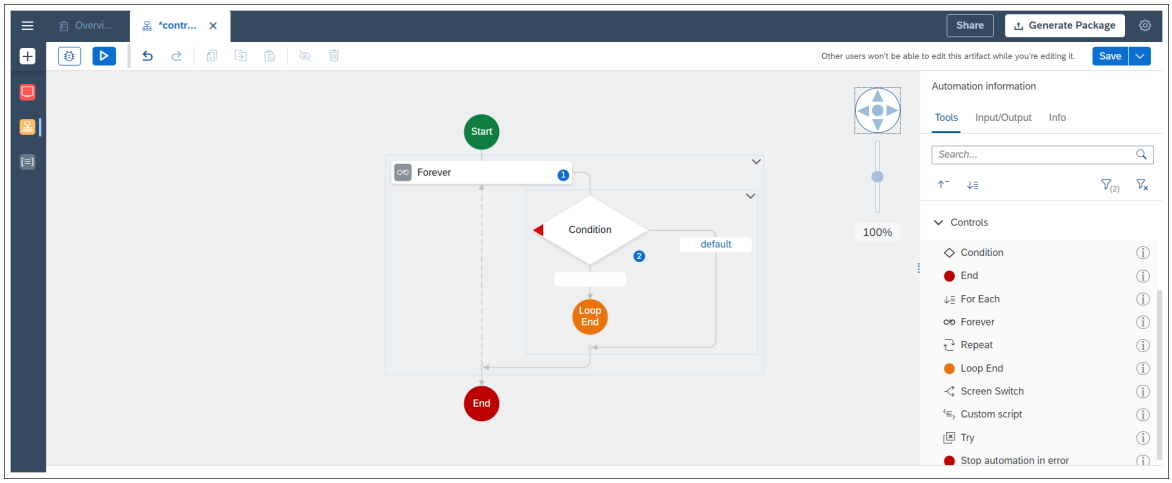
This control allows you to repeat a step in a loop until it meets the required conditions.

Using the Forever Control

1. In your automation, on the right-hand side panel, go to the *Controls* section.

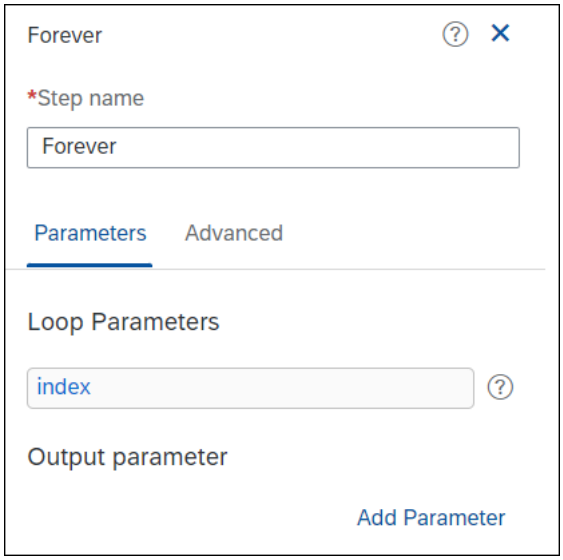


2. Drag and drop *Forever* in the workflow of your automation.



> **Note**
>
> This loop comes with a condition control [page 231] and a loop end control [page 230]. However, you can remove these controls and/or add additional controls in the forever loop.

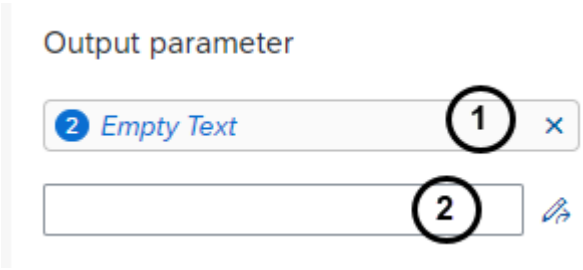
3. Select *Forever*. The *Forever* side panel appears.



The **Forever** control has the following loop parameter:

- *index* - An integer that is the index of the current loop iteration, starts at 0.

4. Set a condition expression. For more information, see Condition [page 231]
   Once the condition you set is met, loop end control allows the automation to exit the forever control, and continue on the main workflow.
5. **Optional**: Add an output parameter to the forever control.



   1. Enter a name.(1)
   2. Set a value for the output parameter.(2)

      > **i Note**
      >
      > You can select an existing output parameter, or edit the output parameter in the expression editor.

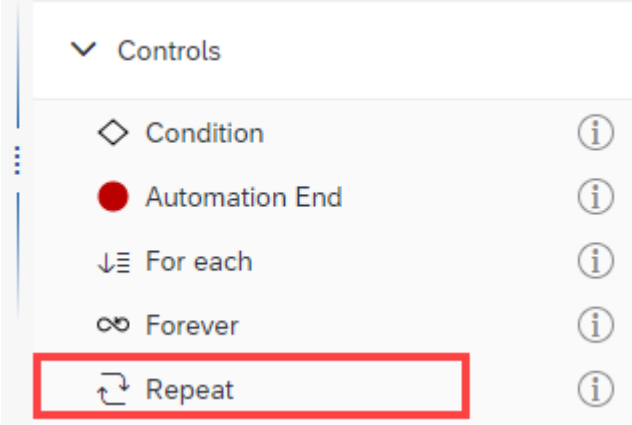
6. Save your automation.

4.2.6.1.3 Repeat

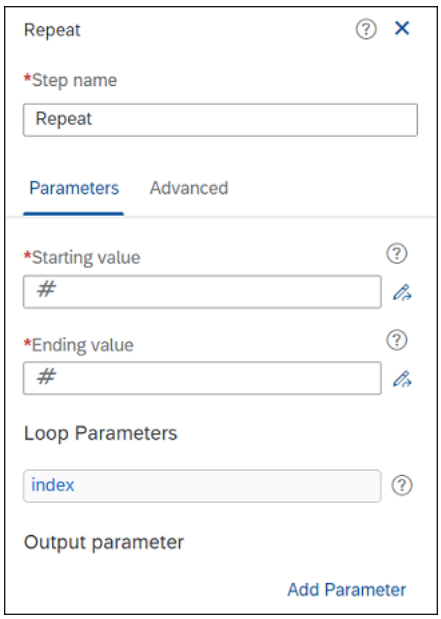
This control allows you to define a loop that will repeat a step for a defined number of times.

Using the Repeat Control

1. In your automation, on the right-hand side panel, go to the *Controls* section.



2. Drag and drop *Repeat* in the workflow of your automation.
3. Select *Repeat*. The *Repeat* side panel appears.



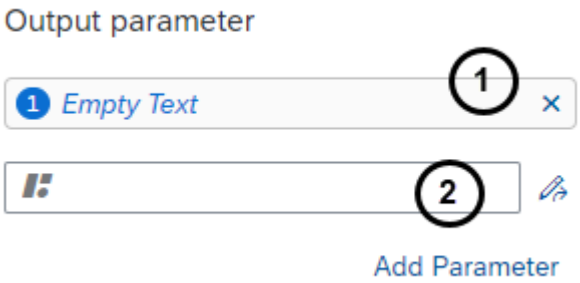
The Repeat control has the following parameters:

- *Starting value*: Set a starting number where the iteration starts.
- *Ending value*: Set an ending number where the iteration stops.

> **Note**
>
> The starting and ending values must be numeric values.

- *index*: A number that starts at the set starting value, and increases by one with every loop until it reaches the set ending value.

4. **Optional:** Add an output parameter to the repeat control.



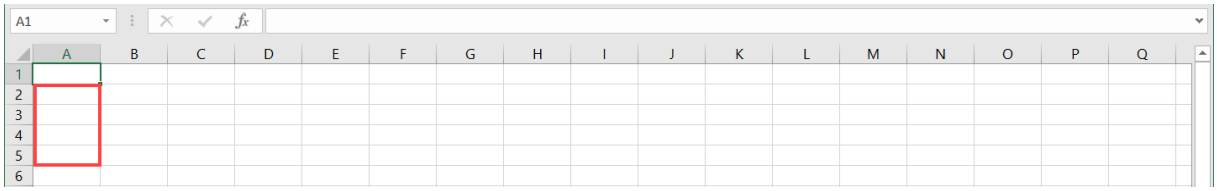
   1. Enter a name.(1)
   2. Set a value for the output parameter.(2)

> **i Note**
>
> You can select an existing output parameter, or edit the output parameter in the expression editor by clicking .

5. Save your automation.

Use Case Example

The following use case demonstrates the use of the repeat control by setting values in an Excel file.

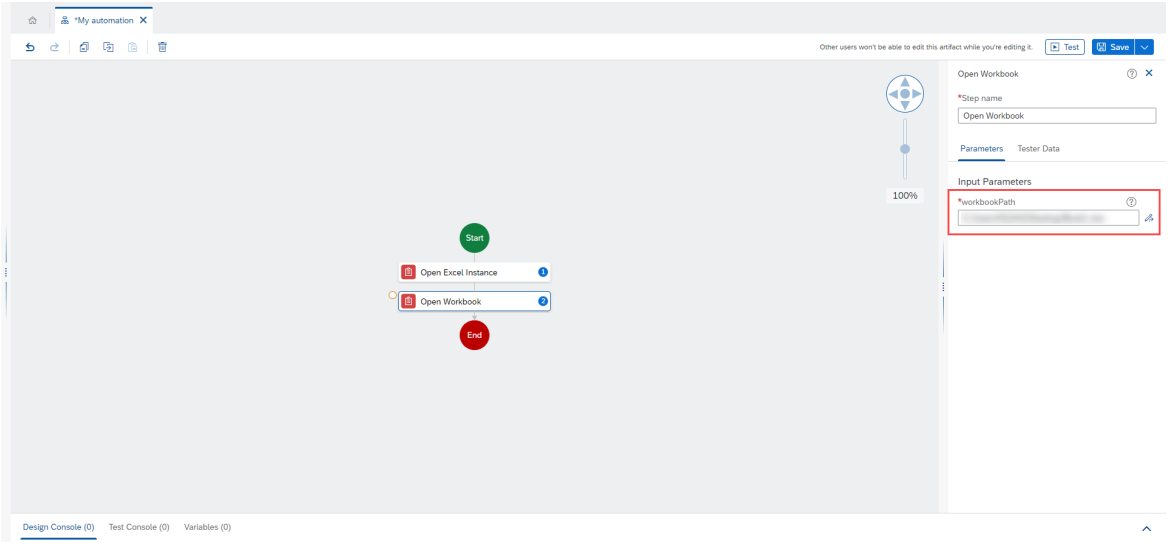


Prerequisite

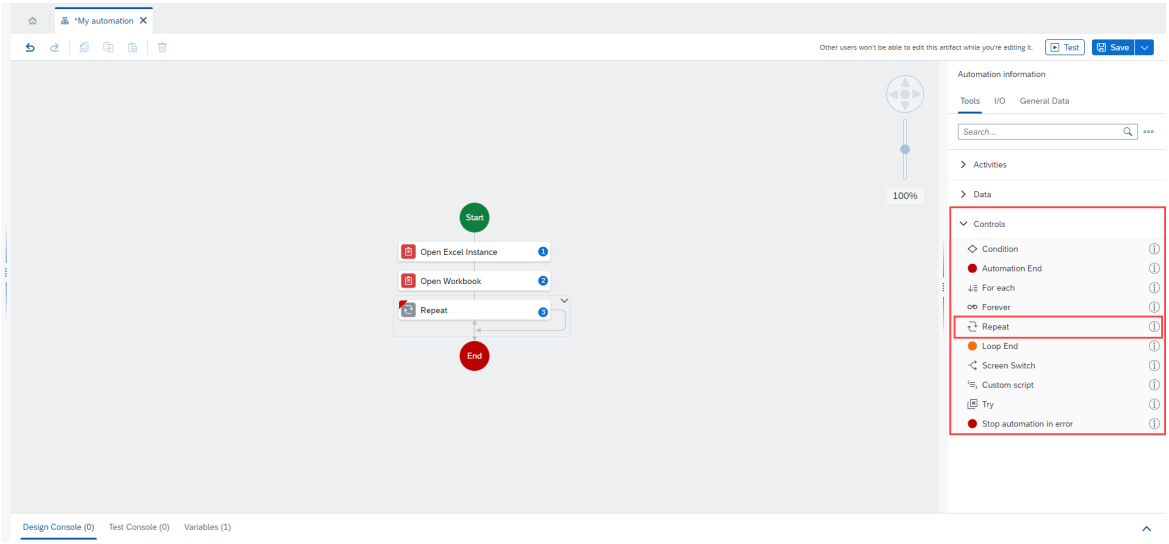
You have created the automation *My automation*.

Procedure

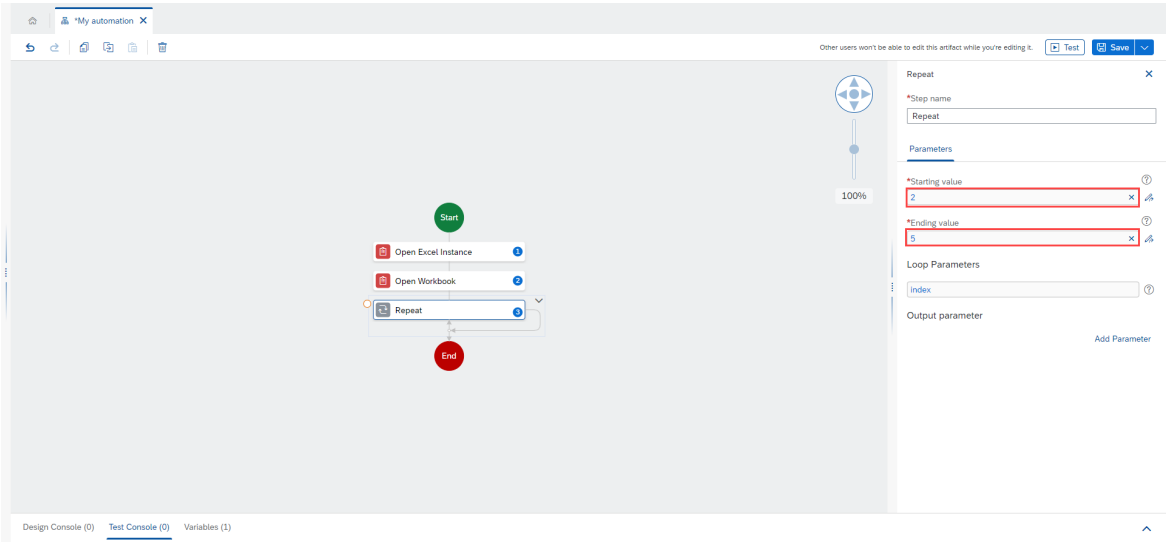
1. In your automation, on the right-hand side panel, go to the *Activities* section.
2. Drag and drop the activity *Open Excel Instance*.
3. Drag and drop the activity *Open Workbook*.
   Write the path of your Excel file in the input parameters.



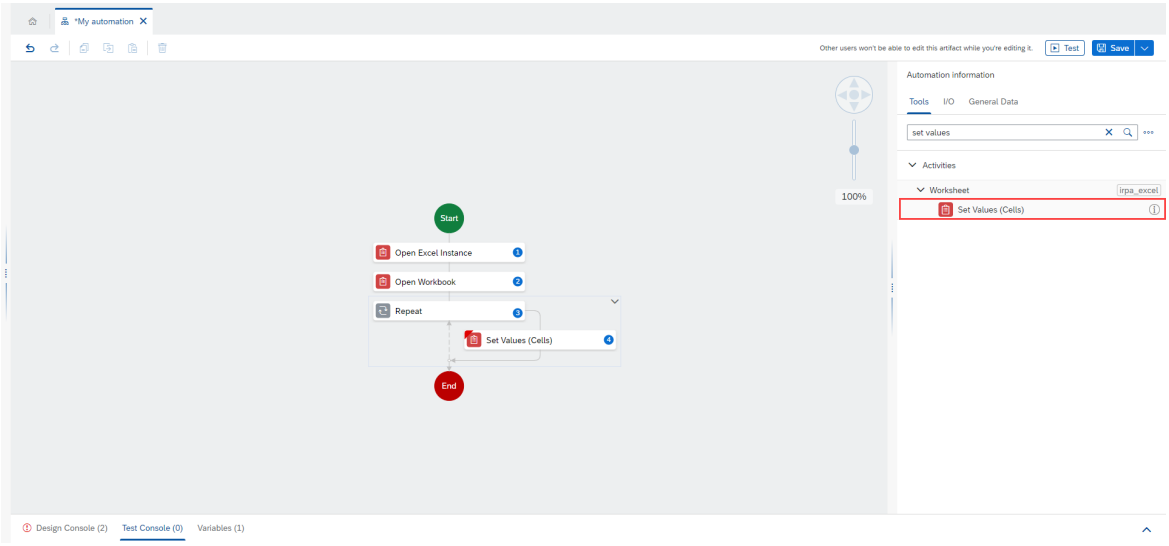
4. Drag and drop the *Repeat* control in your workflow.



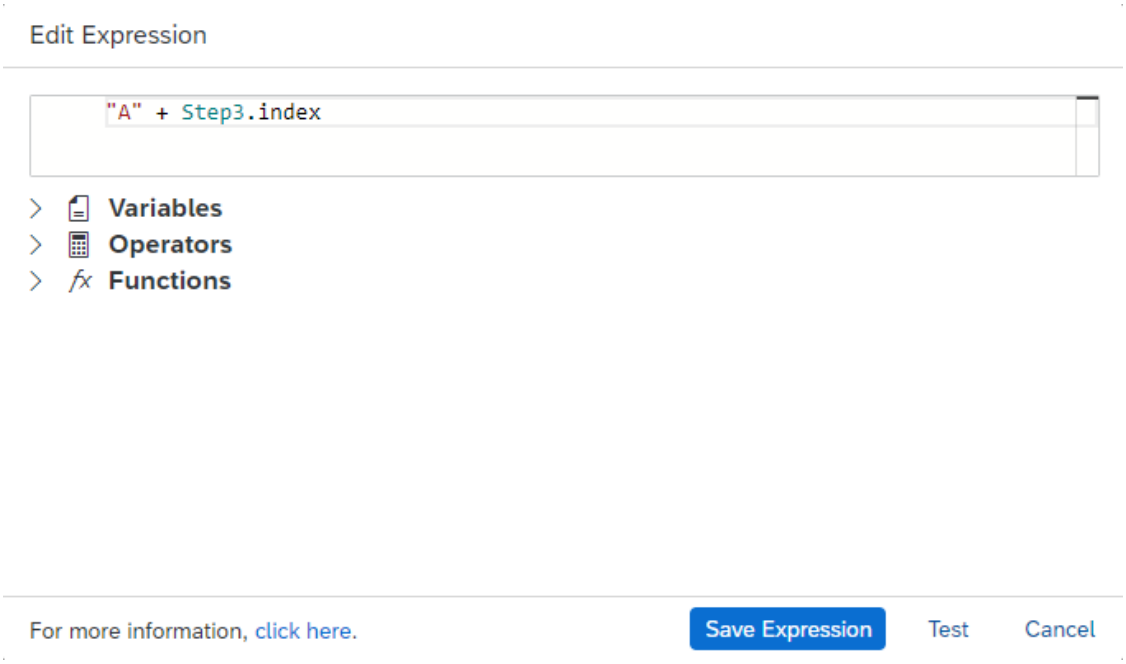
5. Select *Repeat*. On the side panel, enter the starting and ending values in the parameters.



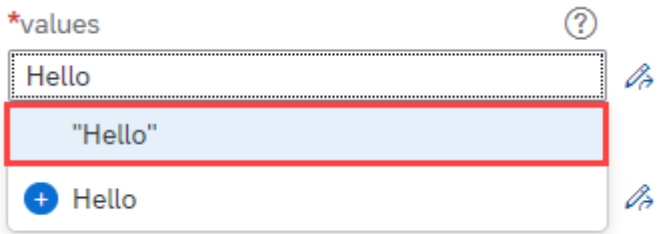
6. Drag and drop the activity *Set Values* on the repeat branch.



7. Select *Set Values*. In the input parameters of the activity:
   1. Click the button to set the **rangeDefinition**.



   2. Enter the value **Hello**.

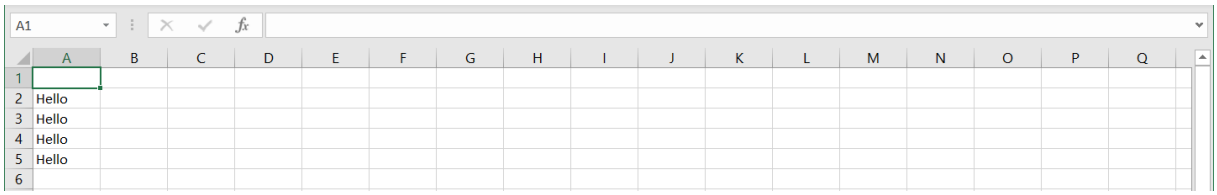


   With these input parameters, the automation sets the value **Hello** in a new cell with each iteration of the loop.

8. Drag and drop the activity *Save Workbook*.
9. Drag and drop the activity *Close Excel Instance*.
10. Save and test your automation.

Result

The value **Hello** is entered on the Excel worksheet in cells A2 to A5.



4.2.6.1.4 Loop End

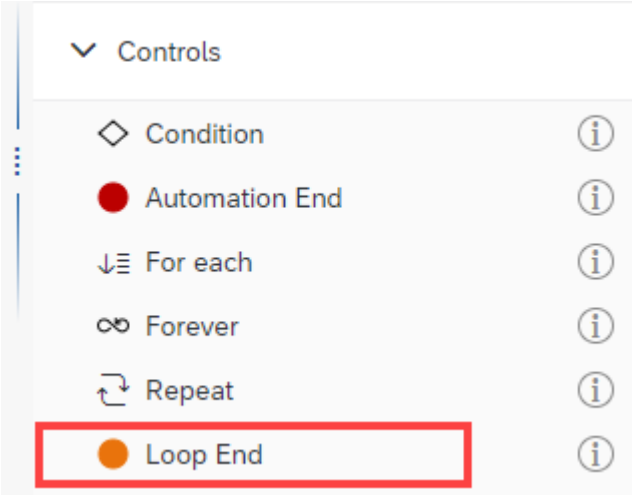
This control allows you to exit a loop before its original end.

You can use the **Loop End** control in the following controls that are set in a loop:

- Condition [page 231]
- For Each [page 217]
- Forever [page 224]

Using the Loop End Control

1. In your automation, on the right-hand side panel, go to the *Controls* section.



2. Drag and drop *Loop End* in a control set in a loop, in the workflow of your automation.

3. Save your automation.

Result

Once the automation enters a loop, and reaches the control, it exits the loop and continues on the main workflow.

4.2.6.2 Condition Controls

Stop Automation in Error [page 336]

The **Stop Automation in Error** control allows you to raise a business error that generates a customized input message in the logs, when your automation is failing.

Condition [page 231]
: The condition control allows a sequence of steps to be executed, only if some conditions are met. If not, the steps are skipped and the execution continues with the following condition steps.

Try Catch [page 239]
: The **Try** control allows you to catch errors raised on specific steps of the workflow and handle them on a side branch of the automation.

Automation End [page 243]
: This control allows you stop the automation before the end of your workflow.

4.2.6.2.1 Condition

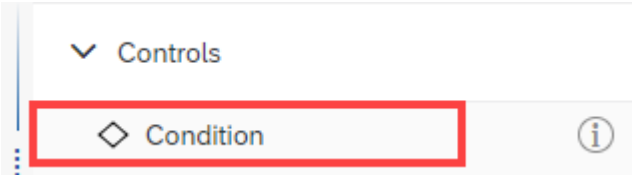
The condition control allows a sequence of steps to be executed, only if some conditions are met. If not, the steps are skipped and the execution continues with the following condition steps.

Based on the result of an expression, it allows your automation to have multiple outcomes, that determine the following step.

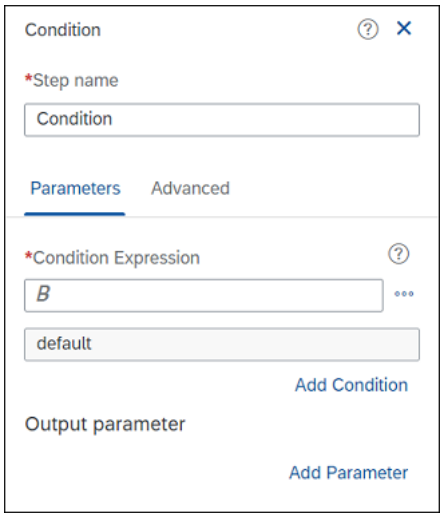
Using the Condition Control

With the control condition, you can add numerous activities to your workflow that will determine the following step, depending on conditions that you set.

1. In your automation, on the right-hand side panel, go to the *Controls* section.



2. Drag and drop **Condition** in the workflow of your automation.
3. Select *Condition*. The *Condition* side panel appears.



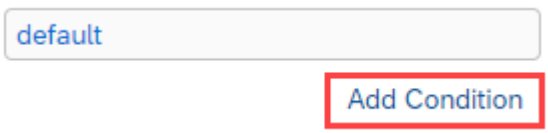
On the side panel you can:

- edit the name of the condition control
- set and add condition expressions

> **i Note**
>
> By default, the condition expression is a boolean variable, but you can change it in the expression editor.
>
> 
>

- add a list of output parameters for the condition

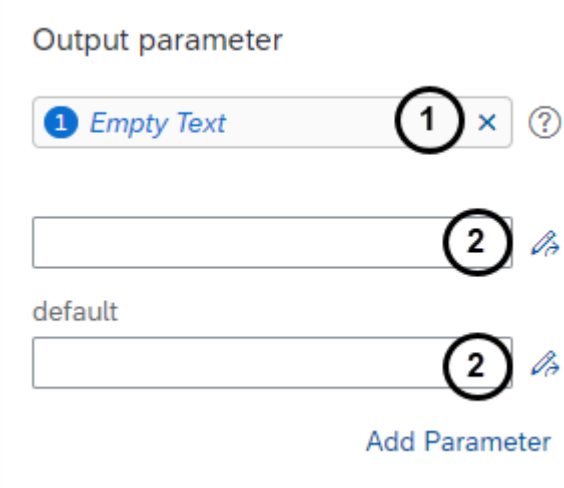
4. To edit the condition expression, click the ••• button on the right of the *Condition Expression* field and click *Edit Formula*.
   The expression editor dialog opens where you can manually type your expression. For more information, see Expression Editor [page 199].
   1. Type an expression.
   2. Click *Test* to check that your expression is valid.
   3. Click *Save Expression*.
5. To add an additonal condition branch, click *Add Condition*.



> **Note**
>
> You can add an infinite number of condition branches, each representing an expression.

> **Note**
>
> The condition control comes with a default condition branch that cannot be edited or removed. Indeed, if none of the conditions are met, the workflow of the automation will continue through the default branch.

6. **Optional**: Click *Add Parameter* to add an output parameter to each condition branch, and set a different value or variable depending on the branch that will be executed.



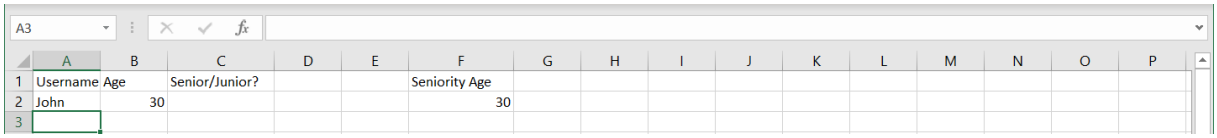
   1. Enter a name. (1)
   2. Set the value or expression for all the condition branches. (2)

      > **Note**
      >
      > You can enter a value, select an existing variable, or edit the output expression in the expression editor.

7. Save and test your automation.

Use Case Example

The following use case determines with the information contained in an Excel file, if the customer is younger or older than the seniority age.

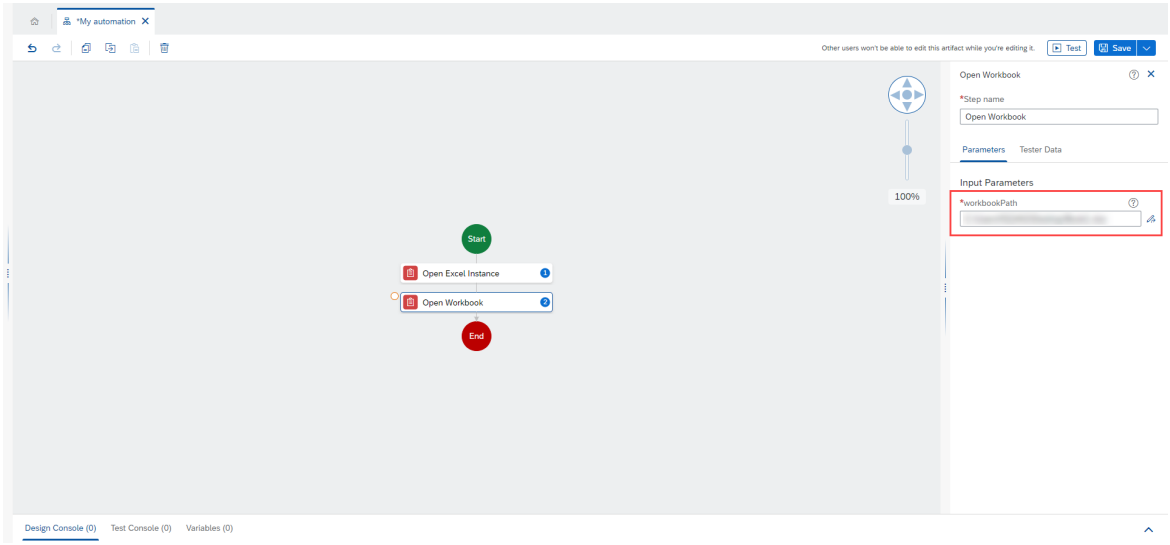


Prerequisite

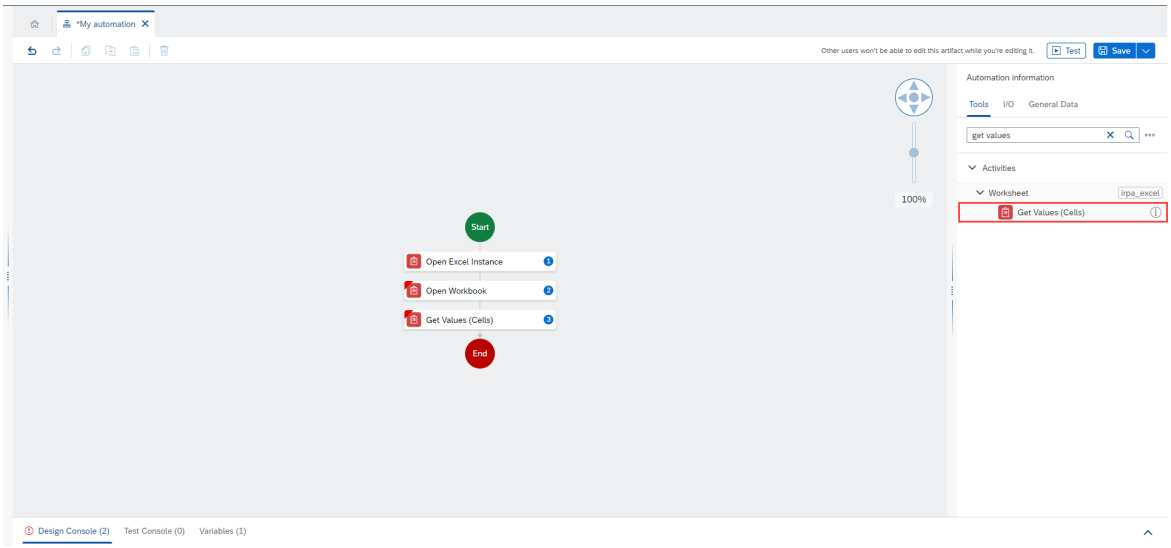
You have created the automation **My automation**.

Procedure

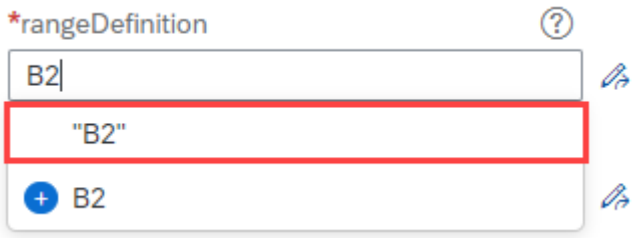
1. In your automation, on the right-hand side panel, got to the *Activities* section.
2. Drag and drop the activity *Open Excel instance*.
3. Drag and drop the activity *Open Workbook*.
   Write the path of your Excel file in the input parameters.



4. Drag and drop the activity *Get Values* in your workflow.

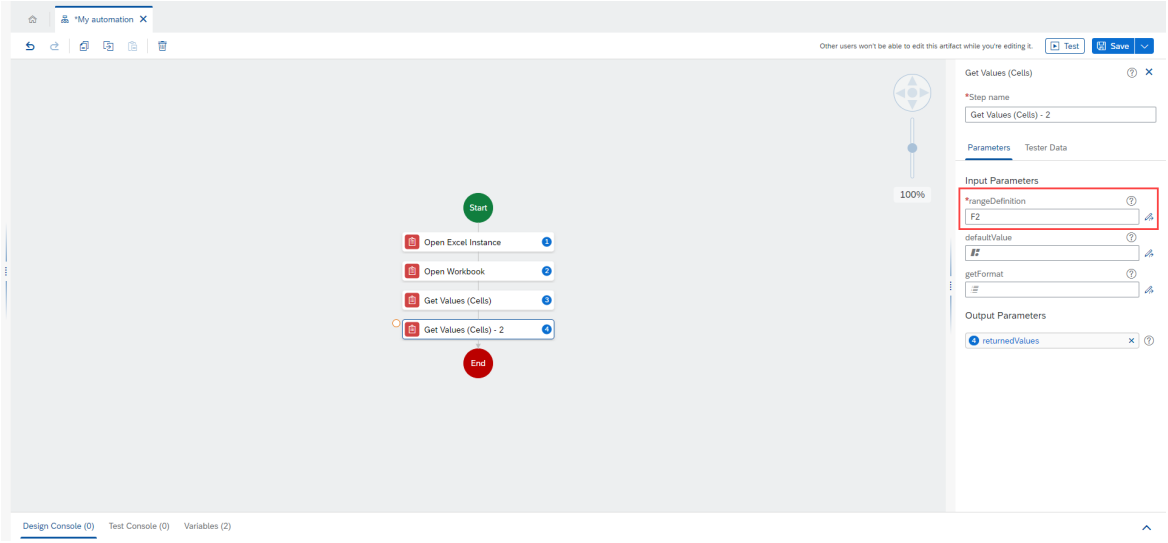


5. Select *Get Values*. In the input parameters of the activity, enter the range of Excel cells in the input parameters, to retrieve the age of the customer.

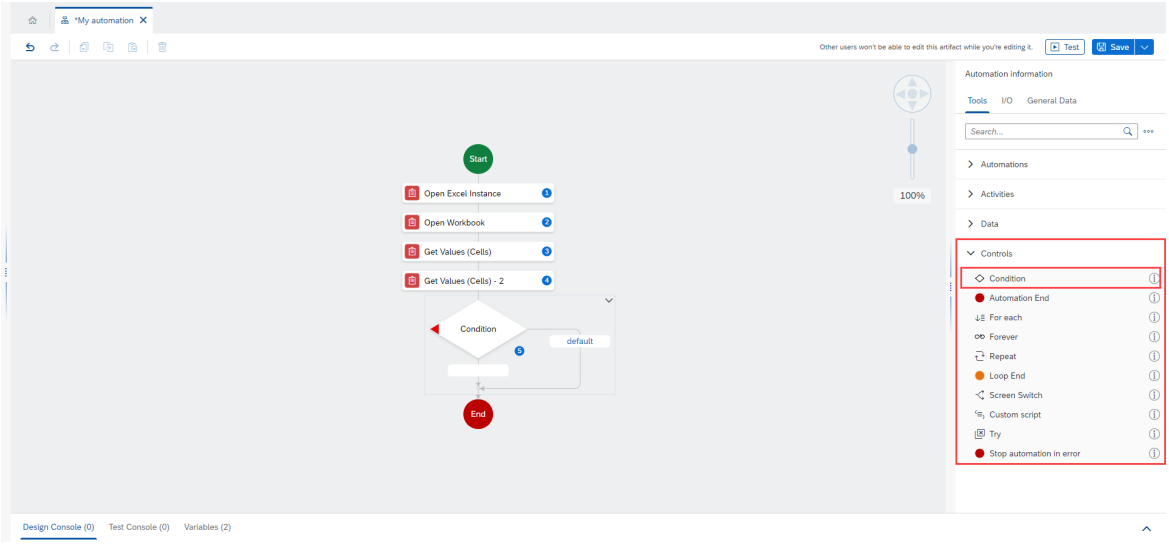


6. Drag and drop a second activity *Get Values* in your workflow.

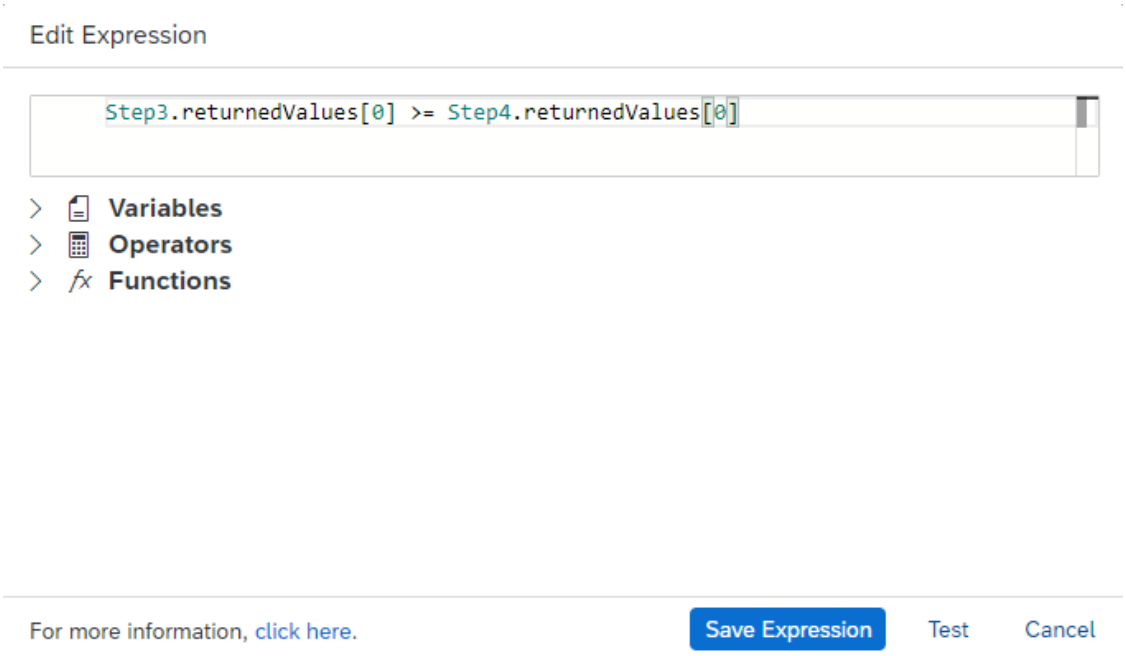
In the input parameters of the activity, enter the range of Excel cells in the input parameters, to retrieve the seniority age.



7. In the *Controls* section, drag and drop *Condition*.

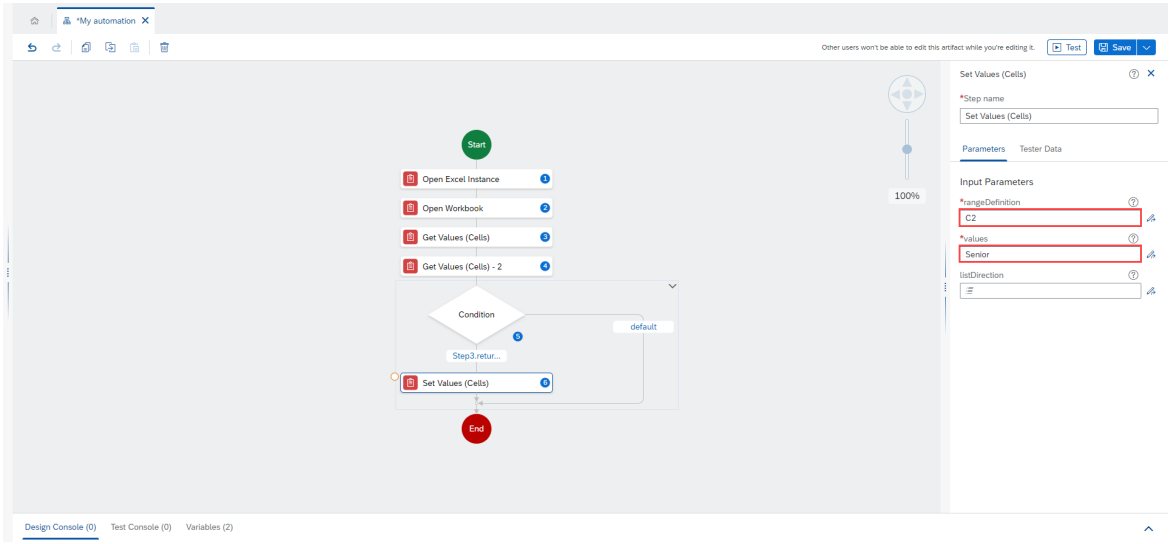


8. Select *Condition*. In the parameters of the control:
    1. Click on the ••• button to set the expression of the first condition branch.

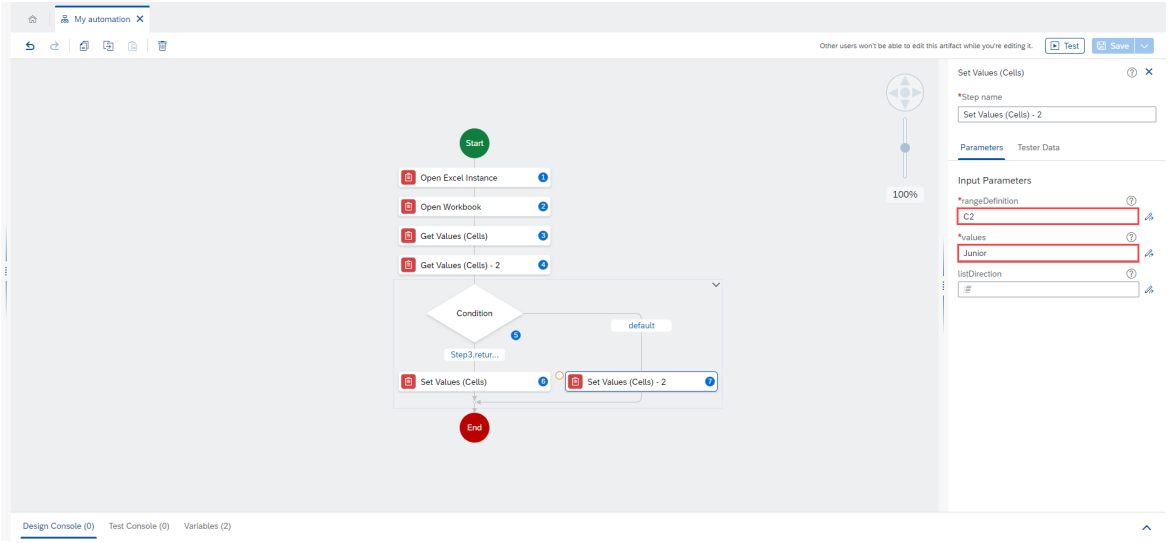


With this first condition expression, if the age of the customer retrieved in *Step3* is higher or equal to the seniority age retrieved in *Step4*, the workflow of the automation continues through the first branch. If not, it continues through the default branch.

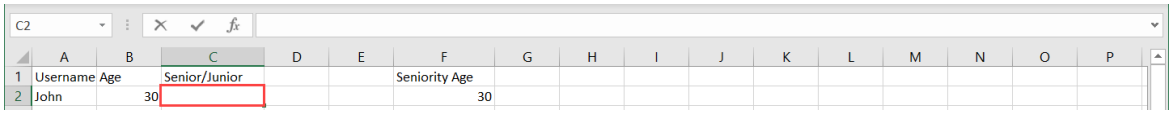
9. Now that the first condition expression is set, drag and drop the activity *Set Values* on the first condition branch and enter the range definition **C2** and the values **Senior** in the input parameters.



10. For the default branch, drag and drop the activity *Set Values* and enter the range definition **C2** and the values **Junior** in the input parameters.

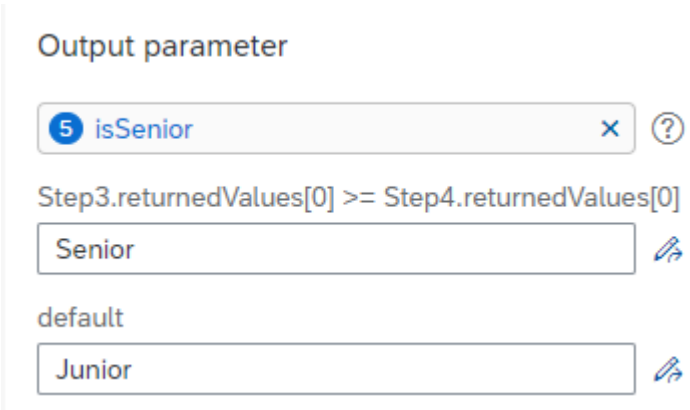


If the age of the customer is higher than the seniority age, the following step sets the value **Senior** on the Excel worksheet. If that condition is not met, the following step sets the value **Junior** on the Excel worksheet.

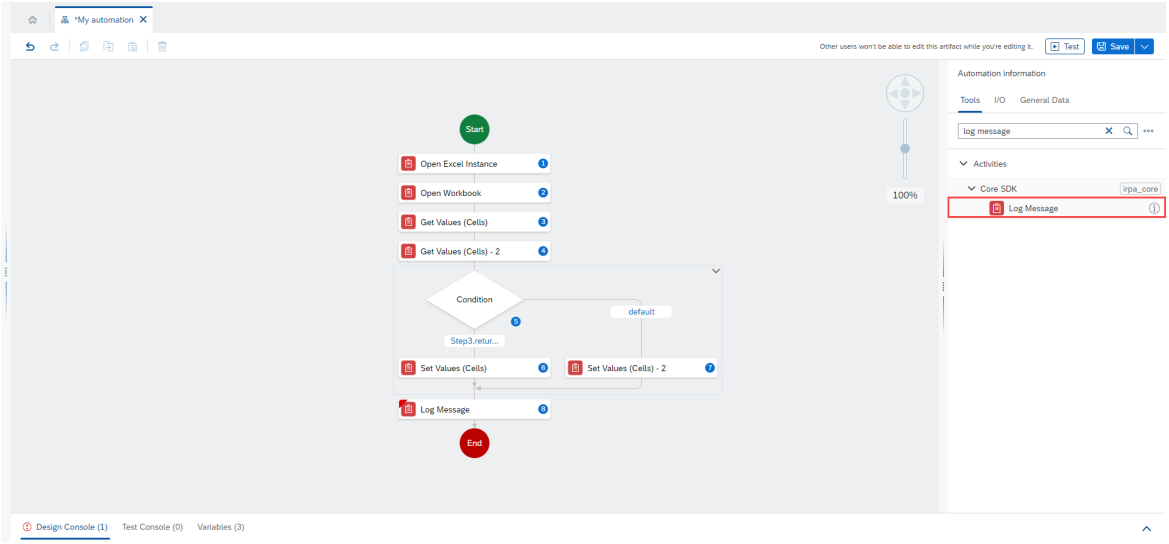


> ### Note
>
> Adding activities in the condition control allows the user to create workflows defined by several conditions.

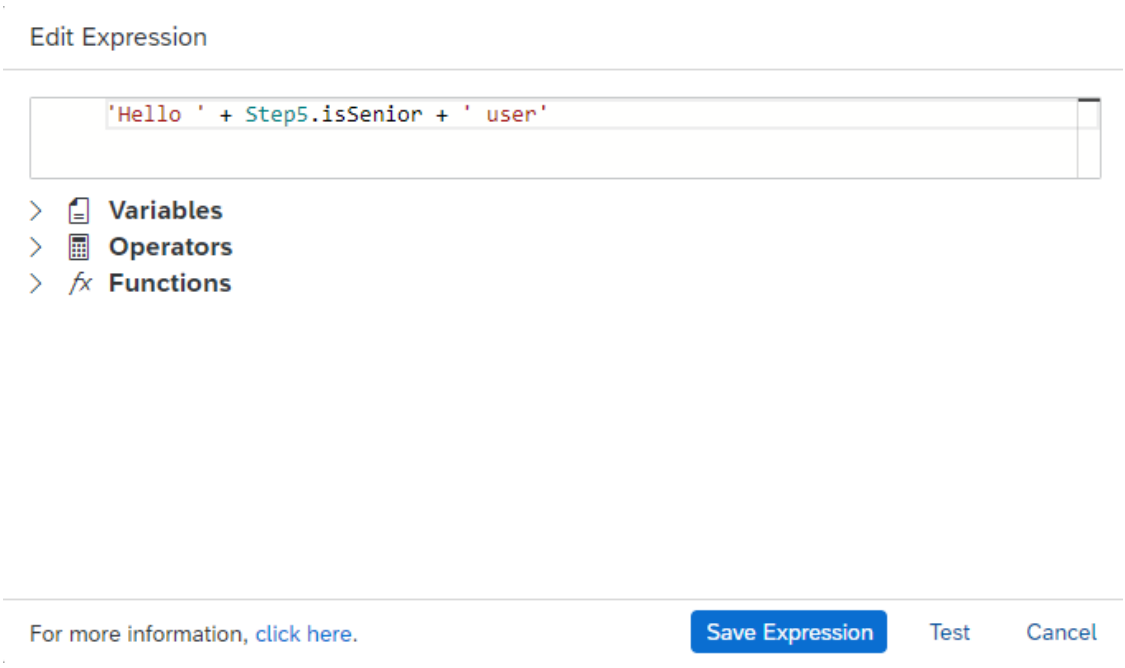
11. Click *Add Parameter* to add the output parameters to each condition branch.



12. Drag and drop the activity *Log*.



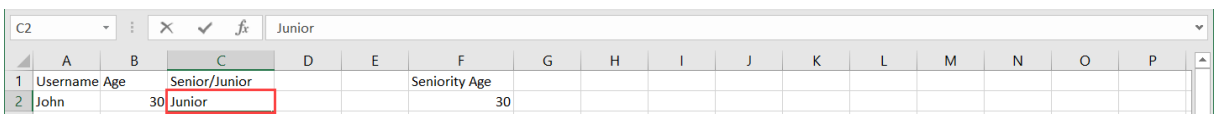
13. Select *Log Message*. In the input parameters of the activity:
    1. Click the button to set the **message**.



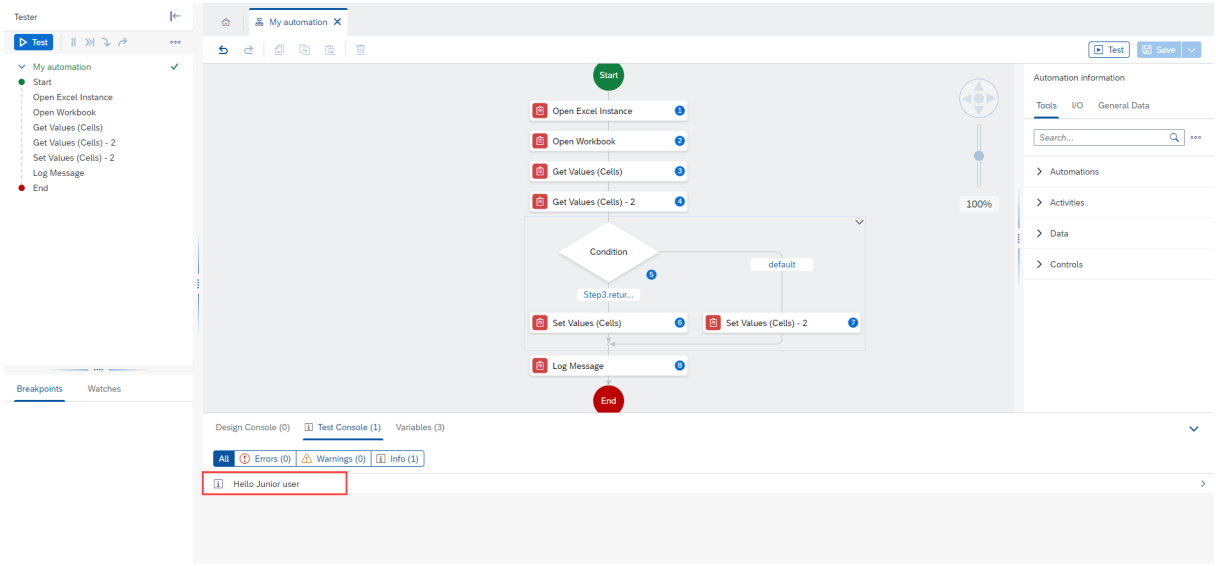
    With this expression, if the first condition is met, the log activity generates the message **Hello Senior user**. If not, the log activity generates the message **Hello Junior user**.
14. Save and test your automation.

Results

The value **Junior** is entered on the Excel worksheet.



The log message **Hello Junior user** is displayed in the debugger.



4.2.6.2.2 Try Catch

The **Try** control allows you to catch errors raised on specific steps of the workflow and handle them on a side branch of the automation.

How to Manage Errors in an Automation

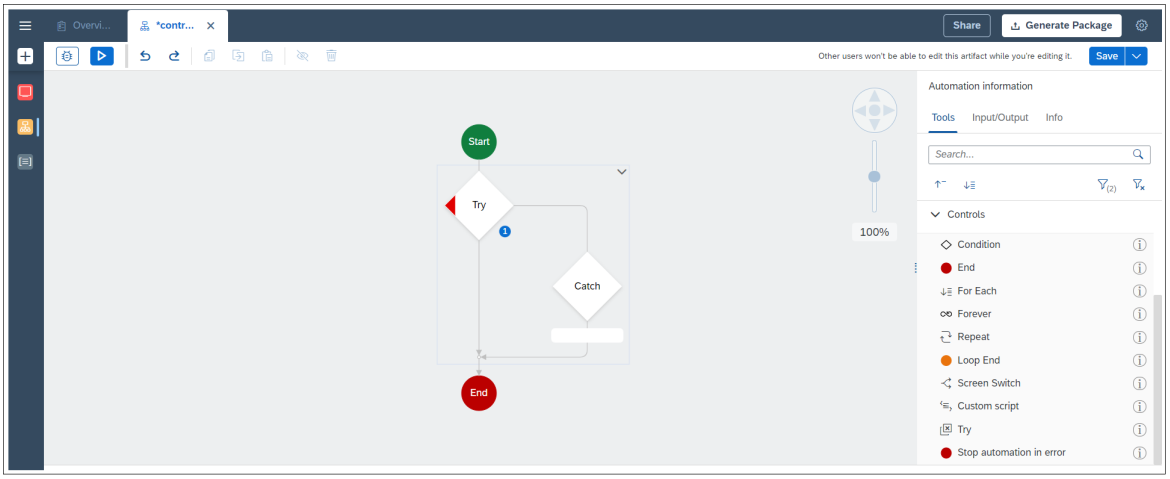
Prerequisites

You have created one or several automations in the Cloud Studio as described in Create an Automation [page 156].

Example

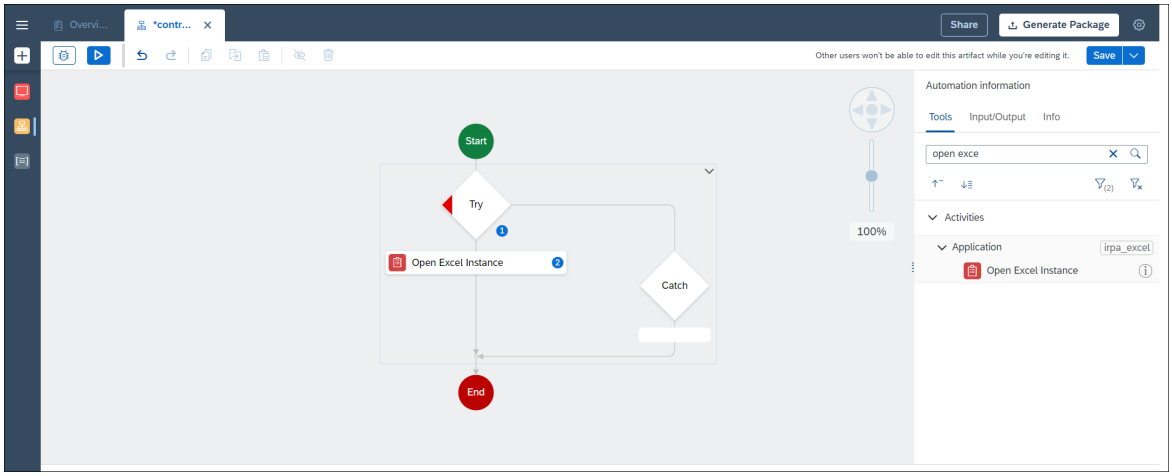
The following use case demonstrates how to manage errors using **Try**/**Catch** pattern in the automation workflow.

1. In your automation, on the right-hand side panel, go to the *Tools* section and click *Controls*.
2. Drag and drop the *Try* control into the workflow of the automation.

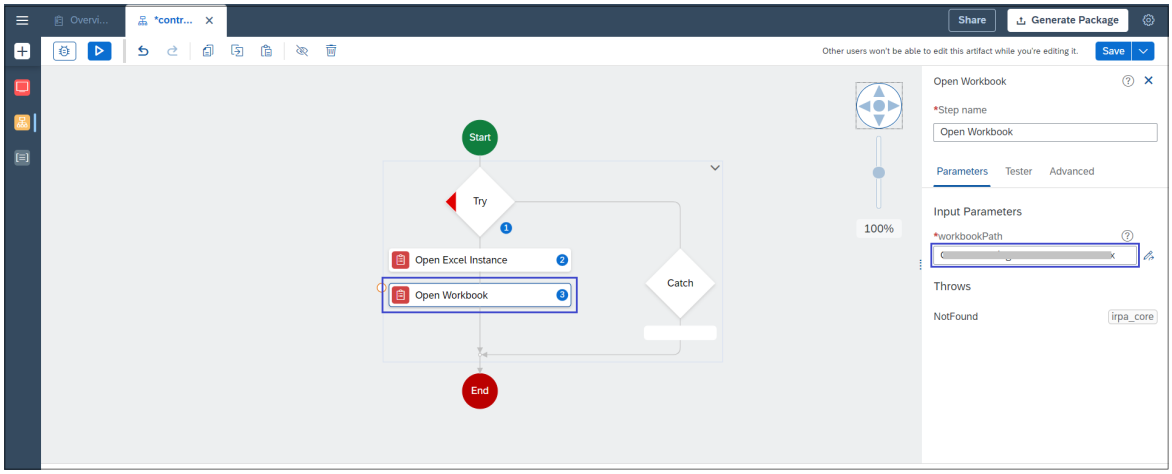


> **Note**
>
> *Try* branch contains all the steps that you try to execute and *Catch* branch contains the steps that you execute if something goes wrong.

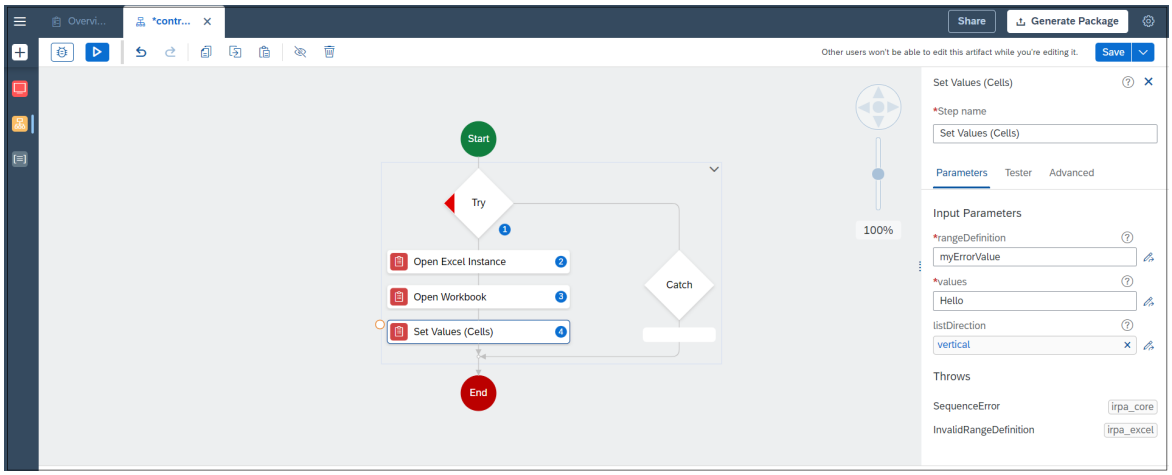
3. Drag and drop *Open Excel Instance* into the workflow of the automation.



4. Drag and drop *Open Workbook* into the workflow of the automation.
5. Select the step *Open Workbook*. The *Step details* panel appears. Set an input parameter in the *workbookPath* field.

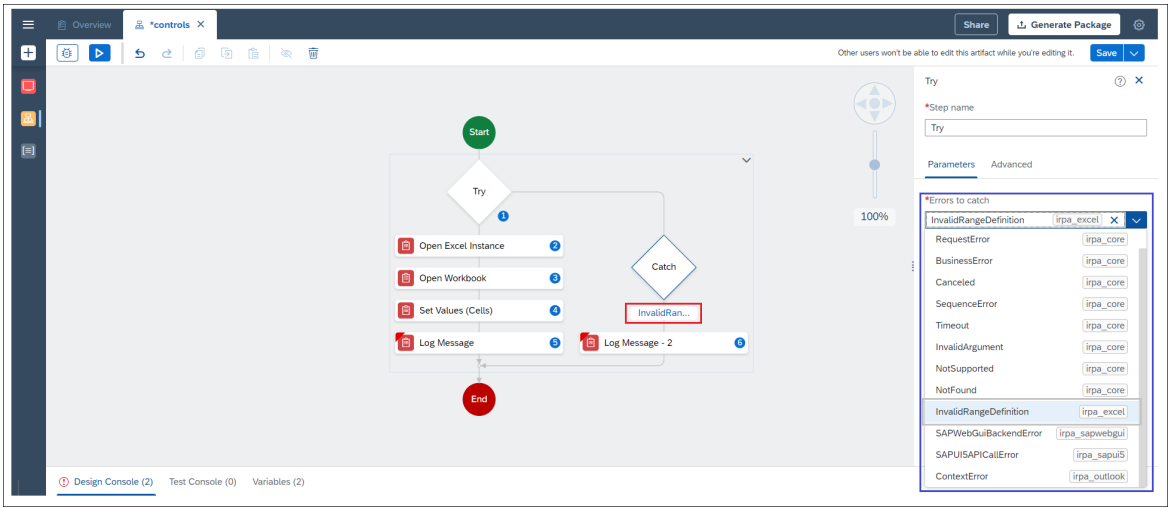


6. Drag and drop *Set Values (Cells)* activity into the workflow of the automation.
7. Select the step *Set Values (Cells)*. The *Step details* panel appears.
   Set an input parameter in the *values* and *rangeDefinition* fields. Enter a wrong value in the *rangeDefinition* field: as a result, the automation will fail and an error will be raised.



8. You can catch the error for entering the wrong value in the *rangeDefinition*. For that, select the *Try* control and choose a specific error from the *Errors to catch* dropdown list.

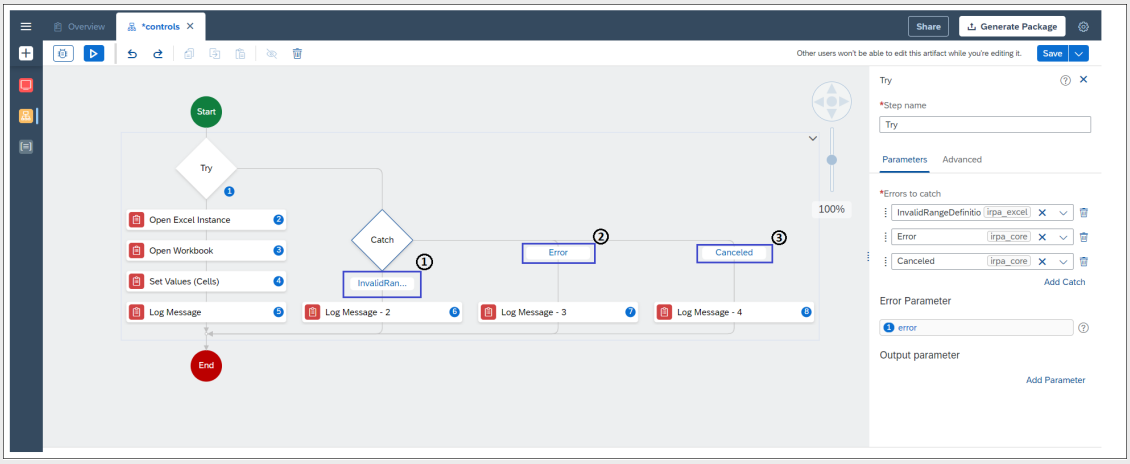
> **i Note**
>
> By selecting *Error* in the *Errors to catch* dropdown list, you can catch all errors in your automation without exception.



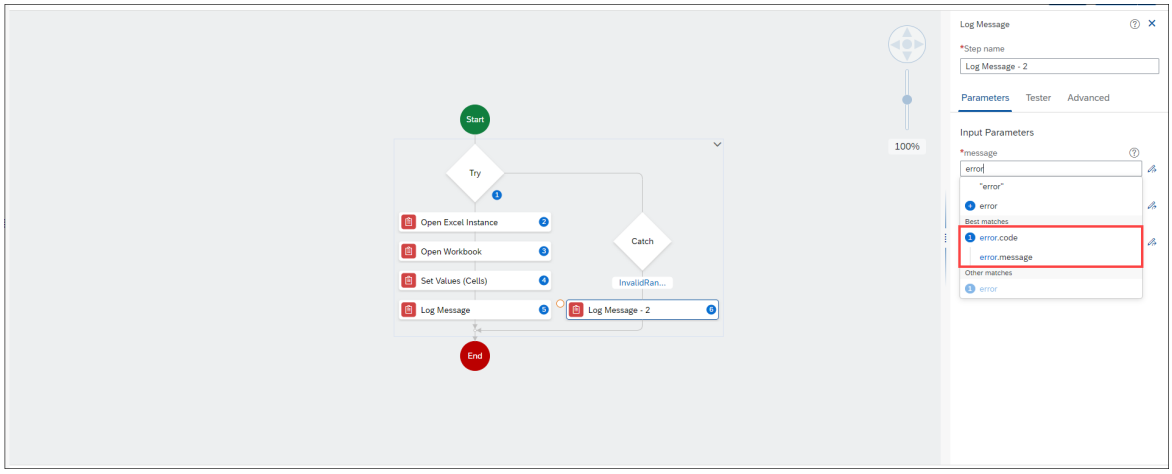
i Note

On the *Catch* branch, you can catch multiple errors with a different workflow for each error.

In the following screenshot, the catch block displays 3 branches with different types of errors: **InvalidArgument** (1), **Error** (2) and **Canceled** (3). At runtime, the bot executes the activities on the *Try* branch and checks in the following order for an **InvalidArgument** error, an **Error** and for a **Canceled** error. If one of the three error types is caught, the bot runs the workflow of the specific error type.



9. You can use the *Log* activity if you want to display the error message at runtime. In the *message* input parameters of the activity, you have access to the error variable's two properties : the `error.code` and the `error.message`. The two properties can be used in the logs and for specific treatments in the automation.



Related Information

Manage Errors within an Automation [page 309]

4.2.6.2.3 Automation End

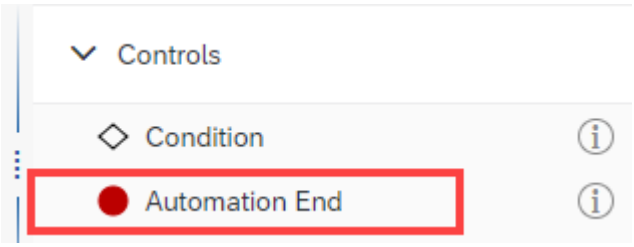
This control allows you stop the automation before the end of your workflow.

You can use the **Automation End** control in the following controls that are set in a loop:

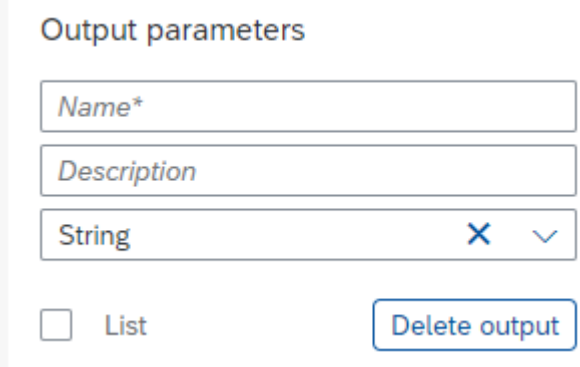
- Condition [page 231]
- Forever [page 224]

Using the Automation End Control

1. In your automation, on the right-hand side panel, go to the *Controls* section.



2. Drag and drop *Automation End* in a loop, in the workflow of your automation.
3. **Optional**: Set an output parameter.
   The output parameter you define for the main automation, is the same as the output parameter for the **Automation End** control. However, you can set different values for both the output parameters.
   1. To add an output parameter for the automation, go to the *I/O* section and click *Add new output parameter*.



2. Enter a name and a description.
3. Click on the dropdown menu to select a type (simple type or data type).

4. Save your automation.

Use Case Example

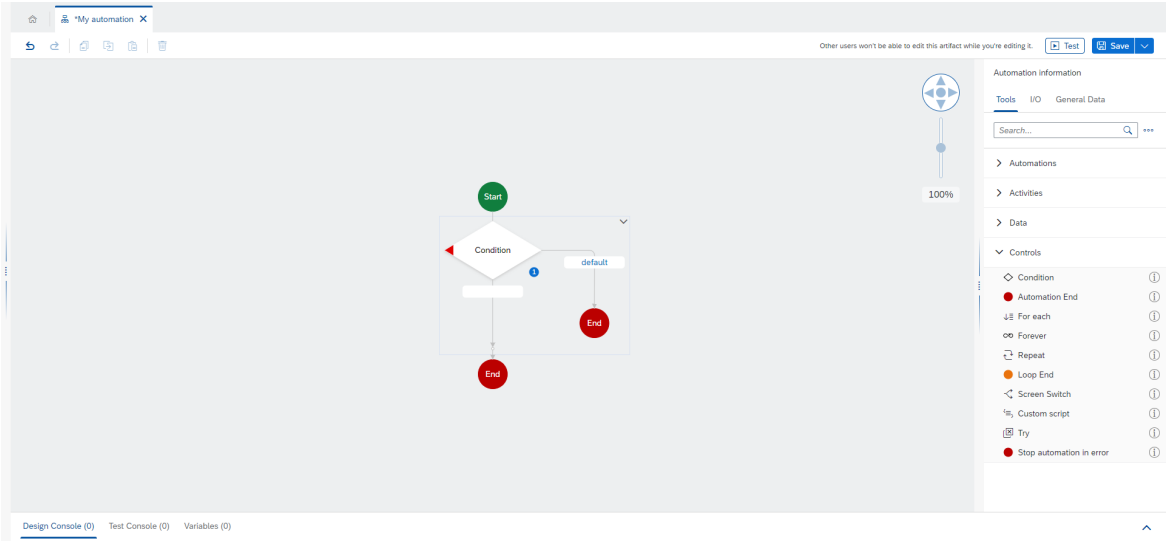
The following use case demonstrates how to set a different output parameter value for the **Automation End** control.

Prerequisite

You have created the automation **My automation**.

Procedure

1. In your automation, on the right-hand side panel, go the the *Controls* section.
2. Drag and drop the *Condition* control in your workflow.
3. Drag and drop the *Automation End* control on the default condition branch.



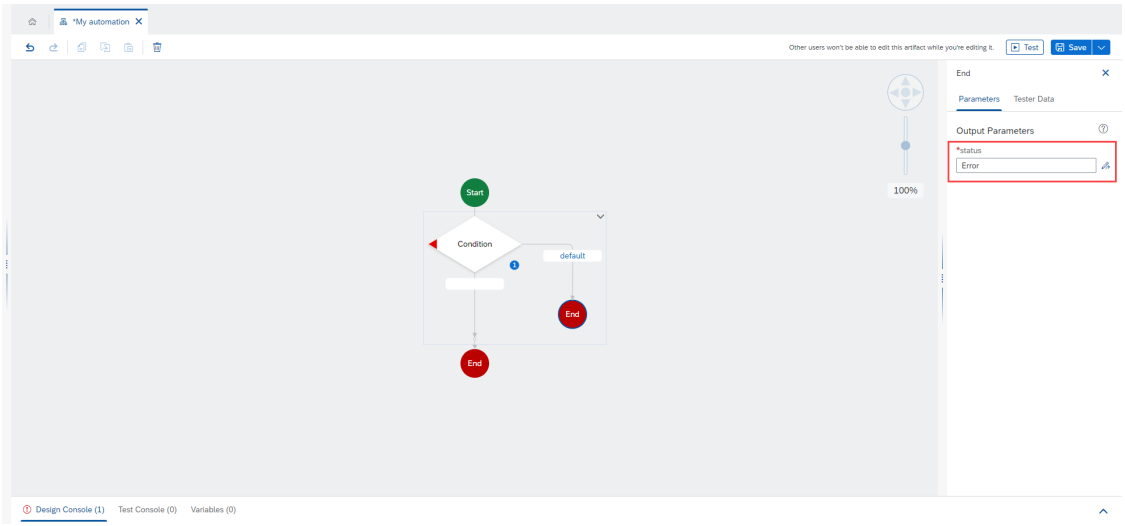
If the workflow of the automation goes through the default branch, the automation will stop instantly.

4. In the *I/O* section of your automation, click *Add new output parameter*.
    1. Enter the name **status**.

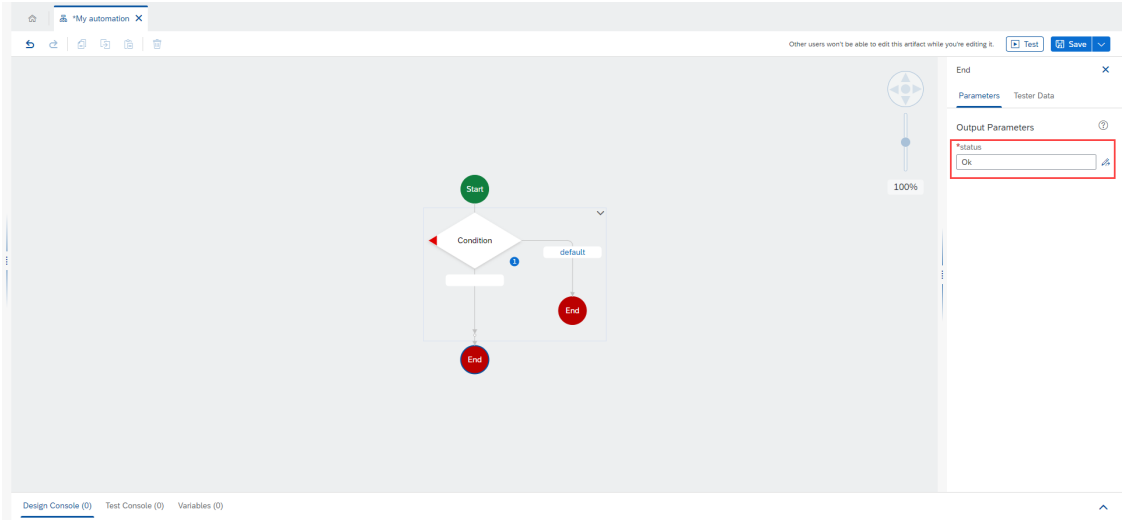
2. Select the type *string*.
3. Save your automation.

As a result, the output parameters of the main automation and the output parameters of the **Automation End** control call for a **status**.

5. Click on the end nodes to set the output parameters values:
    1. For the **Automation End** node, the user sets the value **Error**.



    2. For the main automation, the user sets the value **Ok**.



6. Save your automation.

Result

If the automation goes through the default condition branch, the automation stops instantly with the output parameter **Error**. If not, the automation ends with the output parameter **Ok**.

4.2.6.3 Other Controls

Screen Switch [page 246]
Screen Switch handles multiple situations of a task, defined by which screen we are currently on and determines the next step of an automation.

Custom Scripting [page 251]
In the Cloud Studio, a custom script is an activity that you can define with custom JavaScript code, and insert as a step in the workflow of your automation.

4.2.6.3.1 Screen Switch

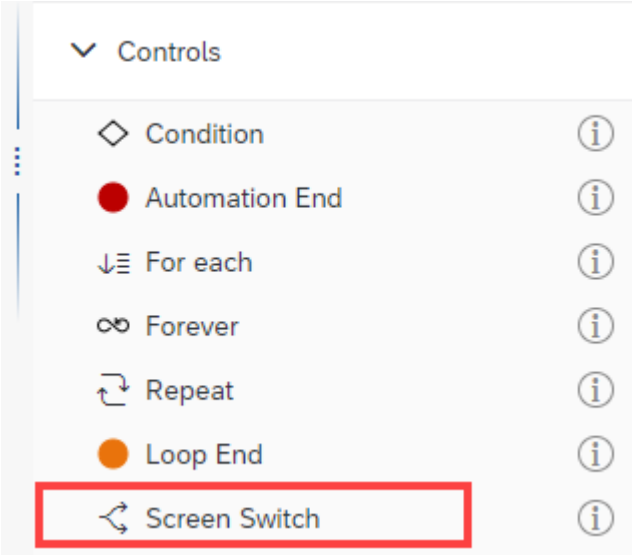
Screen Switch handles multiple situations of a task, defined by which screen we are currently on and determines the next step of an automation.

Based on the provided **Screen Switch Expression** condition, it allows your automation to have outcomes, that determines the following step.

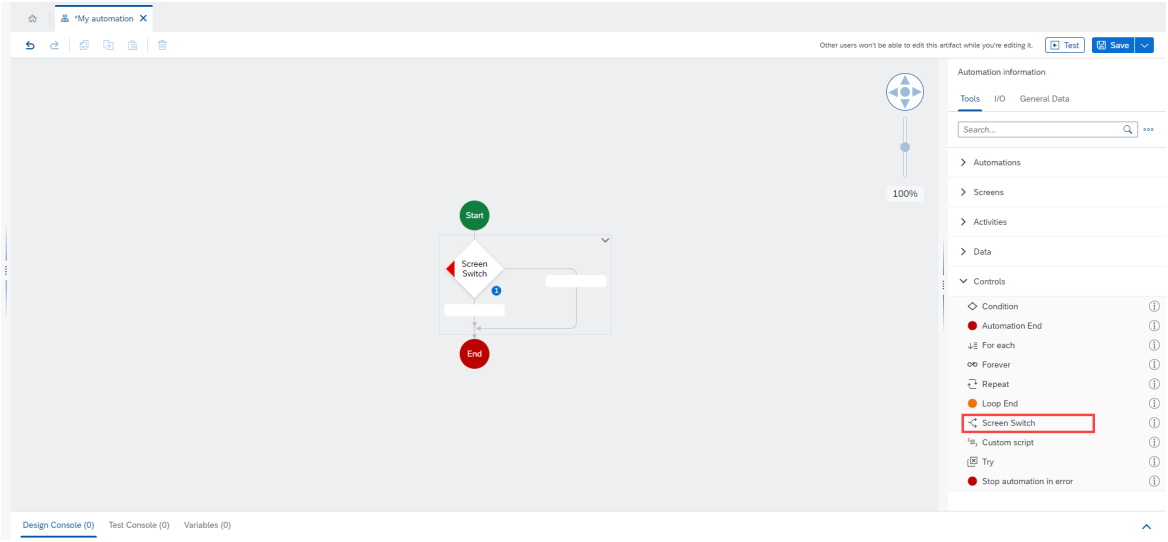
Using the Screen Switch Control

With the **Screen Switch** control, you can add multiple activities to your workflow that will determine the following step, depending on conditions that you set.

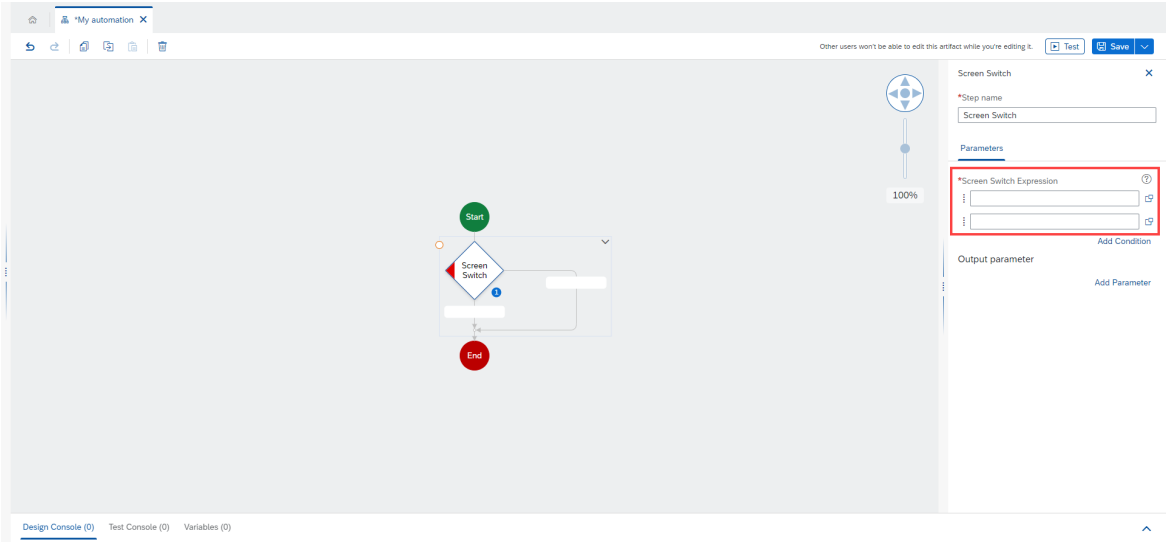
1. In your automation, on the right-hand side panel, go to the *Controls* section.



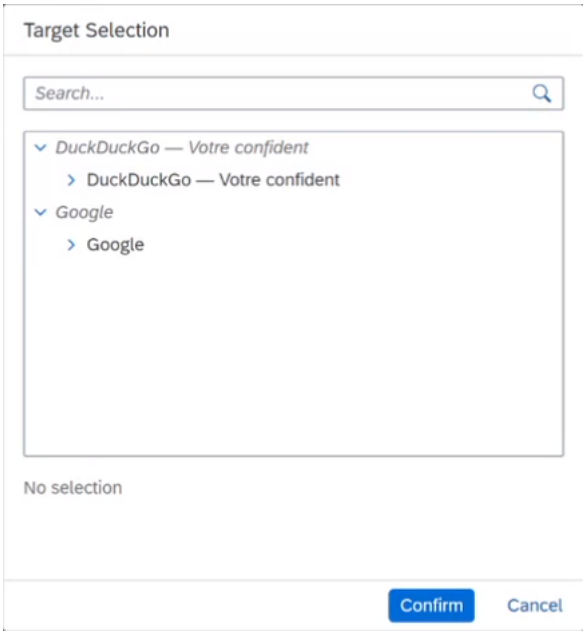
2. Drag and drop *Screen Switch* in the workflow of your automation.



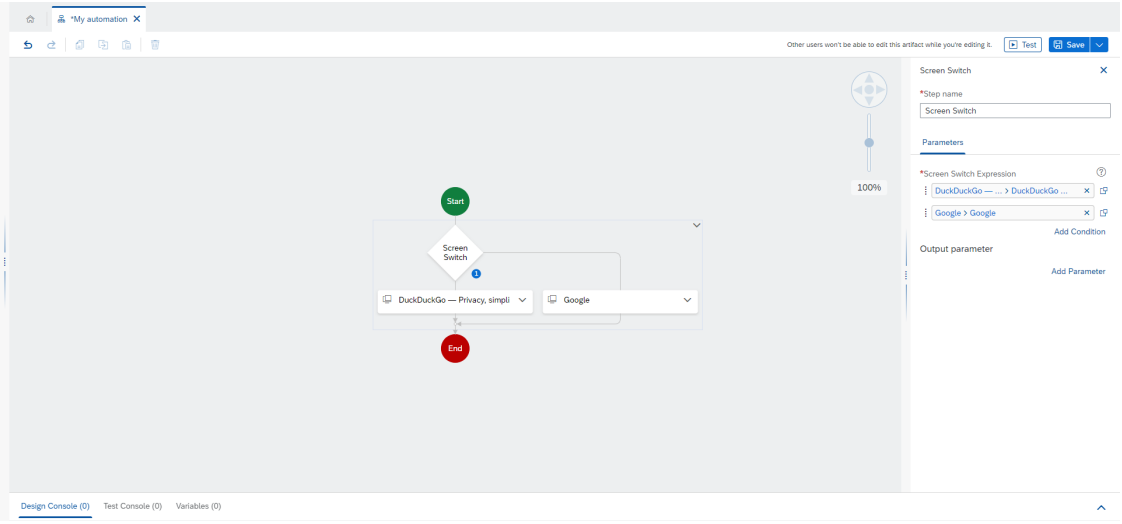
3. Select *Screen Switch*.
   The *Screen Switch Expression* fields are displayed in the side panel where you can provide the target screens detail.



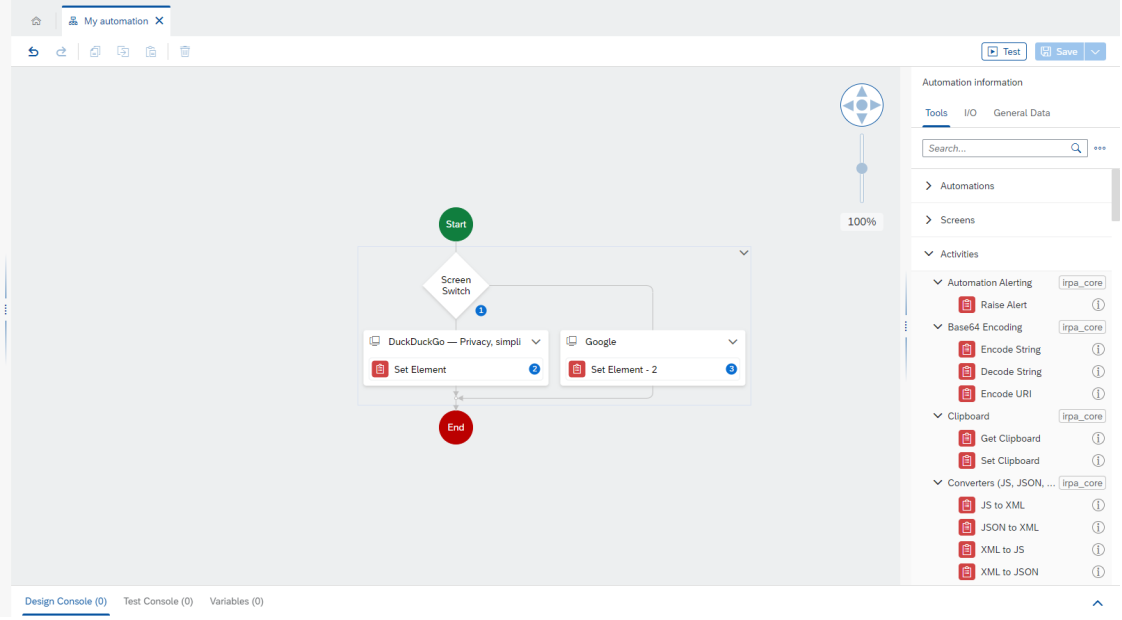
4. Click the icon. The *Target Selection* pop-up is displayed where you can see the list of captured screens. For example,



1. Select the target screen and then click *Confirm*.
   Similarly, you must provide another target screen as the two target screens are mandatory to use the *Screen Switch* control.
   For example,

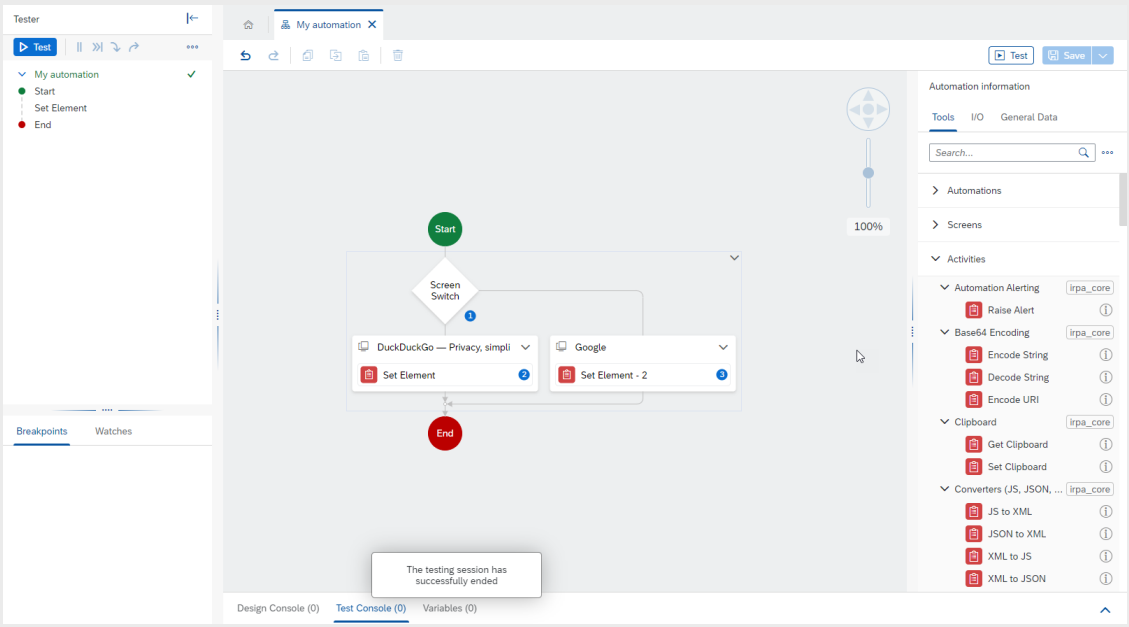


   You can then define the required activities and inputs to determine the next step of an automation.
   In the following example, two screens are defined with *Set Element* activity and its value. As a result, at runtime, the system looks for a screen and executes the step based on the value defined.

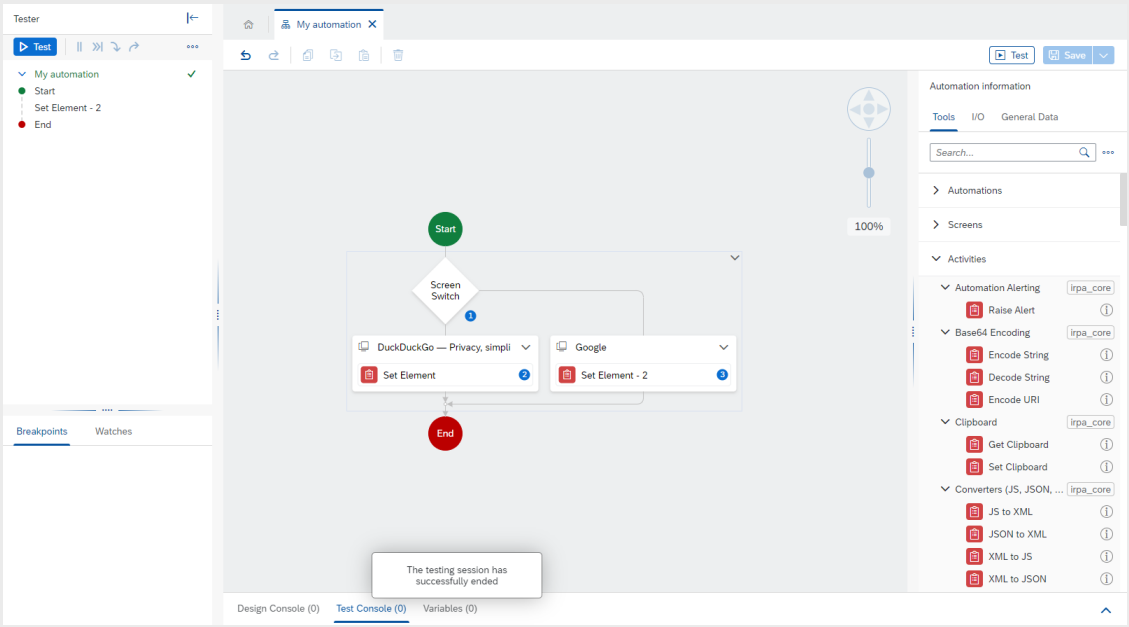


Note

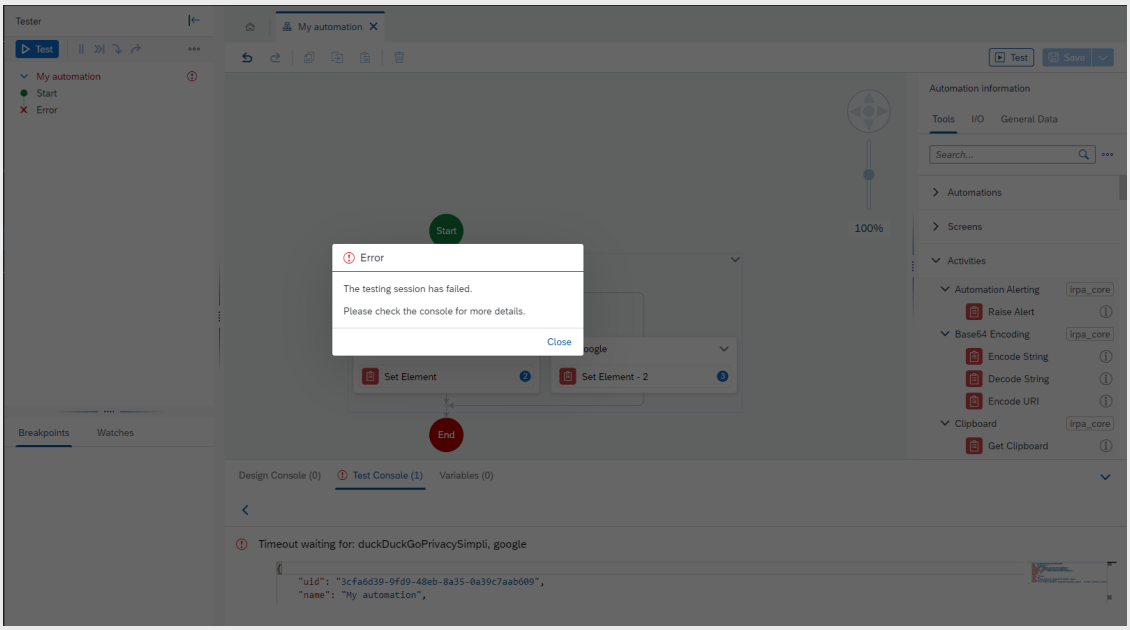
1. Automation is always executed with the screen first detected by the system.
   The following screenshot shows the automation is successful when screen 1 is detected.



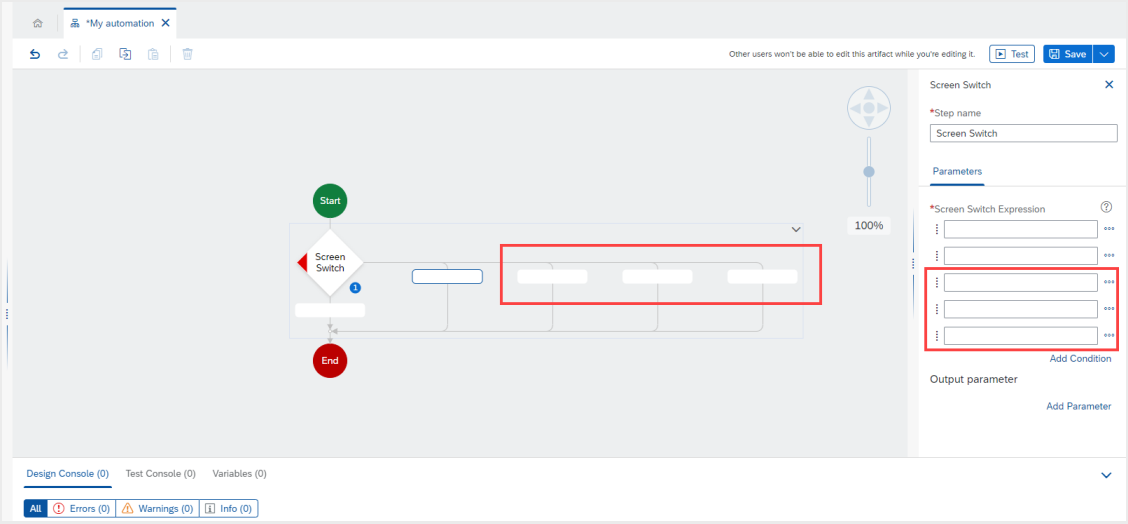
   The following screenshot shows the automation is successful when screen 2 is detected.



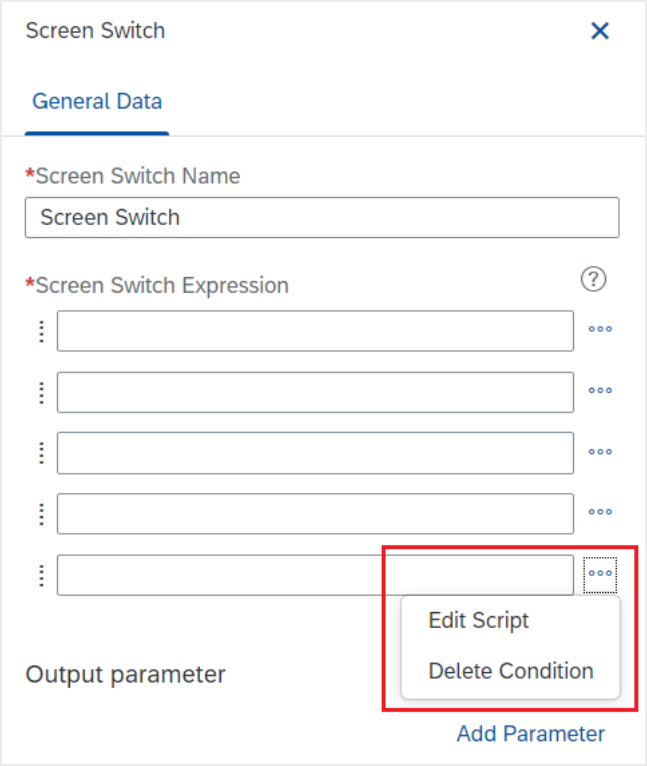
2. At the runtime, an automation waits up to 30 seconds to find one of the specified screens to execute the next step. If the system does not find any screen, the automation fails.
   The following screenshot shows the automation is unsuccessful when one of the specified screens is not found.



3. You can add multiple *Screen Switch Expression* conditions.



4. You can delete *Screen Switch Expression* conditions. However, you cannot delete less than two conditions.



4.2.6.3.2 Custom Scripting

In the Cloud Studio, a custom script is an activity that you can define with custom JavaScript code, and insert as a step in the workflow of your automation.

A custom script supports the ECMAScript (ES6) version that means let, const, and lambda functions.

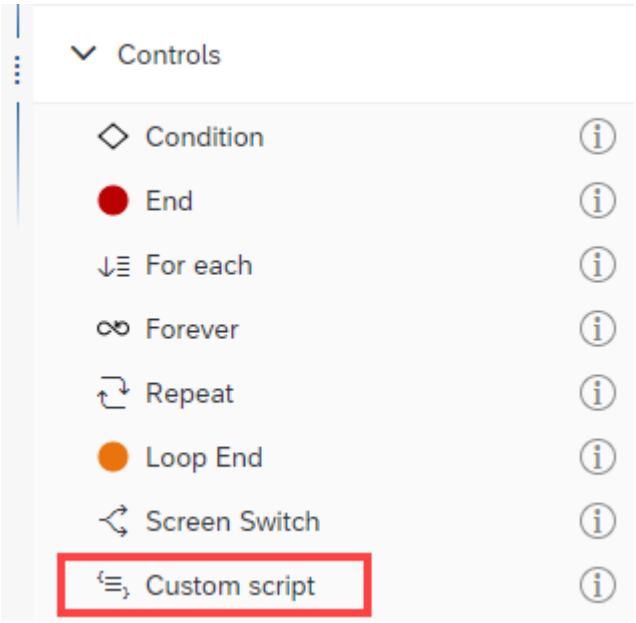
Best Practices

- Only use custom scripting when you cannot use a built-in SDK activity. Always use built-in SDK activities when they exist as these activities are more robust and compatible with further versions.
  Unlike custom script activities, if an SDK activity is planned to be decommissionned, it's flagged as deprecated way in advance. For more information, check the Lifecycle of SDK Activities [page 474].
- Do not call built-in SDK activities in your custom code.

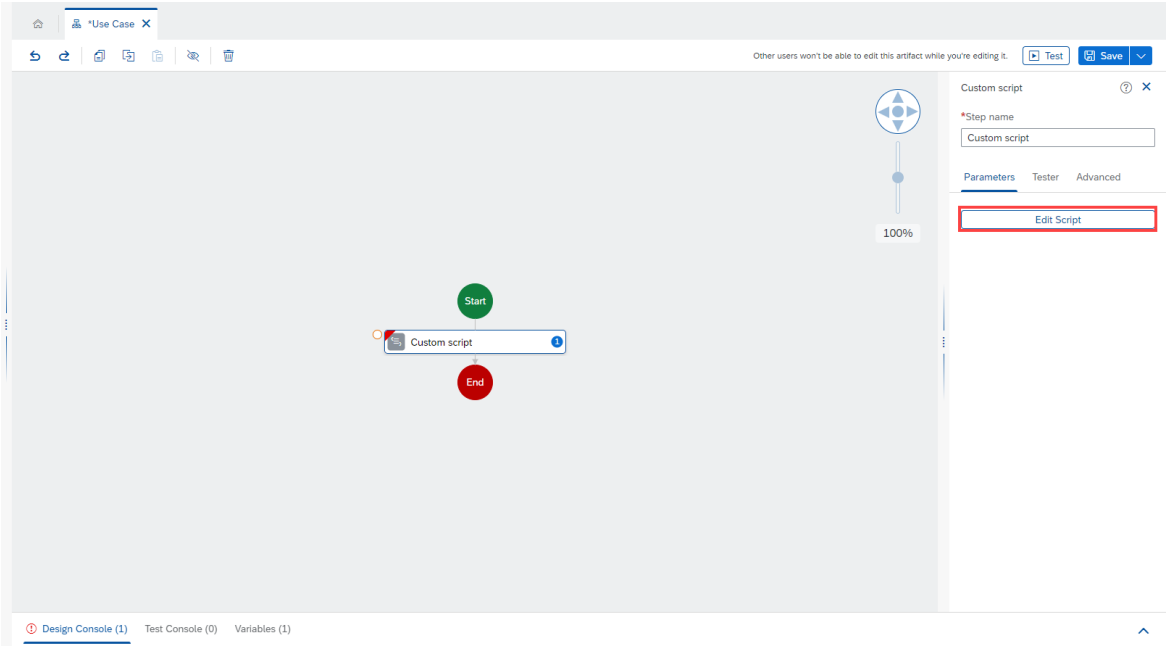
Using the Custom Script

When you write a custom script, you can add input parameters and have output parameters as a result of that script.

1. In your automation, on the right-hand side panel, go to the *Controls* section.



2. Drag and drop **Script** in your workflow.
3. Select *Custom script* and click on *Edit Script*.



A code editor and the Step details side panel appear, where you can set the input and output parameters of your custom script.

4. Click
   - *Add new input parameters*
     Once you define an input, you can use its name as reference for the input parameter.

     Example

     ```
     myInput
     ```
   - *Add new output parameters*
     Once you define an output, it needs to be returned at the end of the script:
     - If you define a single output of type number, this number (e.g. 42) needs to be returned in the end

       Example

       ```
       return 42;
       ```
     - If you define multiple outputs, they must be returned as a map with their names and values

       Example

       ```
       return{myNumberOutput:42, myStringOutput: 'foo'};
       ```
5. Enter a name and a description.
6. Click on the dropdown menu to select a type.
7. Click *Save*.

Note

SDK activities can be used inside a custom script as follows:

```
<sdk_package_alias>.<module_identifier>.<activity_name>(<parameters>);
```

Example

```
irpa_core.core.log('message'); or irpa_core.clipboard.get();
```

Some activities can be accessed to object instances such as applications, screens, or elements.

Example

```
$.myApp.start();
```

Custom Script Editor

In the custom script editor, you can reference the following artifacts present in your project such as:

- Applications
- Application Screens
- Input and output parameters

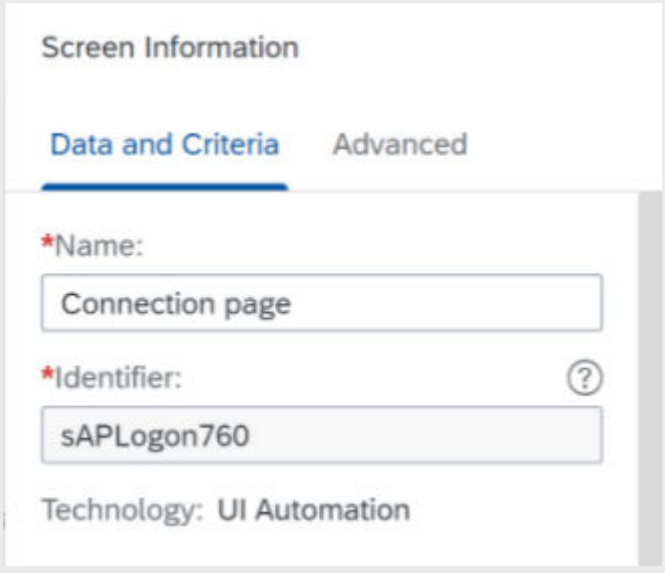
You can reference an artifact within your project like the following:

Example

```
$.myArtifactIdentifier
```

Note

For an artifact, the identifier and the name can be different. Check the artifact information to make sure you are using the right identifier. The following screenshot shows an example of an artifact with a different name and identifier.



You can reference an artifact coming from a dependency like the following:

> **Example**
>
> ```
> dependencyAlias.myArtifactIdentifier
> ```

Intellisense: Code Completion Feature

To help you write your custom script, the editor provides a code completion feature (a.k.a. Intellisense). It shows you what you can access from your script.

> **Note**
>
> When you write a custom script, the editor only detects syntaxical errors in your code before validation. Any other error in your custom script will only be detected during execution time.

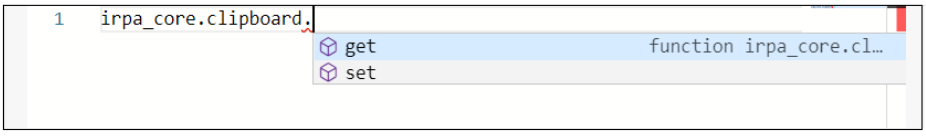
The code completion feature supports two types of artifacts:

- Modules containing
  - *Classes*
  - *Activities*
  - *Enumerations*
- Applications containing
  - *Screens*
  - *Items*

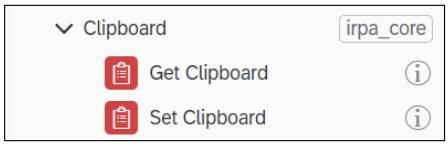
All the information concerning those artifacts is automatically uploaded in Intellisense.

When you start writing your custom script, the activities suggested by the code completion feature match the activities that are listed in the tools section, on the right-hand side panel. The following example shows you the code completion feature for the activities under the subclass **[irpa_core] - Clipboard**.

- In the code editor, write the name of the class followed by a period.



- The three activities that are suggested by the code completion feature, are listed in the activities from the tools section.



To get more information on the artifacts suggested by the code completion feature, you can click on the information button next to the suggestion.



Managing Data and Using Data Types

You can manage data and use data types in two ways:

1. Using SDK to create and set objects
2. Using standard JavaScript code

Data types can be instantiated outside the scripts and passed as script input. For more information, see Data Types [page 168].

You can also create JavaScript objects and pass them as activity parameters.

> **Example**
>
> ```
> const myObject = { name: 'foo', key: 'bar' };
> ```

Accessing Application Elements in Custom Script

To access application elements, the syntax is as follows:

```
$.<application_identifier>.screens.<screen_identifier>.elements.<element_identifier>
```

If the element comes from another package, the syntax is as follows:

```
<package_alias>.<application_identifier>.screens.<screen_identifier>.elements.<element_identifier>
```

Accessing Environment Variables in Custom Script

The preferred way of using environment variables in custom scripts is by defining them as inputs of the script. You must define an input of the proper type for the script and bind an environment variable to it. You can also use the programmatic way; however, the variable type won't be inferred, and there won't be any completion on the variable name.

For a variable defined in the project, the syntax is as follows:

> **Example**
>
> ```
> irpa_core.core.getFactoryVariable('$.myEnvVariable');
> ```

For a variable defined in a dependency, the syntax is as follows:

> **Example**
>
> ```
> irpa_core.core.getFactoryVariable('myDependency.otherEnvVariable');
> ```

Related Information

SDK Packages [page 372]

4.2.6.4 Reuse Variable Defined in Any Block

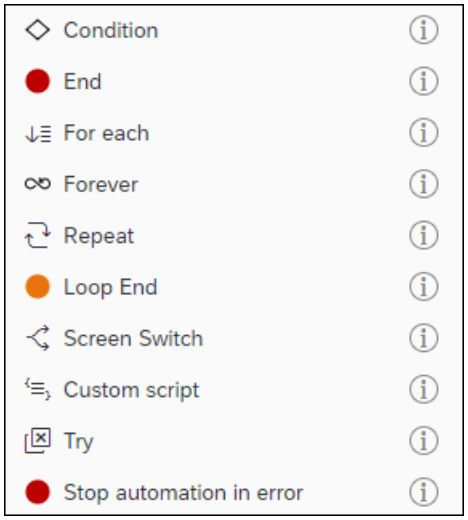
You can build complex automation using lots of different *blocks* and *branches* in the workflow of your automation. The following example of a complex automation displays various blocks and branches used in the automation.



In some cases, several activities inside these blocks produce a result that you can reuse in the rest of your automation.

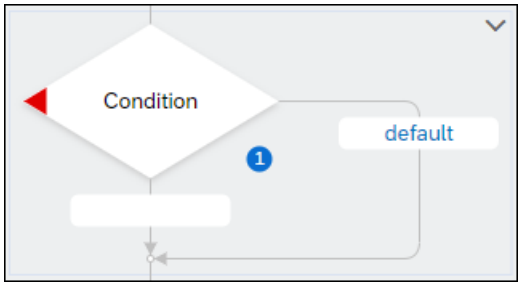
About Block

A *block* is a structure where you can encapsulate activities (except for screens). In Cloud Studio, when you expand the *Controls* menu in the right-side panel, you can see the following list of controls:



In the controls list, each control such as *Condition*, *For each*, *Forever*, *Repeat*, *Screen Switch*, or *Try* is a block.

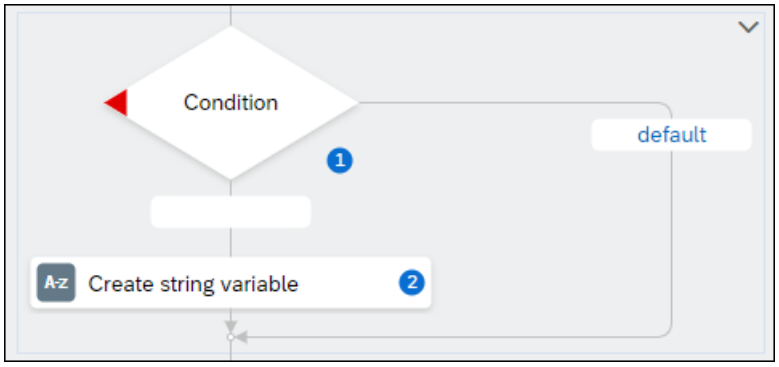
When you drag and drop a control into the workflow of your automation, you can see a light blue outline as a block. The following screenshot is an example of the *Condition* block.



Use Variable Inside a Block

When you use a variable inside any block, the tool will prevent you to use the variable outside the block. The variable is called Scoped Variable.

In the following example, you can see a string variable is used inside the condition block. The variable is only available inside the scope of the condition block, and it has no existence at all outside the block.



Reuse the Scoped Variable

You can reuse the scoped variable as an input parameter of an activity after the block in two ways:

1. Using Output Parameters
2. Using Global Variables

Using Output Parameters

By using output parameters, you can access to variables which are defined inside a block. At the end of each block, you can check the value of the output parameter to determine whether the bot run through the main branch or through another one.

You can define the name of the output parameter and the value of the parameter for each branch of all the blocks that you use in the workflow of your automation.

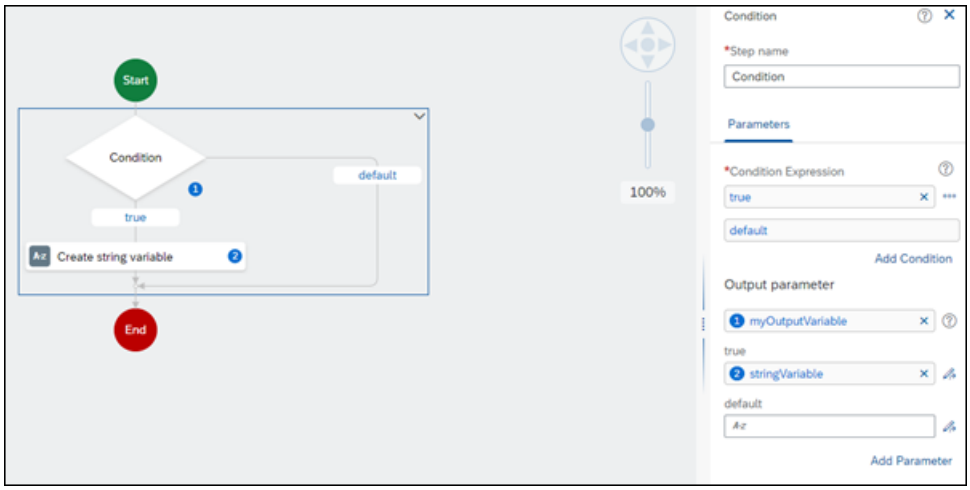
Example 1:

In the following example, a variable is defined inside the *Condition* block.

In the *Condition* block, the value is defined for the two branches:

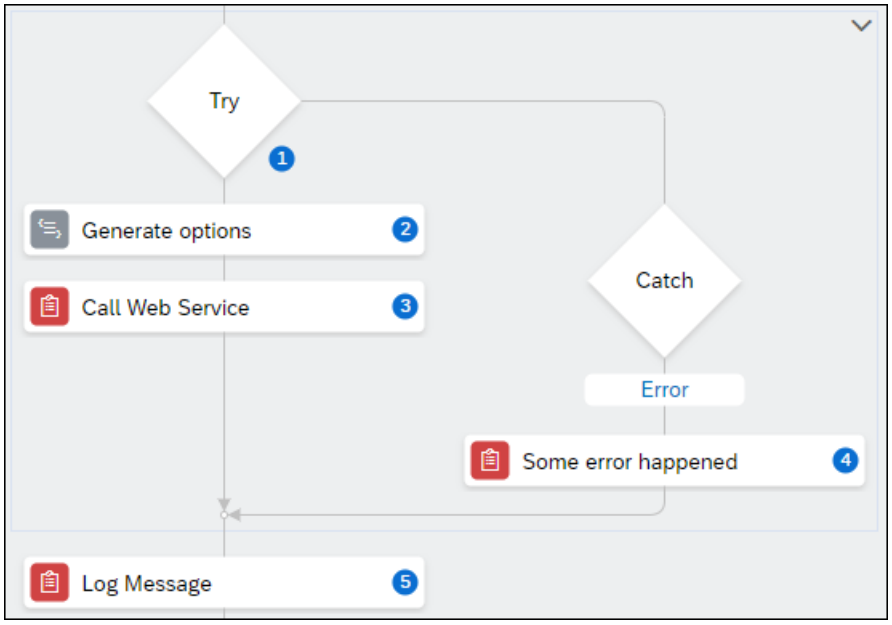
- if the condition is met
- if the condition is NOT met

The output parameter takes the value of the variable *stringVariable* (defined in the branch true of the condition block) if the condition is met during the execution of the automation. If the condition is not met, then no value is assigned to *myOutputVariable*. Therefore, the field is left empty; but it is also possible to define another value to the field.

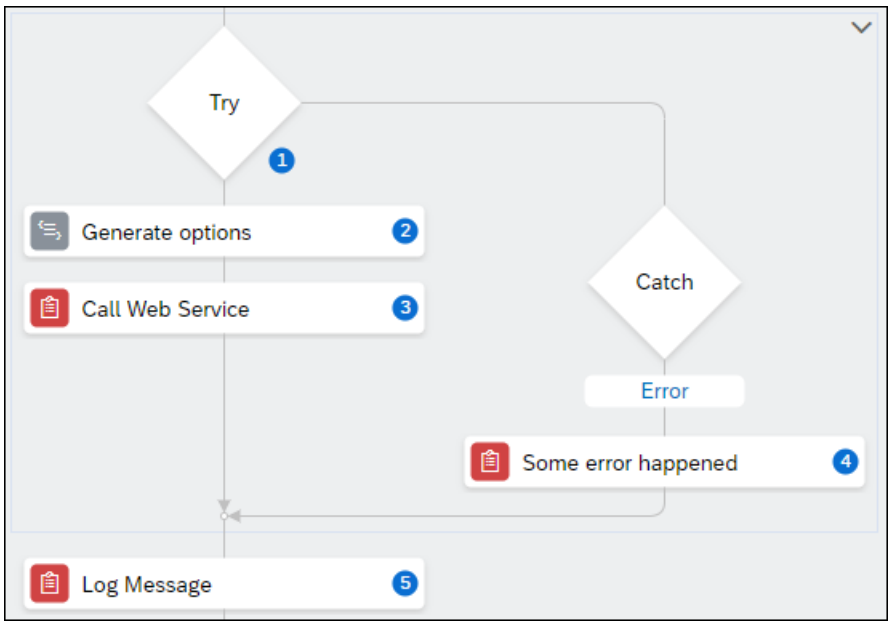


Example 2:

In the following example, a web service is called inside the *Try/Catch* block.

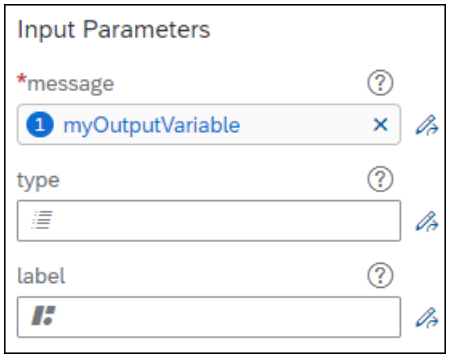


If you want to display the result of the web service call in the *Log Message* activity, you must define an output parameter for the *Try/Catch* block.



If the web service call succeeds, *myOutputVariable* takes the value of *obj*. Otherwise, the *error* object is assigned to it.

In both the cases, you can display the result in the logs as *myOutputVariable* is available outside of the *Try/Catch* block.

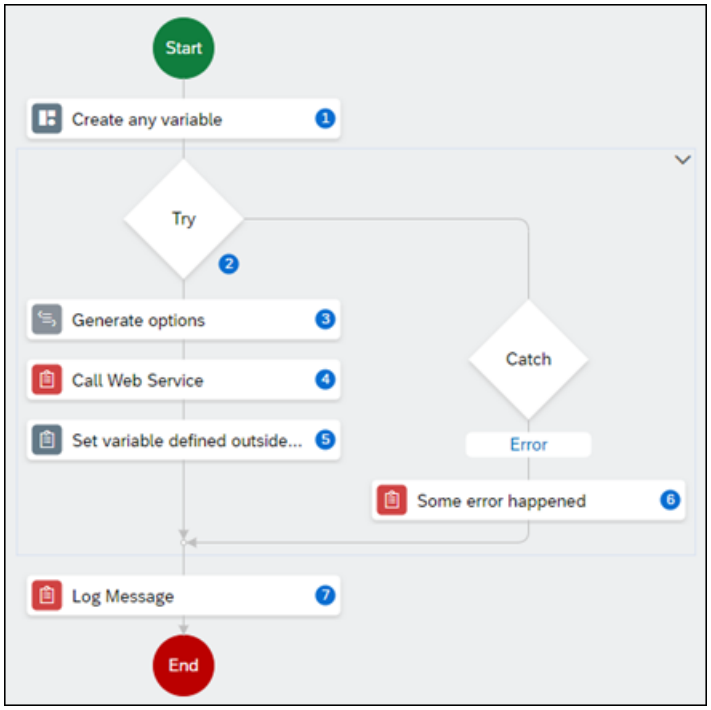


Using Global Variables

By using global variables, you can create a variable outside the scope of a block, and then assign it a value inside the branches of the block.

Example:

In the following example, a variable is defined outside the *Try/Catch* block. You can assign a value to the variable inside the branches of the *Try/Catch* block.



4.2.7 Add a User Task to an Automation

You can add a User Task in an automation where the inputs must be provided from you (involving humans in providing appropriate inputs) at runtime to continue and complete the automation process.

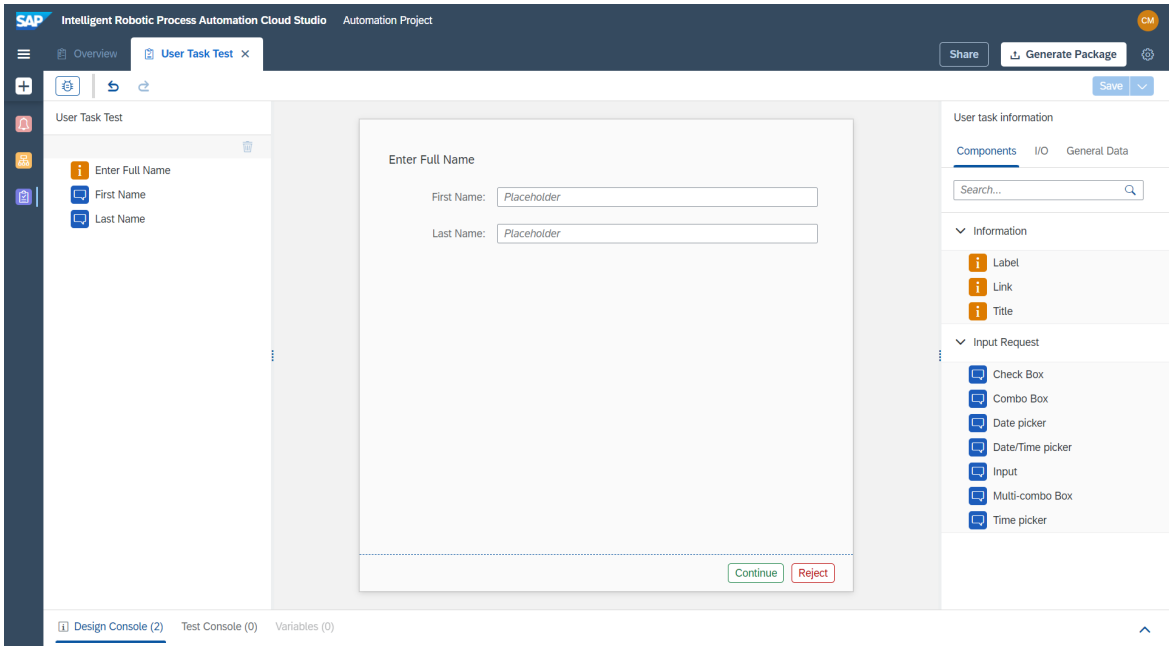
Prerequisites

You have created at least one User Task in the Cloud Studio. For more details about user tasks and defining settings of a user task, refer to the User Tasks [page 489] section.

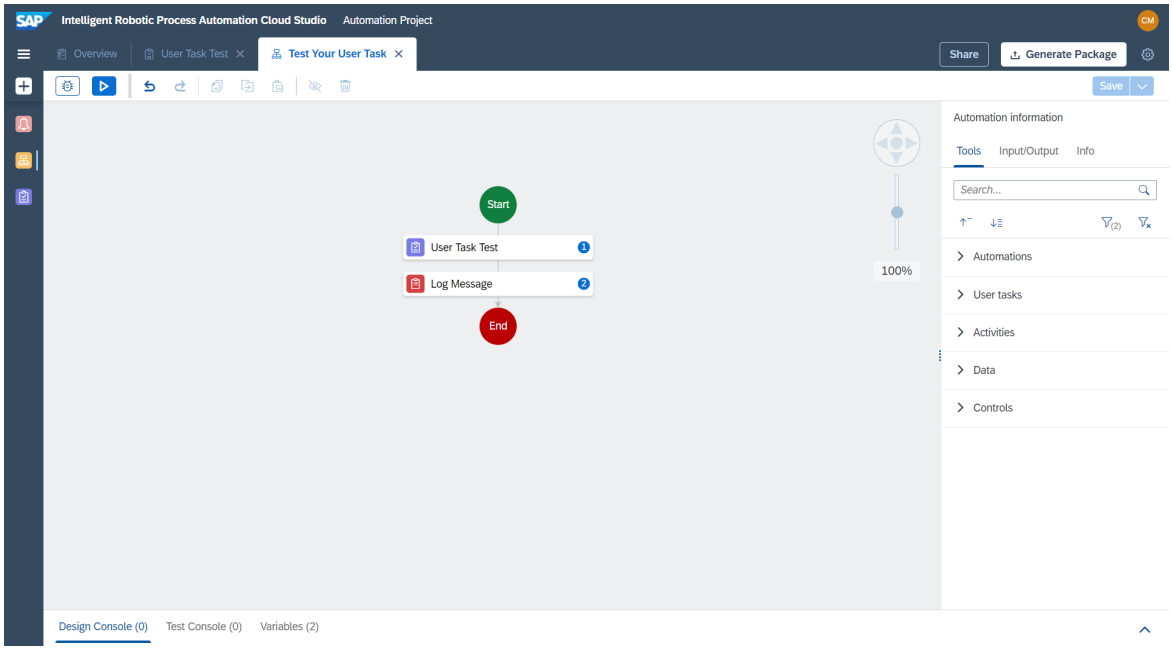
Using a User Task in an Automation

The following use case describes providing user first name and last name as an input to complete the business process.

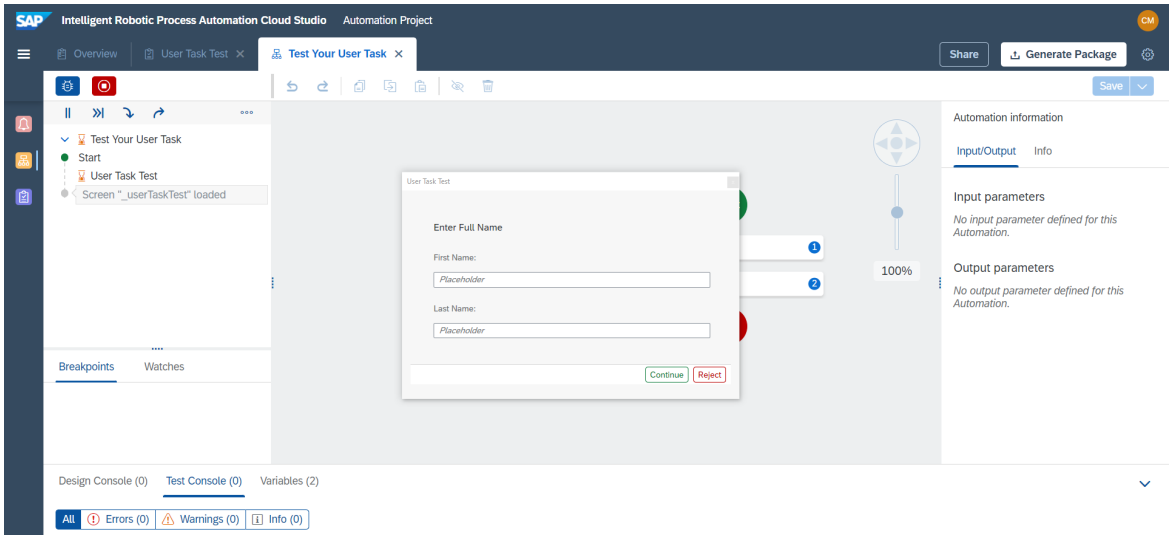
1. Create a user task form to provide the first name and last name of a user in the respective fields. For example, refer to the following screenshot.



2. Create an automation using a user task (User task as mentioned in the step 1). For example, refer to the following screenshot.



3. *Save* the automation.
4. Execute the automation. The Desktop Agent prompts a user task screen for appropriate inputs.



> ### Note
>
> The automation pauses until the maximum job execution timeout period. If providing an input is delayed, the automation fails.

5. Enter the appropriate inputs in the corresponding fields.
6. Click *Continue*. The automation continues to execute further steps of an automation to complete the desired business process.

Related Information

Define the Settings of the User Task [page 491]
Insert Components and Define Parameters to Build a User Task [page 493]
Define Input and Output Parameters [page 505]

4.2.8 Generate a Sub-Automation

You can keep an automation workflow simple by grouping the automation steps to a sub-automation. This helps you to build or manage large automations. You can handle the grouped automation steps in a sub-automation.

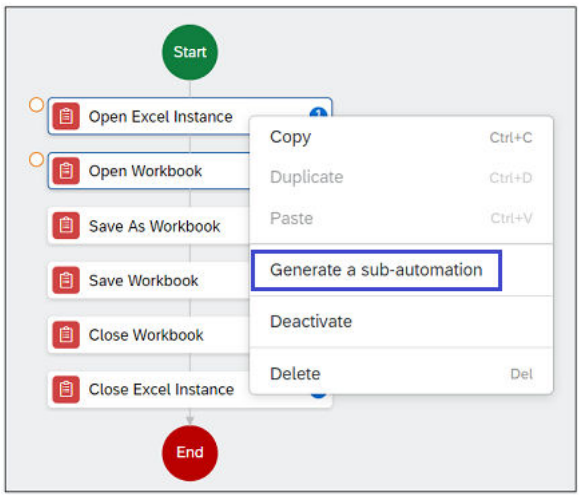
You can reuse the sub-automations in different automations.

> **i** Note
>
> 1. A minimum of two steps are required to create a sub-automation.
> 2. Only continuous steps can be grouped.
> 3. To avoid confusion, do not use a sub-automation within the workflow of the automation from which it is generated.
> 4. Stakeholders of a condition cannot be selected.
> 5. You cannot create a sub-automation if the selected steps contain an end statement. For example, when you select a whole condition and one of the branches contains an end statement, you cannot create a sub-automation.

Using Generate a Sub-Automation

The *Generate a Sub-Automation* option is available in the more options of the selected steps.

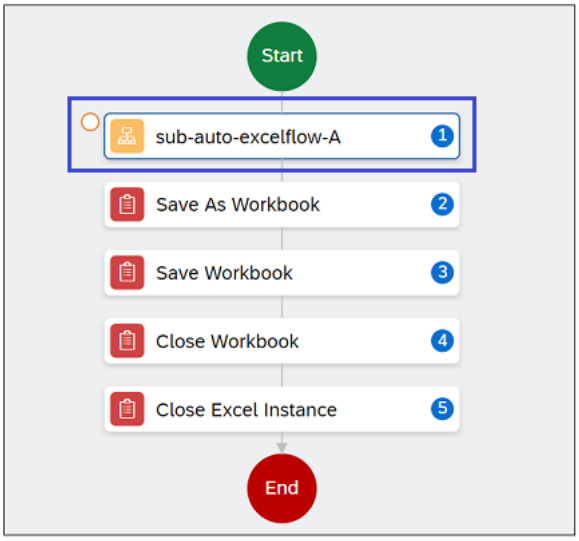
1. In a workflow of your automation, select two or more continuous steps and then right-click. The *Generate a Sub-Automation* option is displayed.



2. Click *Generate a Sub-Automation*. The *Create Automation* pop-up screen is displayed.
3. Enter a name for the sub-automation in the *Name* field.

> **Note**
>
> *Identifier* – If required, you can update the Identifier.
>
> *Description* – Description of a sub-automation. This is not a mandatory field.

4. Click *Create*. The sub-automation is created. It includes the selected steps with all the step details such as input and output parameters.



> **Note**
>
> You can open a sub-automation in a separate automation editor by double-clicking the sub-automation in a currently active workflow.

Related Topics

- Create an Automation [page 156]
- Manage Data within an Automation [page 160]
- Add an Activity to an Automation [page 208]
- Add an Application Screen to an Automation [page 214]
- Add a Control to an Automation [page 216]

4.2.9 Trigger an Automation as a New Job

> **Note**
>
> Automations triggered as a new job are not visible in the test mode. This functionnality is only available for generated packages containing triggers and ready for deployment.

Why Trigger an Automation as a New Job?

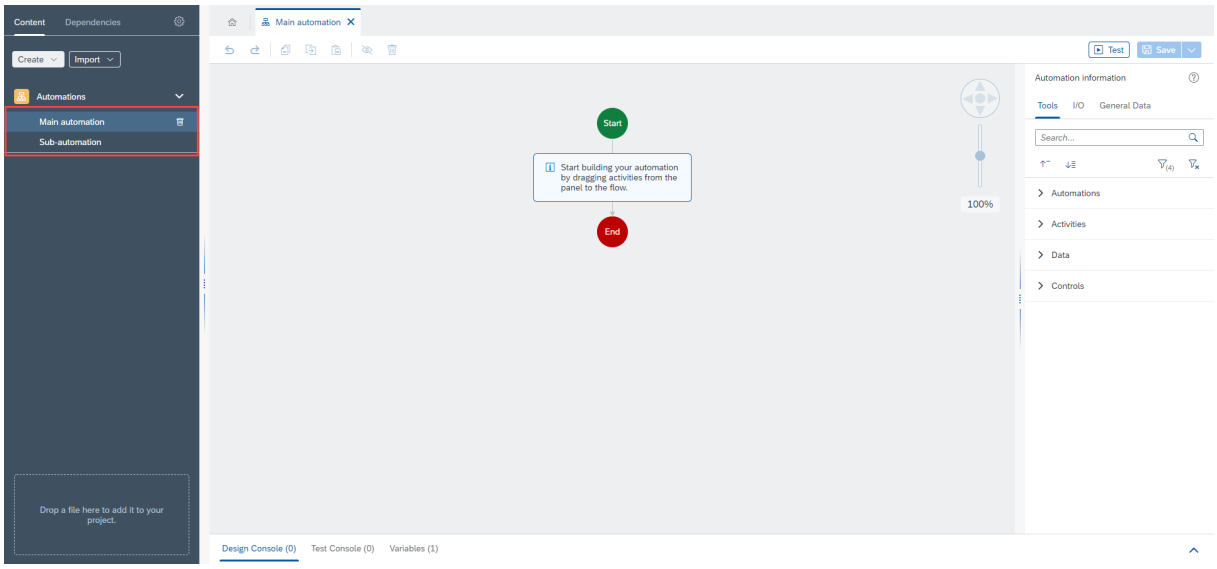
This option allows you to trigger a sub-automation as a new job. Once you have enabled **Trigger as a new job**, a new job is created and the sub-automation execution is done at a different time from the parent automation execution. A sub-automation can also be executed on another agent than the one executing the main automation.

Use Case Example

The following use case demonstrates how to run an automation as a new job.

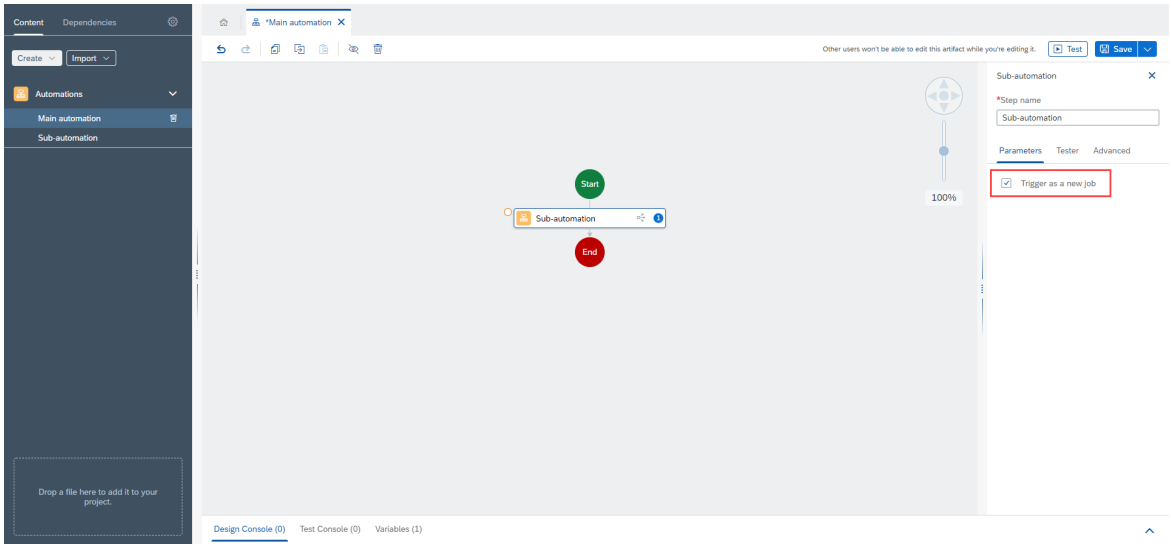
Prerequisite

You have created a project with multiple automations.



Procedure

1. In the *Main automation*, drag and drop the automation *Sub-automation*.
2. Select the step *Sub-automation*.
   The step details side panel appears. In the *Parameters*, you can see the checkbox *Trigger as a new job*.

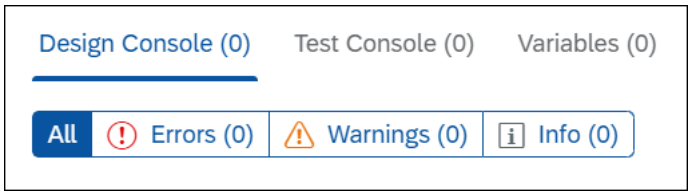


3. Select the checkbox to create a new job.
   During the execution, a job corresponding to your sub-automation will be created and distributed to one of the agent from the environment.

4.2.10 Monitor an Automation Using the Console

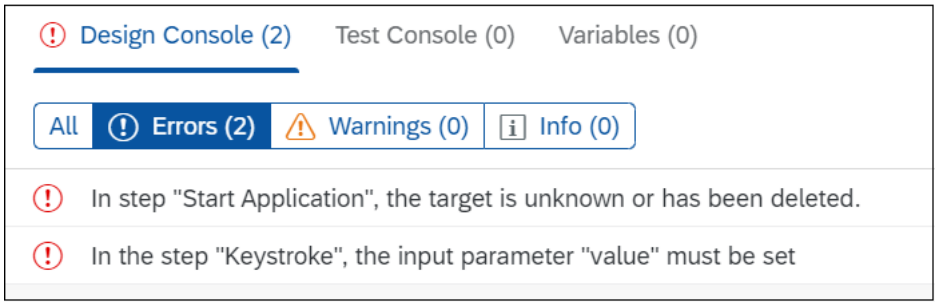
Console

In the Cloud Studio, the console displays every error, warning, variables data, and information related to your automation.

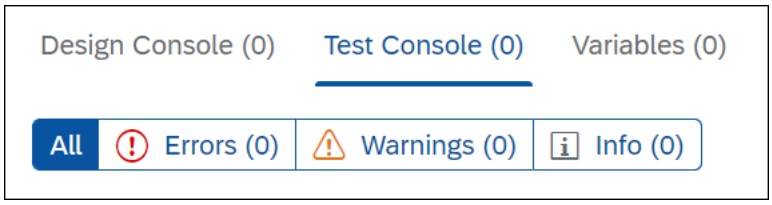


Three modes are available for the console:

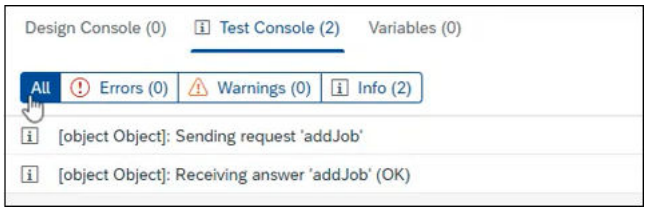
- *Design Console*: to display information related to the editing of your automation content.
  The following screenshot shows the *Design Console Error* information.



- *Test Console*: to display information related to the testing of your automation.



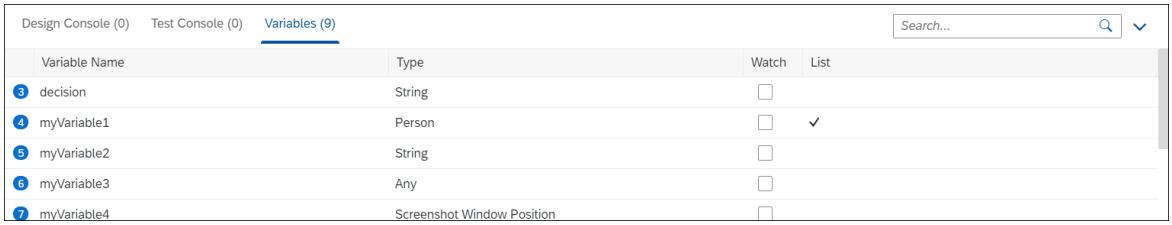
The following screenshot shows the *Test Console Error* information.



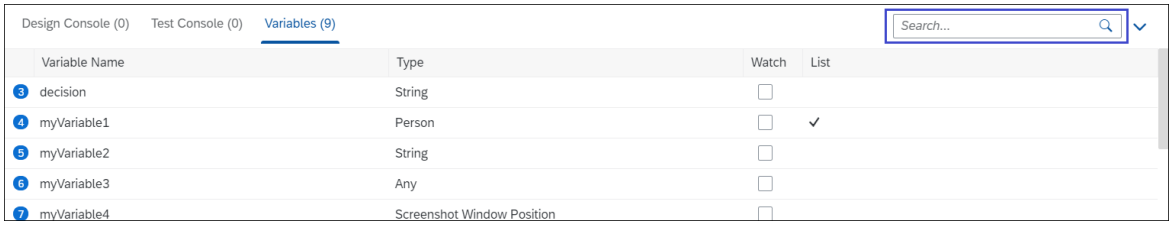
- *Variables*: to display information related to the variables of your automation.



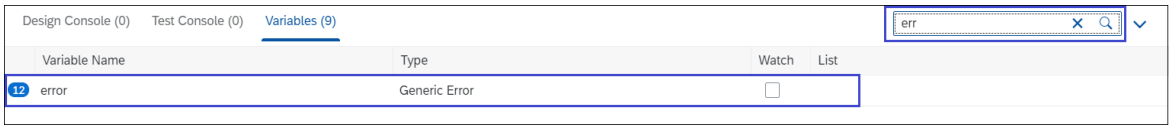
The following screenshot shows the *Variable* console variable data information of your automation.



Search Option



You can search for a specific variable using the *Search* option.



> **Note**
>
> Switching mode will not affect the content of your automation, the testing, or the elements displayed in the timeline.

4.2.11 Document Information Extraction

Document Information Extraction is a SAP BTP service that uses machine learning to automate your document information extraction processes. You can use Documentation Information Extraction directly from the SAP Intelligent RPA Cloud Studio.

Document Information Extraction helps you to process large amounts of business documents that have content in headers and tables. You can use the extracted information to automatically process payables, invoices, or payment notes while making sure that invoices and payables match. After you upload a document file to the service, it returns the extraction results from the header fields and line items.

Prerequisites

You must have access to the *Document_Information_Extraction_UI_Template_Admin* role under IRPA Officer of Role Collection to use Document Information Extraction in the Cloud Studio.

Using Document Information Extraction

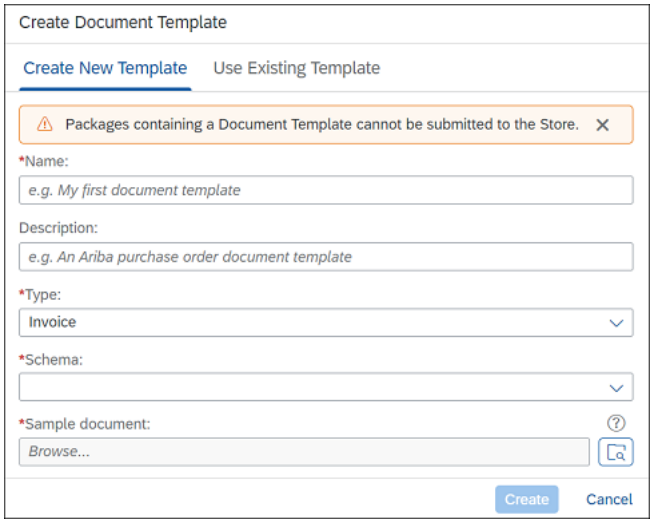
You must create a document template artifact to extract information of a document. After creating a template, you can use the template in an automation to extract a document information.

In the Cloud Studio, you have the following two options to create a document information extraction template.

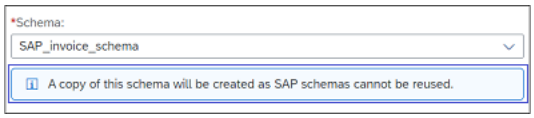
1. *Creating New Template*
2. *Use Existing Template*

Create a Document Template

You can create a new template using document type and schema. In this use case, the template is created in the Document Information Extraction UI and the artifact is created in the Cloud Studio.

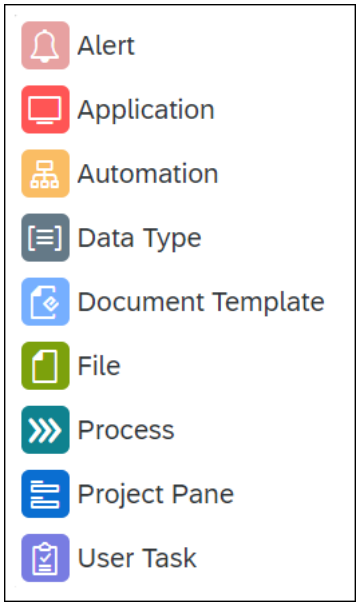


Field Name	Description

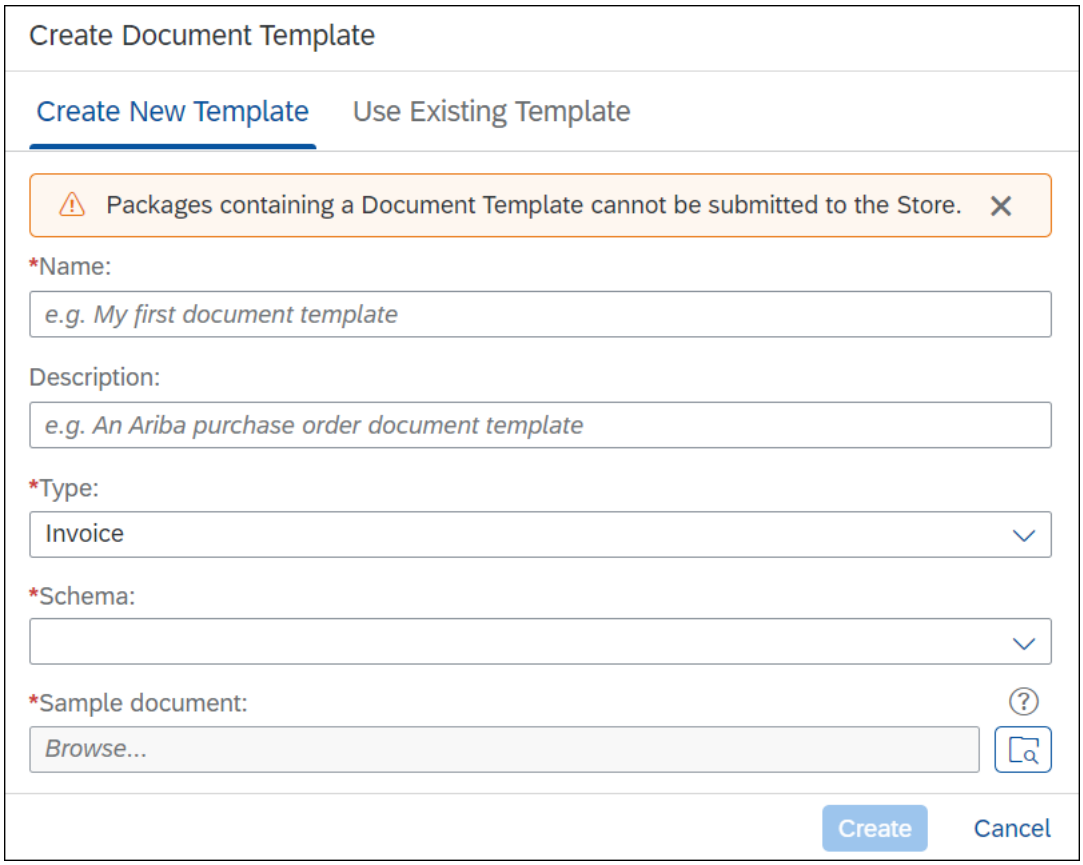
Name	The template name.
Description	The description of the template to be created.
Type	The document type. Currently, the Cloud Studio supports the following document types: • Invoice • Payment Advice • Purchase Order Each *Type* is associated with different header fields and line items. The document information is fetched based on the document *Type*.
Schema	The *Schema* for the document. The schemas are created in the **Document Information Extraction UI** and synchronised with Cloud Studio. These are associated with various header fields and line items which help you to extract required information from a document. The *SAP_invoice_schema* is default. If the *SAP_invoice_schema* is selected from the drop-down list, then the following message is displayed.  **i Note** You cannot create a schema from Cloud Studio. You can create a schema in Document Information Extraction interface. You can configure a schema to add, edit, or delete *Header Fields* and *Line Item Fields* in the Document Information Extraction UI. This will help you to extract only the required information from a document using only the required *Header Fields* and *Line Item Fields*.
Sample Document	Provides the sample document. In the further steps of creating a new template, you can annotate required fields by selecting items in the sample document.

Procedure

1. In the Project Explorer, click + and then click *Create* in the left panel or click *Create* in the Project Explorer main panel.
   The artifact menu is displayed.

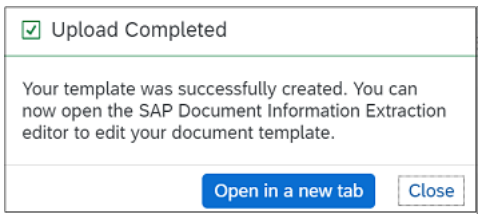


2. In the artifact menu, click *Document Template*. The *Create Document Template* screen is displayed.

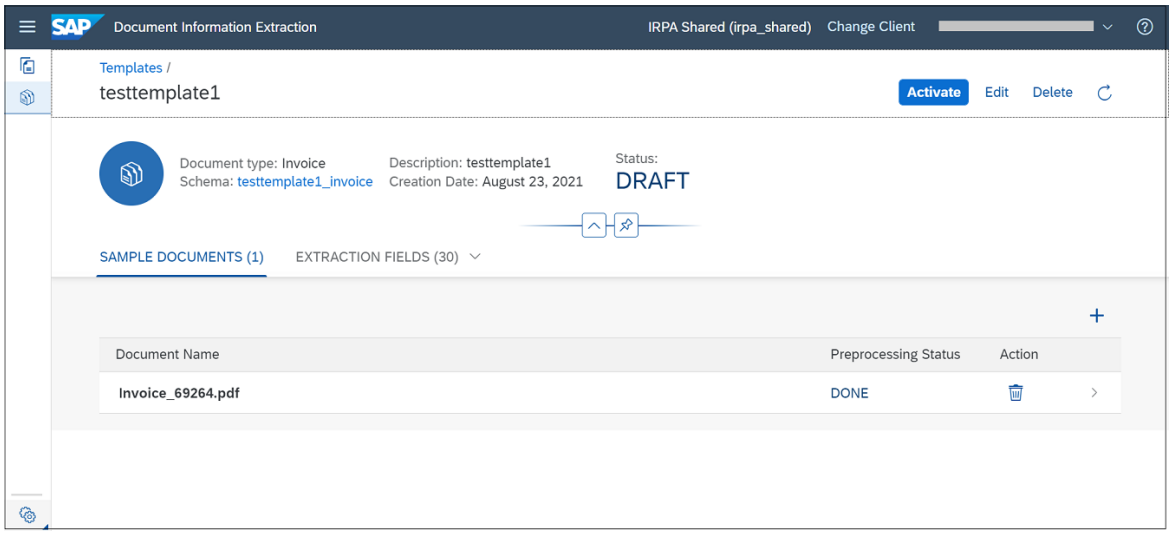


3. Enter a name for the template in the *Name* field.
4. Enter a description for the template in the *Description* field.
5. Select the template type from the *Type* drop-down list.
6. Select the schema from *Schema* drop-down list.
7. Click the icon to browse and upload the sample document.

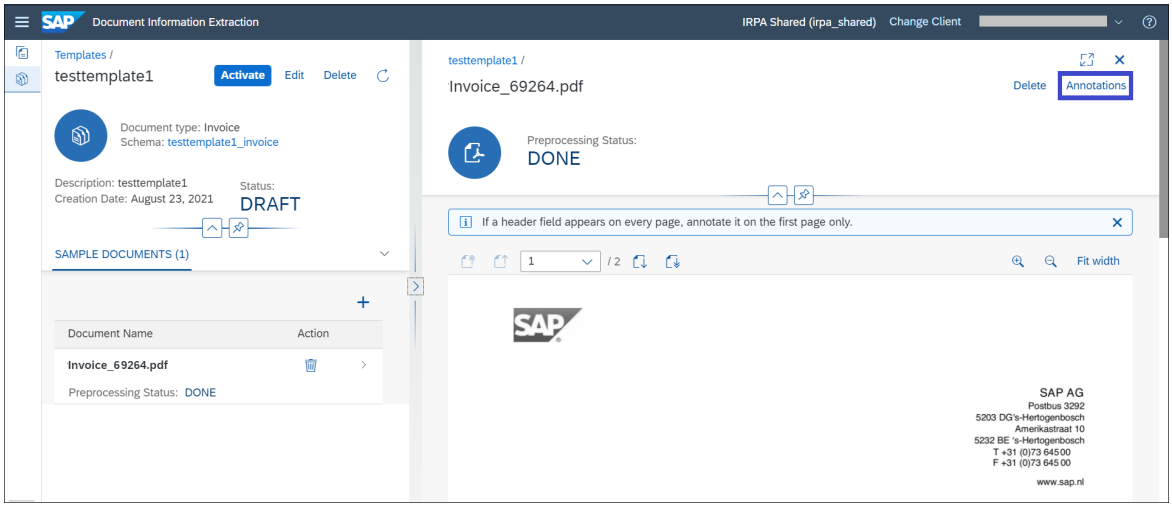
8. Click *Create*.
   After successful upload of the sample document, the following pop-up is displayed.



9. Click *Open in a new tab*. The Document Information Extraction UI opens in a separate tab. For example, refer to the following screenshot.



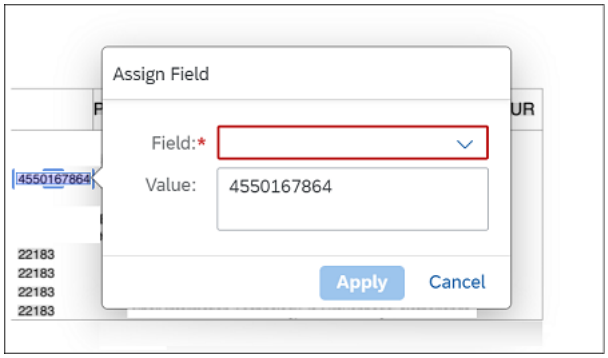
10. Click the uploaded sample document. The *Annotations* option is displayed in the right pane of the Document Information Extraction screen.



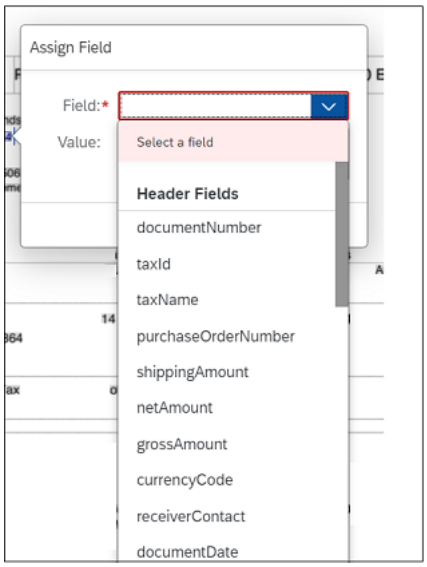
> **Note**
>
> If a business document format differs from the regular format or if you want to extract some information from a business document, then the Annotation feature can be used to create customized fields to extract the required information.

> You can configure a schema to add, edit, or delete *Header Fields* and *Line Item Fields* in the Document Information Extraction UI. This will help you to extract only the required information from a document using only the required Header Fields and Line Item Fields.

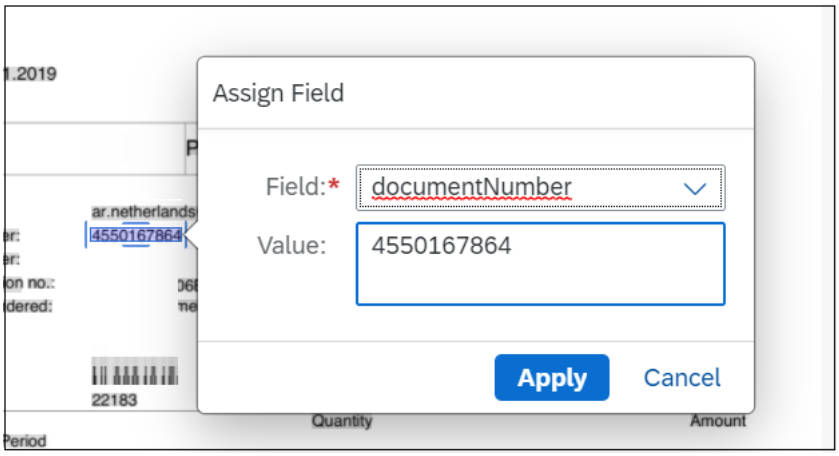
11. Click *Annotations*. The *Edit* option is displayed.
12. Click *Edit* and select the content to be annotated in the sample document.
    For example, refer to the following screenshot.



13. Click the *Field* drop-down arrow and select the appropriate field name from the list.

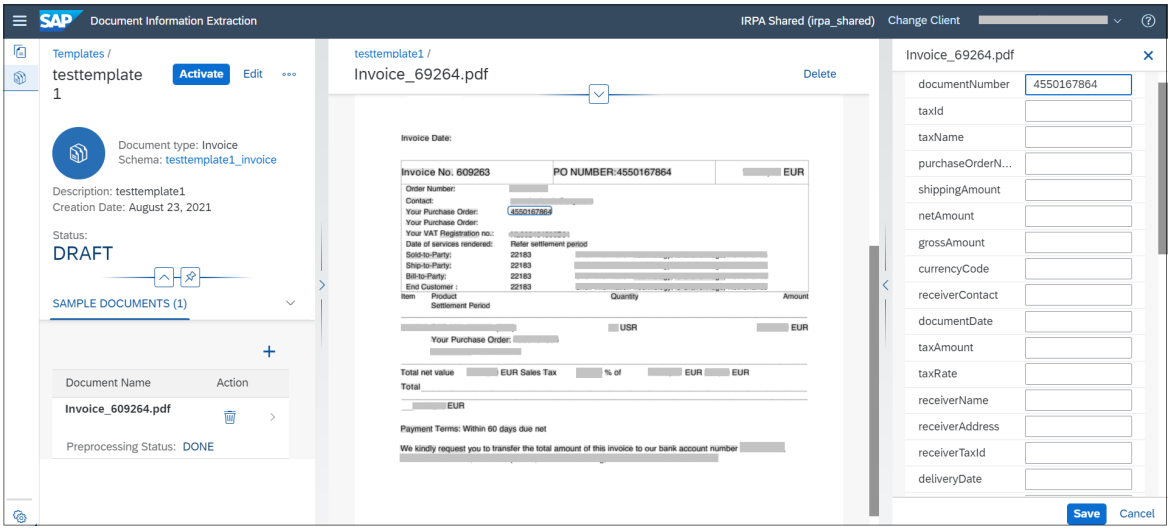


    For example, refer to the following screenshot. In the following example, documentNumber is selected as field name.



> **i Note**
>
> As per the requirement, you can select the *Header Fields* and *Line Items Fields* from the drop-down list.
>
> You must annotate at least one field. The Document Information Extraction UI prompts to create minimum *Header Fields* and *Line Items Fields*

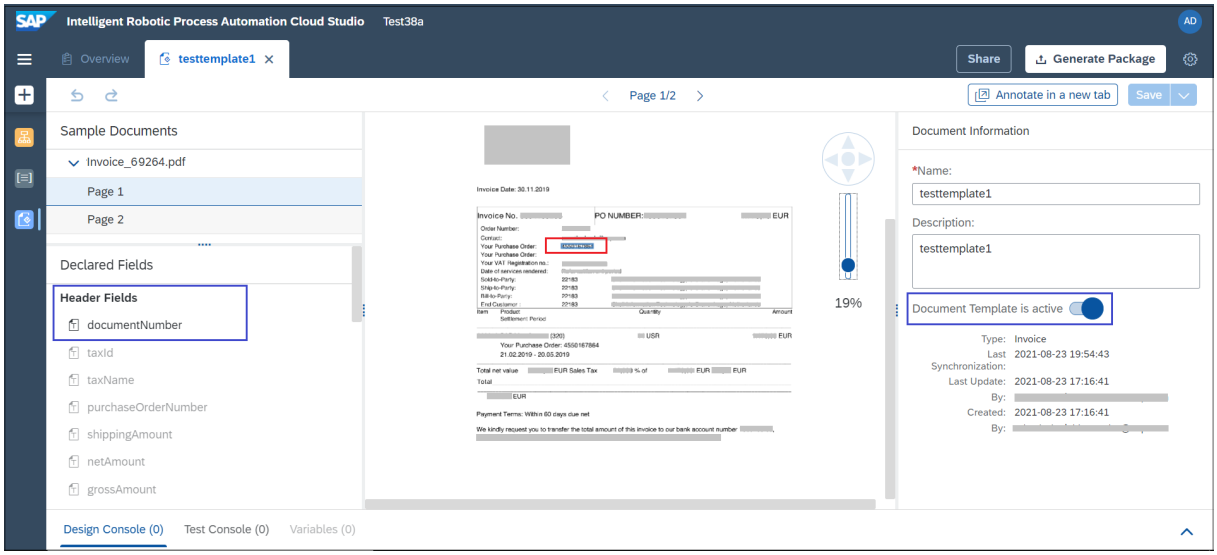
14. Click *Apply*. The documentNumber is applied for the annotated content.



This means, if you extract information from a document through an automation, the annotated content is recognized as documentNumber.

15. Click *Activate*. After successful activation, the **Template Successfully Activated** message is displayed.

Now, the new template you have created is ready. The annotated fields are highlighted in blue colour in the main pane of the Project Explorer. The list of annotated fields is displayed under *Declared Fields* in the left panel and the status of the template is displayed in the right pane.

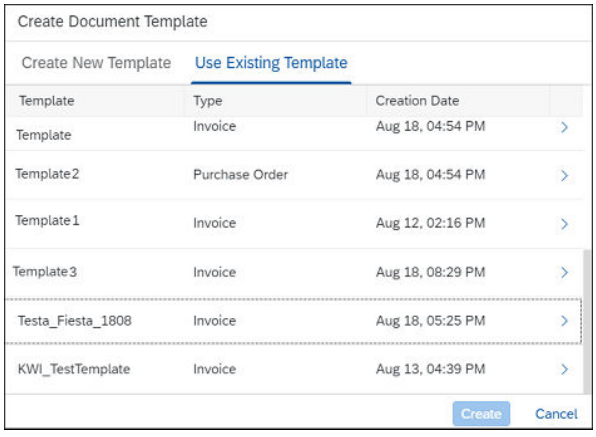


> **i Note**
>
> You can edit the template by clicking the Annotate in a new tab icon.

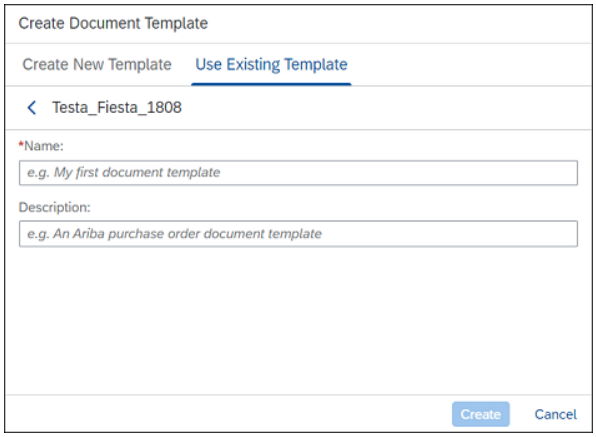
For more details about using template in an automation, refer to the Document Information Extraction Use Cases [page 277] section.

Use Existing Template

You can create a template using an existing template. In this use case, a copy of the existing template is created in the Document Information Extraction UI and an artifact is created in the Cloud Studio. This artifact uses the defined template (the template selected while creating an artifact. Refer to the screenshot 1.) to extract the information.



Screenshot 1



Screenshot 2

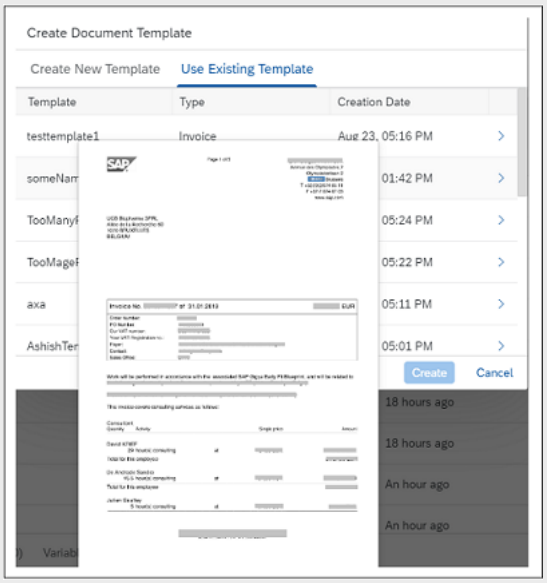
Field Name	Description

Screenshot 1	Shows the list of templates that are created in the **Document Information Extraction UI** and synchronised with Cloud Studio.
Screenshot 2	
Name	The template name.
Description	The description of the template to be created.

Procedure

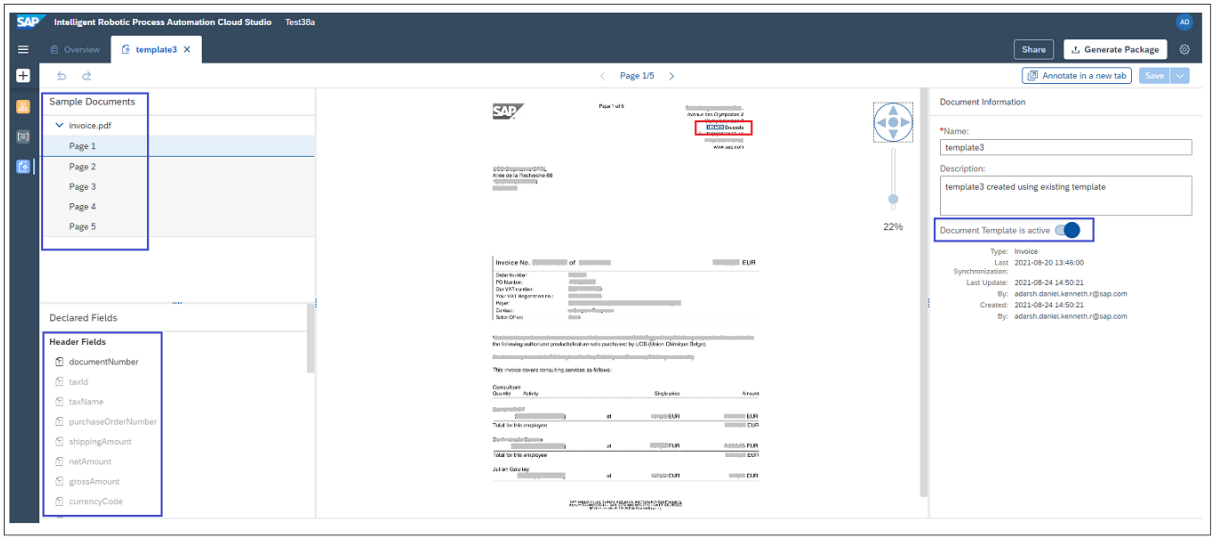
1. In the Project Explorer, click + and then click *Create* in the left panel or click *Create* in the Project Explorer main panel.
   The artifact menu is displayed.
2. In artifact menu, click *Document Template*. The *Create Document Template* screen is displayed.
3. Select the *Use Existing Template tab* and then select the template from the list of templates. Refer to the screenshot 1.

> **i Note**
>
> You can preview the template by hovering the mouse on the template name.



4. Click *Create*. The selected template name is displayed. Refer to the screenshot 2.
5. Enter a name for the template in the *Name* field.
6. Enter a description for the template in the *Description* field.
7. Click *Create*.
   After successful upload, the pop-up is displayed.
   Close the pop-up window. The Project explorer window is displayed.

Now, the new template you have created is ready. The annotated fields are highlighted in blue colour in the main pane of the Project Explorer. The list of annotated fields is displayed under *Declared Fields* in the left panel and the status of the template is displayed in the right pane.



4.2.11.1 Document Information Extraction Use Cases

Extracting Document Information Using Template

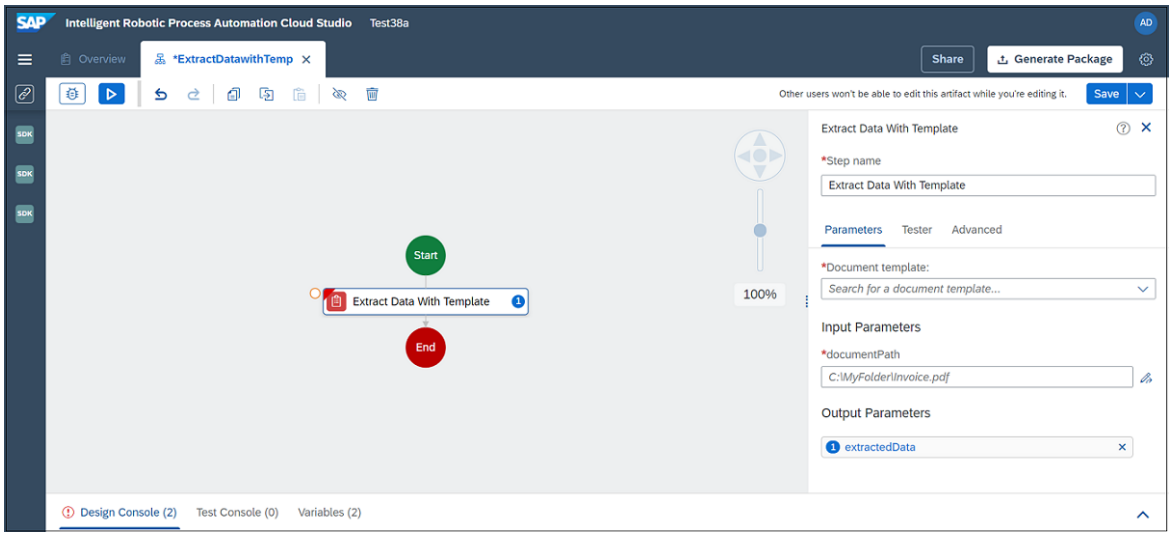
You can extract a document information using the Document Information Extraction service with the defined template in the *Extract Data with Template* activity for a targeted pdf document.

Prerequisites

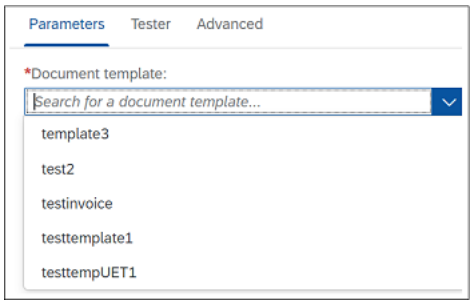
You must have created at least one template. For more details about creating template, refer to the Document Information Extraction [page 269] section.

Procedure

1. In an automation workflow drag and drop the *Extract Data With Template* activity.
2. Select the *Extract Data With Template* activity. The activity parameter fields are displayed.



3. Select an appropriate template from the drop-down list.



> **i Note**
>
> The output (extracted data) depends on the selected template.

4. Enter the document path in the *documentPath* field.
5. Click *Save*. The automation is saved.
6. Click the icon to test the automation.

Result

The extracted information is displayed in the *Output Parameters*. Select the activity in the debug panel to see the extracted data.

Extract Data without Template

You can extract a document information using the Document Information Extraction service for a targeted pdf document.

Procedure

1. In an automation workflow drag and drop the *Extract Data Without Template* activity.
2. Select the *Extract Data Without Template* activity. The activity parameter fields are displayed.
3. Select an appropriate document type from the drop-down list.

> **i Note**
>
> The output (extracted data) depends on the selected document type.

4. Enter the document path in the *documentPath* field.
5. Click *Save*. The automation is saved.
6. Click the icon to test the automation.

Result

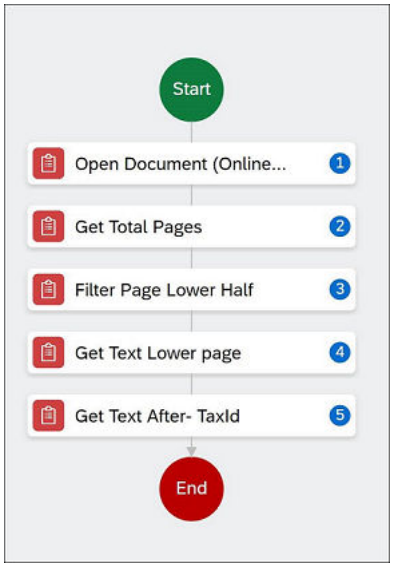
The extracted information is displayed in the *Output Parameters*. Select the activity in the debug panel to see the extracted data.

Open Document (Online OCR)

You can extract information from images and scanned documents using the OCR provided by the Document Information Extraction service. It is an alternative for the existing *Open PDF* activity which only supports machine readable PDF document.

Procedure

1. In an automation workflow drag and drop the *Open Document (Online OCR)* activity.
2. Select the *Open Document (Online OCR)* activity. The activity parameter field is displayed.
3. Enter the document path in the *documentPath* field.
4. Drag and drop the required PDF activities.
   For example, refer to the following screenshot.



5. Click *Save*. The automation is saved.
6. Click the ▷ icon to test the automation.

Result

The extracted information is displayed in the *Output Parameters*. Select the activity in the debug panel to see the extracted data.

Related Topics

Document Information Extraction [page 269]

Create an Automation [page 156]

Add an Activity to an Automation [page 208]

Test an Automation [page 282]

4.2.12 Support Multi-level Collections in an Automation

In the application editors, you can declare elements of N dimensions and the same can be executed in an Automation.

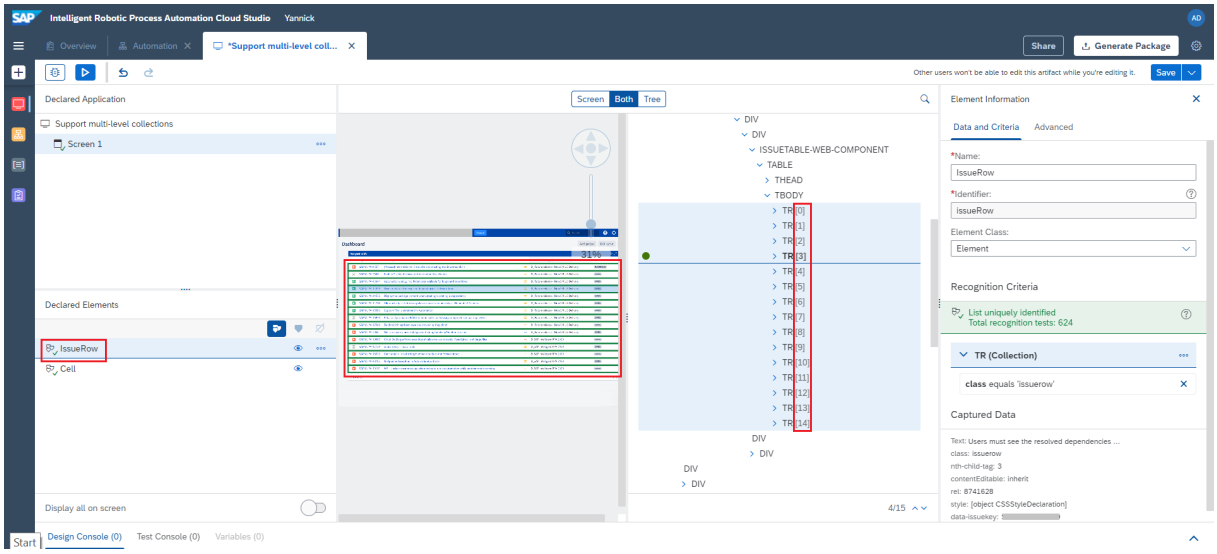
For example, cells in the tables (2 dimensions), cells in the list of tables (3 dimensions), and detailed properties in the list of orders (2 dimensions).

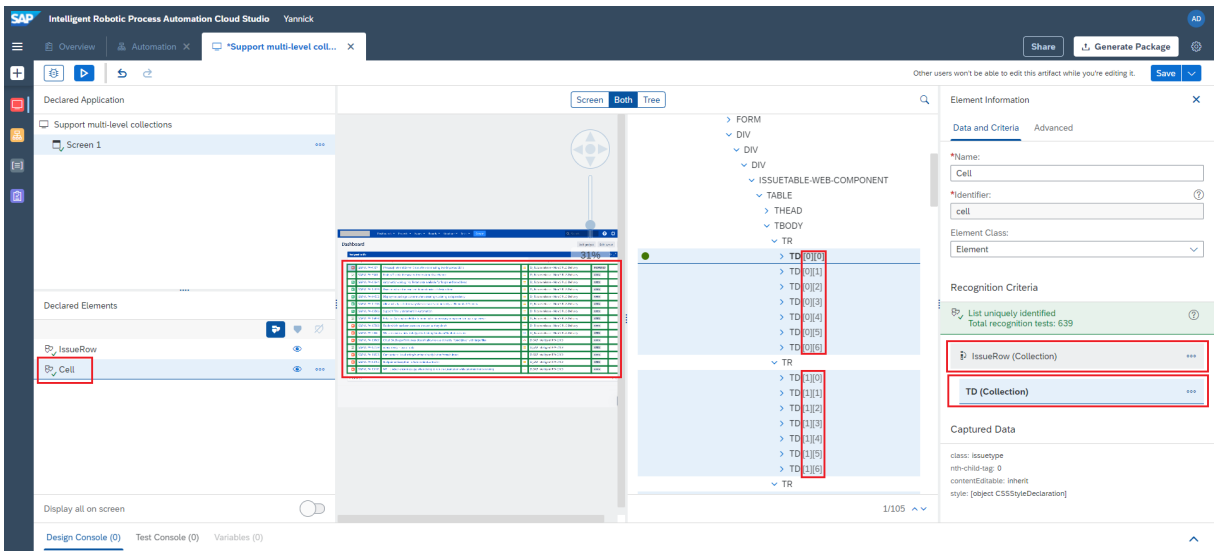
You must iterate each element collections in an automation to support nested loops.

In the following example, each line item in the table is looped to get the targeted table rows and cells data.

Prerequisites

You must capture an application table and declare its rows and cells. For example, refer to the following screenshots.

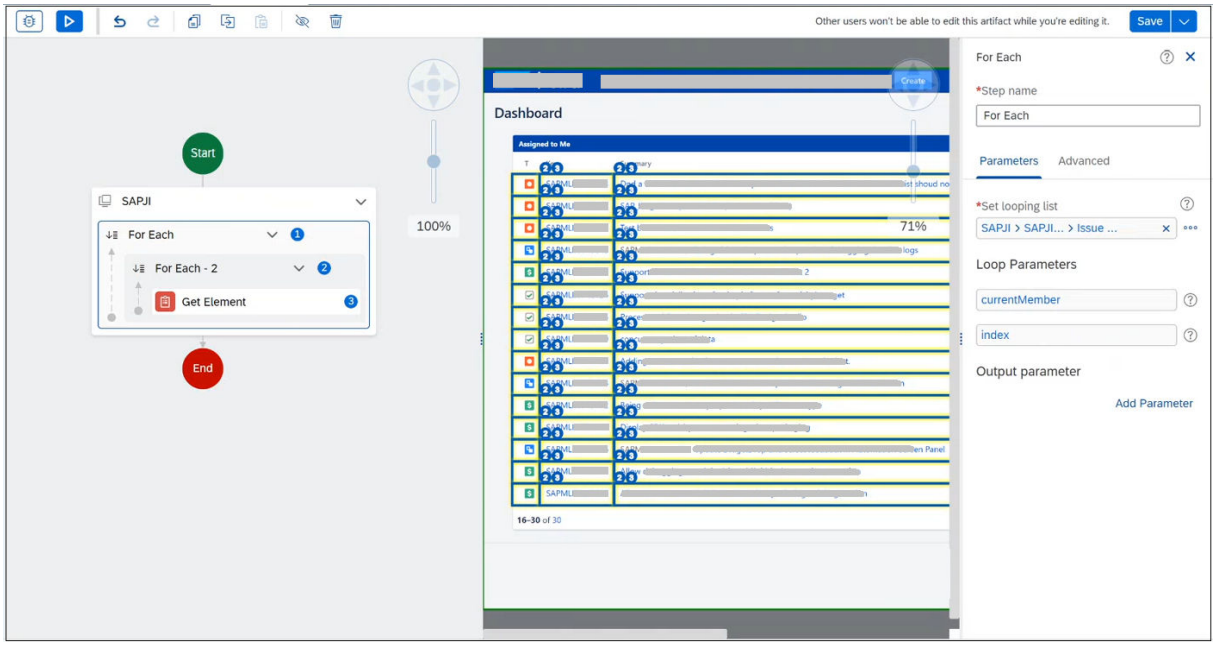




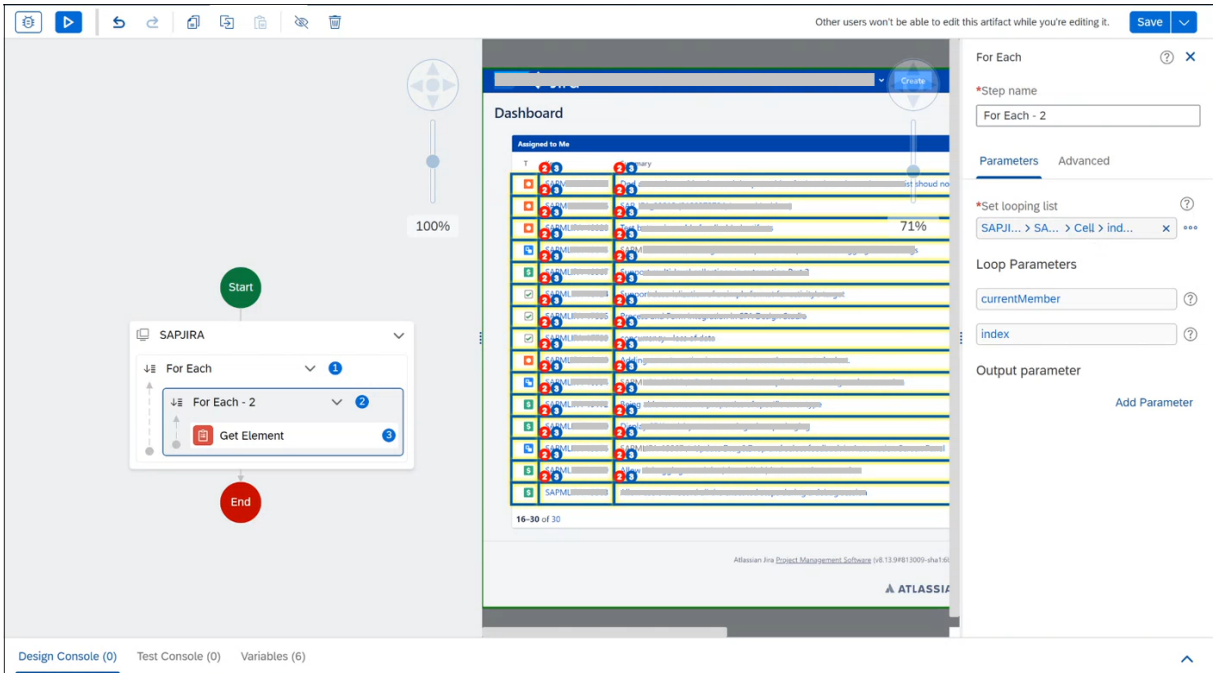
For more details about declaring an element, refer to the following sections:

1. Declare an Application [page 101]
2. Declare an Element [page 110]
3. Declare an Element as a Collection [page 120]
4. Advanced Declaration [page 123]

Automation Workflow



In the above automation workflow, the first *For each* is looping through the rows.



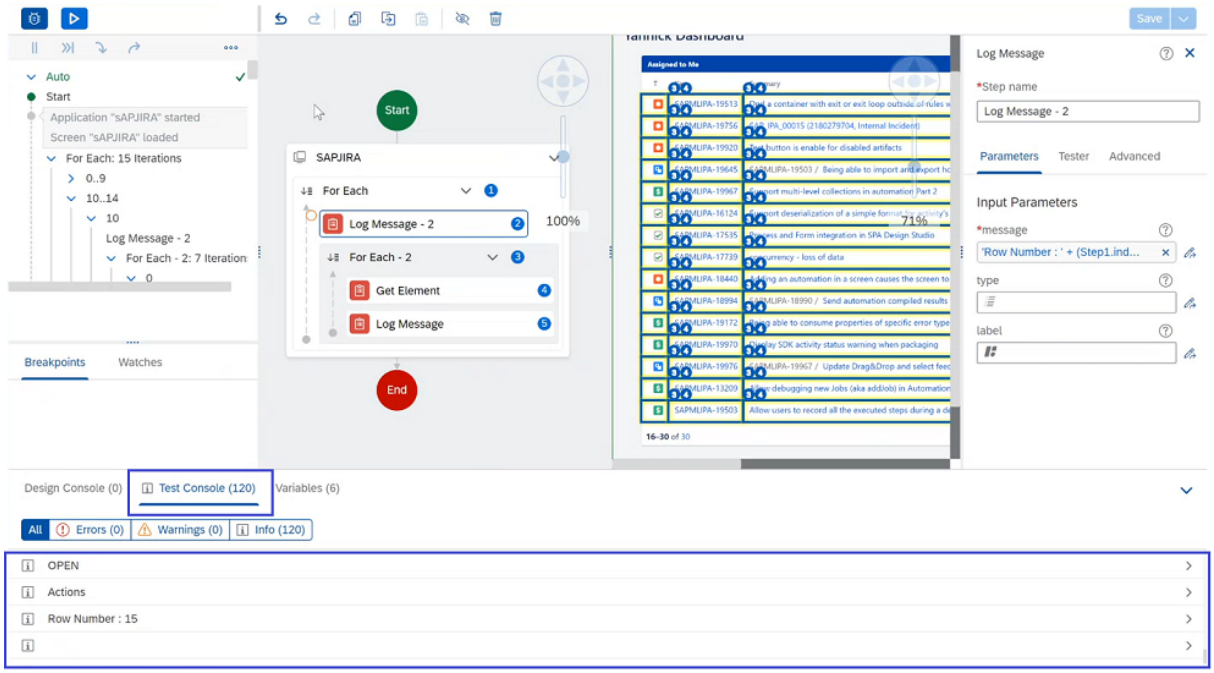
In the above automation workflow, the second *For each* is looping each cell of the row and then gets the text inside each cell.

Related Topics

- For Each [page 217]
- Add an Activity to an Automation [page 208]

Result

The data is extracted from the targeted rows and cells of the table.



4.2.13 Test an Automation

The tester side panel helps you test, troubleshoot, and display the execution flow of your automation.

In the Cloud Studio, on the left side of your project, you have the possibility to expand and collapse the tester side panel. This tester panel displays:

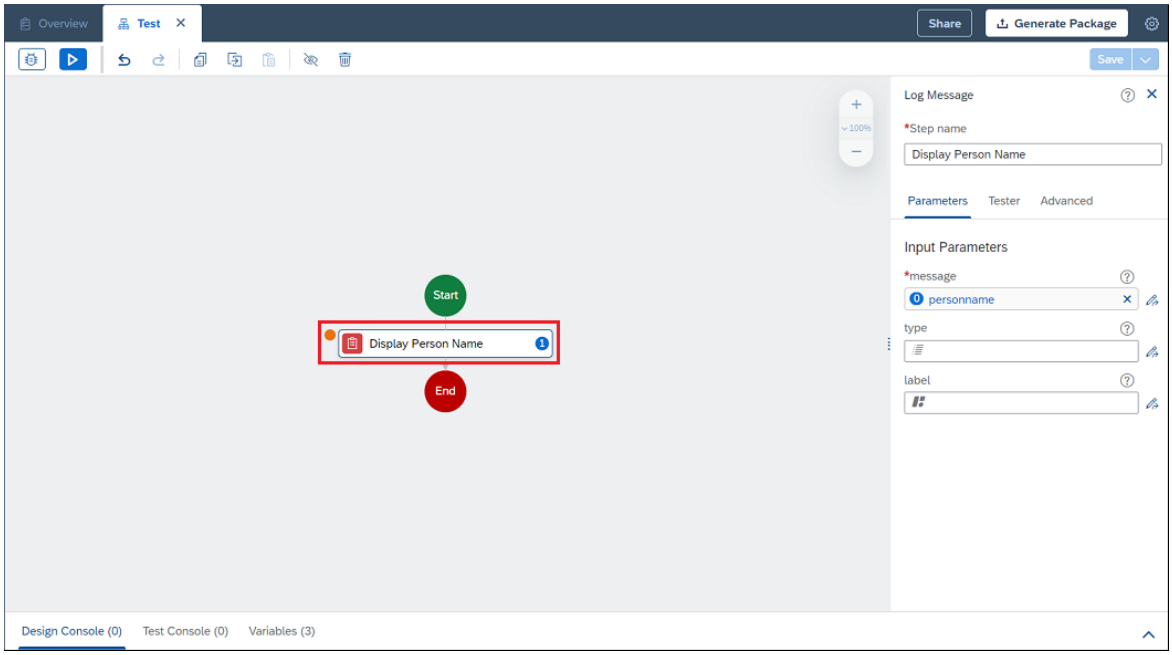
- the automation tester with the debugging toolbar
- the timeline of the execution flow of your automation
- the event logs inside the timeline of the execution flow of your automation. The following event logs that an agent sends to the cloud studio are:
  - an application starts
  - a page starts
  - a page unloads
  - the application stops
- breakpoints on the breakpoint panel

Prerequisite

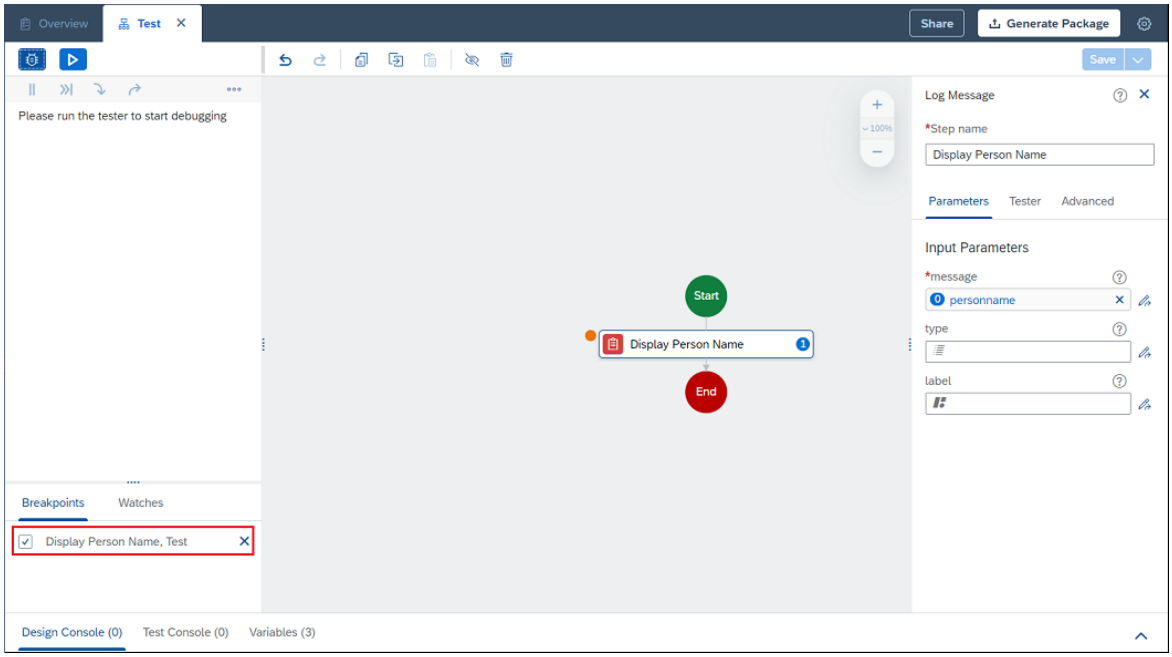
You have created an environment in the Cloud Studio. For more information, see Create an Environment.

Procedure

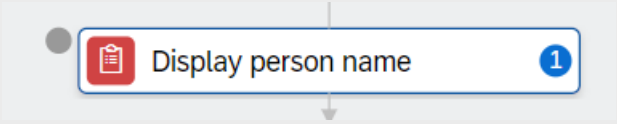
1. In your automation, define a breakpoint by clicking the orange circle on the left side of a step.



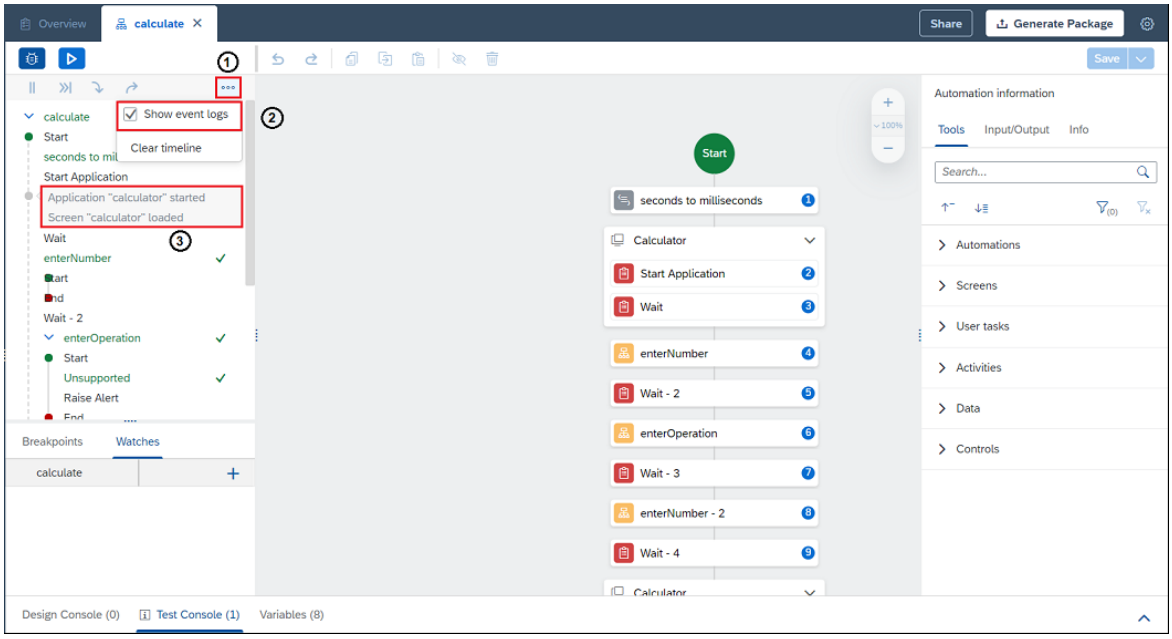
2. Expand the tester side panel by dragging the splitter on the left of the automation editor.
   The label of the step where you set a breakpoint is displayed, as well as the label of the automation.



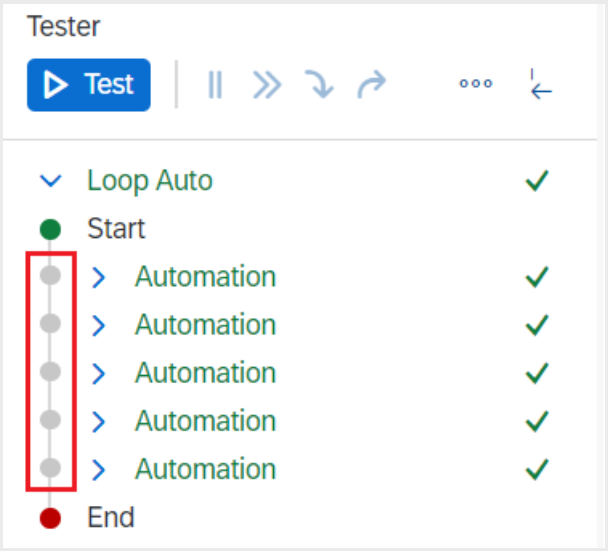
3. **Optional**: To deactivate a breakpoint, untick the checkbox next to the step.

> **i Note**
>
> If you deactivate a breakpoint on the tester side panel, the orange circle next to the step turns gray.
>
> 
>
>
> As a result, the following automation test does not stop at that breakpoint.

4. **Optional**: To delete a breakpoint, click the cross button next to the breakpoint.
5. Click the *Test* button on the debugging toolbar to test your automation.
   A *Test Automation* window pops up.
6. Select the environment containing your local agent.
7. **Optional**: Fill in the required input parameters.
   For more information on how to assign values to your input parameters when testing your automation, see Manage Input Parameters to Test an Automation [page 287].
8. **Optional**: Go to *Options*  (1) and check the *Show event logs* (2) checkbox if you want to include event logs (3) in the timeline on the tester side panel of your automation. By default, *Show event logs* is in enabled mode.



> **i Note**
>
> The following gray circle next to the automation in the timeline on the tester side panel indicates that there are event logs inside the container, and you need to expand it if you want to display those event logs.

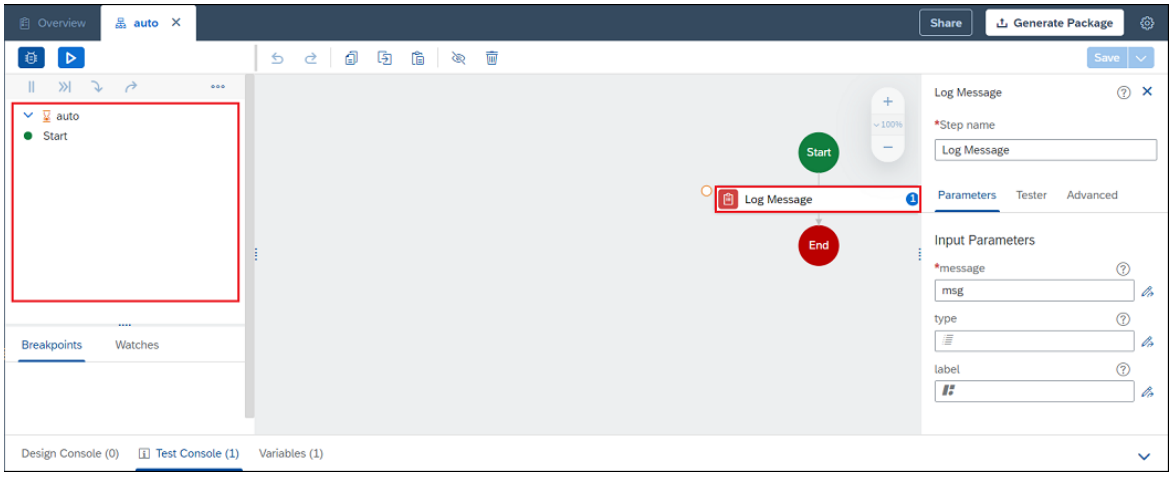


9. Click *Test*.

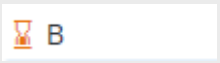
> **i Note**
>
> The debugging toolbar has the following commands:

Debugging commands			
	**Re-sume**	F8	Continue execution until the next breakpoint or the end of the automation.
	**Step over**	F9	Continue execution until the next step in the automation.
	**Step into**	F10	If possible, enter the highlighted breakpoint activity. **i Note** If the activity is an SDK activity, the **Step into** command automatically becomes the **Step over** command.
	**Step out**	SHIFT + F1	Exit the highlighted breakpoint activity.
	**End test**	SHIFT + F8	Stop the tester.

Once the testing starts, the execution flow of your automation is displayed simultaneously in a timeline on the tester side panel.



Activities are executed consecutively in the timeline, meaning that the testing of the next timeline step does not start until the previous activity is finished.

> **i Note**
>
> The following symbol indicates that the activity is still running:
>
> 
>

The execution stops on the breakpoint you previously set in the workflow of the automation.

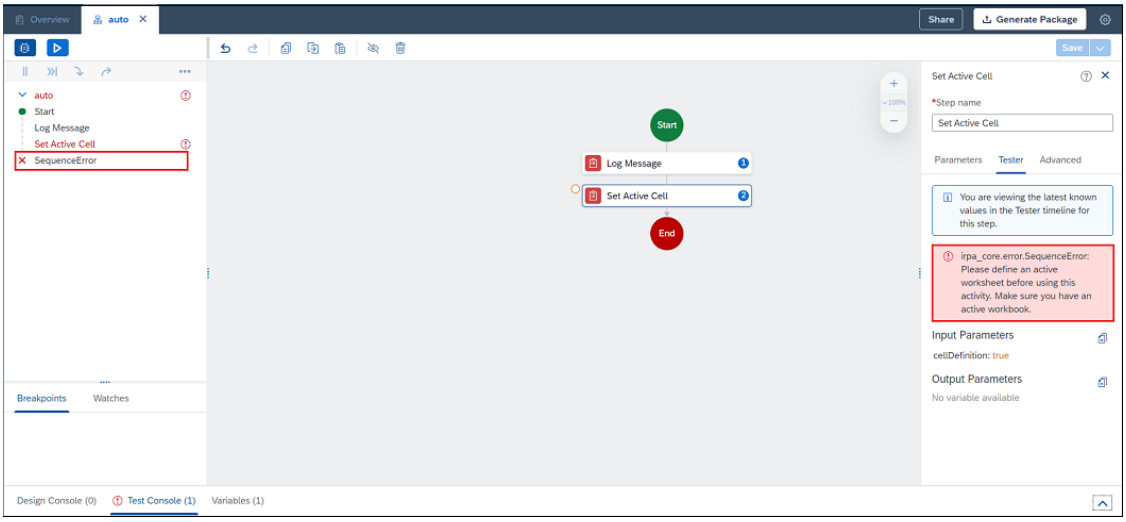
10. Click the ⏩ command to resume the testing of your automation:
    - If the testing session is successful, the automation is displayed in green with a checkmark on the tester side panel.

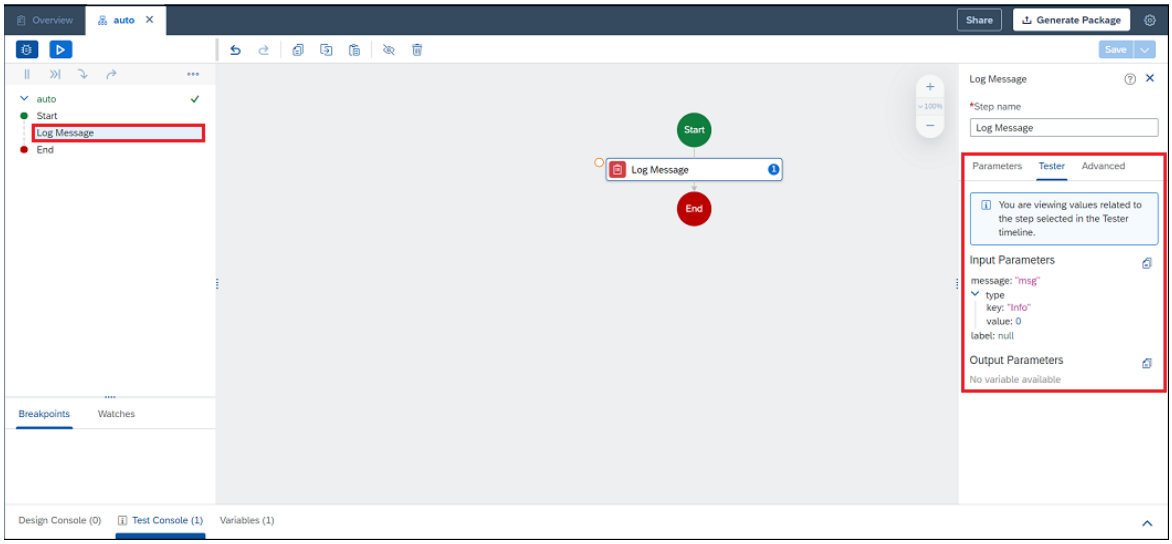
    - If the testing session fails, the automation as well as the failing step in your workflow, are displayed in red with a warning symbol on the tester side panel.

      Additionally, the error in the workflow of your automtion is labeled on the timeline. You can click on the error to display more information on the *Tester Data* side panel.



11. In the timeline, select a step to display its information.



The *Tester Data* side panel appears and displays values such as the input and output parameters of the step. Complex objects are organized in a hierarchy and the elements follow a color code for better visibility.

12. Click the *Stop* command on the debugging toolbar to stop the tester.

4.2.13.1 Manage Input Parameters to Test an Automation

Input parameters are variables storing values, that allow you to pass data in the workflow of your automation.

In the following use case, the user must fill in the required values in the input parameters, to allow the testing of the automation.

Prerequisites

- You have followed the procedure as described in Test an Automation [page 282].
- You have created the automation *A*.

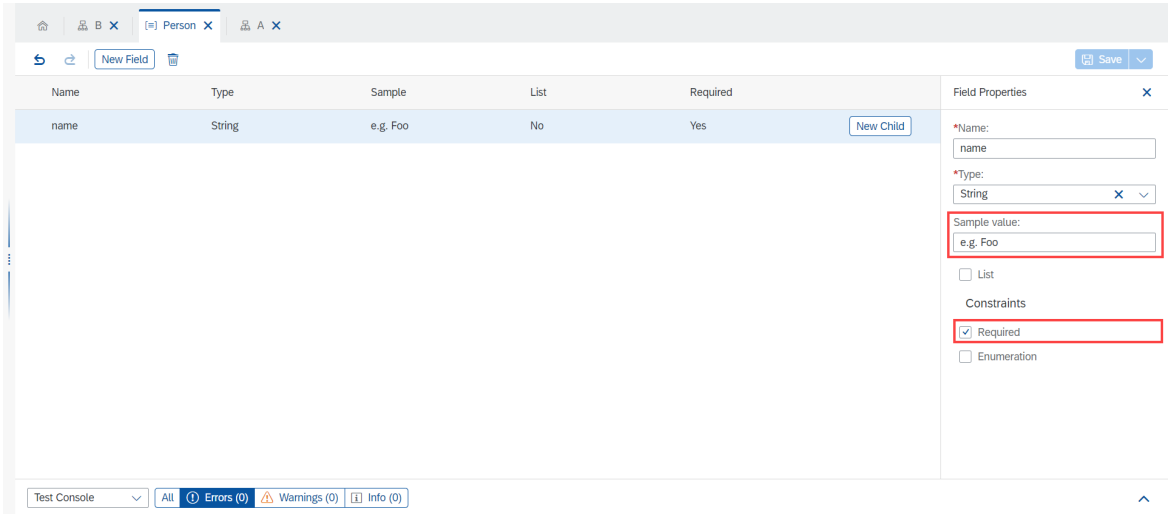
Use Case Example

1. In your project, create the data type *Person*.
   For more information, see Create a Data Type [page 168].
2. In the data type *Person*, add a new field *name* of type string.
3. Select the field *name*.
   1. In the *Field Properties* side panel, tick the *Required* box.

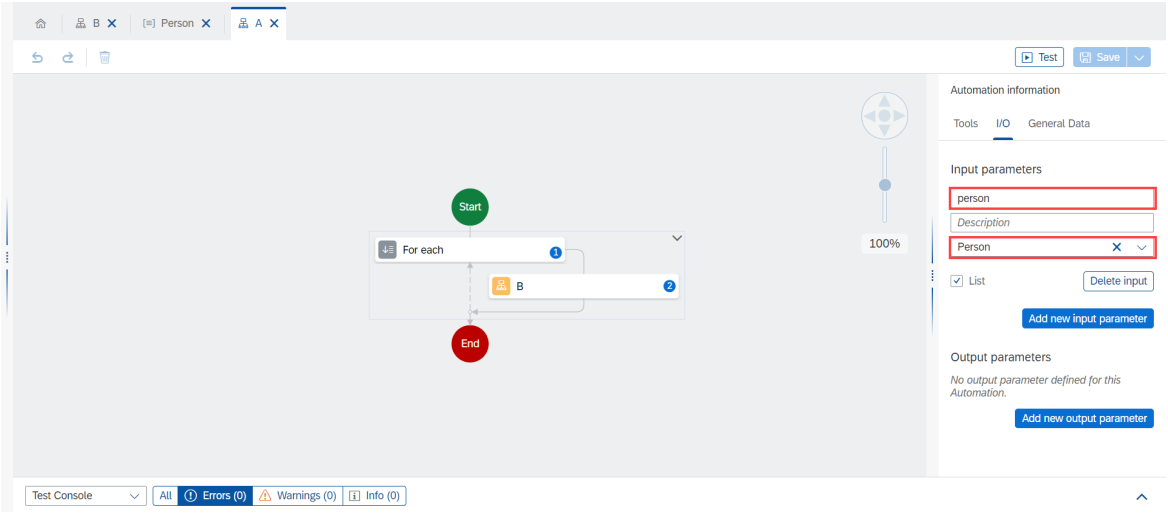
      > **i Note**
      >
      > The selected fields are required to allow the testing of your automation.

   2. Enter a sample value.

      > **i Note**
      >
      > Sample values are only available for variables of type string and number.

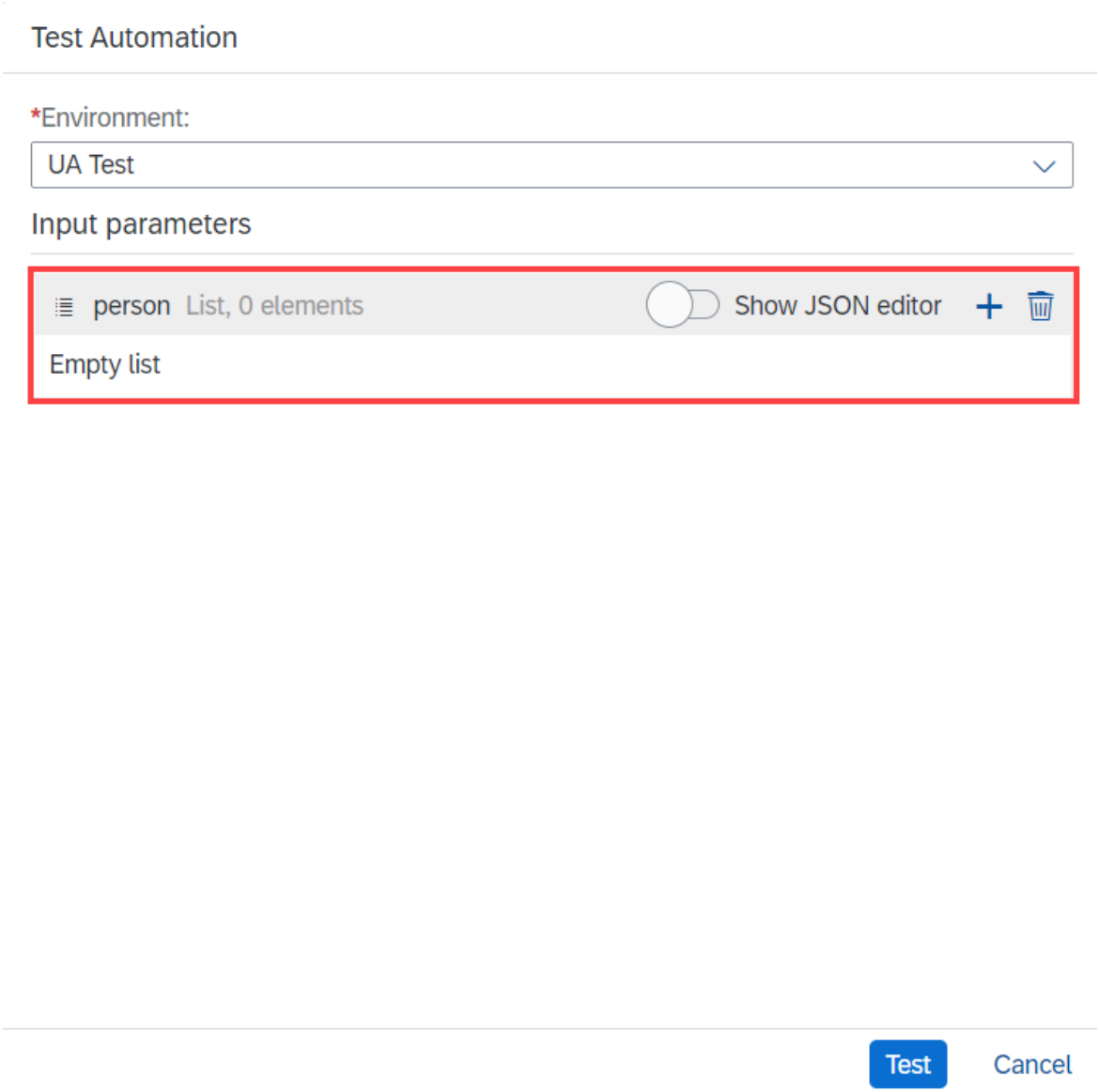


4. In the workflow of automation *A*, drag and drop the control *For Each*.
5. Drag and drop automation *B* on the repeat branch.
6. In the *I/O* section, add the input parameter **person** and select the data type *Person* as type.



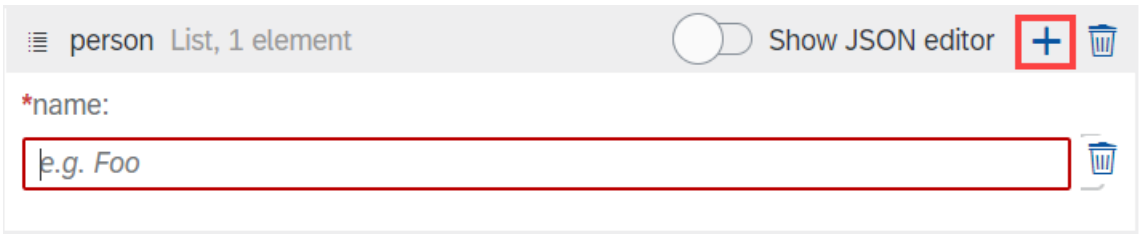
For every person in automation *B*, automation *A* will display the name in the debugger bar.

7. Click *Test* to test your automation.
   A *Test Automation* window pops up.
8. Select an environment.



The visual editor displays the input parameters you have to fill in to properly execute your automation.

9. To add an element to the input parameter list, click +.



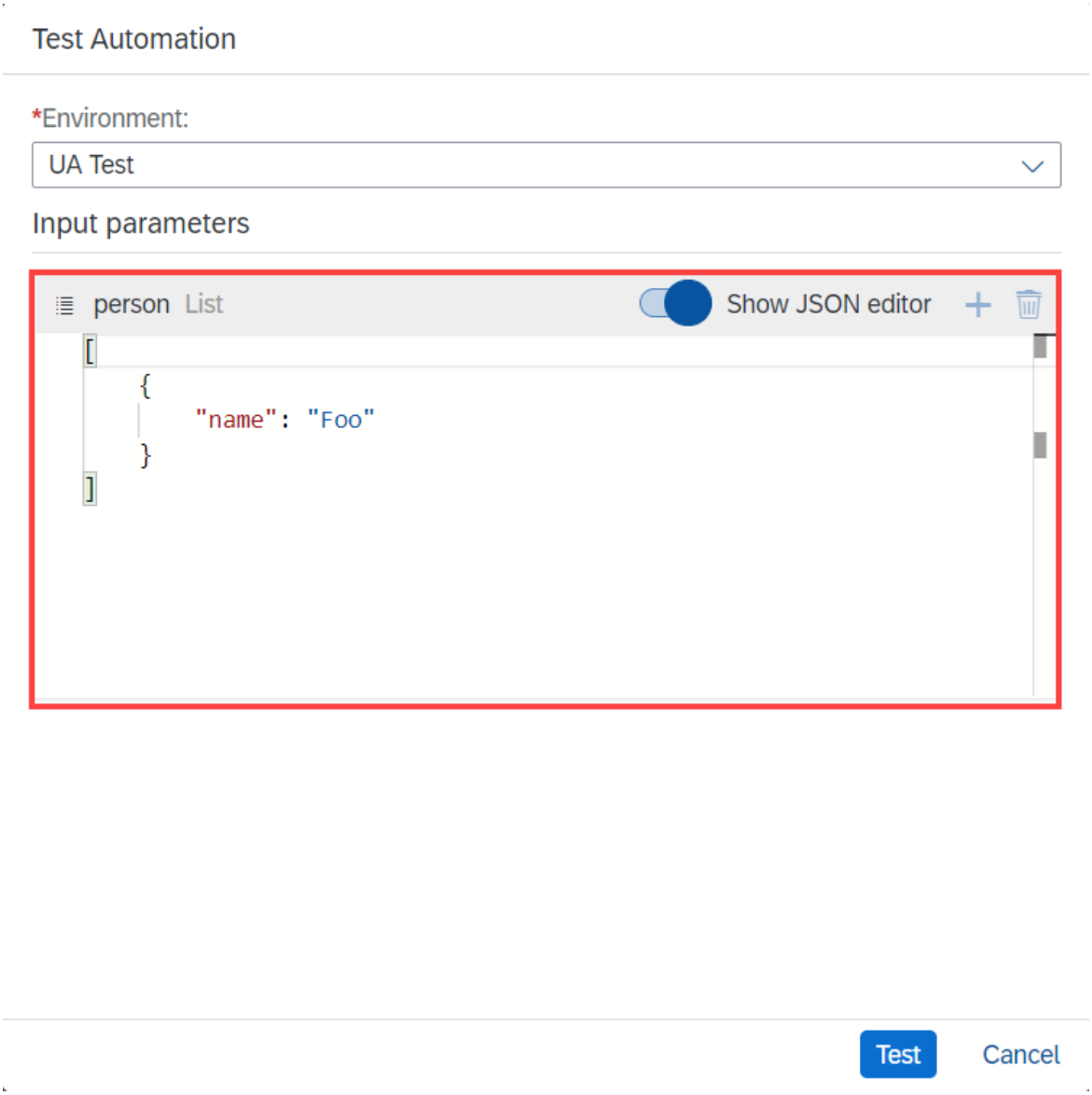
As a result, the automation calls for the input parameter **name** to execute the testing.

10. Enter a value for the input parameter **name**.

> **i Note**
>
> The *Test* button is disabled until you enter the required values.

11. To set the input parameters in JSON format, click *Show JSON editor*.

> **i Note**
>
> The values entered in the visual editor are kept when switching to the JSON editor.



Now that you have filled in the values for the input parameters, the automation is ready for testing.

12. Click *Test*.

4.2.13.2 Manage Environment Variables in Automation Testing

You can create environment variables and then you can consume it in an automation. Environment variables can be used as an input and/or output parameter [page 160] in automation processes.

Use Case

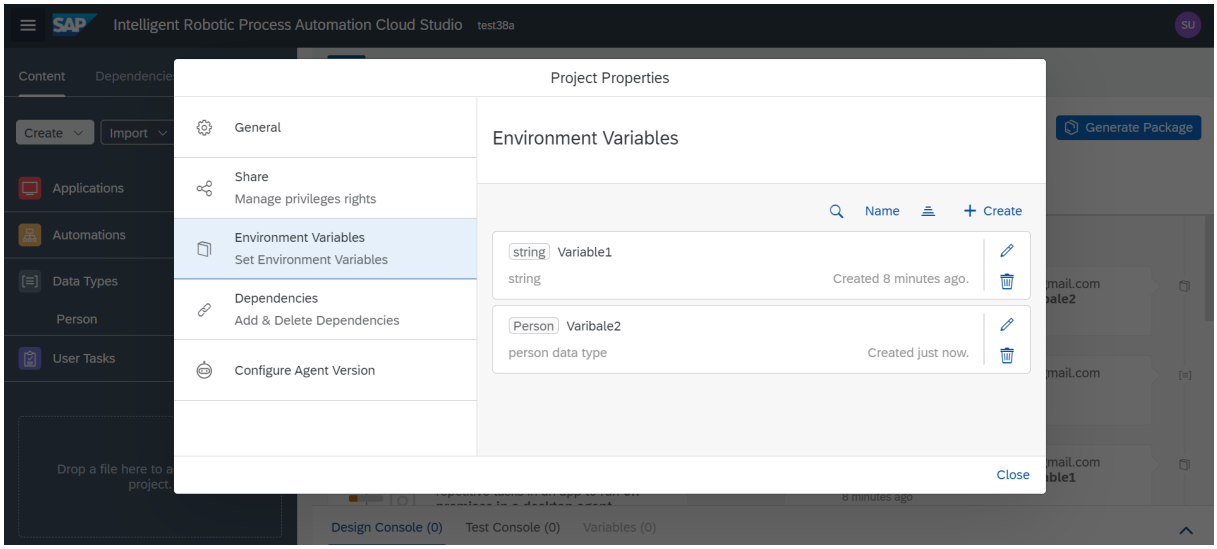
The following use case describes how to handle environment variables in an automation using simple data type.

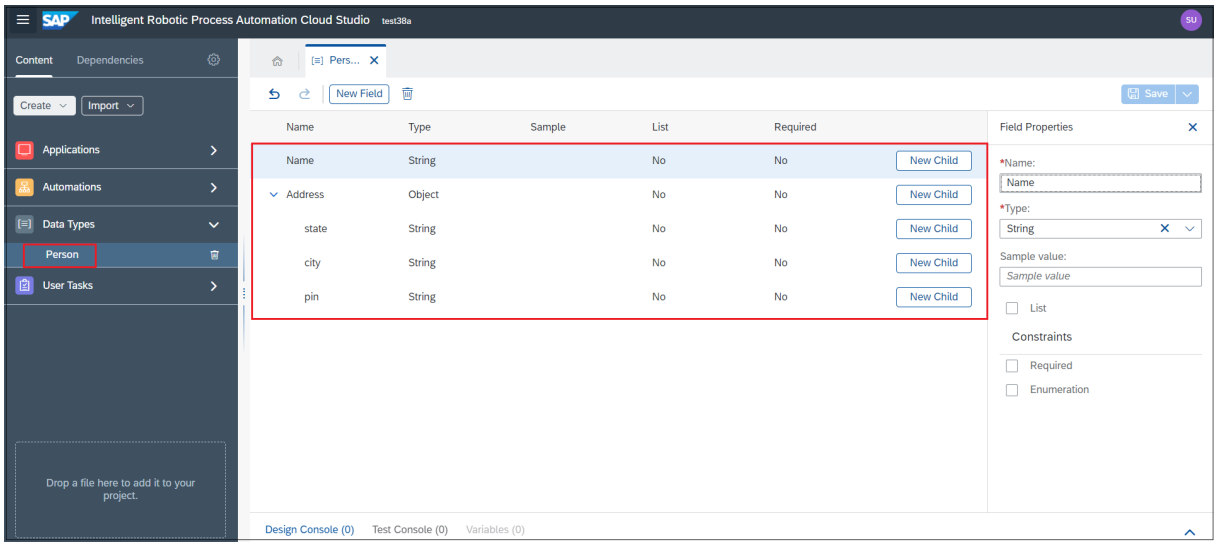
> **Note**
>
> You can also manage any data type such as complex data type, password data type and many more as environment variables in an automation.

Prerequisites

You have created one or more Environment Variables [page 509] using data types [page 160] (String, Number, Boolean, Password, and many more).

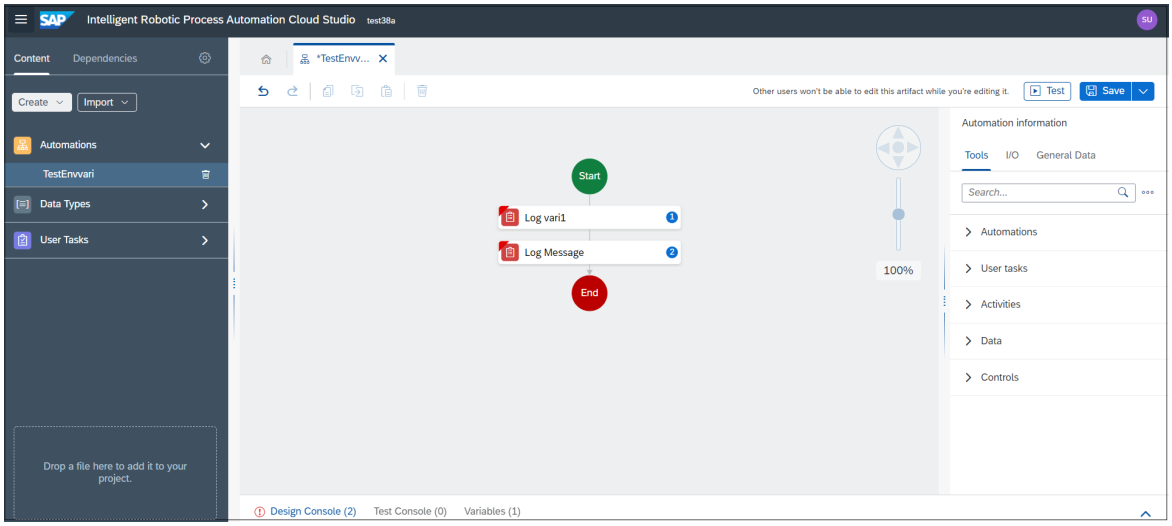
For example, refer to the following screenshots.





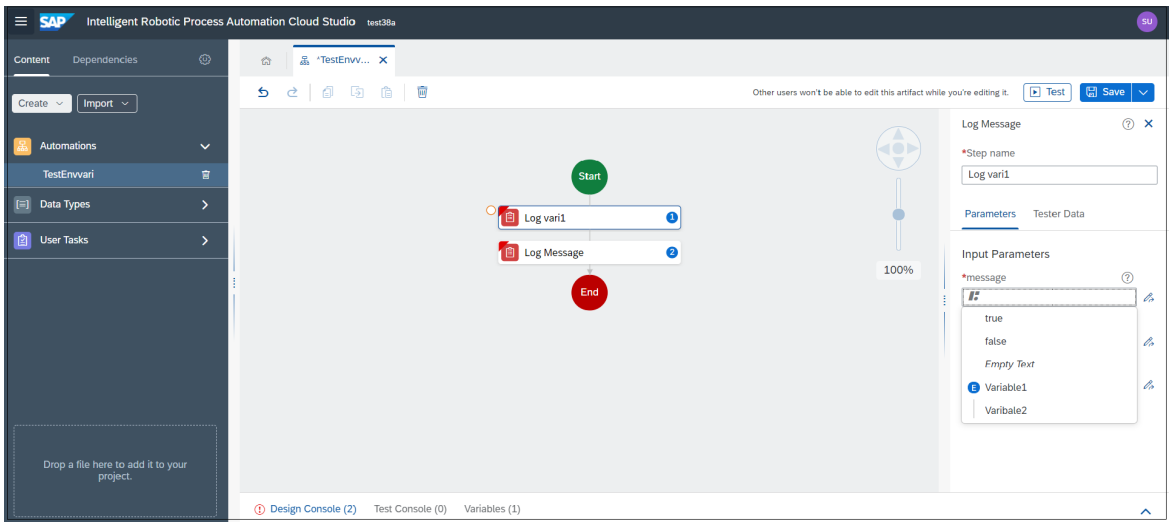
Procedure

1. Create an automation in your project.

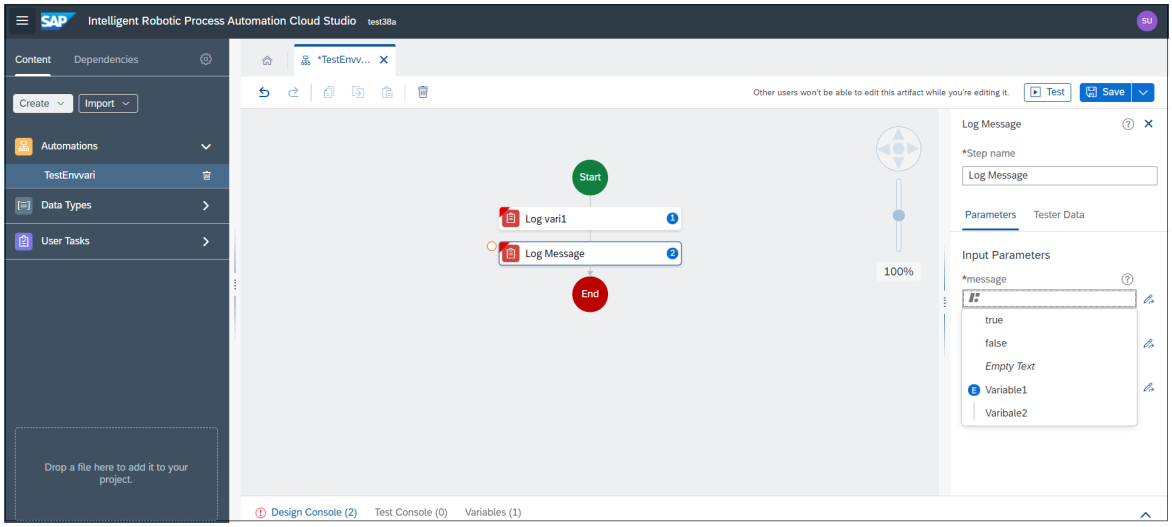


For more details about creating an automation, refer to the Create an Automation [page 156] section.

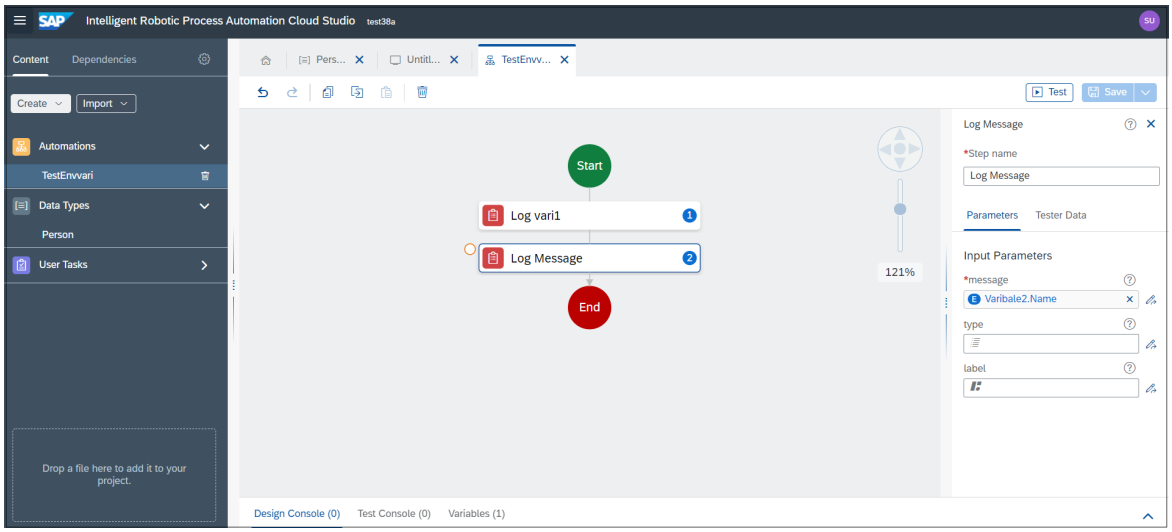
2. Bind the environment variable to the input parameter of an activity.



The input parameter of the first step is the *message* and calls for an input of type string and expects an appropriate variable value at run time to return desired output.



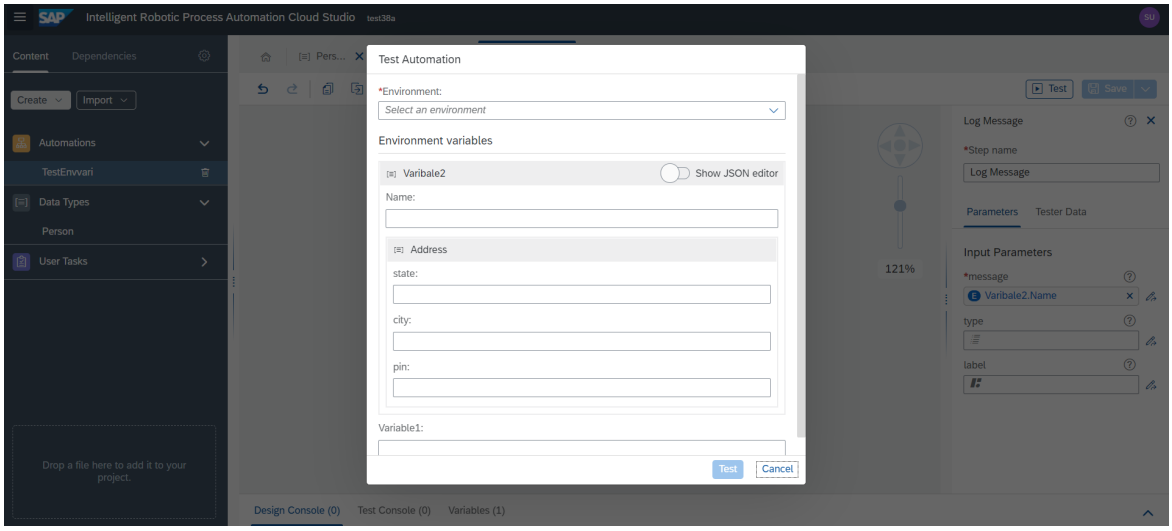
The input parameter of the second step is the *message* and calls for an input of type string and expects an appropriate variable value at run time to return desired output.



> **Note**
>
> You can only set a variable value of a defined type. For example, in the case of a type string, you cannot set a numeric value.

> **Note**
>
> To handle variables in an automation you must bind the environment variables in the input parameters of an activity. To know more details about input and output parameters, refer to the Input/Output Parameters [page 160] section.

3. *Save* the automation and then click *Test*.
   The Cloud Studio prompts a *Test Automation* screen for appropriate variable values.

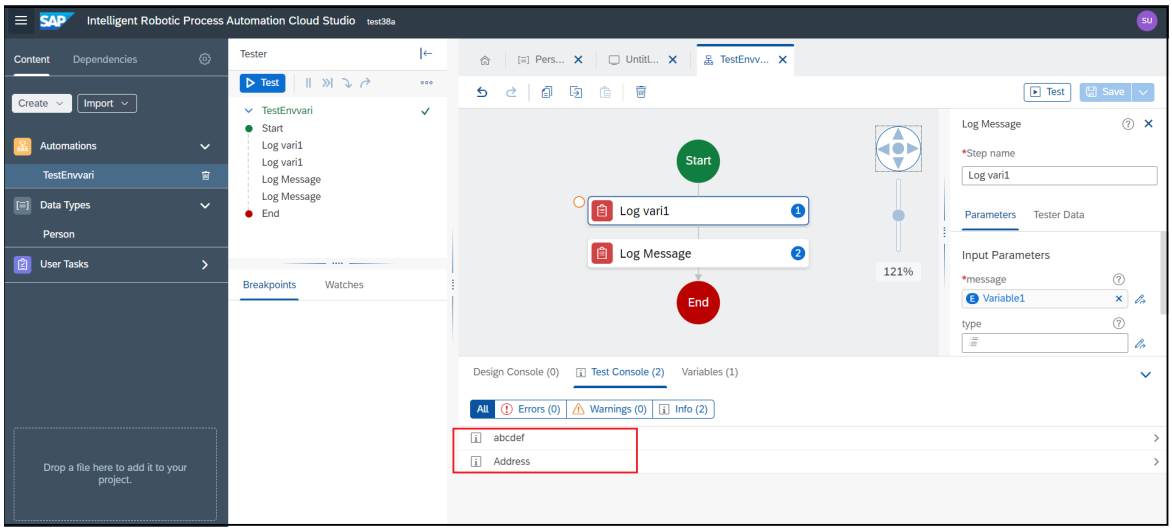


4. Enter the appropriate variable values in the required fields.

> **i Note**
>
> You can also provide the appropriate variable values in JSON format by enabling *Show JSON editor*.

> **i Note**
>
> You must provide the appropriate variable values at run time to complete the automation testing process. After providing the values, the *Test*. button is enabled.

5. Click *Test*.
   After successful execution of the automation, the variable values are displayed in the Test Console [page 267].

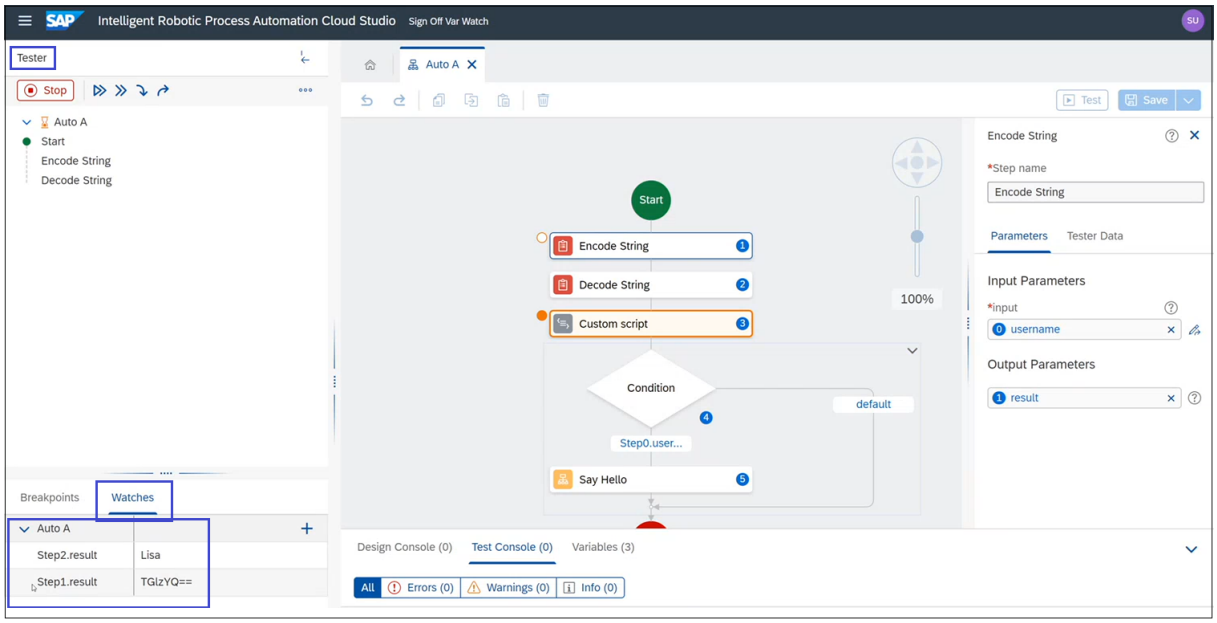


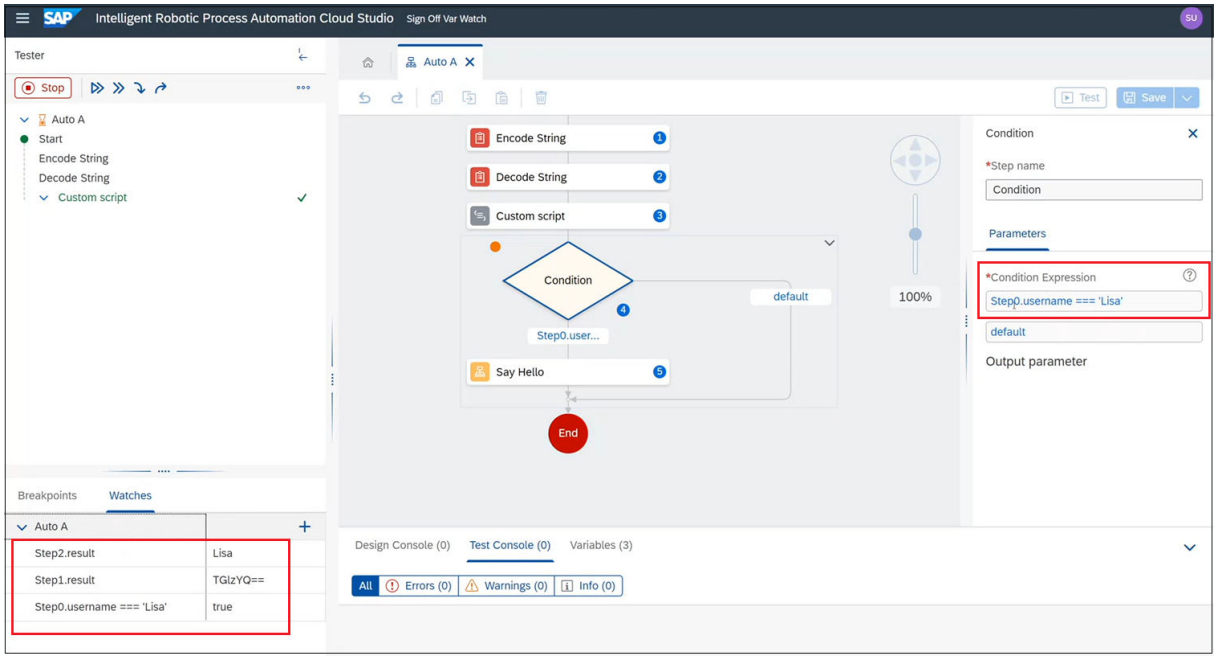
4.2.13.3 Watch Variable Value

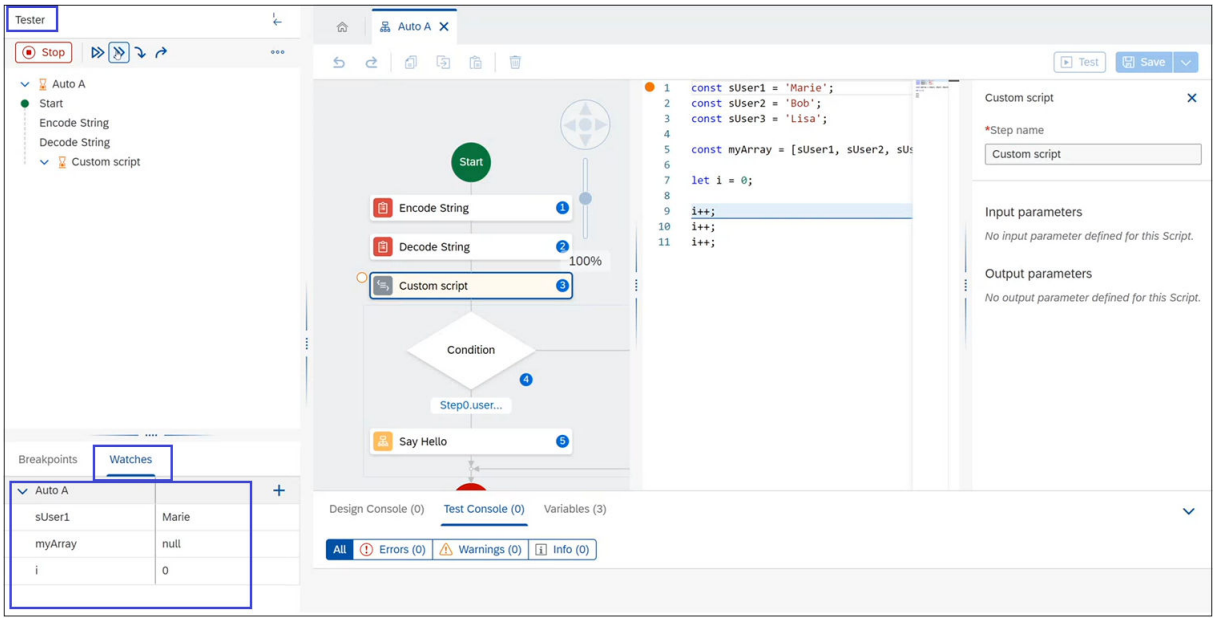
You can watch various variable values during testing of an automation in the *Watches* section of the *Tester* panel.

- You can watch the values of the variables used in an automation.
- You can watch the values of the expressions used in the custom script or formula editor of an automation.
- You can watch the values of the nested elements of a complex list output.
-

For example, refer to the following screenshots:







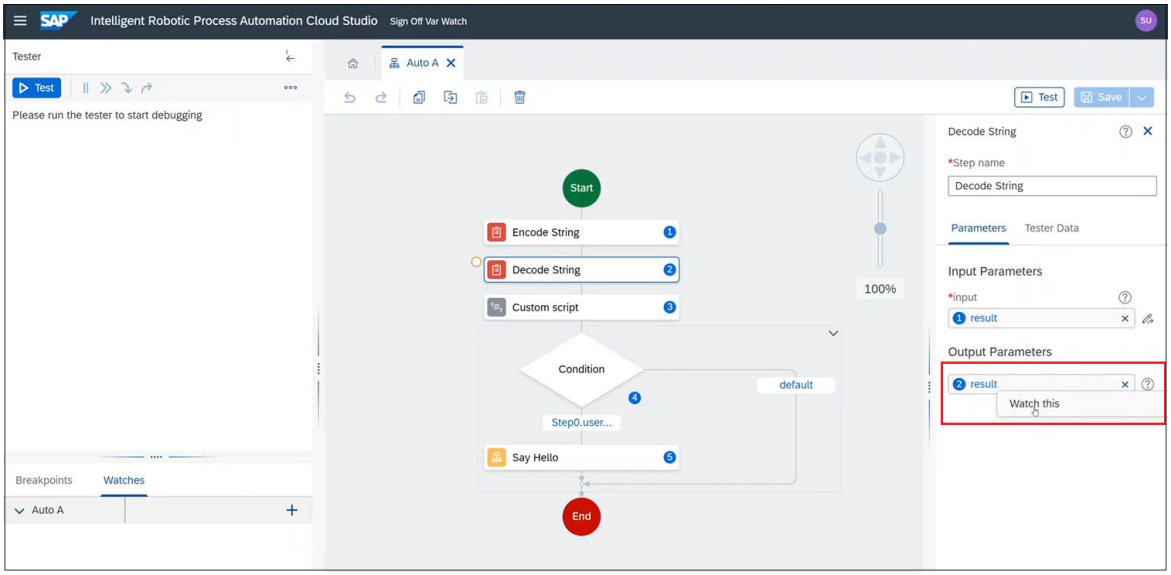
Procedure

To watch a variable value, follow the below procedure:

1. In a workflow of your automation, select the activity for which you want to watch the variable value.

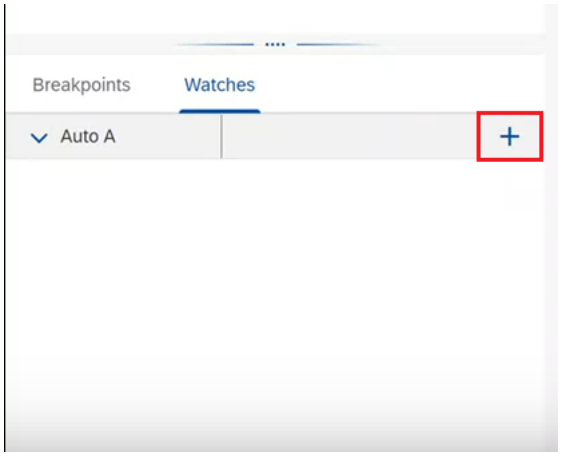
   > **Note**
   >
   > The activity must have an output variable.

2. Under *Output Parameters*, on the output parameter field right-click and then select *Watch this*.

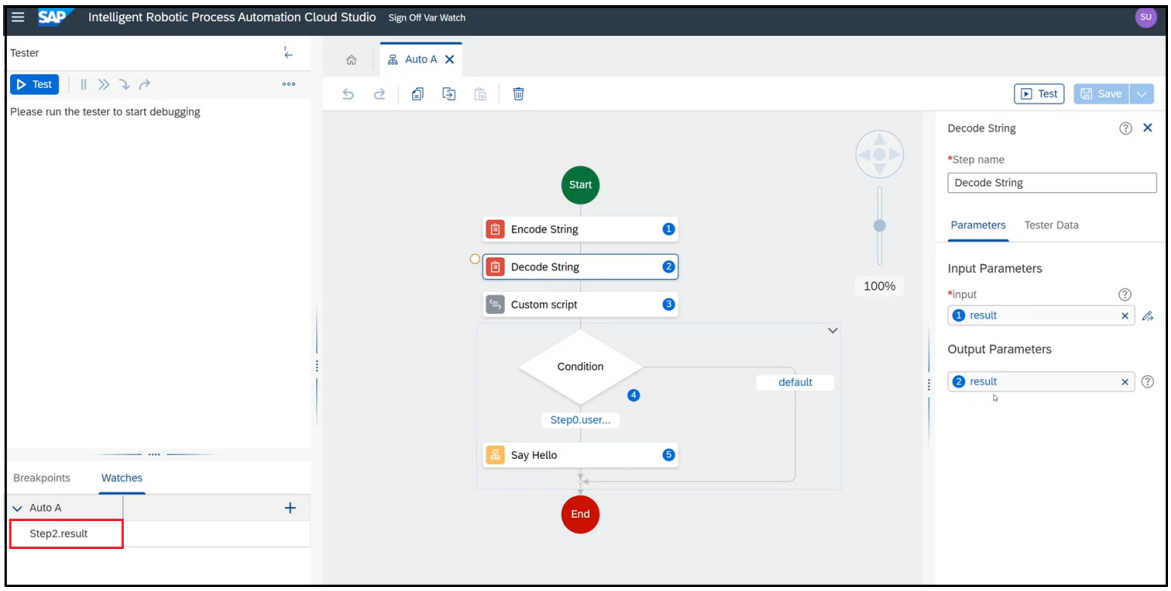


   OR

   Click the + icon.



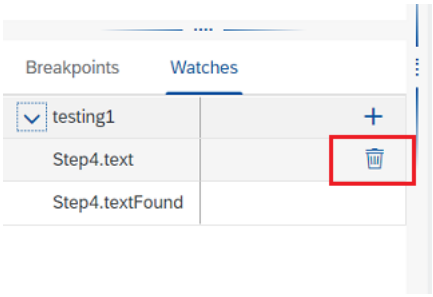
The *Edit Expression* dialog is displayed.
Select the appropriate variable and then *Save Expression*. For more details about editing an expression, refer to the Expression Editor [page 199] section.
To be watched variable is added to the *Watches* list.



3. Add a breakpoint. This makes the automation pause at the breakpoint.
   The breakpoint will help you to evaluate variable value during testing of an automation.
   For example, if you want to watch the variable values of the step 1 and step 2 of a workflow, then add the breakpoint at step 3.
   For more details about defining a breakpoint, refer to the steps 1 to 4, in the Test an Automation [page 282] section.
4. Test an automation to evaluate various variable values.
   For more details about testing an automation, refer to the Test an Automation [page 282] section.

Delete Watch Variable

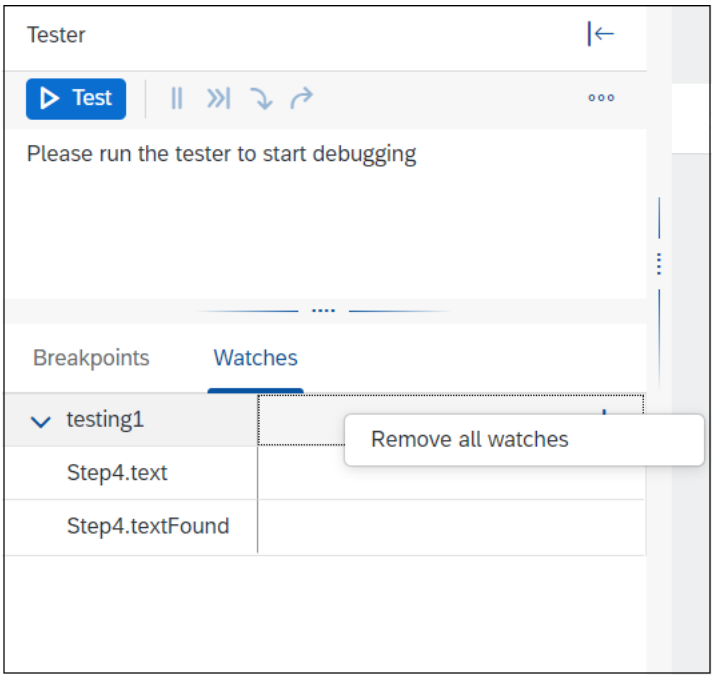
1. In the *Watches* list, select to be deleted variable.



2. Click the 🗑 icon.
   The variable is deleted from the *Watches* list.

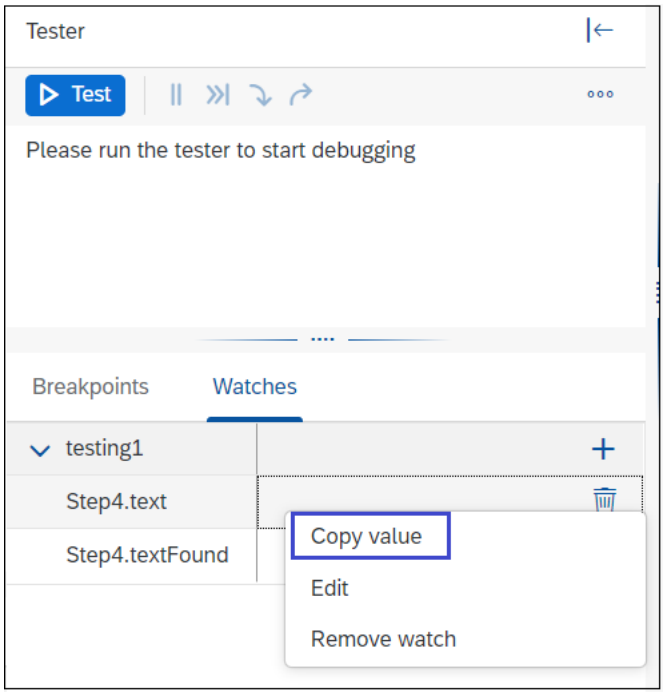
Remove All Watches

Use this option to remove all variables from the *Watches* list.



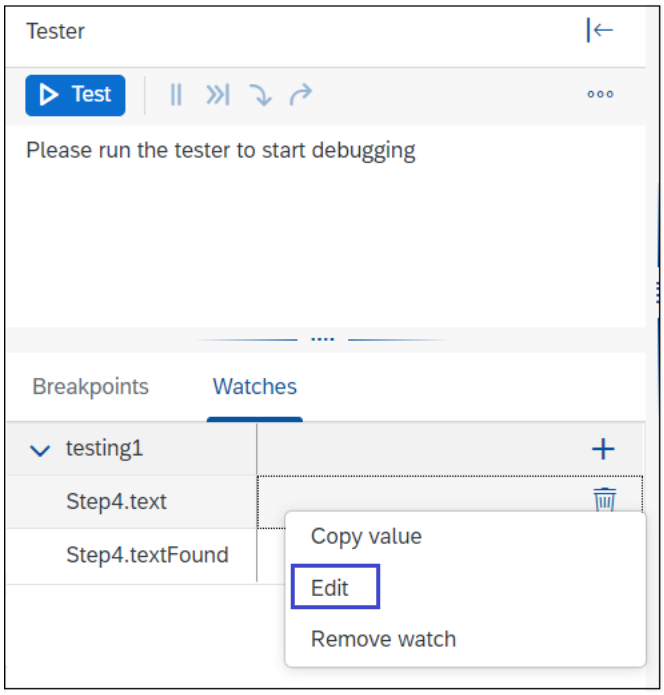
Copy Value

Use this option to copy the variable value.



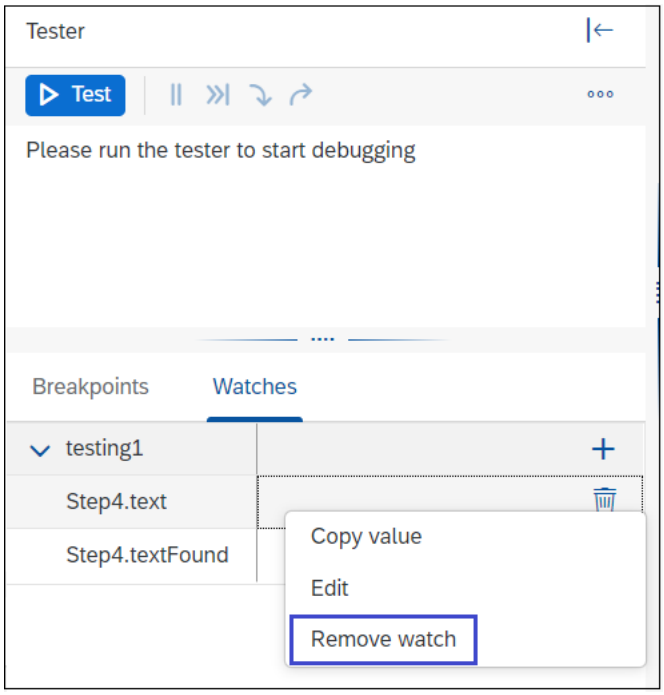
Edit

Use this option to edit the variable value.



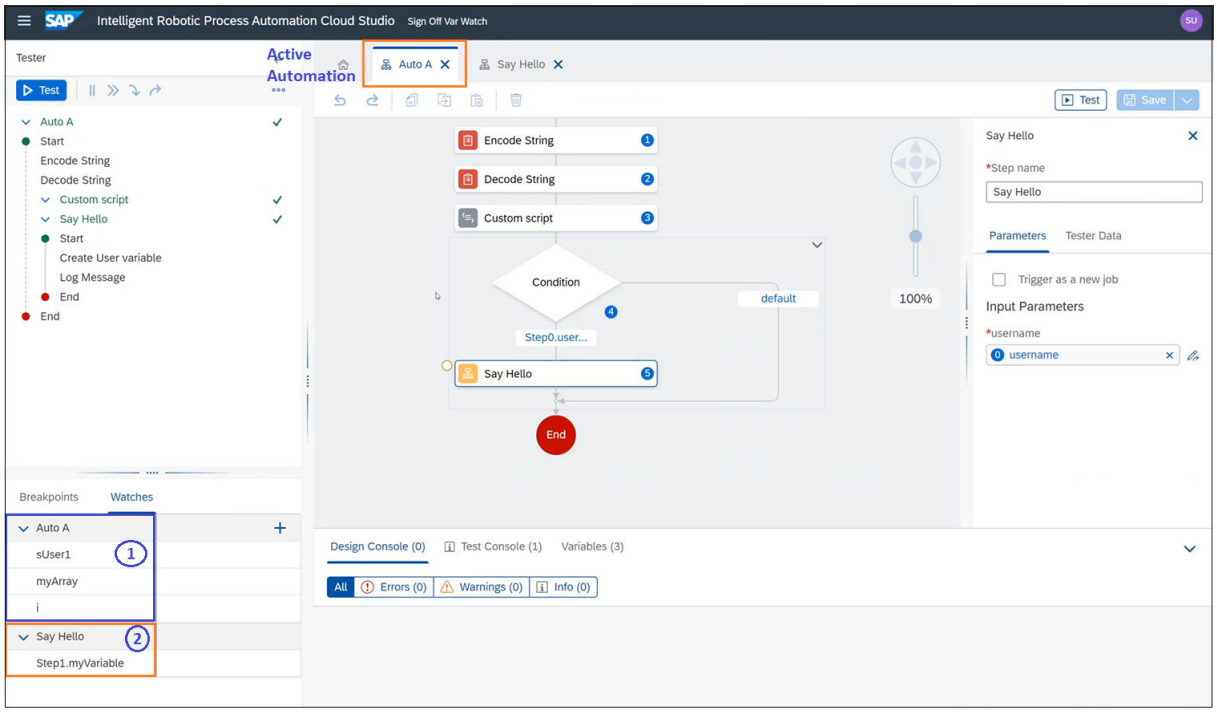
Remove Watch

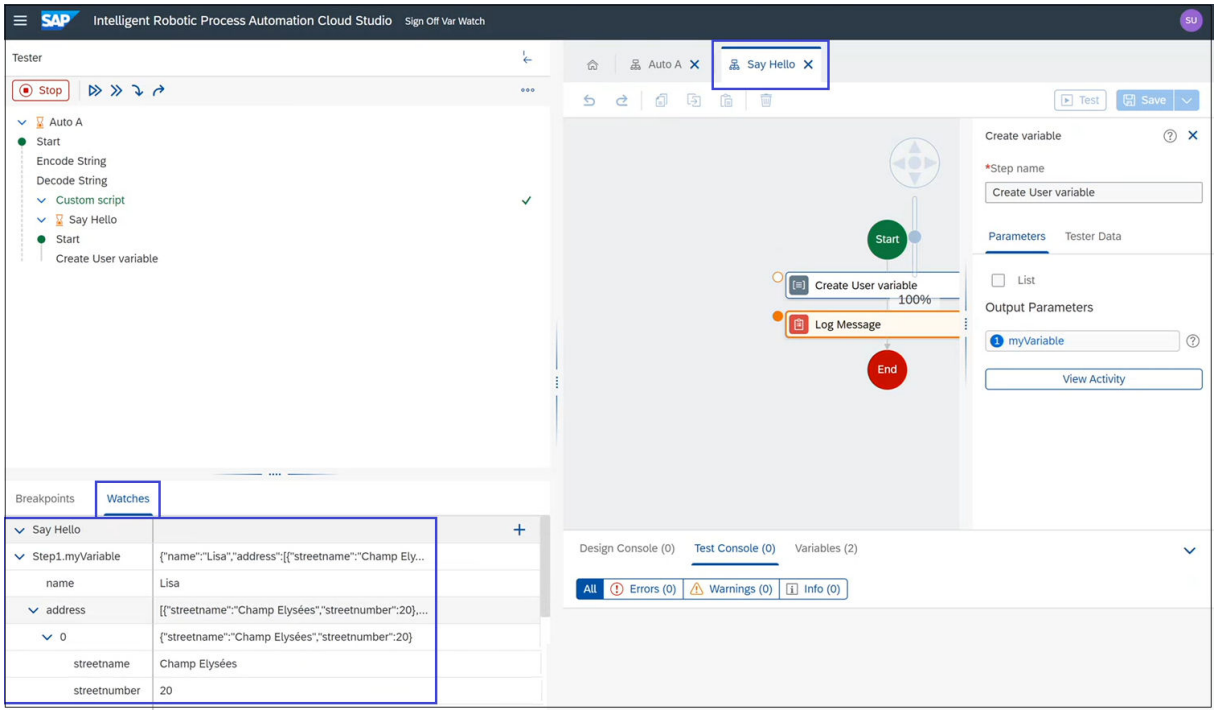
Use this option to remove specific variable from the *Watches* list.



Multiple Automation Watch Variables

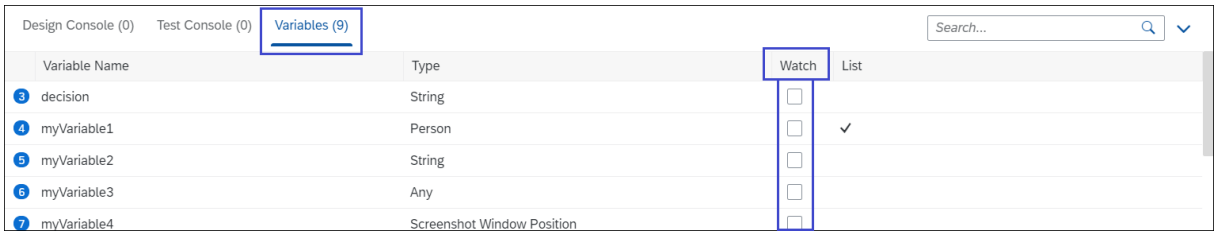
Active automation watch variables shown at the top of the *Watches* list. We can access only the currently executing automation variables.



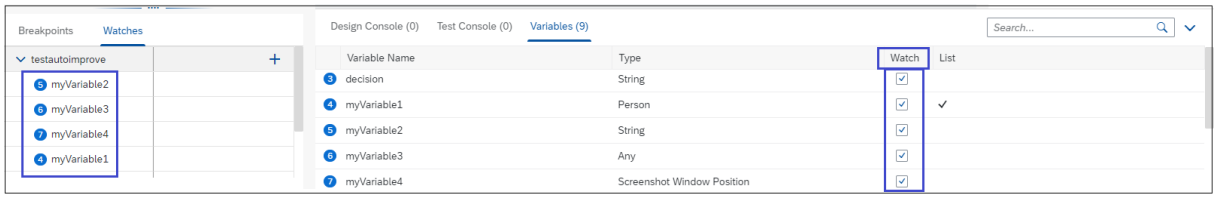


Add Variables to Watches

You can add variables of your automation to *Watches* by selecting the check box of a varibale under the *Watch* column of the *Variables* tab.



After you select a variable, the selected variable is added under *Watches*.

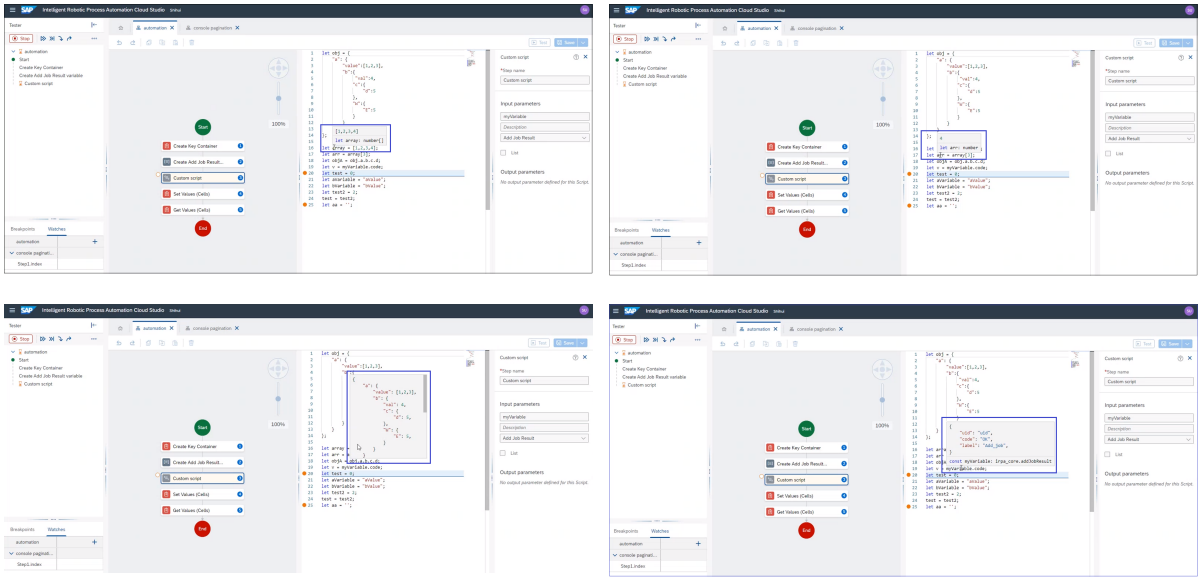


> **Note**
>
> If do not want to watch a variable, then deselect the check box of the variable.

4.2.13.3.1 Watch Variable Value in Custom Script

You can only watch a variable value on mouse over in a custom script of an automation.

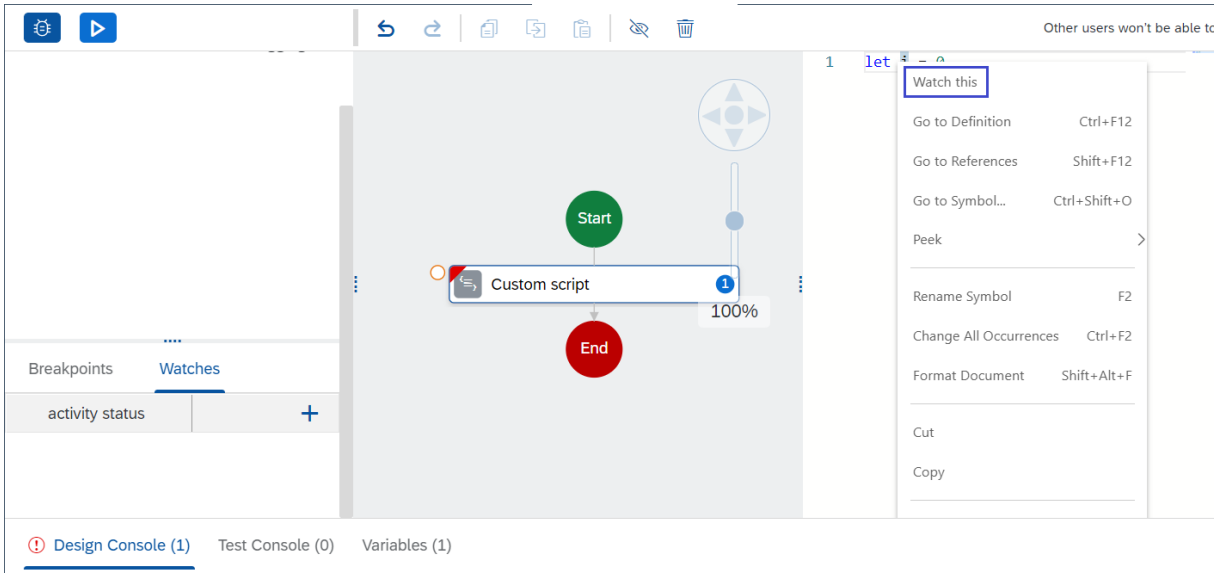
For example, refer to the following screenshots.



4.2.13.3.1.1 Add or Remove Variable from Watch in Custom Script

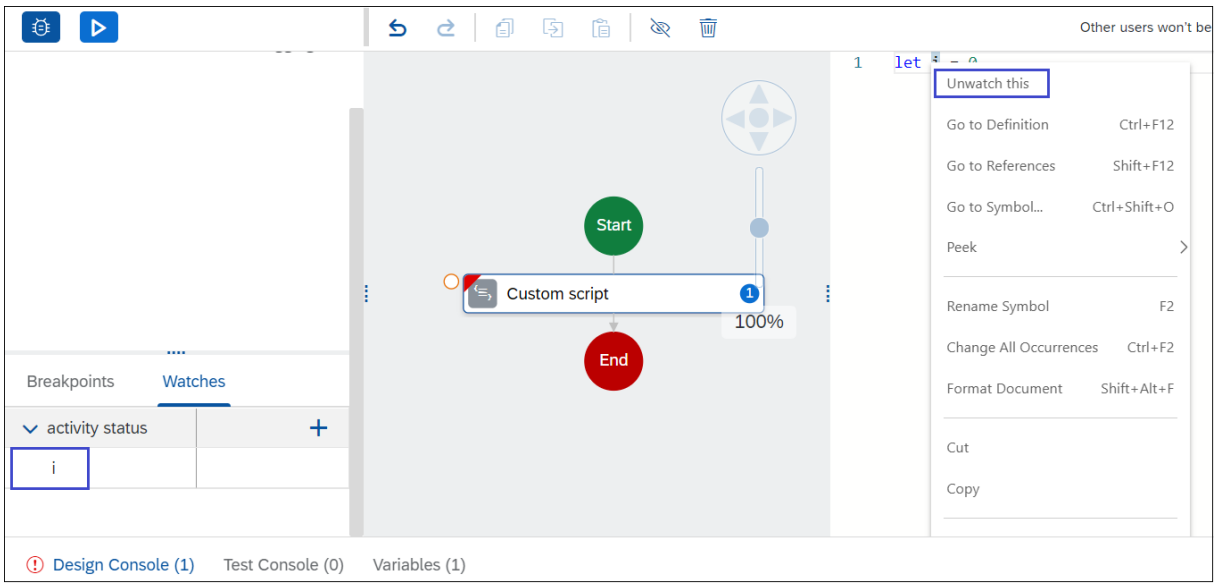
You can add or remove a variable value from Watch in the custom script.

Select the variable value in the custom script and right-click. The *Watch this* option is displayed.



Click *Watch this* to add a variable to the watch list.

Select the variable value in the custom script and right-click. The *Unwatch this* option is displayed.

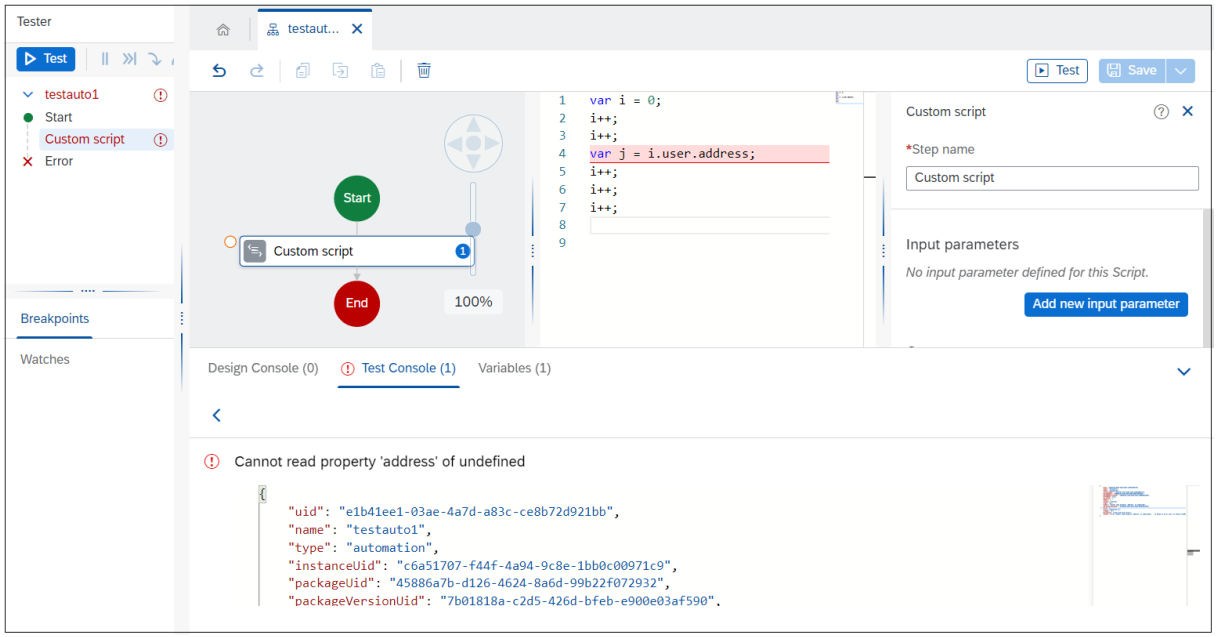


Click *Unwatch this* to remove a variable from the watch list.

4.2.13.4 Custom Script Error Containing Line

You can view the error containing line of a custom script by clicking an activity executed within a custom script from the timeline.

For example, refer to the following screenshot.



Related Topics

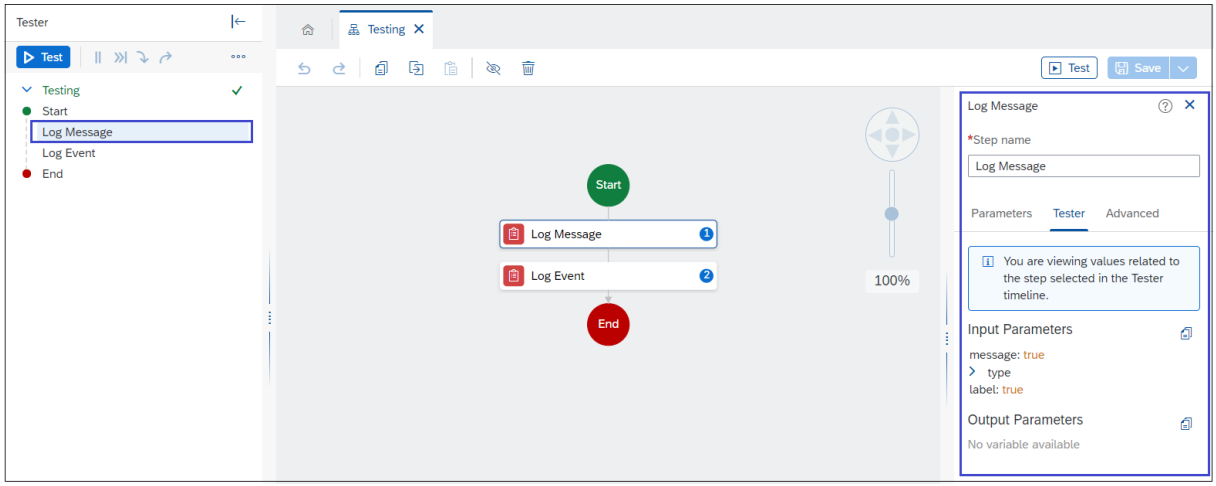
Custom Scripting [page 251]

4.2.13.5 Timeline

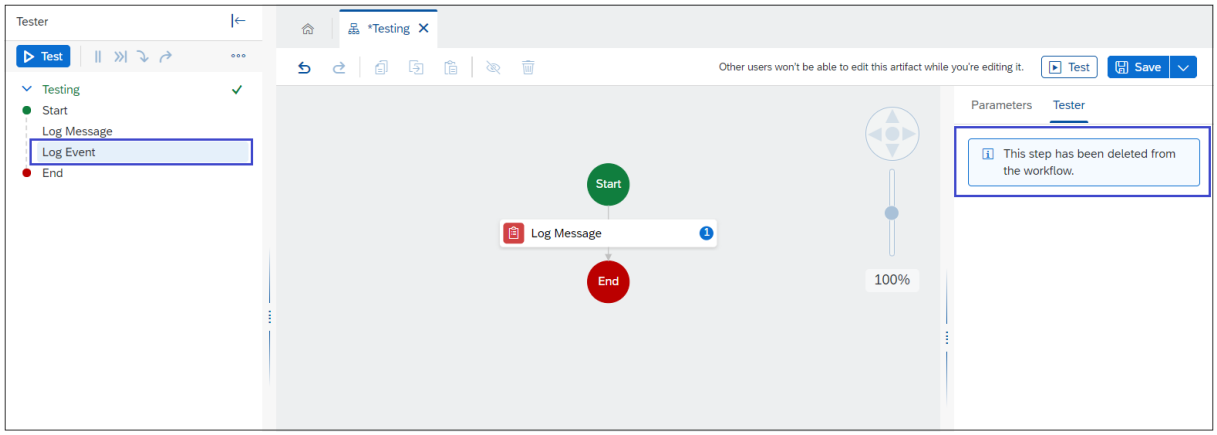
The user interface of the tester panel in Cloud Studio that shows the event logs of an automation.

Workflow Step Details

The workflow step you select in the timeline, according to it the left panel displays the details.

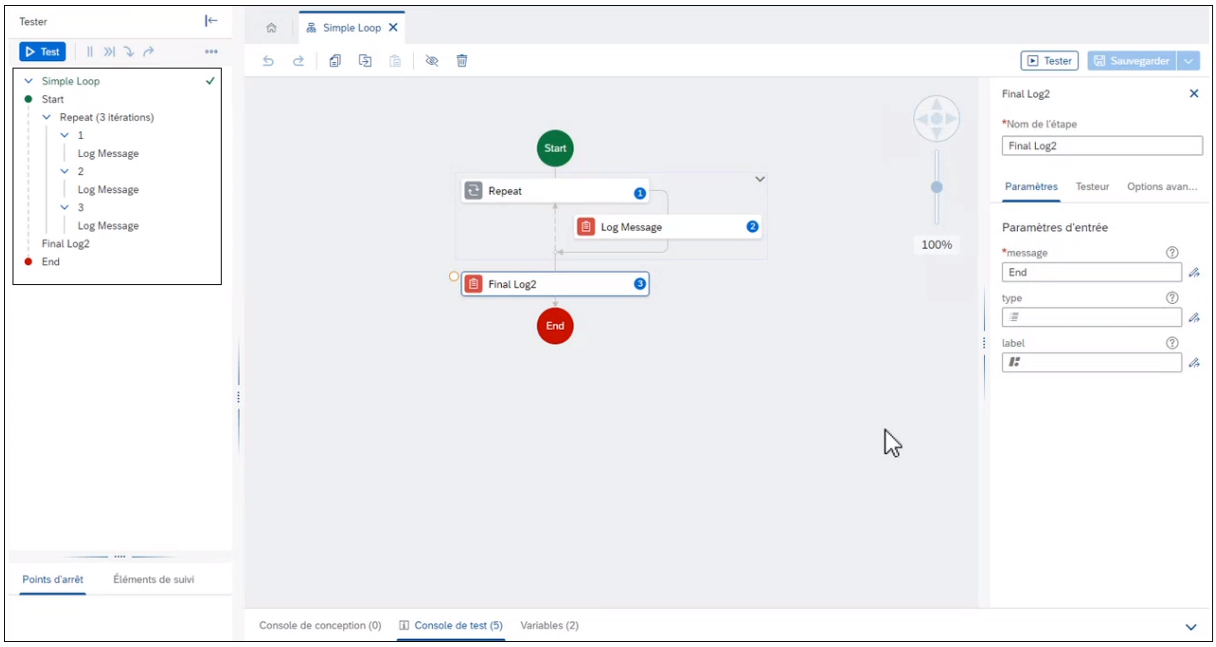


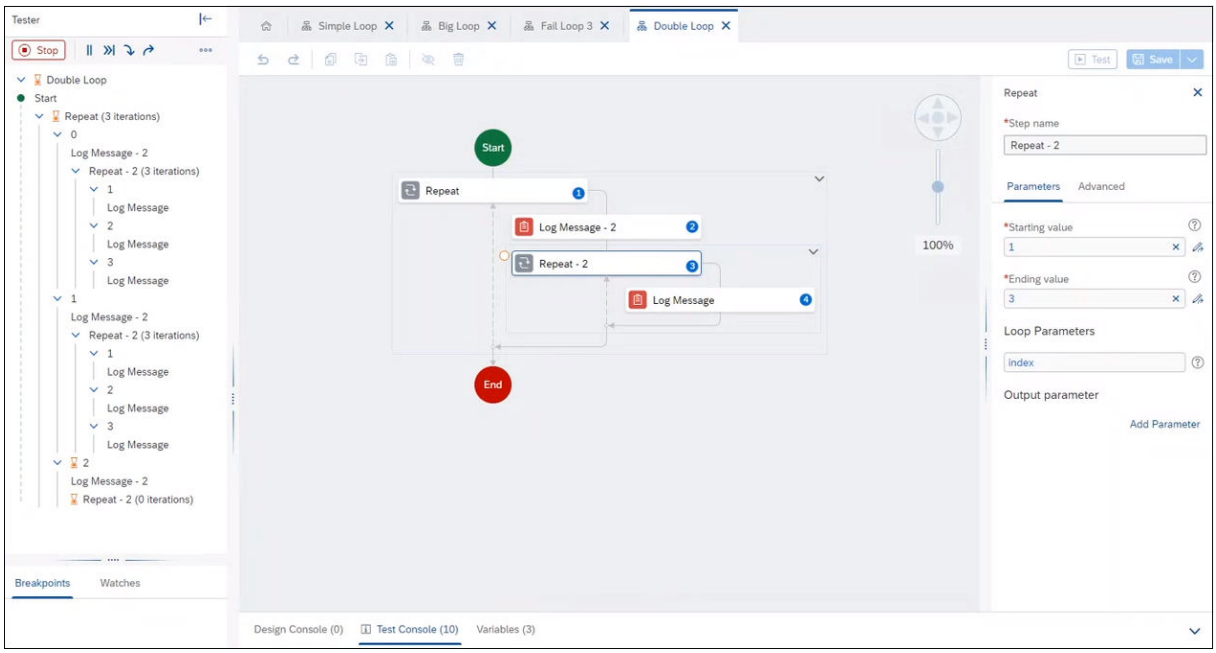
If the selected step is deleted, then an alert message is displayed in the left panel.

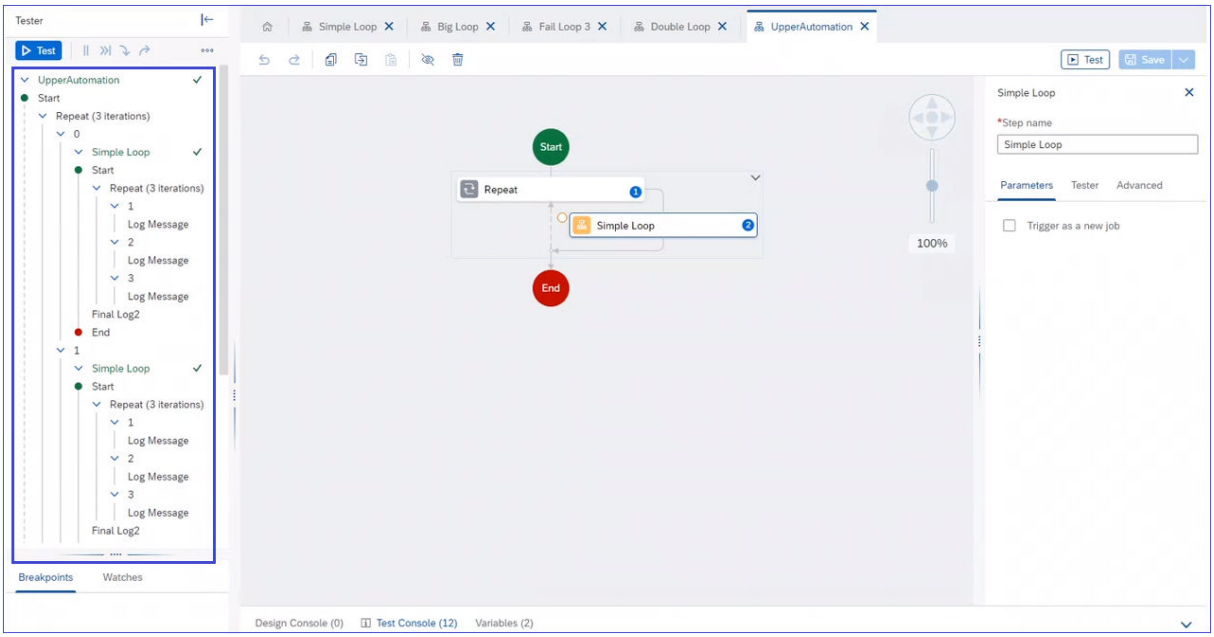


Display Loop Execution in Timeline

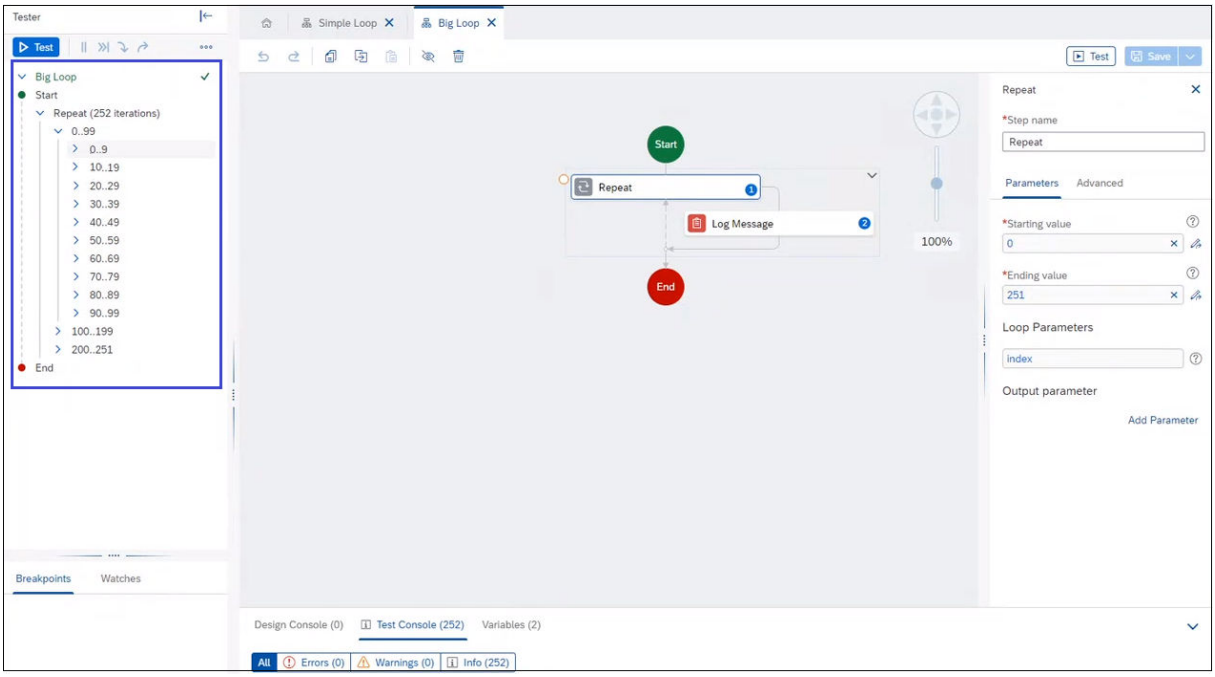
You can view loop execution status in the timeline. The loop iterations are segregated and then displayed. It allows to collapse and expand the set of iterations.

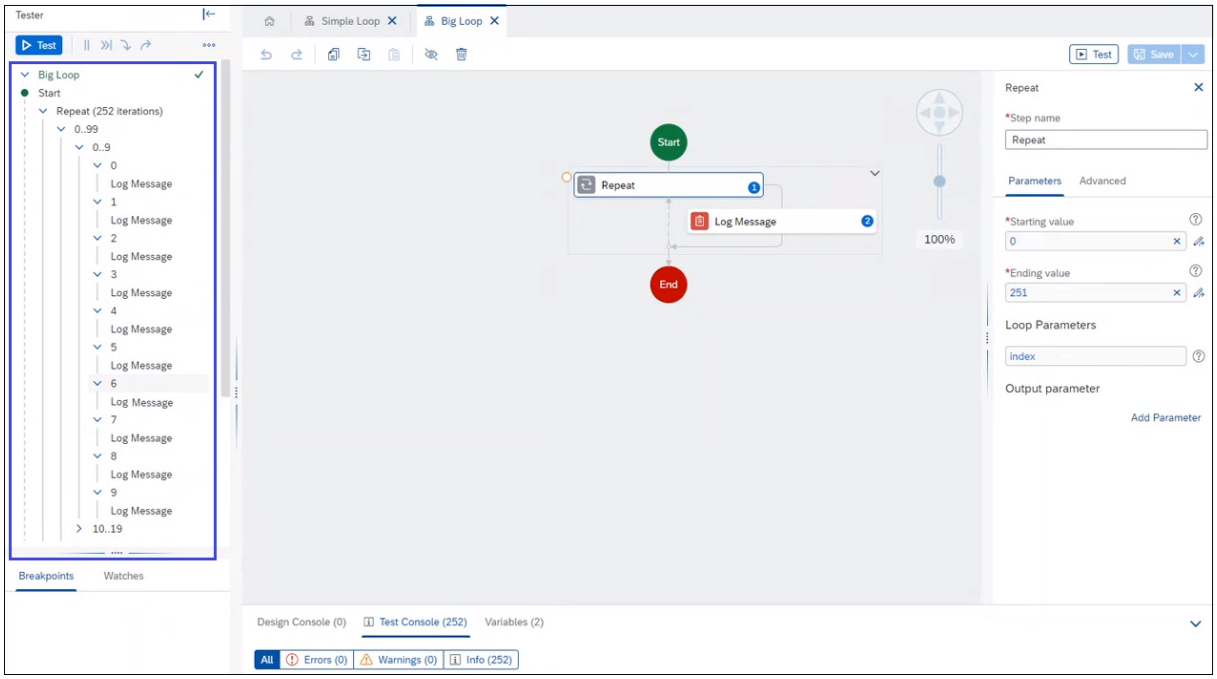






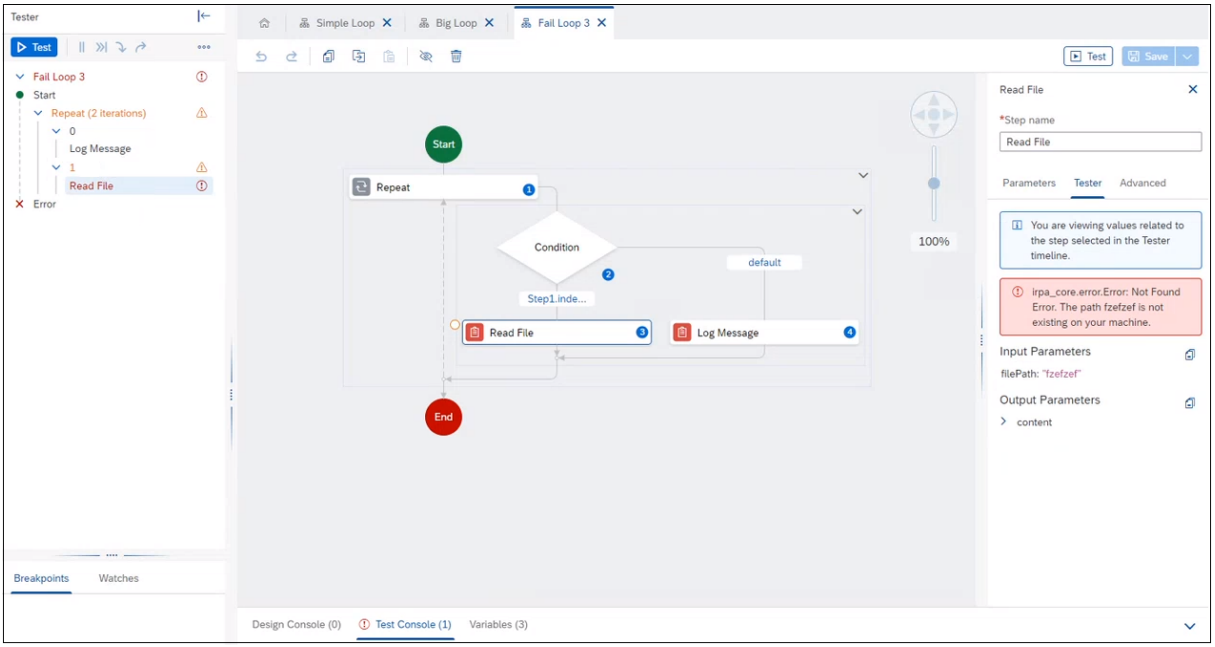
For larger loop execution, the iterations are segregated in various levels. For example, the following screenshots show that the first level contains 100 iterations (0 to 99) and the second level contains 10 iterations (0-9).





Error in Loop Execution

If an error occurs while executing a loop, then it is highlighted in Red colour.



The selected iteration error detail is displayed in the left panel.

4.2.14 Manage Errors within an Automation

In the Cloud Studio, the following types of errors are raised in your automation:

- errors predefined by the different SDK activities:
  - in the SAP Intelligent RPA Core SDK package
  - in theSAP Intelligent RPA Excel SDK package
  - in the SAP Intelligent RPA Outlook SDK package
  - in the SAP Intelligent RPA SAP WebGUI SDK package
- business errors raised by the user

At execution time, uncaught errors will automatically fail your automation. As an automaton designer, to avoid failed executions, you must be able to anticipate possible errors and manage them properly. The following topics explain and demonstrate the methods to manage the errors in your automation.

Error Management Tutorial [page 309]
In this tutorial, you will go through the different steps you need to follow, to manage errors in an automation in the Cloud Studio.

On Error Handling Workflow [page 331]
If an error is raised in your automation, the *On error* node allows you to continue the exeuction of your automation by adding activities on a side workflow. These activities will be executed after the main workflow fails.

Generate a Business Error with the Stop Automation in Error control [page 336]
The **Stop Automation in Error** control allows you to raise a business error that generates a customized input message in the logs, when your automation is failing.

Display List of Errors Associated with Activity [page 339]
In an automation, you can view the list of possible errors associated with an activity.

Display Errors in the Timeline [page 340]
In the Cloud Studio, when you test your automation, the errors are displayed on the tester side panel in the timeline.

4.2.14.1 Error Management Tutorial

In this tutorial, you will go through the different steps you need to follow, to manage errors in an automation in the Cloud Studio.

For this use case example, you will create a local folder and a local file on your machine. Depending on the location of the local folder, SAP Intelligent RPA may limit authorization rights and prevent the creation of the file, which will result in an error.

To manage the errors, you will learn how to use the **On Error** node and the **Try/Catch** control.

Prerequisites

Before you work through this tutorial, make sure an agent has been properly connected to your tenant. Otherwise, you will not be able to complete the tutorial. For more information, see the Desktop Agent Tenant Registration in the Installation Guide.

Make sure you have created an environment and added your agent to this environment. For more information, see Create and Manage Environments in the Factory User Guide.

Create your Automation [page 310]

Create a File [page 312]

Manage an Error with the On Error Node [page 315]

Manage an Error with Try/Catch [page 318]

Catch the Error with Try/Catch [page 326]

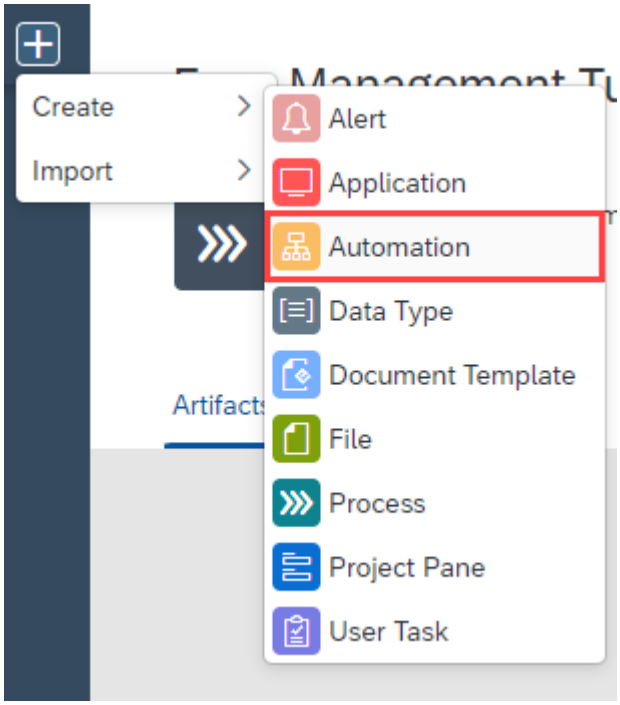
4.2.14.1.1 Create your Automation

Context

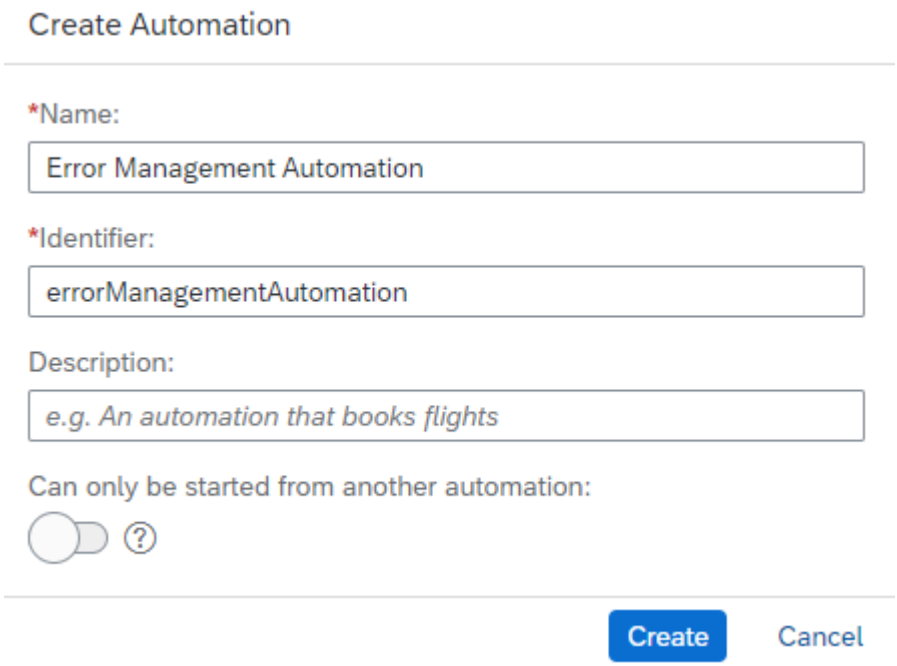
In this section you will create your automation and choose a local path to create a folder on your machine.

Procedure

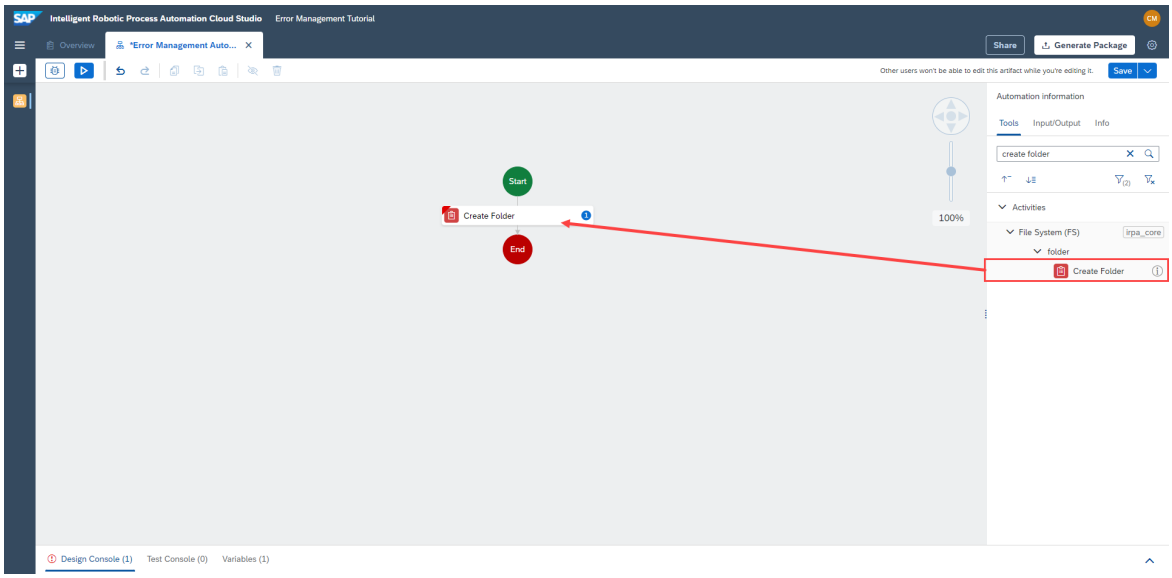
1. Create your automation by clicking the + icon, and selecting *Create* and then *Automation*.



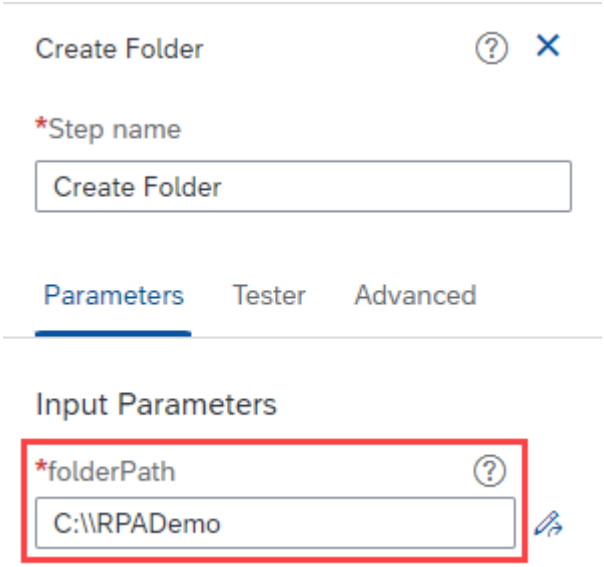
2. Configure your agent version and create your automation.



3. In the search bar of the *Automation information* side panel, enter **`Create folder`**. Drag and drop the *Create Folder* activity in the workflow of your automation.



4. Select the *Create Folder* activity. On the *Create Folder* side panel, enter **`C:\\RPADemo`** in the *folderPath* field.



5. Save and test your automation. A folder named **RPADemo** is created at the C: level on your computer.

Next Steps

Next you need to create a file with an error in your automation.

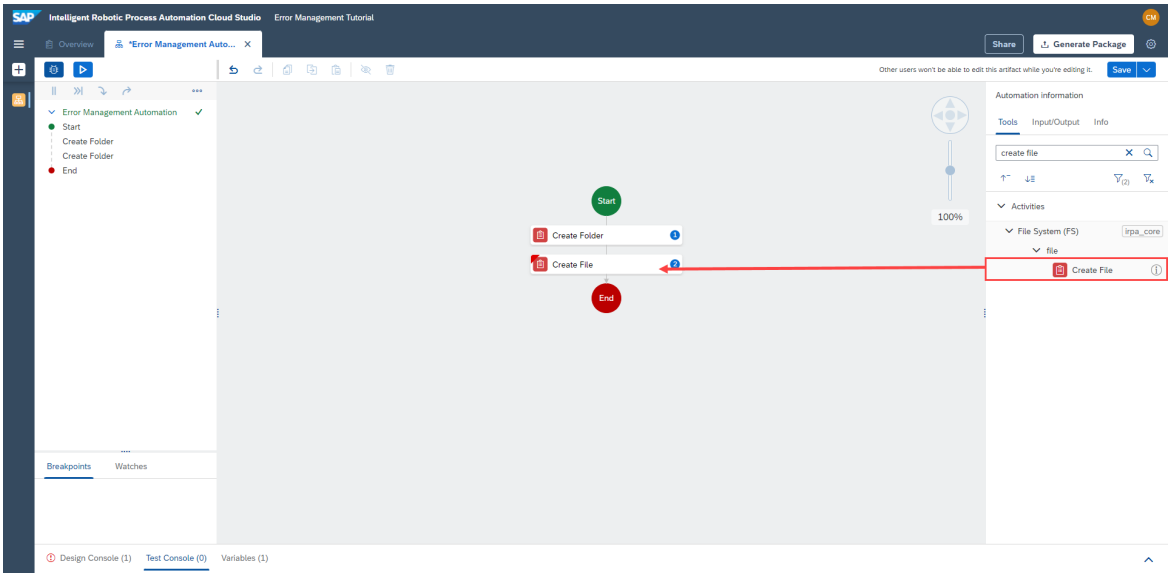
4.2.14.1.2 Create a File

Context

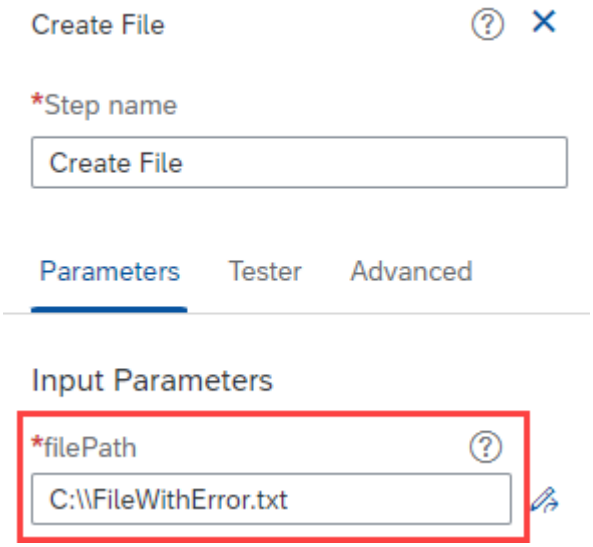
In this section you will add an activity to create a file in the local folder.

Procedure

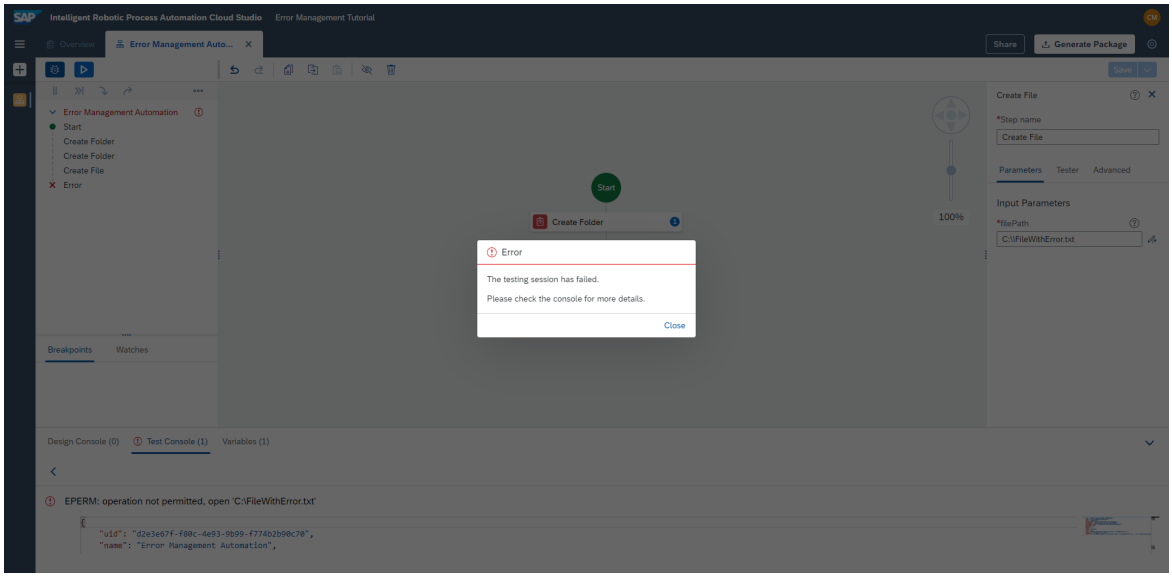
1. In the search bar of the *Automation information* side panel, enter `Create file` and drag and drop the *Create File* activity in the workflow of your automation.



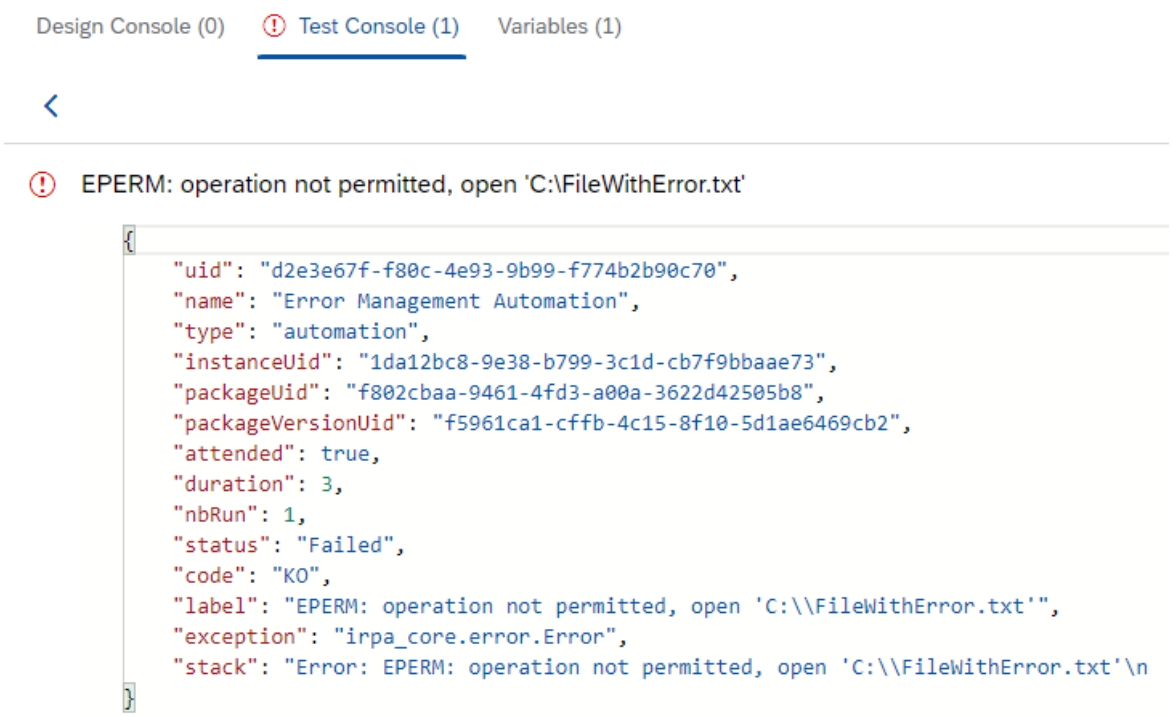
2. Select the *Create File* activity. On the *Create File* side panel, enter `C:\\FileWithError.txt` in the *filePath* field to create an error.



3. Save and test your automation. The testing session fails.



> ⚠ Caution
>
> Due to security concerns, you cannot create a document at this level.



You can review the test data in the *Test Console* tab.

Next Steps

Next, you need to manage the error that have been raised in your automation.

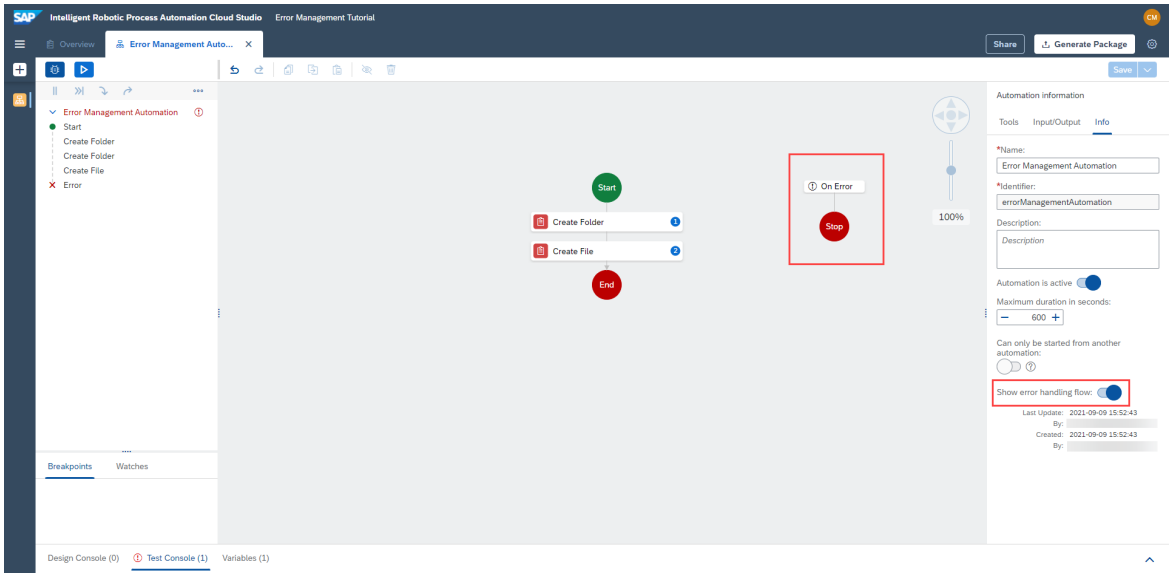
4.2.14.1.3 Manage an Error with the On Error Node

Context

In this section, you will manage an Error with the *On Error* node.

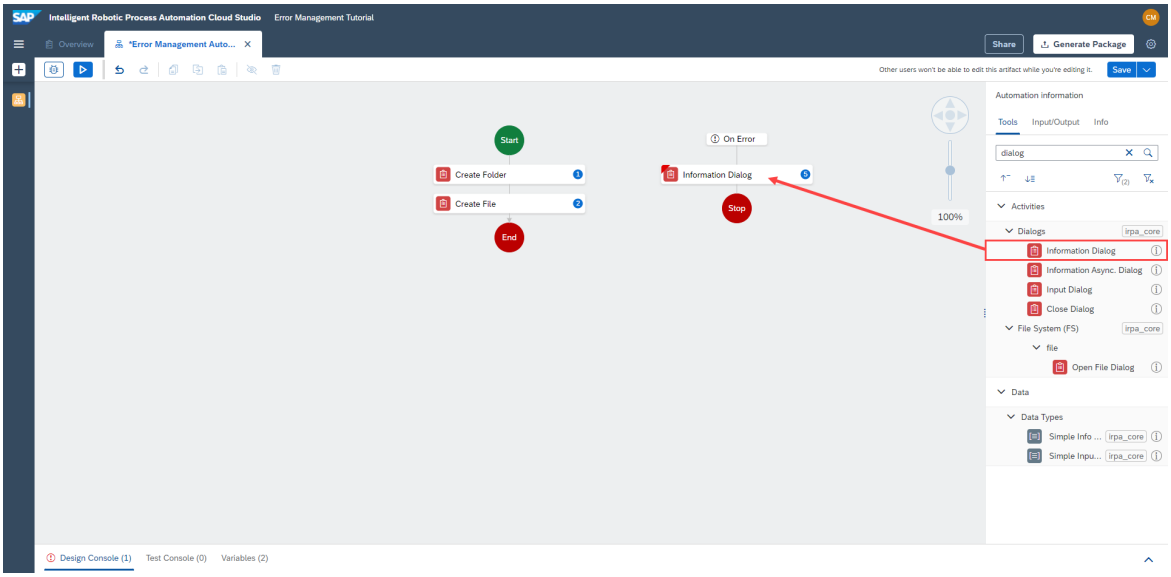
Procedure

1. On the *Automation information* side panel, click the *Info* tab and switch the *Show error handling flow* button on.

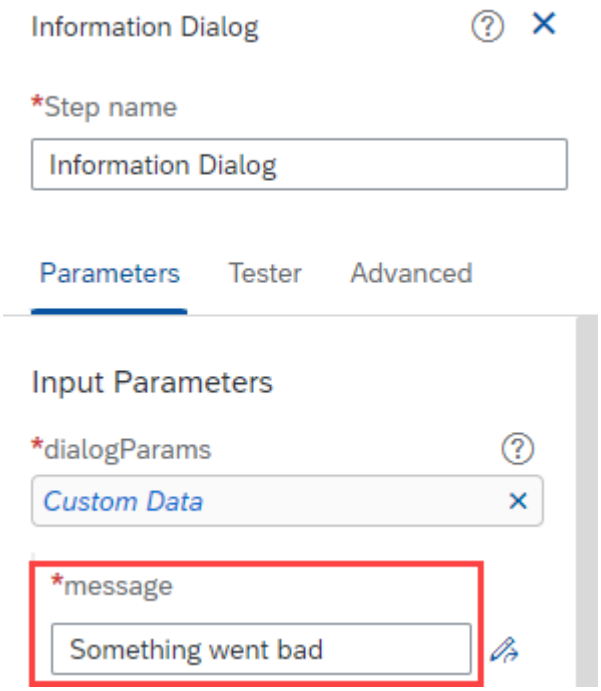


   The *On Error* node appears in the workflow.

2. In the search bar of the *Automation information* side panel, enter `Dialog` and drag and drop the *Information Dialog* activity on the *On Error* node.

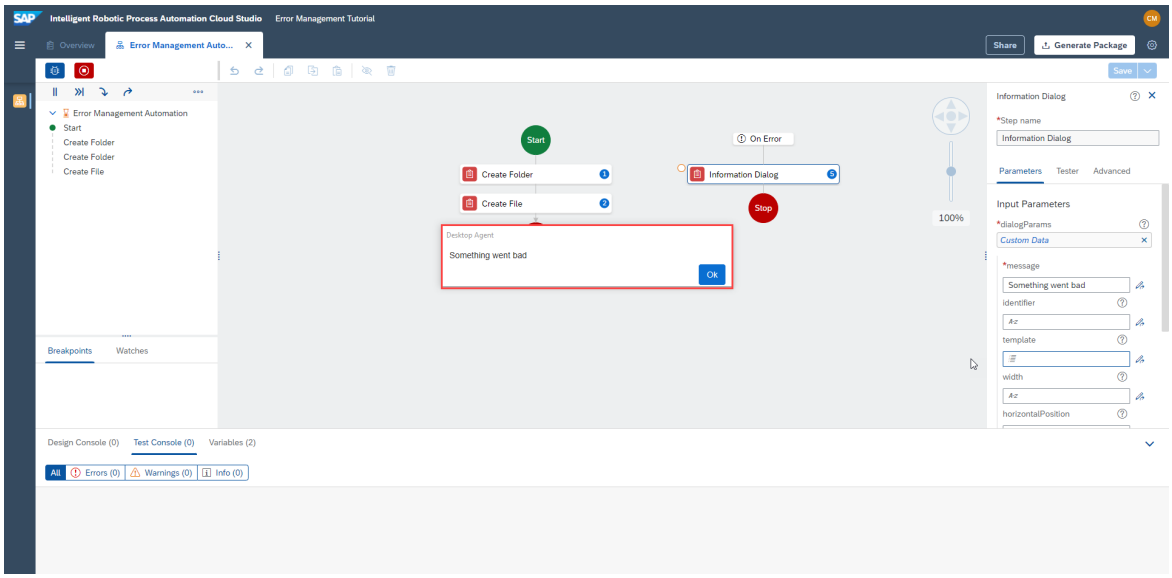


3. In the Input Parameters of the activity, select *Create Custom Data* in *dialogParams*.
4. Enter **`Something went bad`** in the *message* field and save your automation.

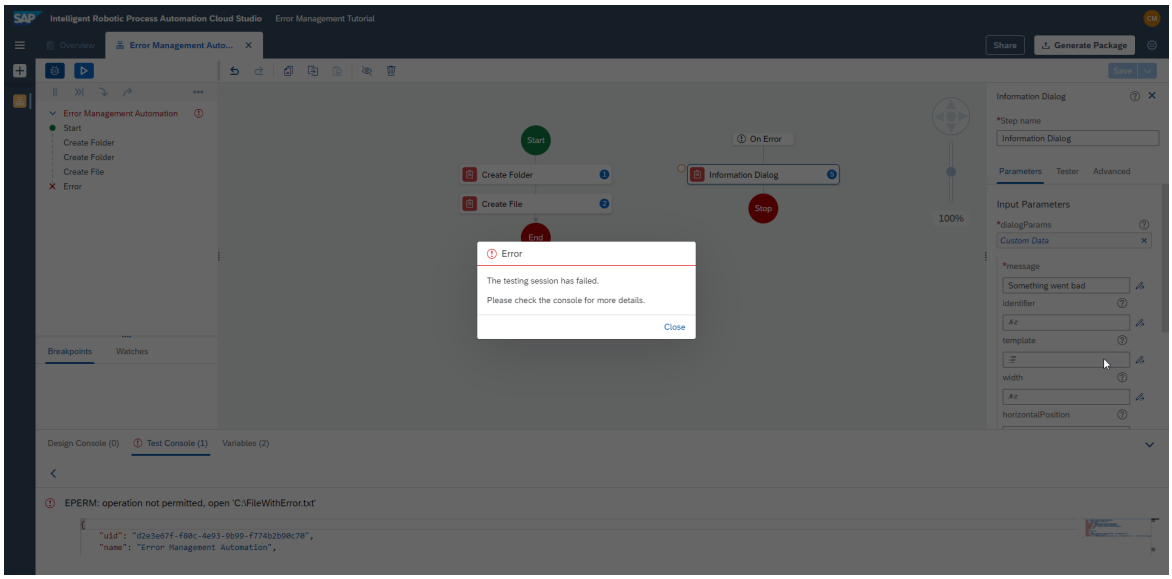


5. Save and test your automation.

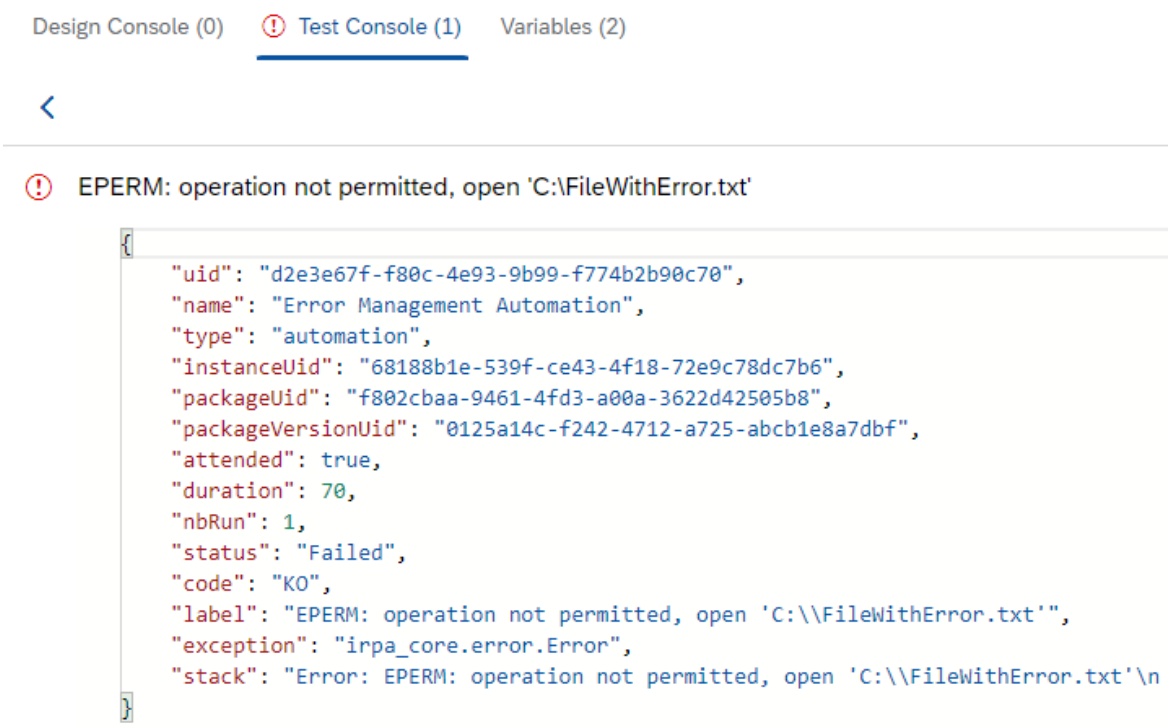
The following message box appears:



6. Click *Ok*. The testing session fails.



You can review the test data in the *Test Console* tab.



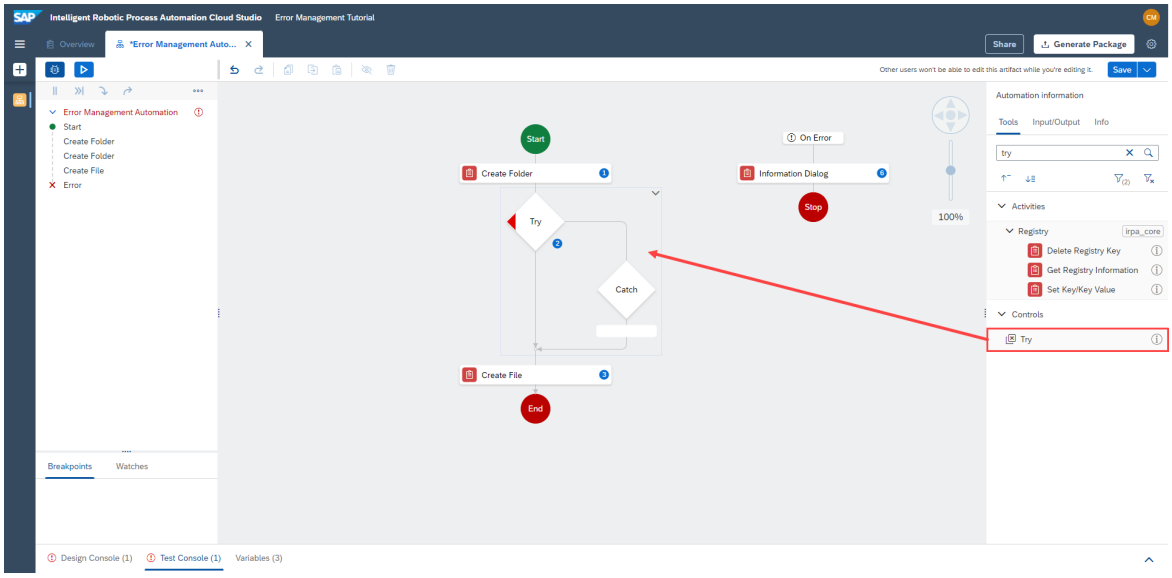
4.2.14.1.4 Manage an Error with Try/Catch

Context

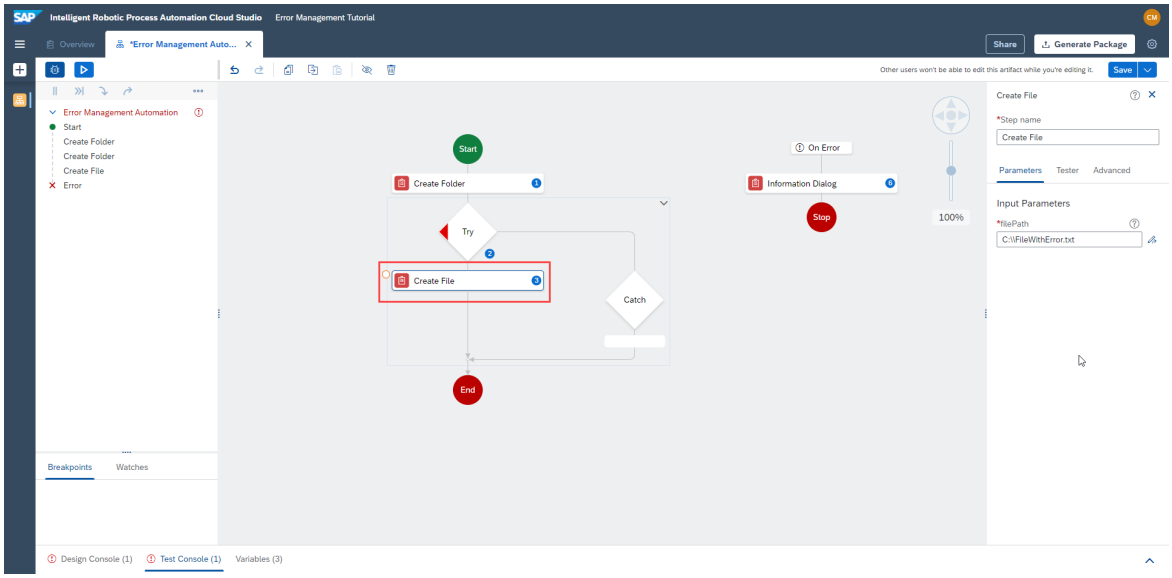
In this section, you will manage an Error with the *Try/Catch* control.

Procedure

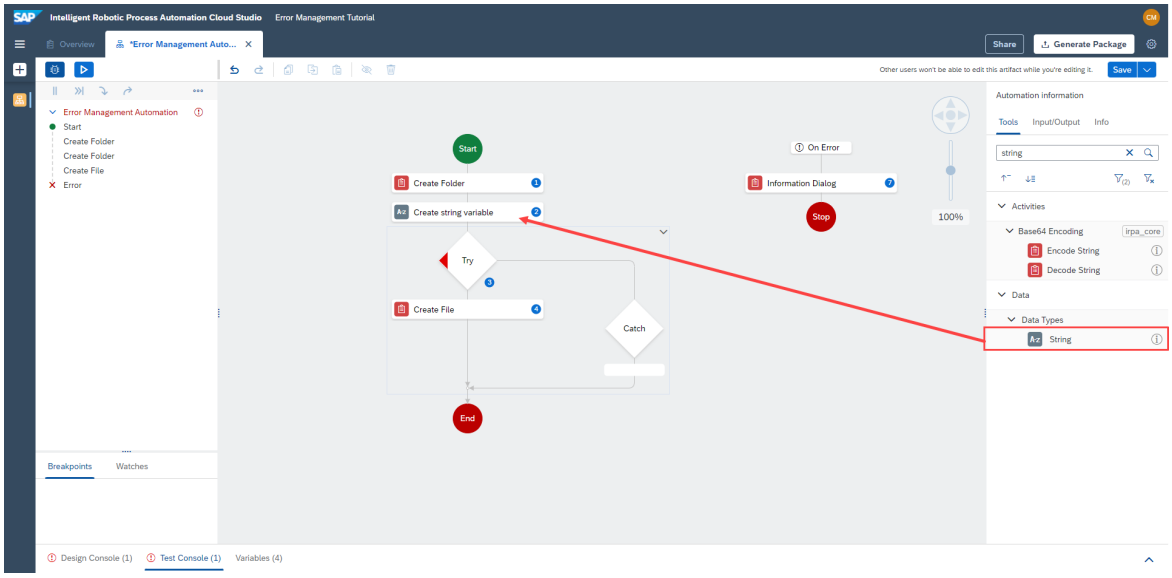
1. In the search bar of the *Automation information* side panel, enter **`Try`**. Drag and drop the *Try* control after the *Create Folder* activity in the main workflow.



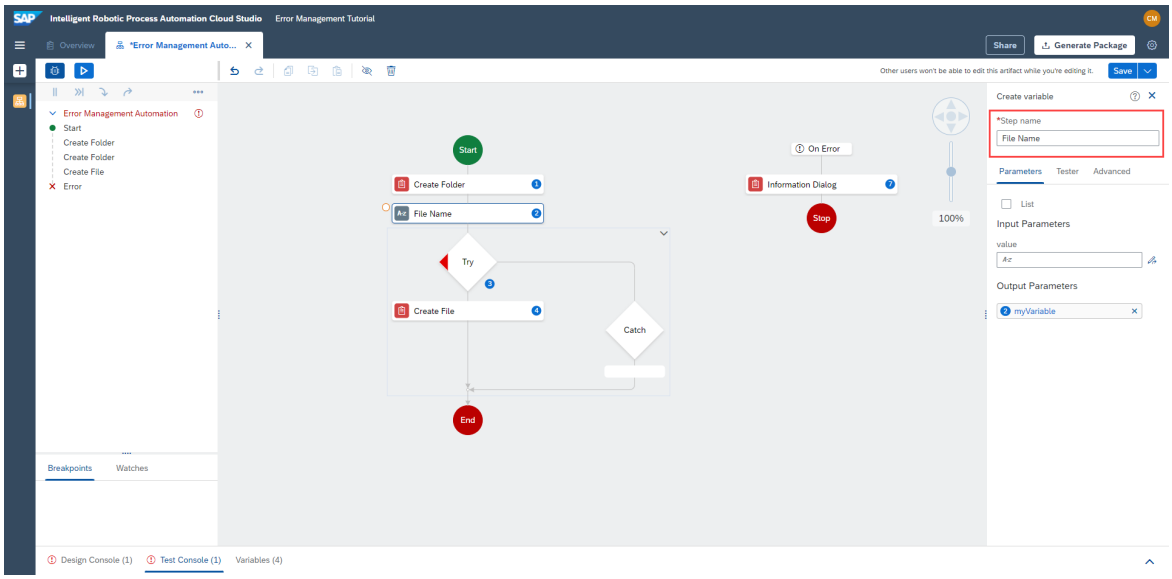
2. Drag and drop the *Create File* activity on the *Try* branch.



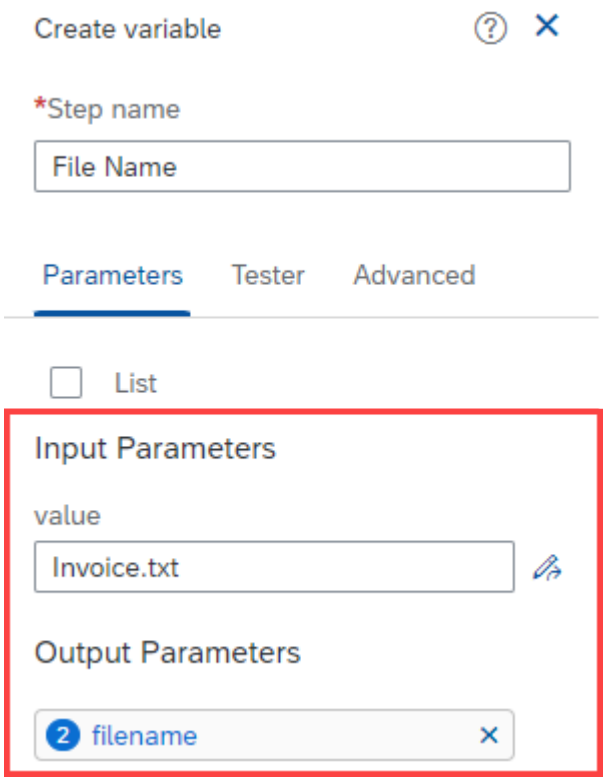
3. In the search bar of the *Automation information* side panel, enter **`String`**. Drag and drop the *String* data type after the *Create Folder* activity.



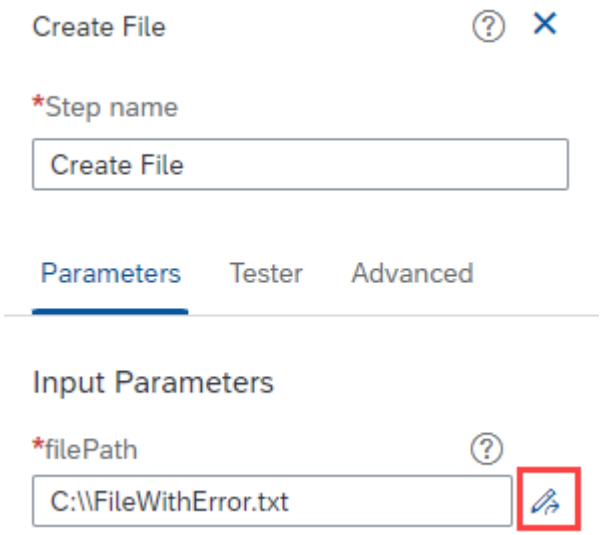
4. Select *Create string variable*. On the *Create variable* panel, enter **`File Name`** in the *Step name* field.



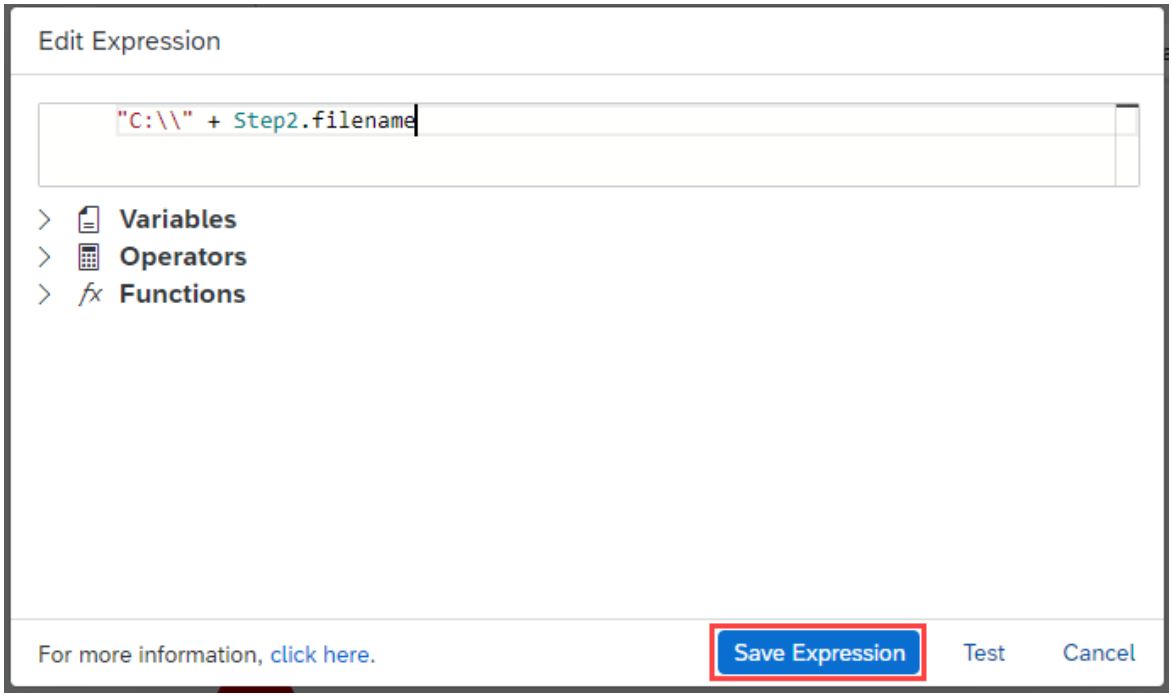
5. Enter your file name in the *Value* field. Here we choose to name the file **`Invoice.txt`**.
6. Enter **`filename`** in the *Output Parameters* field.



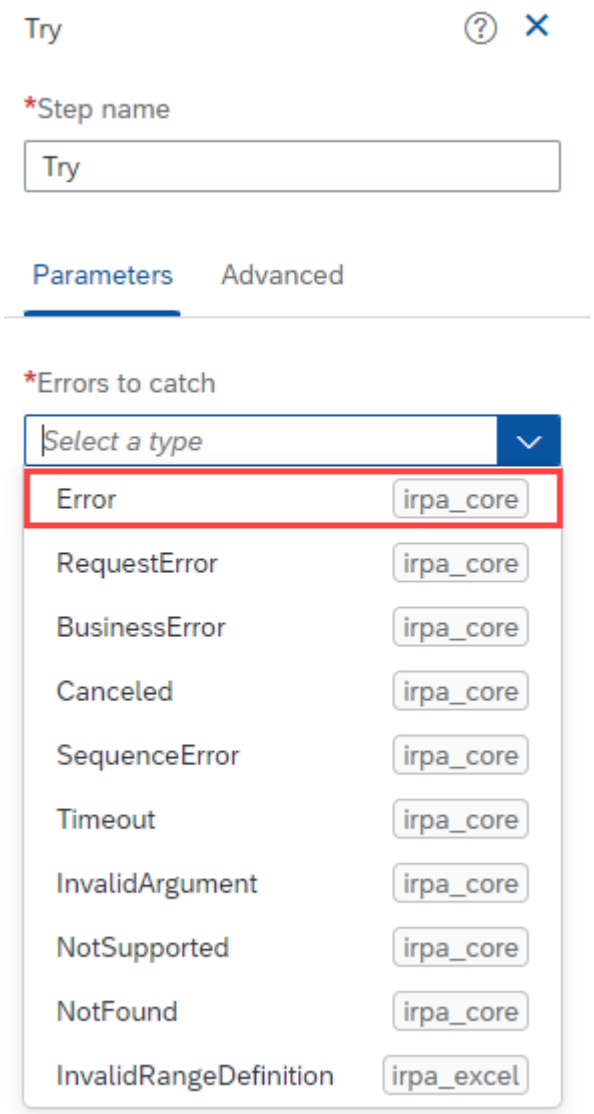
7. Select the *Create File* activity and click the button on the right side of the *filePath* field. The *Expression Editor* opens.



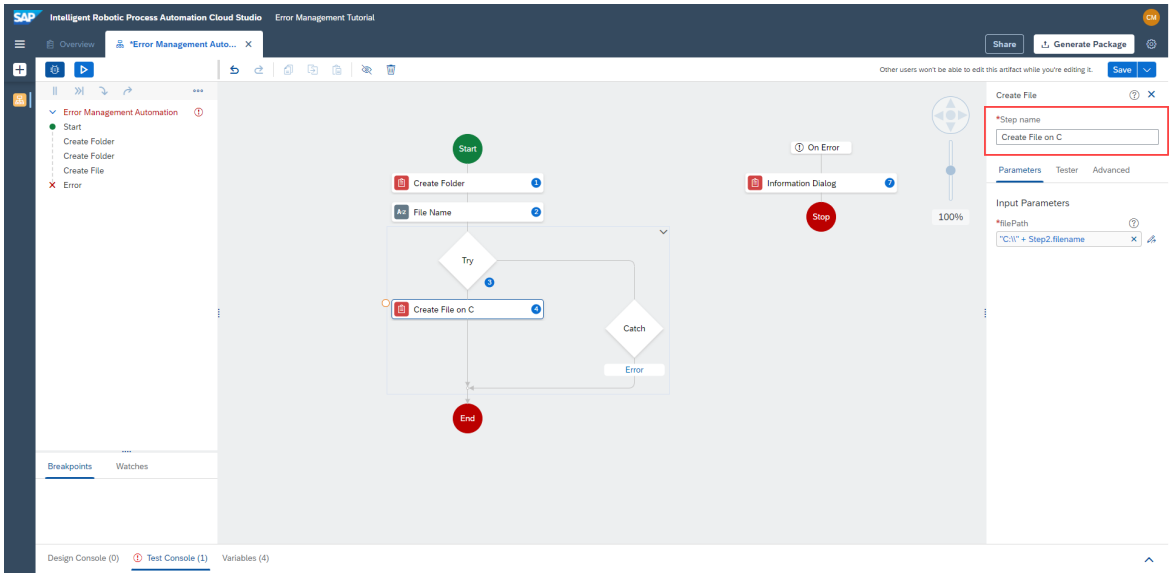
8. Delete the previously set file path and enter `"C:\\" + Step2.filename`. Click *Save Expression*.



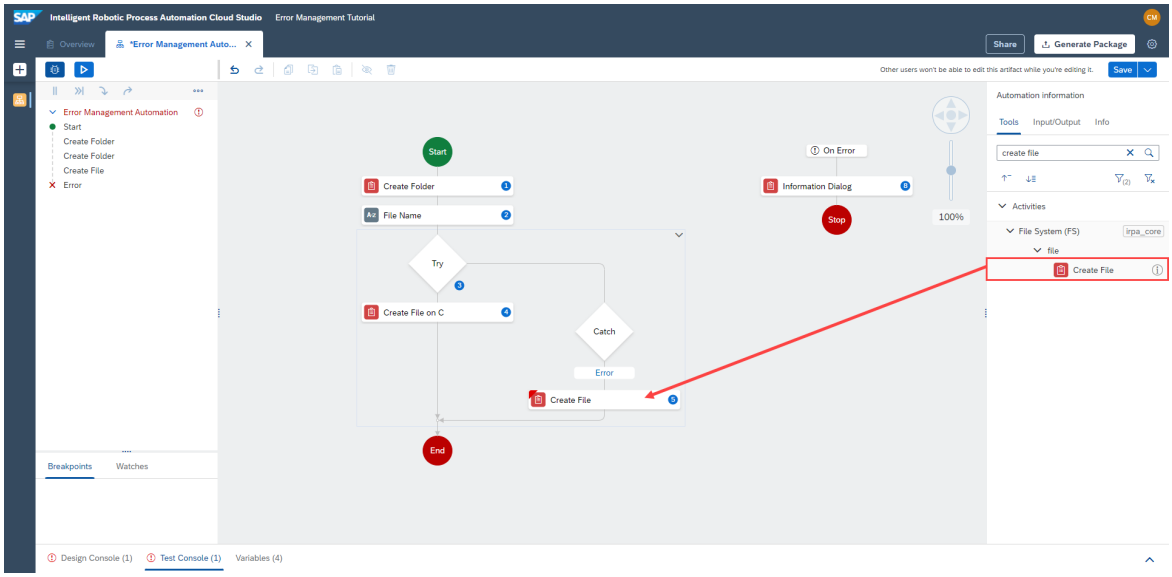
9. Select the *Try* control. On the *Try* side panel, select *Error* from the dropdown list below *Errors to catch*.



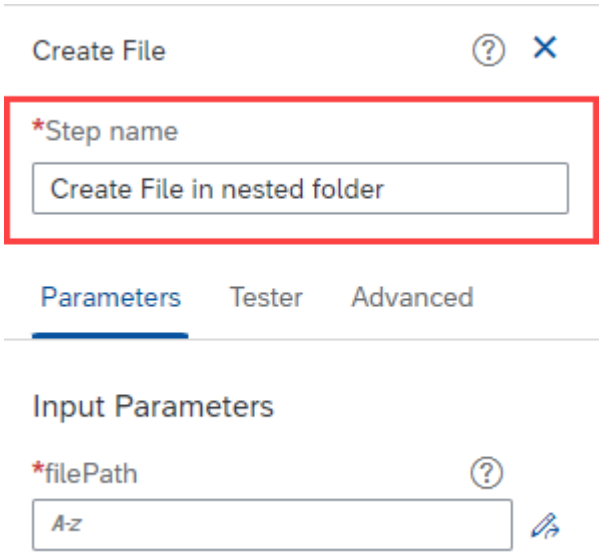
10. Select the *Create File* activity. On the *Create File* side panel, enter the step name **`Create File on C`**.



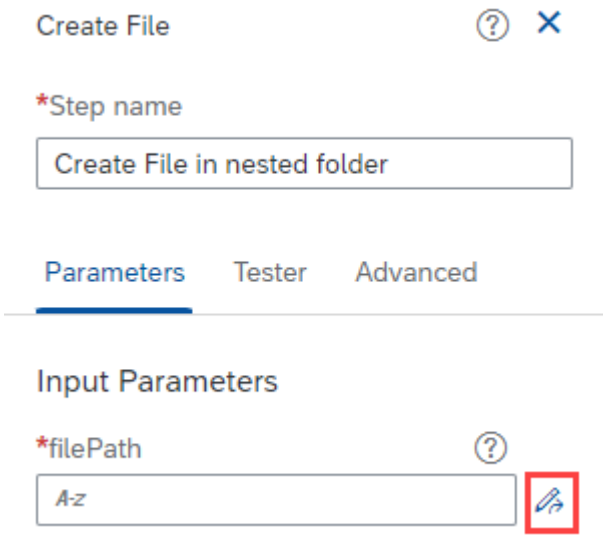
11. In the search bar of the *Automation information* side panel, enter **`Create File`**. Drag and drop the *Create File* activity on the *Catch* branch.



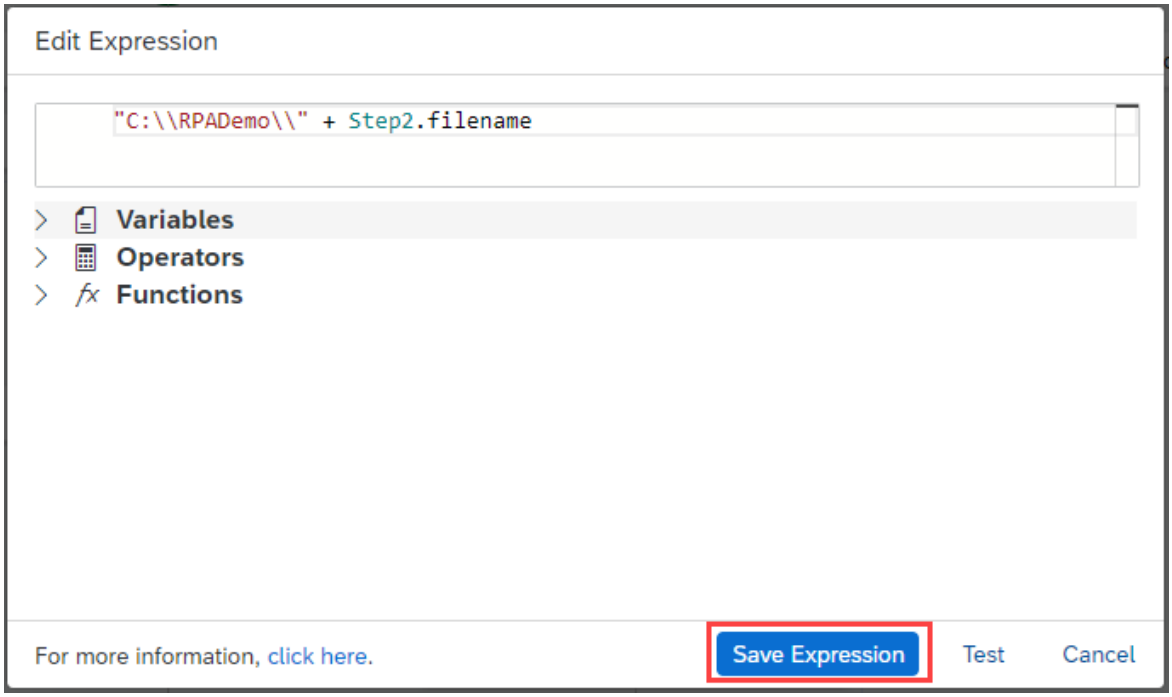
12. On the *Create File* side panel, change the *Create File* activity's name to **`Create File in nested folder`**.



13. Click the button on the right side of the *filePath* field to open the *Expression Editor*.



14. Enter `"C:\\RPADemo\\" + Step2.filename` in the *Expression Editor* and click *Save Expression*.



15. Save and test your automation.

    In your `RPADemo` folder, an `Invoice.txt` file has been created.

Next Steps

We are now going to catch the error raised with the *Try* control.

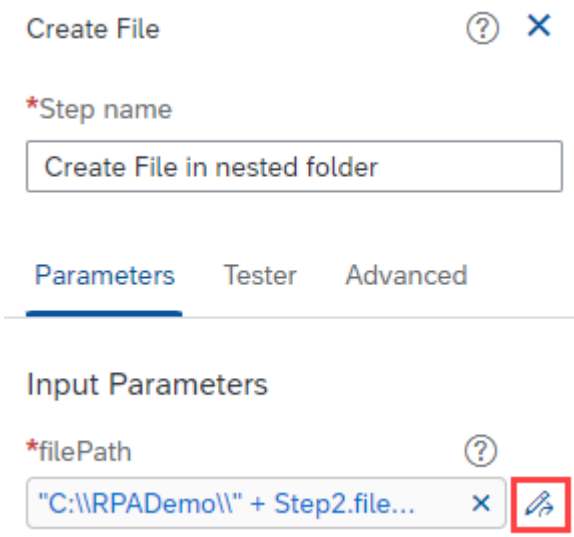
4.2.14.1.5 Catch the Error with Try/Catch

Context

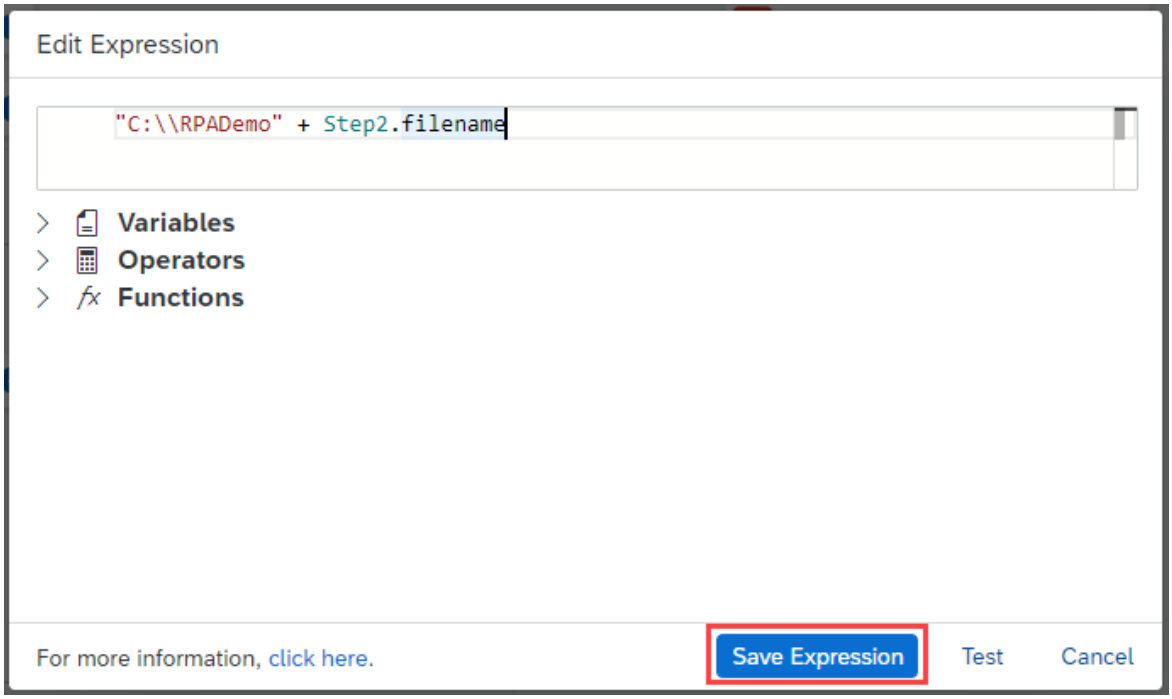
In this section, you will manage an Error with the *Try/Catch* control.

Procedure

1. Select the *Create File in nested folder* activity in the workflow. Click the button on the right side of *filePath* to open the *Expression Editor*.

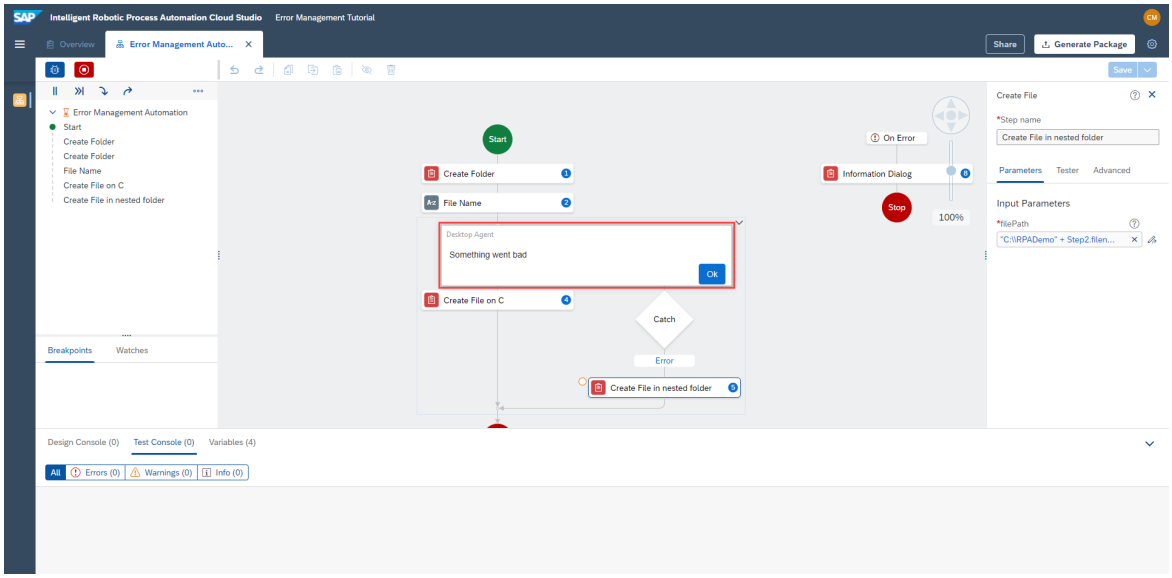


2. Delete the previously set expression and enter `"C:\\RPADemo" + Step2.filename`. Save your expression.

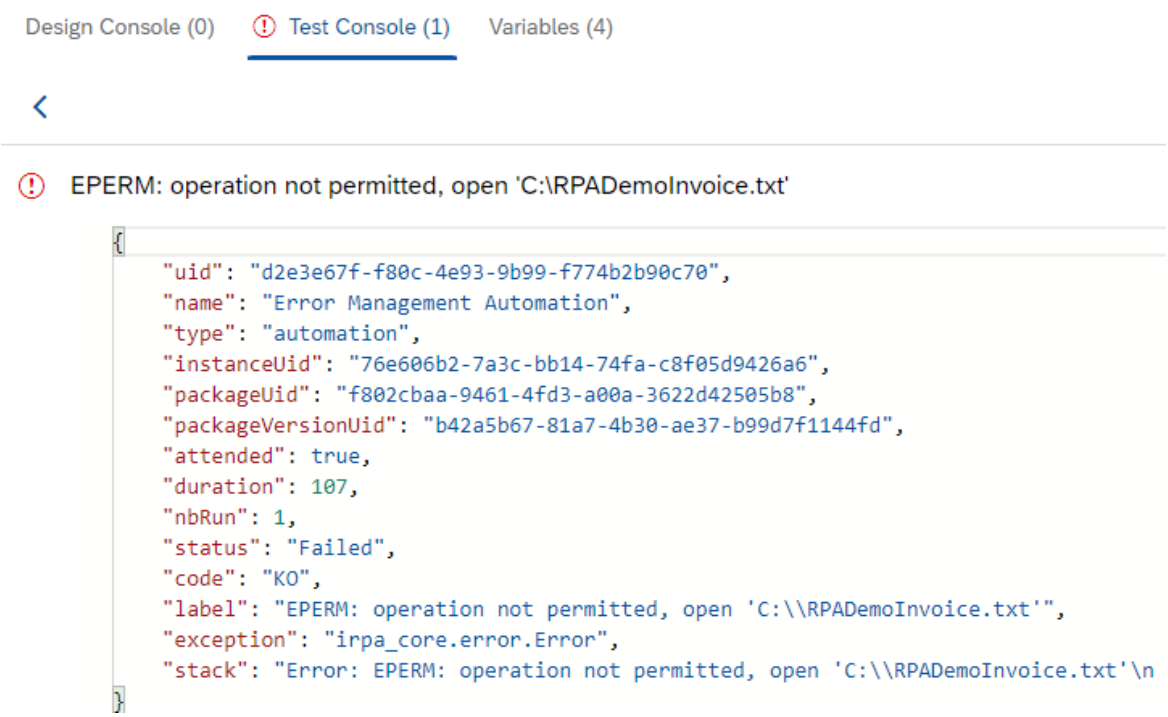


3. Save and test your automation.

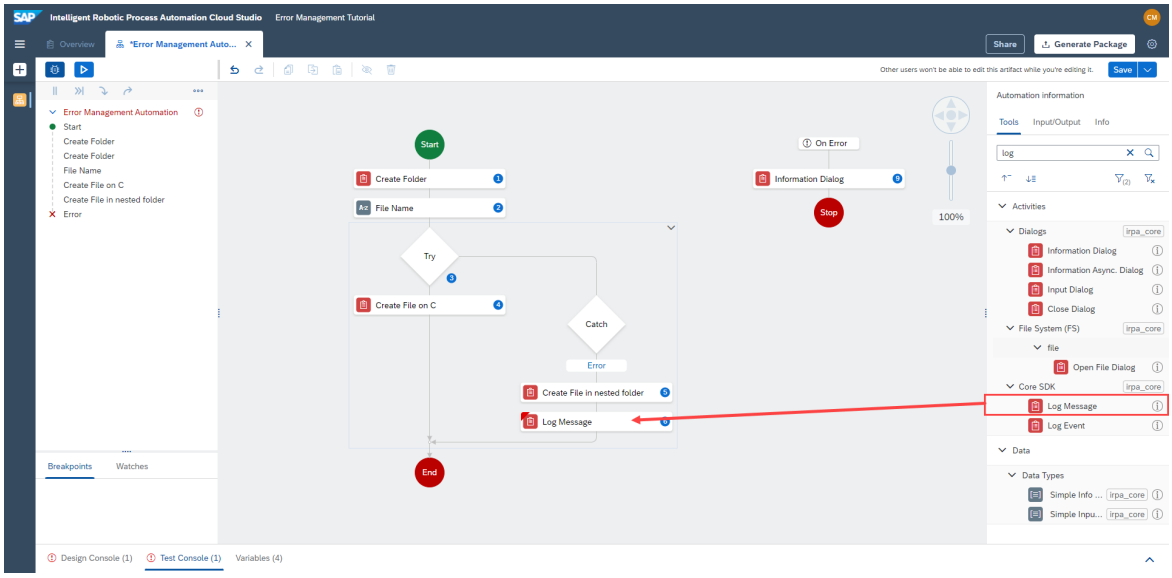
An error has been raised. However, this error is caught by the *On Error* node and not by the *Try* control. The following message box appears:



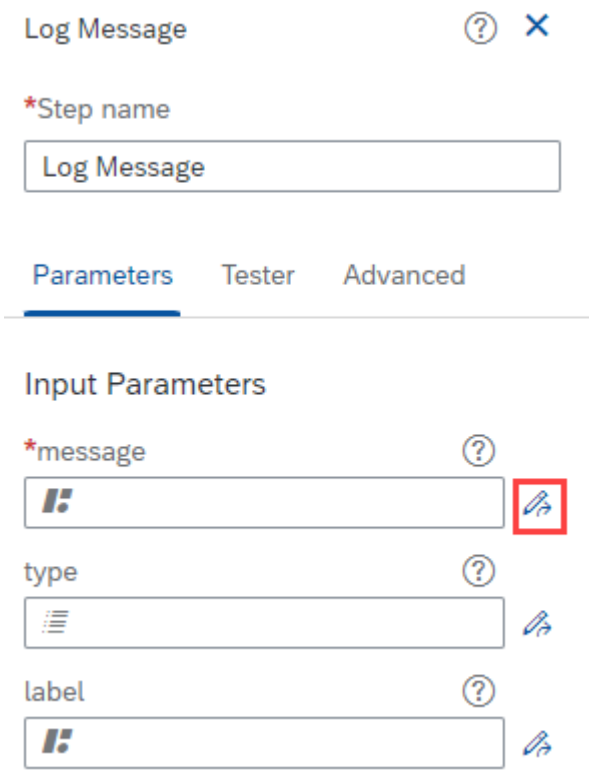
You can review the test data in the *Test Console* tab.



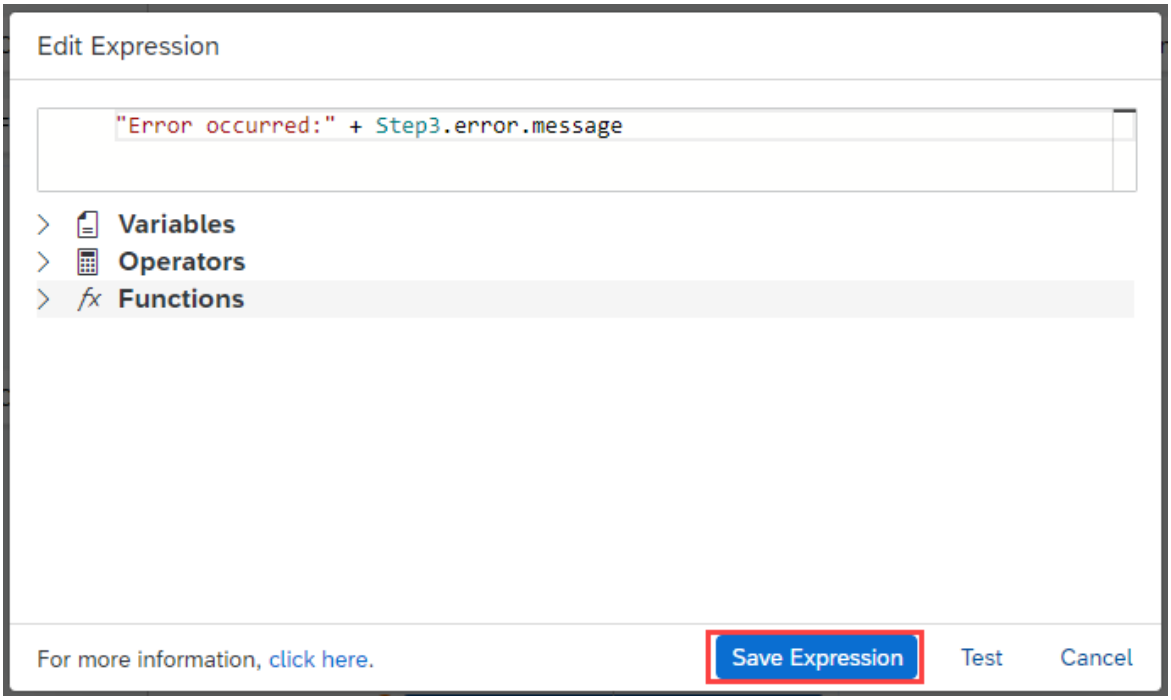
4. Enter **Log** in the search bar on the *Automation information* side panel. Drag and drop the *Log Message* activity on the *Catch* branch.



5. On the *Log Message* activity's side panel, click the *Expression Editor* button on the right side of the *message* field.



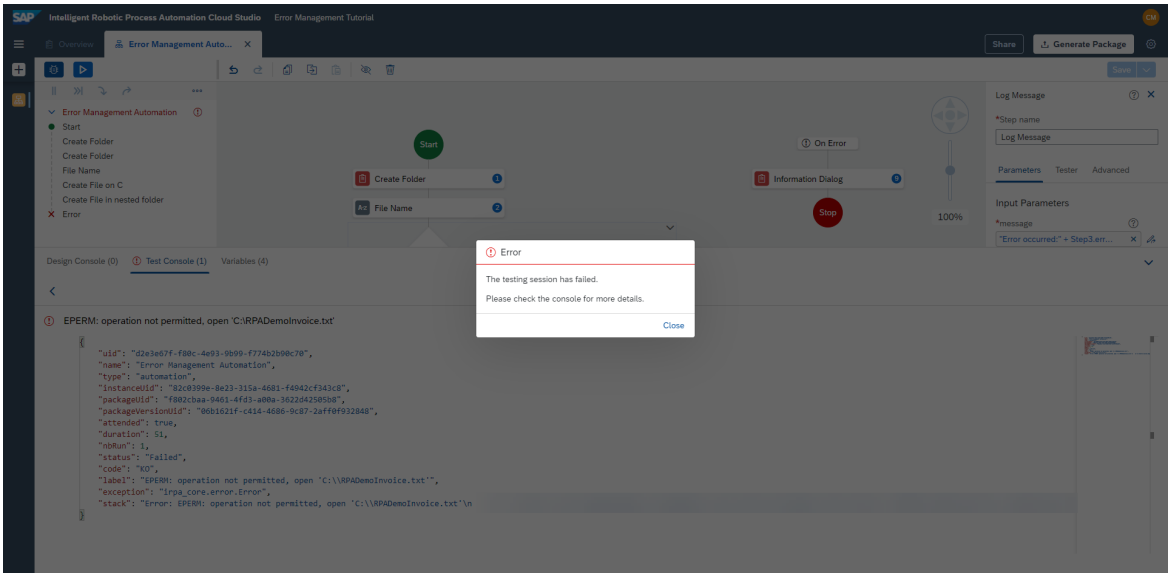
6. Enter `"Error occurred:" + Step3.error.message` and click *Save Expression*.



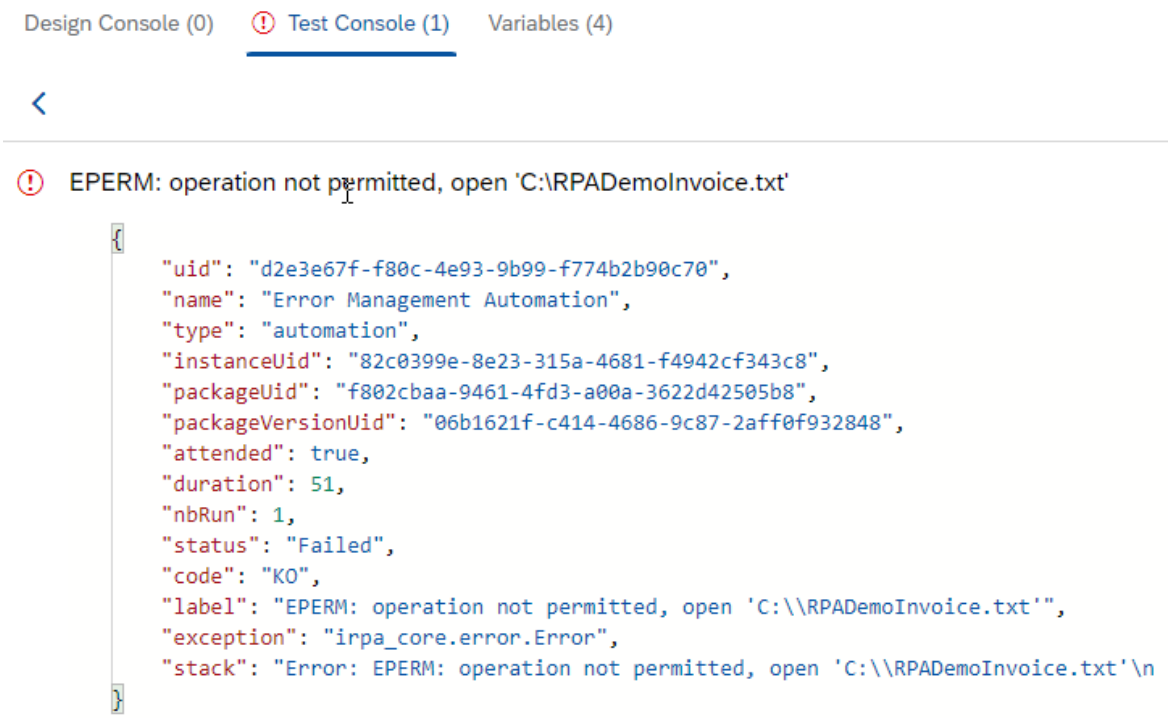
7. Save and test your automation.

   The automation fails when it reaches the *Create File in nested folder* activity and goes to the *On Error* node (because we created a file path which raised an error).

   A message appears, stating that *Something went bad*. Click *Ok*. The testing session fails.

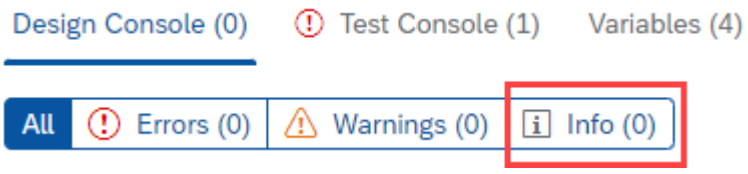


   You can review the test data in the *Test Console* tab.



8. Click the *Info* tab at the bottom of your screen.

   The *Log Message* does not appear in the *Info* tab because the automation failed before reaching this part. The *On Error* node caught the error and stopped the main automation.



Results

You now know how to use the *Try* control in your automation. You also know which errors to choose and how to use them in an automation.

4.2.14.2 On Error Handling Workflow

If an error is raised in your automation, the *On error* node allows you to continue the exeuction of your automation by adding activities on a side workflow. These activities will be executed after the main workflow fails.

> **i** Note
>
> The **On error** node workflow is executed if the errors raised by the automation are not caught with the **Try** control. For more information, see Try Catch. [page 239]

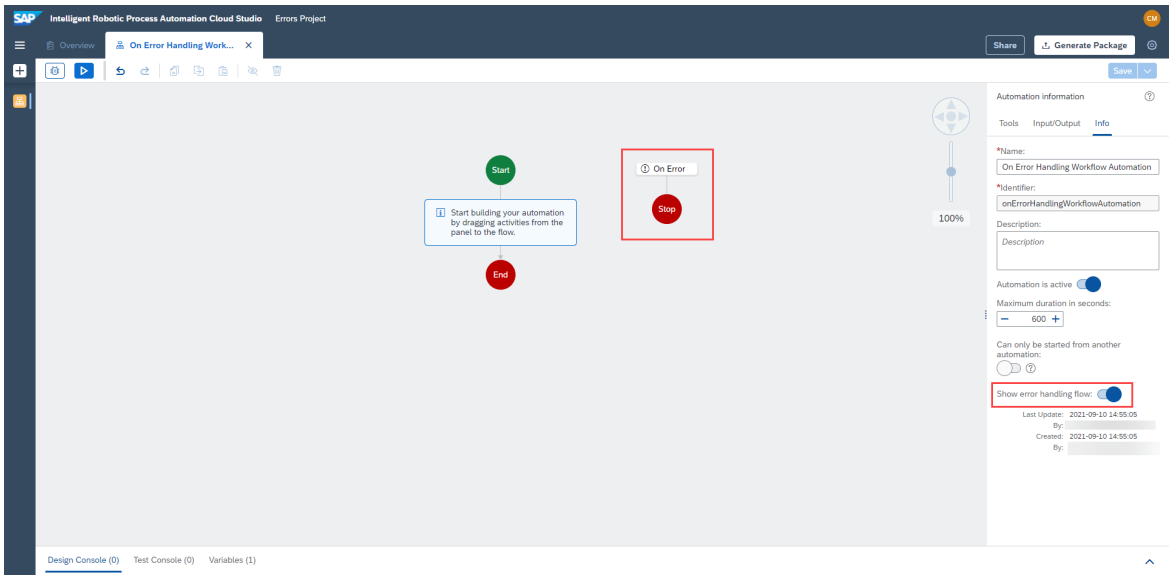
Prerequisite

You have created your automation.

Procedure

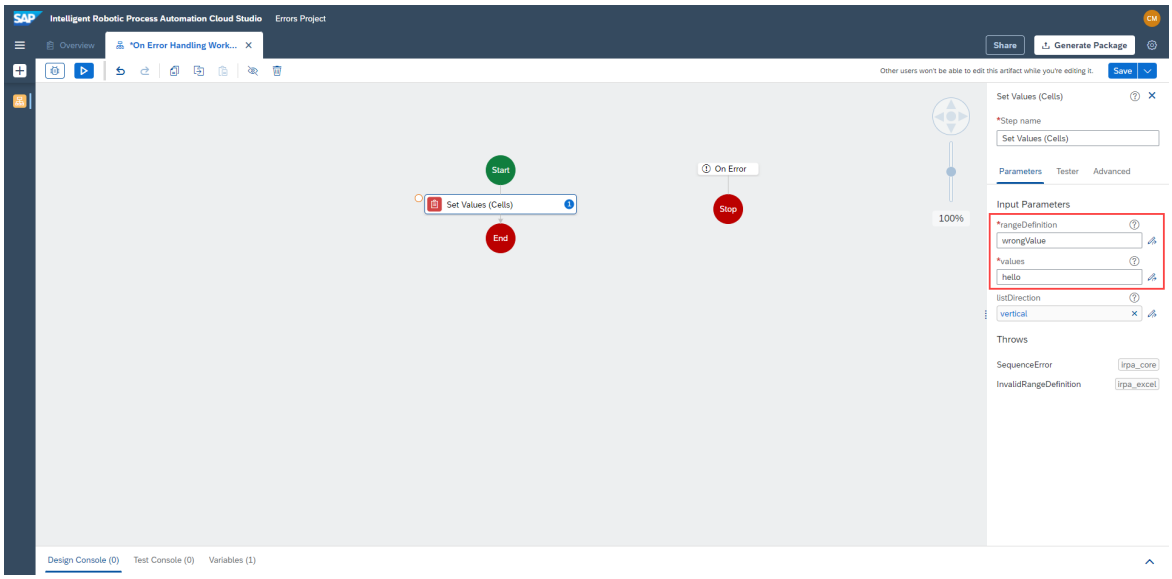
The following procedure demonstrates how to manage a failing automation with the on error node.

1. On the *Automation information* side panel, select *Info*, and enable the *Show error handling flow* button.



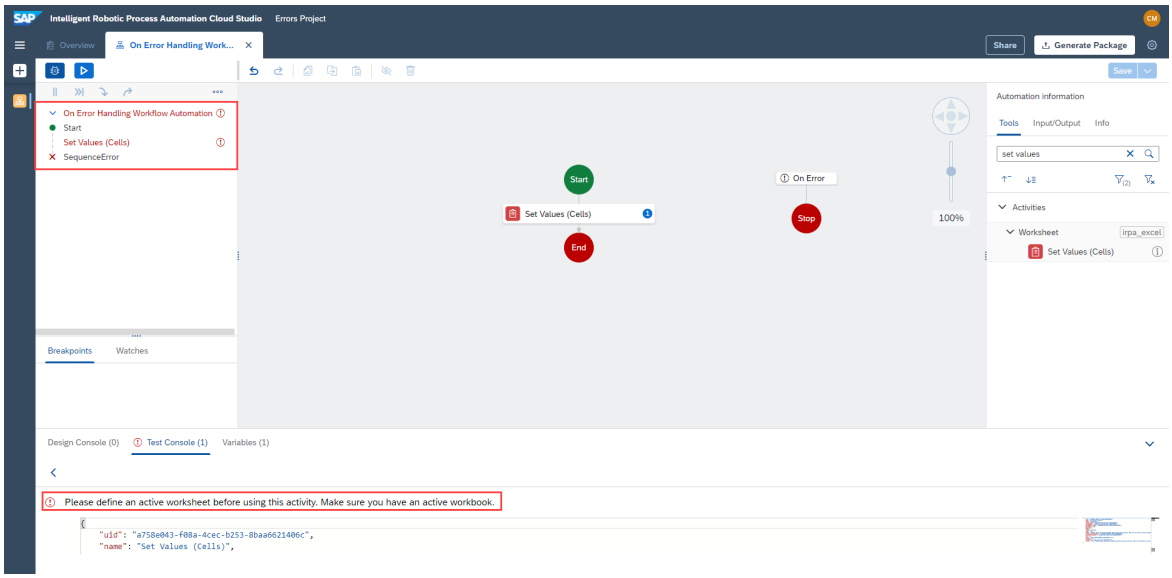
   The **On error** node appears in your automation on a side workflow.

2. In your automation, drag and drop the activity *Set Values (Cells)*. On the *Set Values (Cells)* side panel, set `wrongValue` in the *rangeDefinition* field under *Input parameters*. Set also a value in the *values* field.



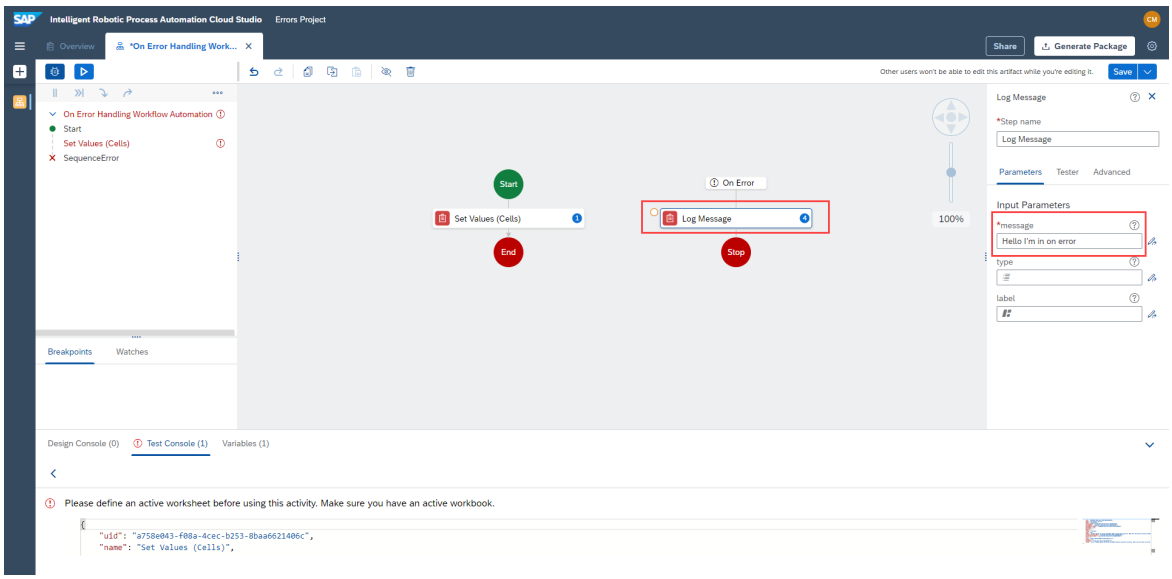
   By adding a *Set Values (Cells)* activity without opening an Excel file first and setting wrong values, the automation will fail at execution time.

3. Save and test your automation. The testing session fails.



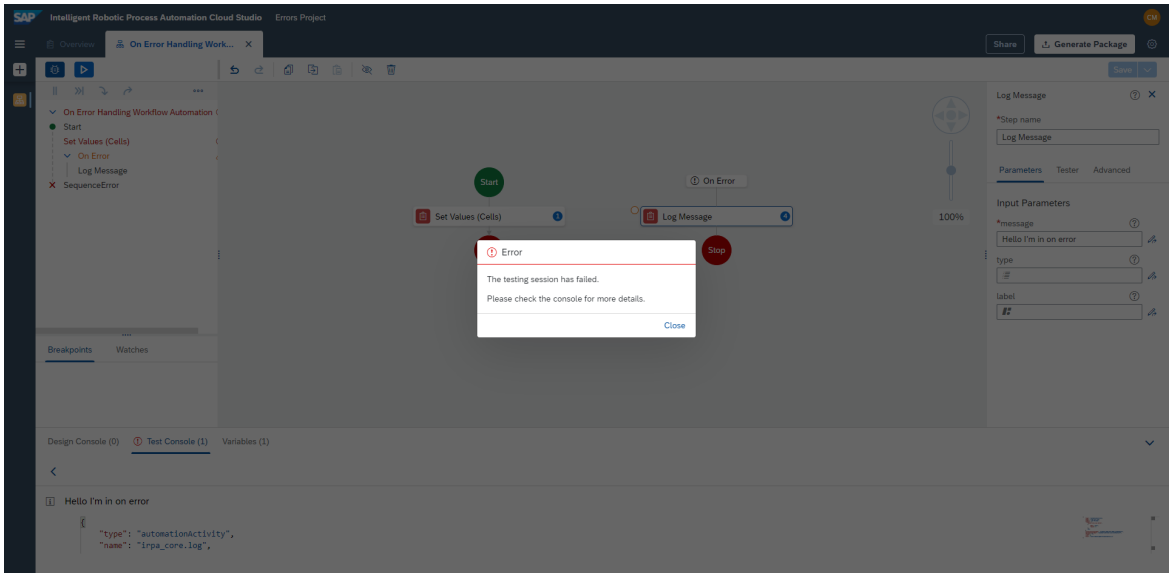
You can review the data of your failing automation in the timeline and in the *Test Console*.

4. Drag and drop the *Log Message* activity on the **On error** node. On the *Log Message* side panel, enter a message in the *Input Parameters*.

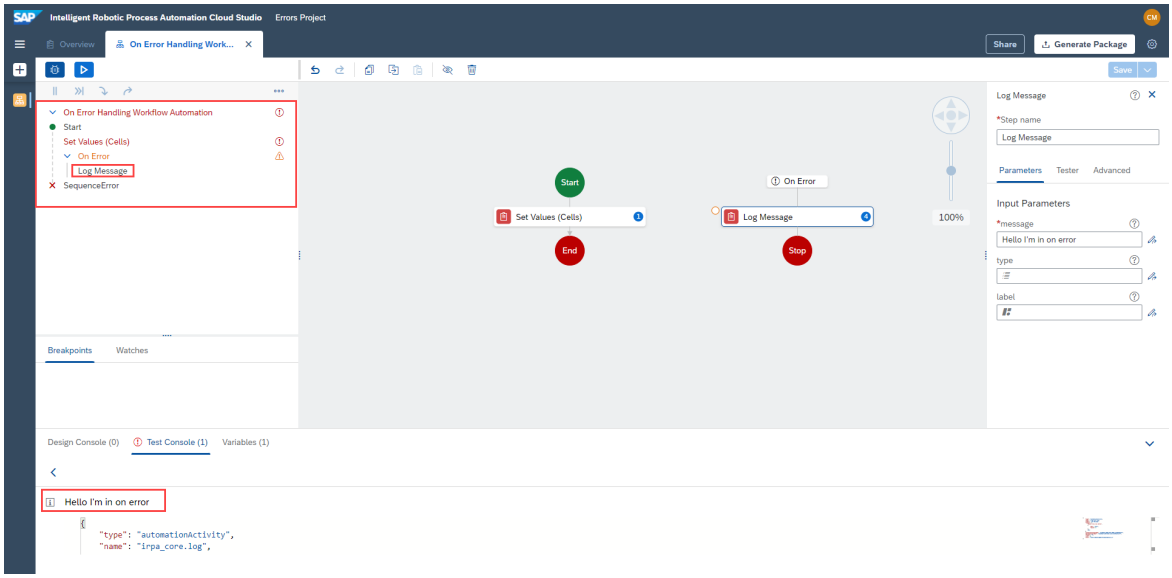


When the automation fails, it will execute the steps inside the *On error* node.

5. Save and test your automation. With the error from the first testing session, the automation still fails and the same error is displayed in the *Test Console*.

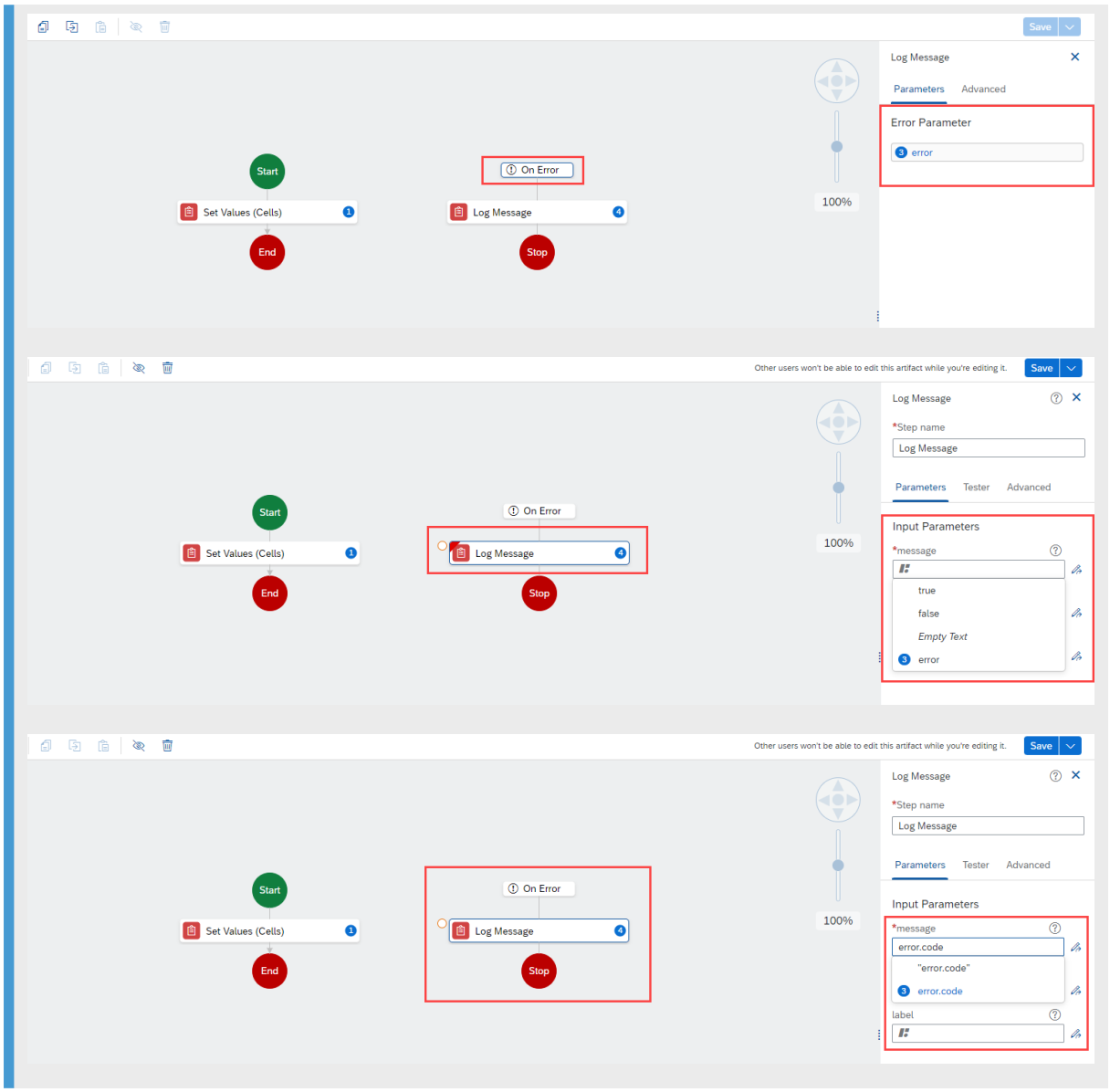


But after the failing step in the timeline, you can see that the *Log Message* activity from the *On error* node has been executed. The message is also displayed in the *Test Console*.



Note

You can access an error object in the *On Error* automation branch. If required, you can manage an error object.



Related Information

Try Catch [page 239]

4.2.14.3 Generate a Business Error with the Stop Automation in Error control

The **Stop Automation in Error** control allows you to raise a business error that generates a customized input message in the logs, when your automation is failing.

In this procedure, you will create a workflow with a condition and raise a business error that will give a different result for your automation. With a business error, you can decide if you want to execute the activity on the condition branch or stop your automation.

Prerequisite

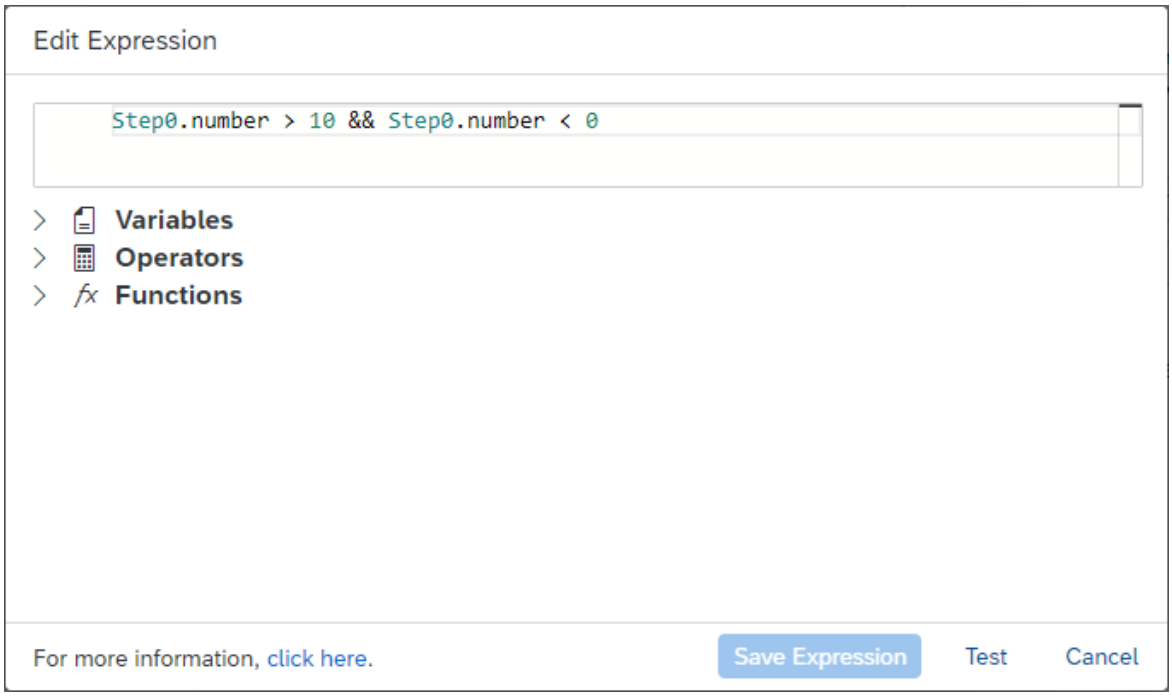
You have created an automation *Throw business error*.

Procedure

1. In the *Input/Output* section, on the right-hand side panel, define the input parameter **number** of type **number**.
2. Drag and drop the *Condition* control in the workflow.

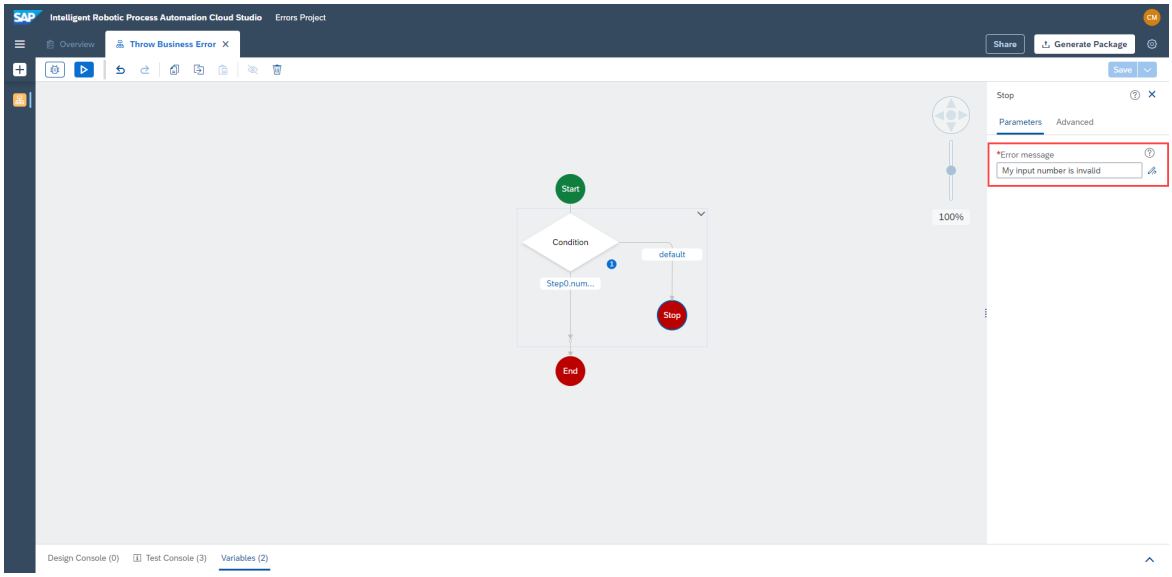
   > **i Note**
   >
   > A business error can only be raised inside a condition control.

   Enter the following condition `Step0.number > 10 && Step0.number < 0`

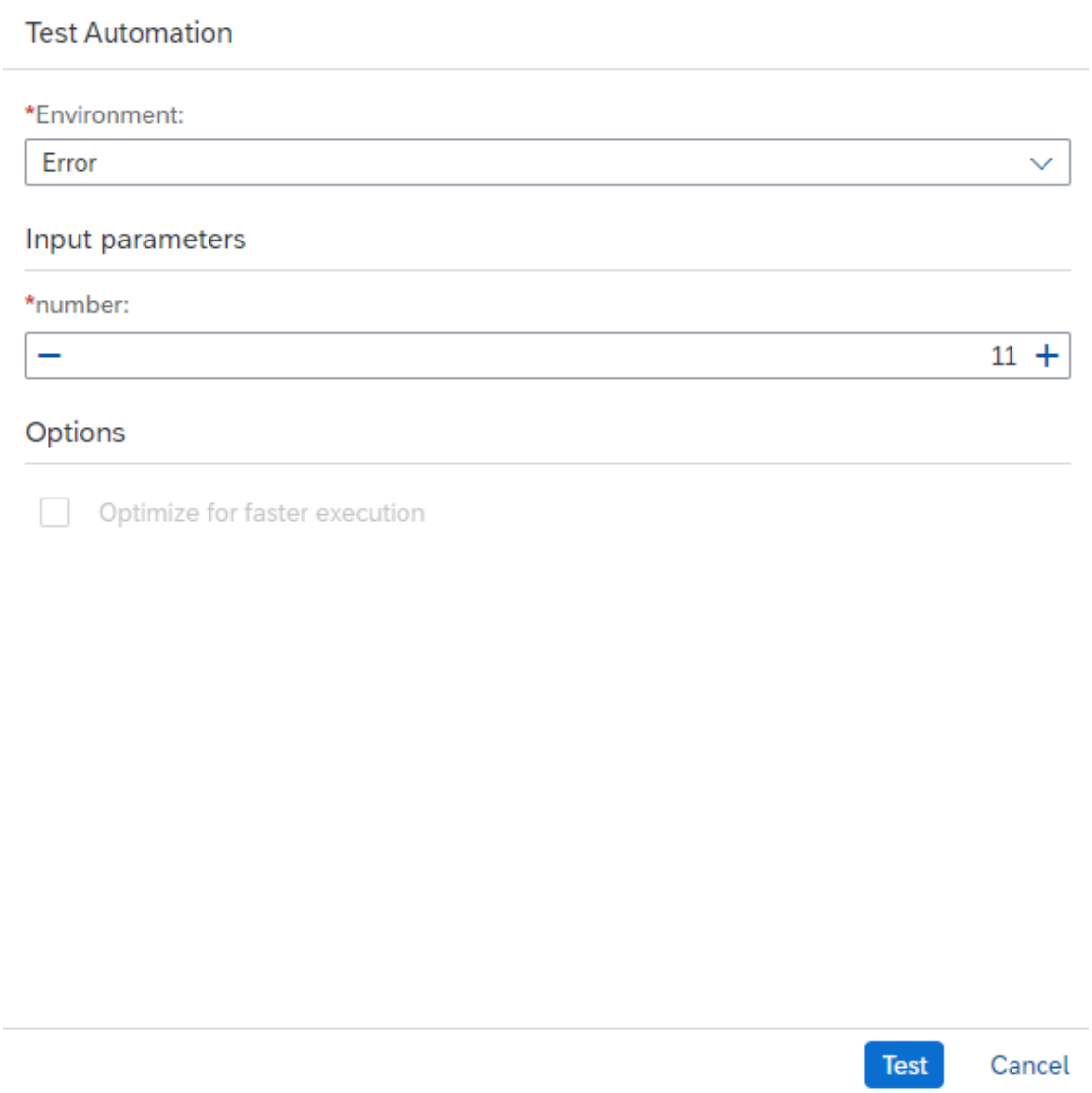


With this condition, if the input number entered at the testing phase is between 0 and 10, the automation goes onto the main branch.

3. Drag and drop the *Stop automation in error* control on the default branch of the condition.
   By dragging this control on the default branch, the automation will end in an error, if the condition set in the second step of the procedure is not met.
4. Select the *Stop automation in error* control.
   On the *General Data* side panel, the control awaits an *Error message* as input. This error message is the business error that you will raise with the *Stop automation in error* control. At execution time, this business error will be displayed on the *Tester Data* side panel and in the logs.
   1. Enter an error message.

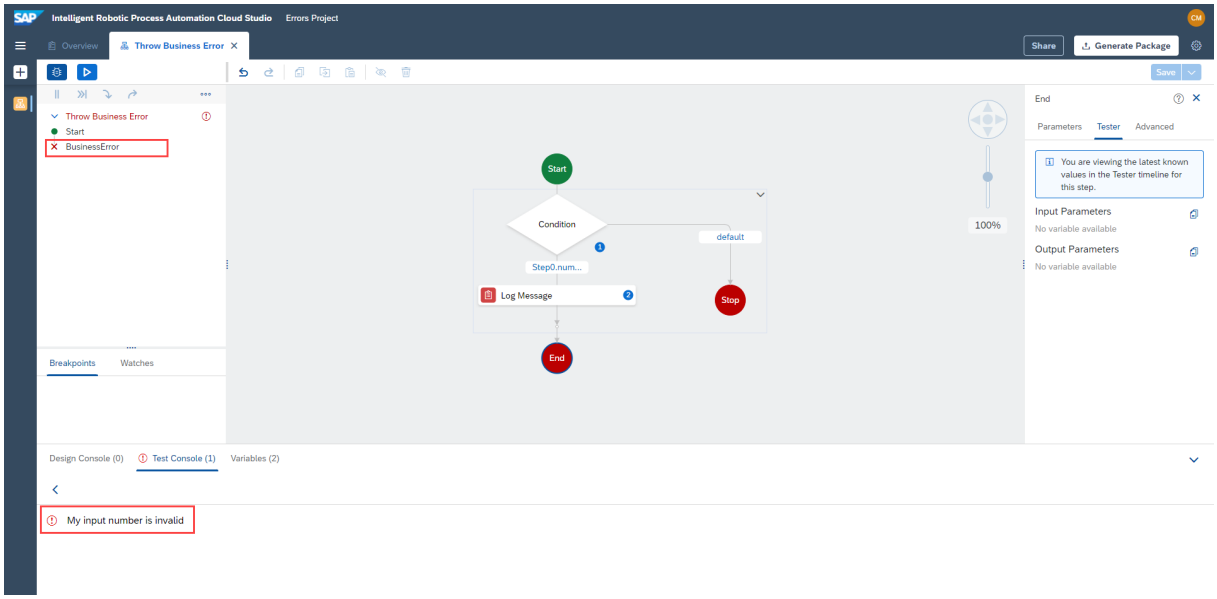


5. Add a *Log Message* activity on the condition branch with the input message **Valid number**.
6. Test your automation.
   1. Select your environment.
   2. In the required input parameters, add a number higher than 10 and click *Test*.



Results

Your automation fails with the *Stop automation in error* control. The error raised by the *Stop automation in error* is displayed as a **BusinessError** on the tester timeline and the error message previously set in the automation is displayed in the logs.

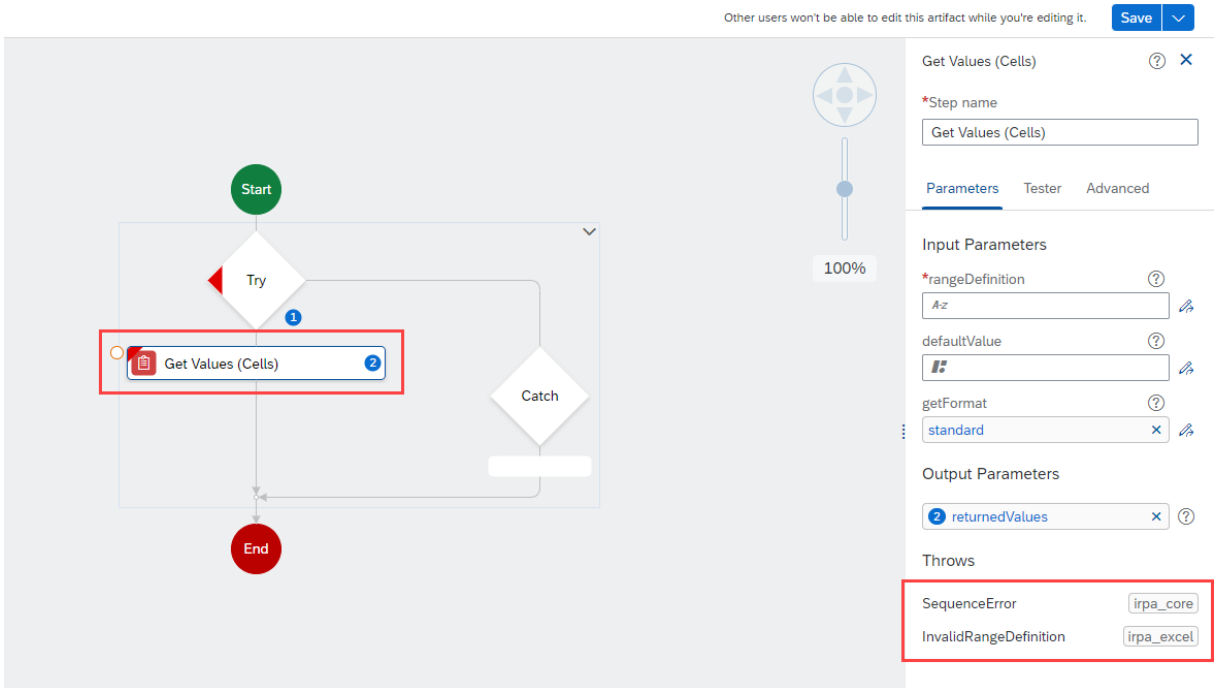


4.2.14.4 Display List of Errors Associated with Activity

In an automation, you can view the list of possible errors associated with an activity.

To view list of errors, you must select an activity in an automation. The list of possible errors associated with an activity is displayed.

For example, refer to the following screenshot.



> **Note**
>
> You must select an activity which has a possibility of raising an error.

You can catch errors raised on specific steps of the workflow and handle them on a side branch of the automation.

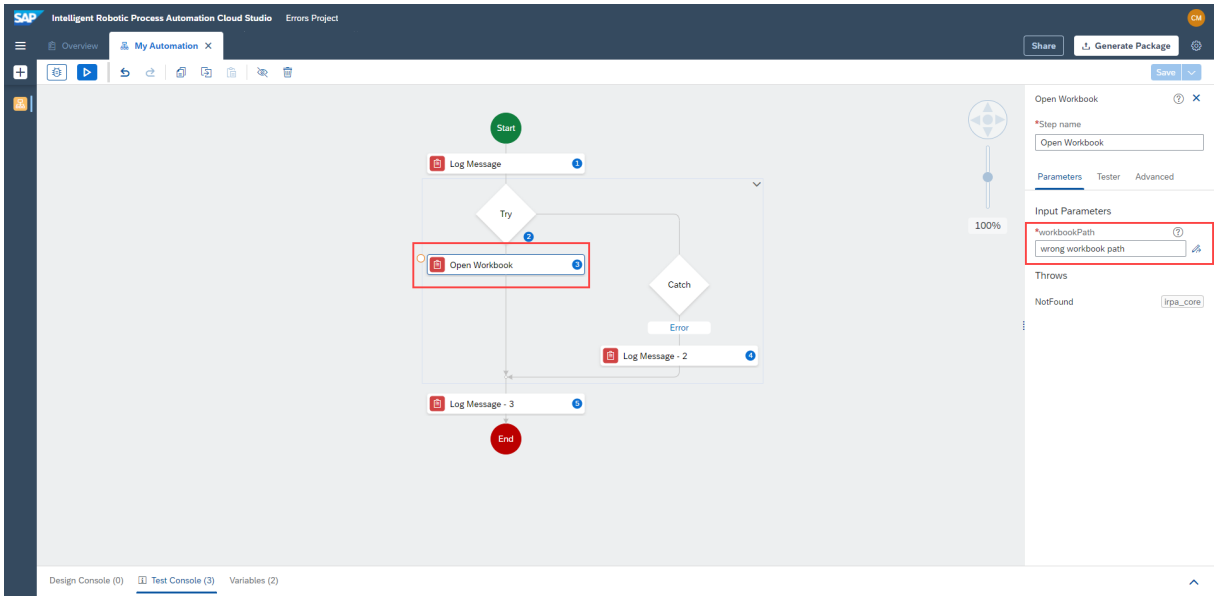
Related Information

Condition Controls [page 231]
Catch the Error with Try/Catch [page 326]

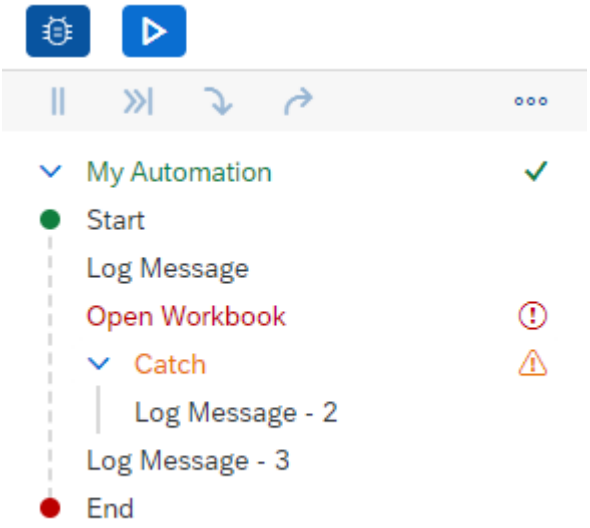
4.2.14.5 Display Errors in the Timeline

In the Cloud Studio, when you test your automation, the errors are displayed on the tester side panel in the timeline.

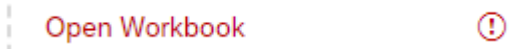
In the following use case, you log a message and want to execute the activity *Open Workbook* in the *Try* control. You enter a wrong workbook path and as a result, the activity fails. It goes on to the *Catch* branch with a log message and the automation ends in a log message.



Once you start the testing session, the errors in the workflow of your automation are labeled on the timeline.



The failing step is displayed in red and you can select the error in the timeline to view more information on the *Tester* side panel.



The *Catch* container is in orange and displays the activity on the *Catch* branch in the workflow of your automation.



> **Note**
>
> By default the activity contained on the *Catch* branch is displayed but you can click the arrow to collapse or expand the list of activities.

Related Information

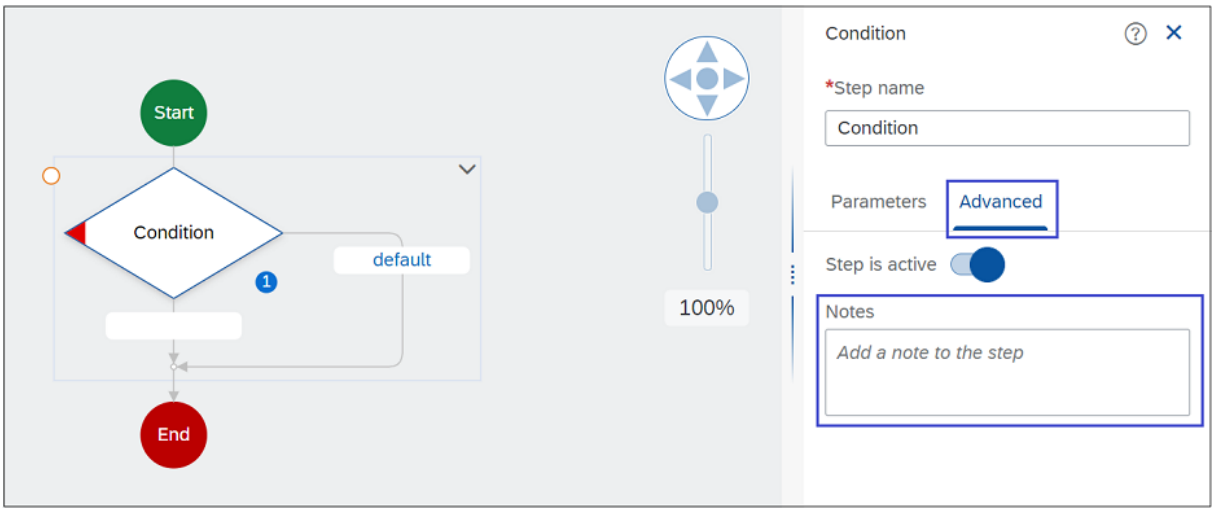
Test an Automation [page 282]

4.2.15 Tips and Tricks

4.2.15.1 Annotation

In the Cloud Studio, you can add a note for each step (activity, node, user task, and many more) in an automation workflow. This helps you to add a brief description about an automation step.

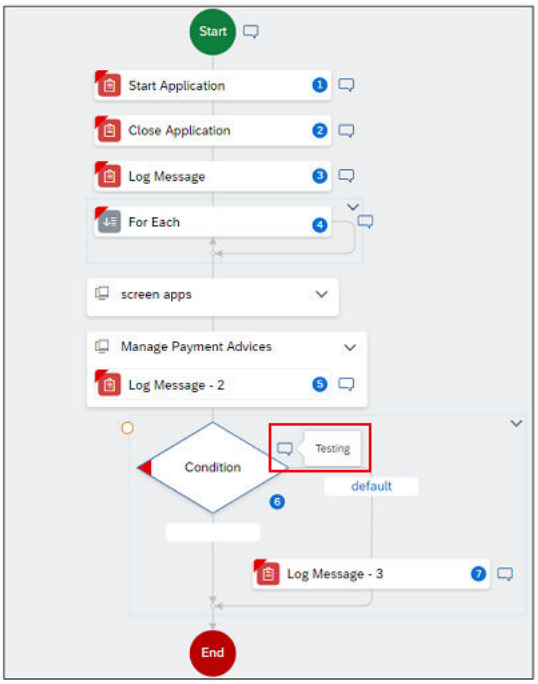
The **Notes** option is available under *Advanced* tab of a selected step or node on the right panel of the Cloud Studio. For example, refer to the following screenshot.



You can enter an appropriate information in the *Add a note to the step* field.

> **i Note**
>
> You can enter a maximum of 1024 characters.

After adding the information for a step, the *Note* icon is displayed on the right side of the step. Click the *Note* icon to view the content of the note.



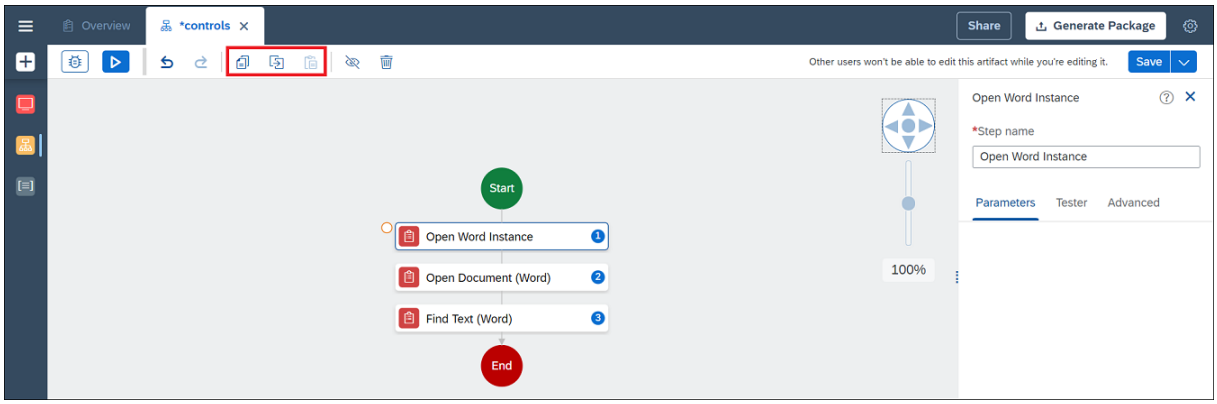
> **Note**
>
> You cannot add a note for a screen.

4.2.15.2 Copy, Paste, and Duplicate Items

You can copy and paste or duplicate the Activities [page 371], Data [page 160] (Data Management and Data Types), and Controls [page 216] with their value in the workflow of your automation.

> **→ Tip**
>
> The Copy, Paste and Duplicate options can be used from one automation to another across your different projects.

The following screenshot shows the Copy, Paste, and Duplicate options in the Cloud Studio.



Icon	Icon Name	Description	Procedure
	Copy	The copy function is handled by the Cloud Studio. Use this option to copy an item (Activity or Data (Data Management and Data Types), and Controls).	Select an item to be copied (Activity or Data (Data Management and Data Types) and Controls) in a workflow of your automation and then click the *Copy* icon. The item is now copied.
	Paste	The Paste function is handled by the Cloud Studio. Use this option to paste a copied item (Activity or Data (Data Management and Data Types), and Controls) at the desired location.	Select an item (Activity or Data (Data Management and Data Types) and Controls) in a workflow and then click the *Paste* icon. The item is now pasted after the selected item. **i Note** If do not select an item and you paste, the copied item will be pasted outside the workflow of your automation.
	Duplicate	The Duplicate function is handled by the Cloud Studio. Use this option to duplicate an item (Activity or Data (Data Management and Data Types), and Controls).	Select an item (Activity or Data (Data Management and Data Types)and Controls) in a workflow and then click the *Duplicate* icon. The item is now duplicated and is available after the selected item.

i Note

When you copy an item from one automation to another, click on the green start node of the second automation to paste the items directly in the workflow of the automation.

You can also use the Duplicate option at item (Activity or Data (Data Management and Data Types), and Controls) level.

The following screenshot shows the availability of the Duplicate option of items.



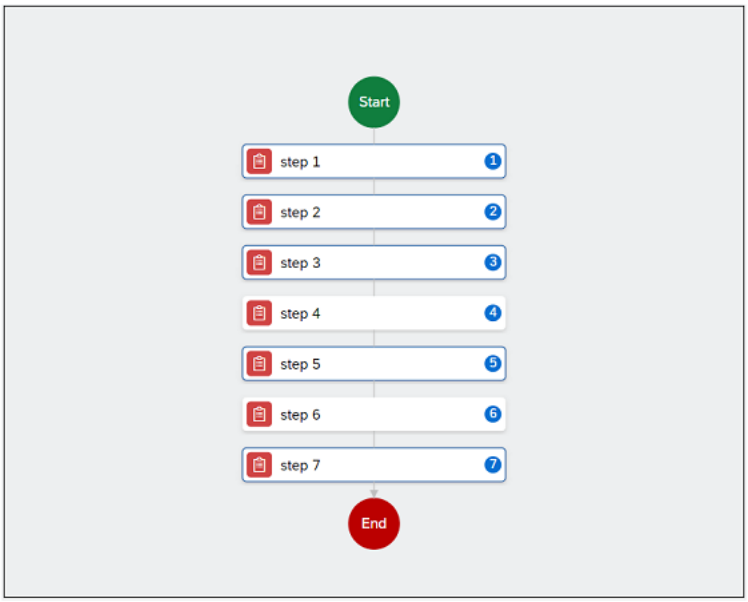
To create a duplicate item, follow the procedure below:

1. Select an item and right-click.
   The more options are displayed.
2. Click *Duplicate*.
   The duplicate item is created after the selected item.

4.2.15.3 Select Multiple Elements

In the Cloud Studio automation editor, you can select multiple elements and various actions such as copy, delete, deactivate, activate, drag-and-drop, and many more can be performed.

To select multiple elements, use **ctrl** key and the select required elements.



> **Note**
>
> - The step details will not be displayed in the left panel if you select multiple elements. To view step details, you must select a singe element.
> - You cannot copy and paste multiple elements but you can delete multiple elements.
> - You cannot duplicate multiple elements.
> - You can drag-and-drop multiple elements.

You cannot drag-and-drop sequenced step(s). For example, if you select step 3, you cannot drag-and-drop between steps 2 & 3 and steps 3 & 4.



- If you drag-and-drop a parent container, then the related or child steps also will be copied.

ℹ Note

You can select multiple elements by pressing the Shift key and then selecting multiple elements in the automation workflow.



Related Topics

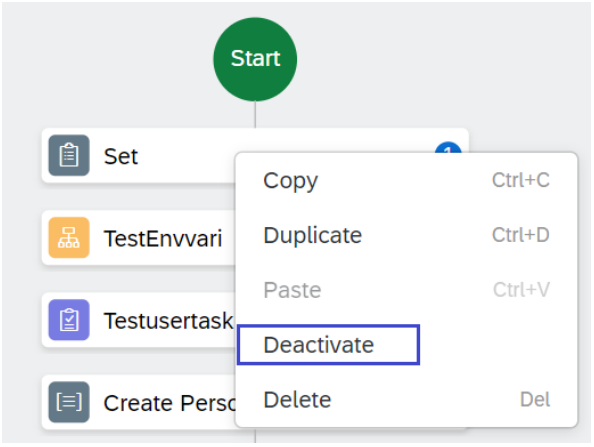
- Copy, Paste, and Duplicate Items [page 343]
- Deactivate an Automation Step [page 347]

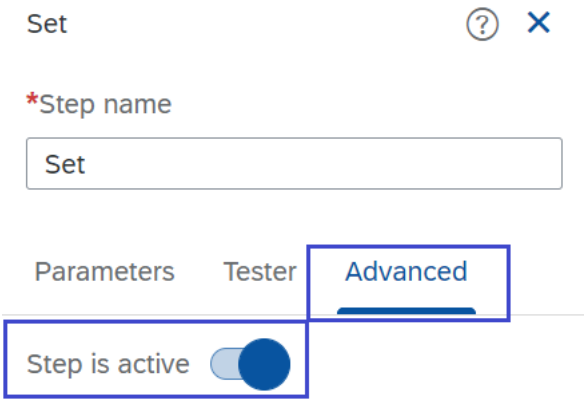
4.2.15.4 Deactivate an Automation Step

You can deactivate certain steps in a workflow of an automation.

To deactivate a step in a workflow, you must select the step or node to be deactivated and then deactivate. The step or node is deactivated. This means the deactivated step or node will not be executed during automation.

In the Cloud Studio, the Deactivate option is available in various locations.

Deactivate option	Description
	By default, this icon is disabled.  Once you select the step, the  icon is enabled. Select the step or node to be deactivated and click the  icon to deactivate a step. **i Note** At any point in time, you can activate the step by clicking the  icon.
	In a workflow of your automation, right click on any step or node. The *Deactivate* option is displayed. Select the step or node to be deactivated, right click, and then click *Deactivate* to deactivate the step or node. After successful deactivation, the step or node will be greyed. **i Note** At any point in time, you can activate the step by clicking *Activate*.



The activate or deactivate option is available under *Advanced* of a selected step or node on the right panel of the Cloud Studio.

By default, the step status is active.

Select the step to be deactivated and then click the  icon to deactivate the step.

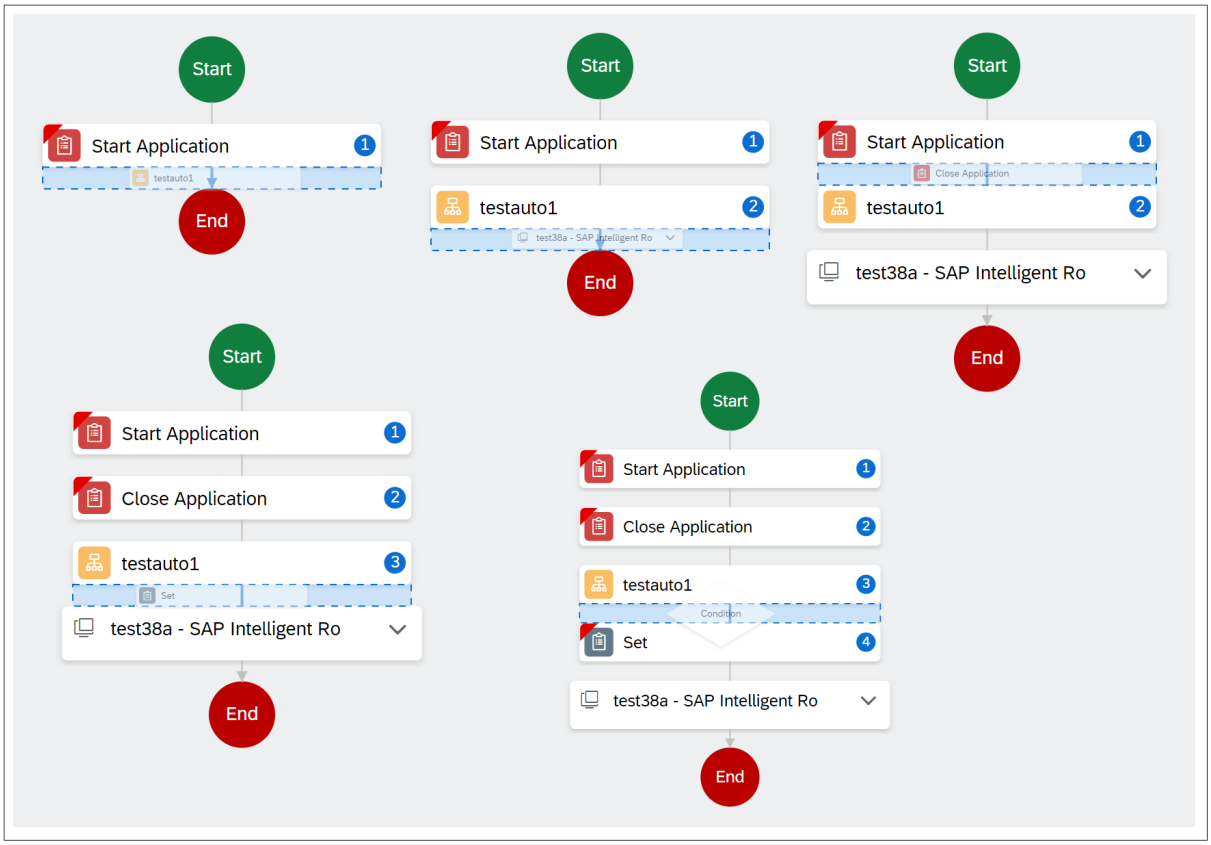
After successful deactivation, the step or node will be greyed.

> **i Note**
>
> At any point in time, you can activate the step by clicking the  icon.

4.2.15.5 Drag and Drop Items

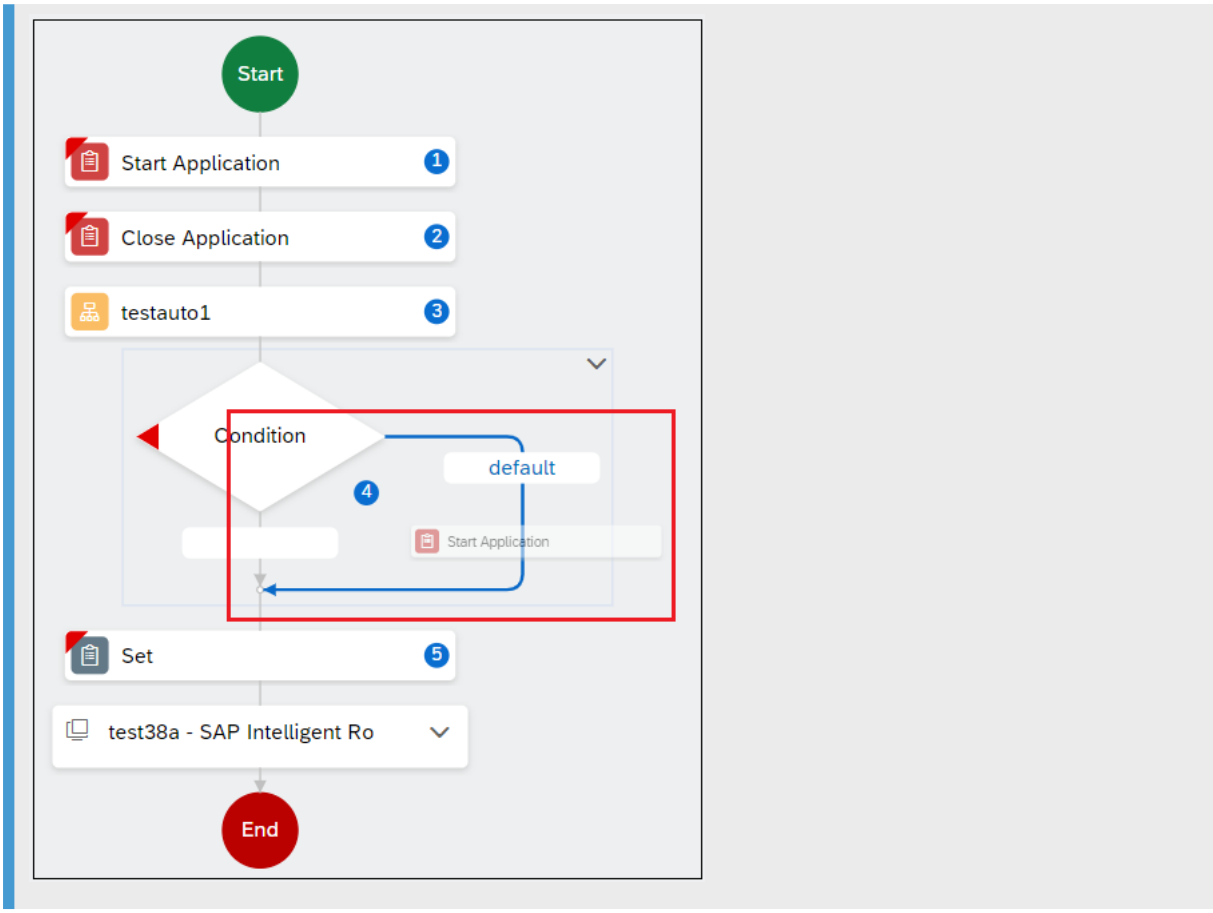
You can easily drag and drop items (Automations, Screens, Activities, Data, and Controls) in an automation.

In an automation, when you mouse hover with an item on the straight line, the straight line is highlighted with a dotted box. This helps you to drop an item at an appropriate place.



Note

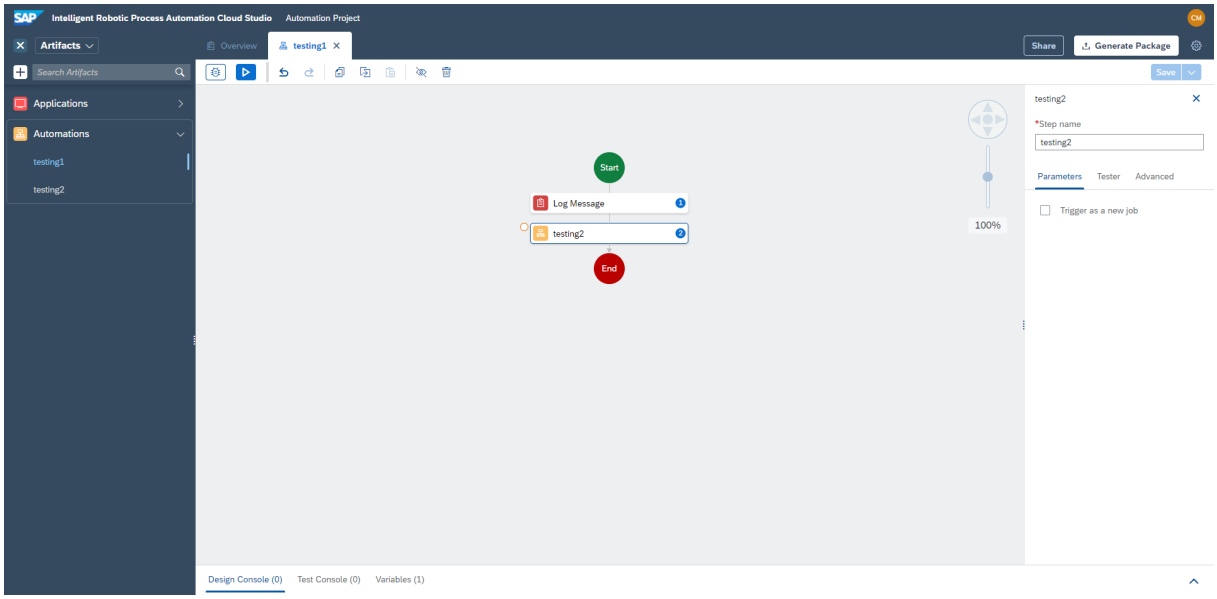
No dotted box is displayed for curved lines.

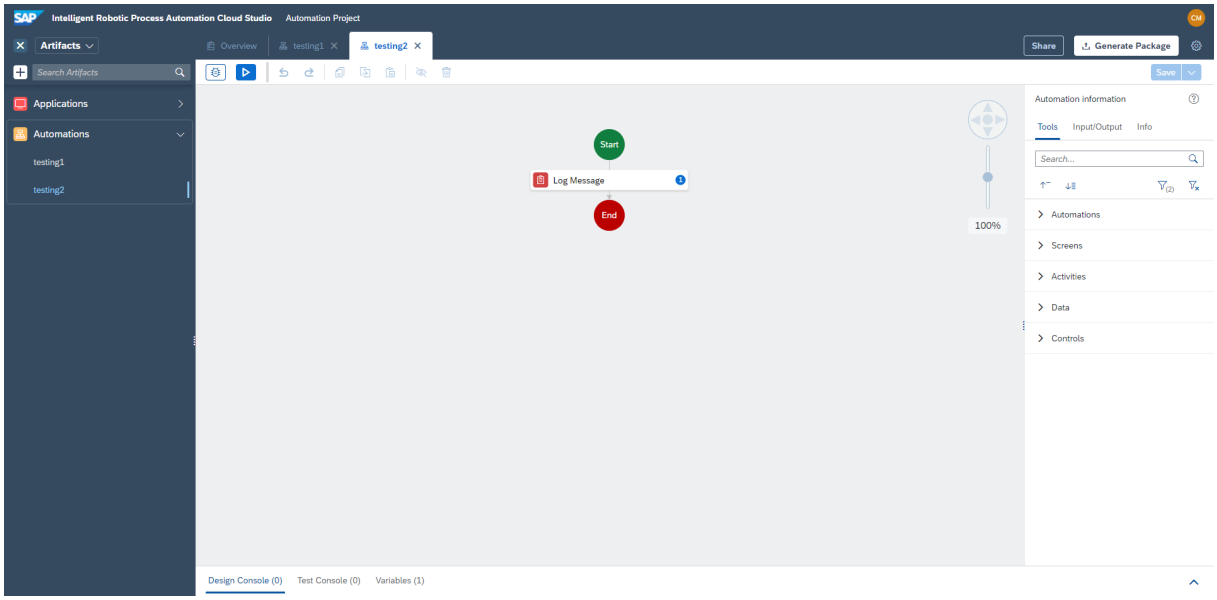


4.2.15.6 Open Automation Node in Separate Automation Editor

You can open an automation node in a separate automation editor by double-clicking the automation node in a currently active workflow. The automation or sub-automation opens in a separate automation editor.

In the following example, the **testing2** automation is opened in a separate automation editor by double-clicking the **testing2 automation node** in the **testing1** automation.


Intelligent Robotic Process Automation Cloud Studio
Automation Project
Artifacts
Overview
testing1
Share
Generate Package
Search Artifacts
Save
Applications
Automations
testing1
testing2
Start
Log Message
testing2
End
100%
testing2
*Step name
testing2
Parameters
Tester
Advanced
Trigger as a new job
Design Console (0)
Test Console (0)
Variables (1)


Intelligent Robotic Process Automation Cloud Studio
Automation Project
Artifacts
Overview
testing1
testing2
Share
Generate Package
Search Artifacts
Save
Applications
Automations
testing1
testing2
Start
Log Message
End
100%
Automation information
Tools
Input/Output
Info
Search...
Automations
Screens
Activities
Data
Controls
Design Console (0)
Test Console (0)
Variables (1)

4.3 Trigger Automations in Attended Mode using the Project Pane

The project pane allows you to design the agent systray menu directly from the Cloud Studio.

Prerequisites

You have created one or several automations in the Cloud Studio, as described in Create an Automation [page 156].

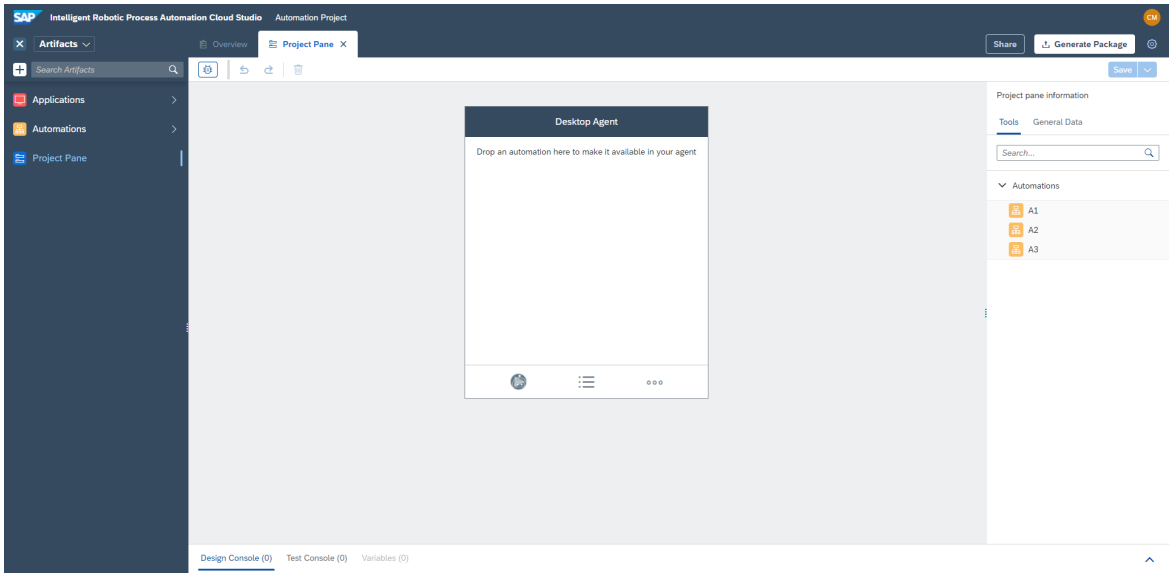
> **i Note**
>
> The project pane is available for Core SDK version 1.10 or above. To learn how to upgrade your Core SDK version, see Update a Dependency [page 530].

> **⚠ Caution**
>
> Project panes support automations with primitive data type [page 181] input parameters. Complex data type inputs are not yet supported in the project pane. Any automations with complex data type inputs in your project cannot be used in your project pane. For more information on data types, see Create and Update Variables [page 179].

Procedure

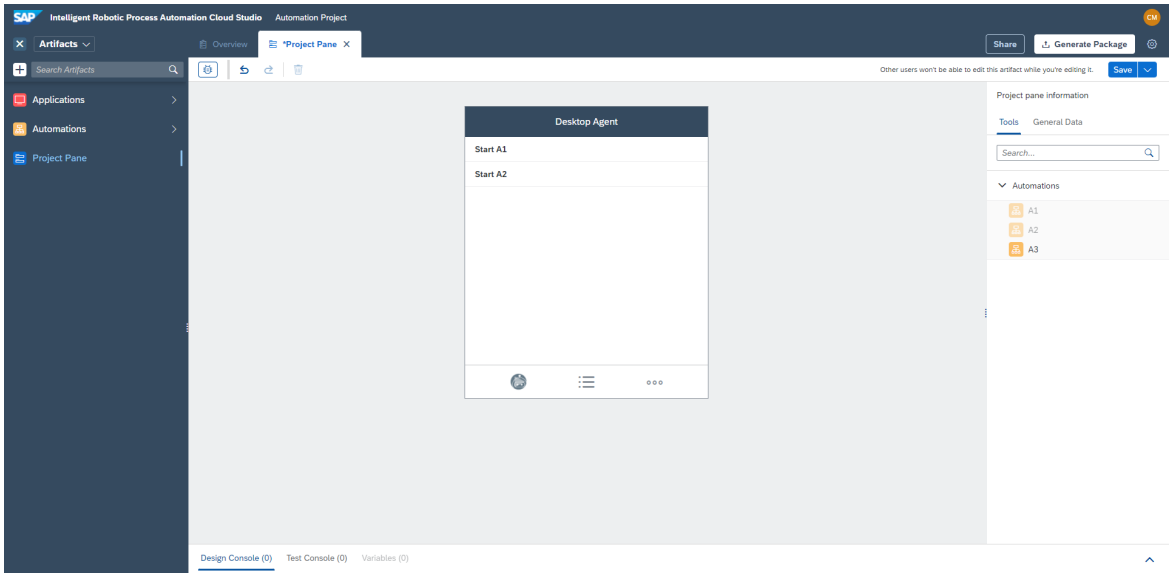
1. In the Project Explorer left-hand side panel, click the + button and then, the *Create* button. Select *Project Pane*.
   A *Create Project Pane* window appears.
2. Enter a name.
3. Click *Create*.

   > **i Note**
   >
   > You can only create one project pane for your project.



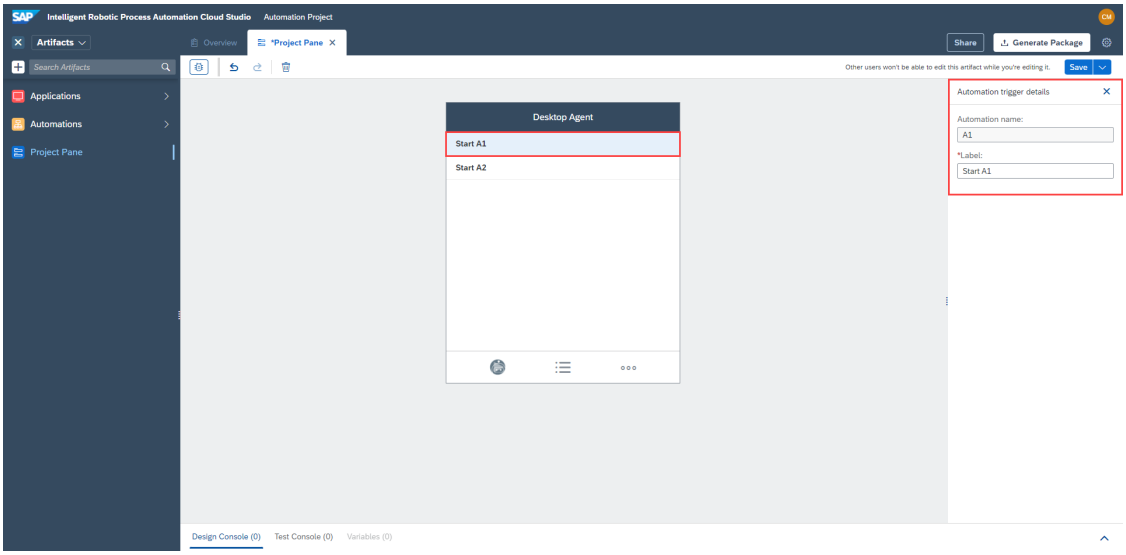
A project pane editor opens in the main panel of the Cloud Studio. On the Project pane information right-hand side panel in the *Tools* section, you can see the automations present in your project.

4. Drag and drop the automation(s) in your project pane.



> **i Note**
>
> Once you've used an automation in your project pane, it is disabled and cannot be used anymore.

5. **Optional**: Edit the label of the automation in the project pane.
    1. Select your automation in the project pane.



An automation trigger details side panel appears where you can edit the label.

> **i Note**
>
> This label is the one that appears in the systray and does not affect the label of the automation in your main project.

6. Save your project pane.

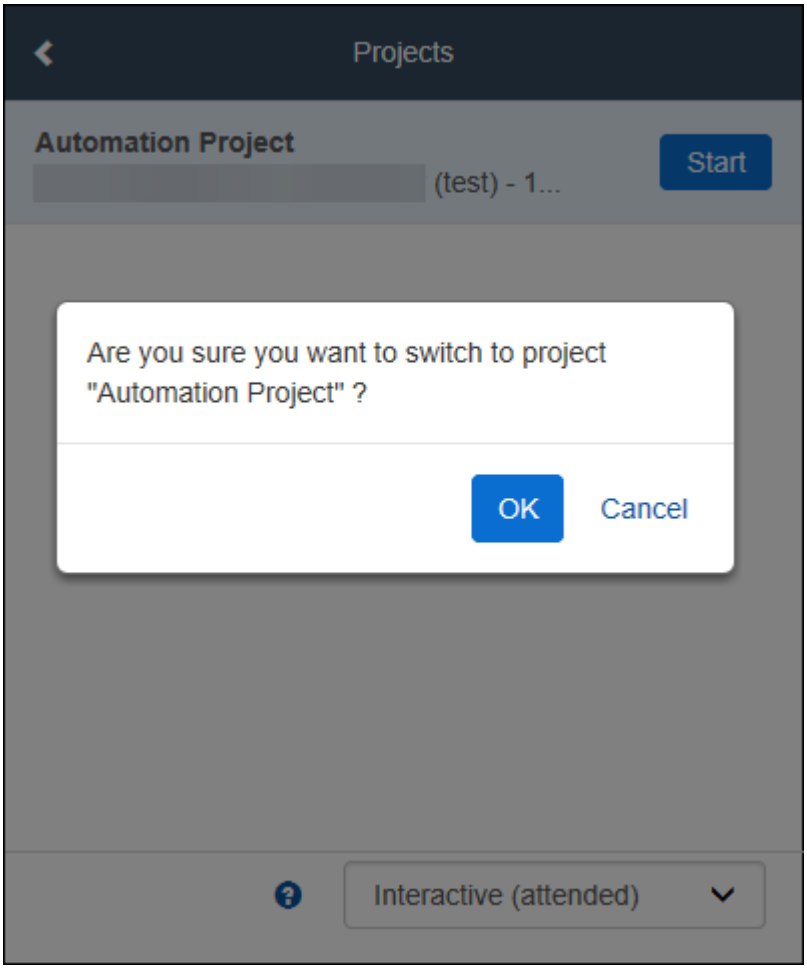
> **i Note**
>
> The project pane is a preview of what will appear in the Agent systray.

7. Generate a package from your project.
8. In the Factory, find your package and create a trigger of type attended. For more information, see Add an Attended Trigger

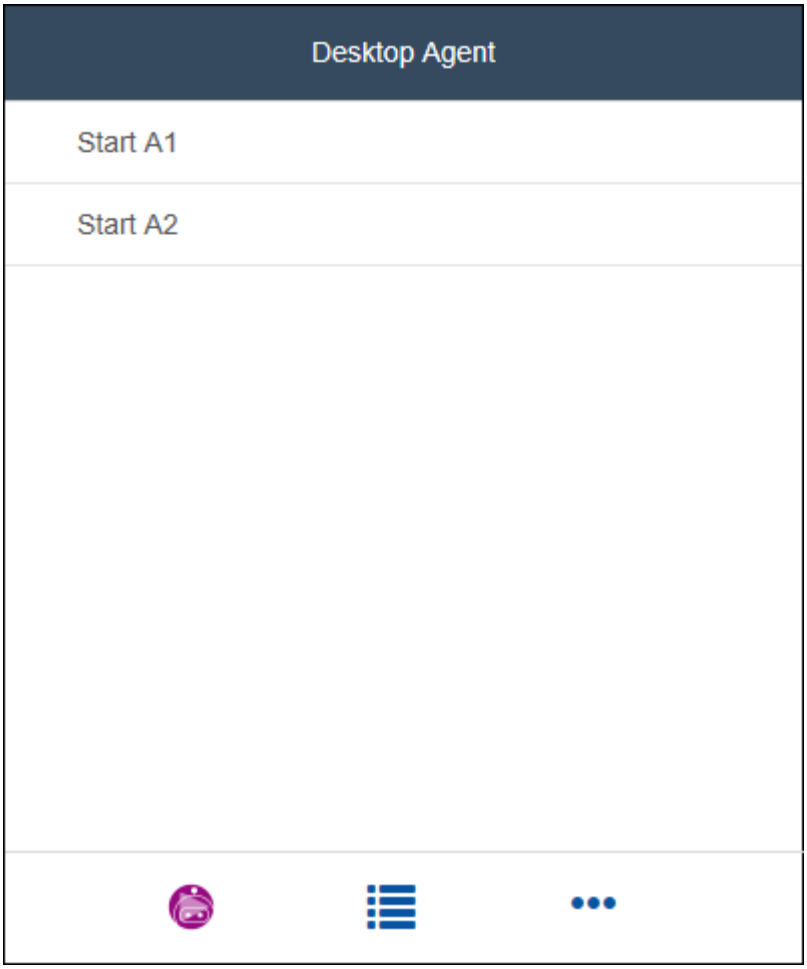
> **i Note**
>
> The project pane artifact is mandatory to create a trigger of type attended. Without a project pane, the attended option is disabled.

Once the attended trigger is planned, the project is delivered to your Agent systray.

9. Switch to your new project.
    1. In the Agent systray, click on ☰ Projects and select your project.
    2. Click on *Start*. A pop-up window appears.



3. Click *OK*.
   The project has been uploaded to your Agent systray. Once you close and open the systray again, you can see the project you previously designed in the project pane editor.



Result

You can now start the automation you added to your project pane, from your Agent systray.

4.4 Alerts

Alerts artifact allows you to define business events. Once an alert is created, it can be raised from an automation. You can use the *Raise Alert* activity from the SDK Core to raise an alert from your automation.

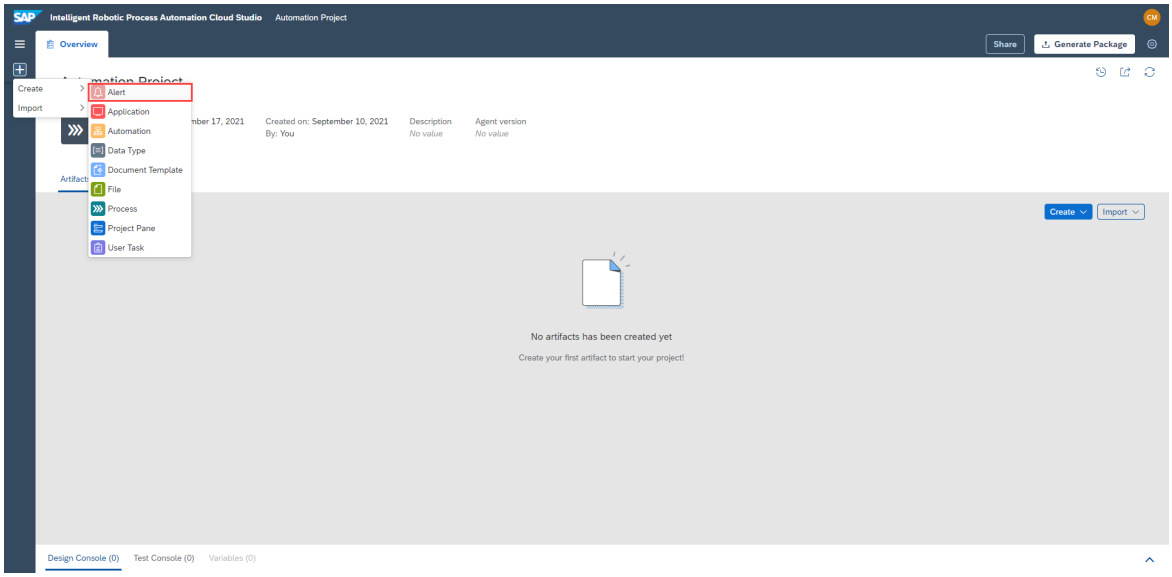
Context

Alerts allow you to raise business events in your automation and pass along information with them for an alert (such as OutOfStock or S4LoginFailed alert). After deploying a package, you can subscribe to these *Alerts* in your environment using *Alert Handlers* to get notifications by email.

> **i Note**
>
> To learn how to use Alerts, you can download the How to use Business Alerts sample on the SAP Intelligent RPA Store.

Procedure

1. In the Project Explorer, click the + button, and then the *Create* button, and select *Alert*.

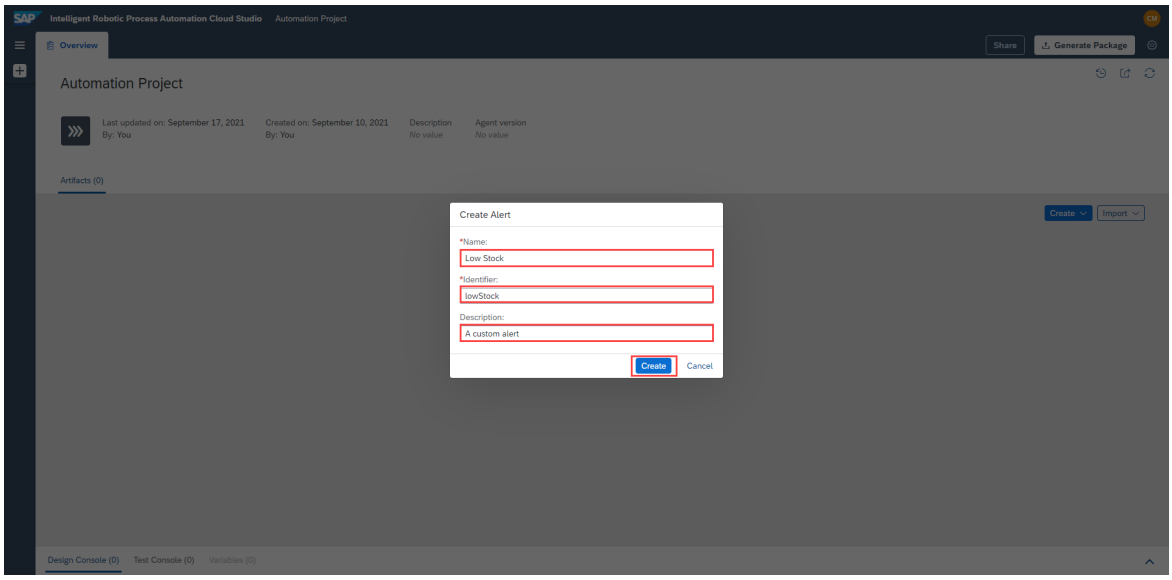


2. Enter a name in the *Name* field.

> **i Note**
>
> By default, the identifier is the same as the name. An identifier is mandatory for every artifact in the Cloud Studio, so you must manually enter an identifier if it's not already pre-filled.

3. **Optional**: Edit the *Identifier* field (without space).

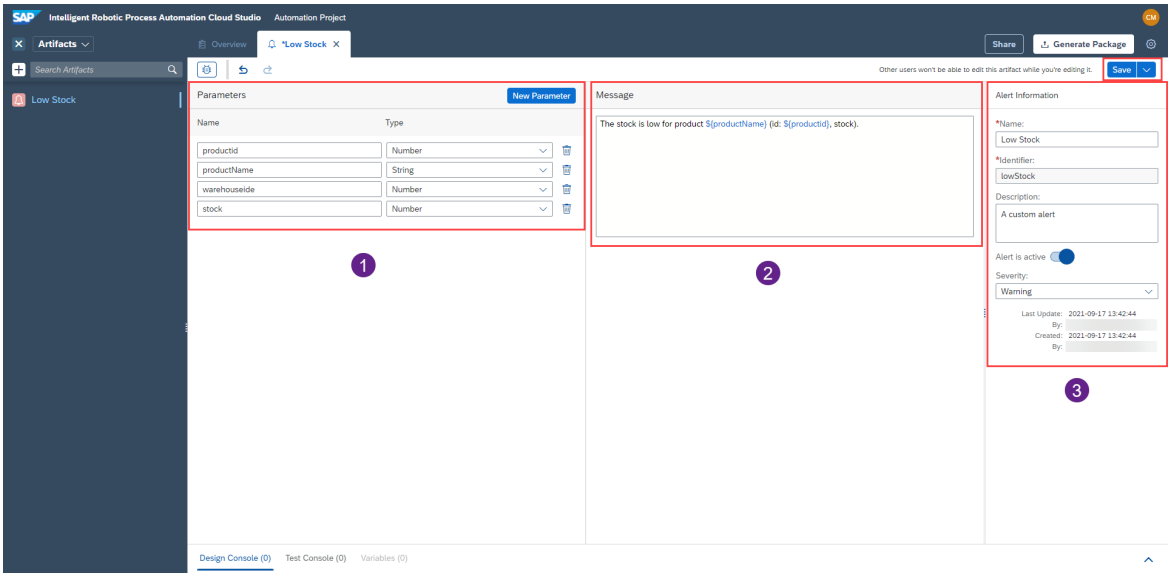
4. **Optional**: Enter a short description of the alert in the *Description* field.
5. Click *Create*.



6. A new tab opens in the main panel of the Cloud Studio. In *Parameters* (1) on the left, you can add the parameters in your alert by clicking the *New Parameter* button.

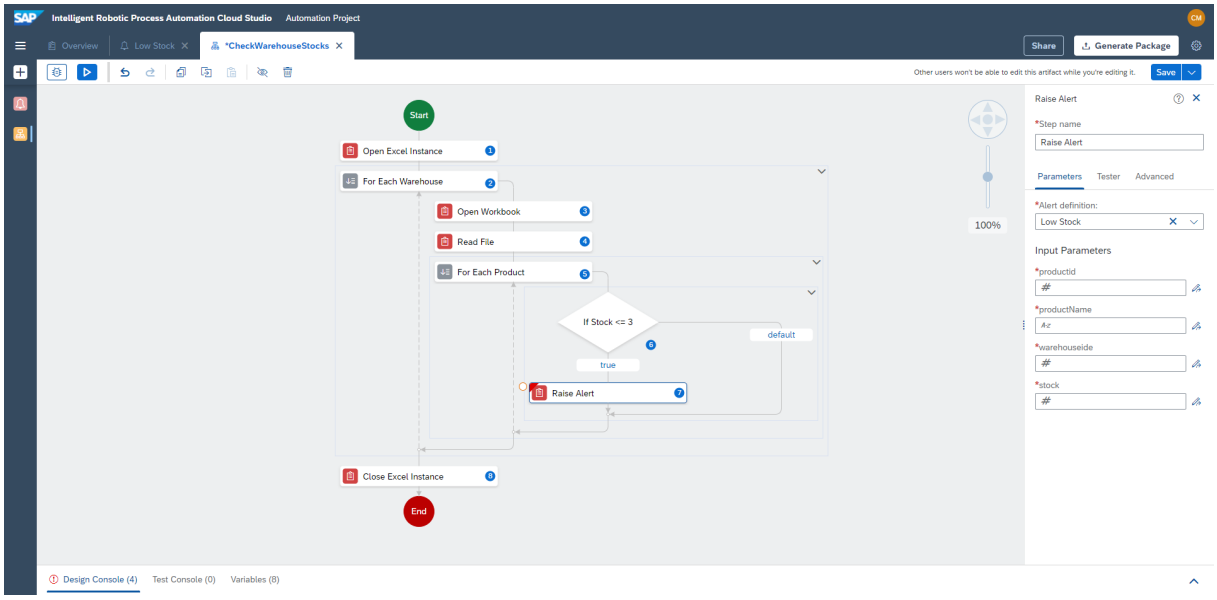
> **i Note**
>
> - Each parameter must have a unique name and must be JavaScript compliant: it must start with an alphabetical character, and subsequent characters can be a letter or a decimal digit. Space characters are not allowed.
> - Each parameter must have an identifier (*Name*), and a type (*String*, *Number*, or *Boolean*).

7. In the center *Message* field, define a parameterized message where you can use the parameters defined in the previous step. The value of the parameter will be dynamically injected during the execution of the automation while using a parameter in the message.
8. In *Alert Information* (3) on the right, *Name* field is already filled with the name of the alert.
9. **Optional:** Edit the *Name* field.
10. **Optional:** Enter a short description about the automation in the *Description* field.
11. Select the type of the *Severity*: *Information*, *Warning*, or *Error*.
12. Click *Save* to save your changes.

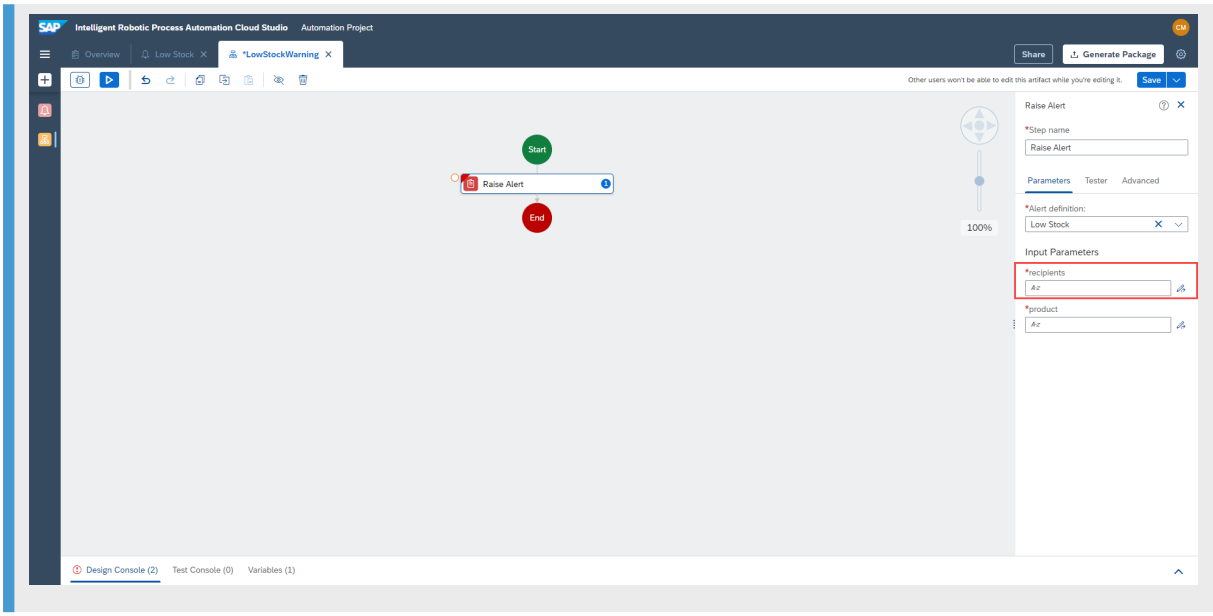


Results

You are now able to use your alert using the *Raise Alert* activity in your automation.



> **Note**
>
> If you want to use dynamic recipients in your alert handler, you need to create an alert parameter (for example, recipients) that contains the email address(es) of the recipients to which you want to send the email. In your raise alert activity that raises your alert, you need to provide the email addresses separated by a comma, space, or semicolon. The number of total recipients is limited to 50 recipients.



Next Steps

Now that you created an alert in your automation, you need to add an alert handler to an environment and link it to the alert you've created.

For more information, see Add Alert Handlers to an Environment.

4.5 Manage Files

In Cloud Studio, you can create, import, and edit a file.

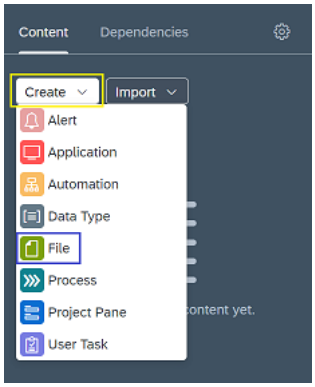
- Create a File [page 360]
- Import and Edit Files [page 362]
  - Integrate Files in Automation [page 364]
- Import Files [page 367]

4.5.1 Create a File

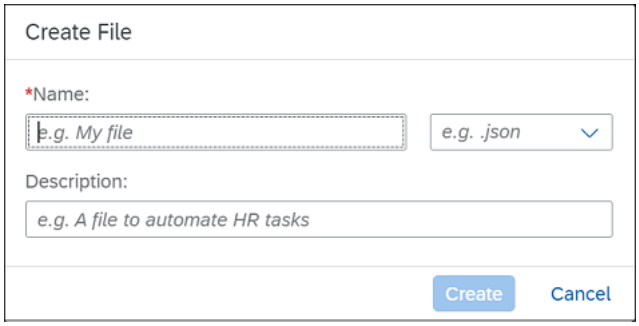
In the Cloud Studio, you can create four types of files such as Text, YAML, XML, and JSON. You can use a file content in an automation. For more details about using created files in an automation, refer to the Integrate Files in Automation [page 364] section.

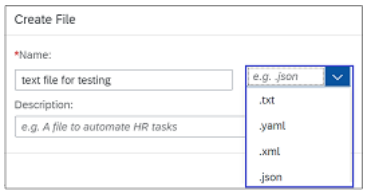
Creating a File

1. In the Project Explorer, click the *Create* button in the Project Explorer left panel and then click *Files*.



The *Create File* pop-up screen is displayed.



Field Name	Description
Name	Name and format of the file. This is a mandatory field. Provide an appropriate name and then select the file format from the drop-down list. 
Description	Description of the file. This is not a mandatory field.

2. Click *Create*. The created file opens in the main pane of the Cloud Studio.

File Features

Name	Description
Undo	Cancel or reverse the last action performed on a file.
Redo	Restore actions that were previously undone using the Undo function.
Save	Save the content of a file.

Save as	Make a copy of the file. **i** Note While making a copy of a file, by default, the file version is provided. You can provide an appropriate file name and description. You cannot change the file format.
Save and keep lock	Save the content and lock the file. This avoids other users from working on the file.
Reset	Clear the content of a file.

4.5.2 Import and Edit Files

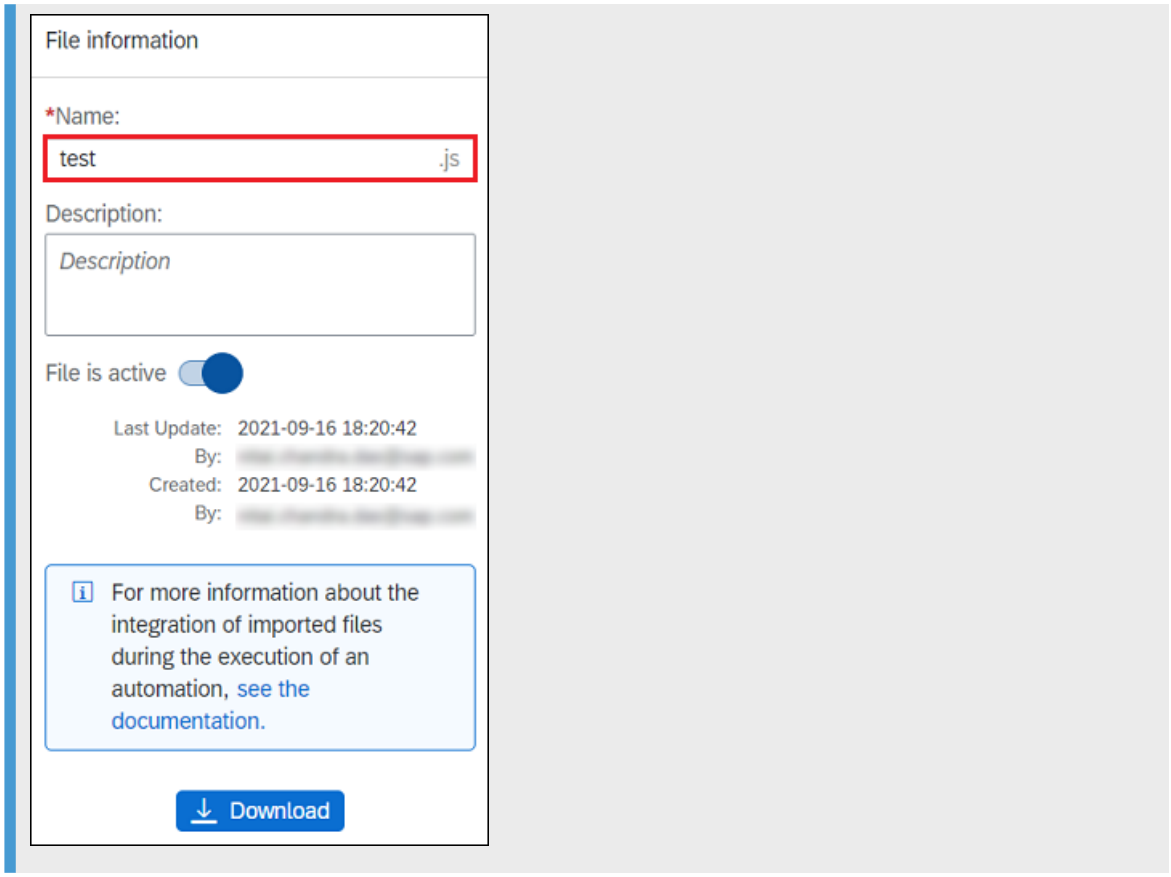
Context

You can import any file into your Cloud Studio project.

Procedure

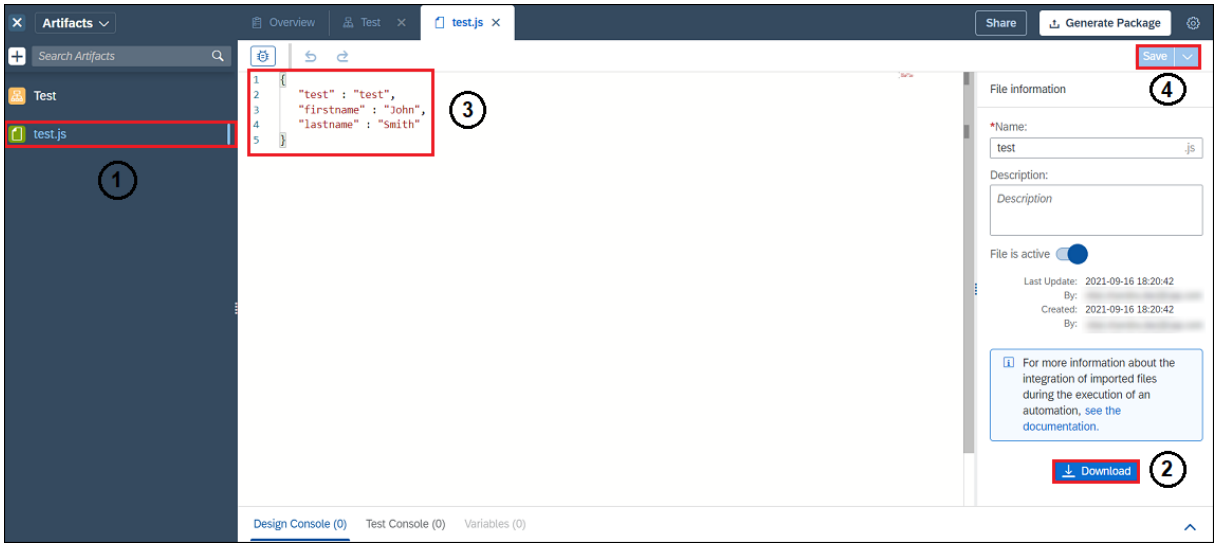
1. In the Home view or in the Project Explorer, click the *Import* ⌄ button and select *File*.
2. Find the file you want to import, enter a name and description if necessary, and click *Import*.

> **i** Note
>
> In the *Name* field, you cannot add backslash or slash before the name of the file to avoid going up in your folder hierarchy. For security reasons, you need to avoid backslash or slash in the *Name* field.



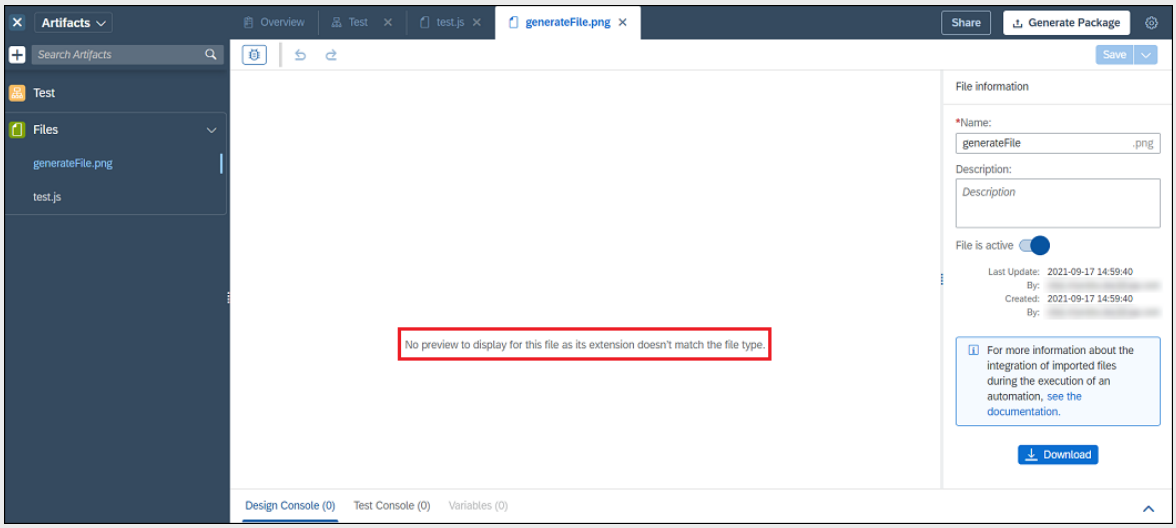
Results

The file (1) is now visible in the Project Explorer of the Cloud Studio and is available to download (2). You can also preview (3) and edit the content of the file in the central panel of the Cloud Studio. You can save (4) the file after the modification.



> **i Note**
>
> You can import different types of files such as JSON, PDF, XML, YAML, TXT, PNG, ZIP, and so on. You will be able to edit files such as JSON, XML, YAML, TXT directly in the Cloud Studio. You can also undo and redo changes using the Undo and Redo features. For the ZIP file, you can only preview the icon and name of the file.

> **i Note**
>
> You must not change the extension of any file before importing it into the Cloud Studio. For example, if you change the extension of the JSON file with some other extension and import the file to Cloud Studio, then the following message is displayed in the central panel of the Cloud Studio.



> **→ Tip**
>
> By default, the Project Explorer pane is hidden. When a user creates an artifact or imports a file, it will open. You can show or hide the Project Explorer pane at any time by clicking the ☰ button in the top left corner.

4.5.3 Integrate Files in Automation

You can use the created files [page 360] and imported files [page 362] in your automation.

Use Case Example

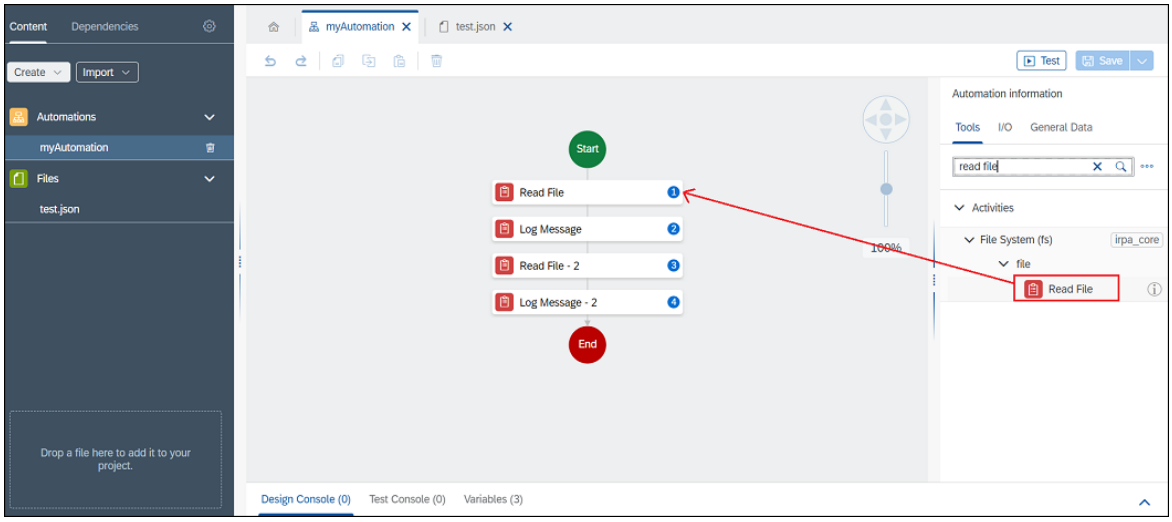
The following use case demonstrates how to integrate a file in a workflow of your automation.

Prerequisites

You have created one at least one automation in the Cloud Studio. For more details about creating automations, refer to the Create an Automation [page 156] section.

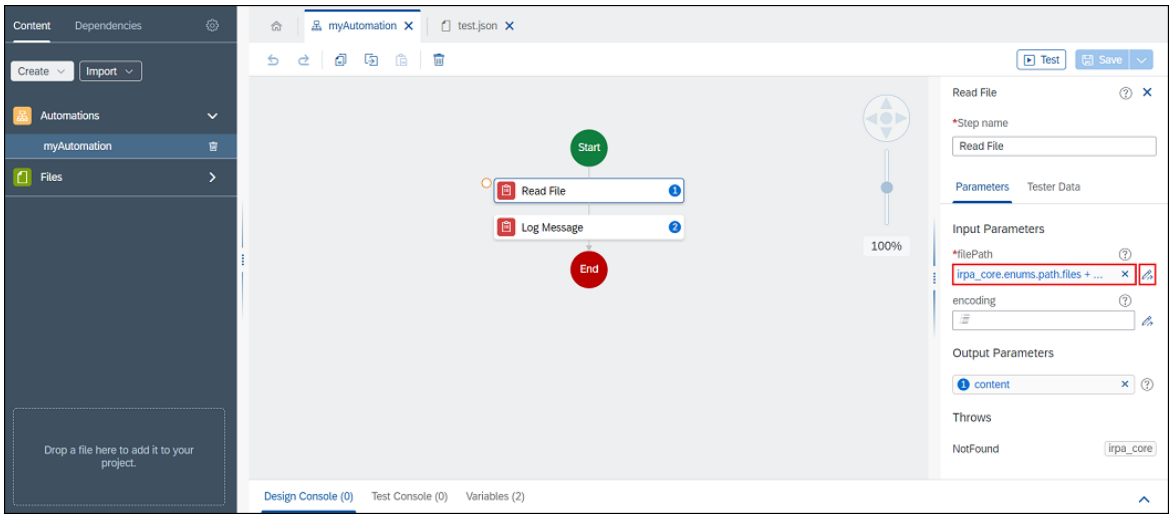
Procedure

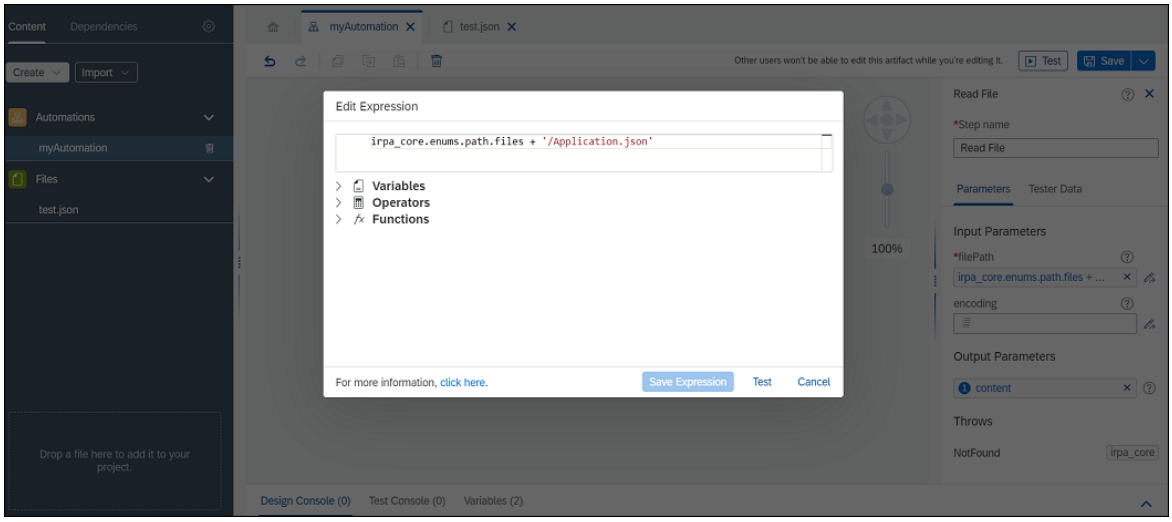
1. In your automation, drag and drop the *Read File* activity.



2. Select the *Read File* activity. On the *Read File* side panel, enter the values in the *Input Parameters*.

> **i Note**
>
> Specify the path of the file by typing the "irpa_core.enums.path.files + '/file name'". In this use case example, the path of the file is specified as "irpa_core.enums.path.files + '/Application.json'".



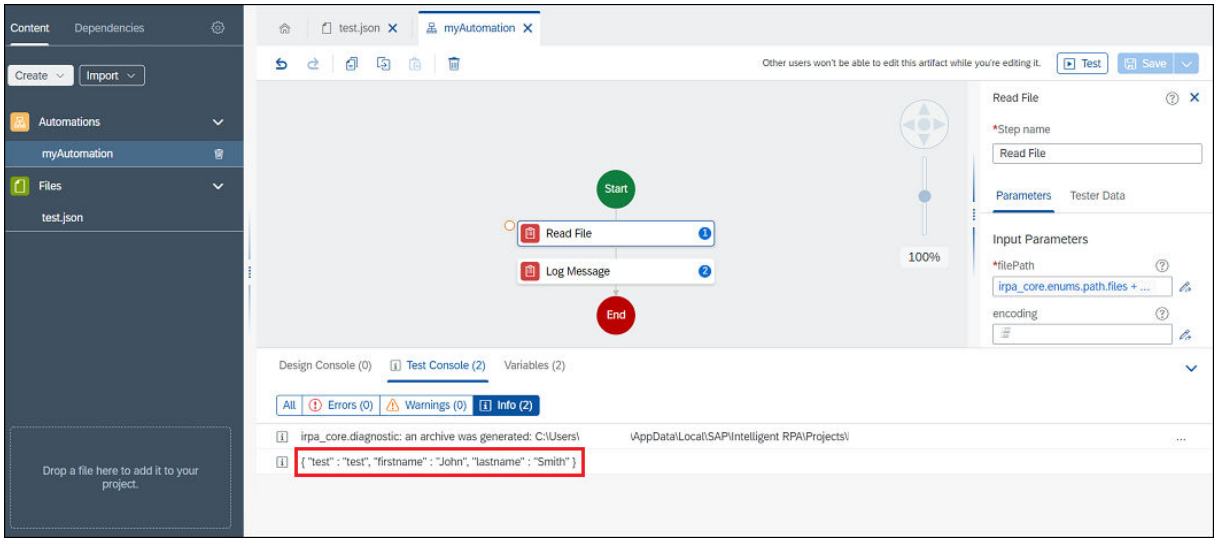


3. You can use the *Log Message* activity if you want to display the content of the JSON file at runtime.
4. Click *Save* to save your changes.
5. Click *Test*.

> **i Note**
>
> The integrity check is done only when a project is loaded from an agent. This check ensures the integrity of a file. If the file has been modified or corrupted, then the file data is retained in its original form (the file downloaded from Factory).
>
> The integrity check is not performed on consecutive executions of the same project as the project is not reloaded from the agent. If the file has been modified or corrupted, then the file data is not replaced with its original file (the file downloaded from Factory).
>
> The `\log` folder is not integrity secured `irpa_core.enums.path.log`. You can use this folder if you want to keep the files.

Results

The log message appears in the *Test Console* > *Info*.



The file is successfully used in your automation.

4.6 Import Files

Context

You can import any file into a Cloud Studio project and then reuse its content into your automation.

Procedure

1. In the Home view or in the Project Explorer, click the *Import* ⌄ button and select *File*.
2. Find the file you want to import, enter a name and description if necessary, and click *Import*.

Results

The file is now visible in the Project Explorer of the Cloud Studio and is available to download.

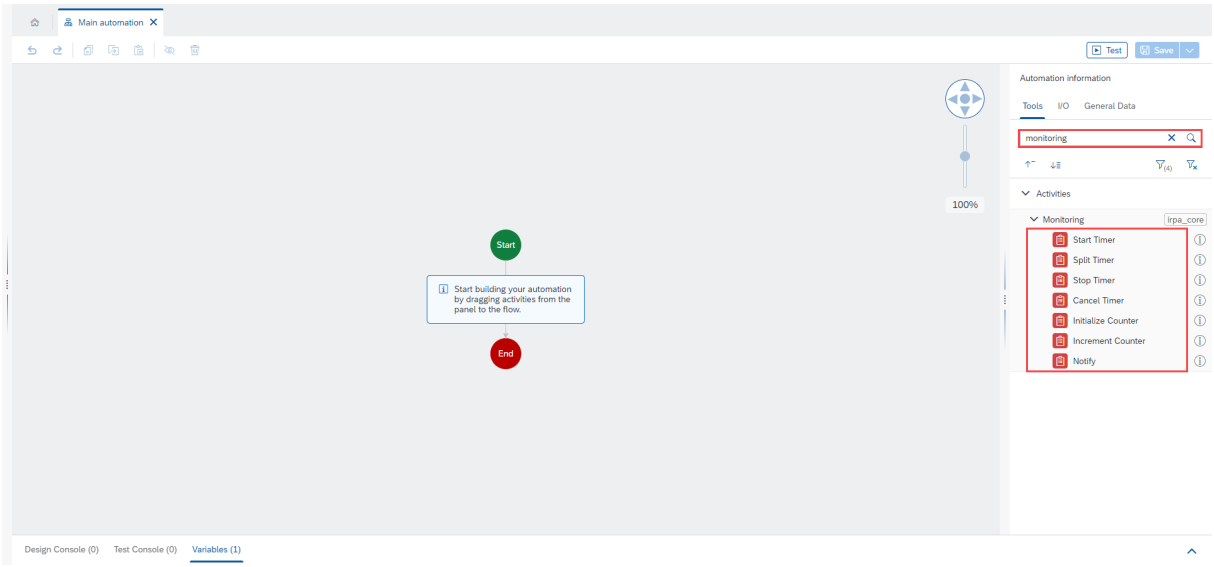
> → Tip
>
> By default, the Project Explorer pane is hidden. When you create an artifact or import a file, it will open. You can show or hide the Project Explorer pane at any time by clicking the ☰ button in the top left corner.

4.7 Business Activity Monitoring (BAM)

Similar to the Business Activity Monitoring in the Desktop Studio (V1), BAM in the Cloud Studio (V2) lets you measure the activities performed in your automation by sending data like counters and timers that you insert at chosen points in your automation. In the Cloud Factory, you can download the BAM data in a CVS file.

Use BAM in your automation

On the *Automation information* side panel, type **`monitoring`** in the search bar.



In the *Monitoring* section, you can find the different Business Activity Monitoring timer, counter and notify activities .

Timer activities

The different types of timer activities:

- **Start timer**: the start timer allows you to start a timer.
  In the input parameters you must enter an ID.
- **Split timer**: the split timer allows the Desktop Agent to send a message counting how much time passed between the execution of the start timer and the execution of the split timer.
  In the input parameters, you must enter an ID matching the ID of a previously defined start timer.
- **Stop timer**: the stop timer will send the duration from the start timer to the Factory as data and then it will reset the timer to zero.
  In the input parameters you must enter an ID matching the ID of a previously defined start timer.
- **Cancel timer**: the cancel timer will stop the timer without sending any data to the Factory.
  In the input parameter you must enter an ID matching the ID of a previously defined start timer.

Counter activities

The different types of counter activities:

- **Initialize counter**: the initialize counter starts a counter with an initial value.
  In the input parameters of the counter you must enter
  - an ID
  - an initial value for the counter.

    > **Note**
    >
    > By default, the initial value is set to 0.

- **Increment counter**: the increment counter allows the Desktop Agent to count how many times a specific activity occurs.
  In the input parameters of the counter you must enter
  - an ID matching the ID of a previously defined initialize counter
  - an increment value

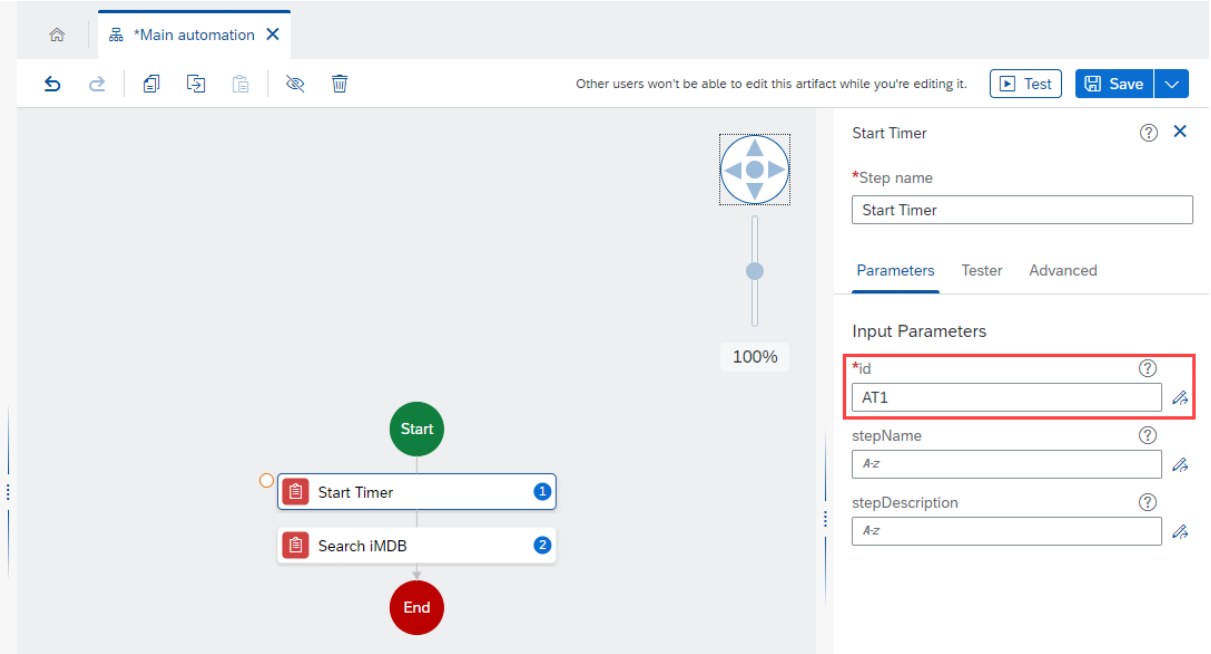
    > **Note**
    >
    > By default, the increment value is set to 1.

Notify activity

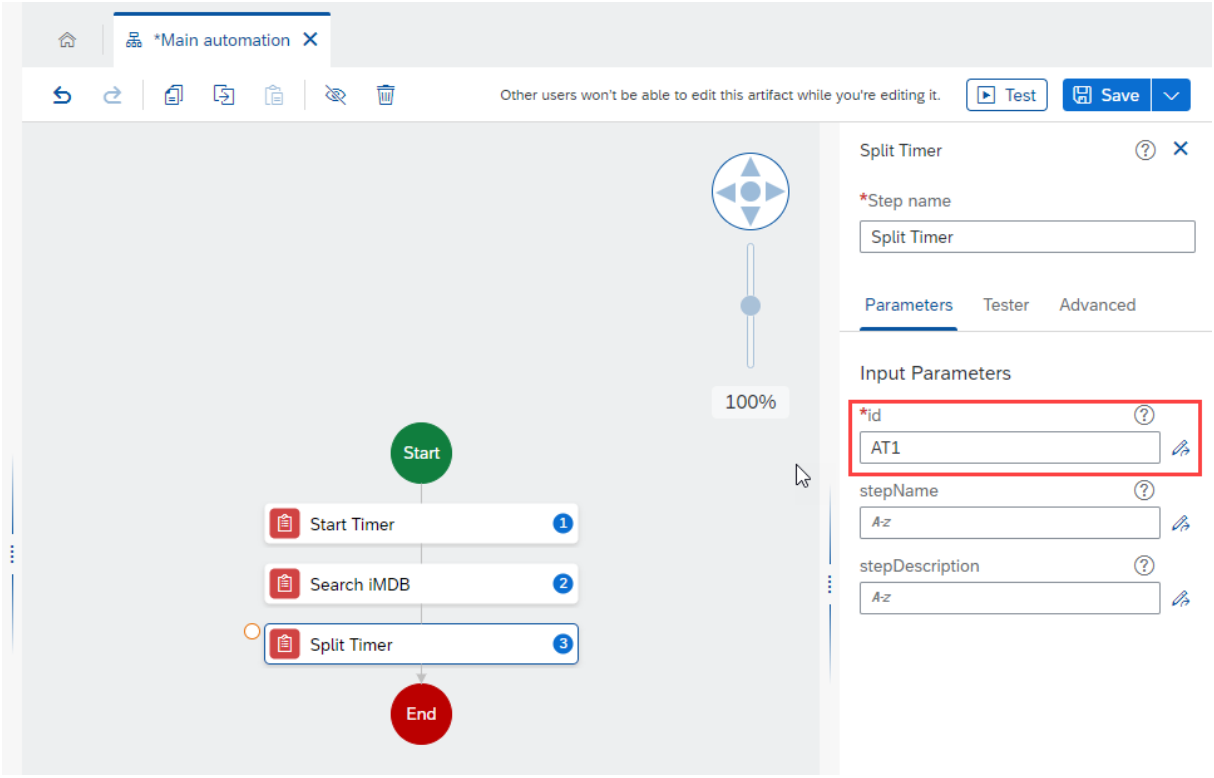
The notify activity sends a notification message to the Factory. In the input parameters you must enter

- a message
- the severity of the message ("info" , "warning" , "error")

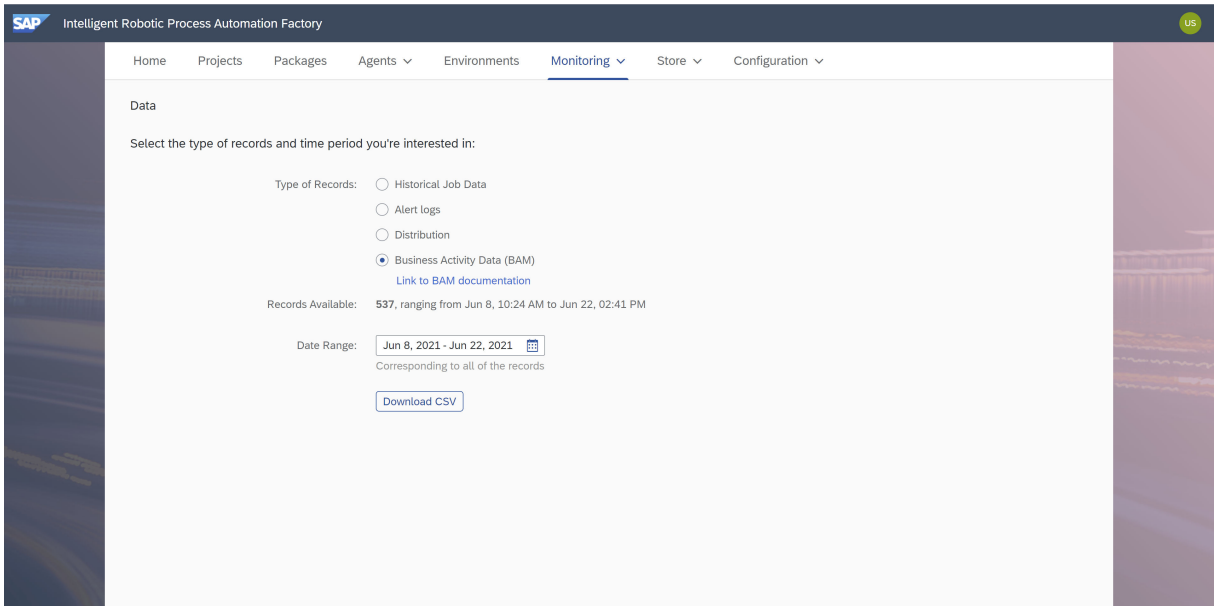
Drag and drop the *Start timer* in the workflow of the automation. In the input parameters of the statement, enter an ID. Add any activity following the timer in the workflow of the automation.



Drag and drop a *Split timer* in the workflow. In the input parameters of the activity, enter the same ID you entered for the start timer.



Once you have tested your automation, go to the Factory on the *Monitoring* tab and click *Data*. Select *Business Activity Data (BAM)* in the type of records.

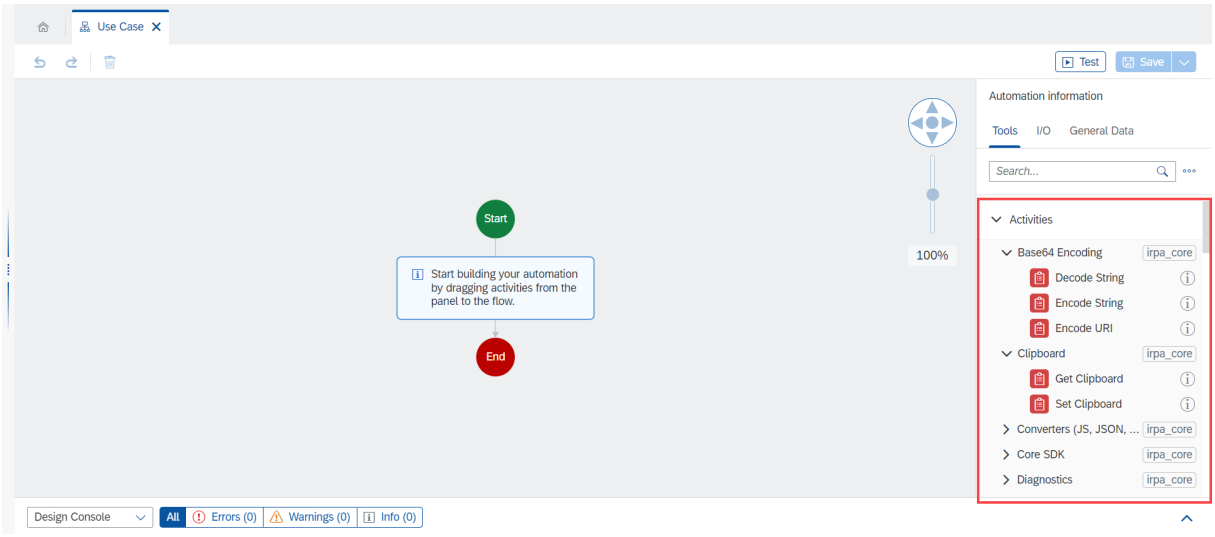


Click *Download CSV*. A file containing all the data with the measurements of your automation activities is downloaded on your computer.

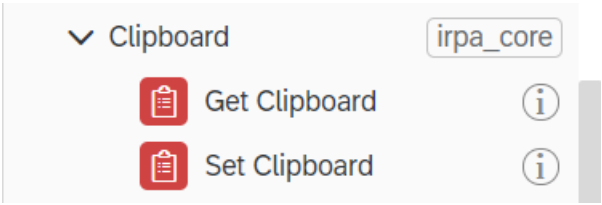
5 Automation Activities provided by SDK Packages

In the Cloud Studio, activities are provided with the SDK packages you import the first time you create an automation, to build the workflow of your automation.

Activities are organized in classes and subclasses in the tools section on the side panel of the automation.



Each activity executes an action according to the class it belongs to. The following example displays the activities in the **irpa-core** subclass activity **Clipboard**.



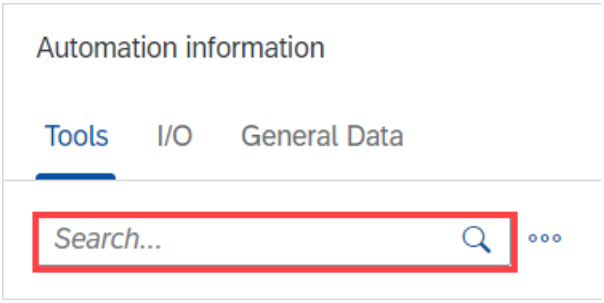
The activities in this subclass perform actions linked to the clipboard:

- **Get Clipboard**: Retrive the textual content of the clipboard.
- **Set Clipboard**: Clear the clipboard and then set text on clipboard.

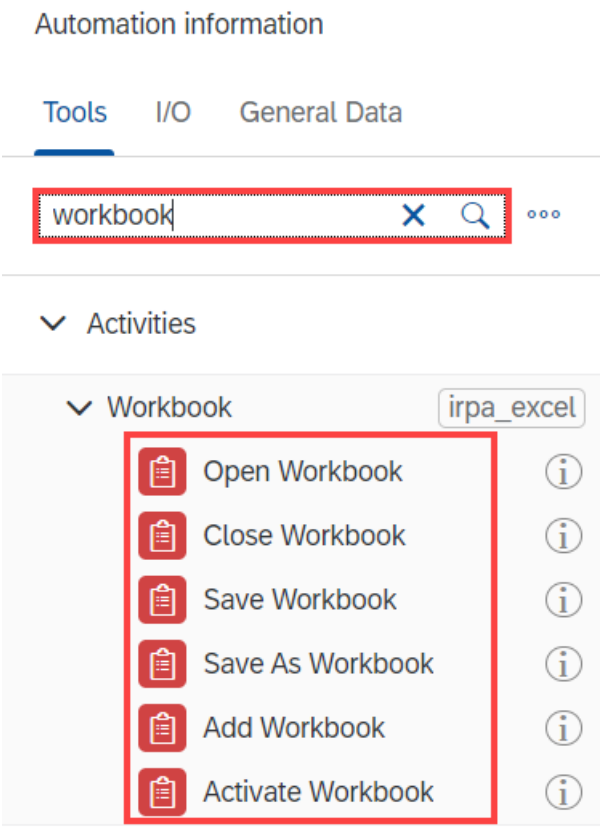
To build your automation you can drag and drop the activities in your workflow where they become the steps of your automation. Each activitiy has different parameters that you can define such as the name of the step, and the input and output parameters.

Filtering Feature

To help you look for an activity, the Cloud Studio provides a filtering feature.



The filtering feature displays the tools from the *Tools* side panel that match your search text. The following example displays the filtering feature when you search for a workbook activity.



5.1 SDK Packages

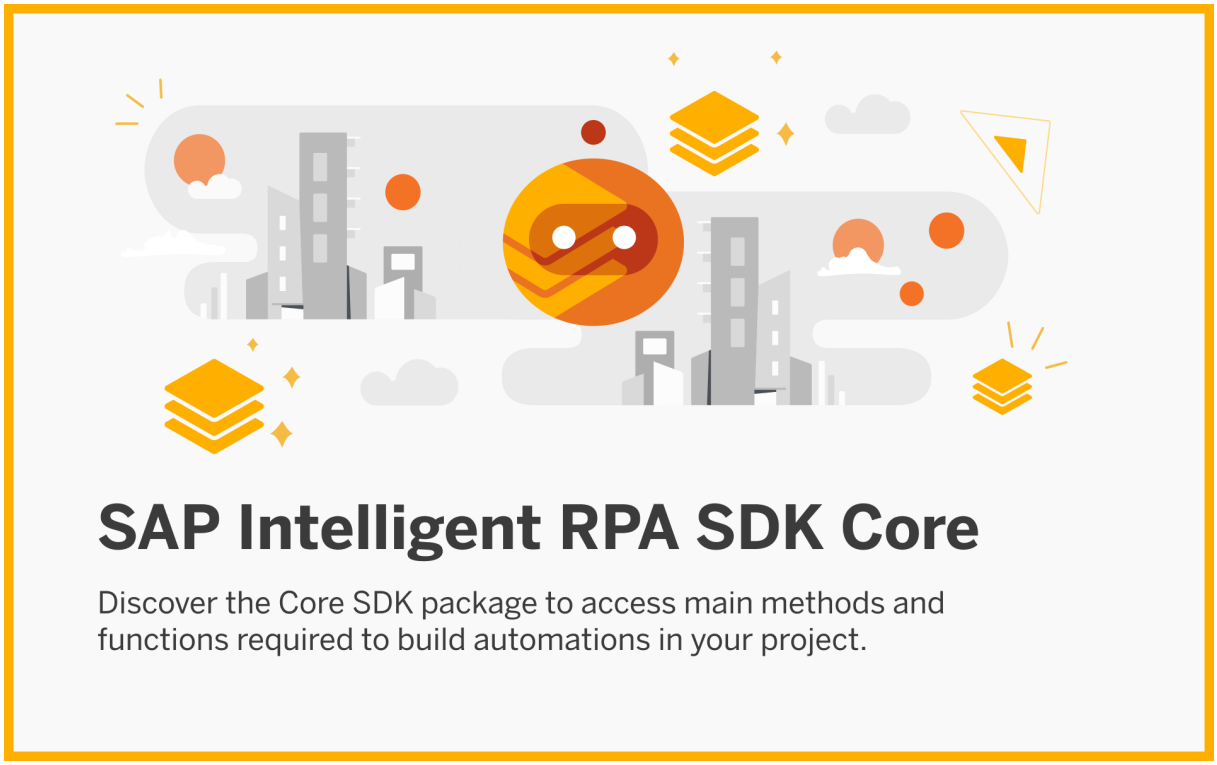
The SDK packages and their contents (activities, classes, modules and data types) are only available in ENGLISH.

- SDK Core Package
- SDK SAPUI5 Package

- SDK SAP WebGUI Package
- SDK Excel Package
  - Best Practices for MS Excel Activities [page 405]
  - Excel Data Mapping [page 412]
- SDK Word Package
  - Best Practices for MS Word Activities [page 418]
- SDK Outlook Package
  - Best Practices for MS Outlook Activities [page 425]
    - Delete Multiple Emails in Outlook [page 437]
- SDK PDF Package
  - Use a Regular Expression [page 448]
- SDK BAPI Package
  - Best Practices for SAP BAPI Activities [page 467]

5.1.1 SDK Core Package

The Core SDK package contains main methods and functions you use to build and run your automation projects.



- https://help.sap.com/doc/d69213c343964f97af7829a37f4c6d4d/Cloud/en-US/SDKdocumentation.html [https://help.sap.com/doc/d69213c343964f97af7829a37f4c6d4d/Cloud/en-US/SDKdocumentation.html]

Integrate Core Activities

The Core SDK package is composed of the most frequent activities used in SAP Intelligent RPA.

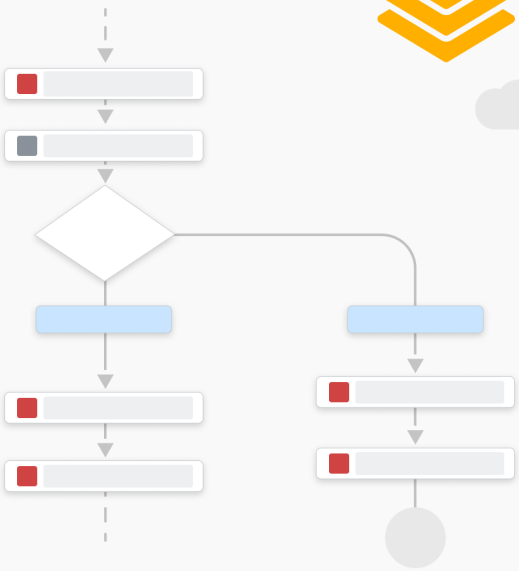


- Mouse Left Click
- Copy (FIle/Folder)
- Keystroke
- Log Message
- Wait

+ Many more...

Build Powerful Workflows

Added by default when you first create an automation in the Cloud Studio, the activities of the Core SDK package will give you access to screens and UI elements.



5.1.1.1 Local Agent Variables

> **Note**
>
> To learn how to use local agent variables, you can download the How to use local variables and credentials sample on the SAP Intelligent RPA Store.

About Local Agent Variables

A set of activities called "Local Agent Variables" allow to create, update, retrieve and delete local variables. Such variables are then stored within the Windows registry of the local computer (inside a dedicated registry, under a dedicated key).

For a given project, up to 100 variables can be created.

For a given local agent, up to 100 shared variables can be created.

What is the use case for Local Agent Variables?

Local agent variables are variables that are partly persisted (half-life) within the local computer. While Automation variables are defined for the whole life of the automation execution, each time the automation starts the variables are reset. Local variables aim to keep a state between the start of the same automation within the same project or beyond the project. For instance, let's say that you want to count the number of times the automation starts. First, you need to create a local variable called "myAutomationCounter". Then you will increment the variable each time the automation starts. In the end, you can retrieve the values.

What kind of information can be stored?

We strongly advise not to store personal or password-related information. Also, a variable is not meant to store a huge amount of information. Please use a file instead to store a large amount of information. To know more about the size limits for the various registry elements, see Registry Element Size Limits It is also important to note that misusing the Windows registry can lead to workstation stability problems.

Scope of Local Agent Variables

If the variables that are not "protected" are "shared" within the local agent, you must check within your automation whether these variables are compliant with the rest of the automation flow. You can leverage cipher/decipher before storing the content if needed.

5.1.1.1.1 Best Practices: Retrieve a Value from a Local Agent Variable

Context

Follow this procedure to retrieve the value of a local agent variable.

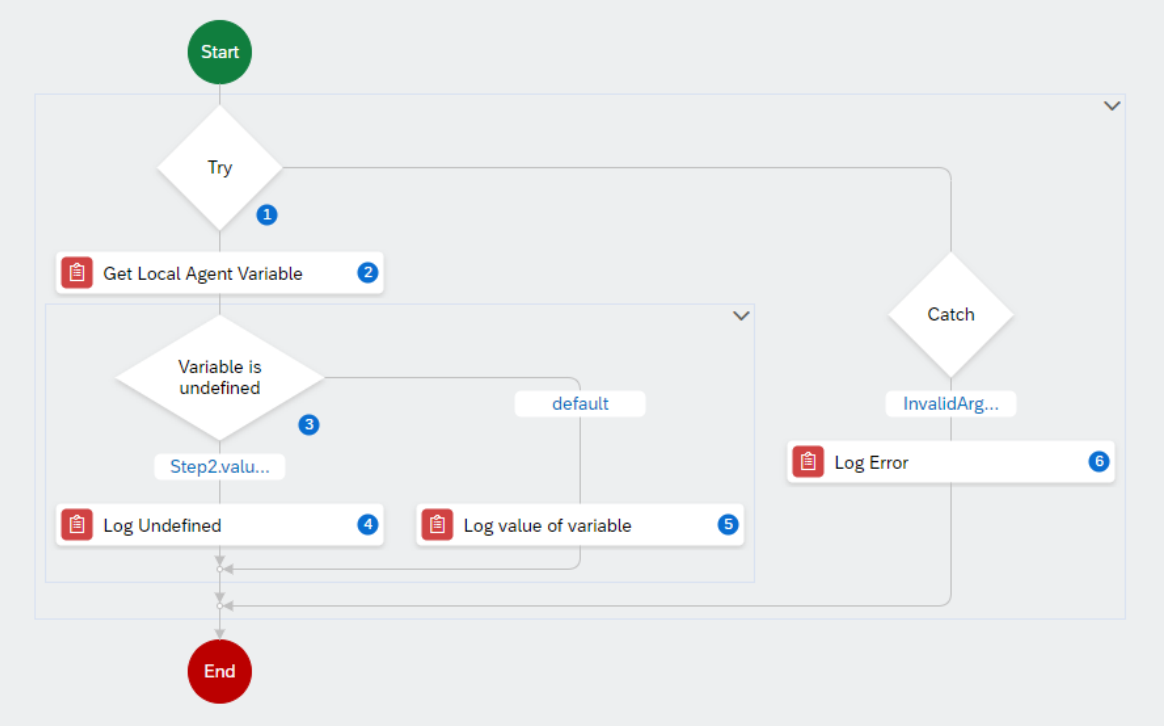
Procedure

1. Add a Try control [page 239] to your automation.

   Always use a *Try* control as all the activities related to the local agent variables can throw an `InvalidArgument` exception.
2. Add the *Get Local Agent Variable* activity to the *Try* branch and set the required parameters.
3. **Optional:** Define a value for the `defaultValue` field.

   > → Tip
   >
   > We recommend leaving the value empty for main use cases.
4. Add a Condition control [page 231] right after the *Get Local Agent Variable* activity to check the value the agent retrieved.

   If the `defaultValue` was left empty, you must check if the value is equal to `undefined`: it would mean that the value couldn't be retrieved. If not, it means that the value was correctly retrieved, and that you can use it in the workflow.

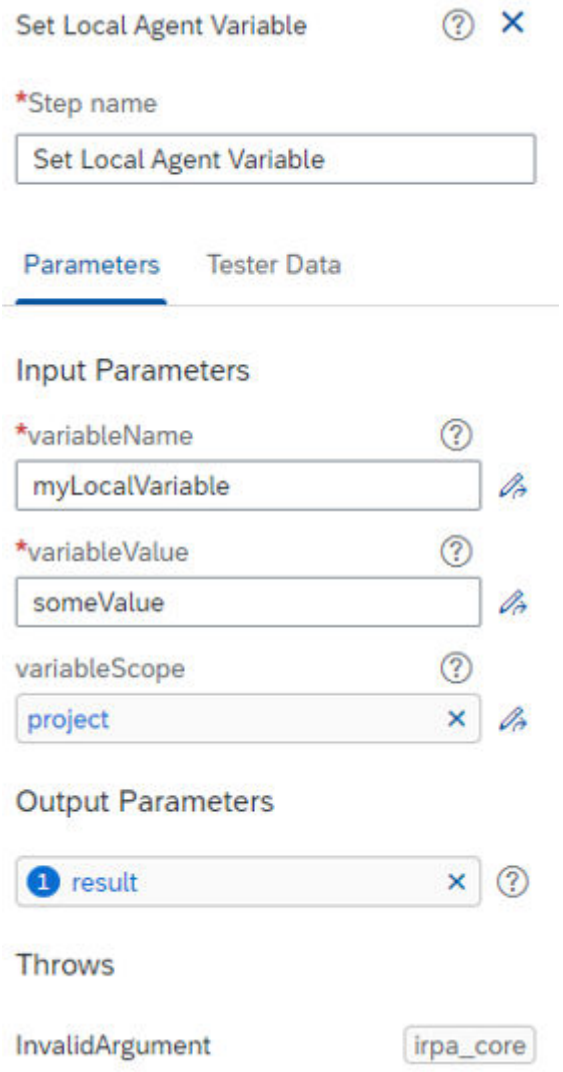


5.1.1.1.2 Best Practices: Retrieve a Value from a Local Agent Variable Created in a Dependency

Context

Follow this procedure to retrieve the value of a local agent variable created or updated in a project used as a dependency.

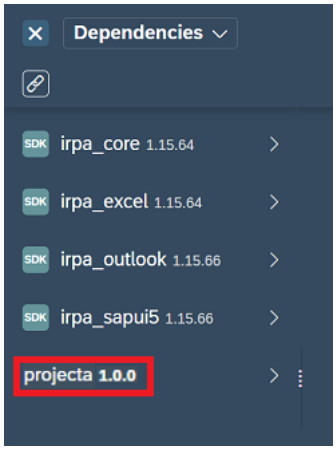
In **Project A**, there is an automation where the *Set Local Agent Variable* activity is using the following parameters:



In **Project B**, **Project A** is used as a dependency.

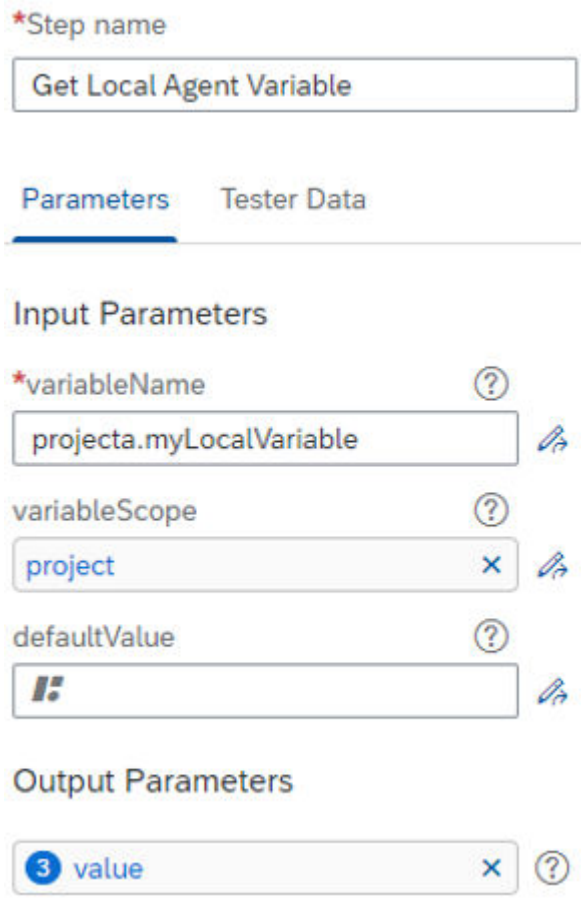
Procedure

1. In **Project B**, add **Project A** as a dependency.
2. Save the identifier of this dependency.



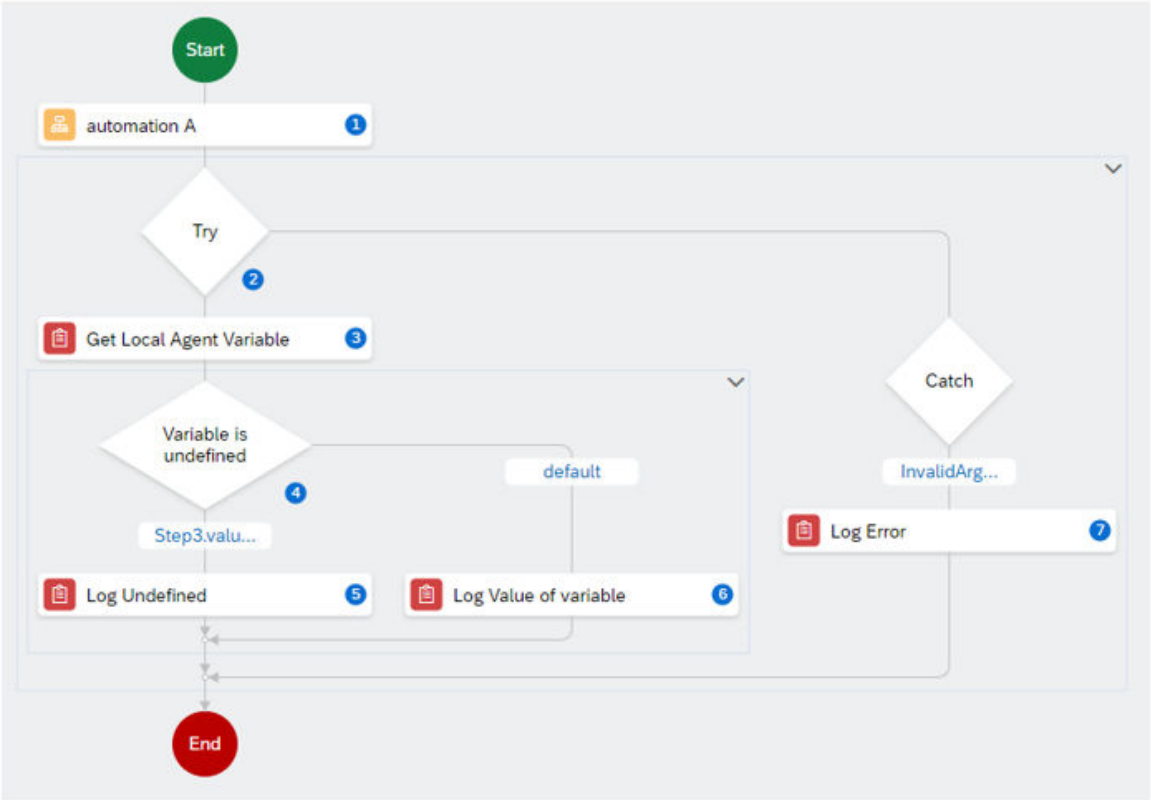
3. Create an automation and insert the nested automation from **Project A** where the local variable is set.
4. Add the activities used to read the value from the local agent variable to the automation.

   For more information about these activities, see Best Practices: Retrieve a Value from a Local Agent Variable [page 375].
5. Select the **Get Local Agent Variable** activity and set the parameters.
6. Enter the name of the variable. Then, select *project* in the *variableScope* and *Value* under *Output Parameters*.



→ Tip

You can get the value of the variable using the following function for the *variableName* parameter: `projectIdentifier.variableName`.



5.1.1.2 Local Agent Credentials

> **i Note**
>
> To learn how to use local agent credentials, you can download the How to use local variables and credentials sample on the SAP Intelligent RPA Store.

About Local Agent Credentials

A set of activities called "Local Agent Credentials" allow to create, update, and retrieve local credentials in the catalog Generic Credentials leveraging Windows Credential Manager. For more information about Windows Credential Manager, please visit the Microsoft website.

The following are the definitions of location:

- CURRENTUSER: The credential persists for the life of the logon session. It will not be visible to other logon sessions of this same user. It will not exist after this user logs off and back on.
- LOCALMACHINE: The credential persists for all subsequent logon sessions on this same computer. It is visible to other logon sessions of this same user on this same computer and not visible to logon sessions for this user on other computers.

For more information, see the Credential Structure .

What is the prefix?

By default, all credentials that are being created, updated, or retrieved will have a prefix. The prefix is "SAPIntelligentRPA_". The prefix can be changed to accommodate your need. If you don't want to use a prefix, use the Empty Text value. However, you must pay attention not to remove, update or retrieve unexpected personal information.

What is the use case for Local Agent Credentials?

The main use case is to store securely credentials information and retrieve this information to fill when needed other applications with such credentials.

What kind of information can be stored?

Usually, the credentials are a combination of "login" (variable of type "String") and "password" (variable of type "Password").

5.1.1.2.1 Best Practices: Retrieve the Value from a Local Agent Credential

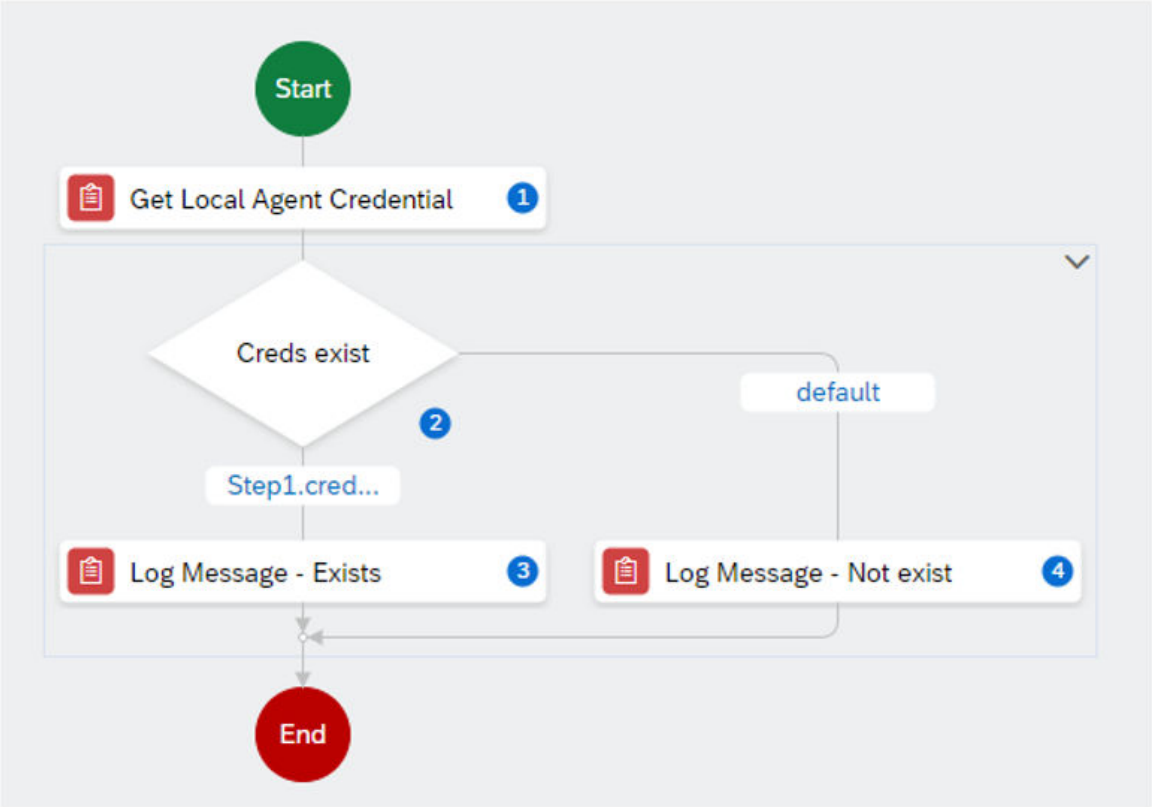
Context

Follow this procedure to get the value of a credential stored in the **Windows Credentials Manager**.

Procedure

1. Insert the *Get Local Agent Credential* activity and set the required parameters.
2. Use a condition to check the value the agent retrieved.

> **i** Note
>
> If the value is `null`, it means that the credential could not be retrieved.



5.1.1.3 Best Practices for Core Activities

Activate Screen Activity

Purpose

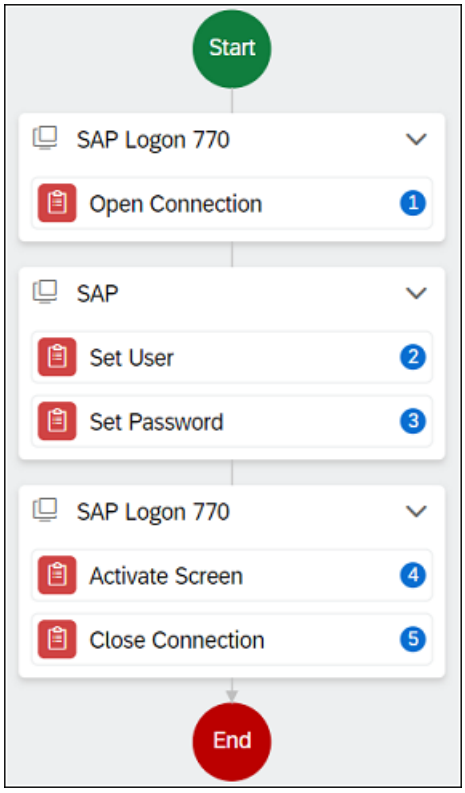
Use the **Activate Screen** activity to activate the screen of an application. This activity brings the screen to the front. For the web technology, it also brings the tab to the top.

Best Practice

Must be used before an activity on the screen, especially if there is human activity on the agent during the automation execution.

Use Case Example

In the following use case example of the automation, you do the **Activate Screen** activity after a human interaction to ensure that the screen is targeted.



Execute Command Activity

Purpose

Use the *Execute Command* activity to launch applications or execute commands during the execution of your workflow.

Best Practice

The *Execute Command* activity performs a `ctx.shellexec` and has the following input parameters:

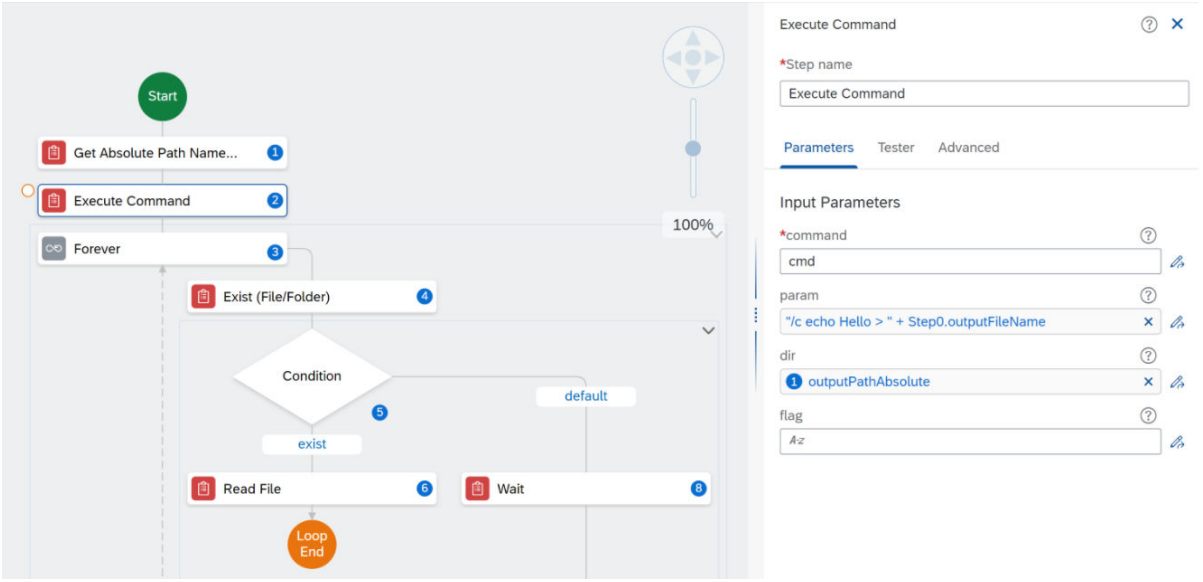
- *command*
- *param*
- *dir*
- *flag*

To retrieve the output of the activity, you can redirect the output to a file and then, read this file.

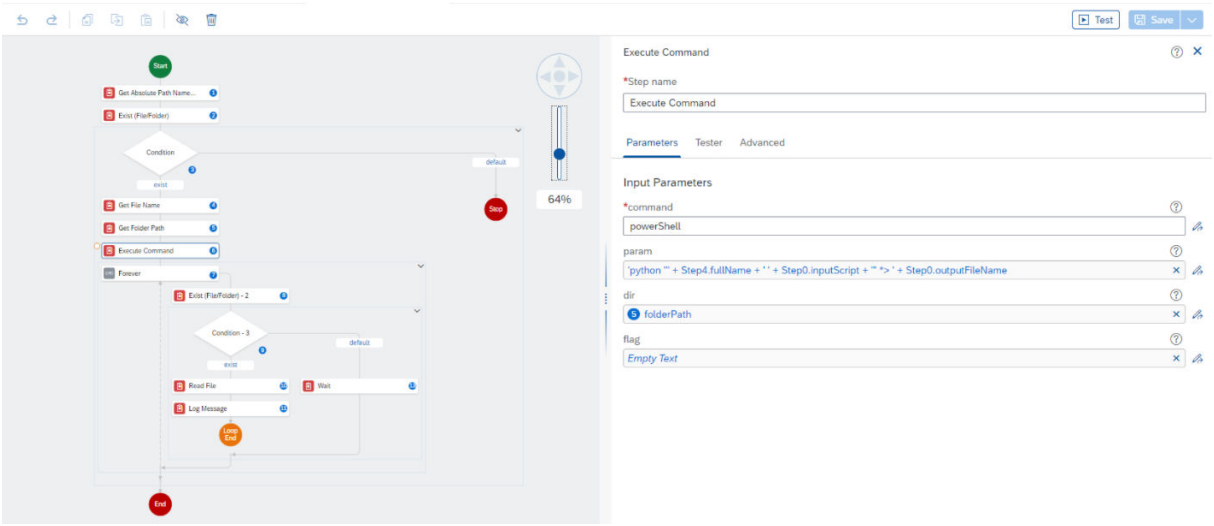
Use Case Examples

In the following use case example, you use the *Execute Command* activity.

For example, you can write the string `Hello World` in an output file using the `echo` command in the *param* field.



In this other use case example, you can use the *Execute Command* activity to launch a python script from a powerShell and redirect the output in a file.



WMI Query Activity

Purpose

Use *WMI Query* activity to execute a WMI query in your Cloud Studio projects. For more information on the WMI query, you can read the Microsoft documentation .

Best Practice

The *WMI Query* activity has the following input parameters:

- *section*

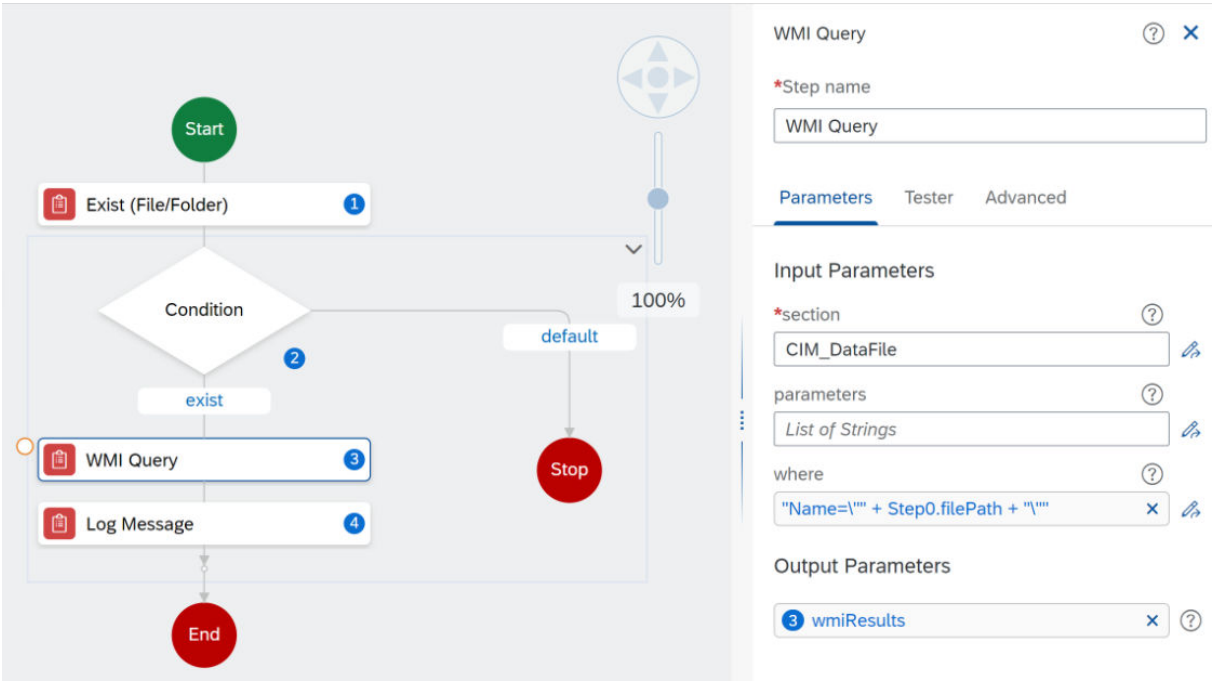
- *parameters*
- *where*

To retrieve the name of a file using the *WMI Query* activity, get the name of the file in the *section* field, and its path in the *where* field.

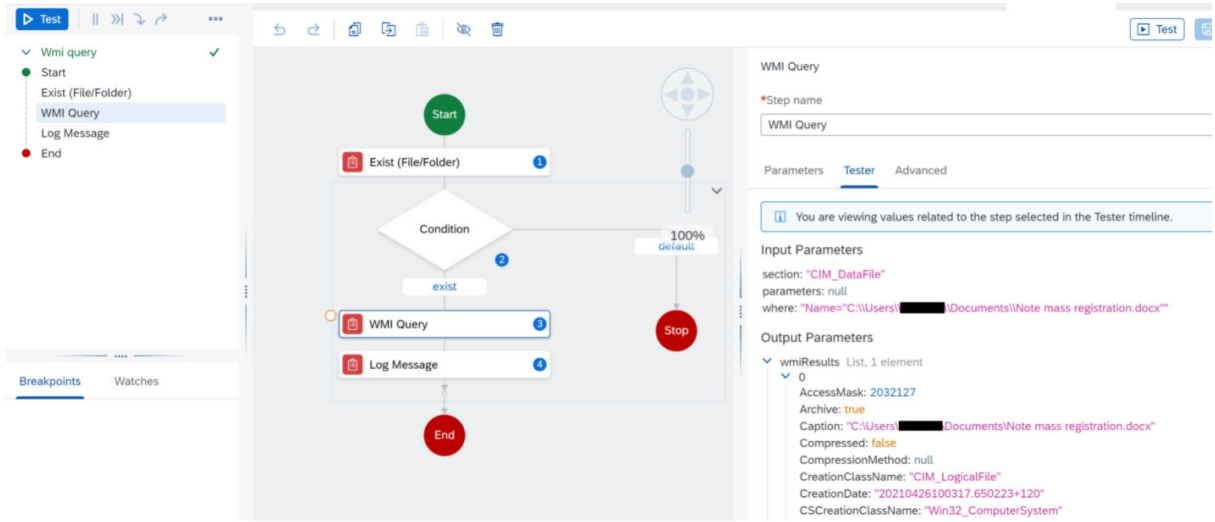
The file path entered in the *where* field must contain double backslashes. For example, you must write **“C:\\Documents\\example.txt”**.

Use Case Example

The following screenshot shows an example of an automation that contains the *WMI Query* activity and its *Input Parameters*:



The following screenshot shows the data displayed in the *Tester* panel:



5.1.1.4 Best Practices for Wait Activities

Wait

Wait activity holds you between activities for a specified amount of time before they move to the next activity. This activity can be useful when some screen elements cannot be detected, or some UI interactions are too fast. The overall duration of the automation will be increased by adding the **Wait** activity.

The different types of wait activity are as follows:

Wait (Screen)

Purpose

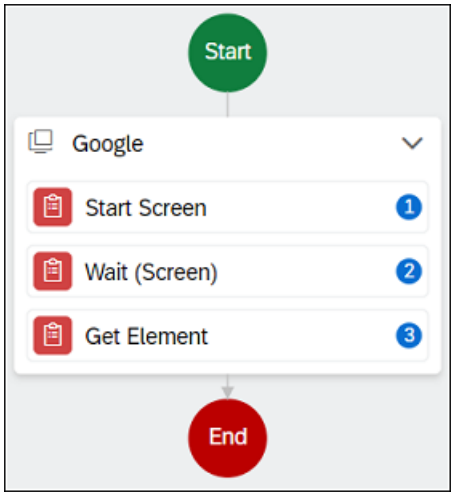
Use the **Wait (Screen)** activity until a screen is present and loaded, and then call a callback. If the screen already exists, the callback is called immediately. Otherwise, it calls the callback on reception of a LOAD event on the screen.

Best Practice

Ensure that the screen is fully loaded before doing any activities especially when the **Start Screen** activity is used.

Use Case Example

In the following example of the UI automation application, you start a screen and then do the **Wait (Screen)** activity on it. This activity ensures that the automation wait enough time to let the screen start and fully load to start interacting with it.



Wait Close (Screen)

Purpose

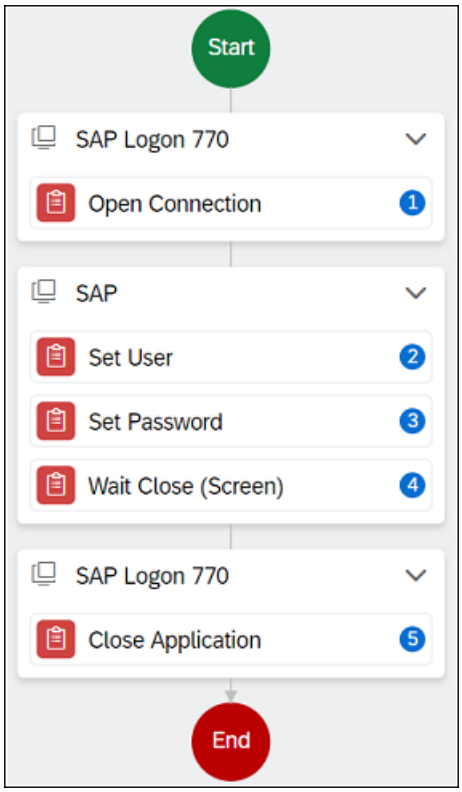
Use the **Wait Close (Screen)** activity until a screen is closed, and then call a callback. If the screen doesn't exist, the callback is called immediately. Otherwise, it calls the callback on reception of a UNLOAD event on the screen.

Best Practice

Use this activity if the automation needs to wait for a human to interact by closing the screen.

Use Case Example

In the following use case example of the simple automation for the connection on SAP Logon, you do the **Wait Close (Screen)** activity at the end to ensure that the automation wait for you to manually close the page and then proceed with the workflow.



Wait (Element)

Purpose

Use the **Wait (Element)** activity until a screen element is present and loaded, and then call a callback. If the screen already exists, the callback is called immediately. Otherwise, the polling is triggered to loop on element existence.

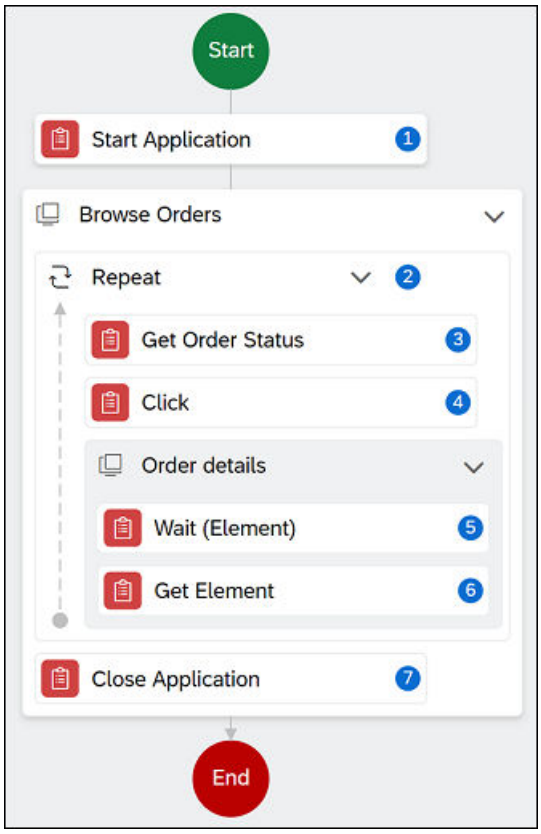
If you use the **Get Element** activity on an element in an application, the **Wait (Element)** activity will be automatically done by the SDK.

Best Practice

For UI Automation, the **Wait (Element)** activity must be used for certain UI elements by trying and troubleshooting. In some cases, the **Must Exist** activity can be added to ensure the existence of an element.

Use Case Example

In the following use case example of the SAPUI5 automation, you add the **Wait Element** activity because you open a tab inside the existing screen. This activity ensures that the tab and the element on the tab are present, and then you can do different activities on it, such as the **Get Element** activity and so on.



Wait Change (Element)

Purpose

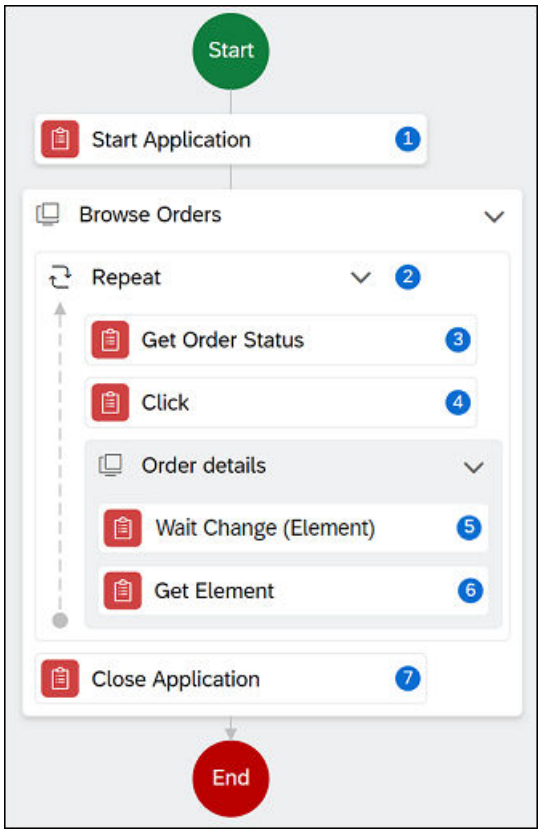
Use the **Wait Change (Element)** activity until the content of a given UI element is changed.

Best Practice

Use this activity for detecting UI changes and readiness rather than using the fixed wait activity.

Use Case Example

In the following use case example of the SAPUI5 automation, you add the **Wait Change (Element)** activity until initial text value of the element is changed, and then you can do different activities on it, such as the **Get Element** activity and so on.



Wait Ready

Purpose

Use the **Wait Ready** activity to keep the runtime execution in wait mode when the screen is in a busy-wait state.

> **i** Note
>
> This action is not integrated by default; therefore, it is mandatory for the user to add the **Wait Ready** activity manually between two intermediate UI5 activities.

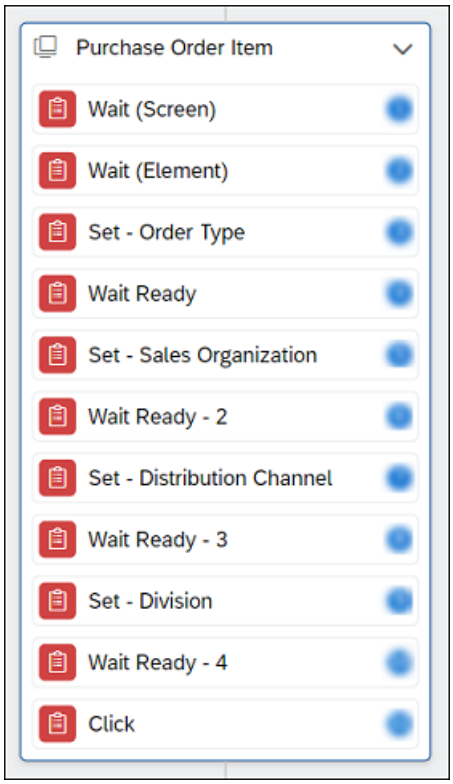
Best Practice

A screen can be loaded although some controls are still loading. To handle this behaviour, you must add the **Wait Ready** activity between two intermediate UI5 activities in your automation. This activity checks the state of each element of all the controls individually on a screen. The screen is not ready until all controls are ready.

The **Wait (Element)** activity can also be added after the **Wait (Screen)** activity so that the screen loads smoothly.

Use Case Example

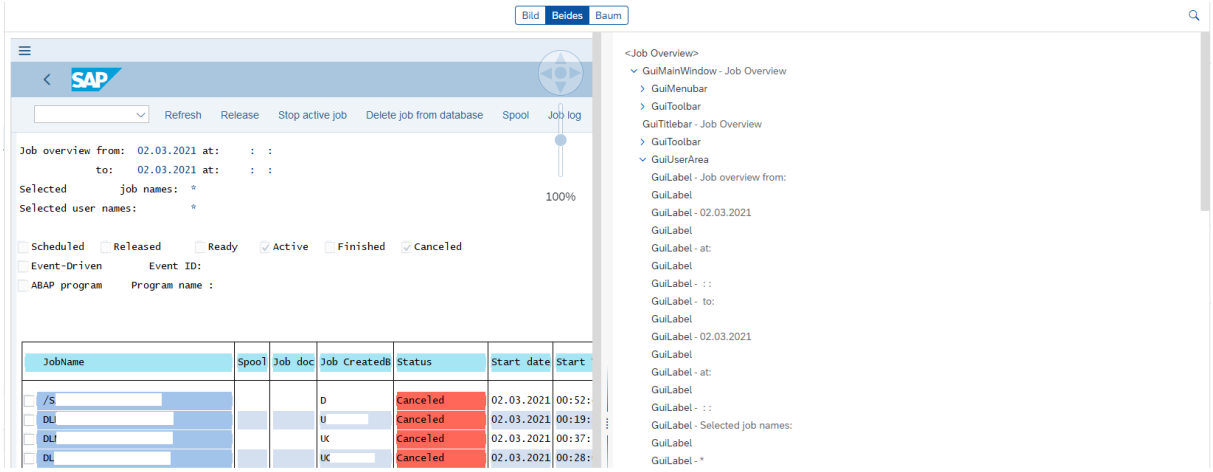
In the following example of the automation (purchase order item) with the SAPUI5 application, you add the **Wait (Screen)** activity to check if the screen is loaded. Then, you add the **Wait (Element)** activity to check if the element is present on the screen. After setting the value for the Order Type element, you add the **Wait Ready** activity to check if the next **Set – Sales Organization** action is performed. Similarly, you add another **Wait Ready** activity after the **Set – Sales Organization** activity to check if the next **Set – Distribution Channel** action performed. Thus, you must add the **Wait Ready** activity between two intermediate UI5 activities to check every UI5 element individually.



5.1.1.5 Best Practices for SAP GUI Activities

Automating SAP GUI Screens with Low Hierarchies

When you capture a table with low hierarchies in the Cloud Studio, a lot of elements are displayed in the tree view.



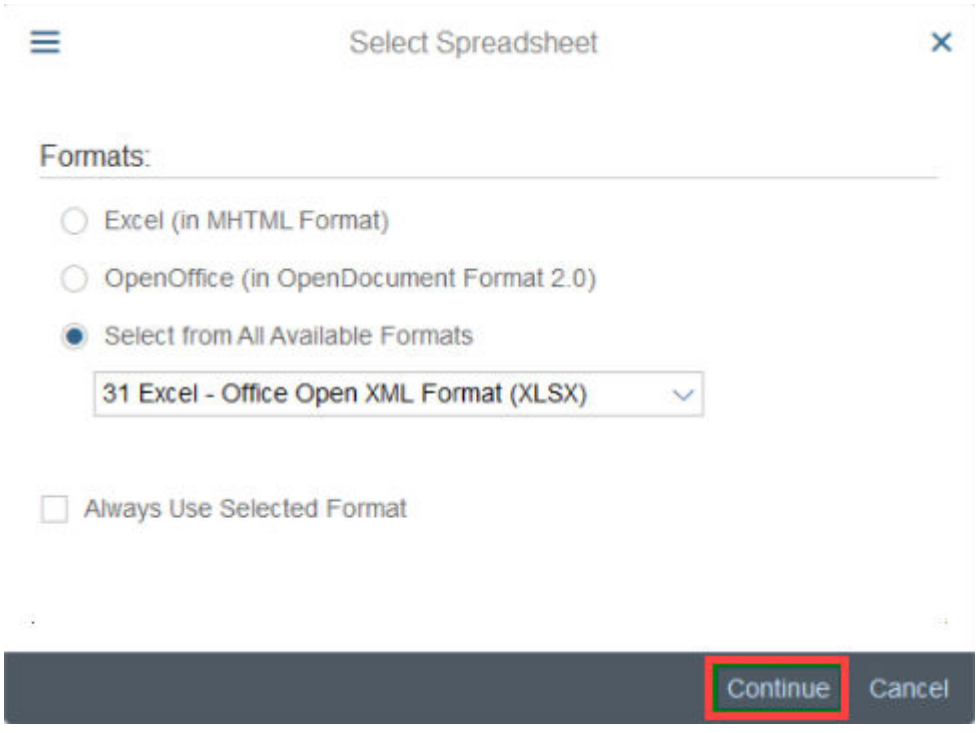
To recognize all the elements on the screen requires a lot of testing that can lead to a bad performance during the bot execution.

To handle this behavior, you can avoid reading the table in the Cloud Studio and instead read the table in Excel. To export the table in an Excel file, you have to press `SHIFT` + `4` and capture two pop-up windows.

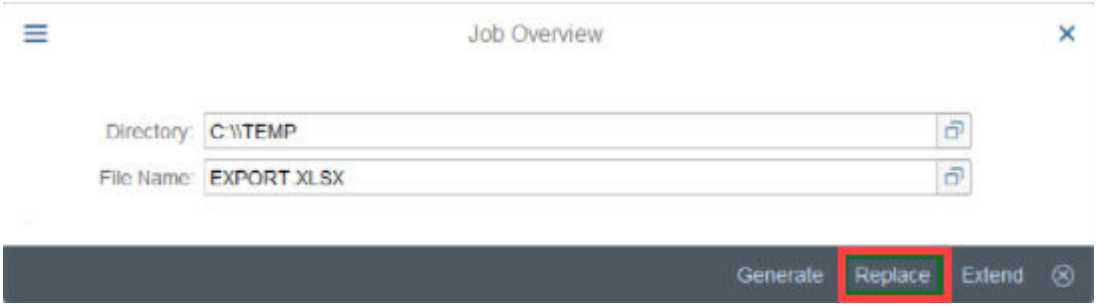
Once you have captured the two pop-up windows, you can automate the captured screens with the following procedure.

Procedure

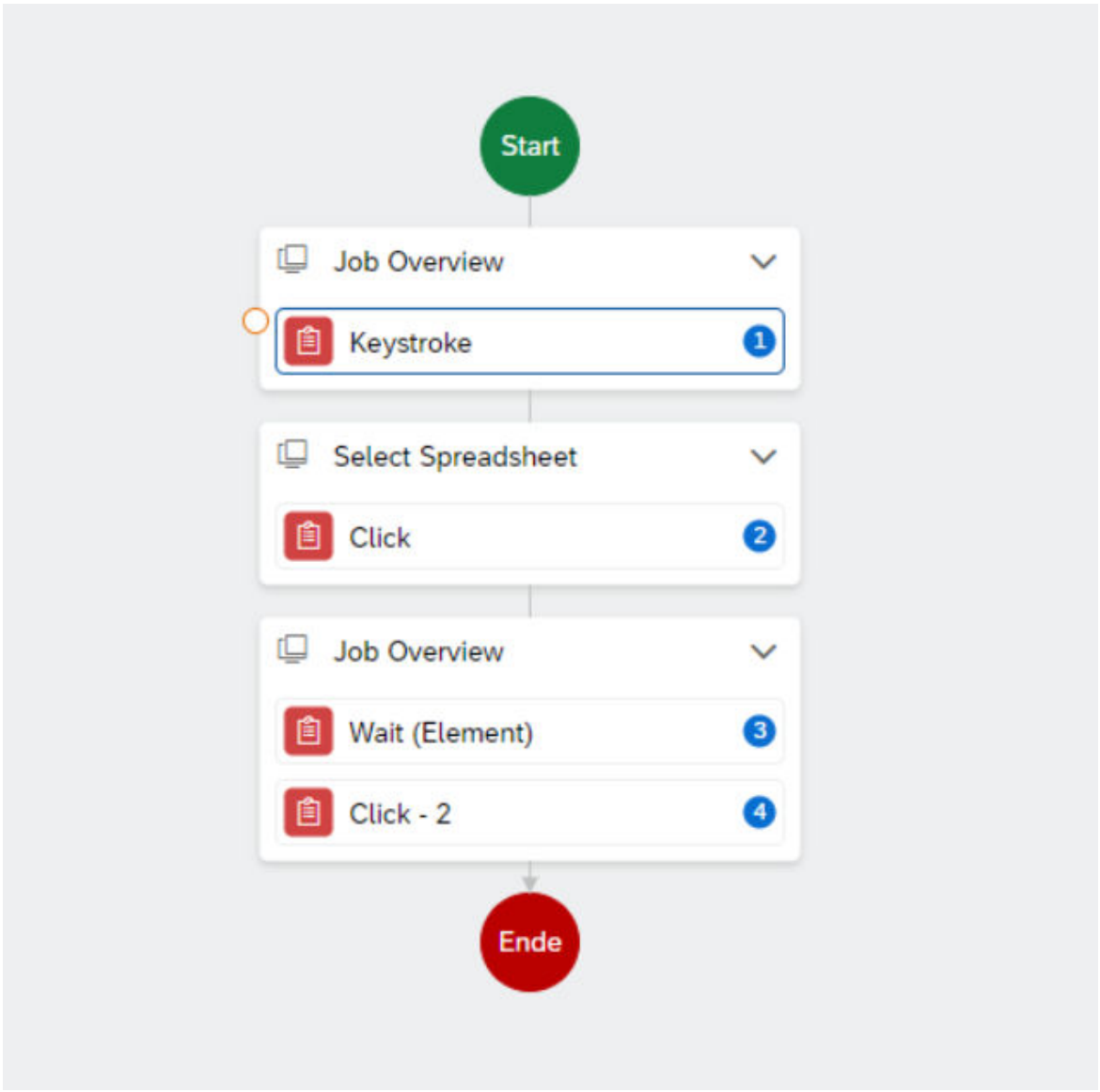
1. Execute the *Keystroke* activity on the captured screen with the table.
   The first pop-up window appears.



2. Click *Continue*.
   The second pop-up window appears.

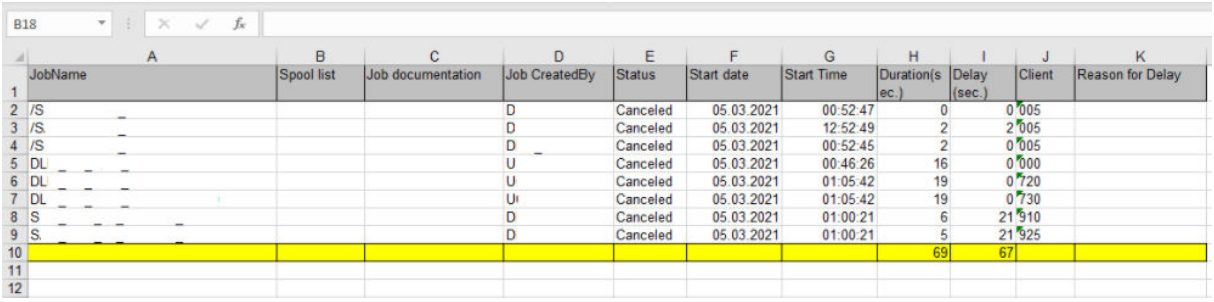


3. Click *Replace*.
   The following sreenshot shows an example of the workflow to automate SAP GUI screens with low hierarchies:



Result

The table is opened and saved in an Excel file.



	A	B	C	D	E	F	G	H	I	J	K
1	JobName	Spool list	Job documentation	Job CreatedBy	Status	Start date	Start Time	Duration(sec.)	Delay (sec.)	Client	Reason for Delay
2	/S			D	Canceled	05.03.2021	00:52:47	0	0	005	
3	/S			D	Canceled	05.03.2021	12:52:49	2	2	005	
4	/S			D	Canceled	05.03.2021	00:52:45	2	0	005	
5	DL			U	Canceled	05.03.2021	00:46:26	16	0	000	
6	DL			U	Canceled	05.03.2021	01:05:42	19	0	720	
7	DL			U	Canceled	05.03.2021	01:05:42	19	0	730	
8	S			D	Canceled	05.03.2021	01:00:21	6	21	910	
9	S			D	Canceled	05.03.2021	01:00:21	5	21	925	
10								69	67		
11											
12											

Hold Session and Release Session Activities

Purpose

In SAP Intelligent RPA, the scripting access is released after performing each SAP GUI action, and the scripting access is obtained again while performing the next action. Therefore, if you need uninterrupted scripting access for a set of actions, you must encapsulate those actions between Hold Session and Release Session activities.

About the Activities

Hold Session activity holds the SAP GUI session.

Release Session activity releases the SAP GUI session. This activity is used only when the Hold Session activity is added in the automation.

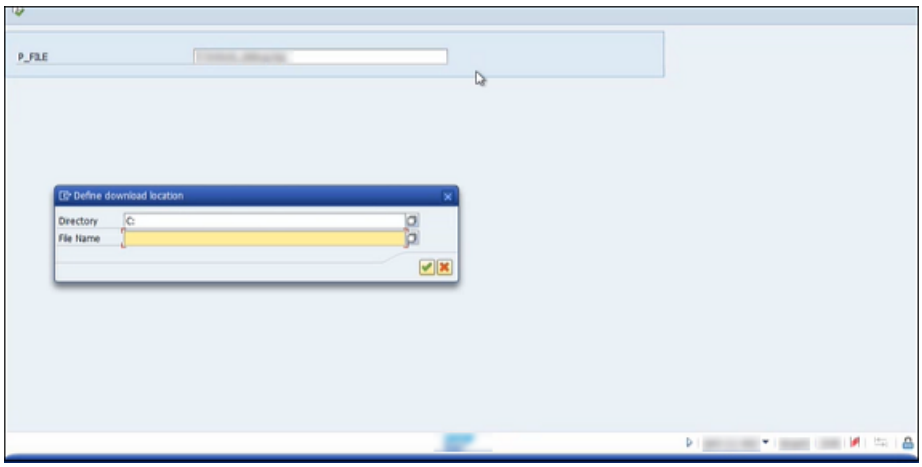
Best Practice

In automation, a set of actions must be encapsulated between Hold Session and Release Session activities so that the SAP scripting does not release until all those actions are completed.

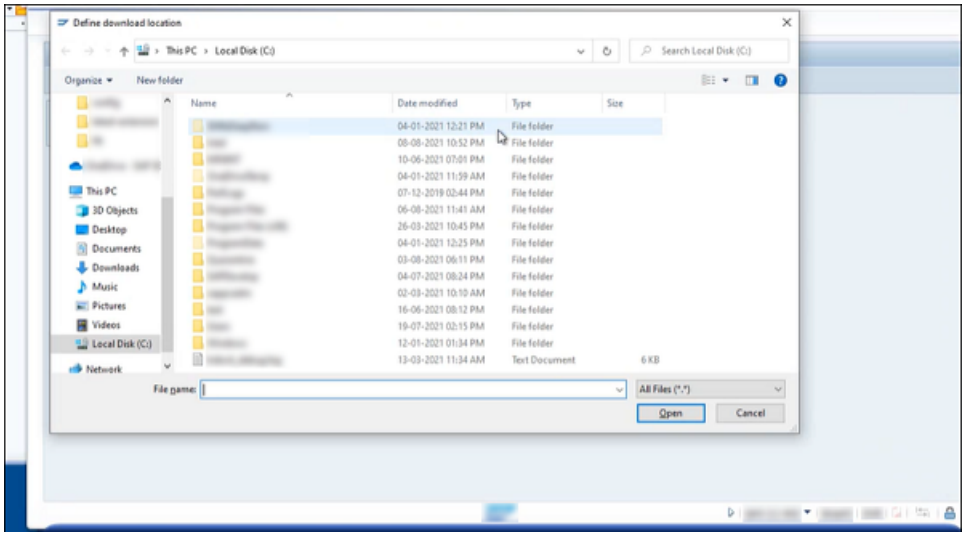
Use Case Example

In SAP GUI, the behavior of certain actions is dependent on the status of the SAP scripting.

If the scripting is active, the SAP GUI pop-up is displayed.

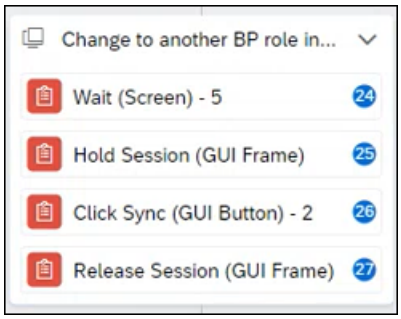


If the scripting is not active, the windows dialog is displayed.



Therefore, there is always an uncertainty displaying the expected SAP GUI pop-up window. To overcome this situation, you must drag and drop Hold Session and Release Session activities inside your automation and then encapsulate the set of actions between the two activities to make sure that the right action is performed when the SAP scripting is active and the SAP scripting is not released until the expected set of actions are completed.

In the following use case example of the automation, you use the click action between Hold Session and Release Session activities to always get the native SAP GUI pop-up window.

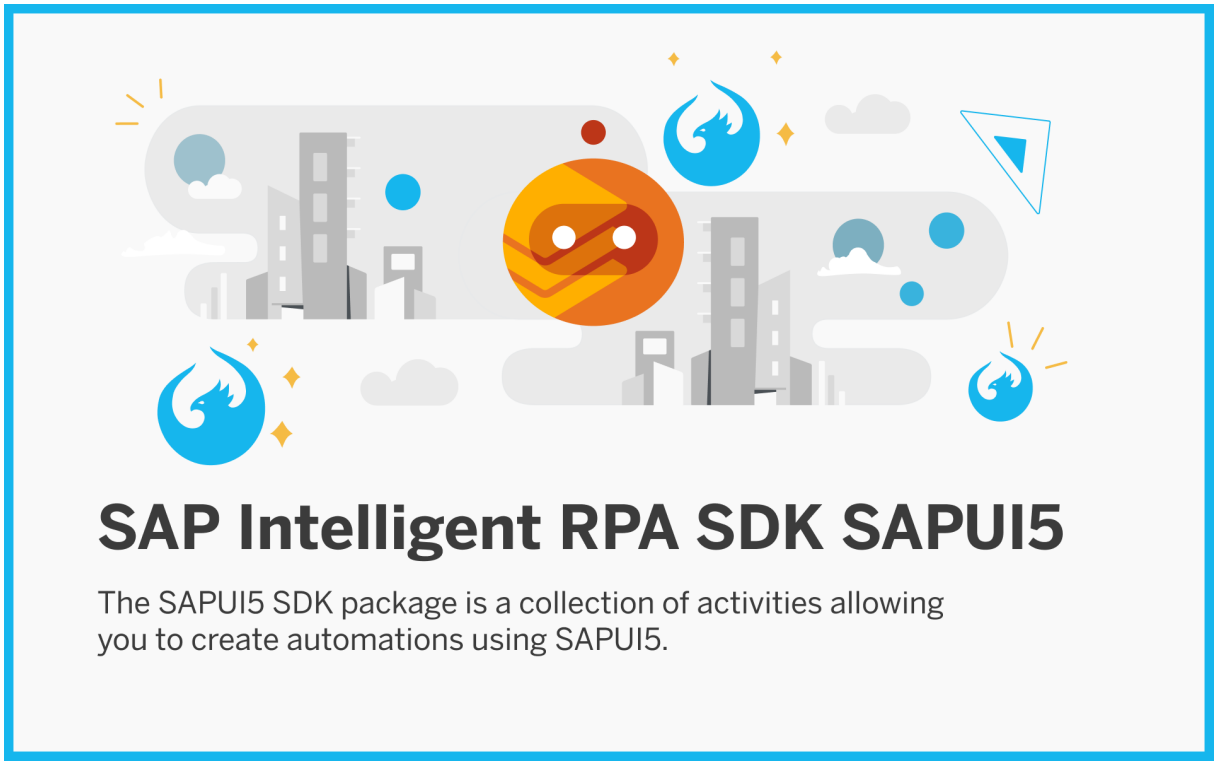


Launch SAP Logon in Unattended Mode

For more information, check the dedicated topic in the Troubleshooting Guide.

5.1.2 SDK SAPUI5 Package

The SAPUI5 SDK package is a collection of activities allowing you to create automations using SAPUI5.



- https://help.sap.com/doc/cd61cb973a064d58935e02a767e6da42/Cloud/en-US/SDKdocumentation.html [https://help.sap.com/doc/cd61cb973a064d58935e02a767e6da42/Cloud/en-US/SDKdocumentation.html]

5.1.2.1 Best Practices for SAPUI5 Activities

Loading Screens

Wait Until All Controls on a Screen are Ready

A feature of SAPUI5 is that a control can have a busy state that's different from that of the screen. A screen can be loaded although some controls are still loading.

To handle this behaviour, you must use the **Wait Ready (SAPUI5 Screen)** activity between two intermediate UI5 activities in your automation. This activity checks the state of each element of all the controls individually on a screen. The screen is not ready until all controls are ready.

For more information on the **Wait Ready** activity, see Best Practices for Wait Activities [page 386].

SAPUI5 Upload Collection

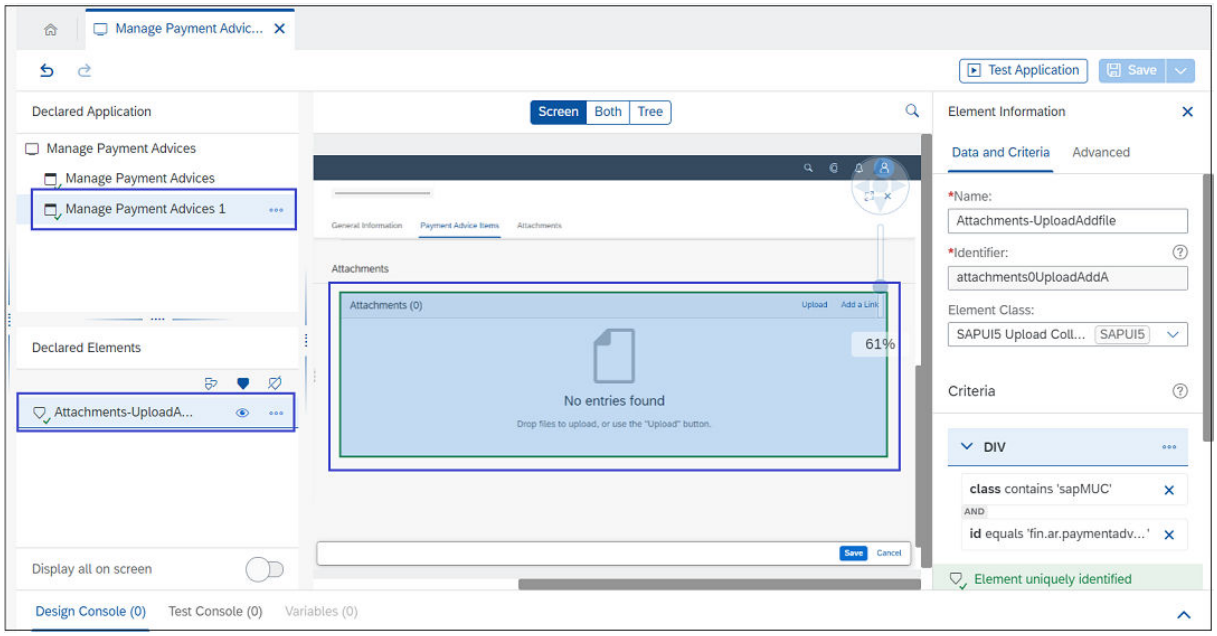
SAPUI5 Upload Collection is a control for attachment management that provides the ability to upload various types of files. You can upload single or multiple files using the Upload Collection control.

Upload File

The *Upload File* is an activity under SAPUI5 Upload Collection Subgroup in SAP UI5 SDK. This activity provides the ability to upload single or multiple files.

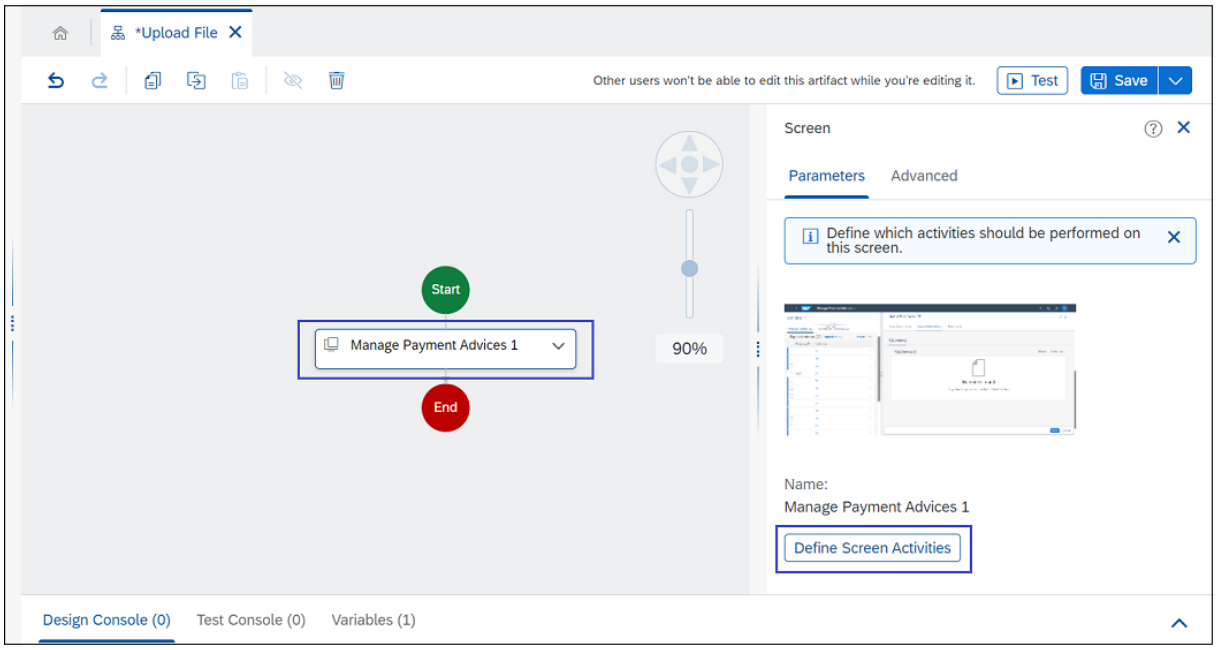
Uploading Single File

In the Cloud Studio, you must capture a screen and then define an appropriate element.

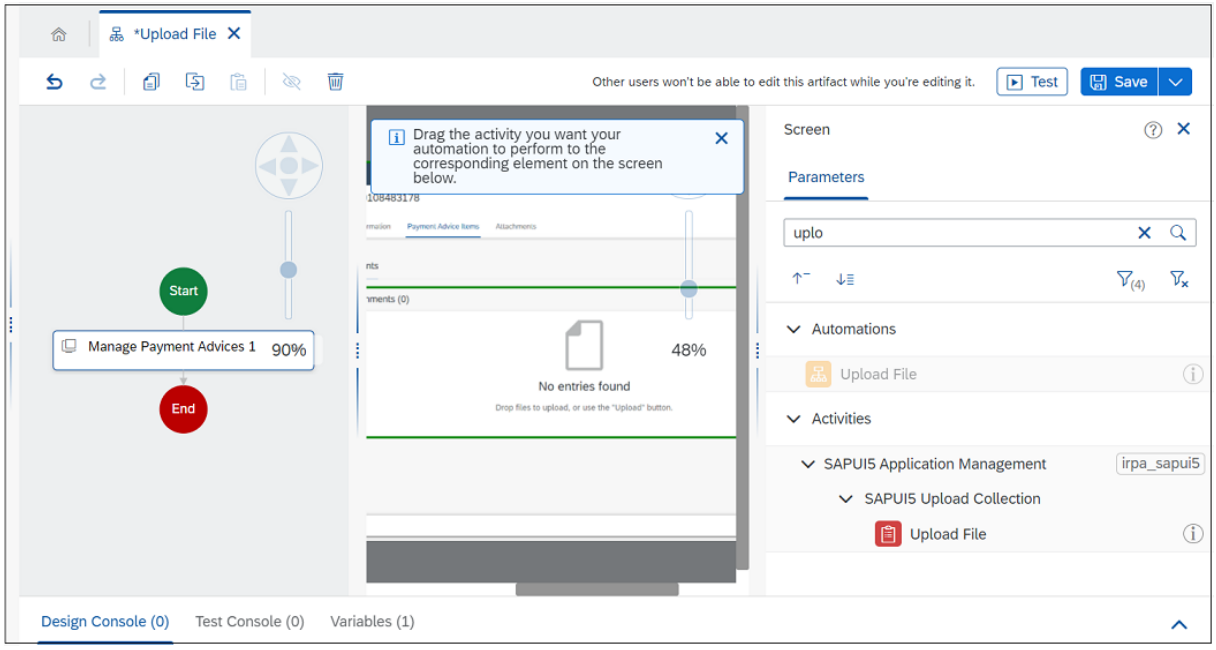


For more details about capturing an application screen and defining an element, refer to the Capture an Application [page 77] and Declare an Element [page 110] sections.

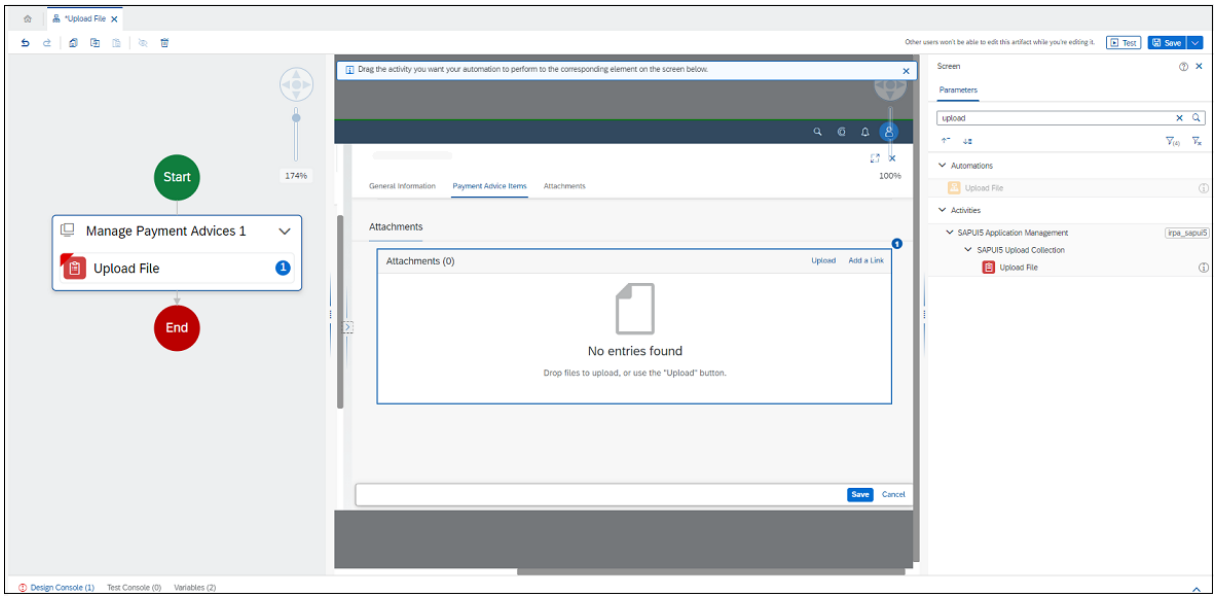
After defining an appropriate element, you must define and select the appropriate screen in the automation workflow. The *Define Screen Activities* option is displayed.



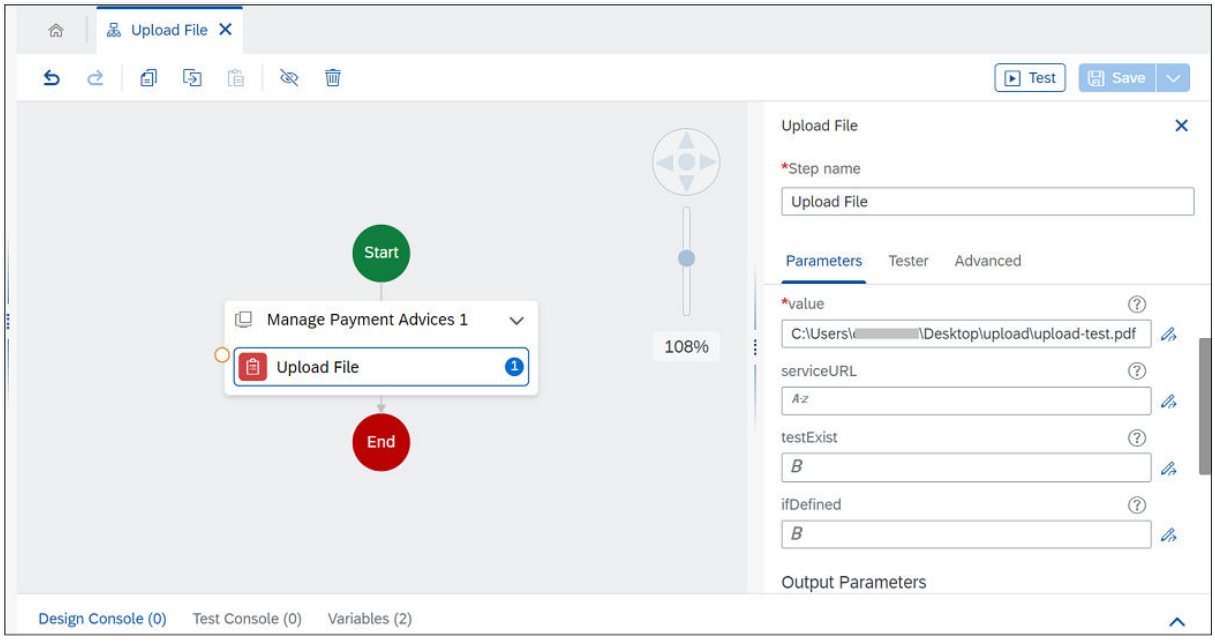
Click *Define Screen Activities* and then search for the *Upload File* activity in the *Search* bar.



Drag and drop the *Upload File* activity on a defined element in the screen. The activity is now added to the automation workflow.



Select the *Upload File* activity and then provide the file path including the file name and format in the *value* field. The file path is the path from where the file is fetched and uploaded.



> ## Note
>
> The *serviceURL*, *testExist*, and *ifDefined* fields are optional.

Upload File Status

If the file upload fails, then the following errors are displayed in the *Test Console* :

1. res="Error:FILE UPLOAD ERROR"
2. res="Error:XCSRF TOKEN FETCH ERROR"

If the file upload is successful, then return statement will be no string.

> **Note**
>
> After the file upload is successful, you must update the page view using the Reload Method call. This method is used to update the page view after the file upload is successful and files are visible on the UI. The Reload Method is available in the Page level.
>
> To call reload method, you must build the code first. Call Application. **page.reload()** in the generated code next to the **uploadFile** method call.

Uploading Multiple Files

You can upload multiple files using the loop control or the *Upload File* activity multiple times.

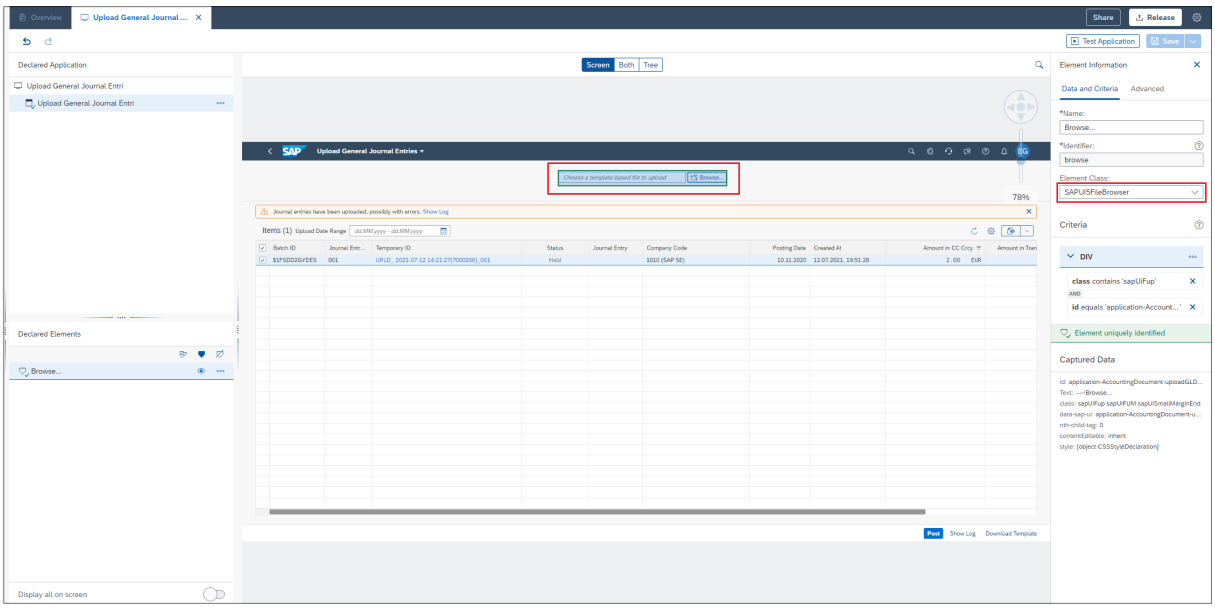
Constraints and Dependencies

1. There is a security restriction on the browser side to upload multiple files in one go.
2. Upload File control does not send the status of an individual file. For example, if you upload four files and one of them fails, then SAP UI5 Upload File method does not return FAIL status. As a result, uploading multiple files in a single method call does not give the correct file upload status (Success/Failure).

Browse File

The *Browse File* is an activity from the SAPUI5 File browser module of the SAPUI5 SDK. This activity provides the ability to browse a file from file system.

In the Cloud Studio, you must capture a screen and then define suitable elements.

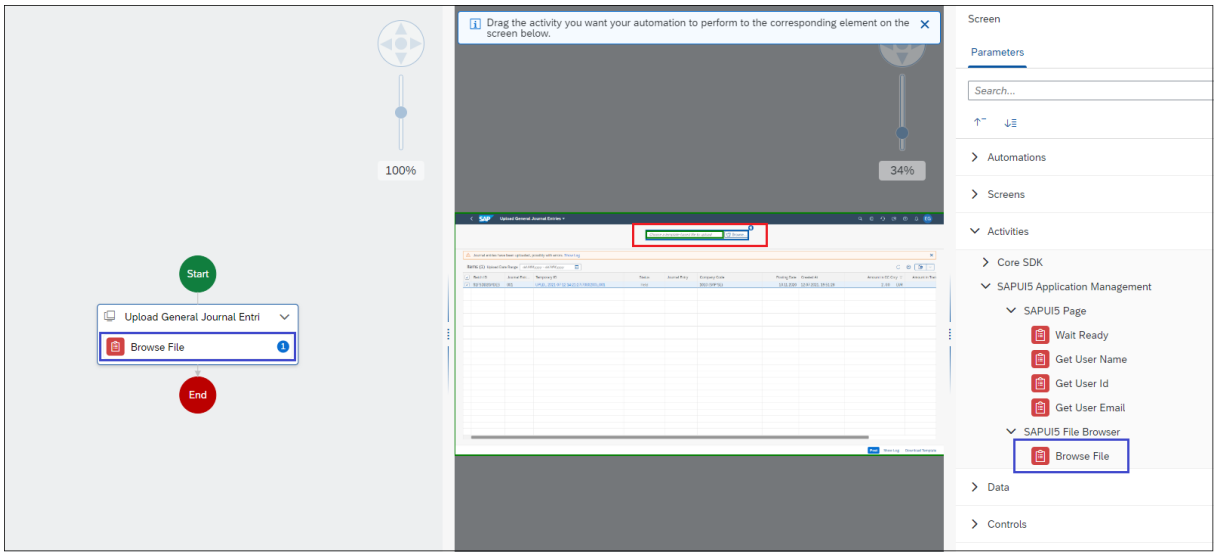


For more details about capturing an application screen and defining an element, refer to the Capture an Application [page 77] and Declare an Element [page 110] sections.

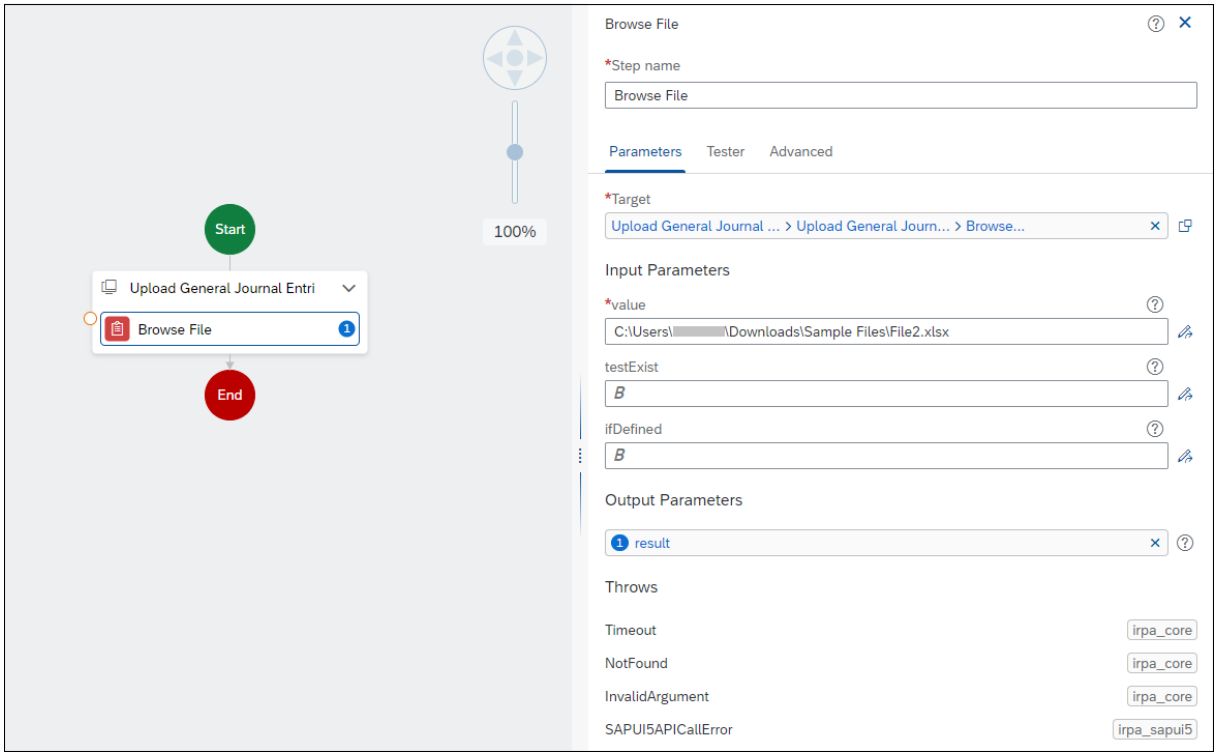
After defining a suitable, you must define and select the rightful screen in the automation workflow. The *Define Screen Activities* option is displayed.

Click *Define Screen Activities* and then search for the *Browse File* activity in the *Search* bar.

Drag and drop the *Browse File* activity on a defined element in the screen. The activity is now added to the automation workflow.



Select the *Browse File* activity and then provide the file path including the file name and format in the *value* field. The file path is the path from where the file is fetched and uploaded.



> ### Note
>
> The *testExist* and *ifDefined* fields are optional.

Upload File Status

If the file type does not match with accepted file type, then the following error is displayed in the **Test Console**:

- res="Error: File Type Missmatch"

If the control is not a file browser control, then the following error is displayed in the **Test Console**:

- res="Error: Not a FileBrowse Control"

If the file upload is successful, then return statement will be no string.

SAPUI5 Samples

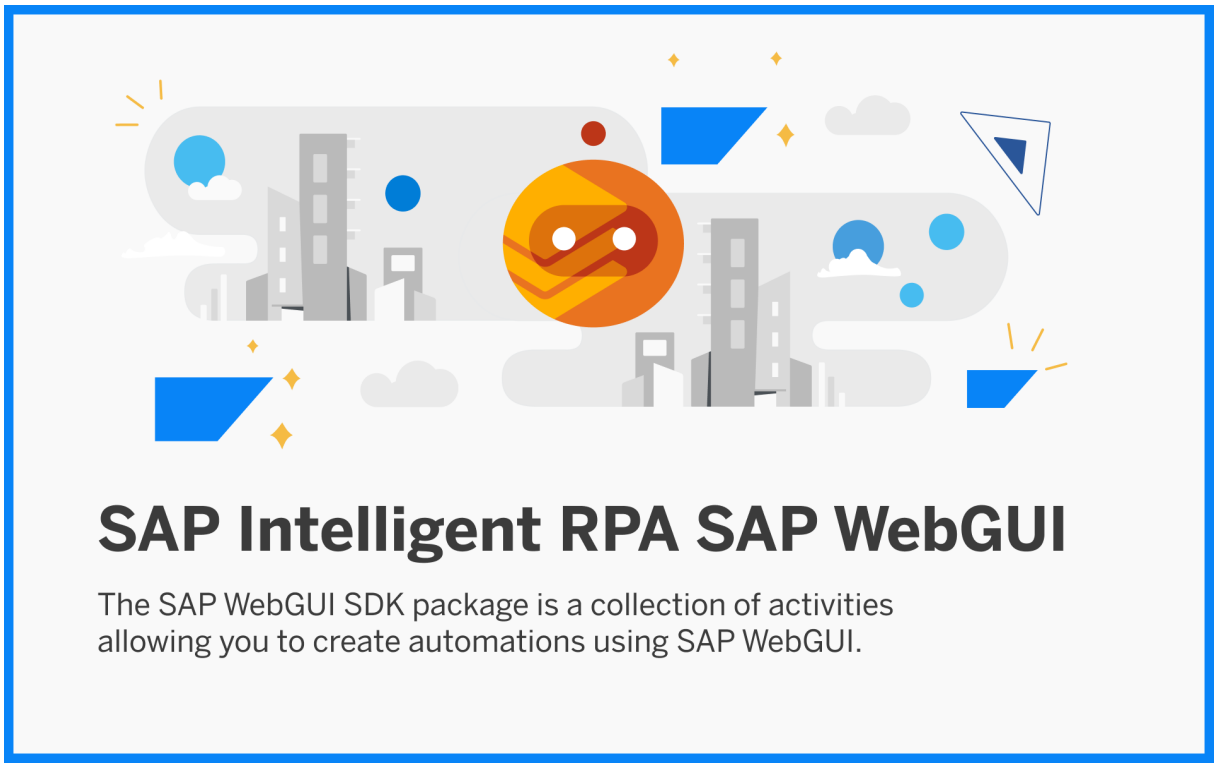
You can download SAPUI5 samples from the SAP Intelligent RPA Store to learn how to use different controls:

- To use the *Generic Tag* control, download the How to use Generic Tag control on UI5 page sample.
- To use the *Button* control, download the How to use Button control on UI5 page sample.
- To use the *Commands* control, download the How to use Commands control on UI5 page sample.
- To use the *Add Bookmark Button* control, download the How to use Add Bookmark Button control on UI5 page sample.
- To use the *Breadcrumbs* control, download the How to use Breadcrumbs control on UI5 page sample.
- To use the *ActionSheet* control, download the How to use ActionSheet control on UI5 page sample.

5.1.3 SDK SAP WebGUI Package

The SAP WebGUI SDK package is composed of activities for accessing and manipulating applications using SAP WebGUI.

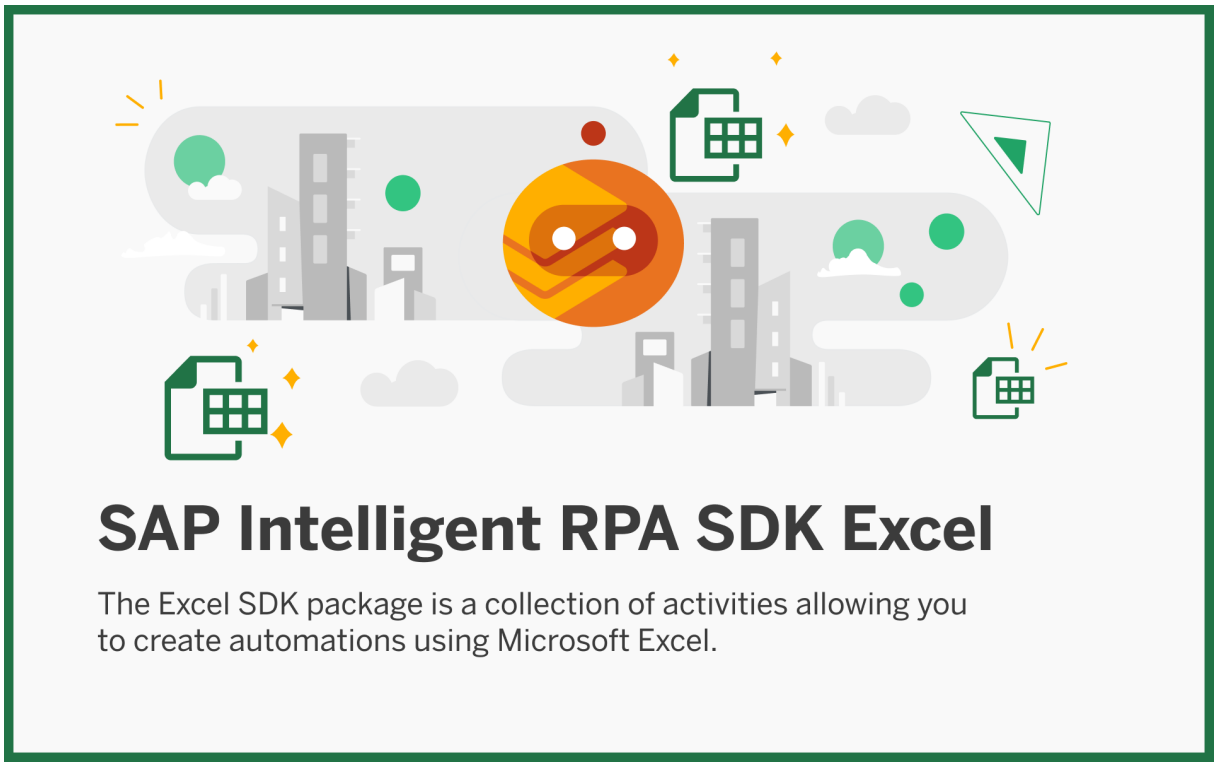
SAP WebGUI, or SAP GUI for HTML, dynamically emulates SAP transaction screens in a Web browser by automatically mapping SAP screen elements to HTML via the Internet Transaction Server (ITS). The SAP WebGUI activities are grouped in the SAP WebGUI Application Management module.



- https://help.sap.com/doc/9297ac6cb5be425383f61f7e76cbc7c4/Cloud/en-US/SDKdocumentation.html [https://help.sap.com/doc/9297ac6cb5be425383f61f7e76cbc7c4/Cloud/en-US/SDKdocumentation.html]

5.1.4 SDK Excel Package

The Excel SDK package is a collection of activities allowing you to create automations using Microsoft Excel.



- https://help.sap.com/doc/b8b5c9bbac3846a0a57aedab3f412880/Cloud/en-US/SDKdocumentation.html [https://help.sap.com/doc/b8b5c9bbac3846a0a57aedab3f412880/Cloud/en-US/SDKdocumentation.html]

Integrate Excel Activities

The Excel SDK package is composed of activities for accessing and manipulating Microsoft Excel files.

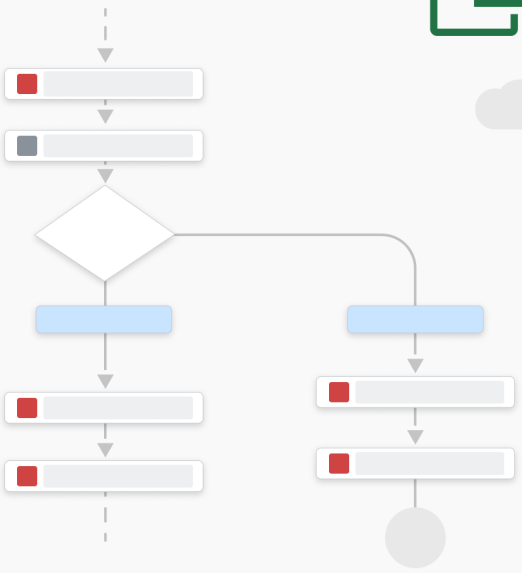


- Set Values
- Add Worksheet
- Get Row
- Push Values
- Excel Cloud Link

+ Many more...

Build Powerful Workflows

The library is structured in different groups of activities called modules that contains activities allowing you to manipulate Excel files.



5.1.4.1 Best Practices for MS Excel Activities

This section describes MS Excel activities that help you design automations to perform various Excel events in an Excel document.

To get started with the **SDK Excel Package**, you can download the Excel Best Practices sample on the SAP Intelligent RPA Store.

Get Values (Cell)

Use the Get Values (Cell) activity to return (read) the values of a specified cell range in the ACTIVE worksheet.

Using the *Get Values (Cell)* activity, an out of memory error can potentially happen at runtime. Different causes can lead to an out of memory error: hardware lack of performance, lots of applications opened at once, important volume of cells...

The *Get Values (Cell)* activity has been tested with one million cells. If you get an out of memory error, the recommended approach is to:

- use the pagination to read you data by "block" of cells
- consider using the 64 bit version of Microsoft Excel instead of the 32 bit version

Select Range

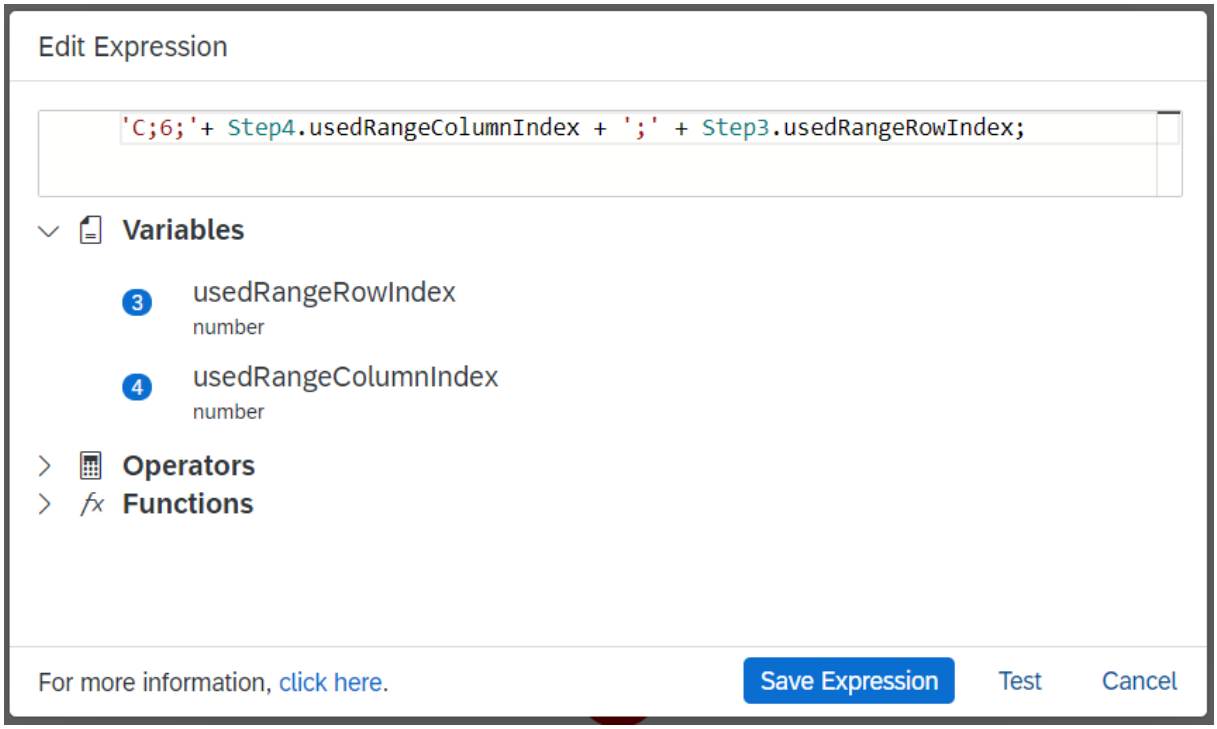
You must use three different activities in your automation to select a range in the active worksheet.

Use the Get UsedRange Row (Cell) activity to return the row index. A variable defined as an output parameter is automatically created.

Use the *Get UsedRange Column* activity to return the column index. A variable defined as an output parameter is automatically created.

Then, use the Select Range activity to select the range of your data in the active worksheet.

To select the range, open the *Expression Editor* next to the *rangeDefinition* field in the *Input Parameters* section. The expression must contain the first cell of the range you want to select and the output parameters of the *Get UsedRange Row* and *Get UsedRange Column* activities.



setFullRangeValues

You can use *setFullRangeValues* to set (write) the values of a range of cells from a given workbook or worksheet.

Sometimes, while using this activity to set (write) the values of a **full range** of cells, an out of memory error can potentially happen at runtime.

To address the out of memory issue, we recommend the following approaches:

- The performance of an algorithm on Excel is totally depending on the resources and the status of the memory of your machine. It is very complex to give a ratio from where the number of cells will not be able to be written. So, we recommend you not to write too many cells with the *setFullRangeValues* activity. Once again, it depends on the resources and memory of your machine, but 10,000 cells should be an average.
- Consider using the 64 bit version of Microsoft Excel instead of the 32 bit version

Get Row From Data

The Get Row From Data activity returns the index of the last row with data. That means that if in the column (either the column of the active cell or the column of the referenceCell) there is a cell without any data it will return the index of the row before this cell.

For example,

If you launch the activity with the A1 cell as the reference cell the output will be 19.

	A	B	C	D	E	F	G	H	I	J	K
1	982	795	316	167	28	589	76	34			
2	743	456	402	437	280	726	297	819			
3	737	426	486	66	574	812	582	476			
4	442	357			65	889	944	557			
5	186	481	151	326	60	237	590	726			
6	636	521	91	897	17	802	79	310			
7	843	64	532	600				152			
8	679	773	570	851				186			
9	304			693	933	342	373	394			
10	17	365	686	858	232	34	769	976			
11	561	565	604	864	344	128	191	148			
12	955	695	25	29	990	403	749	415			
13	387			551	383	7	346	313			
14	944		671	59	937	209		444			
15	135	55	281	392	99		607	537			
16	17	527	393	462	498	39	343	14			
17	727	250	278	568	823	444	764	115			
18	721	428	528	755	601	367	872	926			
19	979	99	243	382	385	613	571	569			
20											
21											
22											
23											

But if you put D1 as the reference cell, the output will be 3. Why 3? Because the D4 cell is empty so the count stops there. This is an excel behavior that can be simulated by pressing CTRL + DOWN ARROW (or UP ARROW) directly in Excel.

Get Column From Data

The Get Column From Data activity returns the numerical index of the last column with data. That means that if in the row (either the row of the active cell or the row of the reference cell) there is a cell without any data it will return the index of the column before this cell.

For example,

If you launch the activity with the A1 cell as the reference cell the output will be 8 (the index of the H column is 8).

	A	B	C	D	E	F	G	H	I	J	K
1	982	795	316	167	28	589	76	34			
2	743	456	402	437	280	726	297	819			
3	737	426	486	66	574	812	582	476			
4	442	357			65	889	944	557			
5	186	481	151	326	60	237	590	726			
6	636	521	91	897	17	802	79	310			
7	843	64	532	600				152			
8	679	773	570	851				186			
9	304			693	933	342	373	394			
10	17	365	686	858	232	34	769	976			
11	561	565	604	864	344	128	191	148			
12	955	695	25	29	990	403	749	415			
13	387			551	383	7	346	313			
14	944		671	59	937	209		444			
15	135	55	281	392	99		607	537			
16	17	527	393	462	498	39	343	14			
17	727	250	278	568	823	444	764	115			
18	721	428	528	755	601	367	872	926			
19	979	99	243	382	385	613	571	569			
20											
21											
22											
23											

But if you put A4 as the reference cell, the output will be 2. Because it the same as the row, the index is the column of the last non empty cell on a row. So here the C4 cell is empty so the column is B which index is 2. This is an excel behavior that can be simulated by pressing CTRL + RIGHT ARROW (or LEFT ARROW) directly in Excel.

> **i** Note
>
> For both activities, if you reference a blank cell, the output will be the index of the first non blank cell of the row or the column referenced.

Get Used Range Row

The Get Used Range Row activity returns the index of the last row used. It means that even if you delete the data of the last row used, the activity will still return the index of this one.

For example,

If you test the activity with the excel as it is (without putting data further than the H19 cell) the result would be 19. Now if you put some data further so for example in the J21 cell, you will have 21 as the result which is expected.

	A	B	C	D	E	F	G	H	I	J
1	982	795	316	167	28	589	76	34		
2	743	456	402	437	280	726	297	819		
3	737	426	486	66	574	812	582	476		
4	442	357			65	889	944	557		
5	186	481	151	326	60	237	590	726		
6	636	521	91	897	17	802	79	310		
7	843	64	532	600				152		
8	679	773	570	851				186		
9	304			693	933	342	373	394		
10	17	365	686	858	232	34	769	976		
11	561	565	604	864	344	128	191	148		
12	955	695	25	29	990	403	749	415		
13	387			551	383	7	346	313		
14	944		671	59	937	209		444		
15	135	55	281	392	99		607	537		
16	17	527	393	462	498	39	343	14		
17	727	250	278	568	823	444	764	115		
18	721	428	528	755	601	367	872	926		
19	979	99	243	382	385	613	571	569		
20										
21										hello
22										

Now you can test if you delete the data of the J21 cell, you will still have 21 as the result. You can test this behavior by pressing CTRL + END on excel, it shows the last cell used.

	A	B	C	D	E	F	G	H	I	J	K
1	982	795	316	167	28	589	76	34			
2	743	456	402	437	280	726	297	819			
3	737	426	486	66	574	812	582	476			
4	442	357			65	889	944	557			
5	186	481	151	326	60	237	590	726			
6	636	521	91	897	17	802	79	310			
7	843	64	532	600				152			
8	679	773	570	851				186			
9	304			693	933	342	373	394			
10	17	365	686	858	232	34	769	976			
11	561	565	604	864	344	128	191	148			
12	955	695	25	29	990	403	749	415			
13	387			551	383	7	346	313			
14	944		671	59	937	209		444			
15	135	55	281	392	99		607	537			
16	17	527	393	462	498	39	343	14			
17	727	250	278	568	823	444	764	115			
18	721	428	528	755	601	367	872	926			
19	979	99	243	382	385	613	571	569			
20											
21											
22											
23											

You can see that, if you delete the data from the J21 cell than do a CTRL + END, Excel will focus the last cell which is still J21.

Get Used Range Column

The Get Used Range Column activity returns the index of the last column used. It means that even if you delete the data of the last column used, the activity will still return the index of this one.

For example,

If you test the activity with the excel as it is (without putting data further than the H19 cell) you will have 8 as the result (H has for index 8). Same as for the row, when you put "hello" on the J21 cell, the activity will output 10, and if you delete the data it will still output 10.

	A	B	C	D	E	F	G	H	I	J
1	982	795	316	167	28	589	76	34		
2	743	456	402	437	280	726	297	819		
3	737	426	486	66	574	812	582	476		
4	442	357			65	889	944	557		
5	186	481	151	326	60	237	590	726		
6	636	521	91	897	17	802	79	310		
7	843	64	532	600				152		
8	679	773	570	851				186		
9	304			693	933	342	373	394		
10	17	365	686	858	232	34	769	976		
11	561	565	604	864	344	128	191	148		
12	955	695	25	29	990	403	749	415		
13	387			551	383	7	346	313		
14	944		671	59	937	209		444		
15	135	55	281	392	99		607	537		
16	17	527	393	462	498	39	343	14		
17	727	250	278	568	823	444	764	115		
18	721	428	528	755	601	367	872	926		
19	979	99	243	382	385	613	571	569		
20										
21										hello
22										

As you may have noticed the column indexes can be letters as well as numbers. The activities seen above, on the columns, outputs numbers. So, it can be useful to have activities that can convert indexes to letters and vice versa.

Convert Column Index to Name

This Convert Column Index to Name activity converts the column index number to column name. For the index 2 it will output B, for 42 it will output AP, etc.

Convert Column Name to Number

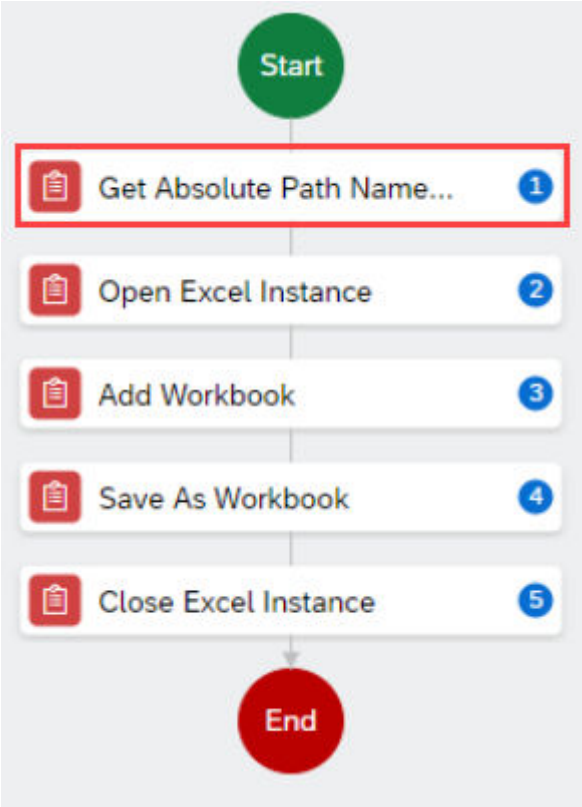
This Convert Column Name to Number activity converts the column name to index number. For the name B it will output 2, for AC it will output 29.

Deal With File Path

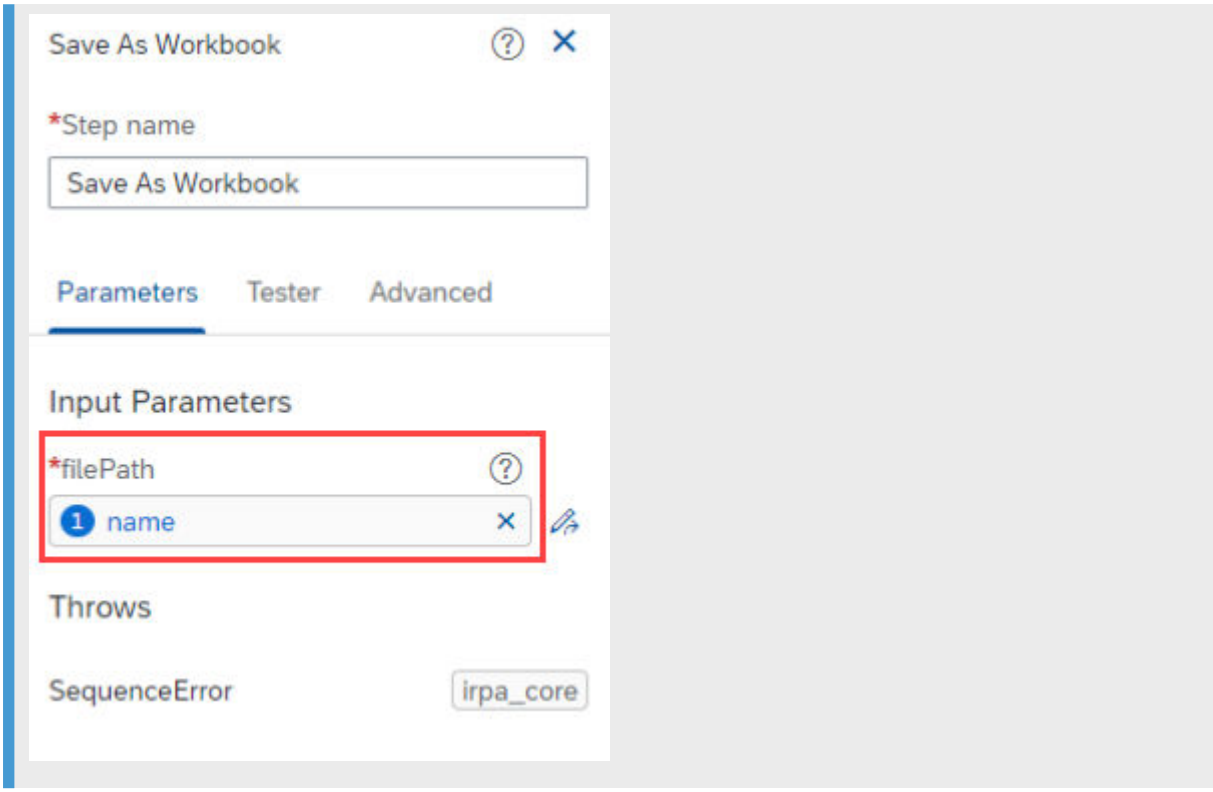
When you are working with MS Excel, you cannot use a file path that contains a slash (**/**). Indeed, if you want to open or save a workbook, an exception will be thrown.

To avoid this exception, use a file path that contains only single backslashes (****). For example, if you use a file path similar to `C:\Temp\book1.xlsx`, Excel does not throw an exception.

Use the *Get Absolute Path Name (File)* activity in your automation to get the correct file path to your Excel document.



> **i** Note
>
> You must select *name* as the *filePath* parameter of the *Save As Workbook* activity on the *Input Parameters* panel.



5.1.4.2 Excel Data Mapping

> **i** Note
>
> To get started with the **SDK Excel Package**, you can download the Excel Best Practices sample on the SAP Intelligent RPA Store.

What is Excel Data Mapping?

Excel Data Mapping allows you to transform columns-based data from an Excel sheet into data that can be used in your automation. The data from the Excel sheet stays the same but the structure becomes a data type structure, making it possible to use throughout your project in the Cloud Studio.

Data Type Mapping Activity

Excel Data Mapping is done with the *Excel Cloud Link* activity. You can find it on the tools side panel, in the activities section.

This activity does not open and close an Excel instance. To use it in the workflow of your automation, you need to drag and drop an activity that opens an Excel instance before, and drag and drop an activity that closes the Excel instance after.

The *Excel Cloud Link* activity includes the opening and closing of the workbook that is provided by the user.

This activity calls and returns the following input and output parameters:

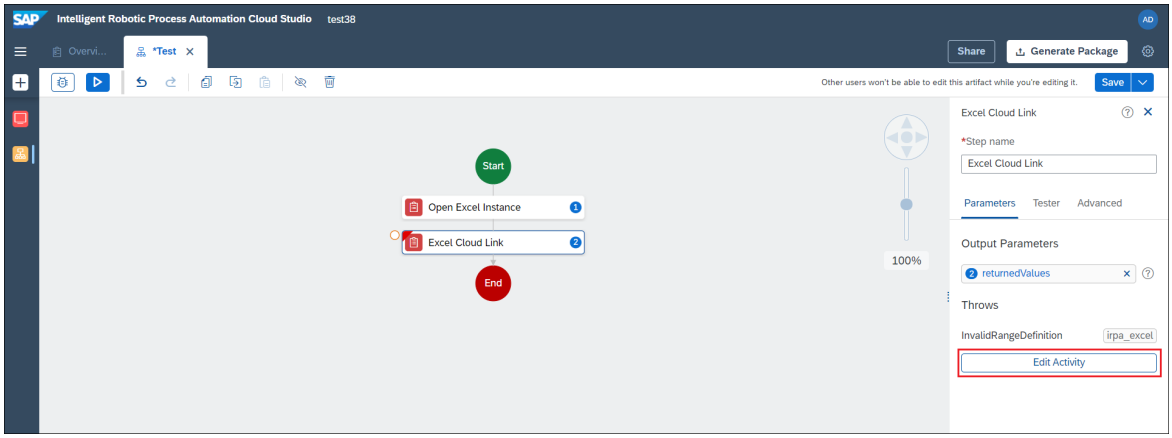
- Input parameters
  - mapping: json
    The mapping is the model used by the activity and the Cloud Studio to manage the data structure transformation.
  - *workbookPath* of type String
    To know what Excel file is used.
  - *worksheetName* of type String
    To use the correct Excel sheet.
  - *rangeDefinition* of type String
    To locate the data we want to transform.
  - *containsHeaders* of type Boolean
    To know if the data in the Excel sheet has column title headers.
- Output parameter
  - *returnedValues* of type Any

Excel Data Mapping Procedure

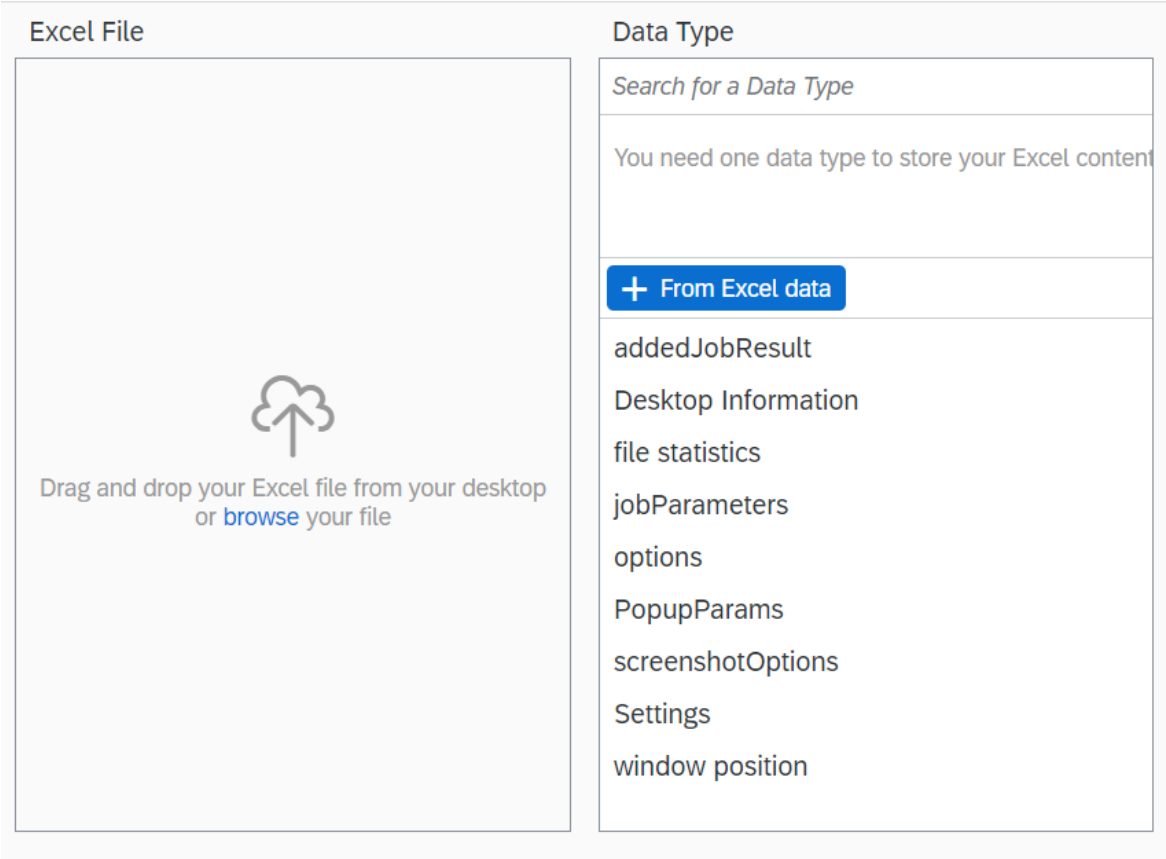
1. In your automation, on the right-hand side panel, go to the *Activities* section.
2. Drag and drop the activity *Open Excel instance*.

   > **i Note**
   >
   > To use Excel data mapping in the workflow of your automation, you first need to drag and drop any activity that opens an excel instance.

3. Drag and drop the activity *Excel Cloud Link*.
4. Select *Excel Cloud Link* and click on *Edit Activity*, or double-click on *Excel Cloud Link*.



The mapping interface appears with an Excel File and Data Type side panel.

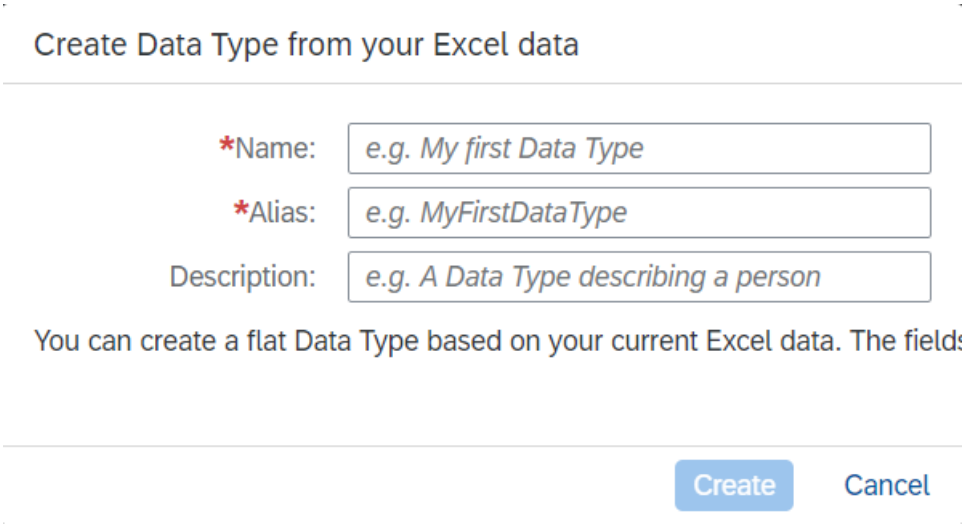


5. On the Data Type side panel, choose a data type to store the Excel data and structure the mapping.
   All the data types present in your project are available for the mapping:
   - the data types you created in the Cloud Studio
   - the data types imported with the core SDK packages

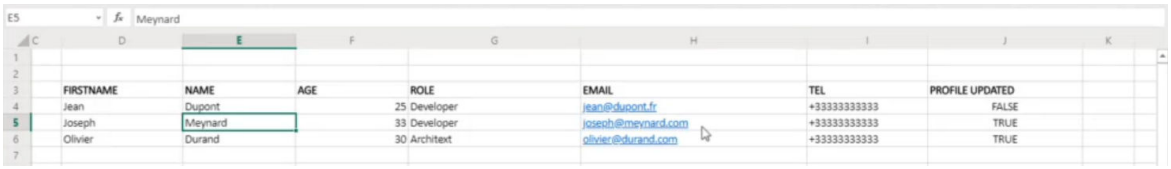
   You can also create a data type directly from this side panel.

> **i Note**
>
> To create a data type from the Excel data, you first need to upload the Excel file.

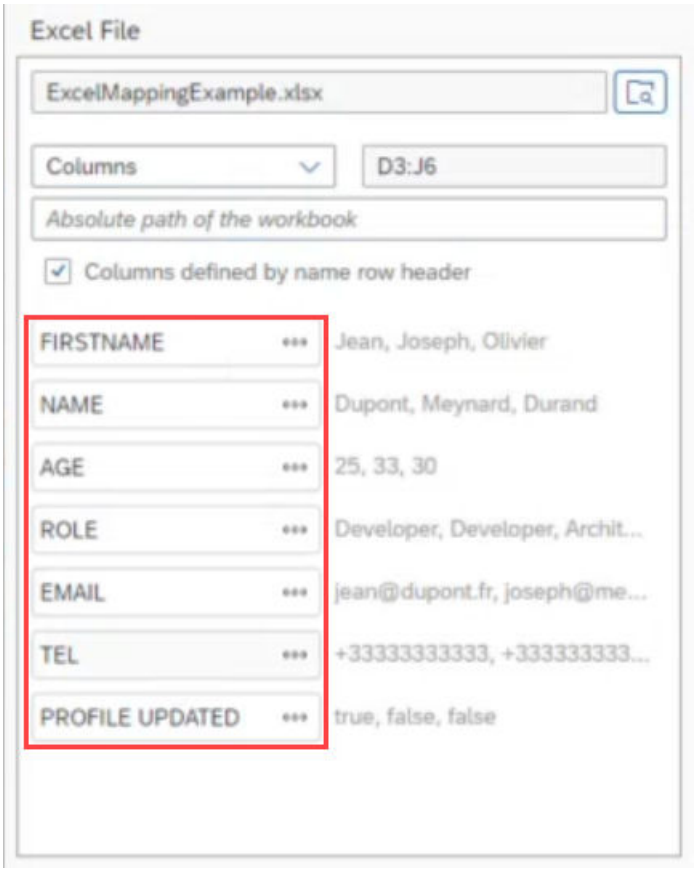
1. Click *From Excel Data*.
   A *Create Data Type from your Excel data* window pops up.



2. Enter a name.
3. Enter an alias.
4. **Optional**: Enter a description.
5. Click *Create*.
   You can now choose the data type you created to structure your mapping.
6. On the Excel File side panel, drag and drop your Excel file or click on *browse* to upload it. Once you upload the Excel file, the data contained in the file is listed on the side panel.

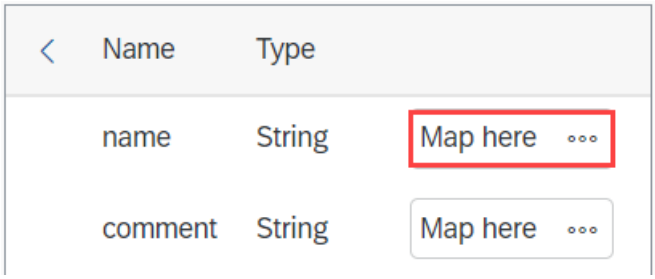


	FIRSTNAME	NAME	AGE	ROLE	EMAIL	TEL	PROFILE UPDATED
4	Jean	Dupont	25	Developer	jean@dupont.fr	+33333333333	FALSE
5	Joseph	Meynard	33	Developer	joseph@meynard.com	+33333333333	TRUE
6	Olivier	Durand	30	Architext	olivier@durand.com	+33333333333	TRUE

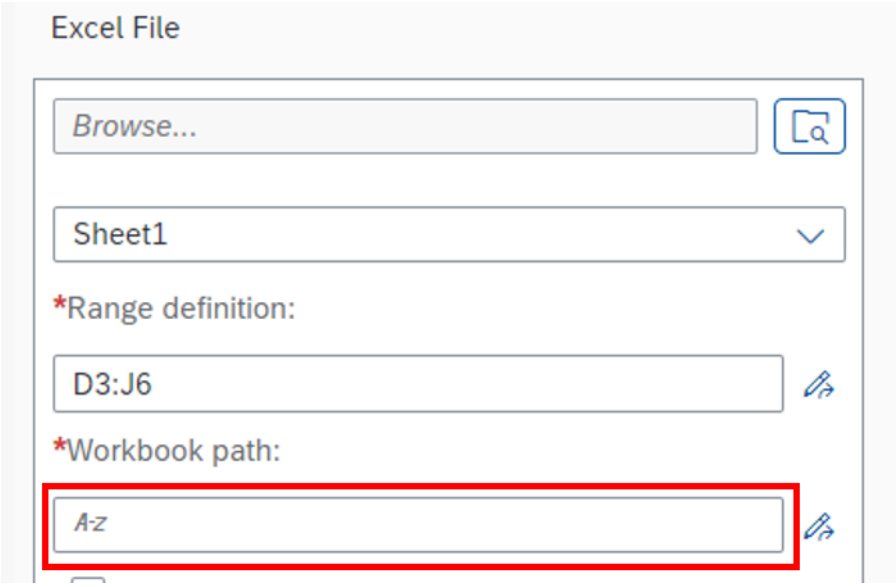


The elements listed represent tokens that you can map with the different data types.

7. Select a data type from the Data type side panel.
   Next to each data type is a *Map here* button.



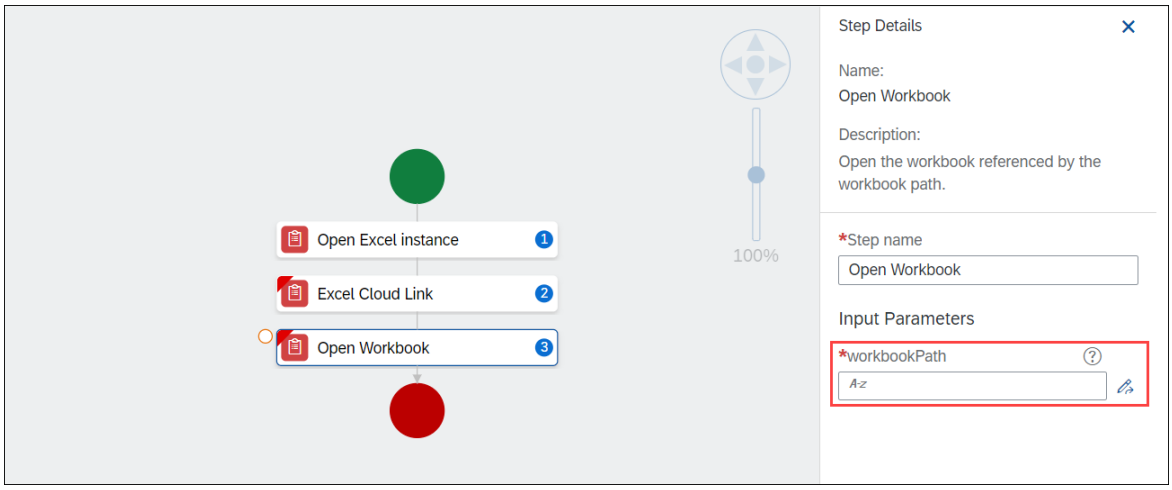
8. Drag and drop a token to the *Map here* button next to a data type.
9. Enter the path of your Excel file in the *Workbook path* field.



The range D3:J6 that you see at the top represents the range containing the data in the Excel sheet.

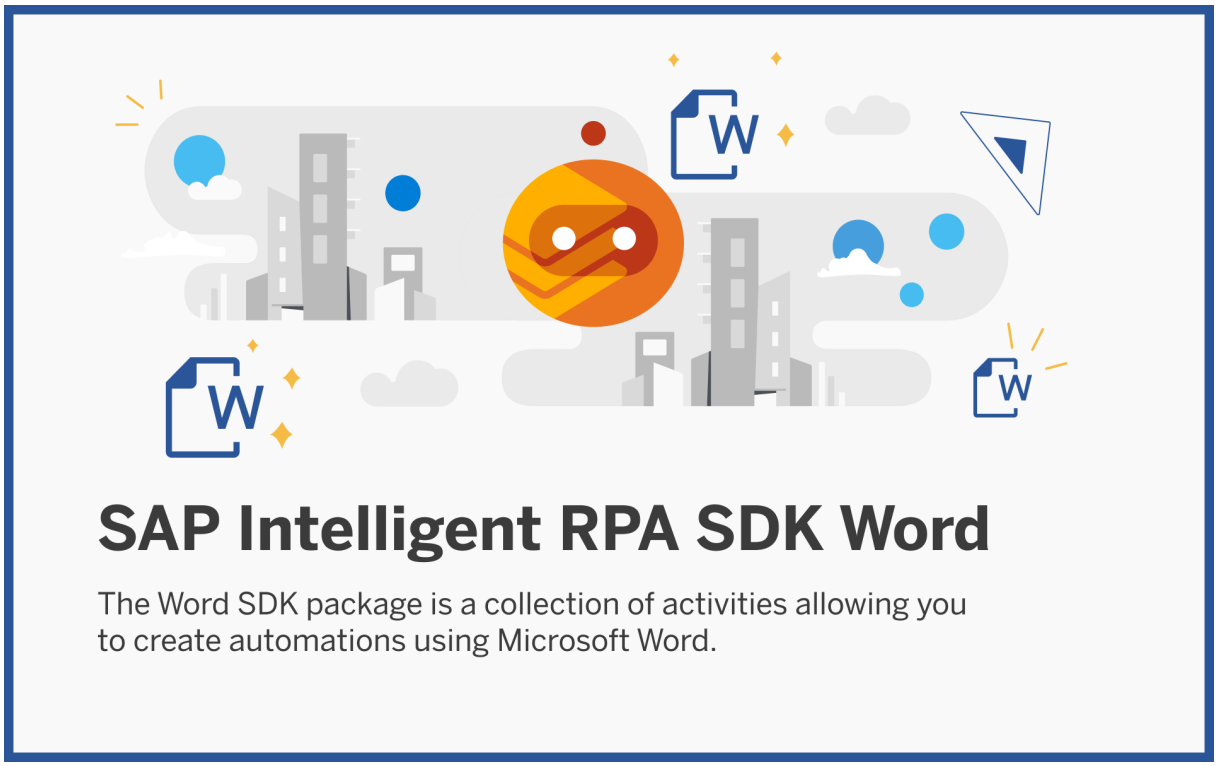
10. Save your work once you are done with the mapping.

- **Optional**: To add modifications or additional information to the workbook included in the *Excel Cloud Link* activity, drag and drop the *Open Workbook* activity in the workflow of your automation.
- The input parameter for this activity is the path of your Excel file.



5.1.5 SDK Word Package

The Word SDK package is a collection of activities allowing you to create automations using Microsoft Word.



- https://help.sap.com/doc/b66efdf2aaad445da3120d548beb81b2/Cloud/en-US/SDKdocumentation.html [https://help.sap.com/doc/b66efdf2aaad445da3120d548beb81b2/Cloud/en-US/SDKdocumentation.html]

5.1.5.1 Best Practices for MS Word Activities

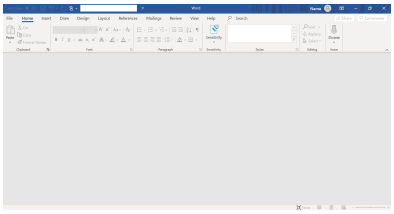
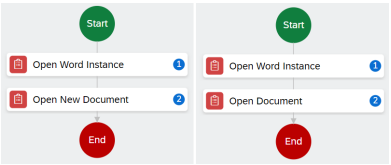
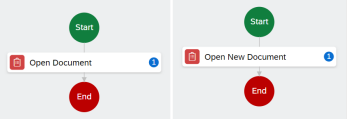
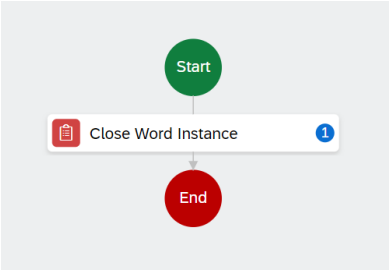
This section describes MS Word activities that help you design automations to perform various Word events in a Word document.

Prerequisites

Before you work with the Word activities, ensure you have added the SDK Word package in SAP Intelligent RPA Cloud Studio. For more details about adding SDK pacakges in the SAP Intelligent RPA Cloud Studio, refer to the Dependencies [page 525] section.

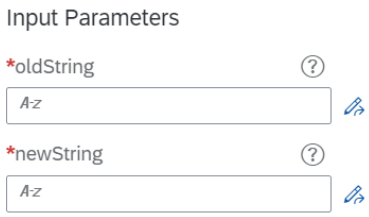
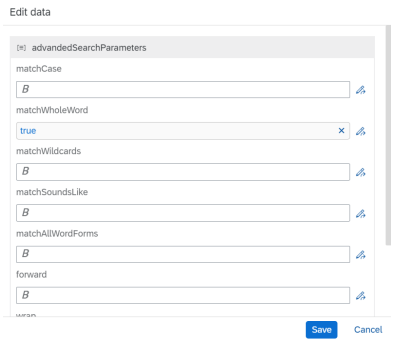
Application

Set of activities and functions related to Microsoft Word application.

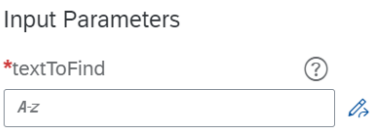
Activity Name	Description	Best Practice
Open Word Instance	Opens a new instance of MS Word. 	This activity is mandatory and helps to perform various Word events in a Word document. You must use this activity at the beginning of the workflow to use other MS Word activities.  **i Note** You cannot open a new or existing Word doc without this activity as the automation will be failed. 
Close Word Instance	Closes an instance of MS Word. Once closed, you need to open a new Word instance to use other MS Word activities.	You must use this activity with the **Open Word Instance** activity. You cannot close the existing word instance using this activity alone. 
Release Word Instance	Releases an instance of MS Word. **i Note** The MS Word instance will not be closed.	NA

Content

Collection of functions related MS Word content such as search text, insert text, replace.

Activity Name	Description	Best Practice
Replace All (Word)	Replace all occurrences of a string in the active document. For Example, replace all occurrences of the string **Hi World** with **Hello World** in the active document.	You must specify string values in the *oldString* and *newString* fields.  **i Note** By default, it is not case-sensitive. You can also find and replace string values by defining advance search criteria in the *advancedSearchParameters* pop-up screen.  *matchCase* – **Boolean**. By default it is False. This is an optional operation. The string value will be searched based on the **matchCase** (case sensitive) property criterion. True if the find operation is case sensitive. *matchWholeWord* – **Boolean**. By default it is False. This is an optional operation. The string value will be searched based on the **matchWholeWord** property criterion.

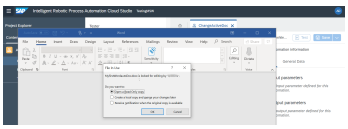
Activity Name	Description	Best Practice
		True if the find operation locates only entire words and not text that's part of a larger word. *matchWildcards* – **Boolean**. By default it is False. This is an optional operation. The string value will be searched based on the *matchWildcards* property criterion. True if the text to find contains wildcards. *matchSoundsLike* – **Boolean**. By default it is False. This is an optional operation. The string value will be searched based on the **matchSoundsLike** property criterion. True if words that sound similar to the text to find are returned by the find operation. For Example, "fun" (for instance, "funny"). *matchAllWordForms* – **Boolean**. By default it is False. This is an optional operation. The string value will be searched based on the **matchAllWordForms** property criterion. True if all forms of the text to find are found by the find operation (for instance, if the text to find is "sit," "sat" and "sitting" are found as well). *forward* – **Boolean**. By default it is False. This is an optional operation. The string value will be searched based on the forward property criterion. True if the find operation searches forward through the document. False if it searches backward through the document. *wrap* – **Boolean**. By default it is False. This is an optional operation. The string value will be searched based on the wrap property criterion.

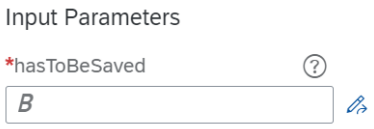
Activity Name	Description	Best Practice
		True to have the find operation searches wrap through the document. *format* – **Boolean**. By default it is False. This is an optional operation. The string value will be searched based on the format property criterion. True if formatting is included in the find operation.
Find Text (Word)	"Find a String representing the text or word to find in the active document. For Example, find all occurrences of the word **Hello World** in the active document."	You must specify to be found string in the *texttoFind* field.  You can also find string value by defining advance search criteria in the *advancedSearchParameters* pop-up screen. For more details about advance search criteria, refer to the Best Practices of the Replace All (Word) activity in this table.
Find Next (Word)	Find a String representing the text or word that matches those same conditions to find in the active document.	You must use this activity with the **Find Text (Word)** activity.
Insert Text (Word)	Insert a String representing the insert text or word in the active document.	This is case-sensitive. You must specify to be inserted string in the *textToInsert* field.

Document

Collection of functions related MS Word document such as open document, save document, save as, save as pdf.

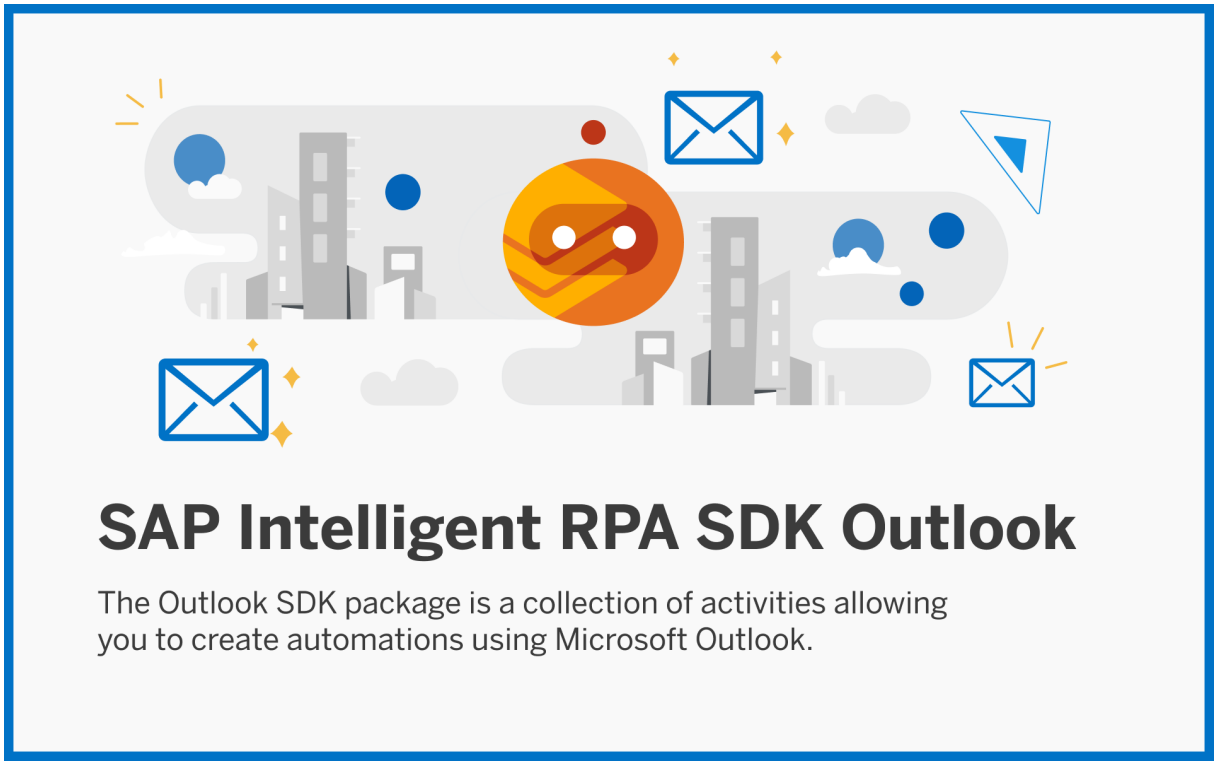
Activity Name	Description	Best Practice
Open Document (Word)	Opens a specified Word document.	1. To be opened document name and extension (document format) must be specified in the *filePath* field. 

Activity Name	Description	Best Practice
		For example, C:\Users**Folder NAME****DocName.docx** To enter the folder path and file name with extension, click in the *filePath* field and then select **EmptyText** and then click icon. 2. If the document is already opened and you run this activity, then the Word application shows **File In Use** pop-up and the automation may fail.  1. Unattended mode execution may fail.
New Document (Word)	Opens a Word document.	You must use this activity with the **Open Word Instance** activity.
Save As (Word)	Saves a specified Word document in a specified path.	You must specify a Folder path and Document name with file extension in the *filePath* field.  For example, C:\Users**Folder NAME****DocName.docx**
Save as PDF (Word)	Saves a specified Word document as PDF.	You must specify a Folder path and Document name with file extension in the *filePath* field.  For example, C:\Users**Folder NAME****PDFName.pdf**
Save (Word)	Saves an active Word document with the latest changes.	You must use this activity after the **Save As** activity. If you use this activty without the **Save As** activity then the Word application shows **Save this file** pop-up and the automation may fail. Unattended mode execution may fail.

Activity Name	Description	Best Practice
Close Document (Word)	Closes the specified Word document after saving all the latest changes.	You must specify a value (True or Flase) in the *hasToBeSaved* field. 
Set Bookmark Value	Sets a value to a specified bookmark range into the current Word document.	This activity sets the requested value in the bookmark placeholder and deletes the bookmark.

5.1.6 SDK Outlook Package

The Outlook SDK package is a collection of activities allowing you to create automations using Microsoft Outlook.



- https://help.sap.com/doc/e57d7b179db649e6b5f2d26d9894f9d9/Cloud/en-US/SDKdocumentation.html [https://help.sap.com/doc/e57d7b179db649e6b5f2d26d9894f9d9/Cloud/en-US/SDKdocumentation.html]

5.1.6.1 Best Practices for MS Outlook Activities

This section describes MS Outlook activities that help you design automations to perform various Outlook events such as search email, send email, delete email, save attachment, and many more.

> **Note**
>
> To get started with the **SDK Outlook Package**, you can download the Outlook Best Practices (mail) sample on the SAP Intelligent RPA Store.

Prerequisites

Before you work with the Outlook activities, ensure you have added the SDK Outlook package in SAP Intelligent RPA Cloud Studio. For more details about adding SDK pacakges in the SAP Intelligent RPA Cloud Studio, refer to the Dependencies [page 525] section.

Application

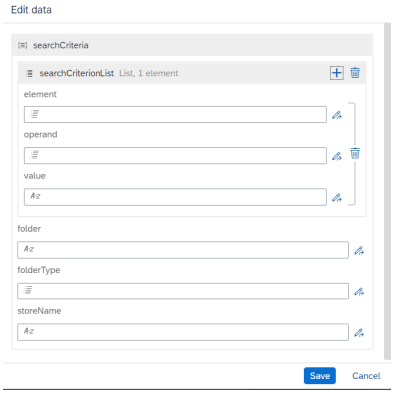
Set of activities and functions related to Microsoft Oulook application.

Activity Name	Description	Best Practice
Open Outlook Instance	Opens an instance of MS Outlook.	This activity is mandatory and helps to perform various Outlook events. You must use this activity at the beginning of the workflow to use other MS Outlook activities.
Close Outlook Instance	Releases the link between the SDK package and MS Outlook and closes MS Outlook.	You must use this activity with the **Open Outlook Instance** activity. You cannot close the existing Oulook instance using this activity alone.
Release Outlook Instance	Releases the link between the SDK package and Outlook, but Outlook remains open. **Note** The MS Outlook instance will not be closed.	Always use the *Release Outlook Instance* activity or *Close Outlook Instance* activity at the end of a workflow.

Email

Set of activities and functions related to Email.

Activity Name	Description	Best Practice
Search Email (Outlook)	Search for a specific email.	You must provide Email search criteria in the **Search Criteria Edit data** pop-up screen.

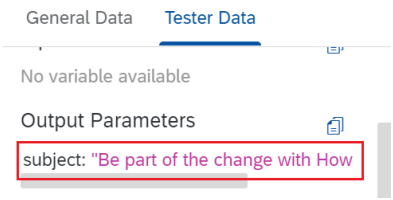
Activity Name	Description	Best Practice
		Note You can search email by defining one or more email search criteria.  • **element** ○ **subject** - Search email based on the email subject. ○ **senderName** - Search email based on the email sender name (you have received a mail from a person). ○ **recipientName** - Search email based on the email recipient name (you have sent a mail to a person). ○ **body** - Search email based on the email body content. **Note** The subject, senderName, recipiendName and body can be used only with operators equals and contains. ○ **dateReceived** - Search email based on the email received date. The date criteria must be in format yyyy_mm_dd format. Search email based on the email received date.

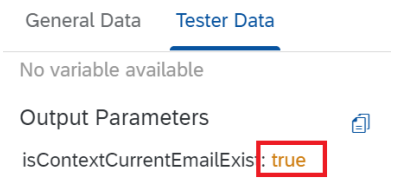
Activity Name	Description	Best Practice
		○ **dateSent** - Search email based on the email sent date. > **i** Note > **dateReceived** and **dateSent** - The date criteria must be in format yyyy_mm_dd. > The dateReceived and dateSent can be used only with equals, lessThan and greaterThan. ○ **hasAttachment** - It can be used only with equals operator and the value must be 0 (has no attachment) or 1 (has attachment(s)). ○ **isRead** - Search email based on the email read status (whether an email has been read or not). > **i** Note > If you want to search read emails, you can enter *true*, *1*, *yes* or *read* in the *Value* field. > If you want to search unread emails, you can enter *false*, *0*, *no* or *unread* in the *Value* field. • **operand** ○ **contains** - Search email based on the contains criterion such as some words matching with the subject line, .xls, .png, .cvs, .pdf, .docx , and many more. ○ **equals** - Search email only those which meet equals criterion (=".csv", "matching exact words of subject line") .

Activity Name	Description	Best Practice
		○ **greaterThan** - Search email based on the greaterThan email size criterion. For example, 2MB - Searches whose mail size is greater than or equal to 2MB. ○ **lessThan** - Search email based on the lessThan email size criterion. For example, 2MB - Searches whose mail size is less than or equal to 2MB. • **value** Provide the appropriate value. For example, if you are searching mail based on the subject then in the *Value* field you can provide the subject line. • **folder** Provide the name a folder that store email messages. • **folderType** Specifies the folder type for a specified folder. For example, olFolderInbox - The Inbox folder. • **storeName** A store is a file name that stores email messages. For example, **.pst** file. **i Note** **Folder** and **storeName** are not mandatory, but **folderType** is mandatory.
Send Email (Outlook)	Send an email	You must provide the one or more **Email Item Parameters** such as To, CC, and BCC. **i Note** One of the To, CC, or BCC value is mandatory but each of them are not mandatory.

Activity Name	Description	Best Practice
		• **To** – Optional value. Intended recipient email ID. • **CC** – Carbon Copy. Optional value. You can define CC recipients if you want to copy publicly. • **BCC** – Blind Carbon Copy. Optional value. You can define BCC recipients if you want to copy email privately. • **Subject** – Subject line of an email. Optional value. • **Body** – Content of an email. Optional value. • **Attachments** - Attachments to an email. Optional value. **i Note** You must specify a folder path in the field to add an attachment.
Create and Display Email (Outlook)	Create and Display a draft email.	The draft email will be created with the defined **Email Item Parameters** and displayed.
Save Attachment (Outlook)	Save an attachment.	The *destinationPath* and *attachmentFileName* are mandatory fields. You must specify a folder path in the *destinationPath* field to save an attachment. You must specify a file name with extension of a to be saved file in the *attachmentFileName* field. For example, **Filename.extension (Flower.png)**.
Save All Attachments (Outlook)	Save all the attachments.	The *destinationPath* is mandatory field. The *extensionsToKeep* and *extensionsToSkip* are optional fields. You must specify a folder path in the *destinationPath* field to save the attachments.

Activity Name	Description	Best Practice
		The two filters *extensionsToKeep* and *extensionsToSkip* are lists of extensions, separated by a "**;**" You must specify the extensions to be skipped in the *extensionToSkip* field. For example, **.png**. You must specify the extensions to be kept in the *extensionToKeep* field. For example, **.png**.
Delete Email (Outlook)	Delete an email from a specified location.	You must provide the appropriate email search criteria to delete required email. Deletes the current mail in the context. To have a context, use the *Search Email Activity*. To select a mail in the context, use *Get First Email*, *Get Last Email*, *Get Previous Email*, *Get Next Email* **i Note** To know more details about deleting multiple emails, refer to the Delete Multiple Emails in Outlook [page 437] section.
Move Email (Outlook)	Move an email from one location to another.	You must provide the appropriate email search criteria to move required email. You must provide the location details such as folder name or folder type in the *folder* (For example, **Archive**) or *folderType* (For example, **olFolderToDo**) field of the **Move Email (Outlook)** activity where the email is moved from the specified location. **i Note** **From** location (For example, **Inbox**) – The email is fetched from the **Outlook Inbox** folder and moved to the **Outlook Archive** folder. The From location details must be define in the **Email Search Criteria**.

Activity Name	Description	Best Practice
		To location (For example, **Archive**) - The email is fetched from the **Outlook Inbox** folder and moved to the **Outlook Archive** folder. The To location details must be define in the **Move Email (Outlook)** activity input parameter fields. To move multiple emails, follow the procedure detailed in Move Multiple Emails in Outlook [page 439].
Get Subject (Outlook)	Returns the subject of an email.	You must provide the appropriate search criteria in the Search Email activity. After successful execution of an automation, the email subject is displayed in the **Output Parameters** field of the **Tester Data**. 
Set Subject (Outlook)	Set the subject of an email.	
Get Body (Outlook)	Returns the body of an email.	You must provide the appropriate search criteria in the Search Email activity. After successful execution of an automation, the email body content is displayed in the **Output Parameters** field of the **Tester Data**.
Set Body (Outlook)	Set the body of an email.	
Get Mail Recipients (Outlook)	Returns the recipient of an email.	You must provide the appropriate search criteria in the Search Email activity. After successful execution of an automation, the Recipient email ids are displayed in the **Output Parameters** field of the **Tester Data**.
Set Recipient (Outlook)	Set the recipient of an email.	**TBD**

Activity Name	Description	Best Practice
Get CC (Outlook)	Receive the CC of an email.	You must provide the appropriate search criteria in the Search Email activity. After successful execution of an automation, the **cc'd** email ids are displayed in the **Output Parameters** field of the **Tester Data**.
Set CC (Outlook)	Set the CC of an email.	
Get BCC (Outlook)	Returns the BCC of an email.	You must provide the appropriate search criteria in the Search Email activity. After successful execution of an automation, the **bcc'd** email ids are displayed in the **Output Parameters** field of the **Tester Data**.
Set BCC (Outlook)	Set the BCC of an email.	
Get Sender (Outlook)	Returns the sender of an email.	You must provide the appropriate search criteria in the Search Email activity. After successful execution of an automation, the **sender's** email id is displayed in the **Output Parameters** field of the **Tester Data**.
Get Sent Date (Outlook)	Returns the sent date of an email.	You must provide the appropriate search criteria in the Search Email activity. After successful execution of an automation, the **sender's** email id is displayed in the **Output Parameters** field of the **Tester Data**.
Check Mail Context (Outlook)	Check if the current context of an email exists.	You must provide the appropriate search criteria in the Search Email activity. Returns True, if the context current email exists, False if it does not. 

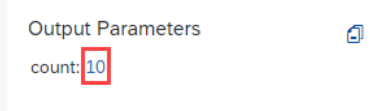
Activity Name	Description	Best Practice
Is Context Current Email Not Exist (Outlook)	Check if the current context of an email does not exist.	You must provide the appropriate search criteria in the Search Email activity. Returns True, if the context current email does not exist, False if it is.
Get Emails Number (Outlook)	Returns email count with respect to Number of Email Items in Context.	You must provide the appropriate search criteria in the Search Email activity. After successful execution of an automation, the email **count** is displayed in the **Output Parameters** field of the **Tester Data**.
Get First Email (Context)	Within a context, make the first email item as the current email item.	You must provide the appropriate search criteria in the Search Email activity. After successful execution of an automation, the output is displayed in the **Output Parameters** field of the **Tester Data**.
Get Last Email (Context)	Within a context, make the last email item as the current email item.	You must provide the appropriate search criteria in the Search Email activity. After successful execution of an automation, the output is displayed in the **Output Parameters** field of the **Tester Data**.
Get Next Email (Context)	Within a context, make the next email item as the current email item.	You must provide the appropriate search criteria in the Search Email activity. After successful execution of an automation, the output is displayed in the **Output Parameters** field of the **Tester Data**.
Get Previous Email (Context)	Within a context, make the previous email item as the current email item.	You must provide the appropriate search criteria in the Search Email activity. After successful execution of an automation, the output is displayed in the **Output Parameters** field of the **Tester Data**.
Get HTML Body (Outlook)	Returns the HTML body of an email.	You must provide the appropriate search criteria in the Search Email activity.

Activity Name	Description	Best Practice
		After successful execution of an automation, the email HTML body content is displayed in the **Output Parameters** field of the **Tester Data**.
Save As (Outlook)	Save the current context email as .msg.	You must specify a folder path in the *filePath* field to save an email as **.msg**.
Set Read (Outlook)	Flag the current context email as read.	
Set Unread (Outlook)	Flag the current context email as unread.	
Get Context Overview (Outlook)	Retrieve the context overview.	
Get Attachments Name	Retrieve the name of attachments.	You must provide the appropriate search criteria in the Search Email activity. After successful execution of an automation, the names of the attachements are displayed in the **Output Parameters** field of the **Tester Data**.

Calendar

Set of activities related to the Outlook calendar.

Activity Name	Description	Best Practice
Search Calendar Items	Search for the calendar items that match the search criteria, and fill the context with the result.	In the *calendarSearchCriteria* field, you must provide a list of search criteria with an account if the search is not in the default account. **i Note** Recurring calendar items are not returned by the Search Calendar Items activity.
Get All Calendar Items	Retrieve all the calendar items from the context.	After the Search Calendar Items activity with the appropriate search criteria, you can retrieve all the results with the Get All Calendar Items activity. In the **propertiesFilter** field, you can filter the items you want to retrieve. Filtering unneeded properties can help reduce the volume of the result and improve performances.

Activity Name	Description	Best Practice
		After successful execution of an automation, the result is displayed in the **Output Parameters** field of the **Tester Data** **i Note** The result may be shortened but the output **calendarItemsDetails** is always accurate.
Check Calendar Context	Checks if the current calendar item is defined.	You can use this activity to check if you are out of the restriced search list while analyzing the context with the Get Next Calendar Item or Get Previous Calendar Item activities. You can also use it to check if there is at least one result in the context after a search. Returns True, if the context current calendar item exists, False if it does not.
Get Calendar Items Number (Context)	Retrieve the number of calendar items within the calendar context.	You must provide the appropriate search criteria in the Search Calendar Items activity. After successful execution of an automation, the calendar items *count* is displayed in the **Output Parameters** field of the **Tester Data**. 
Get Calendar Item Details	Within the calendar context, retrieve the properties of the current calendar item.	Use this activity to get properties of one single calendar item or a few number of calendar items set in a loop. **i Note** If you want to get properties from a lot of calendar items, we recommand you use the Get All Calendar Items and loop on the result.

Activity Name	Description	Best Practice
		After successful execution of an automation, the output is displayed in the **Output Parameters** field of the **Tester Data**.
Get Next Calendar Item (Context)	Within a specified context, make the next calendar item as the current calendar item.	If you come out of the restricted content, the current calendar item will be undefined. The Check Calendar Context activity will return False.
Get Previous Calendar Item (Context)	Within a specified context, make the previous calendar item as the current calendar item.	If you come out of the restricted content, the current calendar item will be undefined. The Check Calendar Context activity will return False.
Save Calendar Item Attachment	Save the attachment of the current calendar item.	The *destinationPath* and *attachmentFileName* are mandatory fields. You must specify a folder path in the *destinationPath* field to save an attachment. You must specify a file name with extension of a to be saved file in the *attachmentFileName* field. For example, **Filename.extension (Item.jpg)**.
Save All Calendar Item Attachments	Save all attachments of the current calendar item.	The *destinationPath* is a mandatory field. You must specify a folder path in the *destinationPath* field to save an attachment. The *extensionsToKeep* and *extensionsToSkip* are optional fields. The two filters *extensionsToKeep* and *extensionsToSkip* are lists of extensions, separated by a "**;**" You must specify the extensions to be skipped in the *extensionToSkip* field. For example, **.png**. You must specify the extensions to be kept in the *extensionToKeep* field. For example, **.png**.

Activity Name	Description	Best Practice
Respond to Calendar Item	Respond to a meeting request.	You must provide one of the three following status in the *response* input parameter: • **accept** • **tentative** • **decline** **i Note** If you decline a calendar item, the item will be removed from the context.
Delete Calendar Item	Delete the current calendar item.	

5.1.6.1.1 Delete Multiple Emails in Outlook

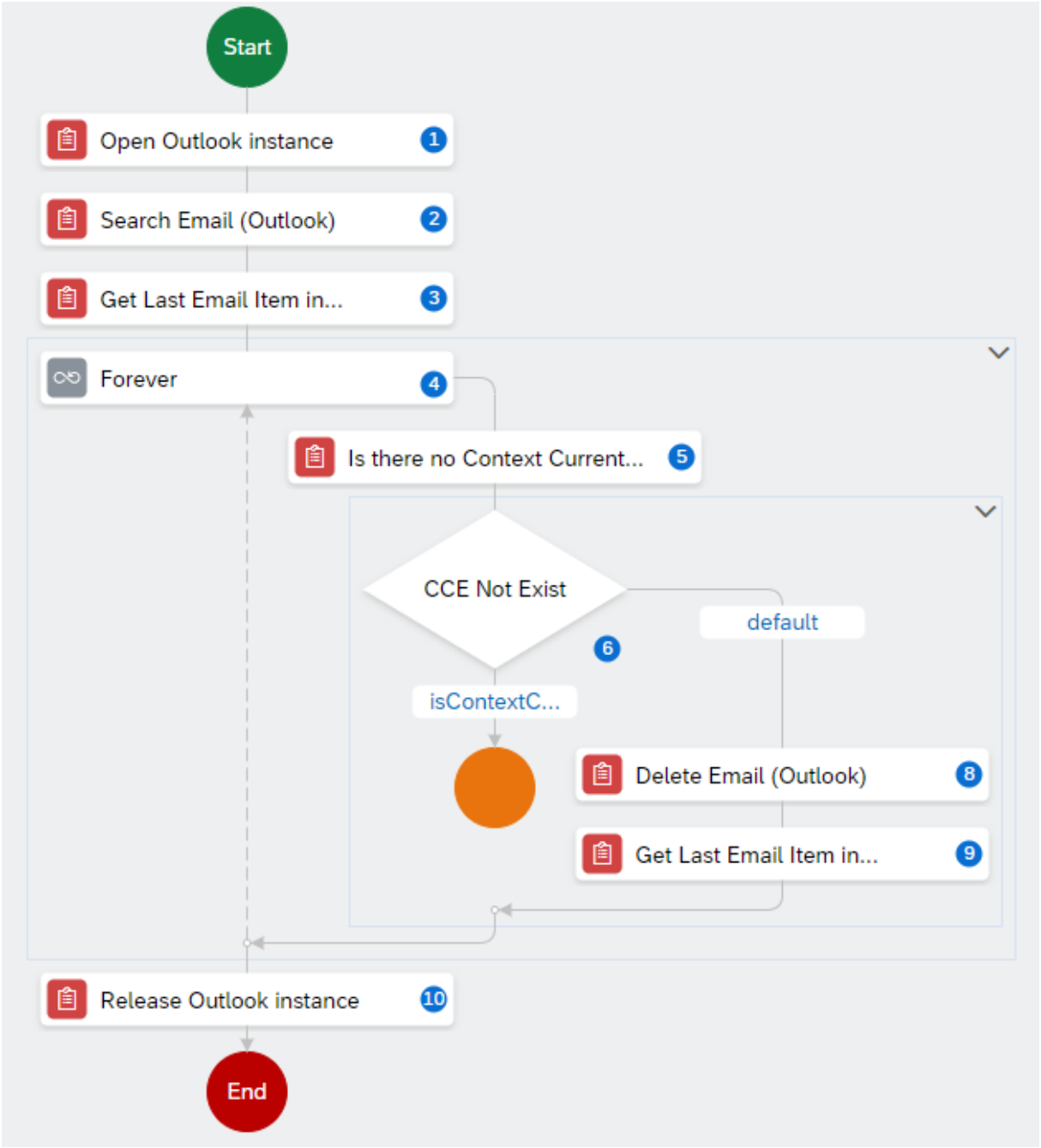
Best Practices.

The loop to delete multiple emails:

1. Always start with the *Open Outlook Instance* activity and at the end of the workflow use *Release Outlook Instance* or *Close Outlook Instance* activity.
2. Add appropriate email search criteria using the *Search Email (Outlook)* activity.
   To know more details about the *Search Email (Outlook)* activity, refer to the Best Practices for MS Outlook Activities [page 425] section.
3. Use the *Get Last Email Item in Context (Outlook)* activity to make the last email item as the current email item, within a context.
4. Use the appropriate controls. For example, *Forever* and *Condition*.
   To know more details about the *Forever* and *Condition* controls, refer to the Forever [page 224] and Condition [page 231] sections.
   To know more details about **Controls**, refer to the Add a Control to an Automation [page 216] section.
5. Delete the selected email using the *Delete Email (Outlook)* activity.
6. Use the *Get Last Email Item in Context (Outlook)* activity.

> **i Note**
>
> Selecting the previous mail in the loop is not possible because the current email has just been deleted. The good practice is to select the last email at each iteration.

The following screenshot shows the example workflow to delete multiple emails in Outlook.


Start
Open Outlook instance
1
Search Email (Outlook)
2
Get Last Email Item in...
3
Forever
4
Is there no Context Current...
5
CCE Not Exist
6
default
isContextC...
Delete Email (Outlook)
8
Get Last Email Item in...
9
Release Outlook instance
10
End

5.1.6.1.2 Move Multiple Emails in Outlook

Best practices for moving multiple emails in Outlook.

Context

Follow this procedure to move multiple emails in Outlook:

Procedure

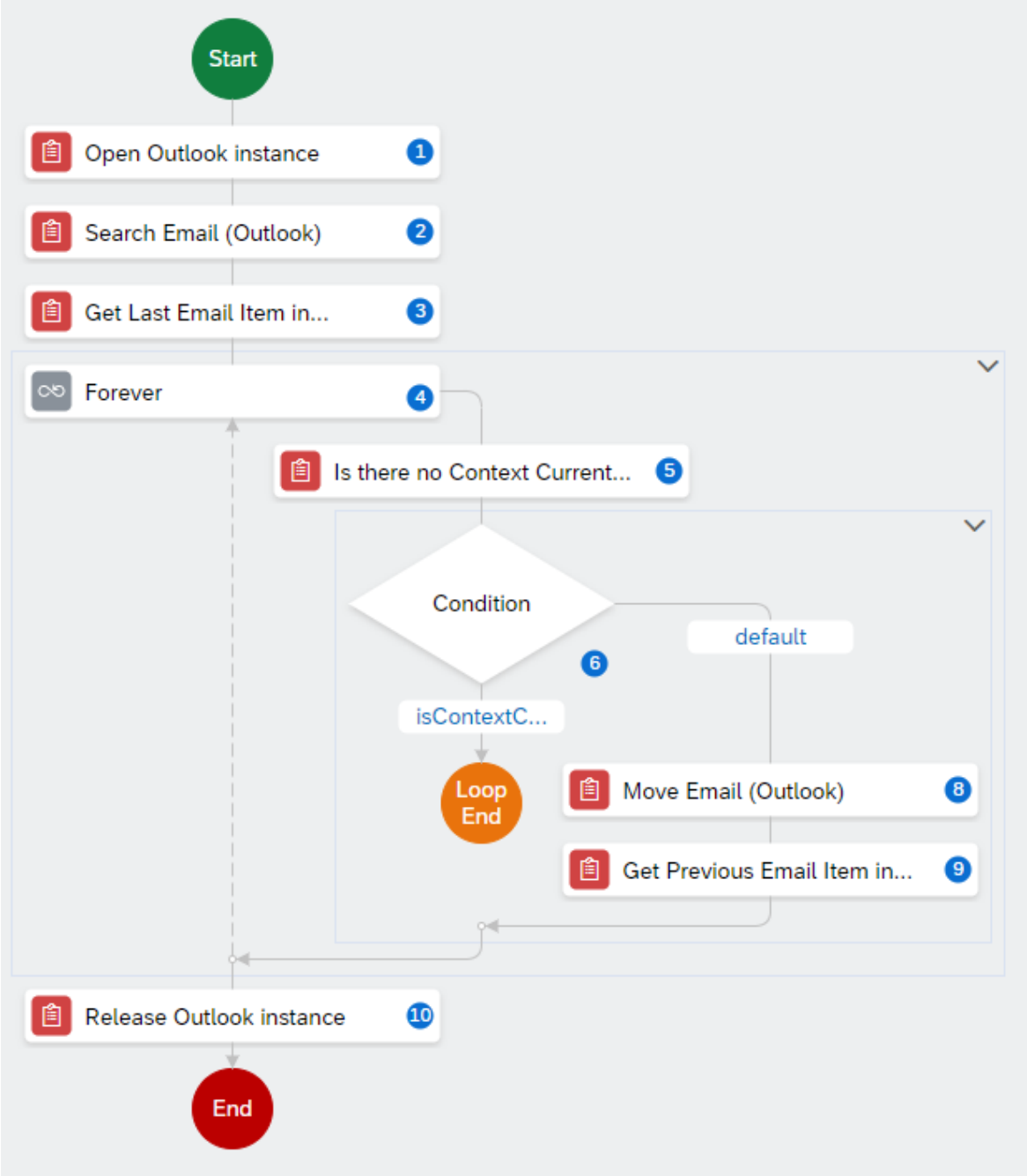
1. Always start with the *Open Outlook Instance* activity and at the end of the workflow use *Release Outlook Instance* or *Close Outlook Instance* activity.
2. Add appropriate email search criteria using the *Search Email (Outlook)* activity.

   To know more details about the *Search Email (Outlook)* activity, refer to the Best Practices for MS Outlook Activities [page 425] section.
3. Use the *Get Last Email Item in Context (Outlook)* activity to make the last email item as the current email item, within a context.
4. Use the appropriate controls. For example, *Forever* and *Condition*.

   To know more details about the *Forever* and *Condition* controls, refer to the Forever [page 224] and Condition [page 231] sections.

   To know more details about **Controls**, refer to the Add a Control to an Automation [page 216] section.
5. Move the selected email using the *Move Email (Outlook)* activity.
6. Use the *Get Previous Email Item in Context (Outlook)* activity.

   The following screenshot shows an example of workflow to move multiple emails in Outlook.



5.1.6.1.3 Get Context Overview Activity

This Get Context Overview activity returns an overview of the context.

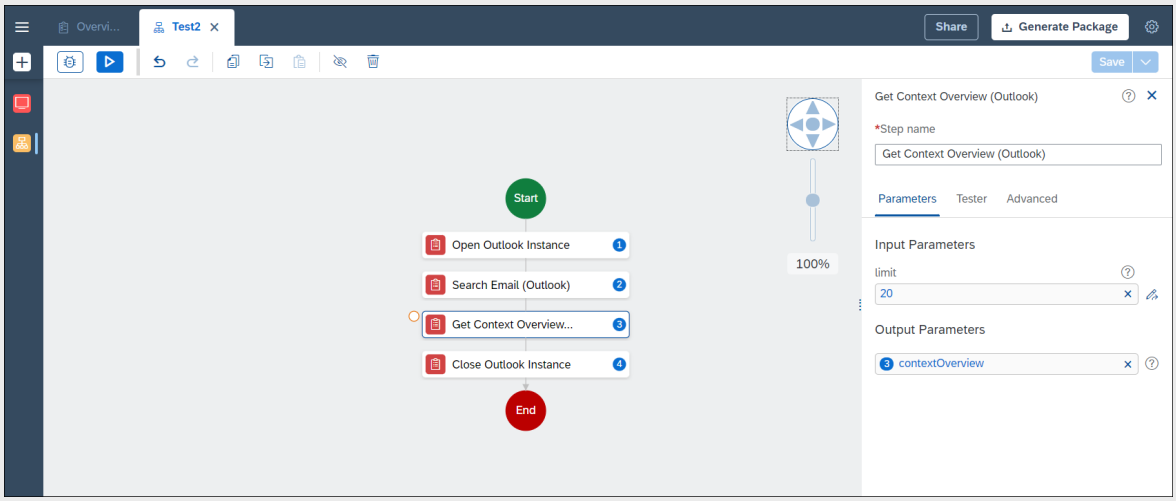
At design time, it is useful to have a clear view of what the last mail search returned and what is the current selected email. It is recommended to delete or disable the **Get Context Overview** at runtime as it may have an impact on performance if you have used the **Get Context Overview** activity in an automation.

The **Get Context Overview** returns:

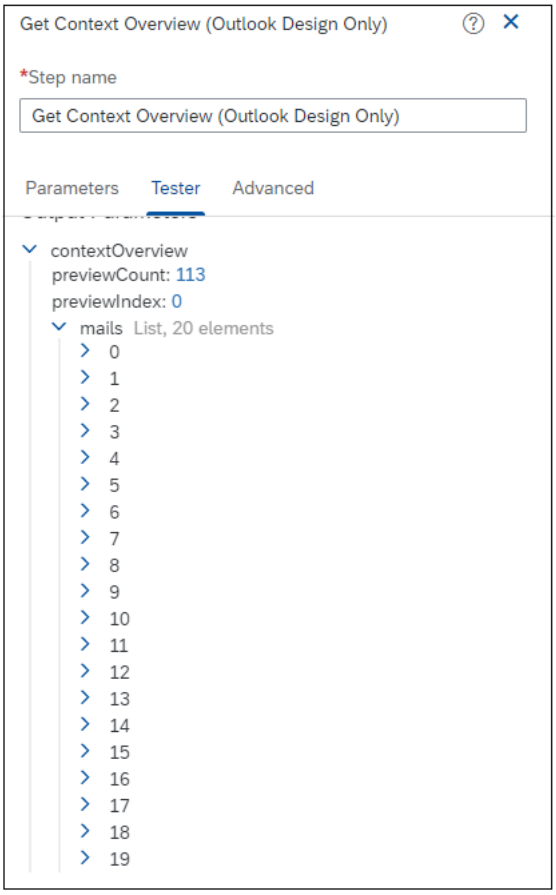
- *previewCount*: The exact number of emails in the context.
- *previewIndex*: The index of the current email in the context.
- *mails*: A limited preview list of emails in the context. For each email preview, there is a title, sender, and a date.

The *limit* parameter (default value, 20) cuts the preview list of emails if needed to avoid performance issues when there are too many mails to return.

> **i Note**
>
> If you do not provide a limit value in the input parameter field, then by default, the limit value is 20.



You only get a maximum of *limit* email samples, but the *previewCount* is always accurate.



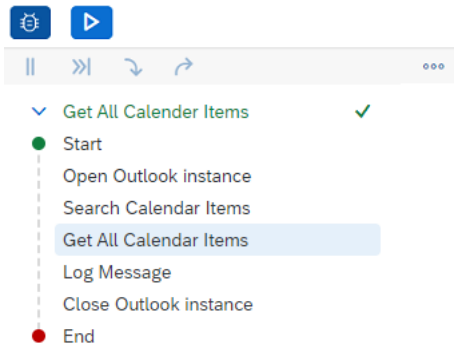
5.1.6.1.4 Get All Calendar Items Activity

This *Get All Calendar Items* activity returns all items from your *Outlook* calendar including:

- *Subject*
- *Start Date Time*
- *End Date Time*
- *Categories*
- *Duration*
- *Body*
- *Attachments*
- *Recipients*

When you use this activity, you can set the *Limit* parameter that allows you to restrict the number of items to return. The default value is **100** elements.

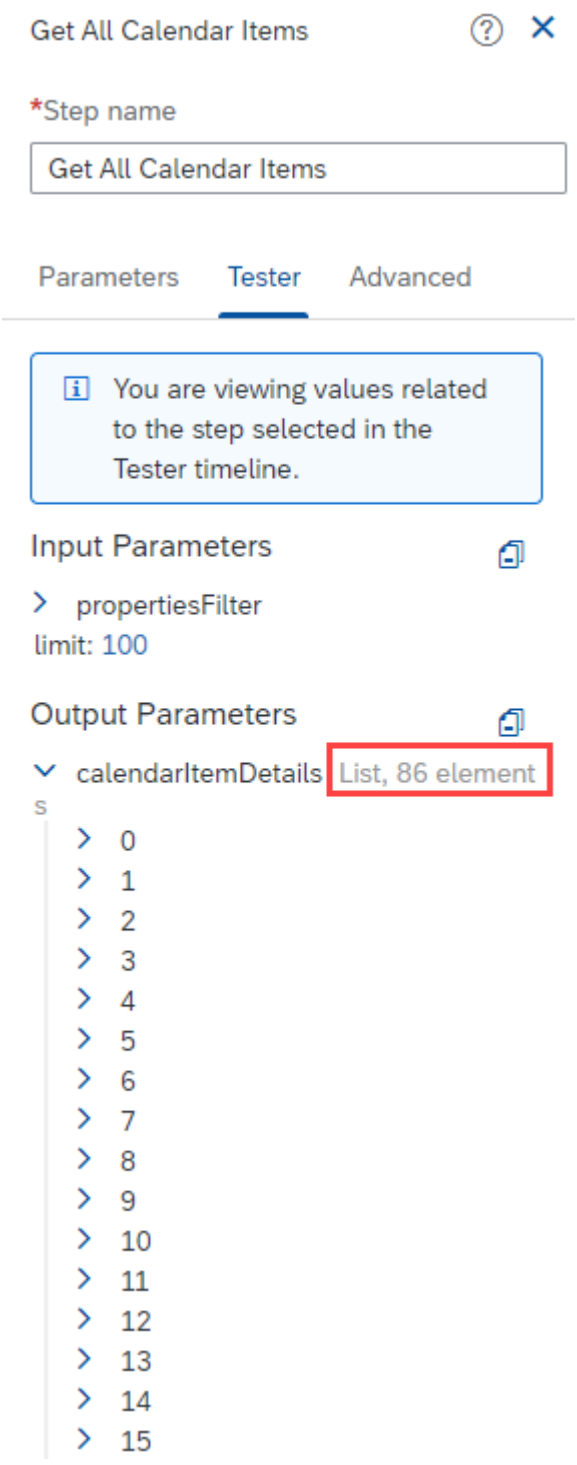
This activity returns a table that contains all items from your *Outlook* calendar. You can see this table if you click the *Get All Calendar Items* activity in the *Tester* timeline.



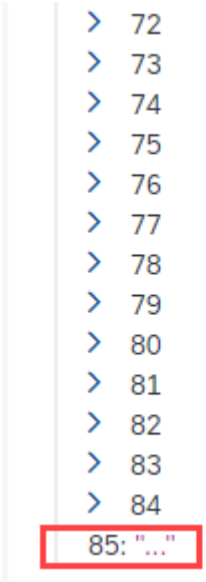
This table appears on the *Tester* panel, under *Output Parameters*.

> **Note**
>
> Sometimes, the list of elements displayed in this area can be truncated.

For example, you asked the *Get All Calendar Items* activity to return 100 elements, but only 86 elements appear on this panel.



You can see that the results are truncated.



This can be due to the fact that you asked the activity to return the elements bodies. Indeed, bodies of the calendar items are truncated to 50 characters under *Output Parameters* because you only need this data for debug purposes.

However, this does not correspond to the number of elements in the activity output.

If you add a *Log Message* activity and select *calenderItemDetails* as *message* on the *Input Parameters* panel. You can notice that the activity has returned 100 calender items. The activity output is complete.

5.1.6.1.5 How to Send Multiple Attachments in an Email

Context

Best practice to attach multiple files to an email when you don't know all the files during the design of your automation.

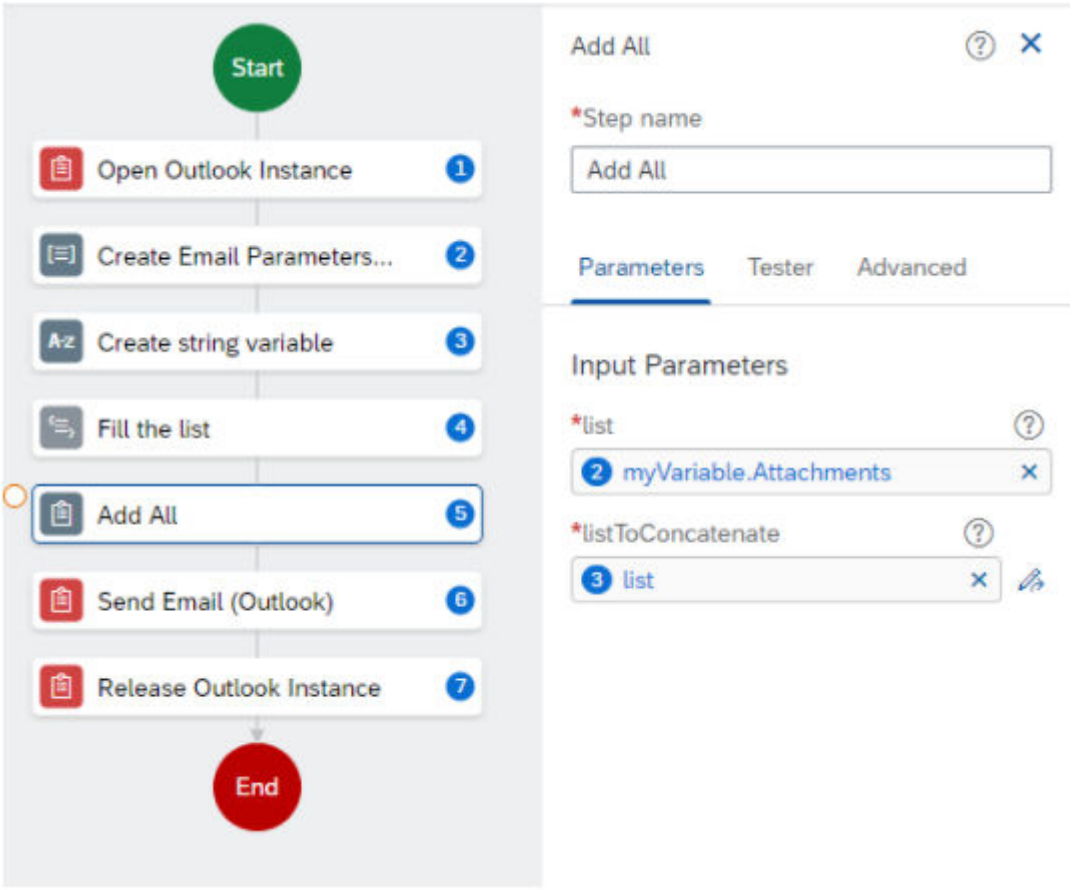
Procedure

1. Insert a variable of type *Email Parameters*.
2. Insert a variable of type *String*. Check the option to make it a list.
3. Add as many items as needed in the list, each item being the path of a file to be attached.
4. Insert the *Add All* activity once the list is complete.

   The **listToConcatenate** attribute must contain the list you created in step two. The **list** attribute must be `myVariable.Attachments`, `myVariable` being the *Email Parameter* created in step one.

5. Insert the activity *Send Email (Outlook)* and use the *Email Parameters* variable as input.

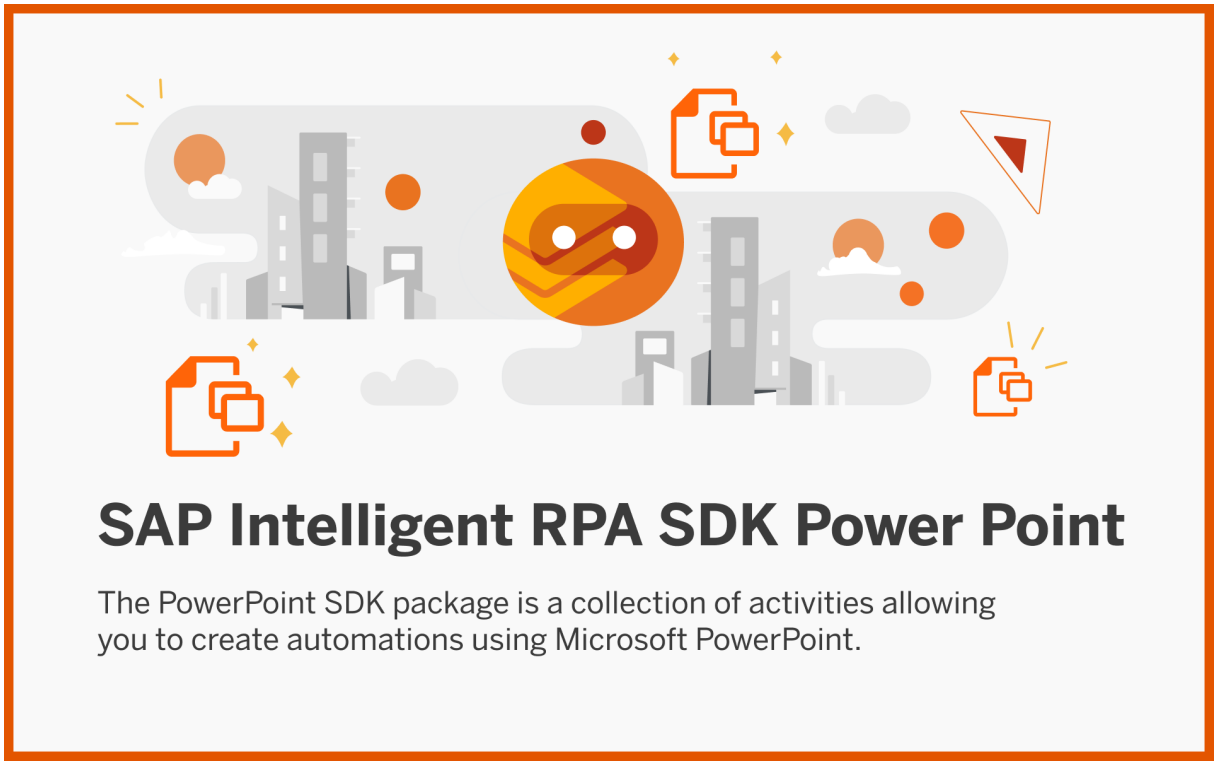
   The following screenshot shows an example workflow to send multiple files.



> **i Note**
>
> You can also use the *Add Item* activity to directly add a string to the attachments list in the *Email Parameter*. You can do it inside a loop, or multiple times in your automation for the same result.

5.1.7 SDK PowerPoint Package

The PowerPoint SDK package is a collection of activities allowing you to create automations using Microsoft PowerPoint.



- https://help.sap.com/doc/57976597a63b424e829153102374b52e/Cloud/en-US/SDKdocumentation.html [https://help.sap.com/doc/57976597a63b424e829153102374b52e/Cloud/en-US/SDKdocumentation.html]

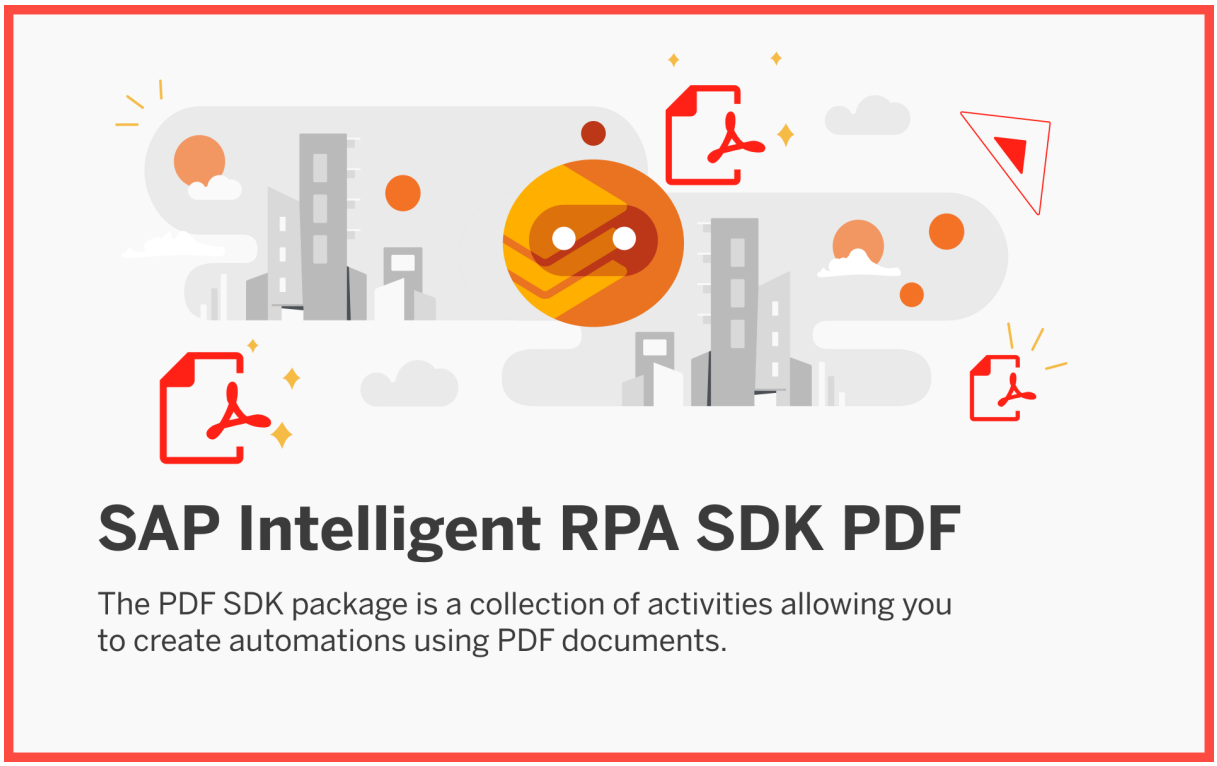
5.1.7.1 Best Practices for MS PowerPoint Activities

Open and Close MS PowerPoint Instances

Keep in mind that MS PowerPoint - unlike Word or Excel - is not multi instance. Thus, when the 'Close' activity is triggered, all running instances of PowerPoint will be closed too.

5.1.8 SDK PDF Package

The PDF package is a collection of activities allowing you to create automations using PDF.

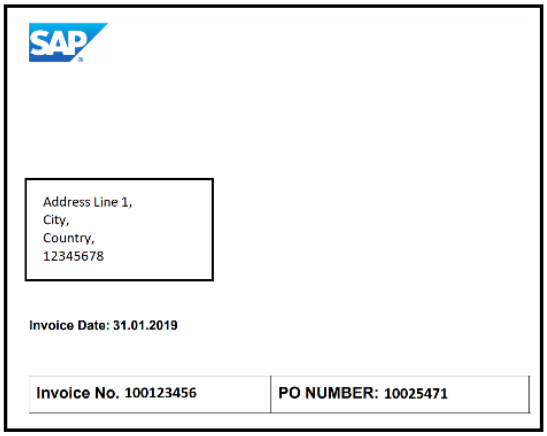


- https://help.sap.com/doc/090ad7af28184e309781d1d0e7debc27/Cloud/en-US/SDKdocumentation.html [https://help.sap.com/doc/090ad7af28184e309781d1d0e7debc27/Cloud/en-US/SDKdocumentation.html]

5.1.8.1 Use a Regular Expression

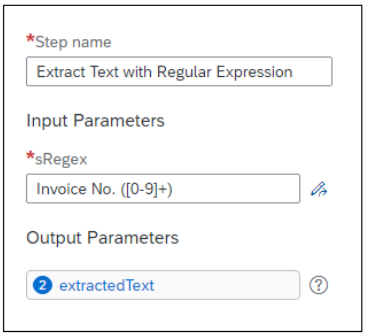
Using a regular search expression to extract a string from the page or specific page area defined by filters, will return the first match found. It also supports the use of capturing groups in the regular expression. If a capturing group was used, the first capturing group is returned as a result.

Sample PDF area



In the above PDF sample, **Extract Text with Regular Expression** could be used to extract the Invoice Number.

Activity 1

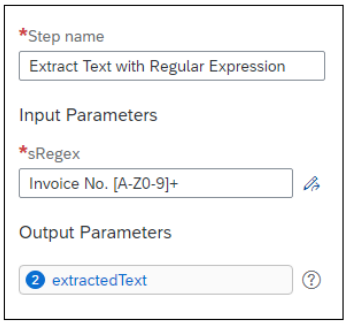


Expression: Invoice No. ([0-9]+)

Output: 100123456

The second part of the expression is surrounded by parentheses, marking a capturing group. As a result, only the string that meets the criteria and comes after "Invoice No. " is returned.

Activity 2

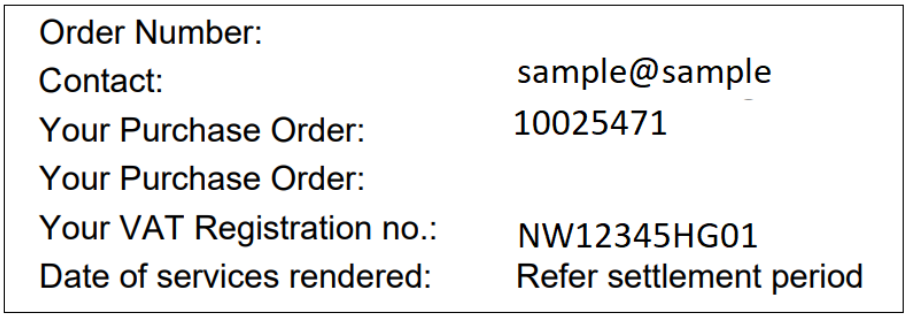


Expression: Invoice No. [A-Z0-9]+

Output: Invoice No. 100123456

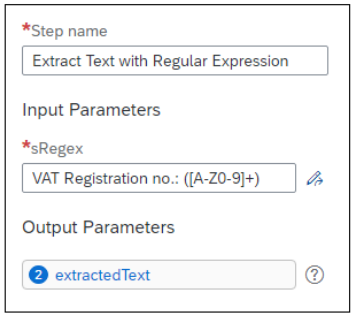
In this example, the parentheses are omitted in the regular expression. So, the entire matching string is returned.

Sample PDF area



In the above PDF sample, **Extract Text with Regular Expression** could be used to extract the field Your VAT Registration no.

Activity 1



Expression: VAT Registration no.: ([A-Z0-9]+)

Output: NW12345HG01

5.1.8.2 Best Practices for PDF Activities

This section describes PDF activities that help you design automations to work with pdf documents.

> **i** Note
>
> These PDF activities for Cloud Studio have some limitations regarding the documents to be read. The documents must contain fewer than 4000 pages. This limit has been greatly improved compared to the SAP Intelligent RPA 1.0 PDF library in which documents could not contain more than 100 pages.

Prerequisite

Before you work with the PDF activities, ensure you have added the SDK PDF package in SAP Intelligent RPA Cloud Studio. For more details about adding SDK pacakges in the SAP Intelligent RPA Cloud Studio, refer to the Dependencies [page 525] section.

Open PDF

Use the *Open PDF* activity to open a PDF document.

This activity is mandatory when using a PDF document. You must drop this activity as first step of your workflow to use other PDF activities.

You must provide the input parameters of the activity:

- **pdfPath**
  Specify a document file to open a pdf attachment. For example `C/Users/FolderName/DocName.pdf`.
- **password**
  Provide a password to open protected PDF documents.
- **reOrdeByPosition**
  The text you get of a PDF can be unordered. Select *true* or *false* to reorder text indexes according to their position in this document.

  > **i** Note
  >
  > By default, it is **false**.

Extract text w/Reg. Expr.

Use this activity to declare a regular expression to extract specific text areas in your PDF document.

You must provide a regular expression in the *sRegex* field. To set an expression, click the button to open the *Expression editor*.

Get Page Dimensions

Use the *Get Page Dimensions* activity to return the dimensions of a specified page.

You must enter a page number in the *pageNum* field in the input parameters of the activity.

If you provide a wrong input for the page number parameter, the automation throws and displays a specific error in the logs at execution time.

5.1.8.2.1 How to Retrieve the Column Data in PDF Table

There are lots of new activities associated with the PDF library in the Cloud Studio. You can retrieve data from the column of a PDF table using the *Get Table Column Entries (PDF)* activity.

Use Case Example

The following use case demonstrates how to retrieve data from the column of a PDF table.

Prerequisites

You have created one or several automations in the Cloud Studio, as described in Create an Automation [page 156].

For example, refer to the following table. You can retrieve the data from three columns of a PDF table using the *Get Table Column Entries (PDF)* activity three times.

Items

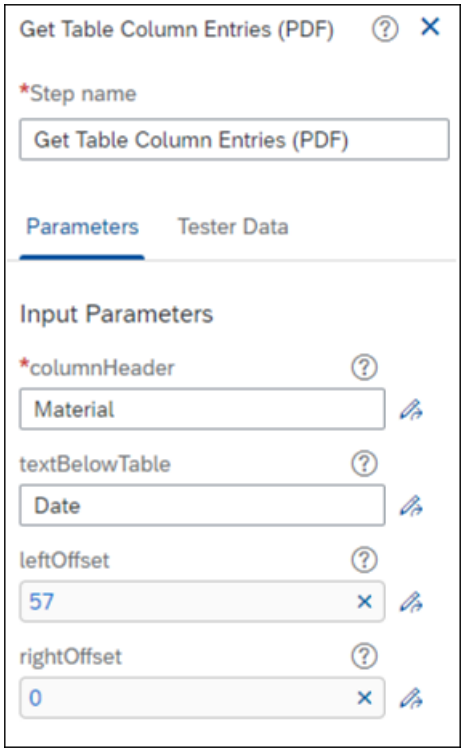
Material	Order Quantity	Storage Location
HW-1761	10	0001
HW-1761	20	0001
HW-1761	30	0001

Date : 17/11/2020

Procedure

1. In your automation, on the right-hand side panel, go to the *Tools* section and click *Activities*.
2. Drag and drop the *Search Text Items (PDF)* activity into the workflow of your automation.

3. Drag and drop the *Get Table Column Entries (PDF)* activity into the workflow of your automation.
4. Select the *Get Table Column Entries (PDF)* activity. The *Get Table Column Entries (PDF)* side panel appears. Enter values in the Input Parameters.



5. Click *Save* to save your changes.
6. Click *Test*.

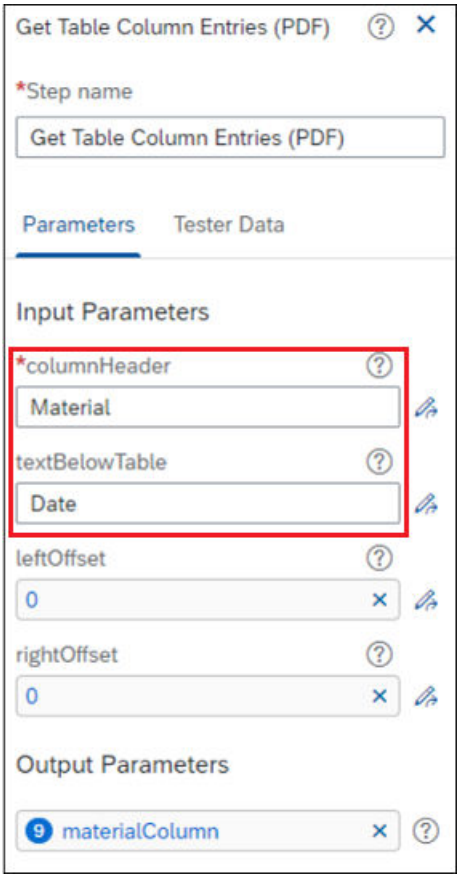
Result

The data is retrieved from the column of a PDF table.

Further Details

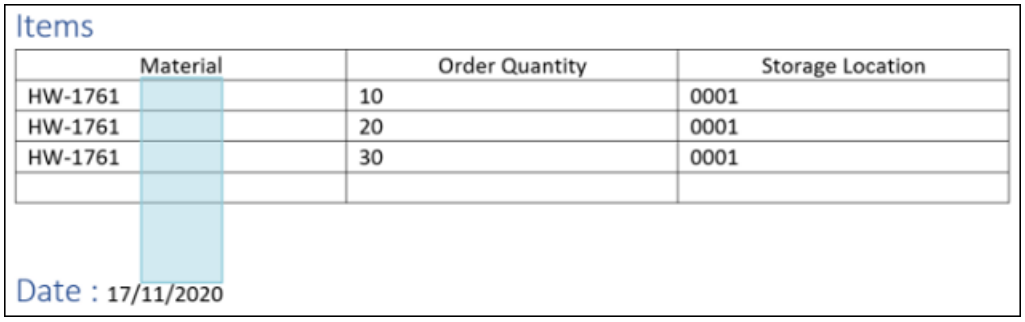
Basic Usage of the Get Table Column Entries (PDF) Activity

For the basic usage of the activity, you must set input parameters only for the *columnHeader* and *textBelowTable* fields.



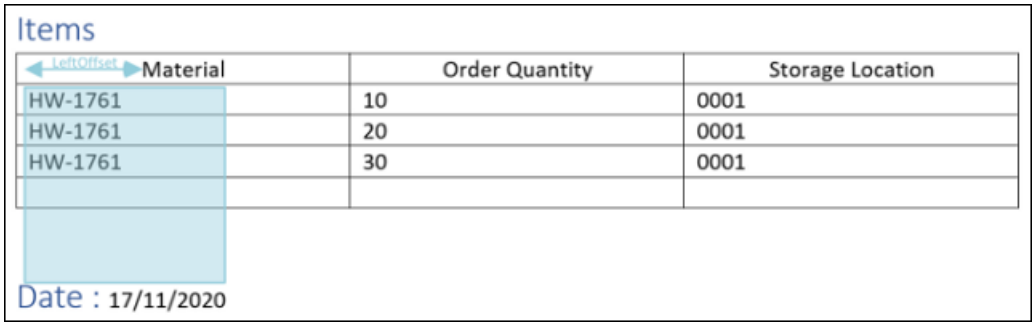
The activity will retrieve the column data between the letter “M” and “L” and below the label “Material” until the label “Date”. If you don’t set the value for the *textBelowTable* field, the activity will retrieve the column data below all the labels until the end of the file.

In this case, the activity will not retrieve any column data as there is no data between the letter “M” and “L”.

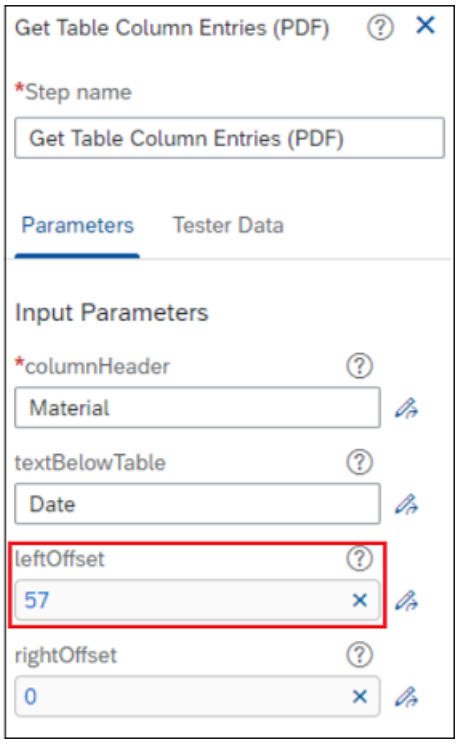


Offset Management of the Get Table Column Entries (PDF) Activity

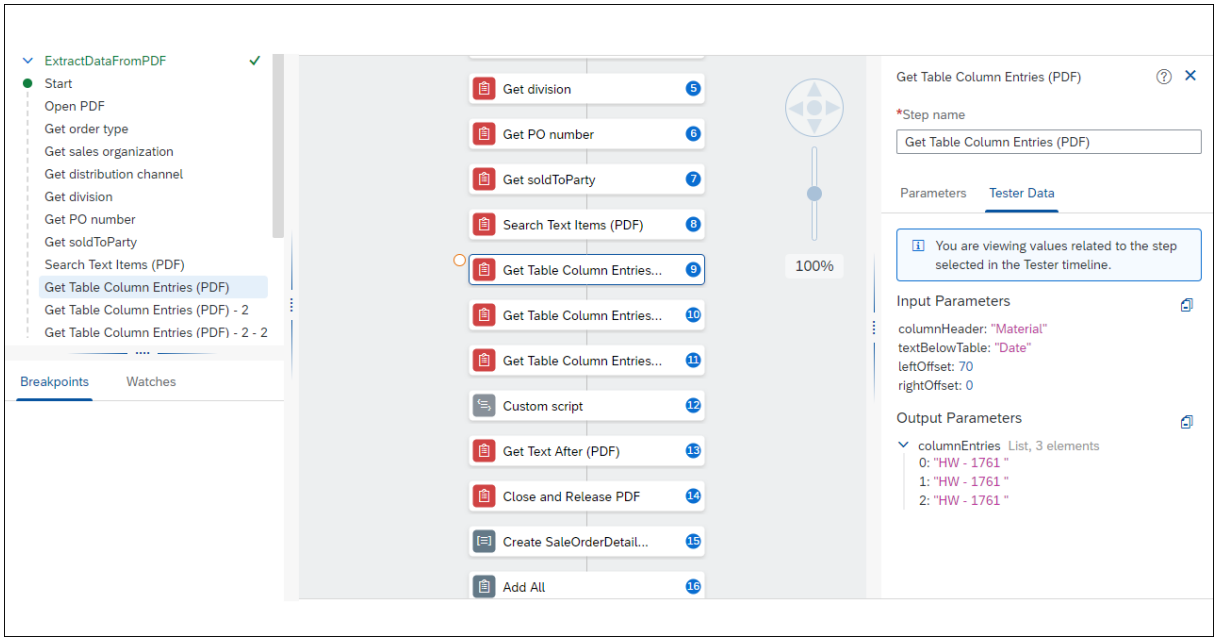
To retrieve the column data of the correct position, you must set values to the offset parameters. The offset parameters are the number of offsets to add to the header label position.



In this case, you must set input parameter for the *leftOffset* to retrieve the column data.



Thus, you can retrieve all the data of the material column.



Calculate the Value of the Offset Parameters

You can calculate the number of offsets to retrieve the column data.

In this case, you need to calculate the value of the left offset parameter to retrieve the column data by using the left position of the label “Material” and the left position of “Items”. You can use the *Search Text Items (PDF)* activity to determine the correct value of the offset parameters.



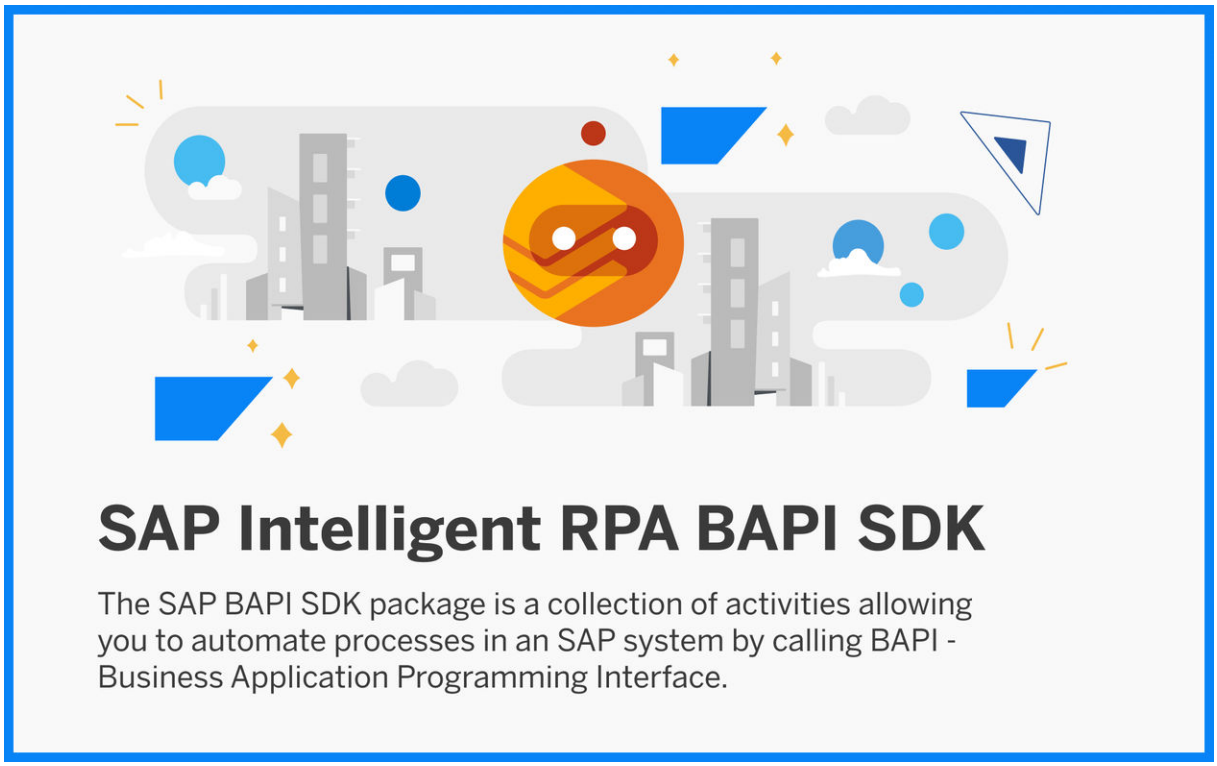


Once you test the automation, you can see that the left position of the label “Material” is 127.73 and the left position of the label “Items” is 70.824. So, the value of the *leftOffset* parameter is 127.73 - 70.824 = 56.906 or 57.

You can also use an expression and let the automation calculate the correct value of the offset parameters.

5.1.9 SDK BAPI Package

The BAPI SDK package is a collection of activities allowing you to create automations in SAP system by calling BAPI, which is the standard interface to the business object model in SAP products.



- https://help.sap.com/doc/f8df5abbb62545d080a6db6daf344013/Cloud/en-US/SDKdocumentation.html [https://help.sap.com/doc/f8df5abbb62545d080a6db6daf344013/Cloud/en-US/SDKdocumentation.html]

5.1.9.1 BAPI Overview

BAPIs (*Business Application Programming Interface*) are specific methods for SAP business objects, which are stored in the Business Object Repository (BOR) of the SAP system and are used for carrying out specific business tasks.

In the SAP system, BAPIs are stored as RFC (*Remote Function Call*) capable function modules in the ABAP (Advanced Business Application Programming) Workbench Function Builder. BAPIs have standard business interfaces that enable external applications (with the help of SAP business objects) to access SAP processes, functions, and data.

Client programs that use BAPIs to access SAP business objects can be part of the same SAP system, part of an external system (for example, a .Net application), part of an HTTP Gateway, or part of another SAP system.

If you want to communicate with BAPIs, you usually use various technical connectors such as JS (*JavaScript*), Java, VB (*Visual Basic*), and ABAP.

Prerequisites

This topic outlines the following requirements you must meet to use a BAPI method in an automation:

- Desktop agent version must be equal to or greater than 2.0.16.
- For *Set SAP Connection (SSO)* SDK BAPI activity, SSO (*SAP Single Sign-On*) must be configured on your system.

Related Information

SDK BAPI Package [page 458]
Best Practices for SAP BAPI Activities [page 467]

5.1.9.1.1 Create an Automation for BAPI

Use Case Example

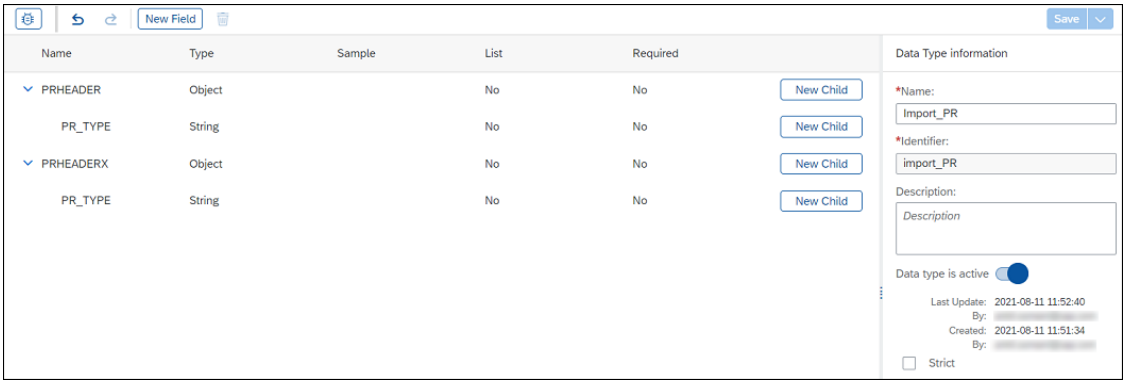
The following use case demonstrates how to execute an automation for BAPI.

Prerequisites

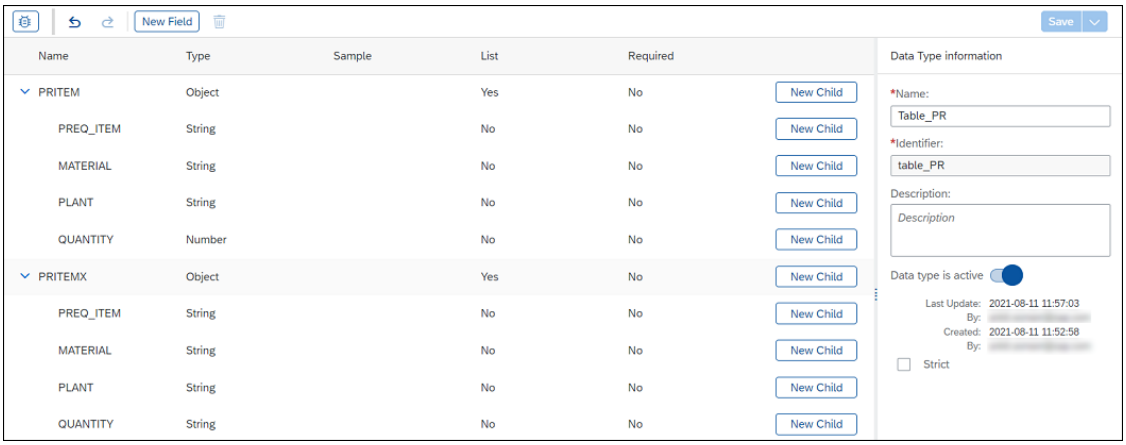
- Configure SAP connection (SSO or Basic) on your system.
- You have created at least one automation in the Cloud Studio. For more information on how to create an automation, see Create an Automation [page 156].

  > **i Note**
  >
  > Make sure that desktop agent version is equal to or greater than 2.0.16.

- Add Core and BAPI SDK packages as dependencies. For more details about adding a new dependency, see Add a New Dependency [page 528].
- Create data types for import, table, and changing parameters. For more details about creating a data type in the Cloud Studio, see Create a Data Type [page 168].
  - *Data Type for Import Parameter*: The following screenshot is an example of *Import_PR* data type for import parameter.

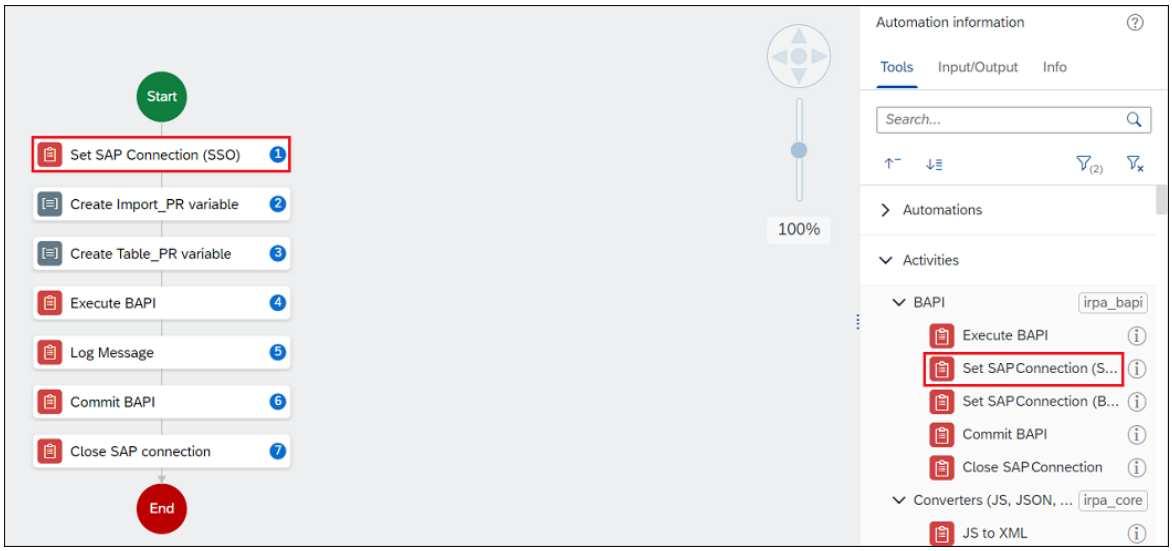


- *Data Type for Table Parameter*: All table parameters must be of list type. The following screenshot is an example of *Table_PR* data type for table parameter.



Procedure

1. In your automation, drag and drop the *Set SAP Connection (SSO)* activity.



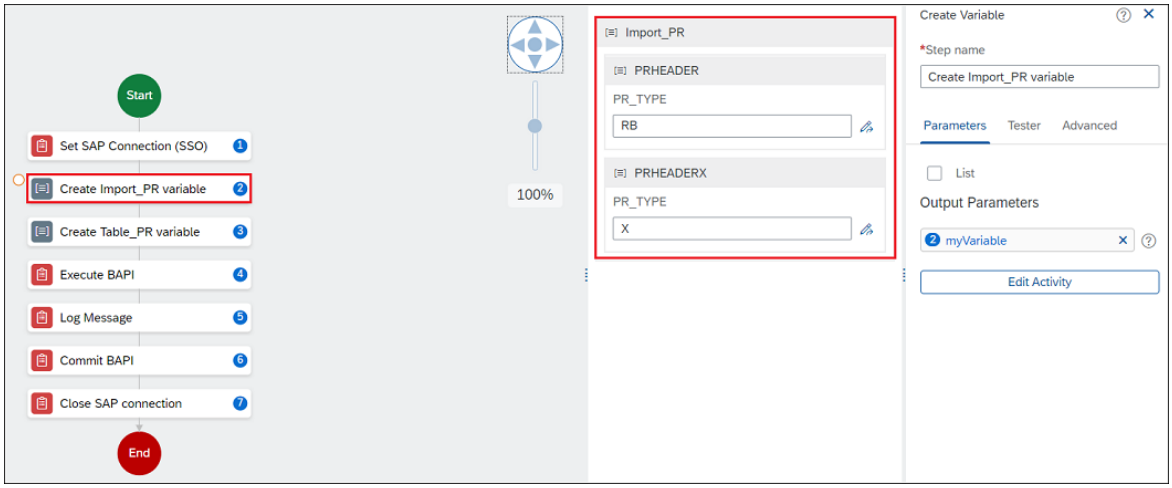
2. Select the *Set SAP Connection (SSO)* activity. The step details side panel appears. Enter the values in the *Input Parameters*.



3. Drag and drop the *Import_PR* data type to create an Import_PR variable.



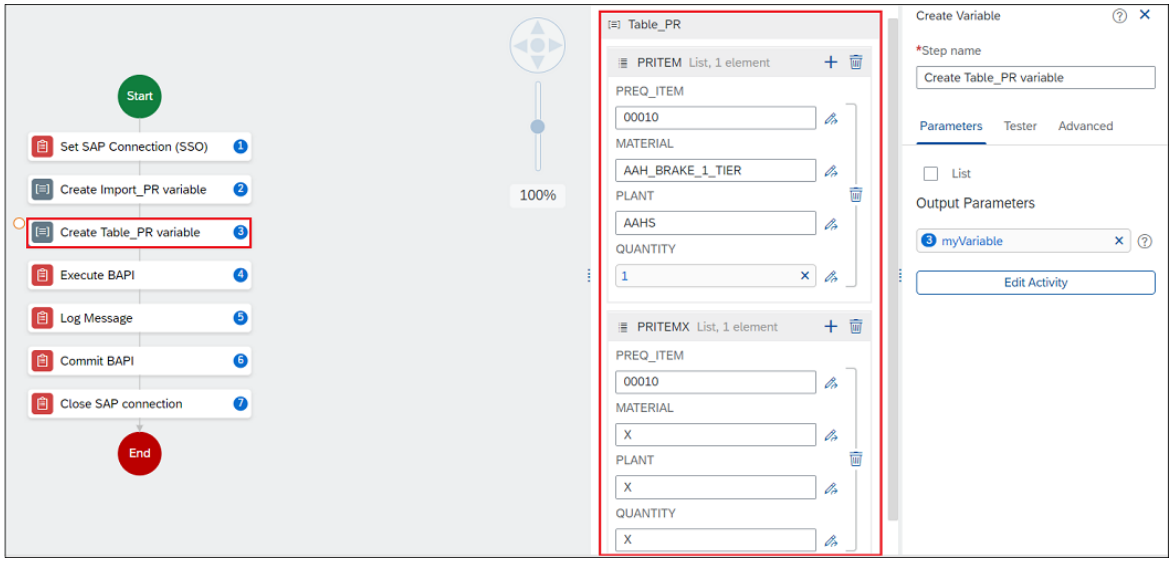
4. Select *Create Import_PR variable*. The step details side panel appears. Enter the parameter values for import parameter.



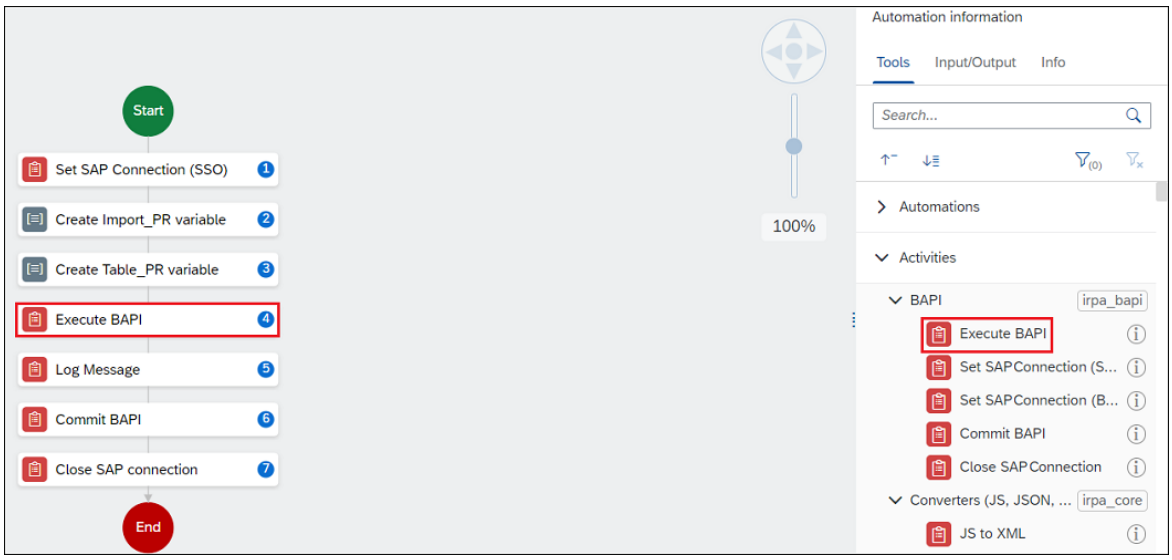
5. Drag and drop the *Table_PR* data type to create an Table_PR variable.



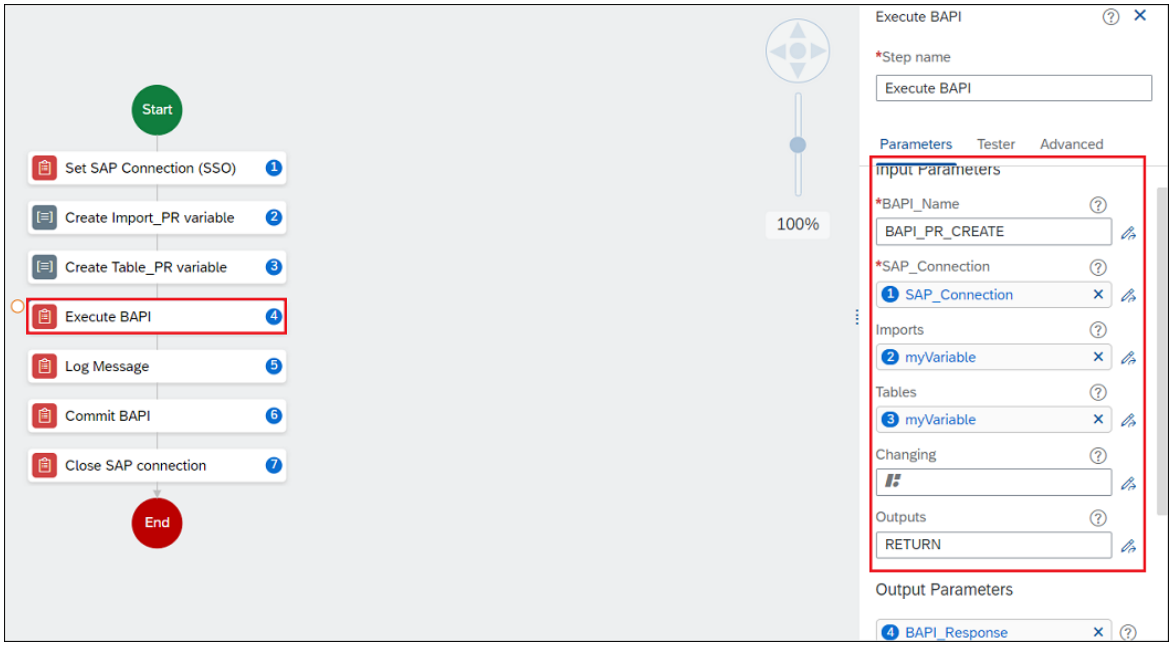
6. Select *Create Table_PR variable*. The step details side panel appears. Enter the parameter values for table parameter.



7. Drag and drop the *Execute BAPI* activity.



8. Select the *Execute BAPI* activity. The step details side panel appears. Enter the values in the side panel appears. *Input Parameters* to execute the BAPI.



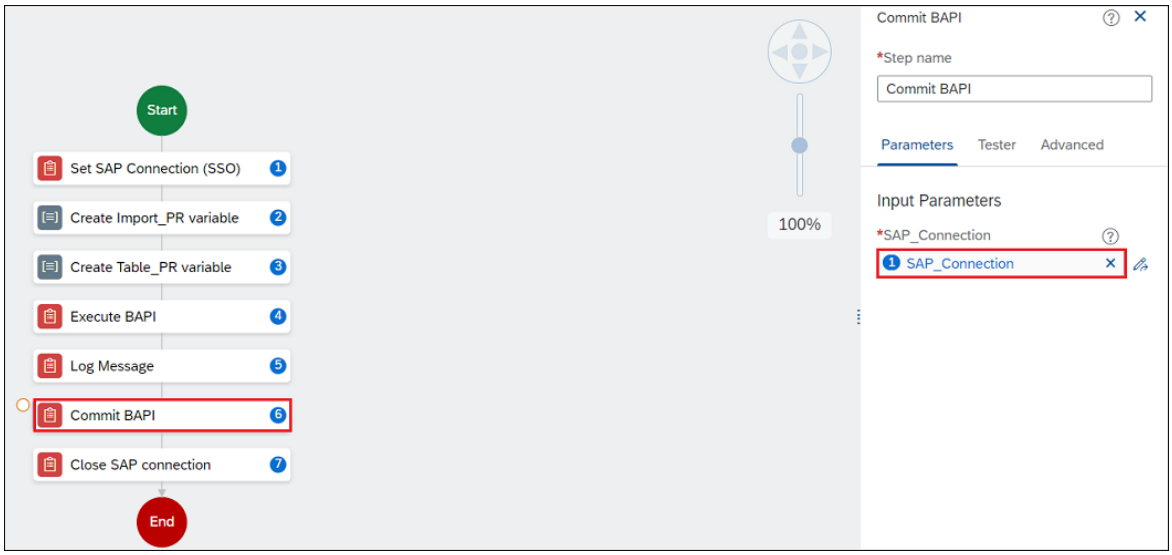
9. Add a *Log Message* activity with the input message *BAPI_Response*.



10. Drag and drop the *Commit BAPI* activity.



11. Select the *Commit BAPI* activity. The step details side panel appears. Enter the value in the *Input Parameters* to commit transactions of the BAPI.



12. Drag and drop the *Close SAP Connection* activity.



13. Select the *Close SAP Connection* activity. The step details side panel appears. Enter the value in the *Input Parameters* to close the SAP connection.



14. *Save* the automation.
15. Click *Test* to test your automation.

Results

The BAPI response is displayed in JSON format.



5.1.9.2 Best Practices for SAP BAPI Activities

This section describes BAPI activities that help you to design automations for BAPI.

Prerequisites

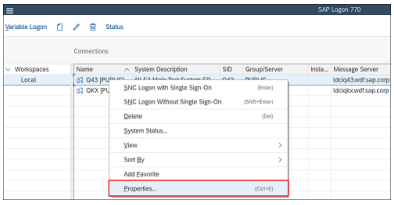
Before you work with the BAPI activities, ensure you have added the SDK BAPI package SAP Intelligent RPA Cloud Studio. For more details about adding SDK pacakges in the SAP Intelligent RPA Cloud Studio, refer to the Dependencies [page 525] section.

Activities

The following is the set of activities related to BAPI.

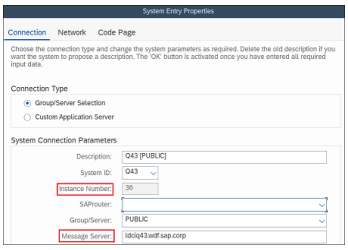
Activity Name	Description	Best Practice
Set SAP Connection (SSO)	Sets SSO (*Single Sign-On*) SAP connection.	You must configure SSO on your system. To set SSO SAP connection, you must specify the values in the *MessageServer*, *InstanceNumber*, *Client*, *Language*, and *SNCName* parameter fields. You can find all the required parameters and its values in SAP Logon Application. Open SAP Logon application and then go to *Properties* of the SAP system.

Activity Name	Description	Best Practice

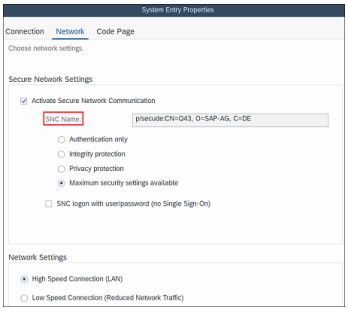


The following is the list of all parameters and its values that are required to set SSO SAP connection:

- *MessageServer*: ldciq43.wdf.sap.corp
- *InstanceNumber*: 36

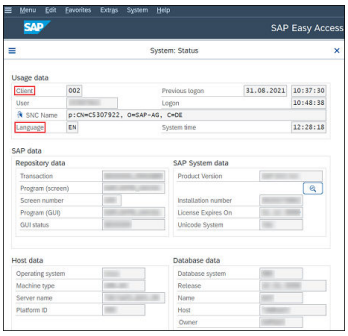


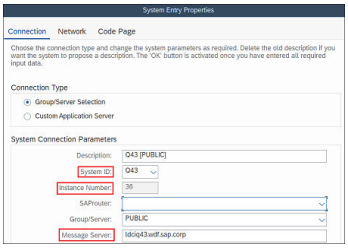
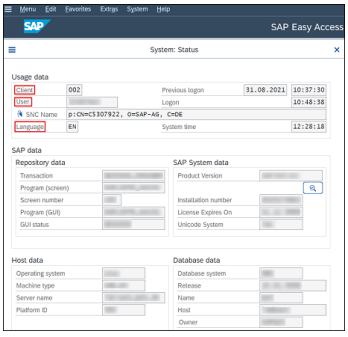
- *SNCName*: p/secude:CN=Q43, O=SAP-AG, C=DE



- *Client*: 002
- *Language*: EN

  Double click on the SAP system name and then go to *System* > *Status*.



Activity Name	Description	Best Practice
Set SAP Connection (Basic)	Sets basic SAP connection.	To set basic SAP connection, you must specify the values in the *User*, *Password*, *MessageServer*, *InstanceNumber*, *Client*, *Language*, and *SystemID* parameter fields. The following is the list of all parameters that are required to set basic SAP connection: • *MessageServer*: ldciq43.wdf.sap.corp • *InstanceNumber*: 36 • *SystemID*: Q43  • *User*: Username to connect with SAP system. • *Password*: Password to connect with SAP system. • *Client*: 002 • *Language*: EN 
Execute BAPI	Validates the parameters that are passed for BAPI and execute the BAPI. The validation of parameters checks whether: • all required parameters are passed or not. • all parameters have the correct name.	In this activity, you must specify the values in the BAPI_Name and SAP_Connection parameter fields. The following is the list of two mandatory and four optional parameters: • *BAPI_Name*: It is a mandatory parameter. The value of the parameter must be the name of the BAPI. For example:

Activity Name	Description	Best Practice
	• all parameters have the correct data type. If the validation fails, the exception *BAPI_InvalidParamException* will be thrown.	BAPI_PR_CREATE or STFC_STRUCTURE. • *SAP_Connection*: It is a mandatory parameter. The value of the parameter must be the *Response* of *Set SAP Connection (SSO/ Basic)* activity. • *Imports*: It is an optional parameter. The value of the all BAPI import parameters must be in JSON format and then the JSON data can be passed in *Imports* parameter field. • *Tables*: It is an optional parameter. The value of the all BAPI table parameters must be in JSON format and then the JSON data can be passed in *Tables* parameter field. • *Changing*: It is an optional parameter. The value of the all BAPI changing parameters must be in JSON format and then the JSON data can be passed in *Changing* parameter field. • *Outputs*: It is an optional parameter. This parameter is related to the response that the *Execute BAPI* activity returns. ○ If you do not pass any parameter name as value in the *Outputs* parameter field, the *Execute BAPI* activity returns the entire BAPI response in JSON format. ○ If you pass single or multiple parameter names as value(s) in the *Outputs* parameter field, the *Execute BAPI* activity returns the BAPI response only of these passed parameter(s) in JSON format. Multiple parameter names are passed as comma separated string.
Commit BAPI	Commits the transactions of BAPI. Commit is done on the connection object. If multiple BAPIs are executed on the same connection, the *Commit*	In this activity, you must specify the values in the *SAP_Connection* field.

Activity Name	Description	Best Practice
	BAPI activity commits all the transactions done on that connection.	The following is the parameter of this activity: • *SAP_Connection*: It is a mandatory parameter. The value of the parameter must be the *Response* of *Set SAP Connection (SSO/ Basic)* activity.
Close SAP Connection	Closes the SAP connection and dump the active connection object.	In this activity, you must specify the values in the *SAP_Connection* field. The following is the parameter of this activity: • *SAP_Connection*: It is a mandatory parameter. The value of the parameter must be the *Response* of *Set SAP Connection (SSO/ Basic)* activity.

5.1.10 SDK ARIBA Package

The ARIBA package is a collection of activities allowing you to create automations using SAP Ariba.

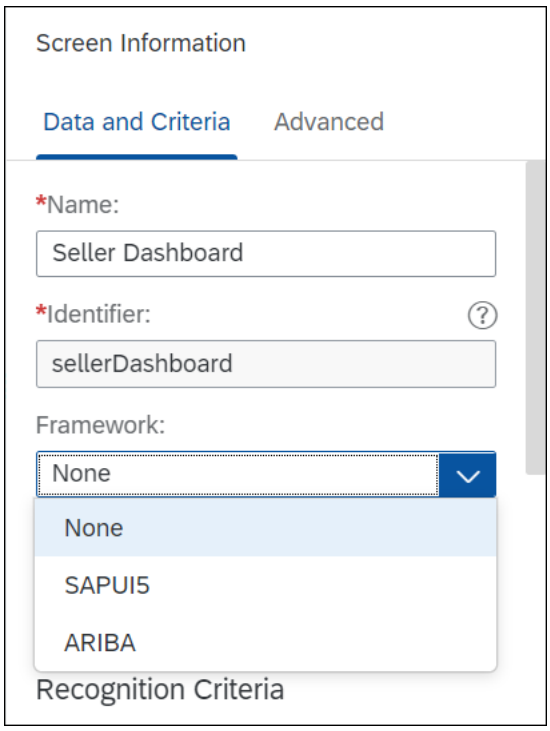


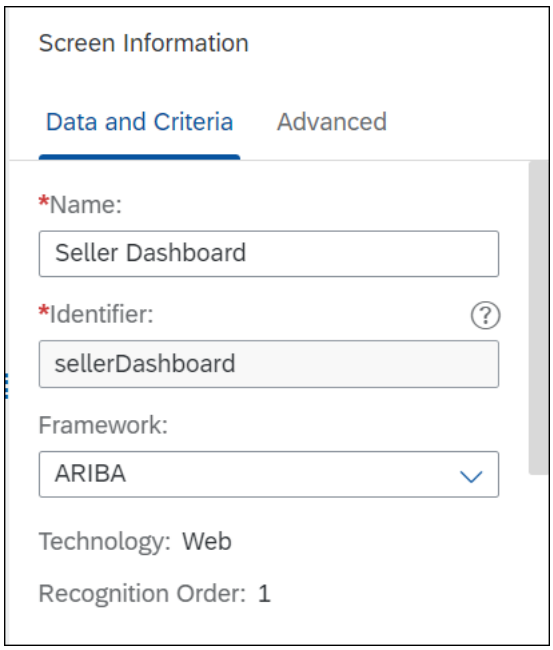
- https://help.sap.com/doc/818b6b5392924c6aa3d7fa433fcad2d0/Cloud/en-US/SDKdocumentation.html [https://help.sap.com/doc/818b6b5392924c6aa3d7fa433fcad2d0/Cloud/en-US/SDKdocumentation.html]

5.1.10.1 Add ARIBA Dependency

Adding ARIBA SDK

Capture an ARIBA application screen. In the captured application screen, you must select **ARIBA** from the *Framework* drop-down list.





After you select ARIBA as framewotk, the SDK loading pop-up is displayed and then the Dependency added successfully pop-up is displayed. You can view the ARIBA SDK in the added dependencies.



Related Topics

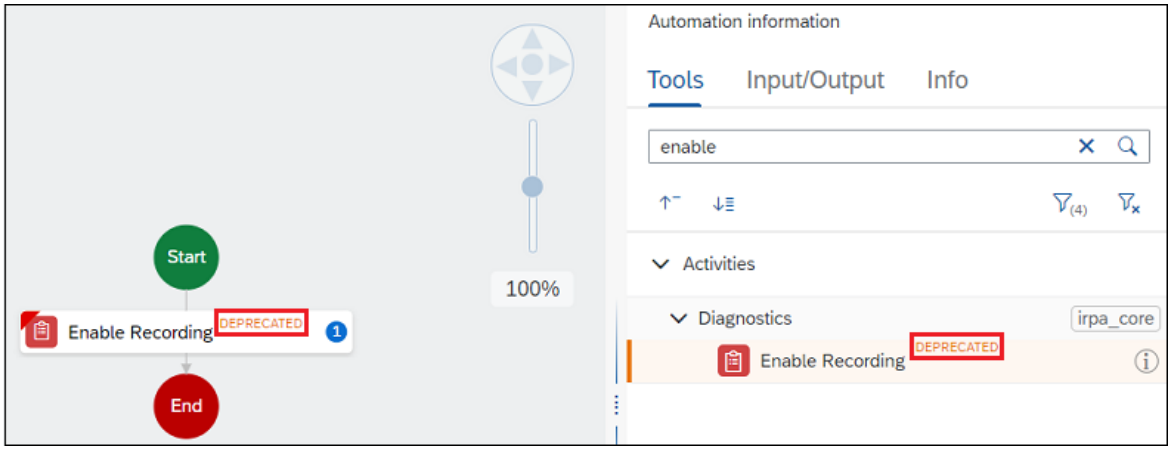
- Capture an Application [page 77]
- Declare an Application [page 101]
- Create an Automation [page 156]
- Add an Activity to an Automation [page 208]

- Add a New Dependency [page 528]

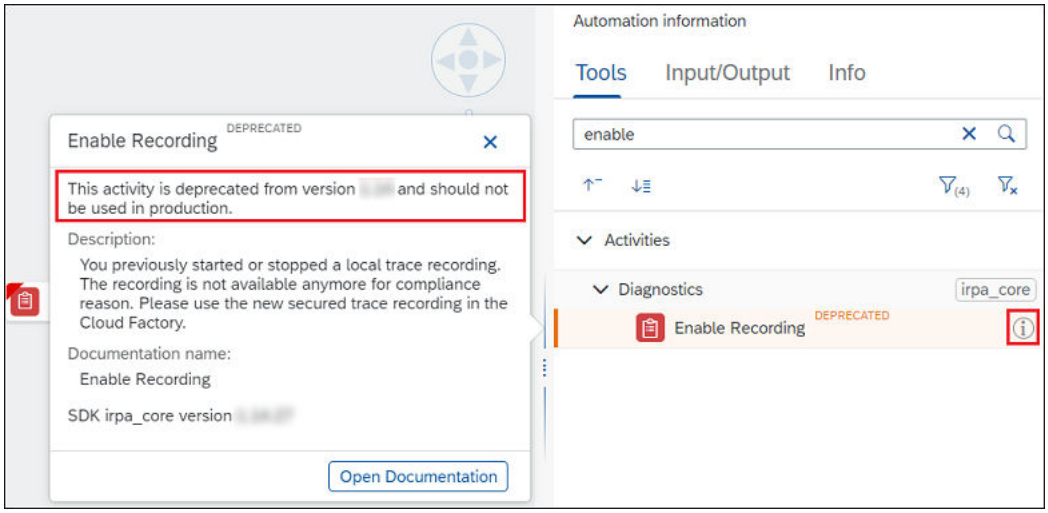
5.2 Lifecycle of SDK Activities

The SDK activities can have the following types of status:

- **Beta** – Beta status means that you can use the SDK activity for testing purposes and provide feedback on any new development set as beta. By default, the beta SDK activities are not displayed on the activities list under the *Tools* section. You must click the *Filter result* ((3)) button and select the *Include unreleased activities* checkbox to display the beta SDK activities. You must not use the SDK activity with the beta status in production. Parameters and behaviours of the SDK activity with the beta status can be changed in the final release.
- **Active** – Active status means the SDK activity can be used in the production. This flag does not appear in the Cloud Studio because it is the default status of the SDK activity.
- **Deprecated** – Deprecated status in the SDK activity indicates that the SDK activity must be avoided to use while performing an action. A warning message can also be provided to the customers to notify them of the substitute SDK activity and the length of the deprecation period. By default, the deprecated SDK activities are not displayed on the activities list under the *Tools* section. You must click the *Filter result* ((3)) button and select the *Include deprecated activities* checkbox to display the deprecated SDK activities. You must not use the SDK activity with the deprecated status in production. After the deprecation period, the SDK activity can move to the decommissioned status.
  In the following example, you can see that the *DEPRECATED* tag is displayed beside the name of the deprecated SDK activity.



Once you click the ⓘ button, the following warning message is displayed.



> ### i Note
>
> You can see the documentation of the SDK activity by clicking the *Open Documentation* button.

- **Decommissioned** – The SDK activity with decommissioned status means it is not supported anymore, and thus it won't be visible in the Cloud Studio.

6 Reuse Desktop Packages and Build Processes

Reuse Desktop Scenarios [page 476]

Processes [page 482]

Processes are built on SAP Workflow and run in the cloud. Unlike a *scenario*, a *process* could be a long-running task lasting several days, for example.

User Tasks [page 489]

A *user task* is a type of skill that you can use during the design of a process. It takes the form of a message that will be sent to the inbox of a user during the running phase of a process.

6.1 Reuse Desktop Scenarios

Scenarios created in the Desktop Studio automate repetitive tasks in desktop or web applications. Desktop scenarios are compiled and exported in the form of Desktop packages, which can be imported to the Cloud Studio to reuse their scenarios and data types.

Import Desktop Packages

You can import Desktop packages to the Cloud Studio just like you do to the Cloud Factory.

> **i** Note
>
> Unlike package imports to the Cloud Factory, when you import a Desktop package to the Cloud Studio, data type artifacts corresponding to the input/output of its scenarios are created automatically.

Before importing, you must have exported the project from the Desktop Studio as a ZIP file.

1. In the Home view or in the Project Explorer, click the *Import* ⌄ button and select *Desktop Package*.
2. Find the package you want to import, enter a name and description if necessary, and click *Import*.

The Desktop Package is now visible in the Project Explorer of the Cloud Studio.

> **→** Tip
>
> By default, the Project Explorer pane is hidden. When a user creates an artifact or imports a Desktop package, it will open. You can show or hide the Project Explorer pane at any time by clicking the ☰ button in the top left corner.

Reuse Desktop Scenarios as Skills

After importing a Desktop package, you can reuse the scenarios available in this package. For instance, you may want to reuse one or more scenarios in a process.

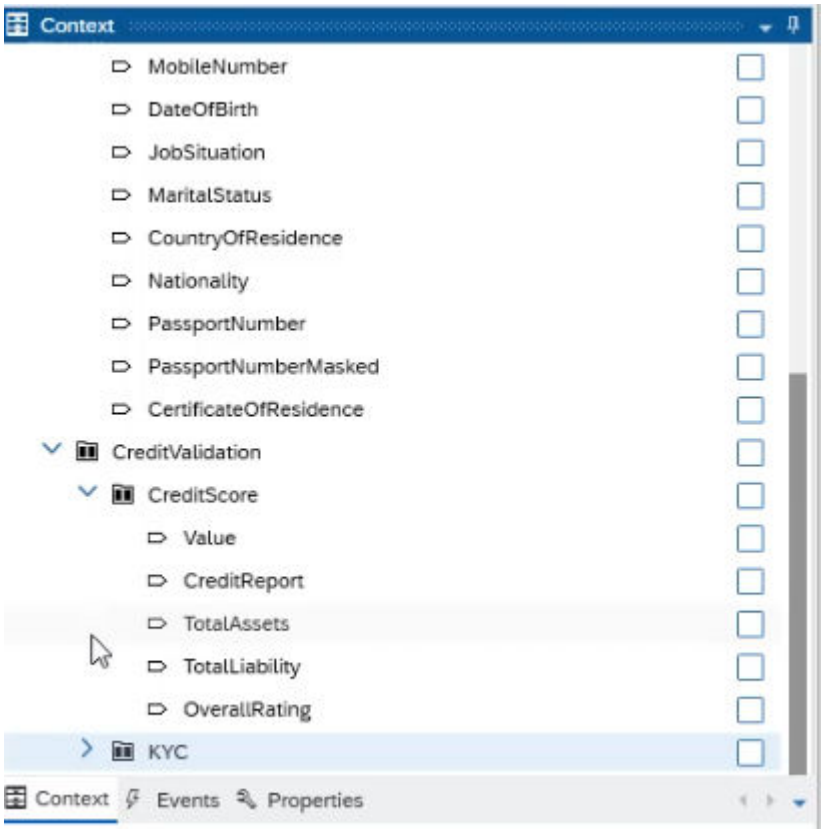
1. Go to the Project Explorer perspective and in the *Desktop Packages* dropdown list, click the name of the Desktop package you imported.
   It opens a tab in the main panel of the Cloud Studio displaying details of the package, including a list of scenarios.
2. Click the scenario you want to reuse: A *Scenario information* panel opens.
3. Click the toggle button to make the scenario usable as a skill in a process.
4. **Optional**: Check the input and output parameters of the scenarios.
   In the *Scenario information* panel, click *I/O*. You can see the input and output parameters of the scenario that were previously defined in the Desktop Studio. For more information, see Data Types created during the import of a scenario [page 477].
5. Save your changes.

The selected scenarios are now available for use as skills when building a process.

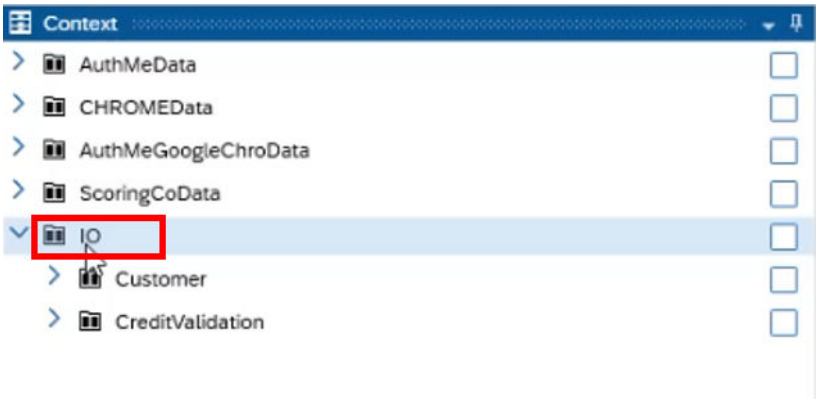
6.1.1 Data Types Created When a Scenario is Imported

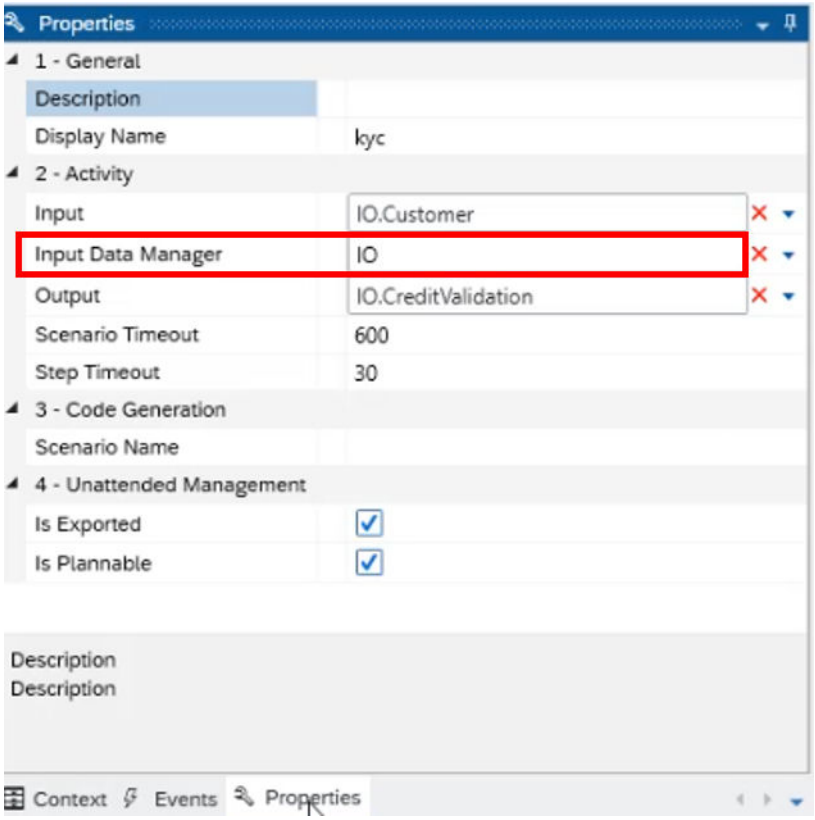
Some data types can be automatically created when scenarios are imported. To make this possible, scenario inputs/outputs must be defined when the scenario is designed in the Desktop Studio.

As a prerequisite, you must have defined variables in the context when building a scenario using the Workflow Designer of the Desktop Studio:

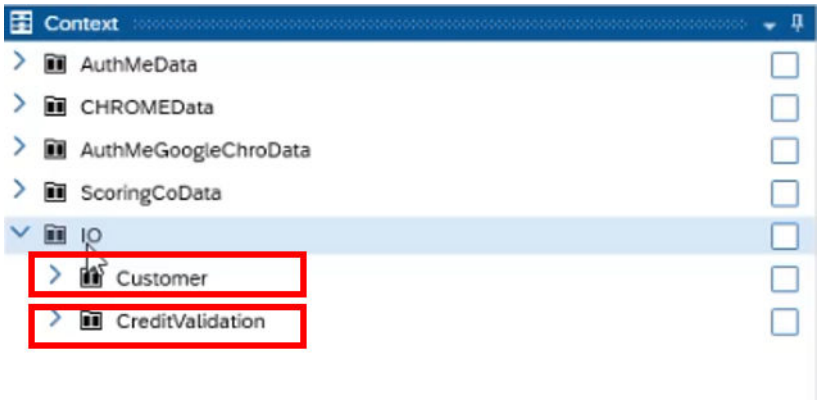


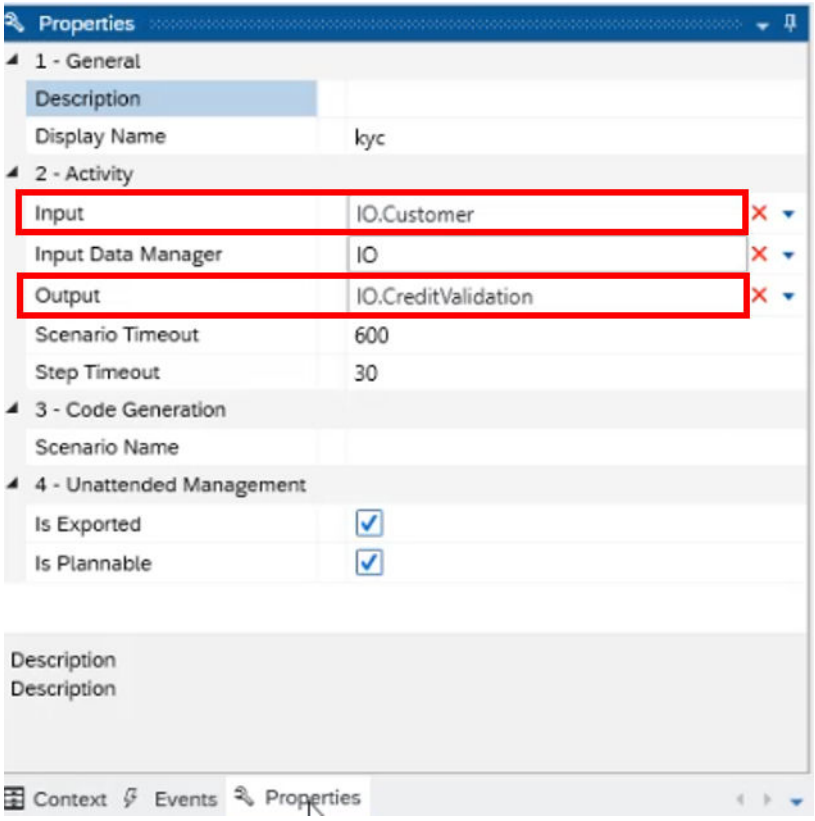
1. Go to the *Properties* panel and in the *Activity* area, enter a root node of variables in the *Input Data Manager*.



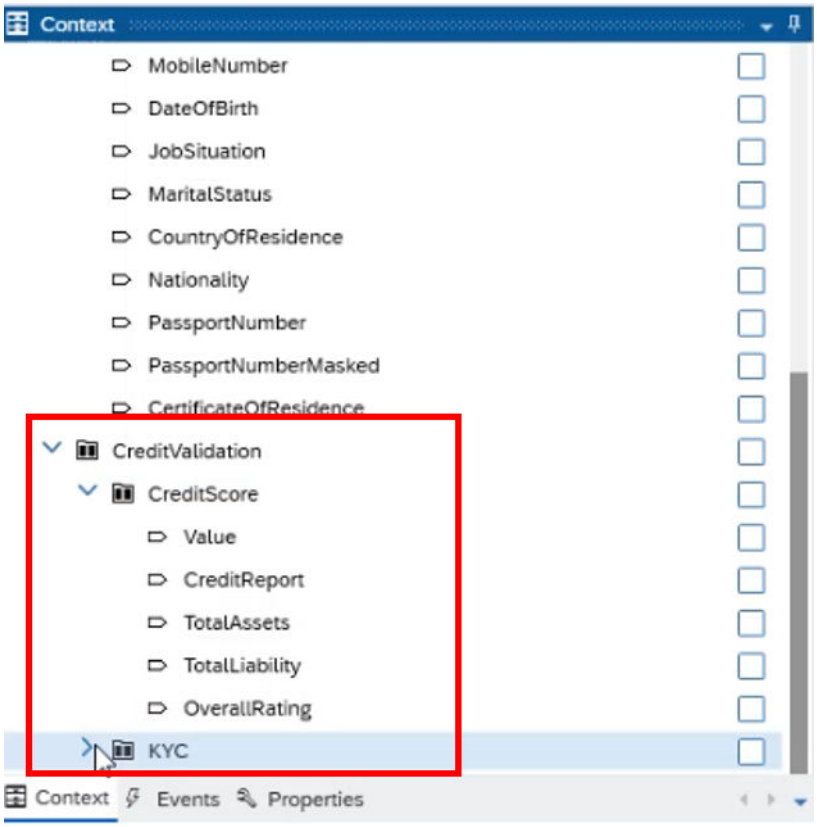


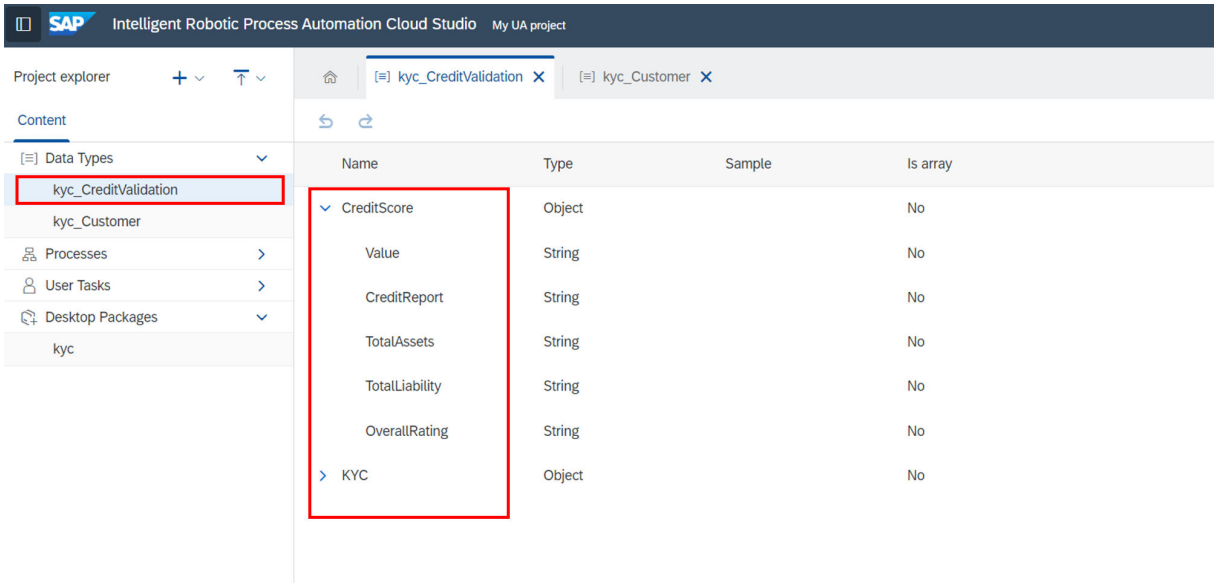
2. Select child nodes of variables as *Input* and *Output*.




Properties
1 - General
Description
Display Name
kyc
2 - Activity
Input
IO.Customer
Input Data Manager
IO
Output
IO.CreditValidation
Scenario Timeout
600
Step Timeout
30
3 - Code Generation
Scenario Name
4 - Unattended Management
Is Exported
Is Plannable
Description
Description
Context
Events
Properties

The nodes and children are mapped as data types and data type fields when the scenario is imported to the Cloud Studio:





For more information, see the topic on Managing Data in the Desktop Studio Developer Guide, and Reuse Desktop Scenarios [page 476].

6.2 Processes

Processes are built on SAP Workflow and run in the cloud. Unlike a *scenario*, a *process* could be a long-running task lasting several days, for example.

Processes can involve multiple systems and orchestrate the execution of coarse-grained tasks (*skills*). A process can also feature interactions between multiple personas.

To get more information about SAP Workflow, check the SAP Workflow official documentation.

> **i Note**
>
> Processes are not available if you use SAP Intelligent RPA in the AWS EU11 data center.

Skills

The tasks forming a process are called *skills*, and can run independently of each other. You can use three types of skills to design a process in the Cloud Studio:

- *Scenarios*
  Designed with the Desktop Studio, a *scenario*, or *Desktop scenario*, automates repetitive tasks on an application or a web application (such as S/4 or SAP GUI). Scenarios are short, as it is best practice to design scenarios lasting about 10 minutes. Scenarios can orchestrate gradual tasks called activities such as web page clicks (note that those task should run in the same session altogether). Another best practice is to have one scenario running with one persona. A scenario runs on-premises in a Desktop Agent.
  With the Cloud Studio, you can import scenarios and use them as skills to build processes. For more information, see Reuse Desktop Scenarios [page 476].
- *User tasks*
  A *user task* is a type of skill that you can use during the design of a process. It takes the form of a message that will be sent to the inbox of a user during the running phase of a process. When you create a user task, you define in a message a specific action that needs to be done by the user. See also Provide Links to User Inboxes for User Tasks [page 488].
  Unlike scenarios, user tasks are not imported but created in the Cloud Studio. For more information, see User Tasks [page 489].
- *Processes*
  With the Cloud Studio, you can use a process as a skill to build another process. In short, a process can contain one or more other processes.

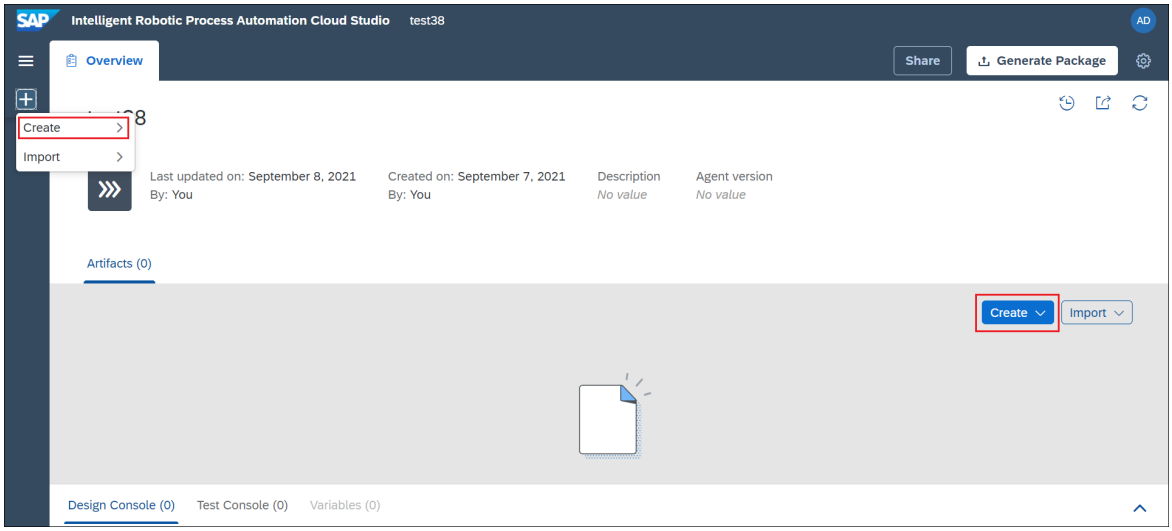
Input/Output Parameters

When designing a process, you can use input/output parameters to pass data from one task (skill) to the other.

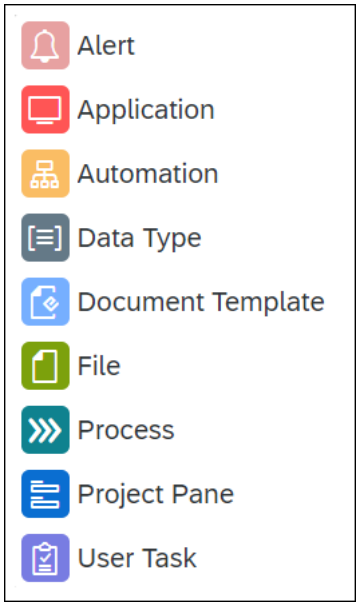
6.2.1 Create a Process

Perform the following steps to build a process.

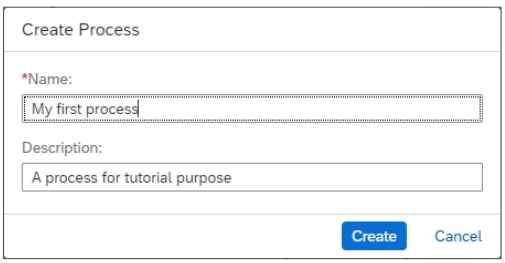
1. In the Project Explorer, click + and then click *Create* in the left panel or click *Create* in the Project Explorer main panel.



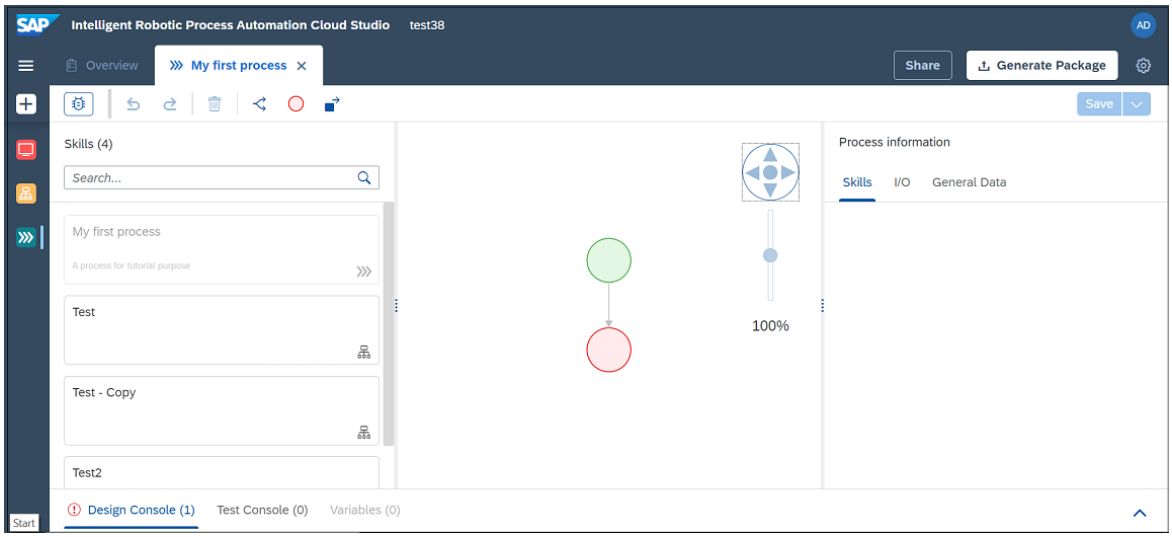
The artifact menu is displayed.



2. In the artifact menu click *Process*. The *Create process* pop-up screen is displayed.



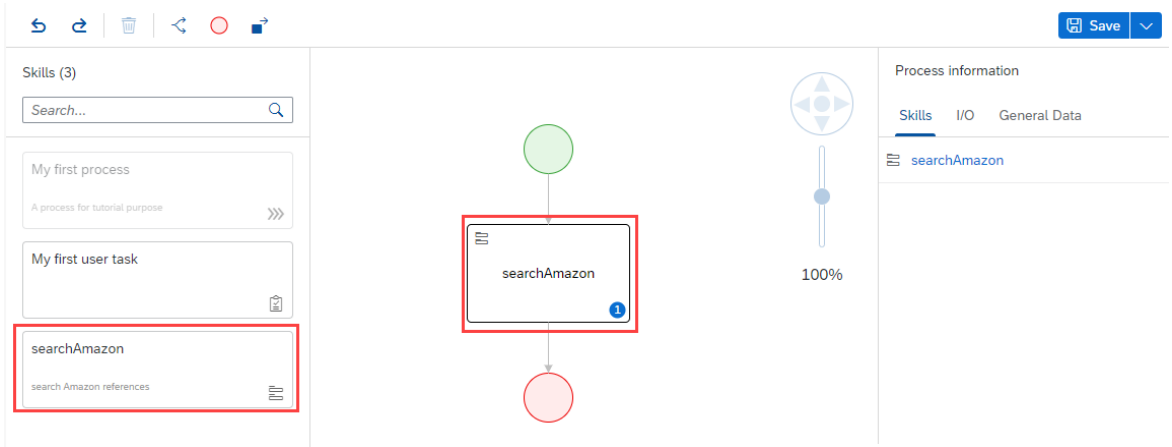
3. Enter a name and description.
4. Click *Create*. A new tab opens in the main panel of the Cloud Studio.



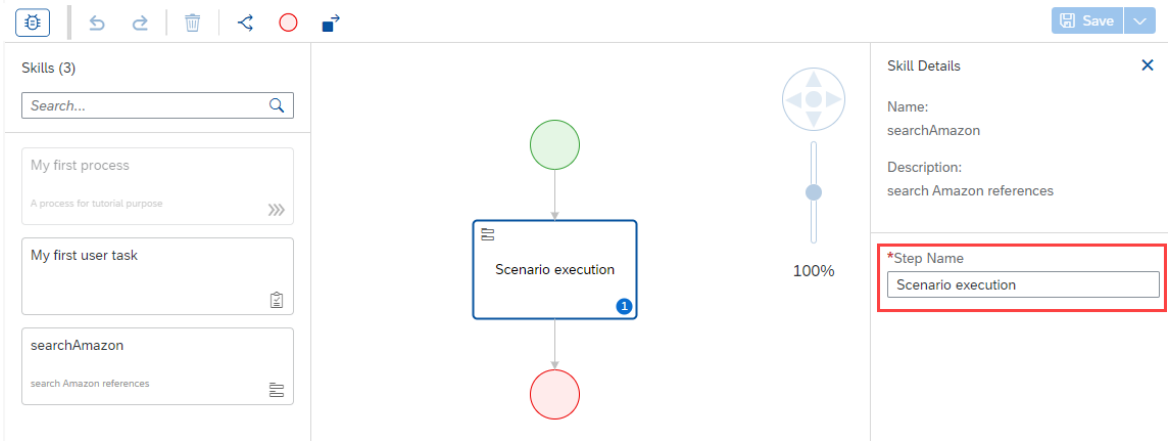
Insert a Scenario

1. Drag a scenario from the main panel and drop it in the flow of the process.

> **i Note**
>
> You can find the scenario(s) you made usable as a skill in the list of skills. For more information, see Reuse Desktop Scenarios [page 476].



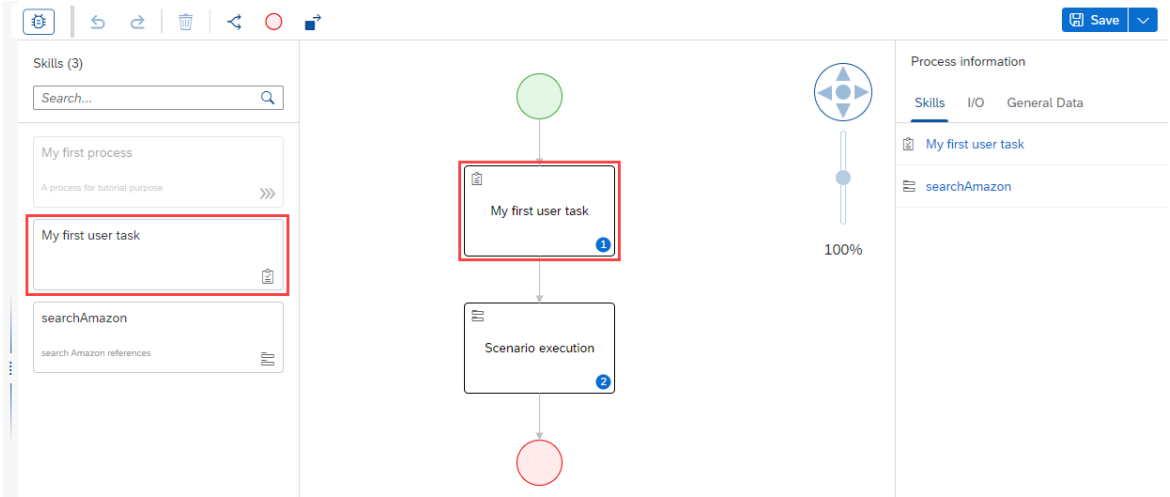
2. Select the scenario and change the name to “Scenario execution” in the *Skill Details* panel.



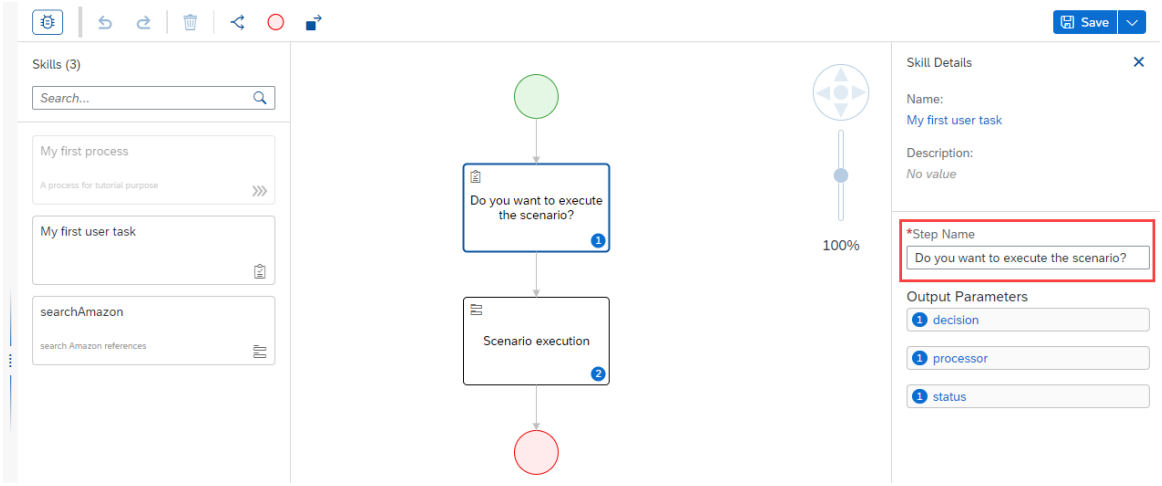
3. Click the *Save* button.

Insert a User Task

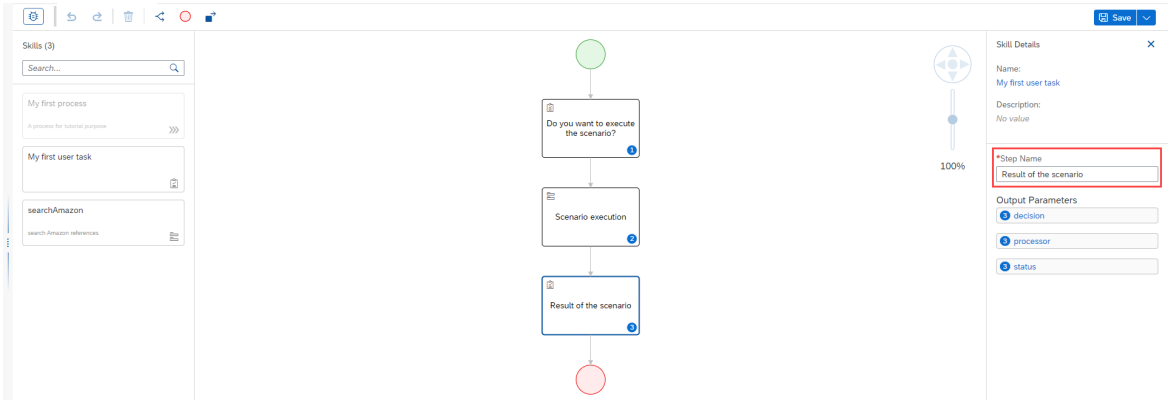
1. Drag the user task you previously created from the main panel and drop it in the flow of the process, before the scenario.



2. Select the user task and in the *Skill Details* panel, change the name to "Do you want to execute the scenario?".



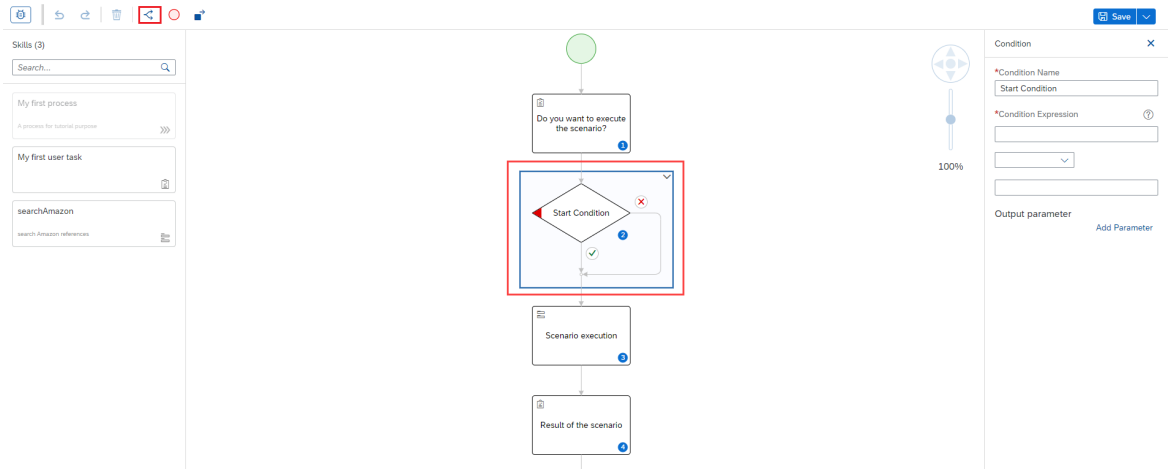
3. Drag the user task you previously created from the main panel and drop it in the flow of the process, after the scenario.
4. Select the user task and change the name to “Result of the scenario”.



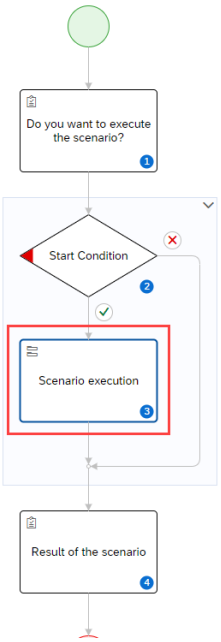
5. Click the *Save* button.

Add a Condition

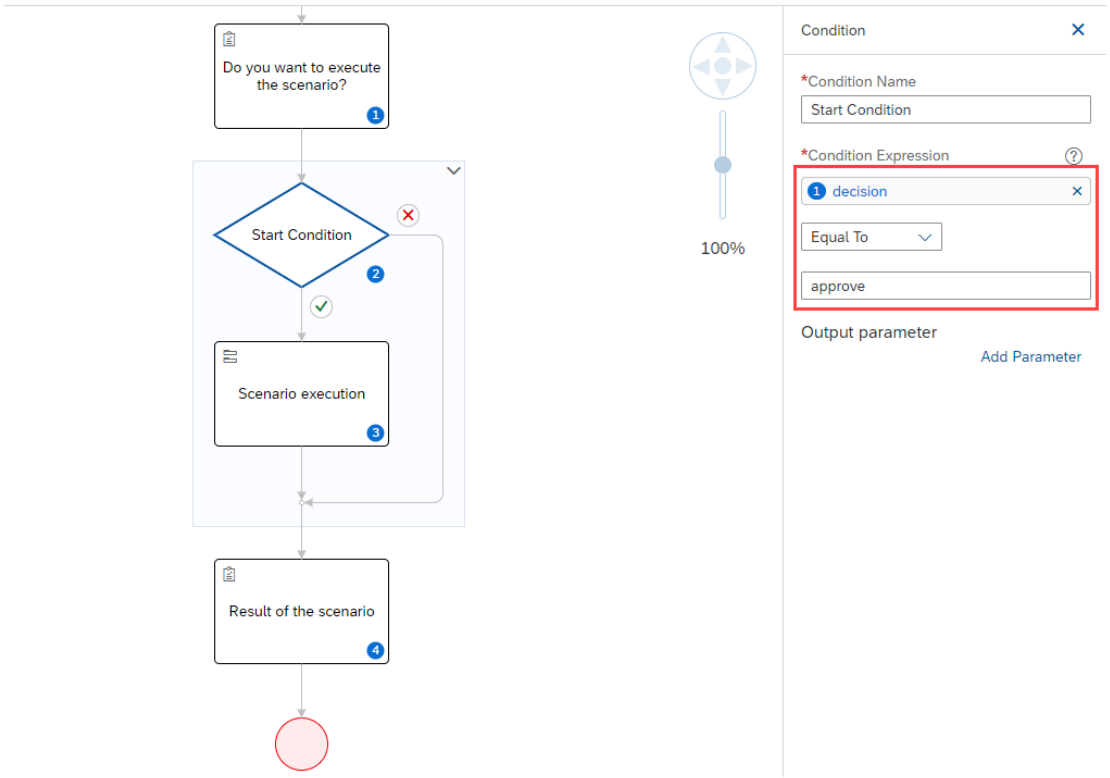
1. Click the button to drag and drop a condition in the flow of the process.



2. Drag and drop "Scenario execution" inside the condition panel.



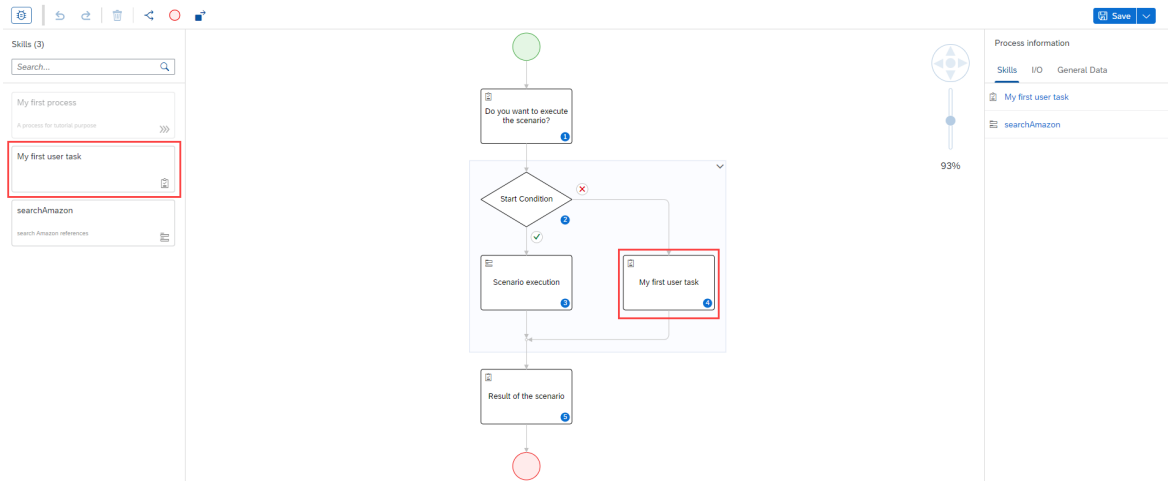
3. Define the *Condition Expression*: set the decision equal to “approve”.



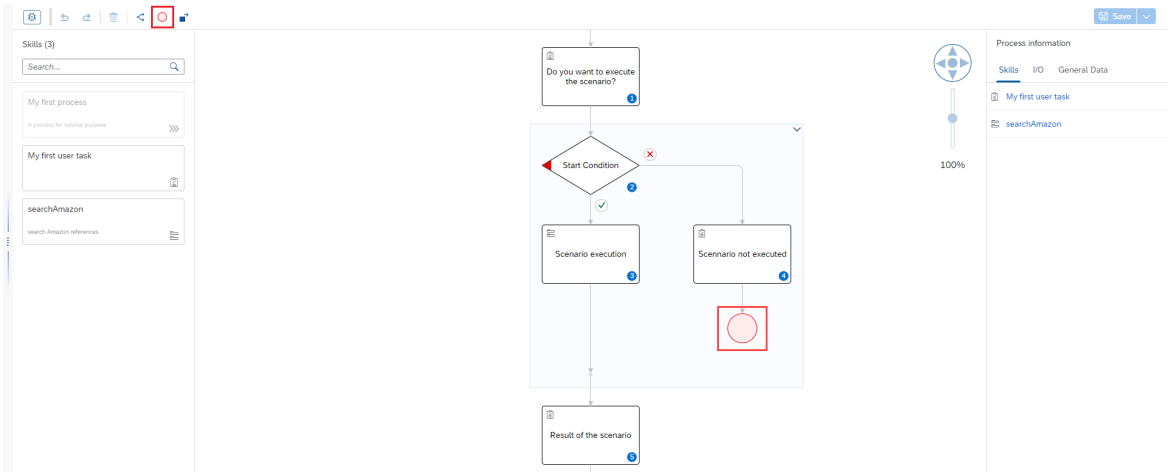
4. Click the *Save* button.

Insert a User Task as a Result of a Condition

1. Drag the user task you previously created and drop it in the condition panel, as the other possible result of the condition.



2. Select the user task and in the *Skill Details* panel, change the name to “Scenario not executed”.
3. Drag and drop an "end" component in the condition panel, after the user task "Scenario not executed".



4. Click the *Save* button.

6.2.2 Provide Links to User Inboxes for User Tasks

If your project includes user tasks, you need to provide the relevant users with links to their inboxes so that they can read their messages. The link is derived from the base URL of the user's SAP Intelligent Robotic Process Automation Factory tenant.

> ### Example
>
> If the URL of the user's Factory tenant is
>
> `https://<tenant subdomain>.app.irpa.cfapps.<region>.hana.ondemand.com/#/home`

then the base URL is

`https://<tenant subdomain>.app.irpa.cfapps.<region>.hana.ondemand.com`

Remove "`app`" from the URL and replace it with "`flp`".

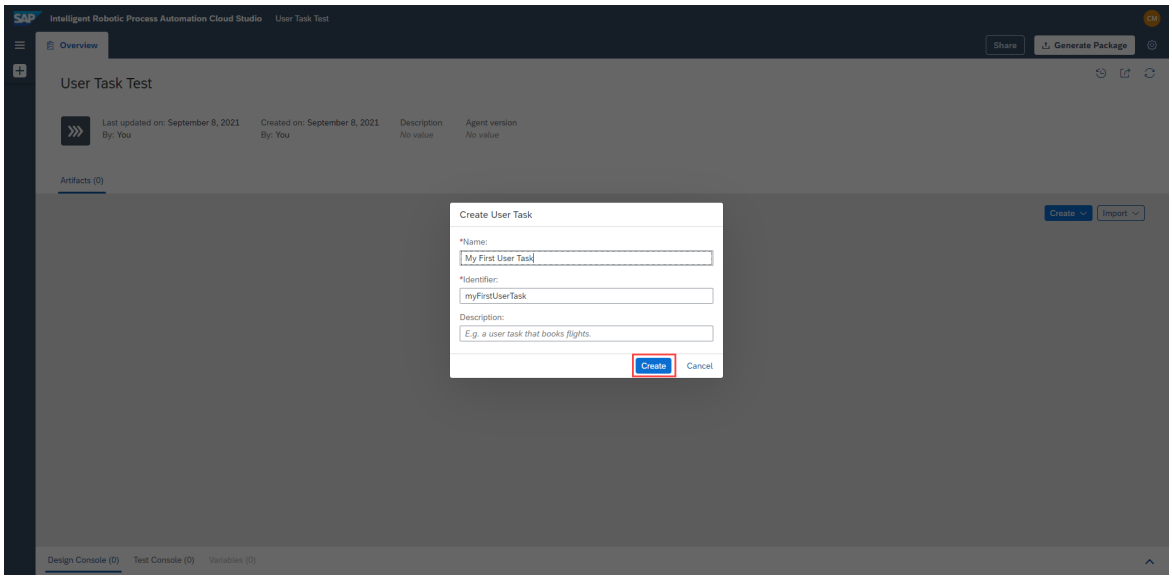
6.3 User Tasks

A *user task* is a type of skill that you can use during the design of a process. It takes the form of a message that will be sent to the inbox of a user during the running phase of a process.

When you create a user task, you define in a message a specific action that needs to be done by the user.

1. In the Home view or in the Project Explorer, click the + *Create* button, select *Create* and then, select *User Task*.
2. Enter a name and description, and click *Create*.

> **i Note**
>
> When you create a user task in a project containing other artifacts, the user task now has an identifier. The identifier is the name of the user task that can be referenced in custom scripting. By default, the identifier is the same as the name. An identifier is mandatory for every artifact in the Cloud Studio, so you need to manually enter an identifier if it's not already pre-filled.



3. A new tab opens in the main panel of the Cloud Studio. You can now define the settings and add the different elements that will form your user task.

> **i Note**
>
> To read the mails in your SAP Workflow inbox, two roles are necessary:
>
> - `WorkflowParticipant`, an SAP BTP (Business Technology Platform) Workflow role

- `IRPAParticipant`, an SAP Intelligent RPA role

For more information about the `WorkflowParticipant` role, see SAP BTP (Business Technology Platform) Workflow documentation. For more information on the `IRPAParticipant` role, see User Roles in SAP Intelligent Robotic Process Automation.

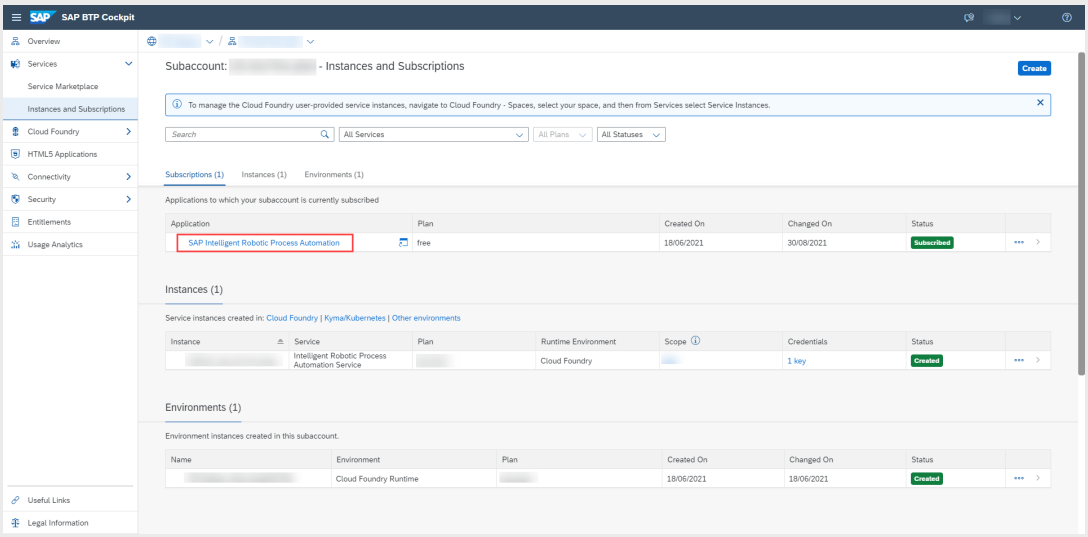
i Note

To read the mails sent by a user task, you need to access a dedicated inbox. Proceed as follows to access that inbox:

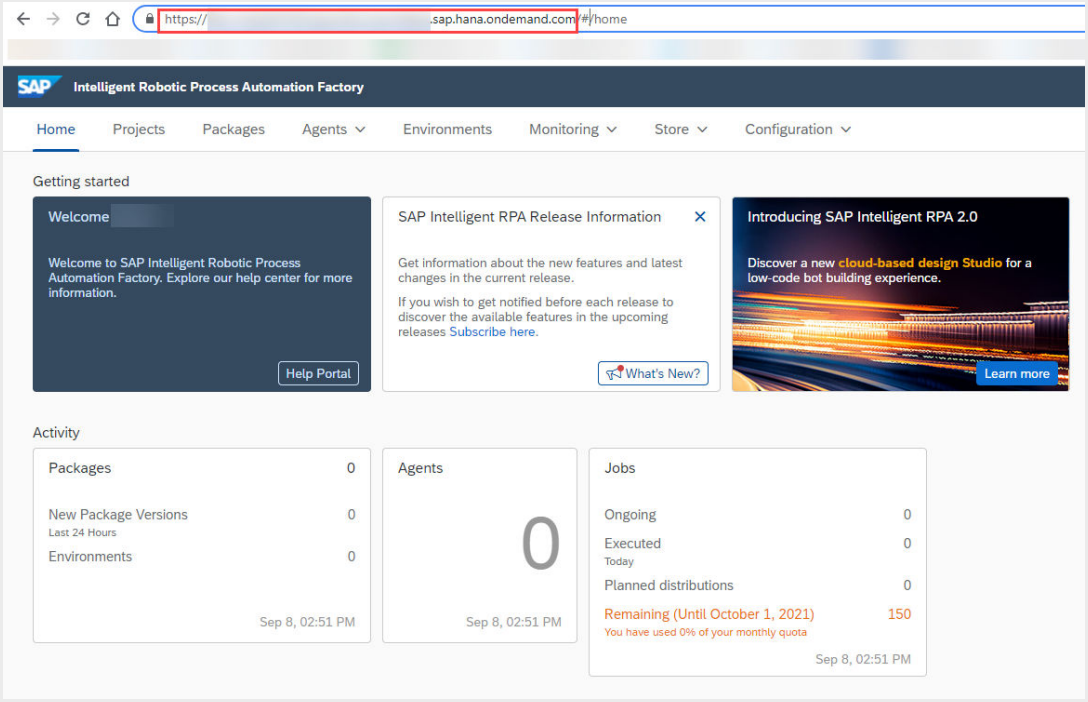
1. Copy the base URL of your SAP Intelligent RPA Factory tenant.

→ Tip

To find the SAP Intelligent RPA Factory tenant URL, click *SAP Intelligent Robotic Process Automation* in your SAP Intelligent RPA Factory account on the SAP BTP Cockpit:



Then copy the base URL and paste it into the browser window.



Example

If the URL of your SAP Intelligent Robotic Process Automation Factory tenant is

```
https://<tenant subdomain>.app.irpa.cfapps.<region>.hana.ondemand.com/#/home
```

then the base URL is

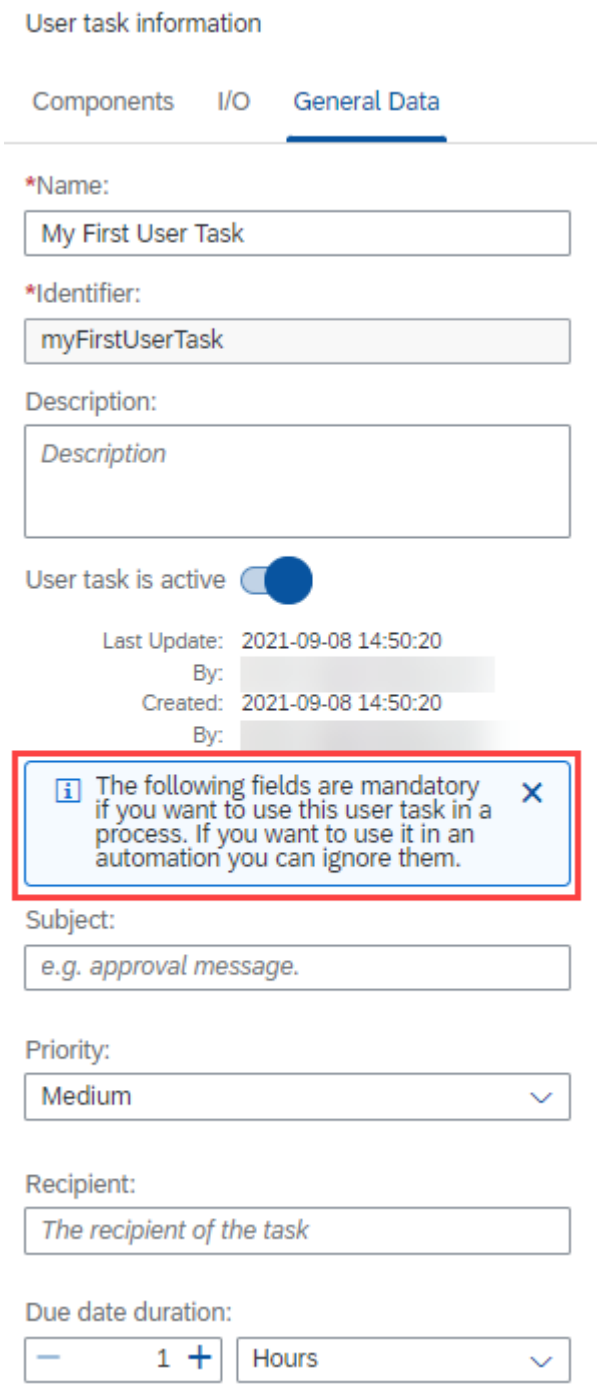
```
https://<tenant subdomain>.app.irpa.cfapps.<region>.hana.ondemand.com
```

2. Remove "`app`" from the URL and replace it with "`flp`".

6.3.1 Define the Settings of the User Task

To create a user task that you will use in a process, you must define mandatory settings in the *General Data* tab:

1. Enter the *Subject* of the user task: it is the subject of the mail the recipient will receive in the inbox.
2. Set the *Priority*.
3. Enter the *Recipient*: enter the email address of the recipient of the user task.
4. Select the *Due date duration*.
5. Save your changes.



> **Note**
>
> If you use the user task in an automation, these fields are optional.

6.3.2 Insert Components and Define Parameters to Build a User Task

You can add several types of components in a user task:

- *Input*
- *Label*
- *Check Box*
- *Title*
- *Link*
- *Combo Box*
- *Multi-combo Box*
- *Date picker*
- *Time picker*
- *Date/Time picker*

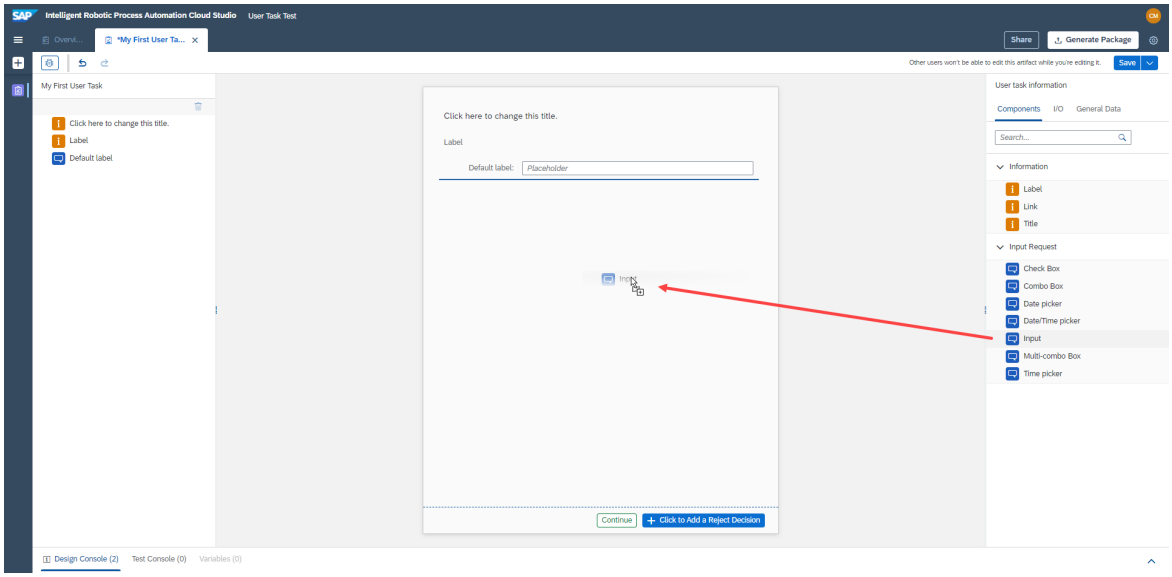
You can only place one component per line. If you select a component, you can define its properties in the *Component Properties* panel.

> ⚠ Caution
>
> The *Input* component differs from the input and output parameters that you bind to different elements in the Cloud Studio.

Insert an *Input* Component

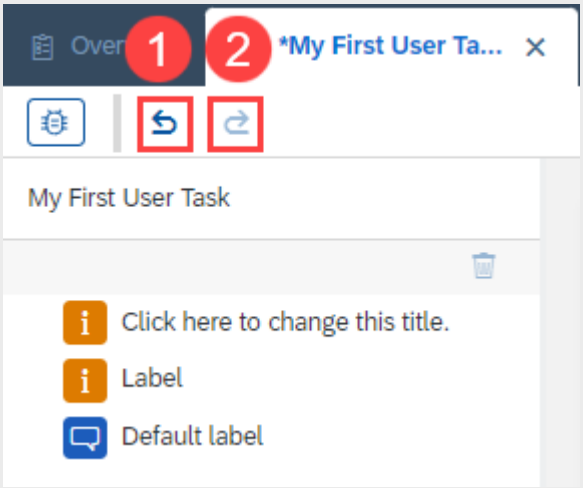
Insert an *Input* component as a text field in which the user can enter a value. To add an *Input* component to your user task:

1. Click or drag and drop the *Input* element from the *Components* panel to the main panel.

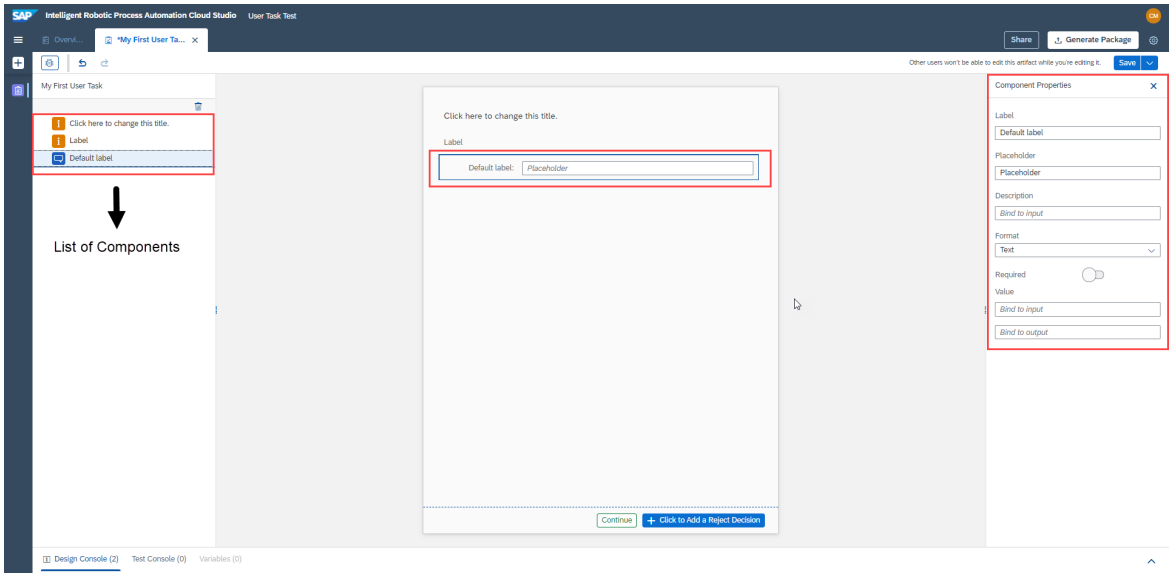


→ Tip

- To remove a component from a user task, click it and then click 🗑.
- To search components name, use search bar under *Components* tab.
- To reorder the list of components on the left side, rearrange the order of the components in the user task panel. You can select a component and move it up or down in the canvas.
- To undo or redo any activity, click *Undo* (1) or *Redo* (2) button.



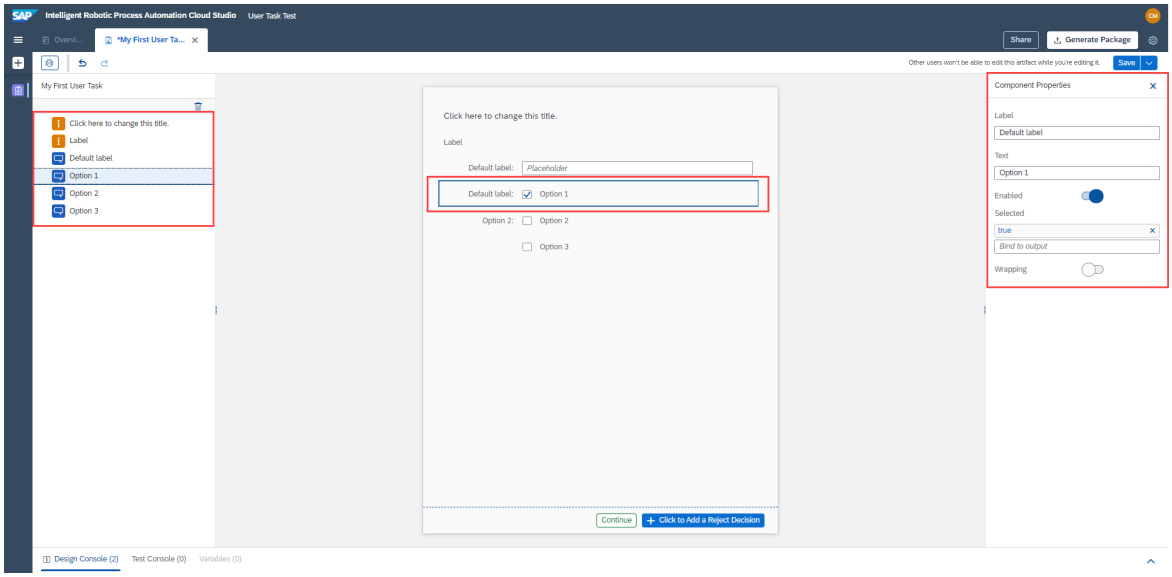
2. Click the *Input* component you added to define its properties in the *Component Properties* panel. You can:
   - Change the *Label* text, which is displayed next to the *Input* component.
   - Insert a *Placeholder* text if you want to help users understand what they need to manually enter in this field.
   - Add a description. This description appears on the right side of the *Input* component.
   - Enter two default values that will be displayed to the user. You can bind the first value to an input parameter. You can also bind the second value to an output parameter (for instance the decision output), that will be reused in the process containing the user task. The values are listed in the *I/O* tab.
3. Save your changes.



Insert a *Check Box* Component

To add a *Check Box* component to your user task:

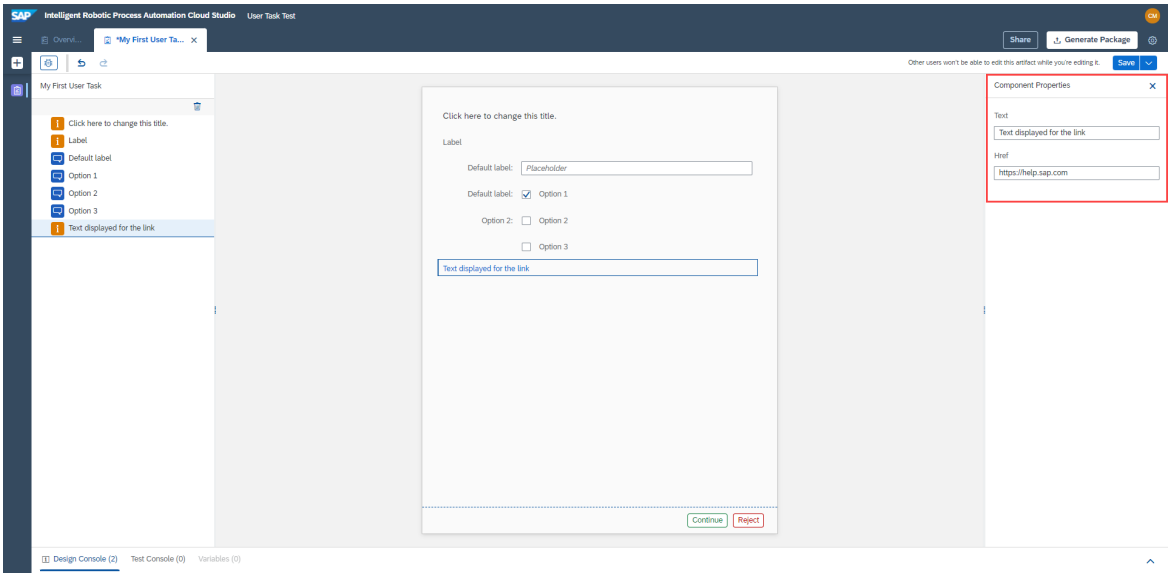
1. Click or drag and drop the *Check Box* component from the *Components* panel to the main panel.
2. Click the *Check Box* element you added to define its properties in the *Component Properties* panel. You can:
   - Change or remove the *Label* text, which is displayed next to the *Check Box*.
   - Add a text. This text is displayed on the right side of the *Check Box*.
   - Enable or disable the *Check Box* with the *Enabled* option.
   - Enter two variables in the *Selected* section. You can bind the first variable to an input parameter. The variable appears in the *I/O* tab as a *Boolean*. If you select *true* in the *Bind to input* field, the *Check Box* will be checked by default. You can also bind the second variable to an output parameter.
   - Enable the *Wrapping* option if the text you added is too long. The text will appear as a paragraph.
3. Save your changes.



Insert a *Link* Component

To add a *Link* component to your user task:

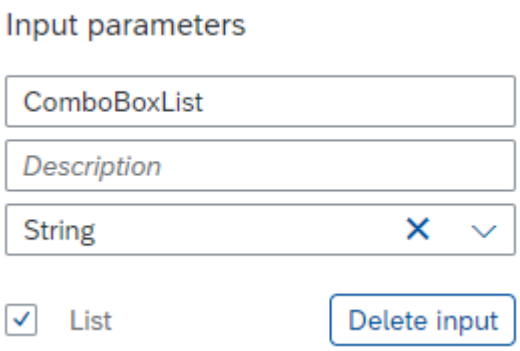
1. Click or drag and drop the *Link* component from the *Components* panel to the main panel.
2. Click the *Link* element you added to define its properties in the *Component Properties* panel. You can:
   - Choose the text of the link that will be displayed by filling in the *Text* field.
   - Enter the address of the website to which the link will lead in the *Href* field.
3. Save your changes.



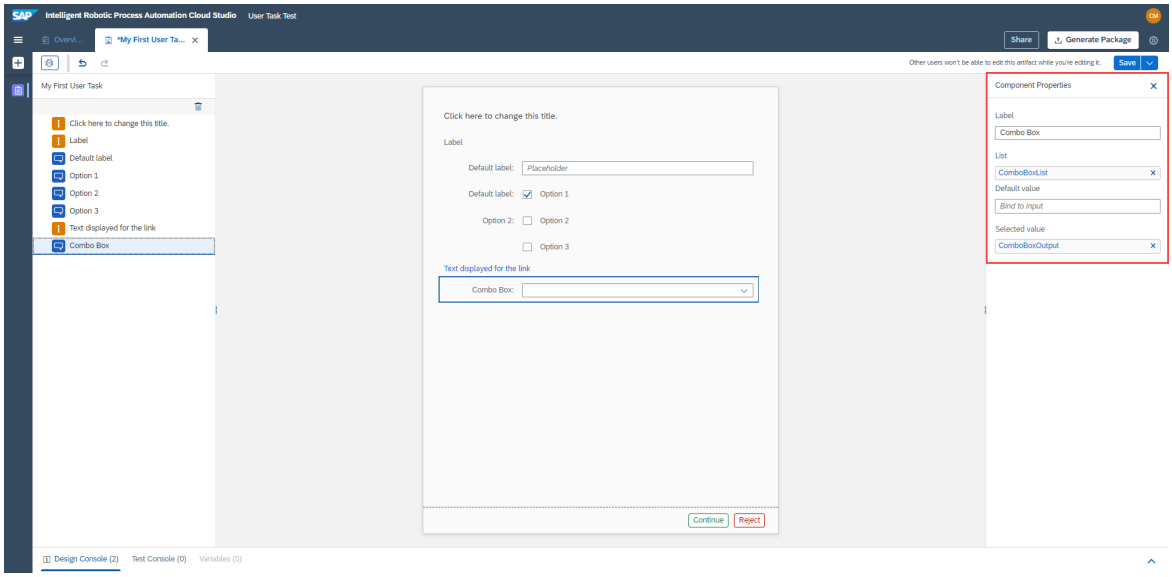
Insert a *Combo Box* Component

The *Combo Box* component allows you to select only one element from a drop-down list in a user task. To add a *Combo Box* component to your user task:

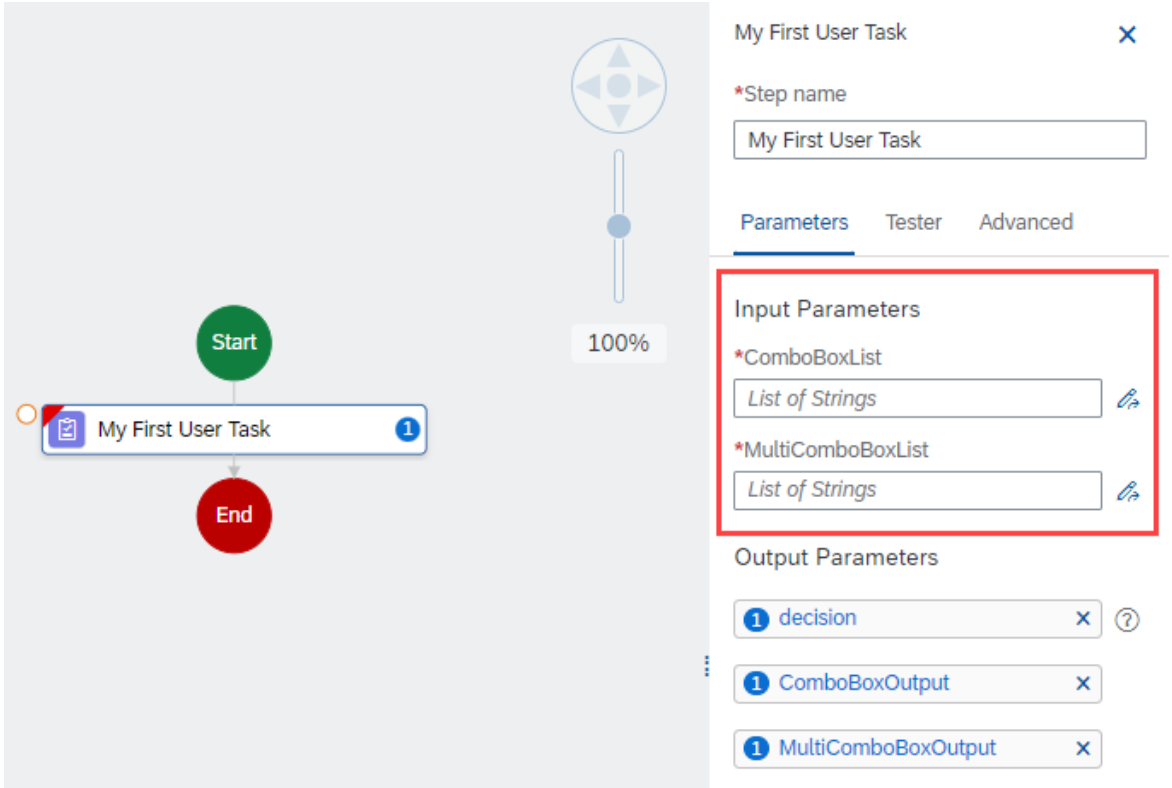
1. Click or drag and drop the *Combo Box* component from the *Components* panel to the main panel.
2. Click the *Combo Box* component you added to define its properties in the *Component Properties* panel. You can:
   - Change or remove the *Label* text, which is displayed next to the *Combo Box*.
   - Bind an optional constant value by filling in the *Default value* field. This value will be set by default but the user who opens the user task can modify it.
   - Bind an optional value to an input by filling in the *List* field.
     The variable that is created appears in the *I/O* tab and is checked as a list.

   - Bind a value to an output by filling in the *Selected value* field.
3. Save your changes.



4. Drag and drop your user task into your automation.
5. On the *Input Parameters* panel, set the list of strings as a variable or with the *Expression Editor*.



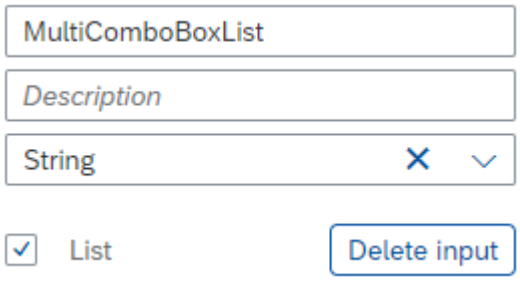
6. Test your automation.

> **i Note**
>
> Before testing your automation, you must define the values of the drop-down list.

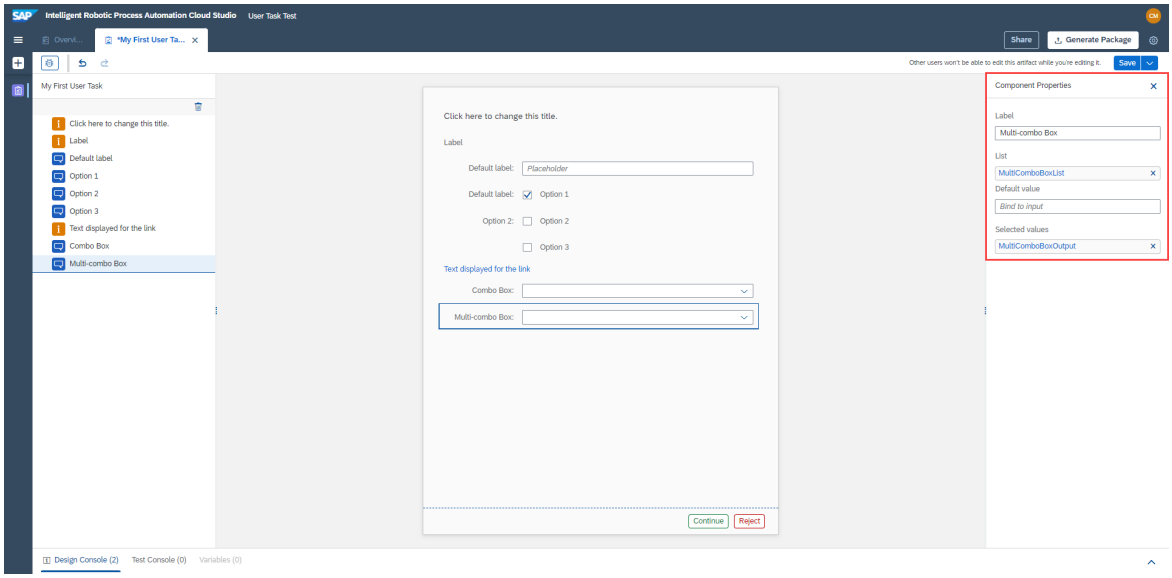
Insert a *Multi-combo Box* Component

The *Multi-combo Box* component allows you to select several elements from a drop-down list in a user task. To add a *Multi-combo Box* component to your user task:

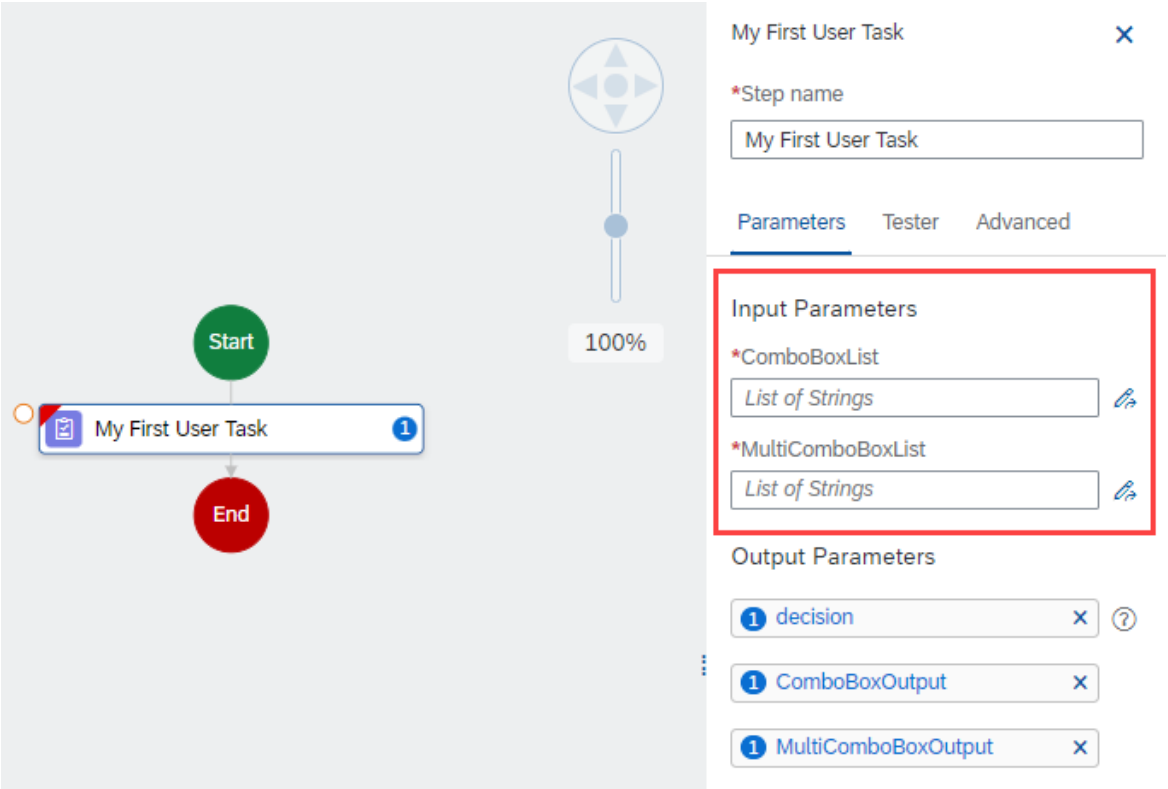
1. Click or drag and drop the *Multi-combo Box* component from the *Components* panel to the main panel.
2. Click the *Multi-combo Box* component you added to define its properties in the *Component Properties* panel. You can:
   - Change or remove the *Label* text, which is displayed next to the *Multi-combo Box*.
   - Bind an optional constant value by filling in the *Default value* field. This value will be set by default but the user who opens the user task can modify it.
   - Bind an optional value to an input by filling in the *List* field.
     The variable that is created appears in the *I/O* tab and is checked as a list.

   - Bind a value to an output by filling in the *Selected values* field.
3. Save your changes.



4. Drag and drop your user task into your automation.
5. On the *Input Parameters* panel, set the list of strings as a variable or with the *Expression Editor*.



6. Test your automation.

> **i Note**
>
> Before testing your automation, you must define the values of the drop-down list.

Insert a *Date picker* Component

The *Date picker* component allows you to select a date in a user task. To add a *Date picker* component to your user task:

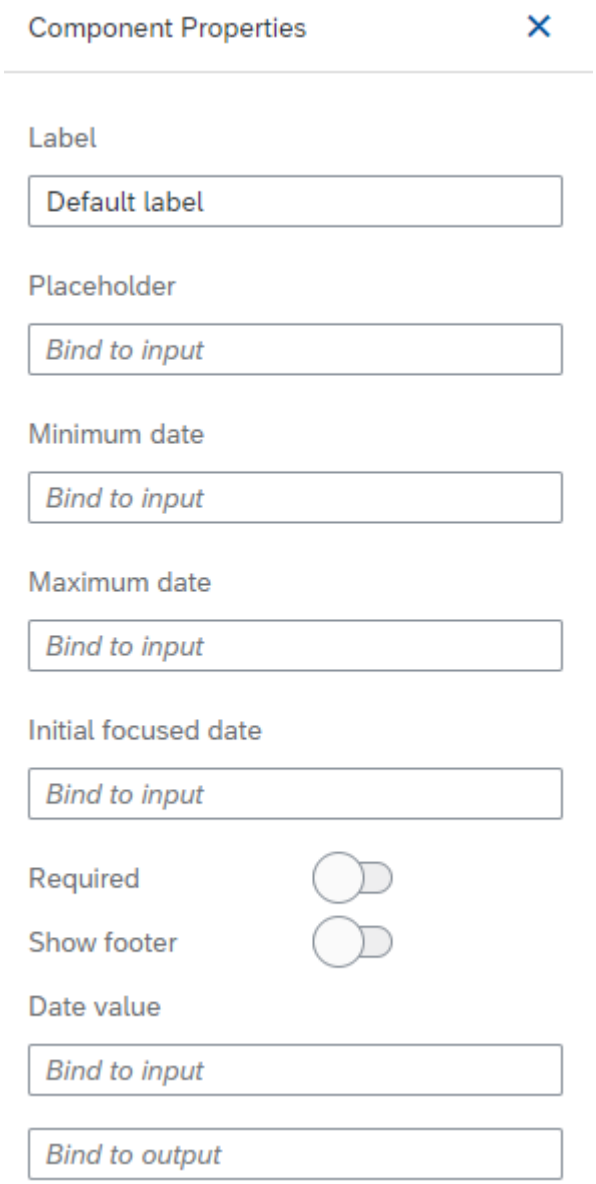
1. Click or drag and drop the *Date picker* component from the *Components* panel to the main panel.
2. Click the *Date picker* component you added to define its properties in the *Component Properties* panel. You can:
   - Change or remove the *Label* text, which is displayed next to the *Date picker*.
   - Insert a *Placeholder* text if you want to help users understand what they need to manually enter in this field.
   - Select the *Minimum Date* and the *Maximum Date* that the user will be able to select.
   - Select an *Initial Focused Date*.

     > **i Note**
     >
     > The *Initial Focused Date* is the date that will be displayed when your user opens the *Date picker*. If you do not select one, the date that will appear when the user opens the *Date picker* will be the current date.

- Enable the *Required* toggle button if you want your user to see that setting a date is required.
- Enable the *Show footer* toggle button.

> **i Note**
>
> If you enable the footer, you will have to click the *OK* button when you select a date in the *Date picker*. If you do not enable the footer you can just select the date and it will be set.

- Enter two variables in the *Date value* section. You can bind the first value to an input parameter and the second one to an output parameter.



3. Save your changes.
4. Drag and drop your user task into your automation.
5. Save and test your automation.
6. A popup appears. Click the 📅 icon to select your date.

> **Note**
>
> A limitation has been set on user tasks depending on the SDK version you use. You cannot use the *Date picker* component if you have a tenant with an old SDK version. An error message will be displayed in the *Test console* panel.

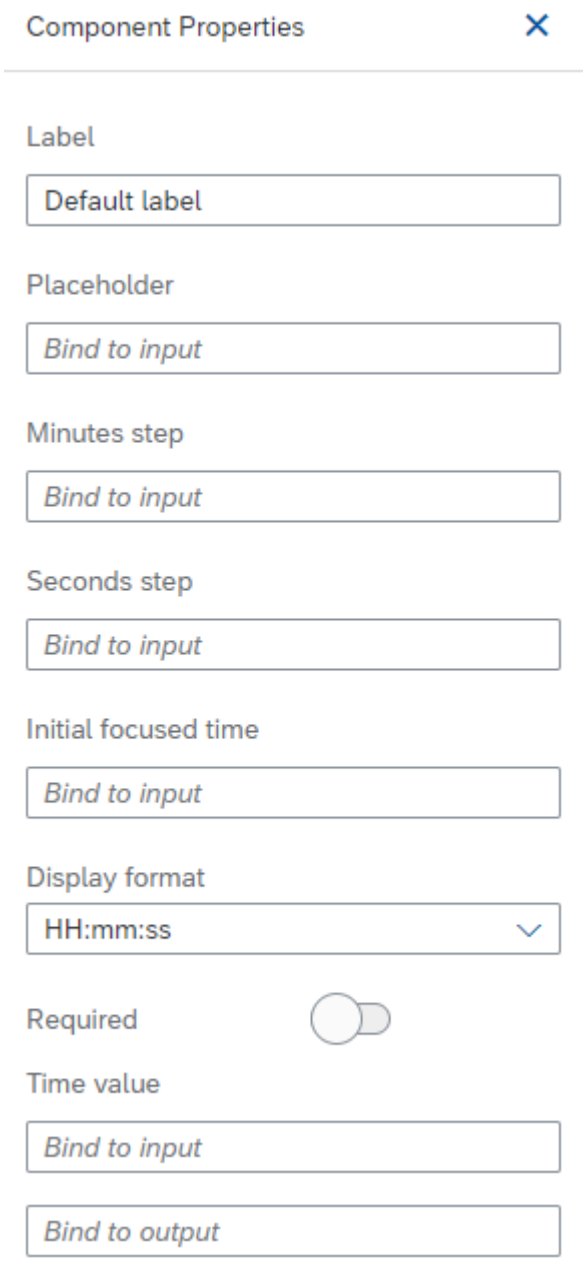
Insert a *Time picker* Component

The *Time picker* component allows you to select a time in a user task. To add a *Time picker* component to your user task:

1. Click or drag and drop the *Time picker* component from the *Components* panel to the main panel.
2. Click the *Time picker* component you added to define its properties in the *Component Properties* panel. You can:
   - Change or remove the *Label* text, which is displayed next to the *Time picker*.
   - Insert a *Placeholder* text if you want to help users understand what they need to manually enter in this field.
   - Select the *Minutes step* and the *Seconds step*. For example, you can set them if you want the user to select a mutiple of 10 for the minutes or the seconds.
   - Select an *Initial Focused Time*.

     > **Note**
     >
     > The *Initial Focused Time* is the time that will be displayed when your user opens the *Time picker*. If you do not select one, the time that will appear when the user opens the *Time picker* will be the current time.

   - Select a time *Display format*.
   - Enable the *Required* toggle button if you want your user to see that setting a time is required.
   - Enter two variables in the *Time value* section. You can bind the first value to an input parameter and the second one to an output parameter.



3. Save your changes.
4. Drag and drop your user task into your automation.
5. Save and test your automation.
6. A popup appears. Click the icon to select your time.

> **i Note**
>
> A limitation has been set on user tasks depending on the SDK version you use. You cannot use the *Time picker* component if you have a tenant with an old SDK version. An error message will be displayed in the *Test console* panel.

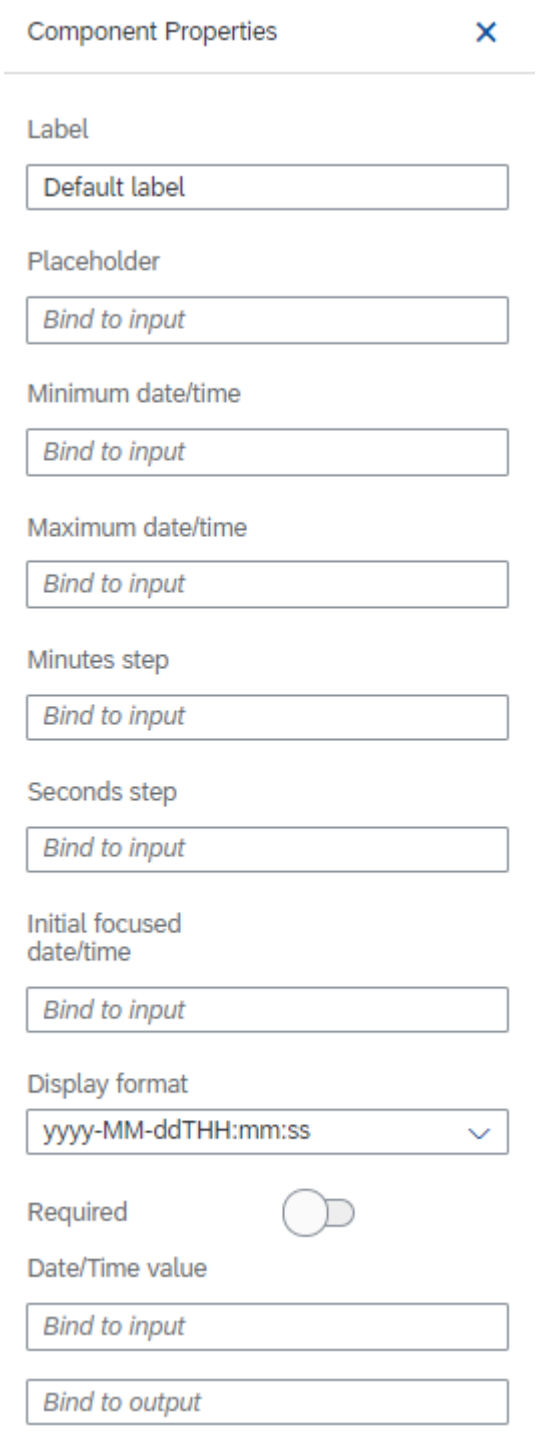
Insert a *Date/Time picker* Component

The *Date/Time picker* component allows you to select a date and a time in a user task. To add a *Date/Time picker* component to your user task:

1. Click or drag and drop the *Date/Time picker* component from the *Components* panel to the main panel.
2. Click the *Date/Time picker* component you added to define its properties in the *Component Properties* panel. You can:
   - Change or remove the *Label* text, which is displayed next to the *Date/Time picker*.
   - Insert a *Placeholder* text if you want to help users understand what they need to manually enter in this field.
   - Select the *Minimum Date/Time* and the *Maximum Date/Time* that the user will be able to select.
   - Select the *Minutes step* and the *Seconds step*. For example, you can set them if you want the user to select a mutiple of 10 for the minutes or the seconds.
   - Select an *Initial Focused Date/Time*.

     > **i Note**
     >
     > The *Initial Focused Date/Time* is the date and the time that will be displayed when your user opens the *Date/Time picker*. If you do not select one, the date and a time that will appear when the user opens the *Date/Time picker* will be the current date and time.

   - Select a date/time *Display format*.
   - Enable the *Required* toggle button if you want your user to see that setting a date/time is required.
   - Enter two variables in the *Date/Time value* section. You can bind the first value to an input parameter and the second one to an output parameter.



3. Save your changes.
4. Drag and drop your user task into your automation.
5. Save and test your automation.
6. A popup appears. Click the icon to select your date/time.

> **Note**
>
> A limitation has been set on user tasks depending on the SDK version you use. You cannot use the *Date/Time picker* component if you have a tenant with an old SDK version. An error message will be displayed in the *Test console* panel.

Optional: Change the Text of the Continue Button

To change the text of the green button whose default text is Continue, click the button, and change the label text in the *Component Properties*. You cannot delete this button.

Optional: Add a Reject Decision

In the footer area, you can add a second button, enabling the recipient of the message to either approve or reject the content defined by the user task. The buttons correspond to the decision output of the user task. A user task can't have more than two buttons.

1. Select *Click to Add a Reject Decision*.
2. A button with the default label "Reject" is added to the footer. Click the button to change the label text in the *Component Properties*.

> **Note**
>
> You can delete the second button via the 🗑 icon.

6.3.3 Define Input and Output Parameters

As for other skills that can be used to build a process, you can define input and output parameters to bind to the components of a user task.

Three predefined output parameters are available by default to bind to components in a user task:

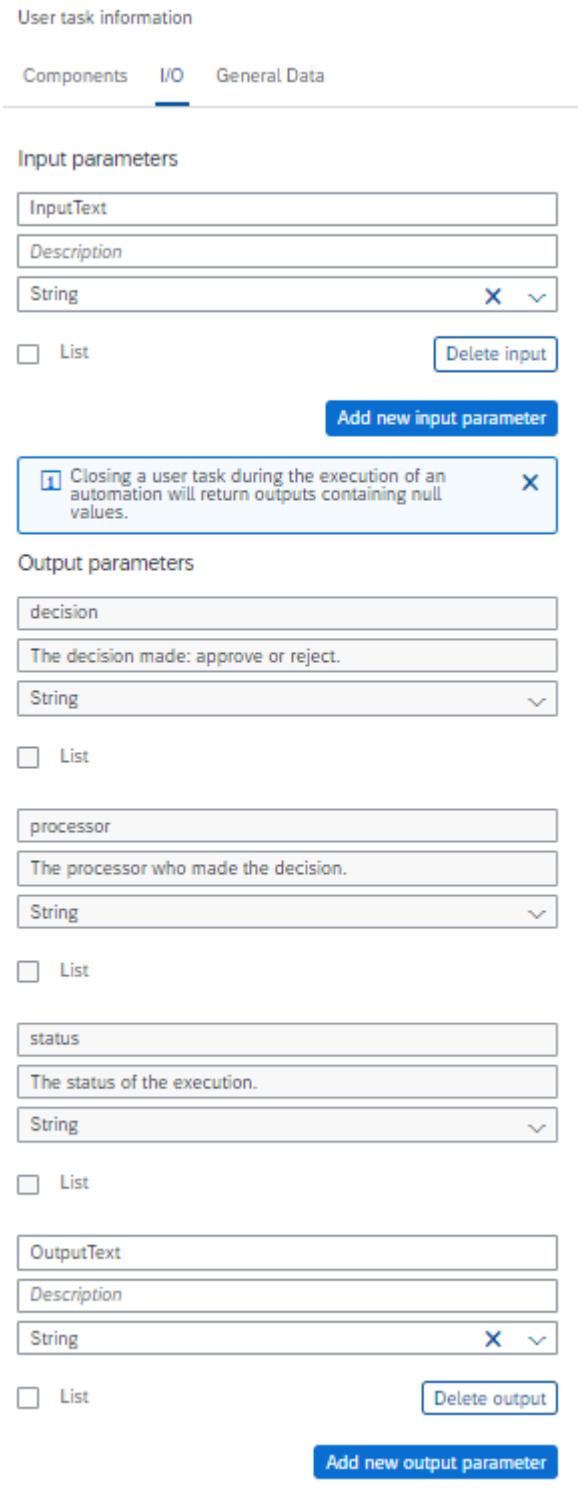
- decision: string parameter defining the decision made (approve or reject)
- processor: string parameter defining the person making the decision
- status: string parameter defining the status of the execution

You can't modify these predefined output parameters, but you can create additional output parameters.

To create input and output parameters for the user task:

1. Go to the *User task information* panel and click the *I/O* tab.
2. Choose *Add new input parameter* or *Add new output parameter*.
3. Enter a name and a description.
4. Select the type of parameter: string, number, boolean or an existing data type.

5. Save your changes.



> **i Note**
>
> You can delete of an input or output parameter that you've created by clicking ✕.

7 Reuse, Deploy Projects and Packages

Reuse a Package from the Store [page 507]
This feature allows you to reuse a package from the Store.

Generate Packages [page 508]

Environment Variables [page 509]
Environment variables allow you to reuse certain information for a given environment.

Manage Compatibility of Projects with Agents [page 513]
Managing the compatibility of a project with your agents is important to avoid future job execution issues.

Export a Project [page 519]
You can export a project from Cloud Studio and then import the exported project to another tenant.

Restore a Project [page 520]

Add Agent Attributes to a Project [page 521]

7.1 Reuse a Package from the Store

This feature allows you to reuse a package from the Store.

Reusing a Package

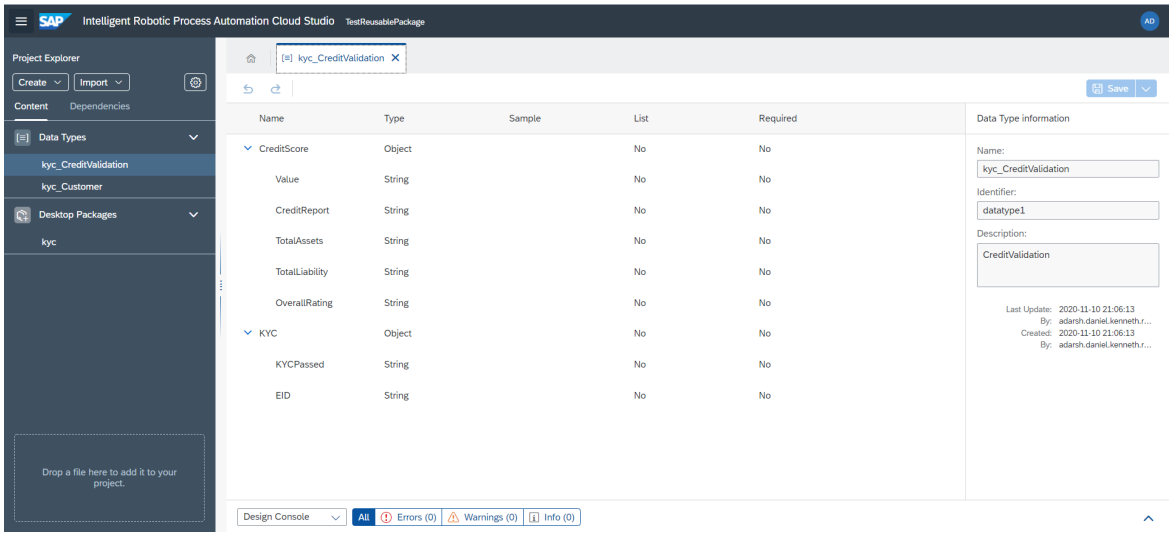
You can reuse a package from the *Store*, the package must be from another or same tenant and status as **Save As... New Project**. You can then create a new project from this package.

The package can be accessed from the *Store* and obtained in the *Project*. All automations, applications, and workflows of a project are available in the Cloud Studio.

Follow the below procedure to get a reusable package from the *Store*:

1. In the SAP Intelligent Robotic Process Automation Factory, click *Store* tab.
2. Find the package and then click the package to view the details. You can find the package status as "**Reuse the original design for building your own project**." under *Save As New Project*.
3. Click *Show All Versions*. The *Package Versions* pop-up appears and displays the package versions list, if available.
4. Click the *Get* button of the package or desired version of the package. After the get package is successfully processed, the confirmation message appears.

5. Click *Close* to exit the *Package Versions* pop-up.
6. Click the *Package* tab.
7. Find the package, click the ••• icon and then click *Save As... New Project*. The *Save As New Project* pop-up appears.
8. Update the **Project Name** and **Description** as required.
9. Click *Save As... New Project*. The *Project* tab opens.
10. Click the **Project**. The project is displayed in the Project Explorer in a separate tab.
    All existing automations, applications, and workflows of a project are now available in the Cloud Studio.



7.2 Generate Packages

Context

Once you have finished designing your project, you can generate a package from it.

Procedure

1. Go to the home page of the Cloud Studio and click *Generate Package*.
2. There are two possible situations:
    - When you generate a new package, enter a name and a description (optional). The Version Number of the package is 1.0.0. Click *Generate Package*.
    - When you generate a modified version of a package already generated:
        - Select *Contains only patches* to indicate bug fixes. It will update the third digit of the version number.

- Select *Contains minor changes* to indicate small modifications. It will update the second digit of the version number.
- Select *Contains significant changes which may impact dependent projects* to indicate important modifications potentially leading to incompatibility between versions. It will update the first digit of the version number.

Click *Generate Package*.

> **i** Note
>
> To prevent errors, please make sure you don't use the same name for multiple packages.

Results

The generated package is automatically available in the Cloud Factory. You can now combine this package to your environment to add one or several triggers that will run automation jobs.

7.3 Environment Variables

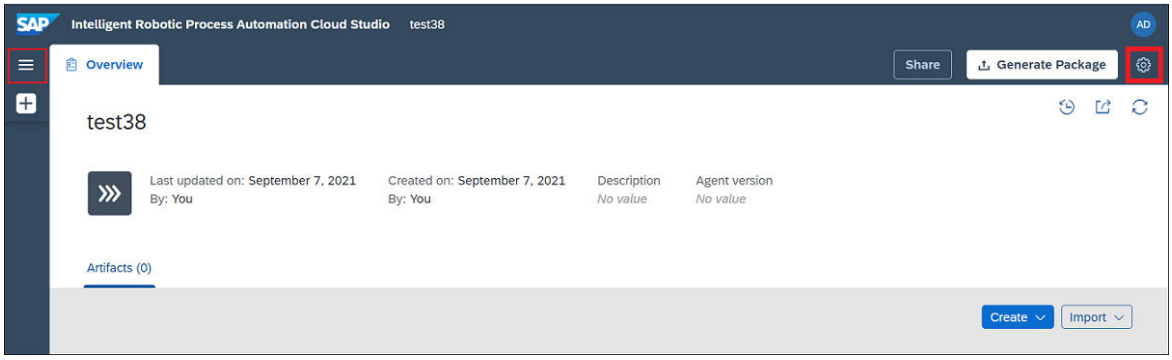
Environment variables allow you to reuse certain information for a given environment.

You use environment variables to pass parameters to automations. You can create environment variables in the Cloud Studio for which you can later set values across specified environments. For example, you can create a "S4HANA" variable with fields URL, login and password.

After creating the project successfully in the SAP Intelligent Robotic Process Automation Factory, the created project is displayed in the **Cloud Studio Project Explorer**.

Perform the following steps to create environment variables:

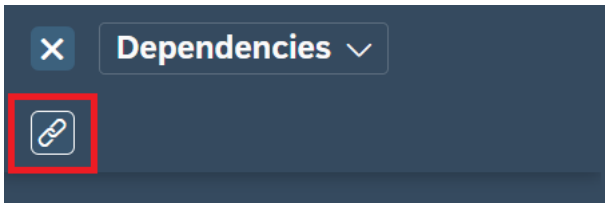
1. In the **Project Explorer**, click ⚙ or click *Menu* in the left panel of the **Project Explorer**.



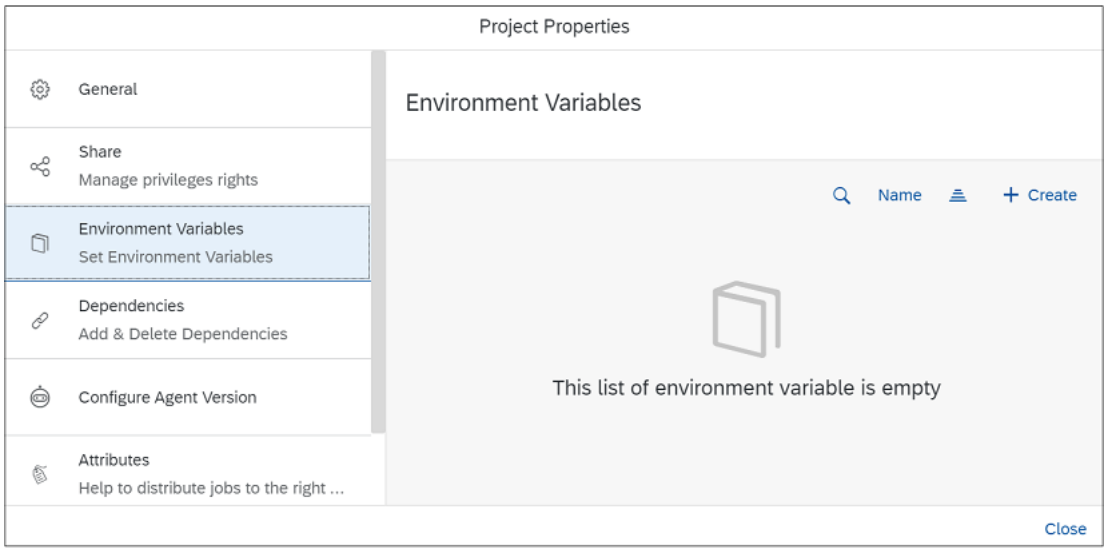
Click the drop-down arrow and then select *Dependencies*.



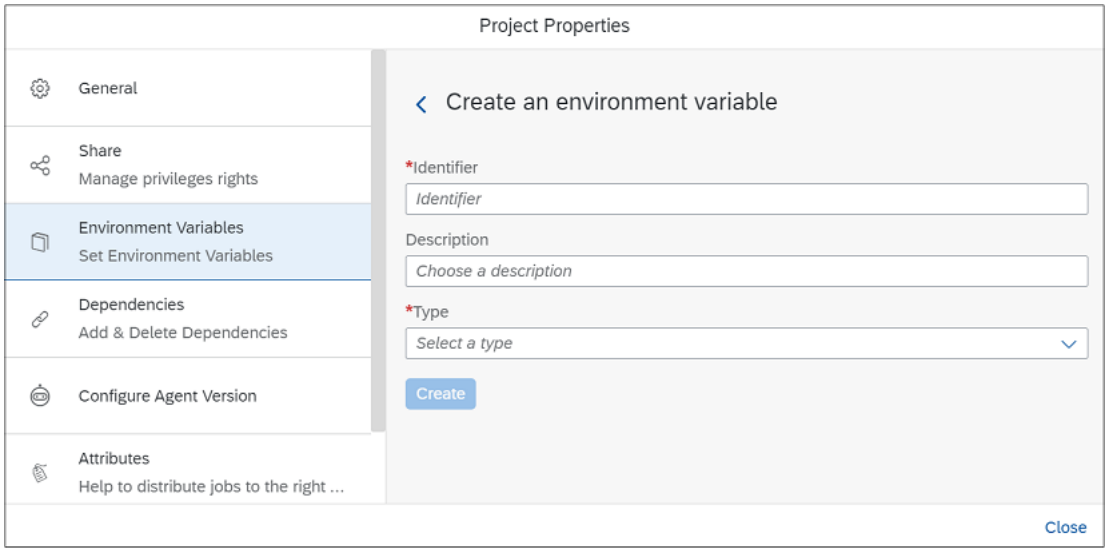
The *Manage Dependencies* icon is displayed.



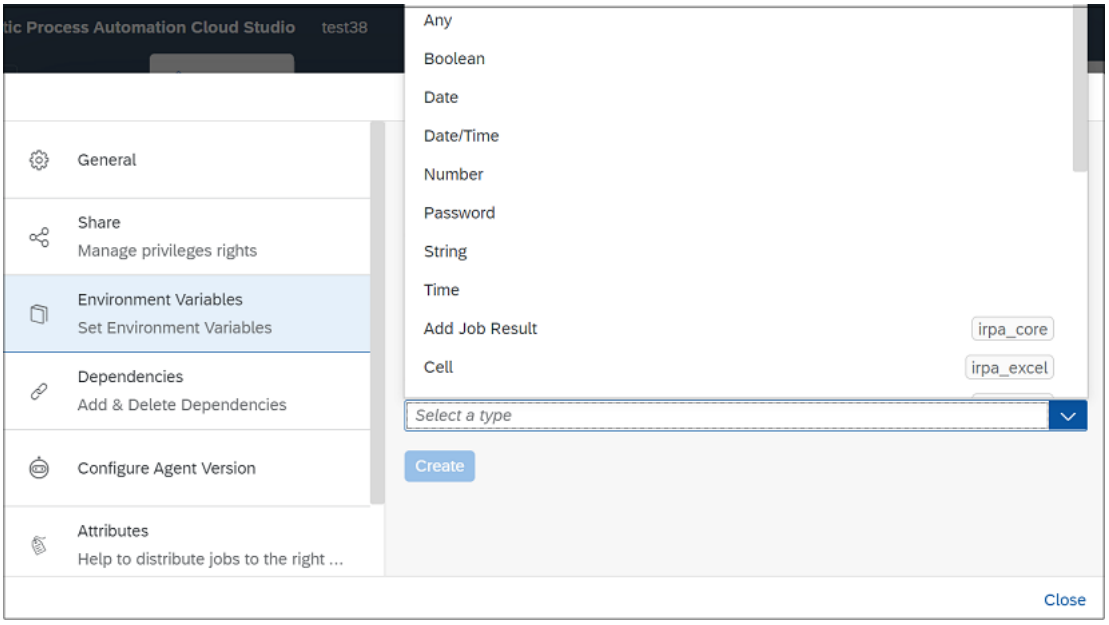
2. Click *Manage Dependencies* and then click the *Environment Variables* tab. The **Environment Variables** screen is displayed.



3. Click *+ Create*.
   The *Create an environment variable* is displayed.



4. Provide the variable details such as variable *Identifier* (An **Identifier** is a unique identity of a variable in the system.) , *Description*, and *Type* in the respective fields.
   1. Under **Type*, click in the *Select a type* field.
      The list of variable types is displayed.

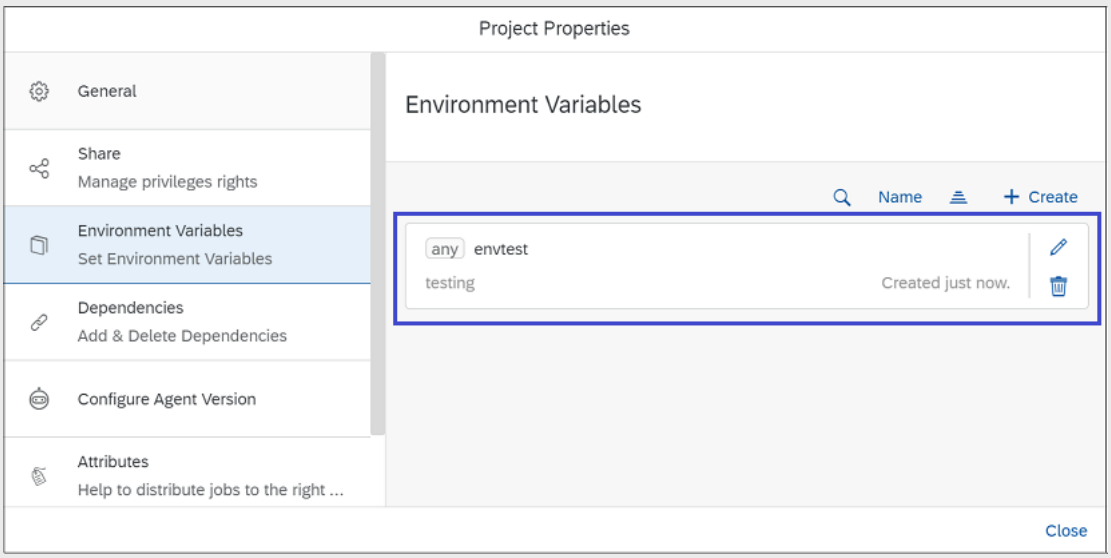


> **i Note**
>
> You can select variable type by:
>
> - Choosing predefined simple types (String, Number, Boolean etc).
> - Choosing a Data Type defined inside the same project.
> - Choosing a Data Type coming from SDK packages (core or excel etc).

2. Select the required variable type from the drop-down list.

5. Click *Create*. The environment variable is created.

> **i Note**
>
> You can create multiple environment variable types for a project.



How to Use Environment Variables?

Once you've created environment variables in your project, you can either:

- Use environment variables during the test of an automation in the Cloud Studio: in this use case, you must fill in the required values in the environment variable fields to allow the testing of the automation.
  For more information, see Manage Environment Variables in Automation Testing [page 291].
- Set values to environment variables when adding the package containing your project to an environment. This way the values of the environment variables are filled for a future automation execution.
  For more information, see Environment Variables from the Cloud Studio.

7.3.1 Best Practices for Environment Variables of Type Password

You can create an environment variable using the Password Type or Data Type that uses a password field. Then you can consume it in an automation.

To know more details about creating environment variables, refer to the following section:

- Environment Variables [page 509]

The following are recommendations for maintaining an environment variable of type password.

Do's

1. The password values must be created and managed in the SAP Intelligent RPA Factory environments:
   - The environment variable storage is secure.
   - The password values are stored in the credential store.

> **i Note**
>
> For more details about binding values to environment variables, refer to the Environment Variables from the Cloud Studio section in the SAP Intelligent RPA Factory User Guide.

Don'ts

- Do not write a password value in an automation as the password is not encrypted in the DB.
- Do not bypass the environment variables for passwords as:
  - The environment variable storage is secure.
  - The password values do not change frequently. So you don't need to input password values for every execution (API triggers).
- Do not use password variables to store information that must be hidden to the user of the session running the agent.
  Variables containing sensitive information are protected in the cloud and in transit to the agent. However, they are **not** protected on the client machine.
  If the information contained in variables must not be disclosed to regular users, use unattended automations on workstations that will be operated and accessed by a limited number of authorized users.

- The password is not hidden in the test mode:
  - TraceViewer do not hide the password values.
  - Timeline and logs can display the password values.

7.4 Manage Compatibility of Projects with Agents

Managing the compatibility of a project with your agents is important to avoid future job execution issues.

At runtime, your project interacts with an agent through SDK activities. But the SDK package you use may contain activities that have been developed after the agent you are using and may not be supported by the agent.

To avoid such a situation (that would cause failures at runtime), you should indicate the targeted agent version in your project settings. This agent version should correspond to the version of the agent that is effectively used in your production environment.

Setting the agent version for your project will have two effects:

- Only activities that can be used with that agent version will be available in the Cloud Studio and can be used within a project.
- The factory will refuse to deploy your project on agents with an older version than the one you specify.

> **i** Note
>
> It is recommended to use the latest agent version.

Configuring Agent Version

The activities that were not available before will be enabled when you configure the agent to a latest version. However, this restricts the number of agents that will support your project.

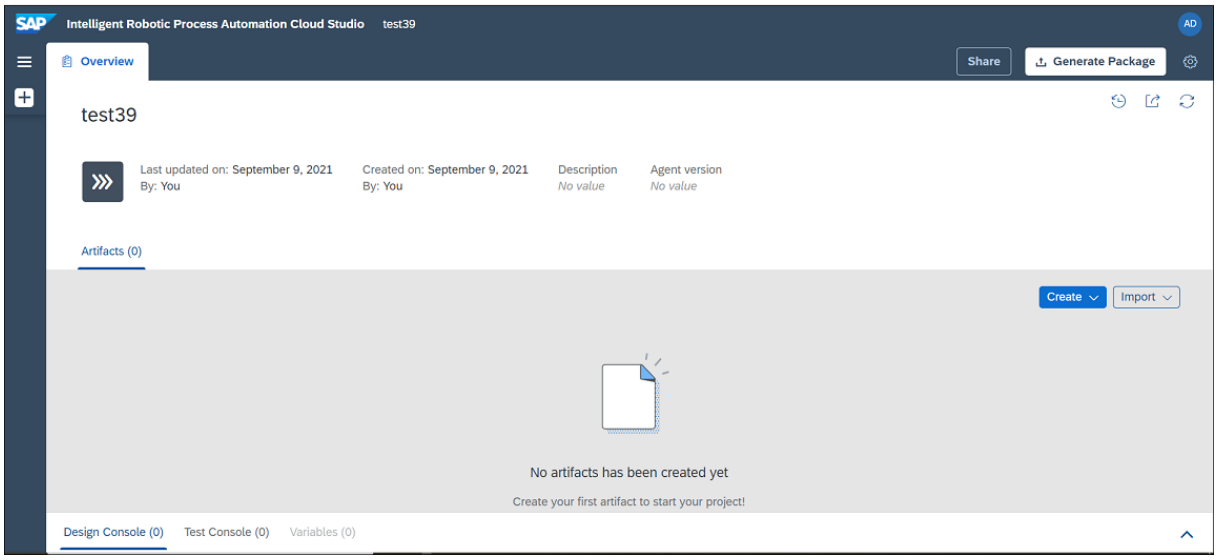
When you configure an agent to an older version, the project validation fails as the activities available in the current agent version may not be available in an older agent version. In this case, you will have to find other ways to implement these functionalities or change the agent version back to its original value.

7.4.1 Configure an Agent Version

You must configure an agent version before adding an automation in the project for the first time.

Context

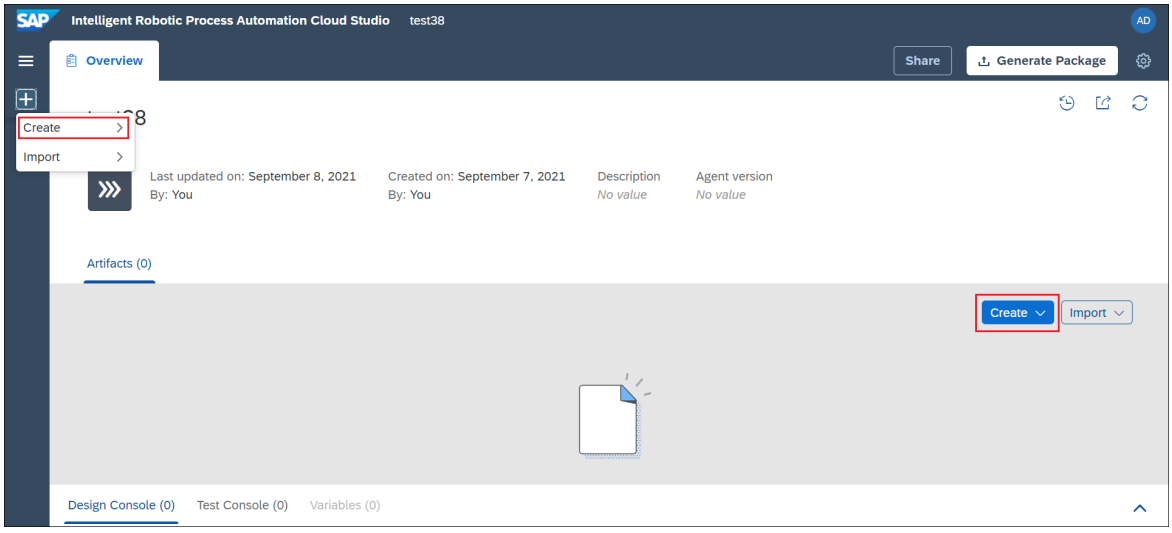
After creating the project successfully in the SAP Intelligent Robotic Process Automation Factory, the created project is displayed in the Project Explorer in a separate tab.



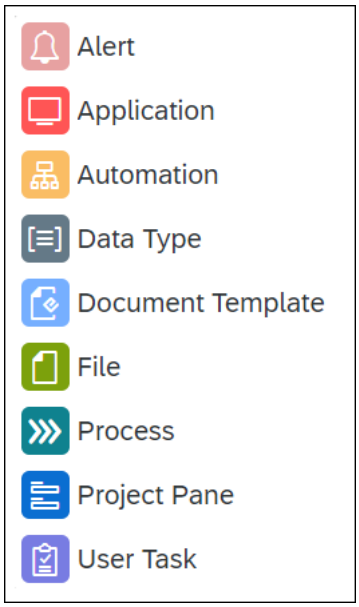
> ⚠ Caution
>
> **Please go through the below points before configuring the agent version:**
>
> - If an agent version is configured for the first time in a project, it is not required to be configured for later automations.
> - The agent version is automatically removed when the last automation of the project is deleted. While creating an automation, the *Configure Agent Version* dialog is displayed (as it is the first automation of the project).

Procedure

1. In the Project Explorer, click + and then click *Create* in the left panel or click *Create* in the Project Explorer main panel.

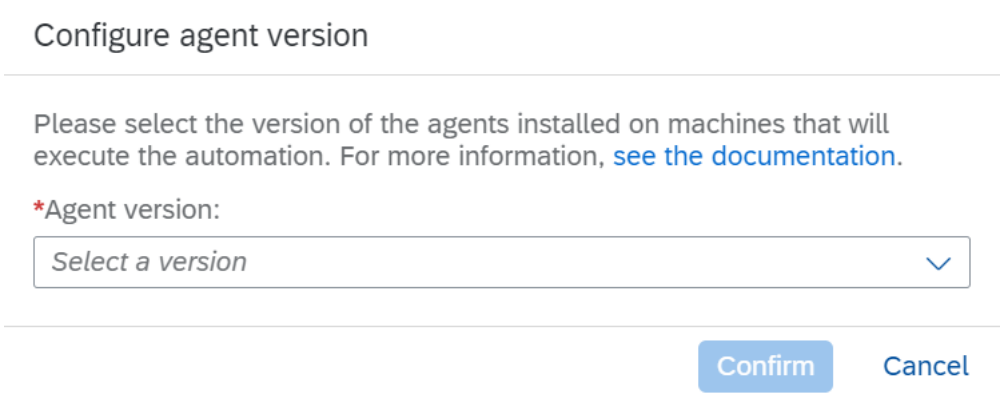


The artifact menu is displayed.

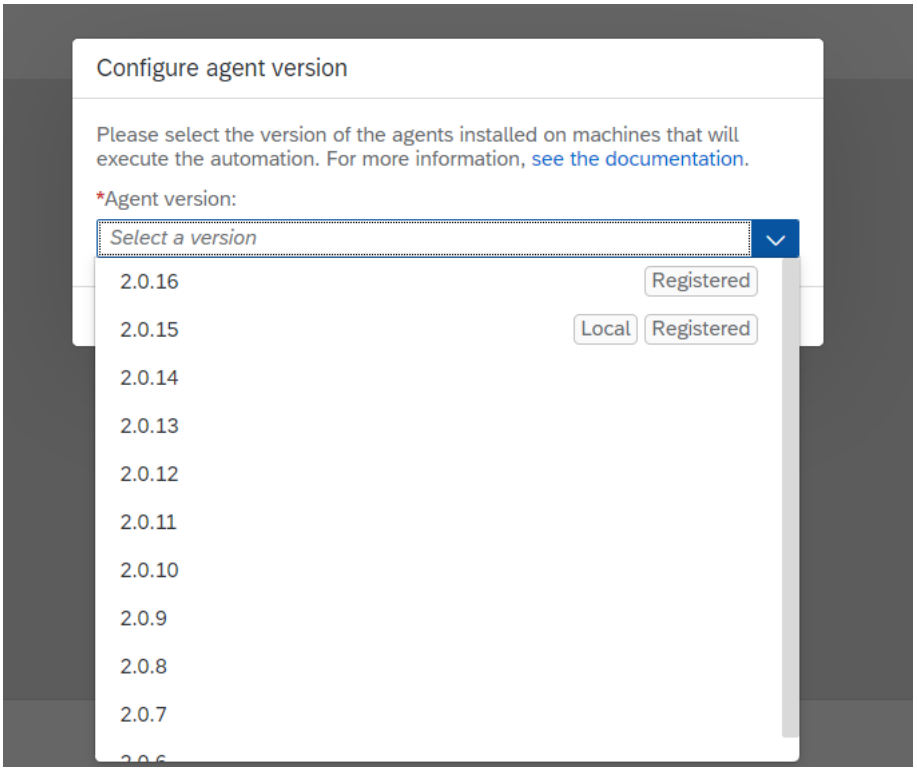


2. In the artifact menu click *Application*.

   The **Configure Agent Version** pop-up screen is displayed.



   1. Under **Agent Version:*, click the *Select a version* drop-down arrow. The agents that are installed in your local systems and registered in the tenant are displayed in the drop-down list. Select the required agent version from the list.

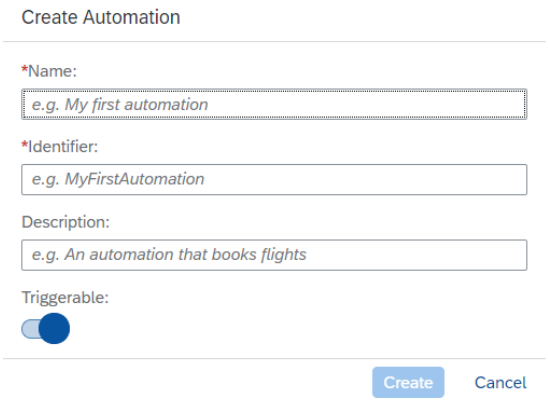


The *Agent version* is the (minimal) version of the agents that will be used to execute the project. The detected agent is tagged as *LOCAL* and *Registered*.

> **i Note**
>
> The minimum SDK version is dependent on the selected agent version.

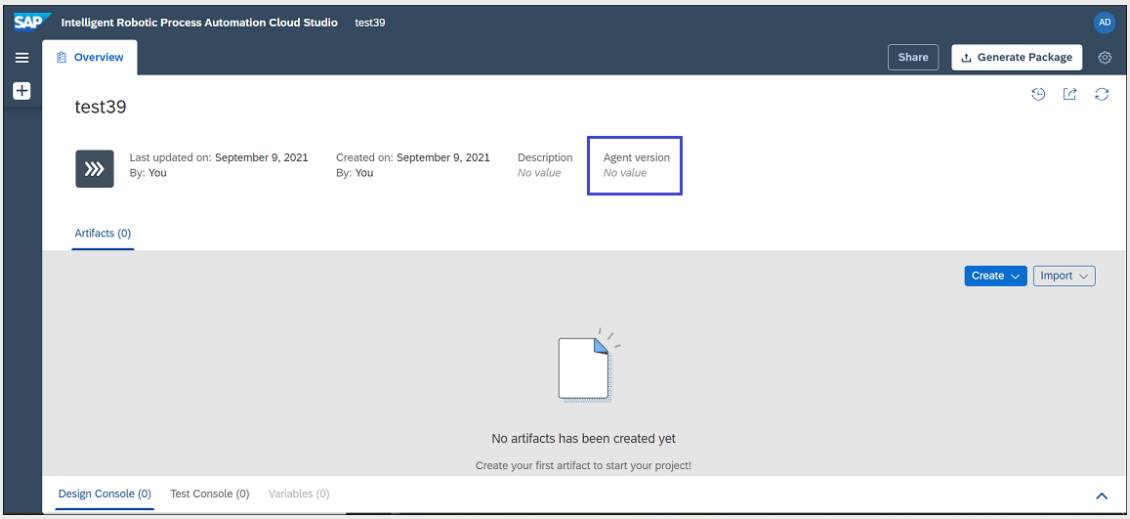
3. Click *Confirm*.

   The **Create Automation** pop-up screen is displayed.

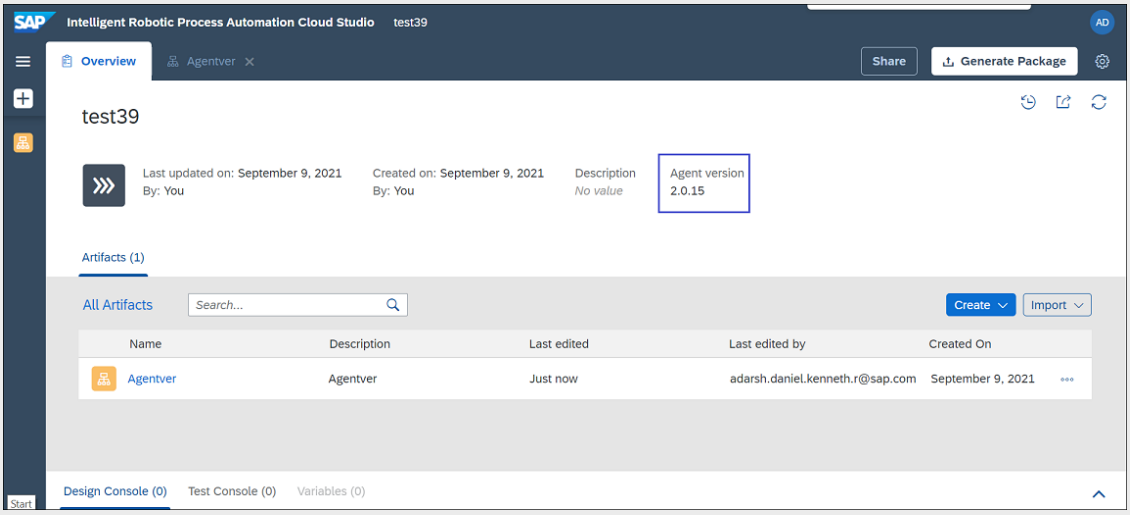


For more information about creating an automation, refer to the Create an Automation [page 156].

> **i Note**
>
> In the *Overview* screen of the Project Explorer, you can see the *Agent version* next to your project details. Before configuring the agent version, the *Agent version* indicates *No value*.



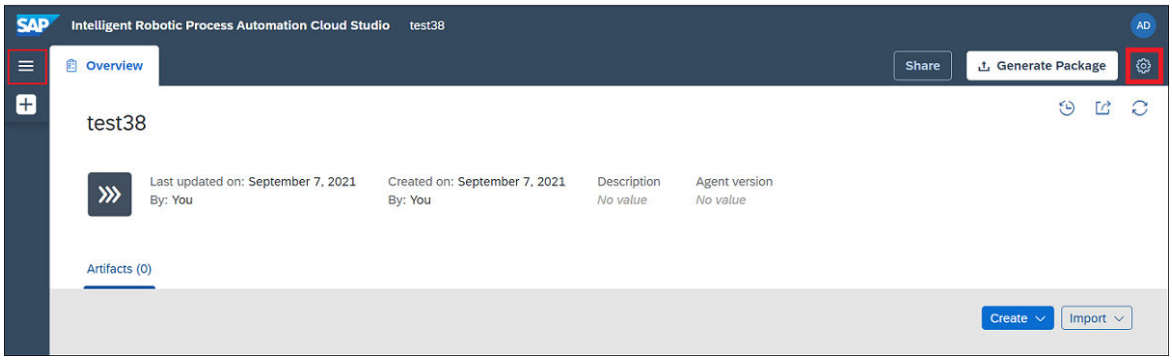
Once you have created an automation and configured the agent version, you can see the *Agent version* in the *Overview* screen of the Project Explorer.



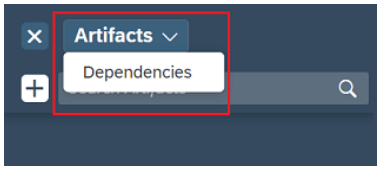
7.4.1.1 Update an Agent Version

You can update the agent version at any point in time.

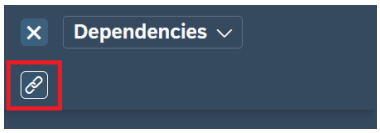
1. In the **Project Explorer**, click ⚙.
   OR
   Click *Menu* in the left panel of the **Project Explorer**.



Click the drop-down arrow and then select *Dependencies*.

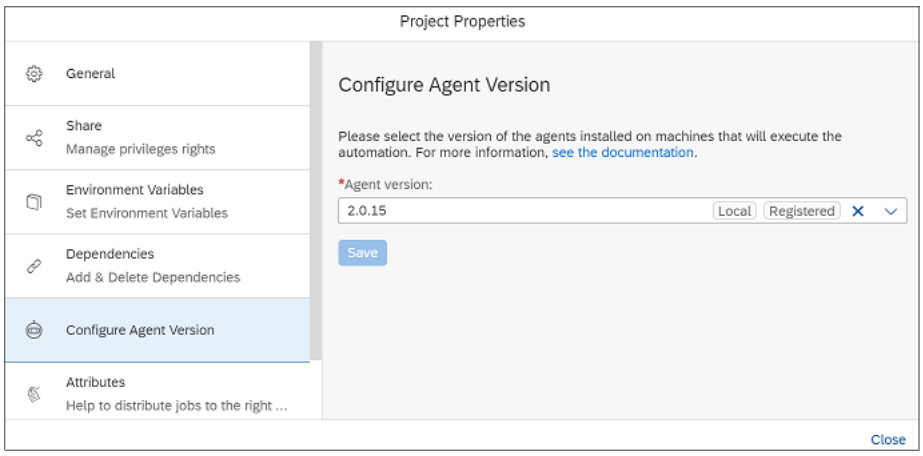


The *Manage Dependencies* icon is displayed.



Click *Manage Dependencies*. The *Project Properties* pop-up screen is displayed.

2. Click the *Configure Agent Version* tab. The **Configure Agent Version** screen is displayed.



3. Edit the agent version or select the desired agent version from the drop-down list.

> ## i Note
>
> You can enter the desired agent version in the **Agent version** field or select the agent version from the drop-down list. The minimum SDK version is dependent on the selected agent version. Based on the

defined agent version, only activities supported by this agent version are displayed and usable during the automation creation.

You can deploy the package version generated from the project on any agent with equal or higher agent version.

4. Click *Save*. The **Agent version updated** confirmation message is displayed.
5. Click *Close* to exit the *Settings* pop-up screen.

7.5 Export a Project

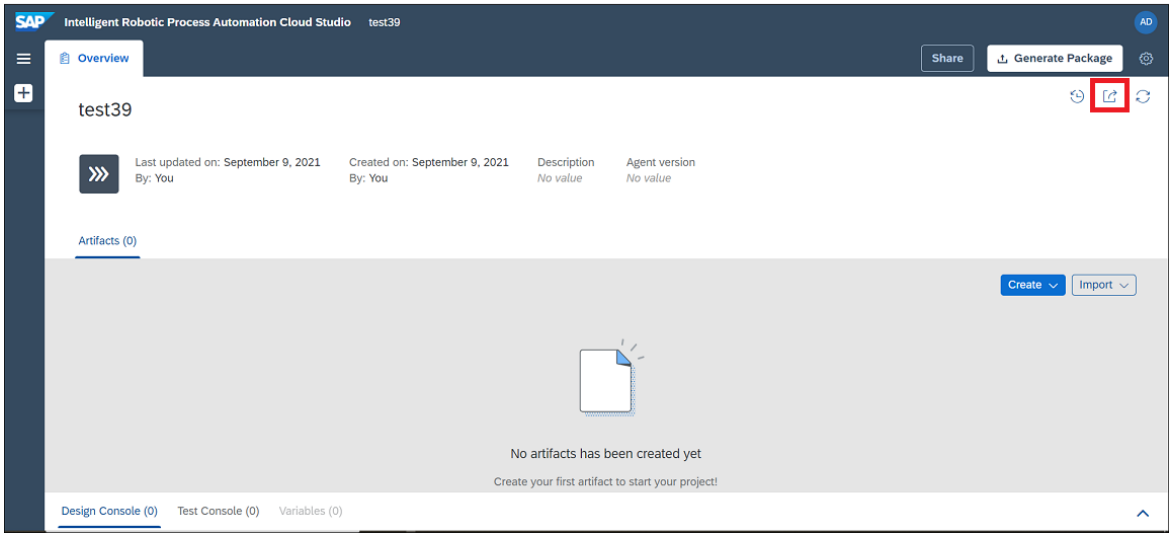
You can export a project from Cloud Studio and then import the exported project to another tenant.

You can export a project from the Cloud Studio home screen. For more details about creating a project, refer to the Create a Project [page 6] section.

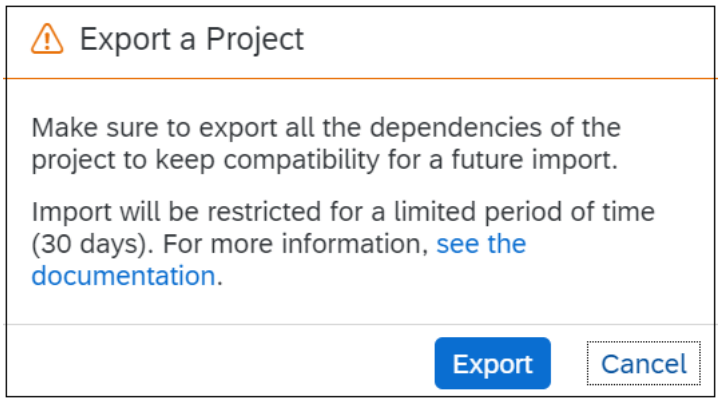
> **i Note**
>
> The project is exported as a ZIP file. It contains the checksum and is signed using a landscape key.
>
> The exported project is available in the archive folder up to 30 days.
>
> The exported project will not have any restrictions on the project validation.
>
> The exported project does not include the contents of Dependencies [page 525]. So, the dependency packages need to be exported separately, except if they're from the Store.
>
> The size of the exported ZIP file must be no larger than 256 MB.

Procedure

1. In the Cloud Studio home screen, click the icon.



The *Warning* pop-up message is displayed.



2. Click *Export*.
   The **.zip** file is now downloaded.

> **i Note**
>
> The download is managed by the browser.

> **i Note**
>
> You can also export a project with errors (error in workflow).
>
> You can also export a project with dependencies from another package. For more details about importing a package with decencies from another package, refer to the Import a Project section.

7.6 Restore a Project

Context

You can restore the content of a project from package versions generated from the project. The content of the project will be replaced by the content of the chosen package version.

> **i Note**
>
> There are restrictions on the versions that can be used to restore a project:
>
> - You can only restore a package with the same major and minor version as the lastest built package and that was created after the latest available release.
> - You cannot restore a version that is strictly prior to the last release. For example, you must delete version 2.0 in order to restore version 1.1.

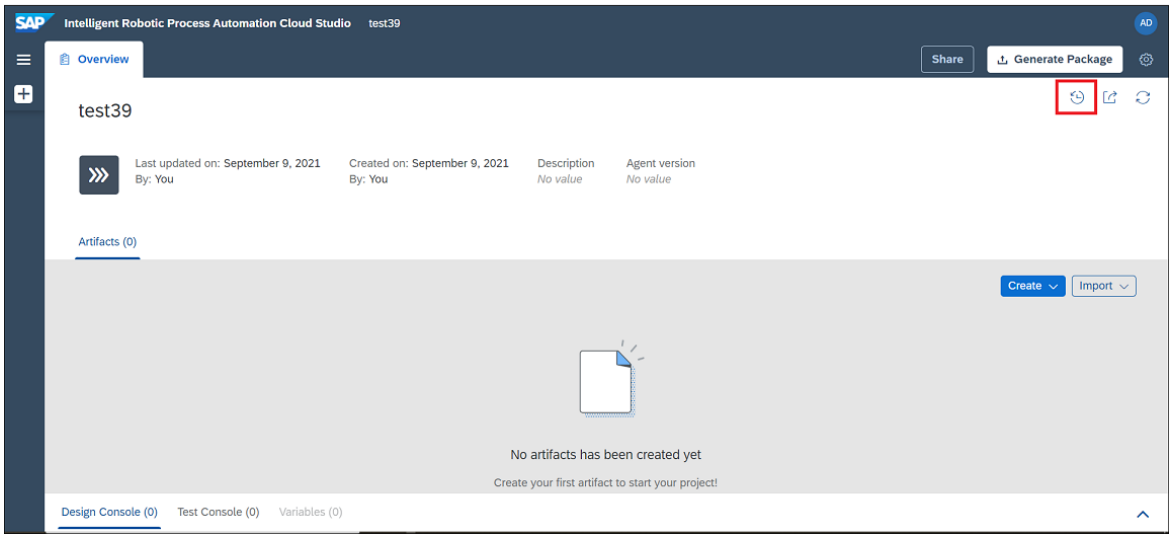
Prerequisites

In the SAP Intelligent RPA Cloud Factory, in *Projects* tab, click the project you wish to restore to open it.

> **i** Note
>
> This project must have several package versions.

Procedure

1. On the home panel, click the *Restore a version* button .



2. In the *Select version to restore* popup window that appears, you can see the dropdown list of the different package versions generated from this project. Select the version you wish to restore and click *Confirm* to restore it.

Results

Your project is now restored.

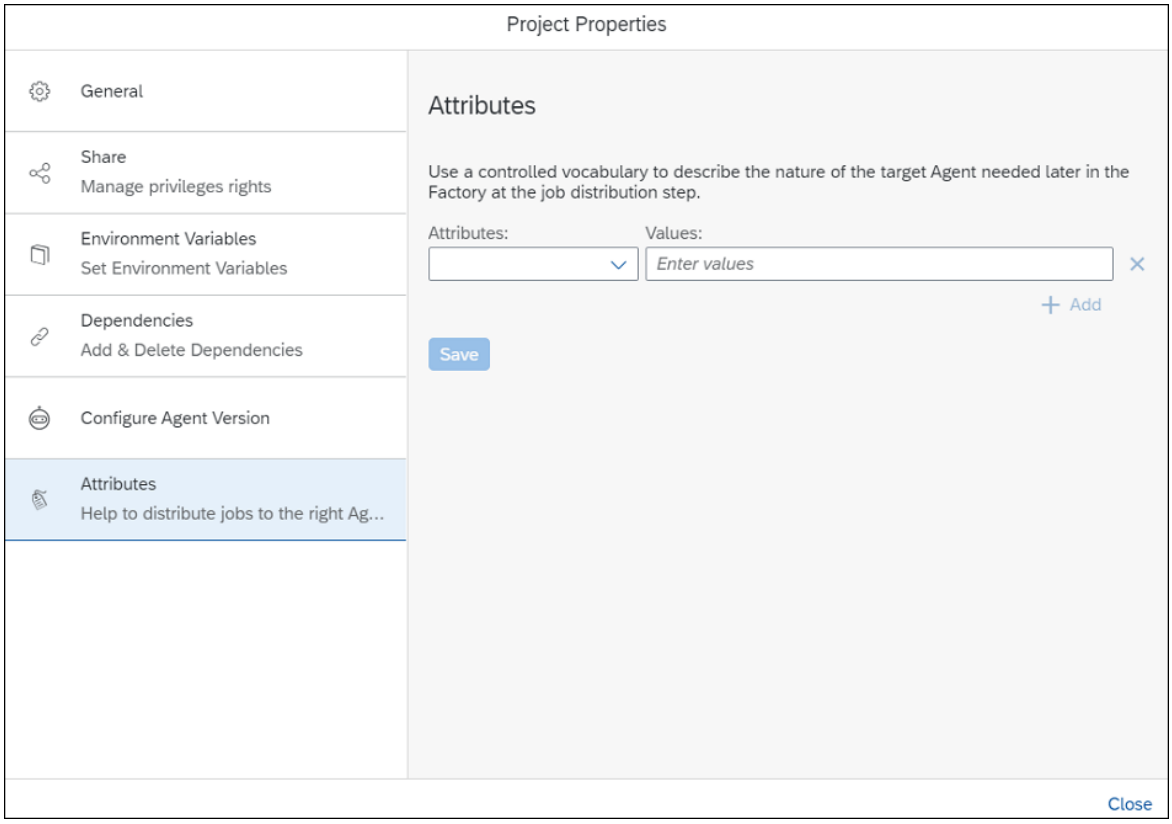
7.7 Add Agent Attributes to a Project

Agent attributes allow you to have a better and more fine-grain job distribution on agents. For more information, see Agent Attributes.

Perform the following steps to add agent attribute directly to a project.

Procedure

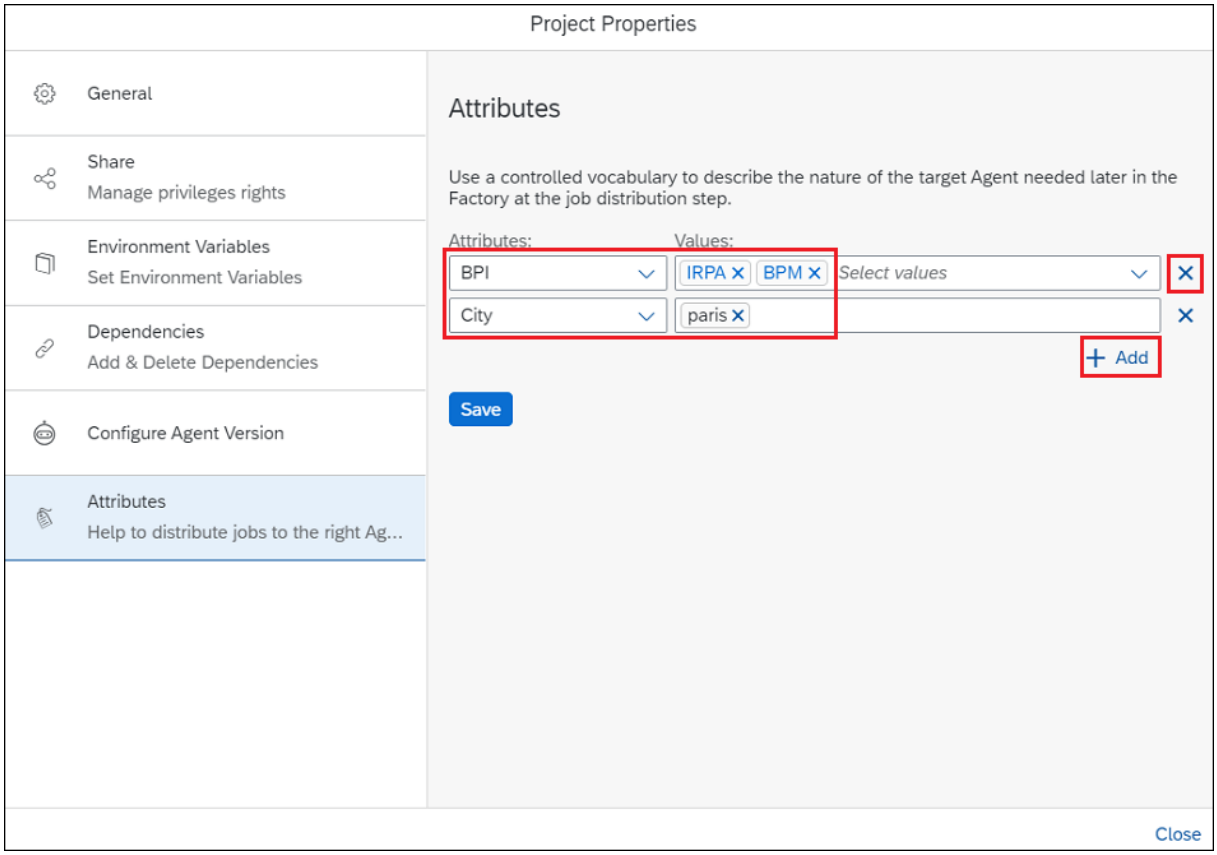
1. Click the *Settings* ( ) button. The Settings pop-up screen is displayed.
2. Click *Attributes*. The attributes pop-up screen is displayed.



3. From the *Attributes* and *Values* drop-down list, select the respective attribute name and its corresponding attribute value(s) for the predefined type attribute. When you select the free type attribute from the *Attributes* drop-down list, enter the attribute value(s) in the *Values* field.
4. Click the *+ Add* button to add new attribute to the list.
5. Click *Save*.
6. Click the *X* button to delete any attribute from the list.

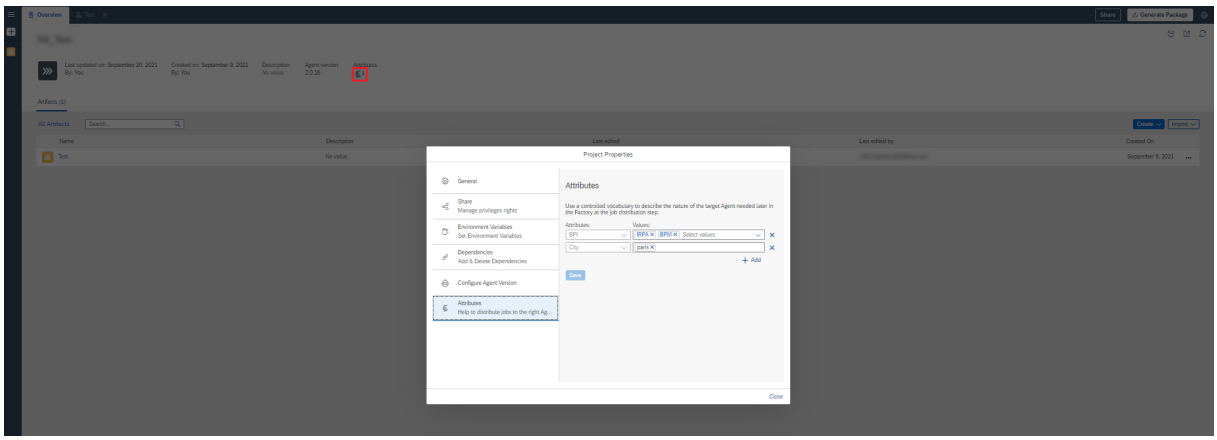
Result

The attributes are successfully added to the project.



Further Details

In the overview page of your project, you can see the number of attributes beside icon. When you click on the number of attributes, the attributes that are defined on the project are displayed.



Related Information

Agent Attributes
Add Agent Attributes to a Trigger
Add Agent Attributes to an Agent

8 Dependencies

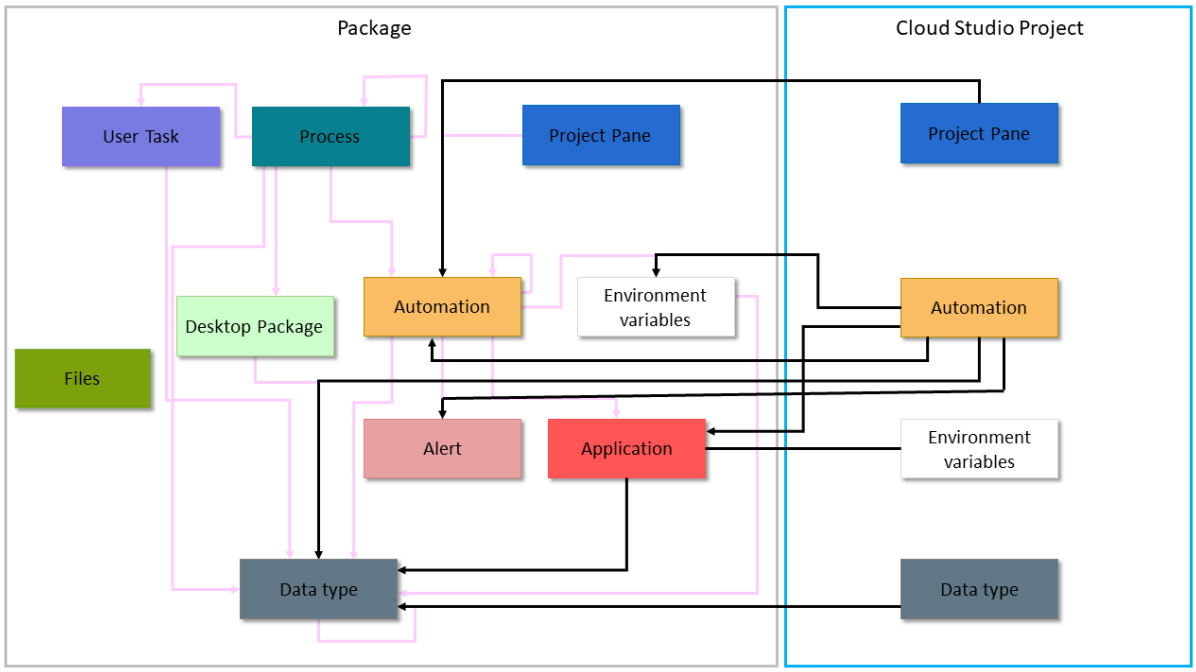
In the Cloud Studio, the characteristics and content of a package can be reused by another project through the structure of a dependency.

How artifacts are integrated through a package dependency

You can import packages as dependencies in a project. The packages contain artifacts such as automations, applications, data types and activities that you can use to build your automation. For more information on how to add a dependency to your project, see Manage Project Dependencies [page 526].

When a package is imported as a dependency in a project, part of the artifacts present in the imported package are available to be integrated to your project (linked in black in the interactive image below). These artifacts can be reused like the other artifacts directly created in your project.

> **i Note**
>
> A package needs to have sharing authorizations to be used as a dependency. For more information, see Share a Package.

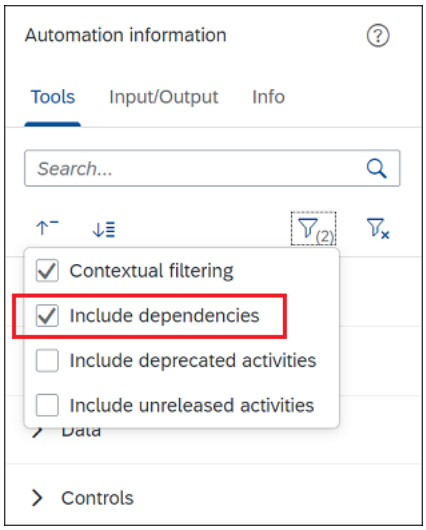


- Design Automations [page 153]

- Design Automations [page 153]
- Environment Variables [page 509]
- Environment Variables [page 509]
- Trigger Automations in Attended Mode using the Project Pane [page 352]
- Trigger Automations in Attended Mode using the Project Pane [page 352]
- Data Types [page 168]
- Data Types [page 168]
- User Tasks [page 489]
- Processes [page 482]
- Capture and Declare Applications [page 75]
- Alerts [page 357]

Further Details

In the *Tools* section of your project, you can filter the content to include or exclude the specific artifacts coming from dependencies.



> **i Note**
>
> You can only use dependencies directly imported in your project. If the package you import as a dependency has dependencies of its own, you cannot use the artifacts present in those dependencies.

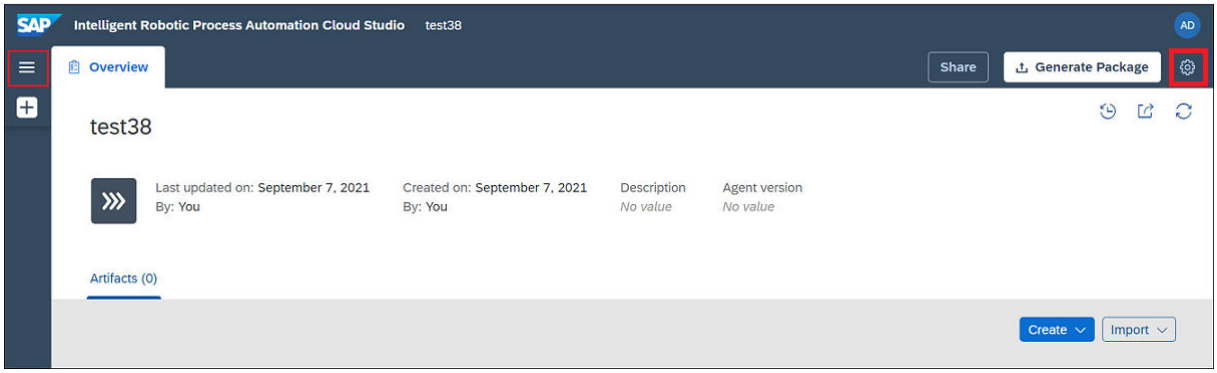
8.1 Manage Project Dependencies

You can add, remove or edit your project dependencies.

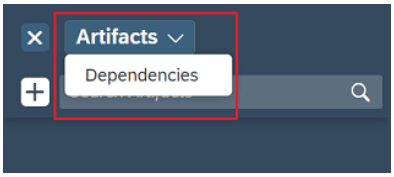
In the **Project Explorer**, click ⚙.

OR

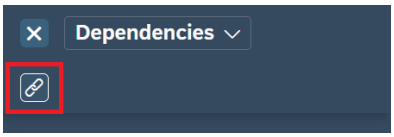
Click *Menu* in the left panel of the **Project Explorer**.



Click the drop-down arrow and then select *Dependencies*.

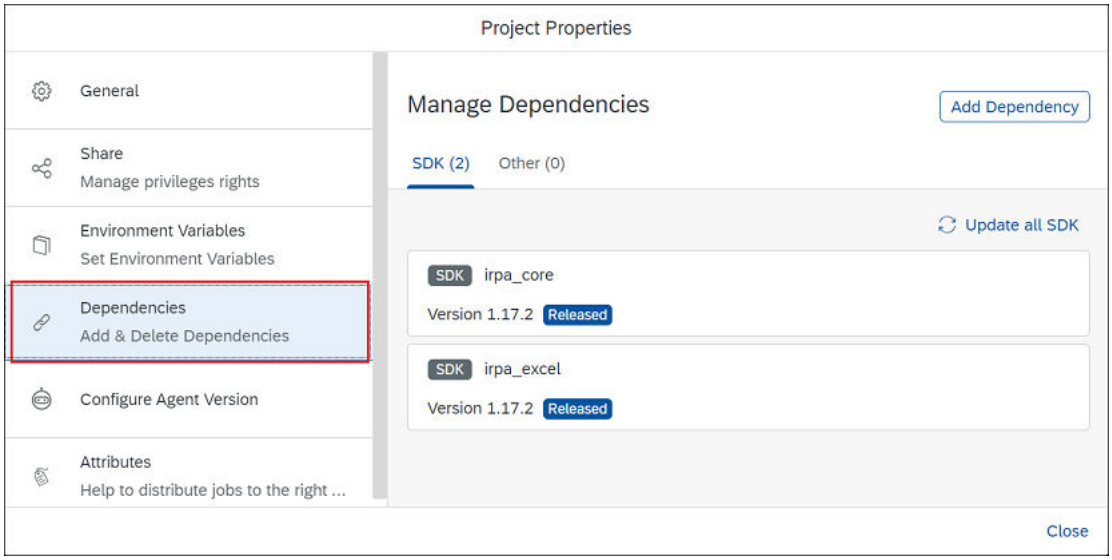


The *Manage Dependencies* icon is displayed.



Click *Manage Dependencies*. The *Project Properties* pop-up screen is displayed.

Click the *Dependencies* tab to manage your project dependencies.



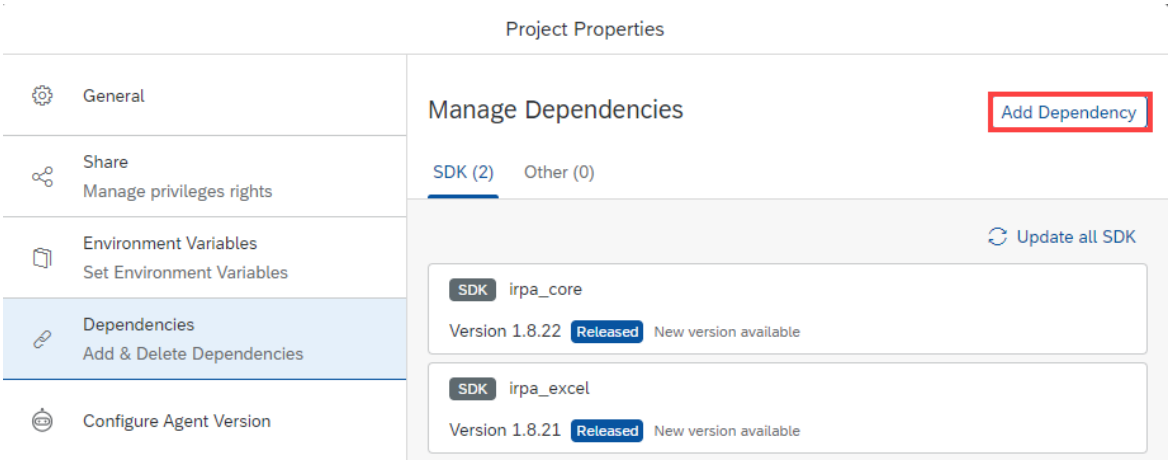
8.1.1 Add a New Dependency

Prerequisites

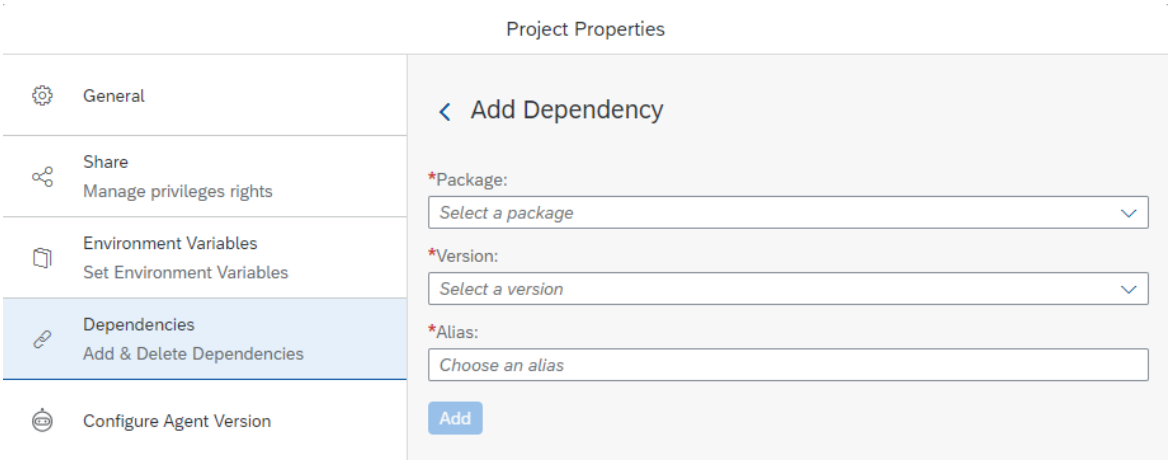
To import any package other than SDK packages as a dependency, it must be shared with *Read* authorization and *Anyone* in the Cloud Factory. For more information on how to share a package, see Share a Package.

Procedure

1. In the *Manage Dependencies* tab, click *Add Dependency*.



2. In the *Add Dependency* tab, select a package from the drop-down list.



   The latest version of the package is automatically selected.
3. **Optional:** Change the version of the package by choosing another version from the drop-down list.
4. Click *Add*.
   The package is now shown as a dependency in the *Dependencies* view. You can now use its content to create an automation.

8.1.1.1 Add an Alert as Dependency

Prerequisites

You have created one or several alerts in the Cloud Studio, as described in Alerts [page 357].

To import any package other than SDK packages as a dependency, it must be shared with *Read* authorization and *Anyone* in the Cloud Factory. For more information on how to share a package, see Share a Package.

Context

Now, you can add an alert as a dependency in the cloud studio. For more information on dependency, see Dependencies [page 525].

Procedure

1. In the *Manage Dependencies* tab, click *Add Dependency*.

2. In the *Add Dependency* tab, select the alert package from the drop-down list.

   The latest version of the alert package and alias of the alert are automatically selected.
3. **Optional:** Change the version of the alert package by choosing another version from the drop-down list.
4. **Optional:** Edit the alias of the alert.
5. Click *Add*.
   The alert package is now displayed as a dependency in the *Dependencies* View.

Results

You are now able to use this alert using the *Raise Alert* activity in your automation.

8.1.2 Update a Dependency

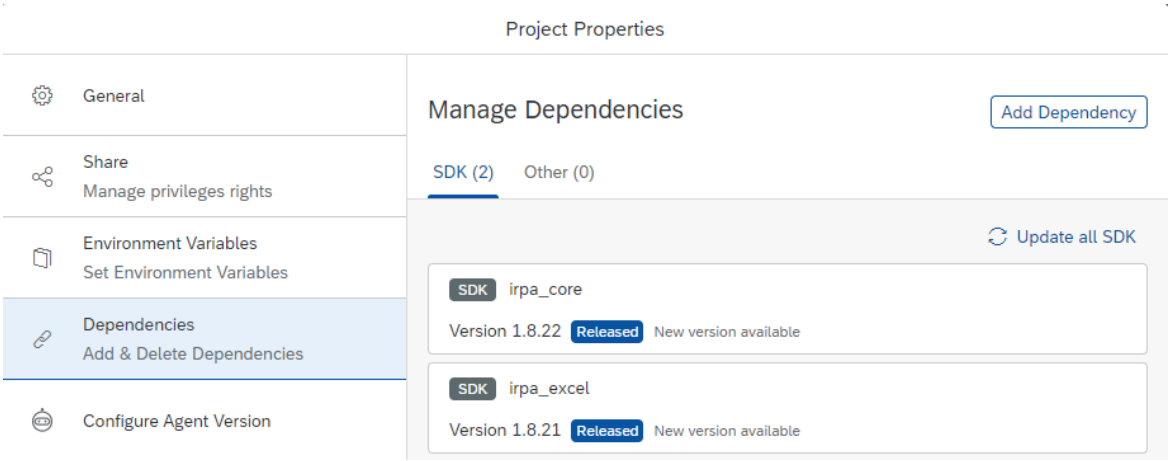
Context

If you go to the *Dependencies* tab in the *Settings* panel, you can see all the dependencies of your project. This is where you can update the version of your packages.

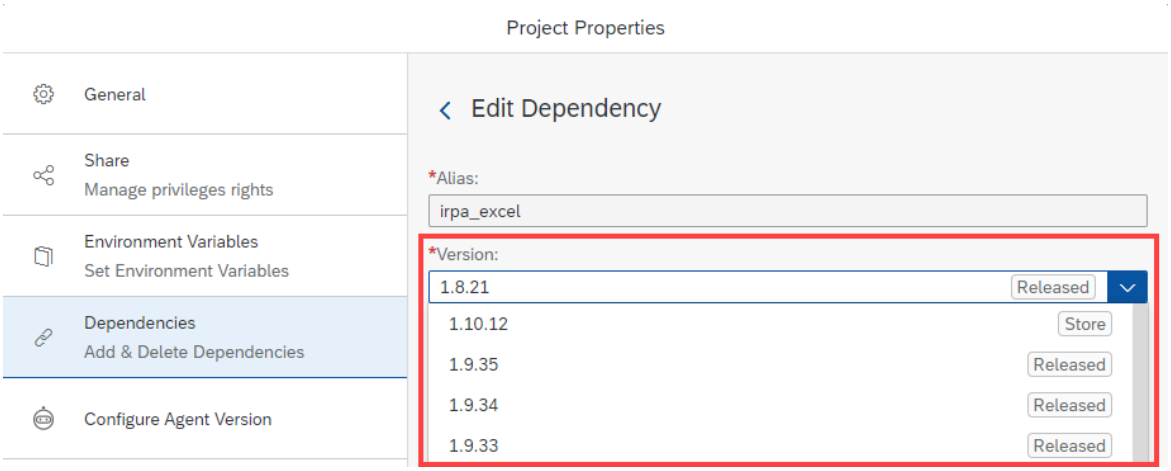
> **i Note**
>
> SAP Intelligent RPA grants upward compatibility for built-in SDK activities. However, the upward compatibility for the content of custom scripts cannot be granted. For more information, see Custom Scripting [page 251].

Procedure

1. In the *Manage Dependencies* tab, click the package you want to update.

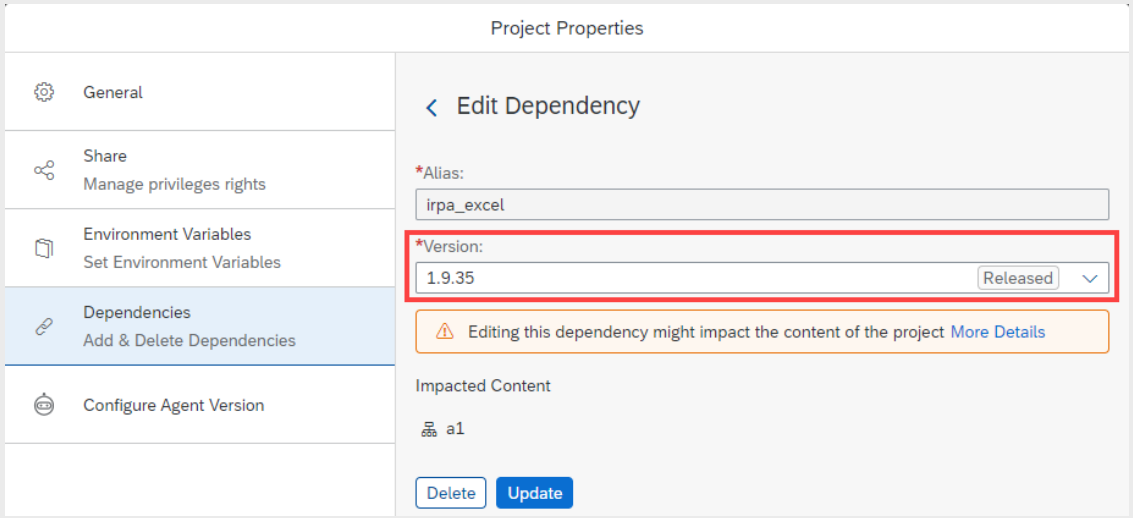


2. In the *Edit Dependency* tab, select the version of the package.



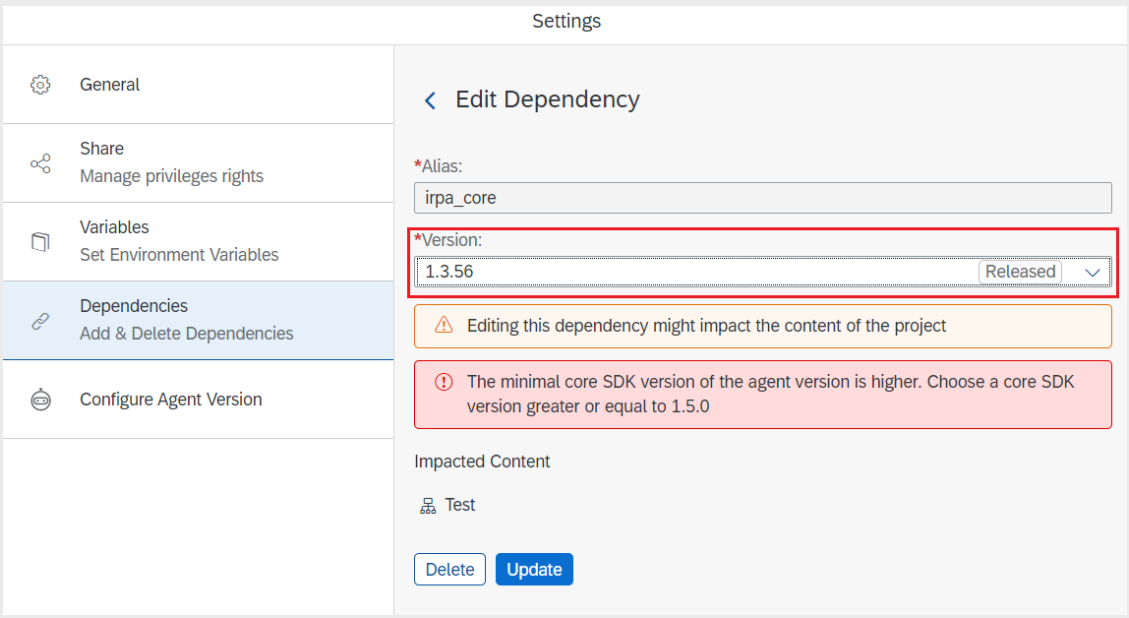
⚠ Caution

If you edit the current version of the dependency, the content of the project can be impacted.



⚠ Caution

While updating an SDK version inside your project, make sure you choose a version that is equal to or higher than the minimal core SDK version required by the agent version.



> ⚠ Caution
>
> It is not recommended to downgrade the SDK version. If you downgrade the SDK version, it can cause irremediable issues.

3. Click *Update*.

> ⚠ Caution
>
> This is the recommended approach to update a dependency. Indeed, this procedure guarantees a continuity with the previous version of a dependency without breaking anything in your project.

8.1.3 Delete a Dependency

Procedure

1. In the *Manage Dependencies* tab, click the package you want to delete.
2. Click *Delete*.

> ⚠ Caution
>
> Decommissioning a package used in a project prevents all other users from using it. For more information, see Package Versioning and Lifecycle Management.

8.1.4 Decommission a Dependency

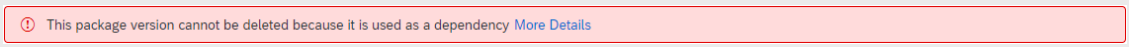
Context

Decommissioning a package used in a project prevents all other users from using it. For more information, see Package Versioning and Lifecycle Management.

Procedure

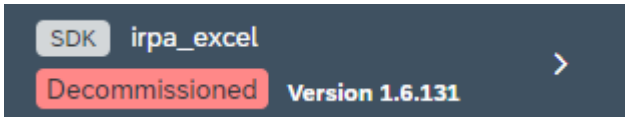
1. On the *Packages* tab, select your package and click *Show previous versions*.
2. Select the package version you want to decommission.
3. Click *More Options* ••• .
4. Select *Decommission*.

   A *Decommission Package* window appears.
5. Click *Decommission*.

> **i Note**
>
> If the package version is referenced in multiple projects or packages, a warning message appears.
>
> 
>
>
> If you click on *More Details*, a *Package Deletion Error* window appears where you can see the dependencies referencing the package version.

Results

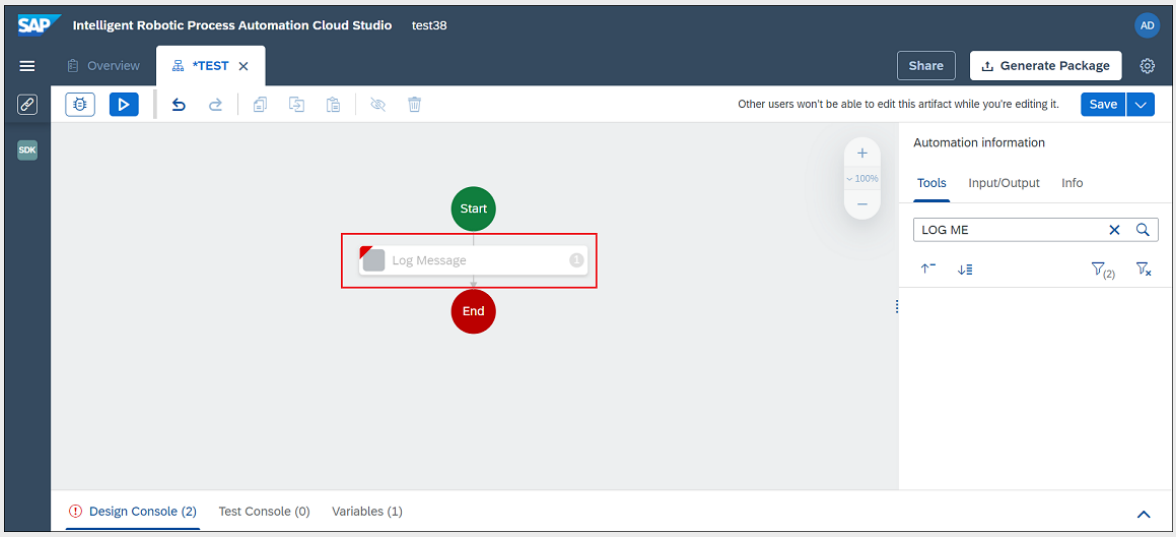
The dependency is decommissioned and a warning label is now visible next to the dependency in the Cloud Studio.



8.2 Project Dependency Removal and Re-creation

At any point in time if you remove a dependency of a project, you can re-add the dependency with the same dependency version and the alias must be the same as earlier. The project will behave same as earlier (before removing the project dependency) and you can generate a package.

> **Note**
>
> Once the dependency of a project is removed, the dependent steps in a workflow will be grayed and you cannot generate a package. For example, refer to the following screenshot:



After re-creating the dependency with the same dependency version and alias same as earlier, the greyed steps will be enabled and behave the same as earlier (before removing the project dependency) and you can generate a package.

Consider you have three package versions generated with different content of a project. In this scenario, if you remove a dependency of a project and then re-create dependency the following scenarios may arise:

Re-creating Dependency with Lower Version

If you re-create the dependency with a lower dependency version, the project will behave the same as earlier (before removing the project dependency) if the used artifact from the dependency exists in the lower version.

> **Note**
>
> There may be some errors due to the content of the dependent artifact which may be modified between the two version of the package.

If the dependency does not exist in the lower version, then the dependent steps of a workflow will be grayed and you cannot generate a package.

Re-creating Dependency with Same or Higher Version

If you re-create the dependency with the same dependency version and alias, the grayed steps will be enabled and behave the same as earlier (before removing the project dependency) and you can generate a package.

If you re-create the dependency with the higher version and the alias same as earlier, the grayed steps will be enabled and behave the same as earlier (before removing the project dependency) and you can generate a package.

> **i** Note
>
> There may be some errors due to the content of the dependent artifact which may be modified between the two versions of the package, for example, an artifact was present in version 1.2.0, but removed in version 1.3.0.

Related Information

Manage Project Dependencies [page 526]
Add a New Dependency [page 528]
Update a Dependency [page 530]

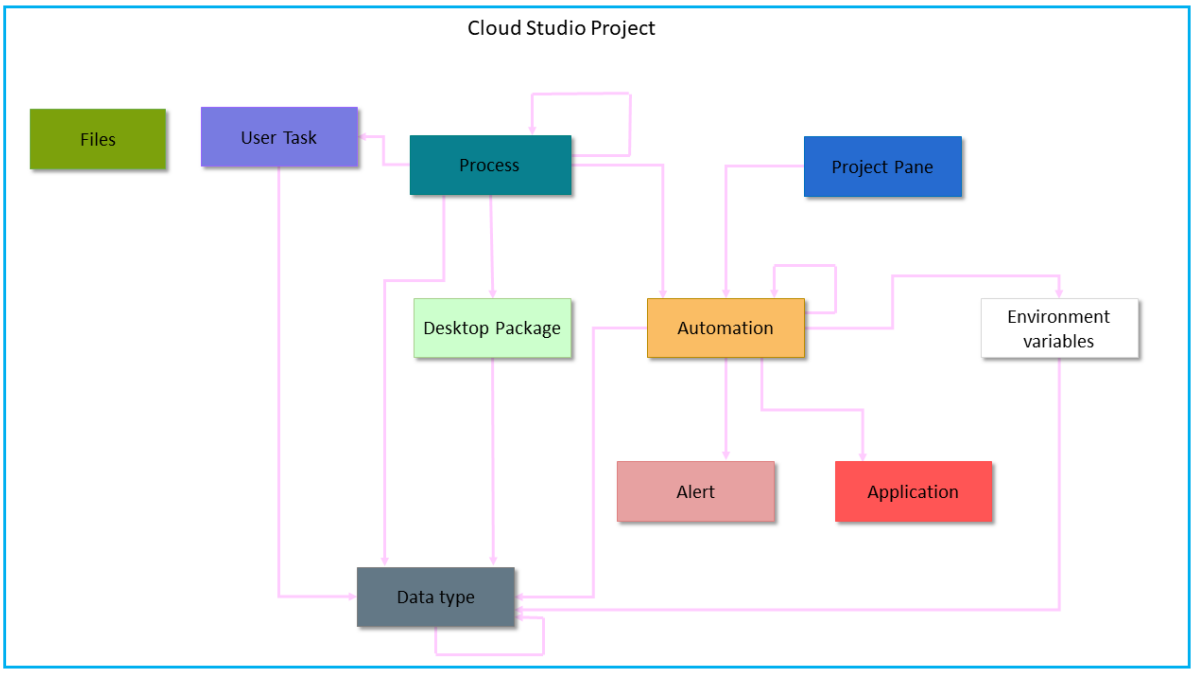
9 Miscellaneous

- Artifact Integration Within a Project [page 536]
- Collaborative Editing [page 537]
- Deactivate an Artifact [page 541]
- Duplicate Artifact [page 539]

9.1 Artifact Integration Within a Project

In the Cloud Studio, you can integrate different artifacts within a project.

How artifacts are integrated within a project



- User Tasks [page 489]
- Processes [page 482]
- Trigger Automations in Attended Mode using the Project Pane [page 352]

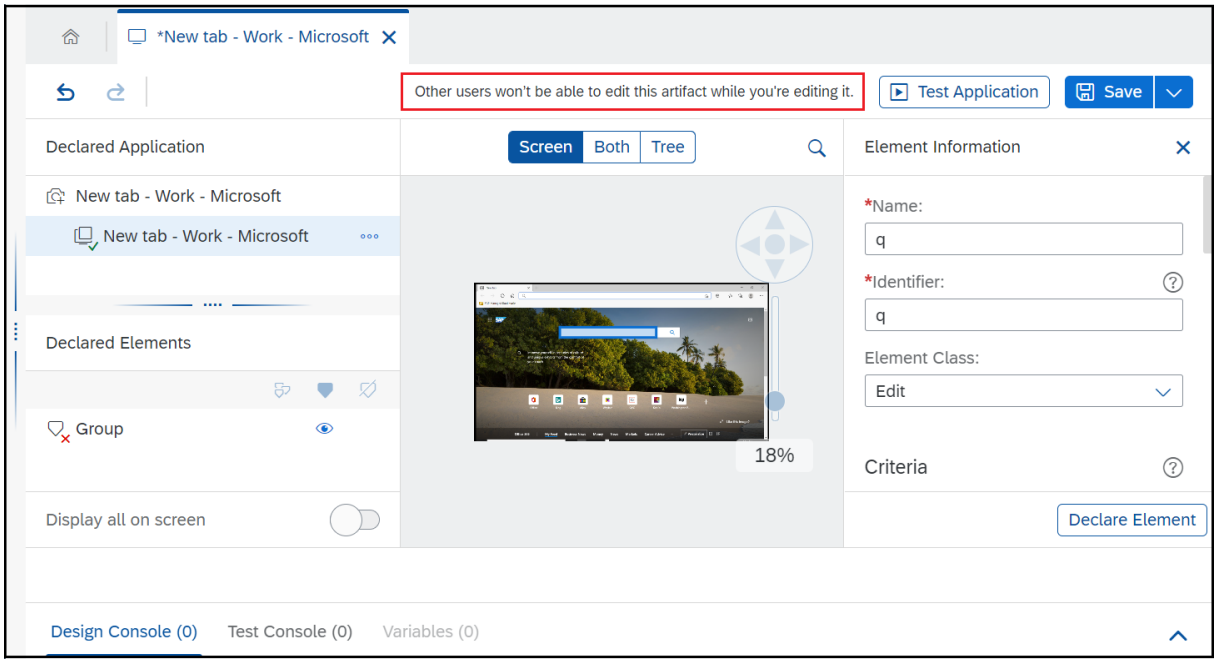
- Design Automations [page 153]
- Environment Variables [page 509]
- Data Types [page 168]
- Alerts [page 357]
- Capture and Declare Applications [page 75]

9.2 Collaborative Editing

A Collaborative Editing is handled automatically by the Cloud Studio.

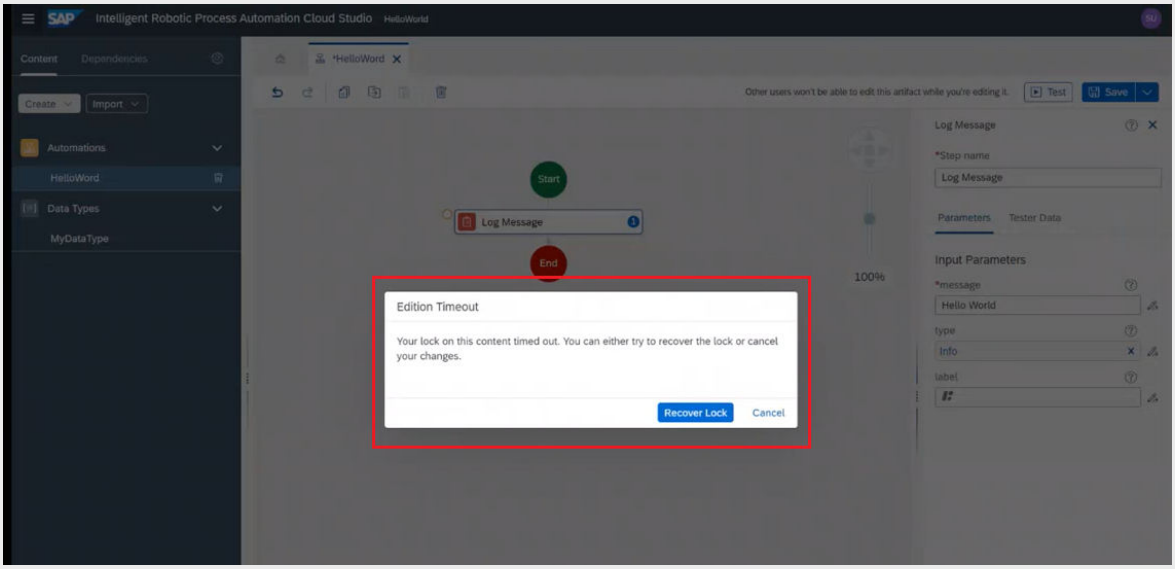
When you are working on an artifact (an automation, application, data type, user task, process) in the Cloud Studio, the artifact is automatically locked and other users will be unable to edit or modify the artifact.

When an artifact is locked, the following alert is displayed in the Cloud Studio.



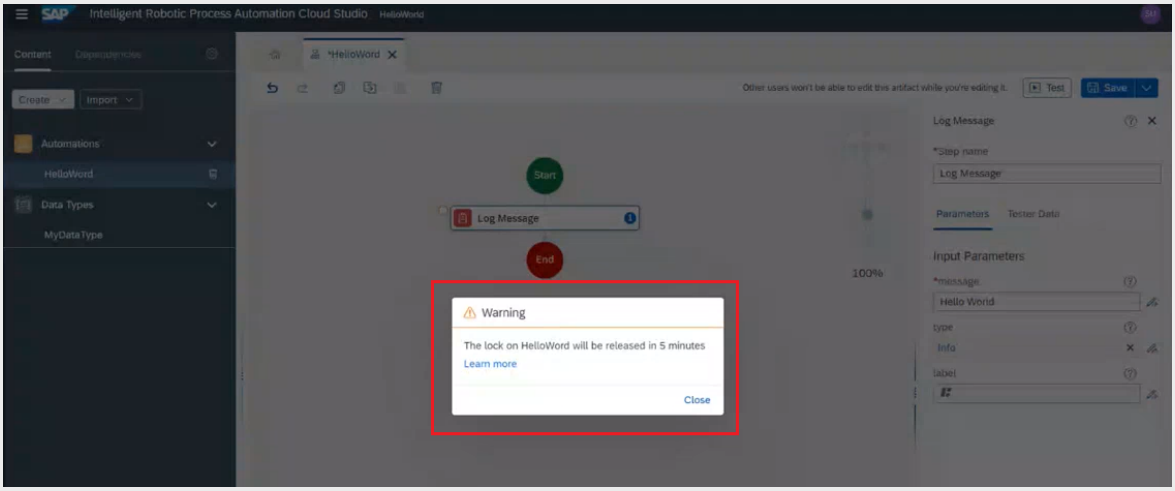
When an artifact is locked by a user, other users can see and open the artifact in read-only mode. The read-only artifact shows the name of a user who locked the artifact. Also, other users can edit different artifacts from a same project in parallel.

> **i Note**
>
> An artifact authoring or editing default timeout is set to 60 minutes. An alert message is displayed 10 minutes before the actual artifact authoring or editing timeout period.



An artifact is locked until you are working on it. If an artifact is inactive for 15 minutes, then the lock is released, and the other users can modify the artifact.

An artifact is locked for 30 minutes. A warning message is displayed 5 minutes before locking an artifact.



If you close the editor or release an artifact, then the lock is released, and the other users can modify the artifact.

9.3 Duplicate Artifact

In the Cloud Studio, you can duplicate artifact one or more times, such as automation, application, datatype, Processes, and User Tasks.

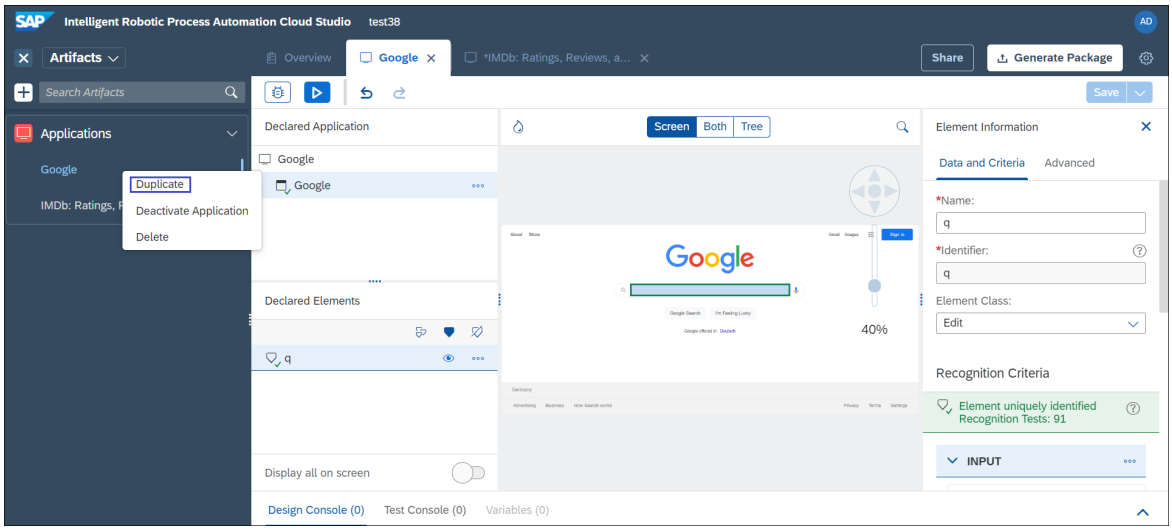
Prerequisites

You have created at least one artifact, such as automation, application, and datatype. For more details about creating an automation, an application, and a datatype, refer to the following sections:

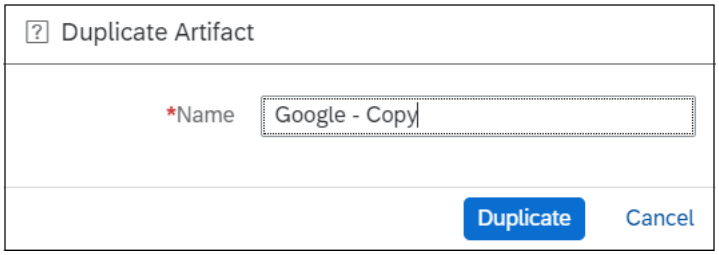
- Capture an Application [page 77]
- Create an Automation [page 156]
- Create a Data Type [page 168]

Duplicate Application Artifact

1. In the main panel of a **Project Explorer**, under *Applications*, select an artifact and then right click.
   The more options are displayed.

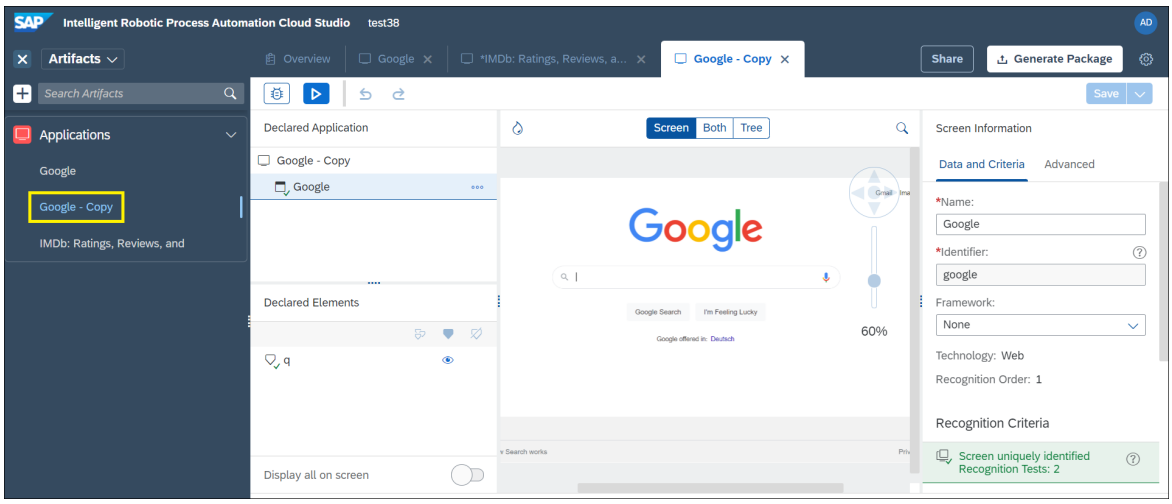


2. Click *Duplicate*.
   The *Duplicate Artifact* pop-up screen is displayed.



> **i Note**
>
> If required, you can update the name of an artifact.

3. Click *Duplicate*.
   The duplicated artifact is created.



Duplicate Automation Artifact

1. In the main panel of a **Project Explorer**, under *Automations*, select an artifact and then right click.
   The more options are displayed.
2. Click *Duplicate*.
   The *Duplicate Artifact* pop-up screen is displayed.

> **i Note**
>
> If required, you can update the name of an artifact.

3. Click *Duplicate*.
   The duplicated artifact is created.

Duplicate Data Type Artifact

1. In the main panel of a **Project Explorer**, under *Data Types*, select an artifact and then right click.
   The more options are displayed.
2. Click *Duplicate*.
   The *Duplicate Artifact* pop-up screen is displayed.

> **i Note**
>
> If required, you can update the name of an artifact.

3. Click *Duplicate*.
   The duplicated artifact is created.

Processes Artifact

1. In the main panel of a **Project Explorer**, under *Processes*, select an artifact and then right click.
   The more options are displayed.
2. Click *Duplicate*.
   The *Duplicate Artifact* pop-up screen is displayed.

> **i Note**
>
> If required, you can update the name of an artifact.

3. Click *Duplicate*.
   The duplicated artifact is created.

User Tasks Artifact

1. In the main panel of a **Project Explorer**, under *User Tasks*, select an artifact and then right click.
   The more options are displayed.
2. Click *Duplicate*.
   The *Duplicate Artifact* pop-up screen is displayed.

   > **i** Note
   >
   > If required, you can update the name of an artifact.

3. Click *Duplicate*.
   The duplicated artifact is created.

9.4 Deactivate an Artifact

In the Cloud Studio, you can deactivate certain artifacts of *Alerts*, *Applications*, *Automations*, *Data Types*, *Files*, *Processes*, *Project Panes*, and *User Tasks*.

> **i** Note
>
> At any point in time, you can re-activate the deactivated artifacts.
>
> The deactivated artifacts are no longer available in the side panel of Cloud Studio for creating the workflow of an automation.
>
> You can click the deactivated artifact to view the details in the automation designer.
>
> If an artifact is used in a workflow and then you deactivate it and try to run the automation, the automation fails and displays an error.

Prerequisites

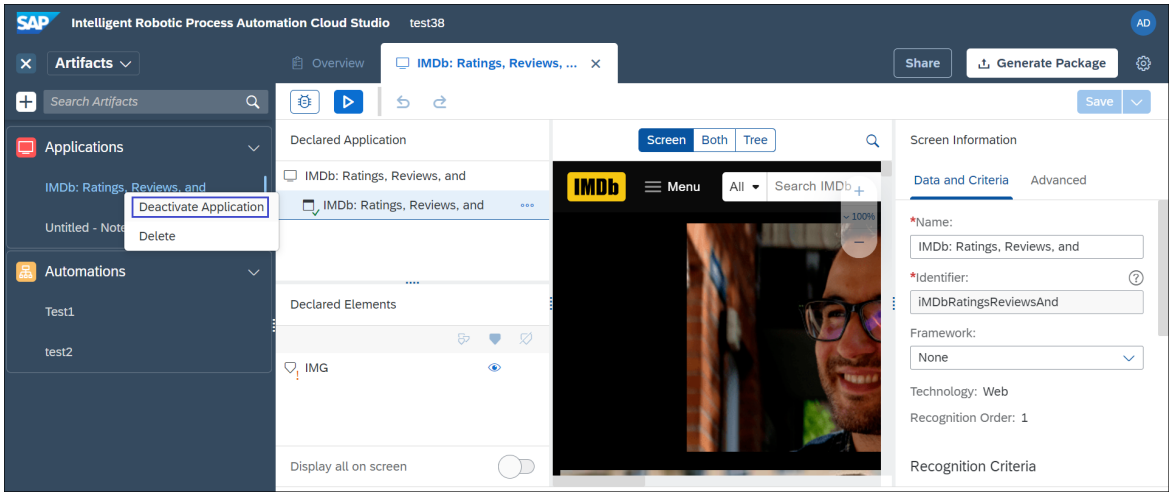
You must create at least one artifact such as *Alerts*, *Applications*, *Automations*, *Data Types*, *Processes*, *Project Panes*, and *User Tasks*. For more details about creating artifacts, refer to the following sections:

- Alerts [page 357]
- Capture an Application [page 77]
- Create an Automation [page 156]
- Create a Data Type [page 168]
- Create a File [page 360]
- Processes [page 482]
- Create a Data Type [page 168]

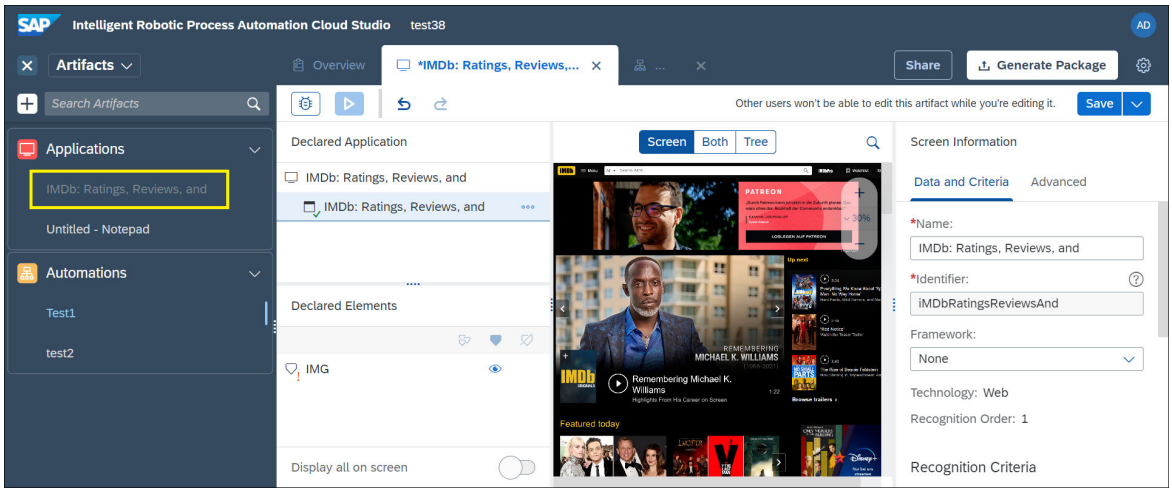
- User Tasks [page 489]

Deactivate an Application

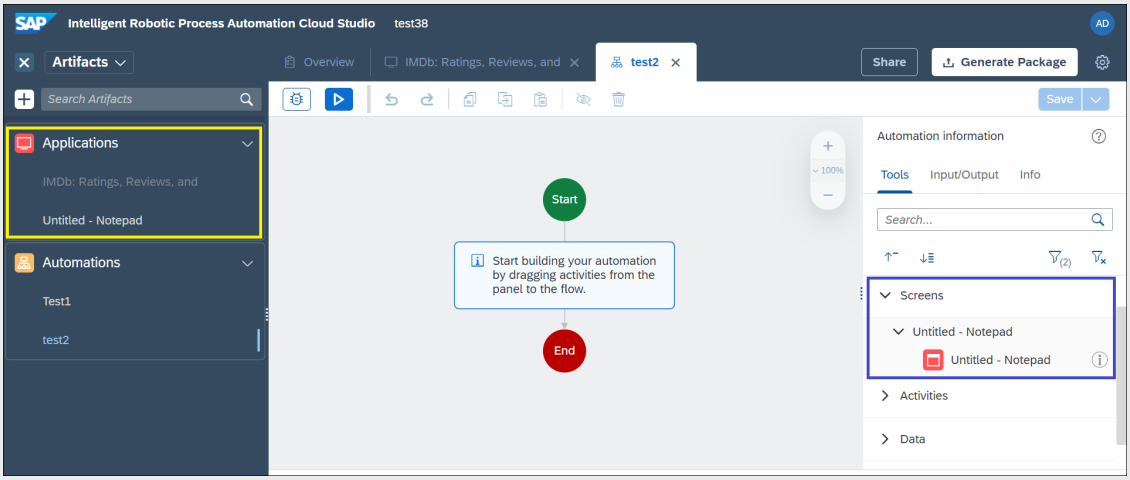
1. In the main panel of a **Project Explorer**, under *Applications*, select an Application and then right-click. More options are displayed.



2. Click *Deactivate Application*.
The application is deactivated.

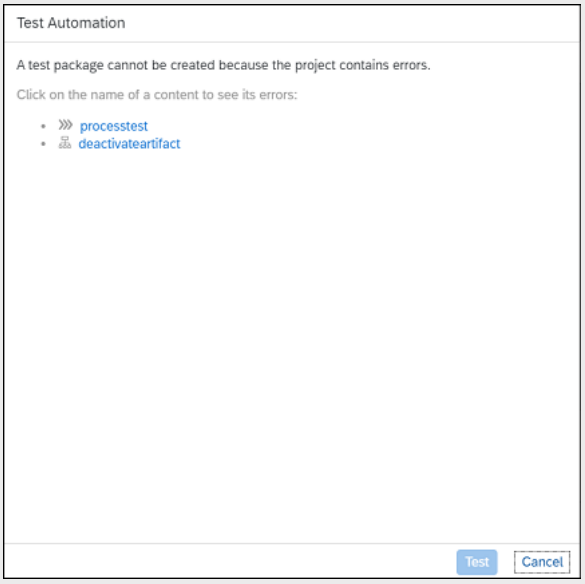


> **i Note**
>
> The deactivated artifacts are no longer available in the side panel of Cloud Studio for creating the workflow of an automation.



You can click the deactivated artifact to view the details in the automation designer.

If an artifact is used in a workflow and then you deactivate it and try to run the automation, the automation fails and displays an error.



Activate an Application

1. Select the deactivated Application and then right-click.
   More options are displayed.
2. Click *Activate Application*.
   The application is activated.

Deactivate an Automation

1. In the main panel of a **Project Explorer**, under *Automations*, select an Automation and then right-click.
   More options are displayed.

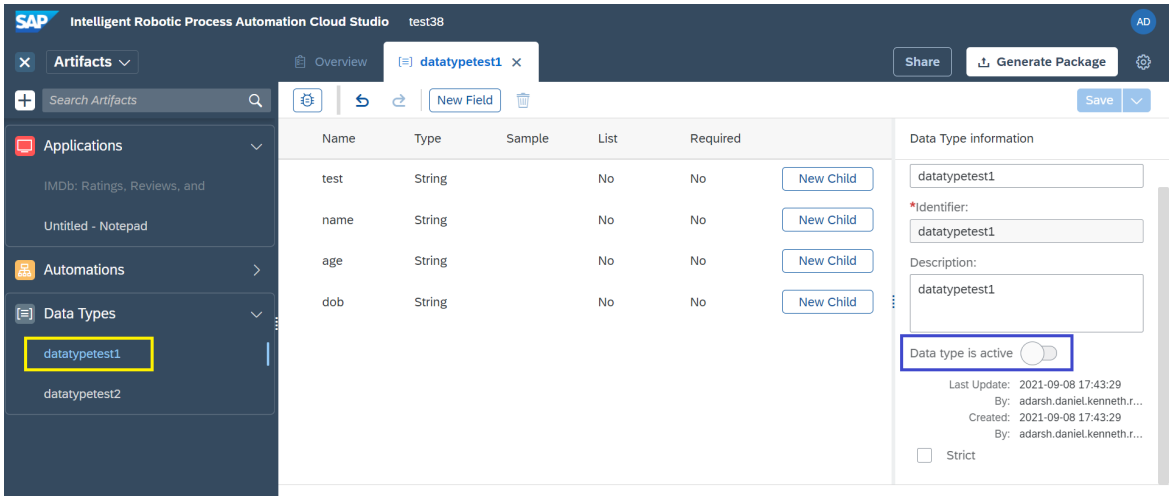
2. Click *Deactivate Automation*.
   The Automation is deactivated.

Activate an Automation

1. Select the deactivated Automation and then right-click.
   More options are displayed.
2. Click *Activate Automation*.
   The Automation is activated.

Deactivate a Data Type

1. In the main panel of a **Project Explorer**, under *Data Types*, select a Data Type and then right-click.
   More options are displayed.
2. Click *Deactivate Data Type*.
   The Data Type is deactivated.



Activate a Data Type

1. Select the deactivated data type and then right-click.
   More options are displayed.
2. Click *Activate Data Type*.
   The Data Type is activated.

> **i Note**
>
> The activate or deactivate option is available under *Data Type information* of a selected Data Type artifact on the right panel of the Cloud Studio. You can activate or deactivate Data Type artifact using the *Data Type is active* toggle button.
>
> To activate or deactivate a Data Type artifact, you must click the *Data Type is active* toggle button and then click the *Save* button. The data type artifact is activated or deactivated.

Deactivate a File

1. In the main panel of a **Project Explorer**, under *Files*, select a File and then right-click.
   More options are displayed.
2. Click *Deactivate File*.
   The File is deactivated.

Activate a File

1. Select the deactivated File and then right-click.
   More options are displayed.
2. Click *Activate File*.
   The File is activated.

> **i** Note
>
> The activate or deactivate option is available under *File information* of a selected file on the right panel of the Cloud Studio. You can activate or deactivate a file using the *File is active* toggle button.
>
> To activate or deactivate a File, you must click the *File is active* toggle button and then click the *Save* button. The File is activated or deactivated.

Deactivate a Process

1. In the main panel of a **Project Explorer**, under *Processes* , select a Process and then right-click.
   More options are displayed.
2. Click *Deactivate Process*.
   The Process is deactivated.

Activate Process

1. Select the deactivated Process and then right-click.
   More options are displayed.
2. Click *Activate Process*.
   The Process is activated.

> **i** Note
>
> The activate or deactivate option is available in the *General Data* tab under the *Process information* on the right panel of the Cloud Studio.
>
> To activate or deactivate a Process, you must click the *Process is active* toggle button and then click the *Save* button. The Process is activated or deactivated.

Deactivate a Project Pane

1. In the main panel of a **Project Explorer**, under *Project Panes* , select a Project Pane and then right-click.
   More options are displayed.
2. Click *Deactivate Project Pane*.
   The Project Pane is deactivated.

Activate a Project Pane

1. Select the deactivated Project Pane and then right-click.
   More options are displayed.
2. Click *Activate Project Pane*.
   The Project Pane is activated.

> **i** Note
>
> The activate or deactivate option is available in the *General Data* tab under the *Project pane information* on the right panel of the Cloud Studio.
>
> To activate or deactivate a Project Pane, you must click the *Project pane is active* toggle button and then click the *Save* button. The Project Pane is activated or deactivated.

Deactivate a User Task

1. In the main panel of a **Project Explorer**, under *User Tasks*, select a User Task and then right-click.
   More options are displayed.
2. Click *Deactivate User Task*.
   The User Task is deactivated.

Activate a User Task

1. Select the deactivated User Task and then right-click.
   More options are displayed.
2. Click *Activate User Task*.
   The User Task is activated.

> **i** Note
>
> The activate or deactivate option is available in the *General Data* tab under the *User tasks information* on the right panel of the Cloud Studio.
>
> To activate or deactivate a User Task, you must click the *User task is active* toggle button and then click the *Save* button. The User Task is activated or deactivated.

Deactivate an Alert

1. In the main panel of a **Project Explorer**, under *Alerts*, select an Alert and then right-click.
   More options are displayed.
2. Click *Deactivate Alert*.
   The Alert is deactivated.

Activate an Alert

1. Select the deactivated Alert and then right-click.
   More options are displayed.
2. Click *Activate Alert*.
   The Alert is activated.

9.5 Glossary

Activity A function that you drag and drop in the workflow of automations or scenarios.

Each activity has parameters and executes an action according to the class or library it belongs to.

For more information, see Automation Activities provided by SDK Packages [page 371].

Agent A local component consisting of a computer system (PC, Desktop or Server) and a user session that executes an automation scenario initiated by SAP Intelligent RPA Factory.

A single computer system can host several agents.

For more information, see About the Desktop Agent.

Alert An artifact that allows you to define business events.

Once an alert is created, it can be raised from an automation.

For more information, see Alerts [page 357].

Application An artifact that captures an application running on your local machine, selects and captures screens, and recognizes UI elements on those screens for use in automation.

An application can be a browser app, a Windows application, or an SAP application.

For more information, see Capture and Declare Applications [page 75].

Automation A succession of steps, built in the Cloud Studio with the automation designer, that can orchestrate multiple activities on different applications and screens available on a specific computer.

Design Automations [page 153].

Automation End A control that ends an automation in a valid state before the end of the workflow.

Automation End [page 243].

Breakpoint An option that you can set to stop the execution of the tester at one step of an automation or a scenario.

Test an Automation [page 282].

Collection A list of elements of the same type (for example, a list of titles, URLs, summaries, etc.).

Once you have captured a screen, you can declare an element as a collection using the *Is a Collection* button.

For more information, see Declare an Element as a Collection [page 120].

Condition A control that allows you to insert multiple situations, defined by an expression, that determine the following step in an automation flow.

For more information, see Condition [page 231].

Connector A component that allows you to identify and control the applications you want to automate depending on their technology (web, UI automation, SAP GUI, SAPUI5, etc.).

For more information, see Technology Connectors [page 132].

Console A panel that displays every error, warning, variables data, and information related to your automation.

For more information, see Monitor an Automation Using the Console [page 267].

Control A specific type of step that allows you to add tools such as conditions, loops, and scripts to the workflow of your automation.

For more information, see Add a Control to an Automation [page 216].

Custom Scripting An activity that you can define with custom JavaScript code, and insert as a step in the workflow of your automation.

For more information, see Custom Scripting [page 251].

Data variable Data storage that has a name, a type, and a value and that you use to build your automation.

For more information, see Create and Update Variables [page 179].

Data type An artifact describing a data structure that you can use as an input and/or output parameter in automations or processes.

For more information, see Data Types [page 168].

Dependency A package that you import in a project and that contains artifacts such as automations, applications, data types and activities that you can use to build your automation.

For more information, see Dependencies [page 525].

Decommission An action that automatically deletes a package if it is not referenced in a project or by another package.

You can only decommission preview or released packages.

For more information, see Decommission a Dependency [page 533].

Element The declared component (such as a button, an input field, or a control) of a captured screen that you use to build automations.

For more information, see Declare Elements [page 107].

For Each A control that allows you to go through a list of members provided as input to your automation and execute an action for each member in that list.

For more information, see For Each [page 217].

Forever A control that allows you to repeat, in an automation flow, a step in a loop until it meets the required conditions.

For more information, see Forever [page 224].

Input Parameter A variable that is passed, received, or sent from one automation.

An input parameter allows you to exchange data in the workflow of your automation between activities, screens, and scripts.

For more information, see Input/Output Parameters [page 160].

Loop End A control that allows you to exit a loop before its original end.

For more information, see Loop End [page 230].

Must exist An element that means that the screen is not recognized unless the elements defined as **Must exist** are detected.

For more information, see Declare a Screen [page 104].

Must not exit An element that means that the screen is not recognized if an element defined as **Must no exist** is detected.

For more information, see Declare a Screen [page 104].

Output Parameter A variable that is passed, received, or sent from one automation.

An output parameter allows you to exchange data in the workflow of your automation between activities, screens, and scripts.

For more information, see Input/Output Parameters [page 160].

Package A project that is imported to SAP Intelligent RPA and configured to execute one or more automations, processes, or scenarios.

For more information, see Reuse a Package from the Store [page 507].

Process A sequence of skills (imported scenarios, user tasks, or other processes) designed to automate one or more tasks.

Processes are built on SAP Workflow, run in the cloud and can involve multiple systems.

For more information, see Processes [page 482].

Project A set of one or more automated processes or scenarios and associated resources (user tasks, data types, files).

For more information, see Create a Project [page 6].

Project Pane A feature that allows you to design the agent systray menu directly from the Cloud Studio.

For more information, see Trigger Automations in Attended Mode using the Project Pane [page 352].

Recorder A tool that records the steps you perform across the screens of an application.

The Recorder captures applications and designs automations accurately at the same time.

For more information, see Design Automations Quickly with the Recorder [page 38].

Repeat A control that allows you, in an automation flow, to define a loop that will repeat a step for a defined number of times.

For more information, see Repeat [page 226].

Screen Switch A control that handles multiple situations of a task, defined by which screen we are currently on and determines the next step of an automation.

For more information, see Screen Switch [page 246].

SDK Package A package that contains the main methods and functions from the application it comes from and that you use to build and run your automation projects.

For more information, see SDK Packages [page 372].

Step A fundamental unit of a workflow composing an automation or a scenario.

For more information, see Design Automations [page 153].

Stop Automation in Error A control that allows you to raise a business error that generates a customized input mssage in the logs, when your automation is failing.

For more information, see Generate a Business Error with the Stop Automation in Error control [page 336].

Store A web application where you can find automation packages.

For more information, see Reuse a Package from the Store [page 507].

Tester A panel that helps you test, troubleshoot, and display the execution flow of your automation.

For more information, see Test an Automation [page 282].

Timeline The user interface of the tester panel in the Cloud Studio that shows the event logs of an automation.

For more information, see Display Errors in the Timeline [page 340].

Try (or Try Catch)	A control that allows you to catch errors raised on specific steps of the workflow and handle them on a side branch of the automation. For more information, see Try Catch [page 239].
User Task	A type of skill that you can use during the design of a process. A user task takes the form of a message that will be sent to the inbox of a user during the running phase of a process. For more information, see User Tasks [page 489].
Workflow	A sequence of steps forming the visual interface of an automation.

Important Disclaimers and Legal Information

Hyperlinks

Some links are classified by an icon and/or a mouseover text. These links provide additional information.
About the icons:

- Links with the icon : You are entering a Web site that is not hosted by SAP. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) to this:
  - The content of the linked-to site is not SAP documentation. You may not infer any product claims against SAP based on this information.
  - SAP does not agree or disagree with the content on the linked-to site, nor does SAP warrant the availability and correctness. SAP shall not be liable for any damages caused by the use of such content unless damages have been caused by SAP's gross negligence or willful misconduct.
- Links with the icon : You are leaving the documentation for that particular SAP product or service and are entering a SAP-hosted Web site. By using such links, you agree that (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

Videos Hosted on External Platforms

Some videos may point to third-party video hosting platforms. SAP cannot guarantee the future availability of videos stored on these platforms. Furthermore, any advertisements or other content hosted on these platforms (for example, suggested videos or by navigating to other videos hosted on the same site), are not within the control or responsibility of SAP.

Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.
The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phrasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP's gross negligence or willful misconduct.

Bias-Free Language

SAP supports a culture of diversity and inclusion. Whenever possible, we use unbiased language in our documentation to refer to people of all cultures, ethnicities, genders, and abilities.

www.sap.com/contactsap

© 2021 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.

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.

Please see https://www.sap.com/about/legal/trademark.html for additional trademark information and notices.

THE BEST RUN SAP