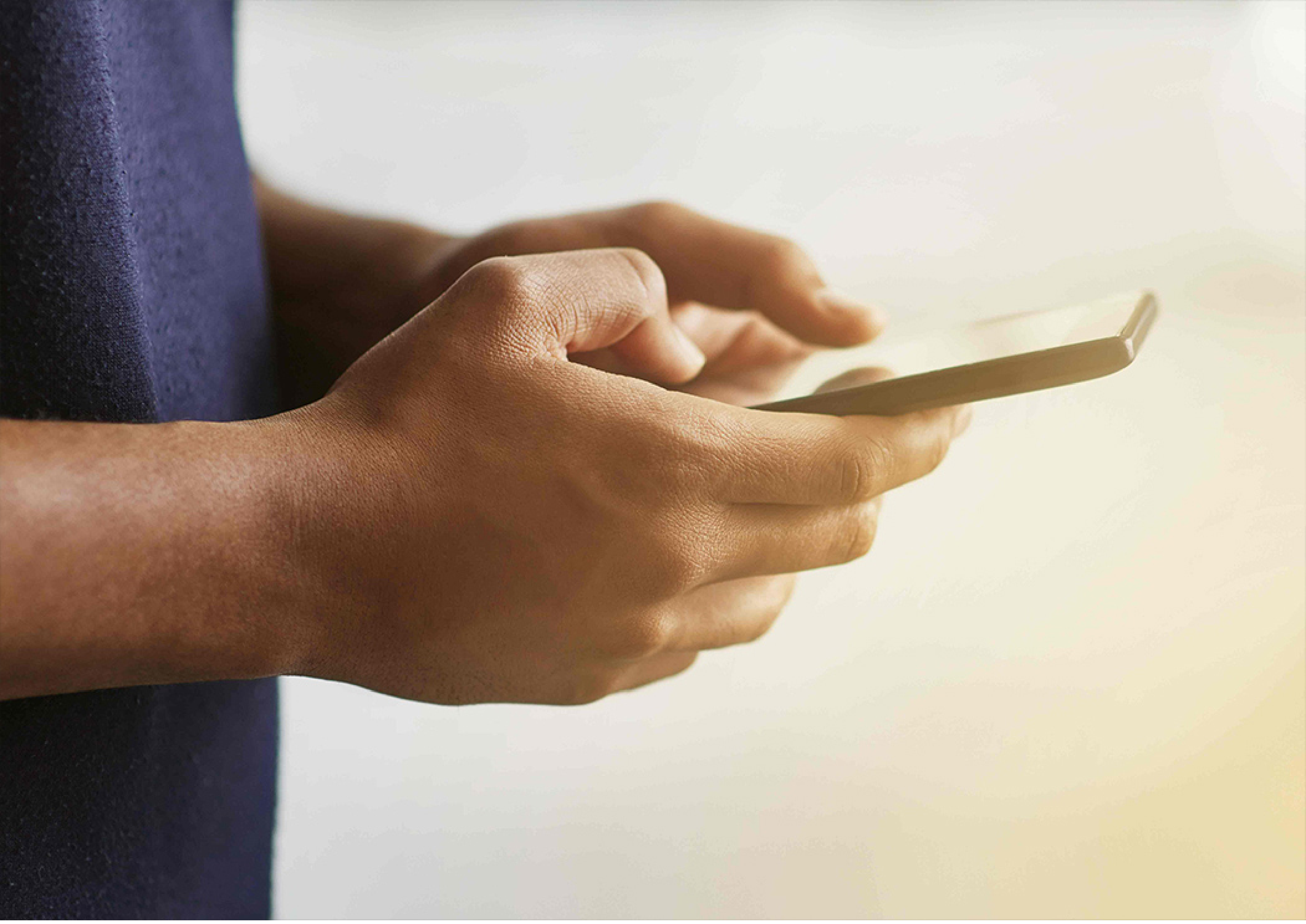


PUBLIC
2024-04-04

# Troubleshooting

# Content

# 1 Troubleshooting and Support

Welcome to the Troubleshooting and Support topic. In this topic we list several Knowledge Base notes available in https://support.sap.com/ or directly by setting the component.

If you experience issues when using SAP Intelligent RPA, we suggest you raise a request through the SAP Support Portal. To get more information on how to create a new incident, see Getting Support.

**By Component**

- Desktop Agent related notes [page 3]
- Desktop Studio related notes [page 11]
- Cloud Factory related notes [page 48]

**Cross-Component and Other Issues**

- Why can't I see the SAP Intelligent RPA product in the SAP Marketplace ? 2824501
- Web Browsers Configuration Checks - 2796986
- Where can I find pre-built content for S/4HANA ? - Content is available on rapid.sap.com. See this note 2788986 for release strategy.

**Installation and Setup**

- Technical Prerequisites and System Requirements
- SAP Intelligent Robotic Process Automation Factory Configuration
- On-premise Components Setup
- Updating your SAP Intelligent RPA 1.0 Landscape (Desktop Agent and Studio)
- Configuring Web Browsers

## 1.1 Desktop Agent

- How to prevent connection issues: 3155610
- Agent XXX already connected" error when attempting to login to the Desktop Agent - 2796483
- Nothing happens when launching an unattended scenario using SAP Intelligent RPA Desktop Agent - 2796491
- Desktop Agent icon does not appear in the Windows system tray - 2796377
- When downloading a Package from the Factory an error occurs "CCtxtRun2Dlg::OnGoToSocket : SetIpaSocket failed" - 2798996
- No Registry key or bad value & Can't find a json file - 2796396
- Problem to restart the Agent (new window session, switching project or new project from Factory). To enable the restart of the Agent, the service "SAP Intelligent RPA Service" (CxAgent.exe) must be up and running.

- Need to investigate Agent and Factory connection: download the events history of an agent in the SAP Intelligent RPA Factory. For more information, see Overview of Agents.
- How to fix agent connection issues happening during Cloud Factory update: 2981018
- How to hide machine or login information: 3056143

## 1.1.1 Activate Support Mode

Desktop agent 3 features a support mode. You can use the support mode to record traces or install a specific version of the agent.

### Prerequisites

- You must have the role IRPAOfficer.
- If you want to install a specific version of the agent after enabling support mode, the agent must be in attended mode. For more information, see Change the Agent Mode.

### Procedure

1. In SAP IRPA, choose the *Agents* tab.
2. Search for the relevant agent and choose ••• *More Actions* and *Activate Support*.
3. Choose a duration between 1 hour and 5 days and *Activate*.
4. Optionally, you can further configure the tracer settings including tracer components in the agent under *Settings* and *Tracer*.

### Results

- The support mode is activated for the specified time.
  You can *Deactivate Support* mode manually by following step 2.
- Automatic updates are enabled.
  You can disable automatic updates in the desktop agent by choosing *System Settings* and unchecking *Enable automatic updates*.

  > ⓘ Note
  >
  > When support mode is deactivated, *Enable automatic updates* is not visible.
- Functional traces are recorded per job, technical traces are recorded at startup, and traces are encrypted and uploaded to the tenant.
  In the desktop agent, you can access *Tracer* options under *Settings*, while support mode is activated.

## 1.1.2 Export Health Metrics

Export health metrics for your agent.

### Prerequisites

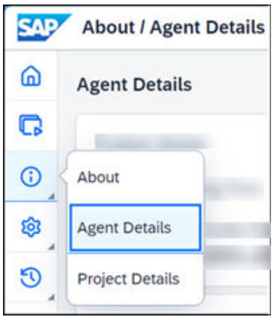
You need to have activated support mode to be able to export health metrics. For more information, see Activate Support Mode [page 4].

### Context

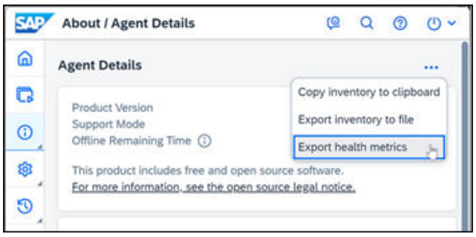
You can export metrics relating to your agent and send these to the SAP support team for investigation. Once the support mode is activated, the health metrics are stored for a maximum of 7 days.

### Procedure

1. In the desktop agent, choose ⓘ *About* and *Agent Details*.



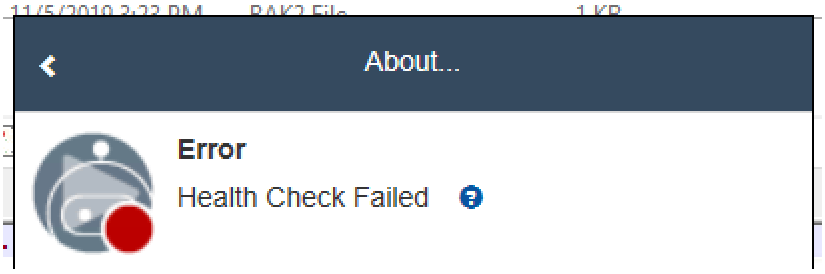
2. Choose ••• *More* and *Export health metrics*


About / Agent Details
Agent Details
Product Version
Support Mode
Offline Remaining Time
Copy inventory to clipboard
Export inventory to file
Export health metrics
This product includes free and open source software.
For more information, see the open source legal notice.

### Results

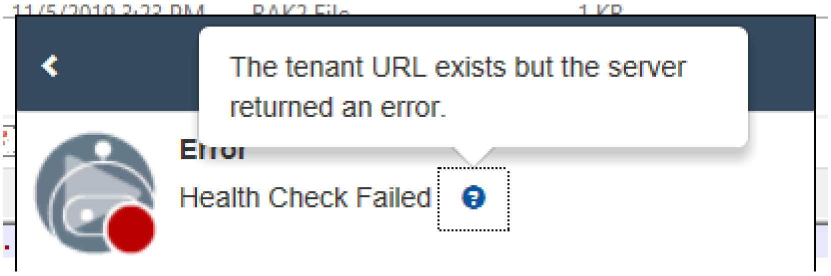
A zip file containing CSV files for the different metrics is exported to your PC. You can send this file to SAP Support.

## 1.1.3 Desktop Agent Error Messages



Error

Health Check Failed



The tenant URL exists but the server returned an error.

Error

Health Check Failed

Within the Desktop Agent systray, status, error and warning are reported in the About screen. Warning icon (orange) would not require a human interaction because the Agent will try again. Error icon (red) often requires a human interaction.

Please find below some common errors that could happen:

- Health Check Failed (HEALTHCHECK_FAILURE) - Tenant URL exists but an error has been returned by server
  - Cause: The URL exists, but the Agent is not able to connect.
  - Resolution:
    - Check the URL with a web browser.
    - Contact your administrator to have the correct URL
    - Check the connectivity

    If you are using a VPN, this issue may come from the fact that your VPN doesn't allow Websockets. For more information, see the **About WebSockets protocol** section at the bottom of this page: Technical Prerequisites and System Requirements.
- Tenant URL server is not accessible (RETRY_HEALTHCHECK_ON_FAILURE)
  - Cause: The URL is no longer available
  - Resolution:
    - Connectivity is broken : Restart the network interface, check IP and DNS.
    - Wrong URL : Contact your administrator to have the correct URL
- Authentication not completed (AUTHENTICATION_NOT_COMPLETED)

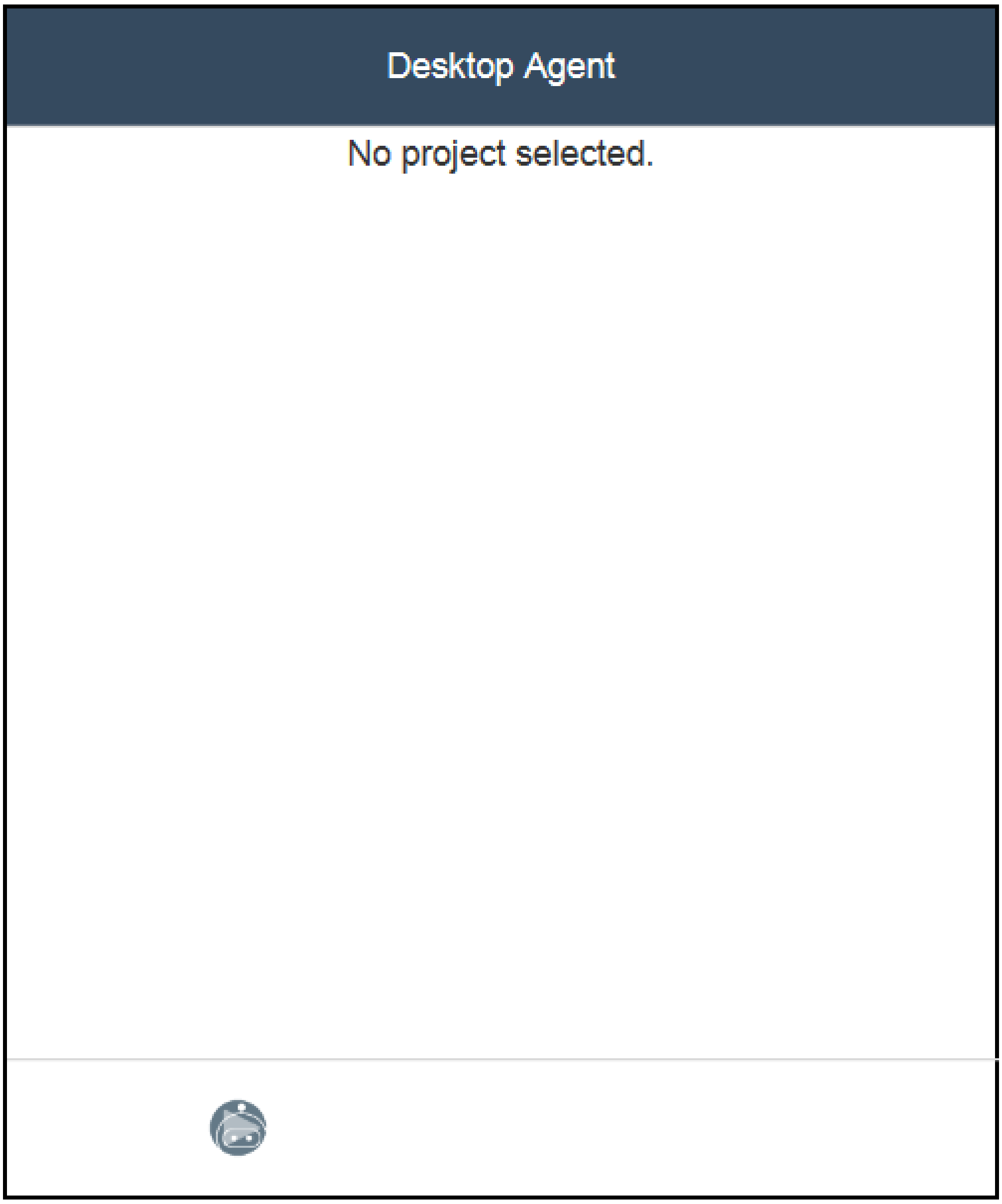
    - Cause: Authentication was not fully completed
    - Resolution: Restart the Agent in order to launch the authentication again
- Connection has been lost (NEED_SENDING_CLIENT_INFO)
    - Cause: Cable unplug, WiFi disconnection...
    - Resolution: Check your connection, when it will be up Agent will reconnect automatically
- Agent not found in agent group ( INVALID_SEND_CLIENT_INFO_ANSWER)
    - Cause: Your Agent is not declared in Cloud Factory
    - Resolution: Contact the Administrator for adding your Agent in an agent group (Agents > Agent groups)
      For more information see Create an Agent Group.
- Connection is closed forever
    - Cause: Communication error between the Desktop Agent and the Factory
    - Resolution: The Desktop Agent must be updated. The minimum version needed to resolve this issue is the 2011 version 2.0.6. For more information see this SAP note 2997097.
- Package not found in the systray
    - Cause: Your package does not appear in the systray
    - Resolution: Go to *Scripts* and look for: `GLOBAL.events.START.on(function (ev)`. If you see a snippet like the one shown below, comment the if condition:

      Sample Code

      ```
      GLOBAL.events.START.on(function (ev){
          if(ctx.options.isDebug){
          //Add item in systray menu.
      ....
      }
      });
      ```
- Waiting for user to enter credentials
    - Cause: The Desktop Agent waits for the user to enter their credentials
    - Resolution: Shutdown the Desktop Agent and start again, it will prompt you to sign in. If you are still having issues:
        1. Open the tenant URL in Internet Explorer, log in and save the user id and password.
        2. Select *Tools > Internet Options > Security*.
        3. Click *Internet > Custom level...*.
        4. Under *Settings > Scripting*, set *Active scripting* to *Enable*.
        5. Go back to *Internet options > Security*. Click *Trusted sites* and then *Sites*.
        6. Enter the tenant URL and click *Add* to include it in the list of trusted sites.

      Now you should be able to see the pop-up indicating that the Desktop Agent is getting logged in using the credentials that have been stored. It is now able to get online.
- Registration failed
    - Cause: Registration has failed
    - Resolution: Make sure you use the correct URL by reviewing the tenant registration steps.
      For more information, see Desktop Agent Tenant Registration.

## 1.1.4 Desktop Agent Blank Systray



Desktop Agent

No project selected.

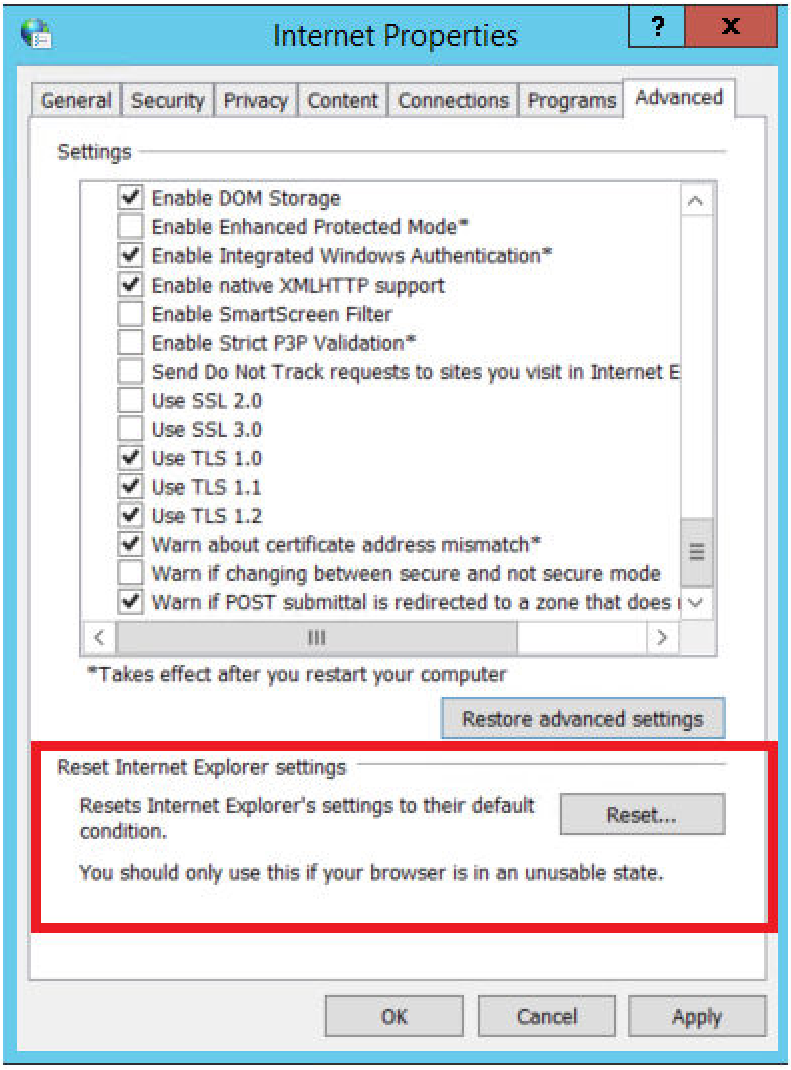
You have checked all the technical prerequisites and configured your browser, but the agent systray appears blank.

**Cause**

Internet Explorer is in an unstable state.

**Resolution**

1. Reset Internet Explorer settings.



Settings

- ☑ Enable DOM Storage
- □ Enable Enhanced Protected Mode*
- ☑ Enable Integrated Windows Authentication*
- ☑ Enable native XMLHTTP support
- □ Enable SmartScreen Filter
- □ Enable Strict P3P Validation*
- □ Send Do Not Track requests to sites you visit in Internet E
- □ Use SSL 2.0
- □ Use SSL 3.0
- ☑ Use TLS 1.0
- ☑ Use TLS 1.1
- ☑ Use TLS 1.2
- ☑ Warn about certificate address mismatch*
- □ Warn if changing between secure and not secure mode
- ☑ Warn if POST submittal is redirected to a zone that does i

*Takes effect after you restart your computer

Reset Internet Explorer settings

Resets Internet Explorer's settings to their default condition.

You should only use this if your browser is in an unusable state.

2. Configure Internet Explorer settings again. For details, see Configuring Web Browsers

## 1.1.5 Using the PC while a Bot is Running

You may want to use your PC while a bot is running.

> ⓘ Note
>
> If your bot does not require human intervention, you should not use your PC while the bot is running.

Running a bot in background is possible, however you should take into account the following information regarding the design of your bot:

- It should not contain any **mouse click activities**, **double click activities** and other functions which require mouse interactions (**normal click activity** is working);
- It should not contain any **keystroke activities**;
- You can use your PC if your workflow does not contain any activities that require the opening of an application or a window;

- You can minimize the application to automate depending on its technology;
- If you interact with the same application as the bot, there is an important risk of conflicts (even if the application is minimized).

> ⓘ Note
>
> If you want to use your PC while a bot is running, the best thing to do is to do some tests. Indeed, the behavior of the bot depends widely on the use case.

## 1.1.6 System Reboot during Unattended Mode

The system may reboot while the Desktop Agent is in unattended mode.

### Context

When a Desktop Agent is running on a system in unattended mode, it may happen that the system reboots. You may want to start automatically the Agent.

### How to start the Desktop Agent immediately after the system reboot

In order to start the Desktop Agent immediately after this reboot, you must have stored your Windows credentials in the Desktop Agent. You must also have checked the *Automatically Start at Windows Logon* checkbox in the Desktop Agent systray. To check this option, click the *More Actions* button and then click *Settings*.

The session opens in non-interactive mode when the system has rebooted. Then, the Agent starts automatically.

Fore more information on how to store Windows credentials for lauching SAP Intelligent RPA Desktop Agent in unattended mode, see Set Windows Password in Desktop Agent.

### How to remove the credentials

The credentials can be removed using the following command:

```
CxStoreCred.exe /forget
```

With this command, the Desktop Agent stops restarting automatically after the reboot.

For more information on how to launch the Desktop Agent when the server restarts, check the following SAP Note: 3010368.

## 1.1.7 Memory Allocation Limit Reached

Out-of-memory errors occur when too much data is processed at once or when an automation does not release memory after use. This can cause an automation crash. Follow the recommendations to avoid reaching the memory allocation limit.

We recommend that you:

- Follow Best Practices for Custom Scripts.
- Process or retrieve large data sets in smaller subsets to process them by batch. Do not load the entire set of data at once. For example, instead of reading an entire Excel file at once, you can loop it and process X lines by X lines.
- Free unused data in memory. For example, a variable containing a list of hundreds of objects can be deleted when it is not used anymore.
- Regularly review memory usage to identify abnormal consumption. On a Windows machine, you can review the consumption of the memory using the Task Manager application.

If the out-of-memory error still occurs after following the recommendations, try to execute the task which requires a heavy memory load using an external tool, such as a Python script.

# 1.2 Desktop Studio

**The following Knowledge Base notes relating to the Desktop Studio are available:**

- Long waiting time when capturing a page using UI Automation - 2788781
- Unable to create Process Debug Manager component - 2781601
- "System.InvalidOperationException" error when debugging a project in the Desktop Studio - 2833621
- Black screen when capturing a page using Intelligent RPA Desktop Studio with Chrome - 2842523
- Could not find file [...] form.pscm - 2849305

### How-tos for Desktop Studio

- How to retrieve variable (Text) from the Factory - 2822776 (See also this topic in the Developer Guide)
- How to retrieve variable (Credentials) from the Factory - 2822500 (See also this topic in the Developer Guide)
- How to insert a picture in an email - 2864114
- How to use another Diff/Merge tool - 2867241
- How to Rename a Project - 2956757

### Best Practices

- Best Practices for Screen Lock and Disconnected Virtual Desktop Infrastructure [page 12]
- Best Practices for PDF Library [page 20]
- Best Practices for Web Service Calls [page 21]
- Best Practices to Handle Important Amount of Data During Scenario Execution [page 31]
- Best Practices for Captchas [page 31]

- Best Practices for Automating SAP GUI Screens with Low Hierarchies [page 32]
- Best Practices for Managing Files [page 34]
- Best Practices for managing Server Busy Pop-up [page 39]

## 1.2.1 Best Practices for Screen Lock and Disconnected Virtual Desktop Infrastructure

The SAP Intelligent RPA tool sometimes requires the user to handle applications with low-level actions such as mouse clicks or direct keystrokes. In certain circumstances, these low-level actions can cause limitations.

This behavior isn't limited to unattended execution, as the user can be expected to launch attended scenarios on their machine and let it work while the screen is locked.

### Context

The user confirms that a job is properly executed during the implementation and test stages:

- With a debugger and tester tool.
- By supervising the job execution in real time on their machine.

However, as soon as the agent is on a Virtual Desktop Infrastructure and with a locked screen, the job isn't executed properly and stops running during some technical actions, in the middle of the automation.

This issue occurs while:

- The screen is locked on the user’s machine, during the execution of a job.
- A session is open on the Virtual Desktop Infrastructure but there is no activity from the user.
- The Virtual Desktop Infrastructure is minimized after a few dozen minutes.

### Reason

To process specific automation actions such as mouse clicks and keystroke sequences, the user needs an open session that prevents the screen from locking during the execution of an automation. But the use of a physical or virtual keyboard isn't necessary.

This behaviour is characteristic of the technologies and applications that are manipulated within the automation.

This kind of issue mostly comes from simulations of using a mouse and a keyboard:

- `application.page.item.mouseClick();`
- `application.page.item.keystroke("text");`

## 1.2.1.1 Causes

The following topics explain the main causes for the screen lock and Virtual Desktop Infrastructure limitations.

### 1.2.1.1.1 Mouse Click

Using the following simulated mouse click actions causes limitations:

- `application.page.item.click(true)`
- `application.page.item.clickMouse(`
- every `ctx.mouse` methods

To identify this kind of action, check if the cursor moves to the item position when calling the click action.

### 1.2.1.1.2 Keystrokes

Using the following keyboard actions causes limitations:

- `application.page.item.keyStroke(e.key.xxx)`
- `application.page.item.keyStroke("Test")`
- `ctx.keyStroke("Test")`

To check if the action can be performed with a locked screen or a disconnected Virtual Desktop Infrastructure, you can send the following keystroke sequence:

```
ctx.wait(function(ev){
application.page.item.keystroke(“text”);
},10000);
```

This keystroke sequence will create a delay during which you can manually lock the screen or disconnect the session.

### 1.2.1.1.3 UI Automation

The UIAutomation connector helps automate any kind of desktop application that implements the UIAutomation interface. If an automated component does not respond to a higher level action request, the UIAutomation connector sends mouse clicks or keystrokes. In this case, simulated actions such as `click()` or `set()` can cause limitations.

For example, in Acrobat Adobe Reader, if you perform a click on *File menu* with the sequence `application.page.item.click()` instead of the sequence `application.page.item.clickMouse()`, the action will fail when the screen is locked.

The same behaviour can occur if you manage *Windows File Browser* with the UIAutomation connector.

To identify this kind of action:

- Check if the cursor moves to the item position when calling the click action.
- Combine the action with a delay during which you can manually lock the screen or disconnect the session:

```
ctx.wait(function(ev){
application.page.item.click();
},10000);
```

### 1.2.1.1.4 Surface Automation with OCR

Surface automation relies on mouse clicks and keystrokes that require a desktop with an unlocked screen.

## 1.2.1.2 Resolutions

The following topics suggest best practices to resolve the screen lock and Virtual Desktop Infrastructure limitations.

Use Unlock Activity [page 14]

Track Mouse Click and Keystroke Actions [page 15]

Web Applications Automation [page 15]

Keep Screen Unlocked [page 19]

Keep a Connection to the Virtual Desktop Infrastructure [page 19]

Identify Another Functional Resolution [page 20]

### 1.2.1.2.1 Use Unlock Activity

If your bot is running in unattended mode, you can use the *Unlock* activity to resolve issues with mouse clicks and keystroke sequences.

> ⓘ Note
>
> For SAP GUI technology, the *Unlock* activity (or
>
> ```
> ensureUnlocked
> ```
>
> function) isn't required for keystroke sequences.

You can add the *Unlock* activity in Desktop Studio or use the following function:

```
ctx.workstation.ensureUnlocked();
```

The *Unlock* activity unlocks a locked session and allows you to perform mouse clicks or keystroke sequences during the execution of an automation. This activity automatically relocks the session 10 seconds after the action.

You must configure the password of the session to:

- Automatically open the session when your machine starts and without human intervention.
- Unlock the session during a few seconds when the bot is executing an automation.

You can directly store your Windows credentials in the Desktop Agent. For more information on how to set Windows credentials in the Desktop Agent, see Set Windows Password in Desktop Agent.

For more information on how to store Windows credentials for lauching a Desktop Agent in unattended mode, see 3010368.

## 1.2.1.2.2 Track Mouse Click and Keystroke Actions

To resolve issues with mouse click and keystroke actions, you need to track down and identify those actions. These actions may have been implemented in the project that way, to avoid complicated code or overcome improper item handling by the tool.

If you find the following keystroke action:

-
  ```
  ctx.keystroke / application.page.item.keystroke
  ```

you can replace it with a simulated event:

-
  ```
  application.page.item.set()
  ```

If you find the following click action:

-
  ```
  ctx.mouse.click / application.page.item.clickMouse / click(true)
  ```

you can replace it with a simulated event:

-
  ```
  application.page.item.click()
  ```

## 1.2.1.2.3 Web Applications Automation

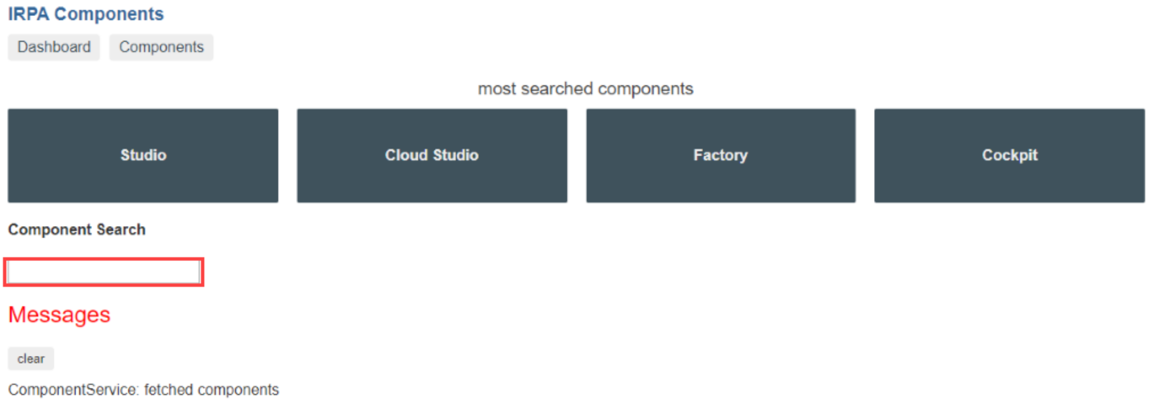
Setting a value with the standard set method may not work on every website, especially on a website with rich framework (e.g. Bootstrap, Angular ...).

This behavior is related to:

- Javascript events attached to items on the web page
- surface controls
- a complex page structure

The following example displays a standard case for a website with rich framework.

1. When setting a value, the user gets a tooltip list as input in the "Component Search" search bar.



IRPA Components

Component Search

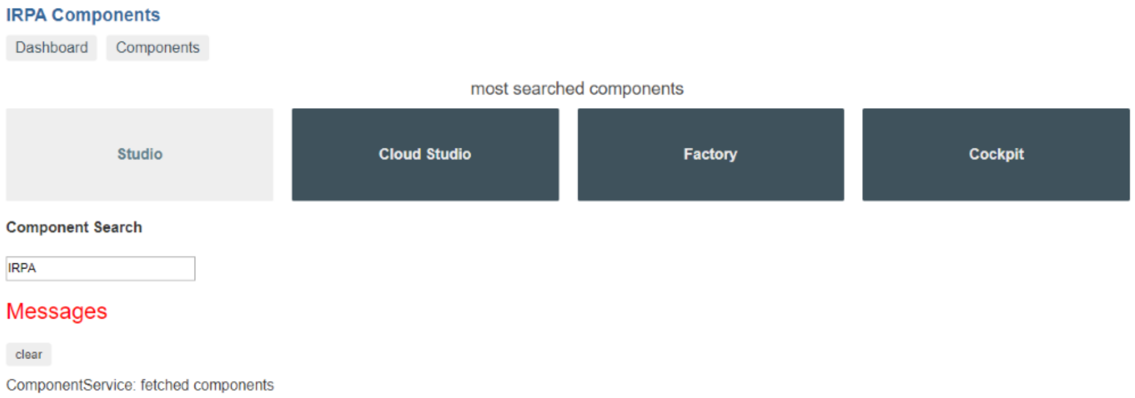
Messages

ComponentService: fetched components

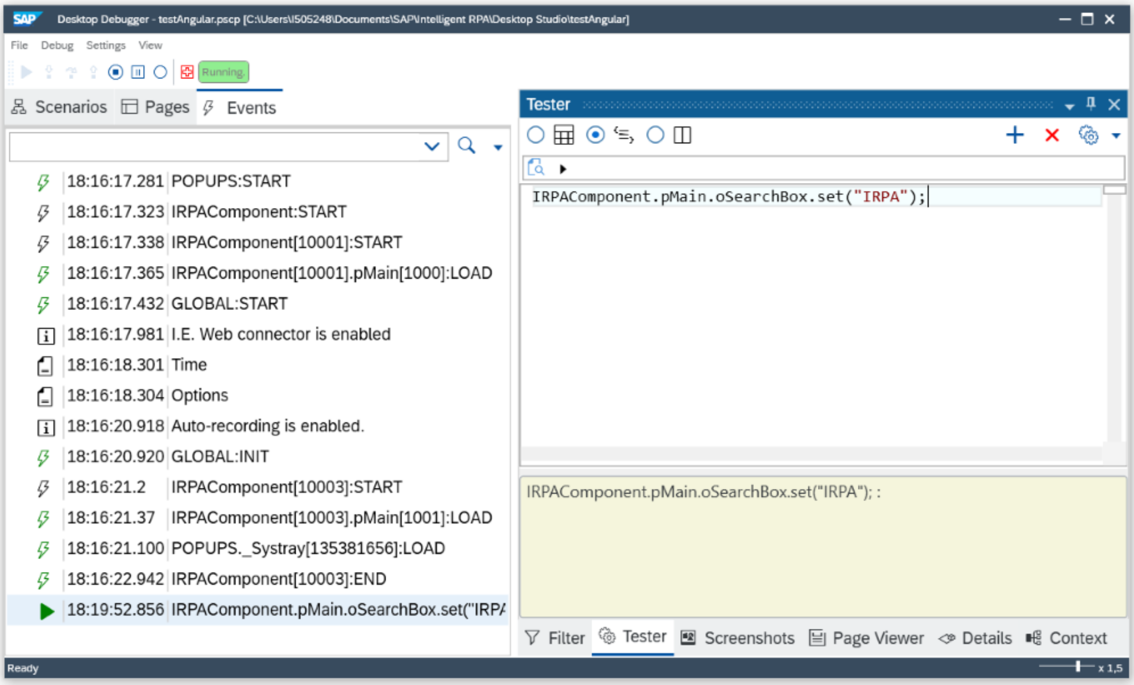
   A new message appears on the page to indicate the kind of search that has been performed and if an unknown value has been set.

2. The user sets the following simulated set event:

```
IRPAComponent.pMain.oSearchBox.set("IRPA");
```



ComponentService: fetched components



```
18:16:17.281 POPUPS:START
18:16:17.323 IRPAComponent:START
18:16:17.338 IRPAComponent[10001]:START
18:16:17.365 IRPAComponent[10001].pMain[1000]:LOAD
18:16:17.432 GLOBAL:START
18:16:17.981 I.E. Web connector is enabled
18:16:18.301 Time
18:16:18.304 Options
18:16:20.918 Auto-recording is enabled.
18:16:20.920 GLOBAL:INIT
18:16:21.2   IRPAComponent[10003]:START
18:16:21.37  IRPAComponent[10003].pMain[1001]:LOAD
18:16:21.100 POPUPS._Systray[135381656]:LOAD
18:16:22.942 IRPAComponent[10003]:END
18:19:52.856 IRPAComponent.pMain.oSearchBox.set("IRP/
```

```
IRPAComponent.pMain.oSearchBox.set("IRPA");
```

```
IRPAComponent.pMain.oSearchBox.set("IRPA"); :
```

The set event does not work for the following reasons:

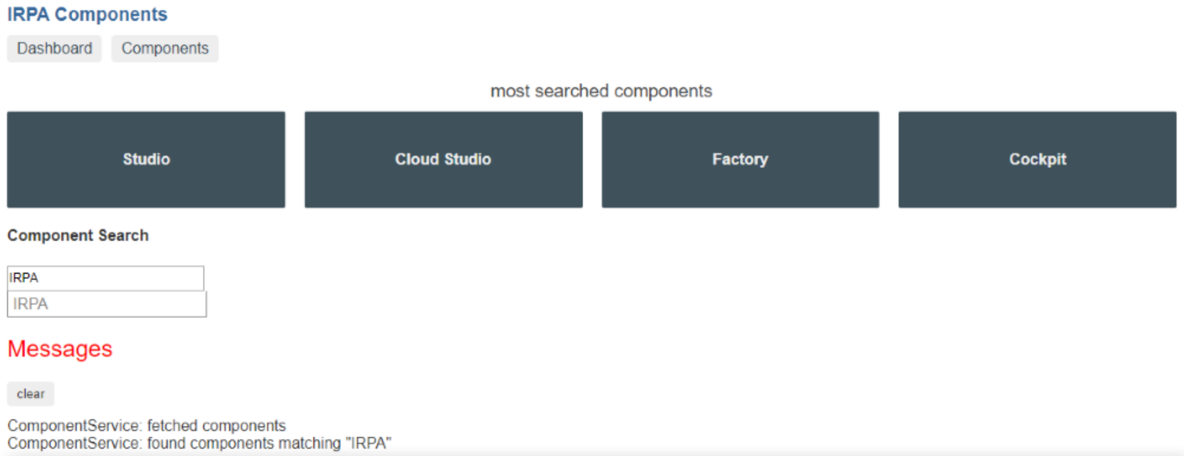
- No tooltip is displayed.
- No new message is displayed.

3. The user sets the following simulated keystroke sequences:

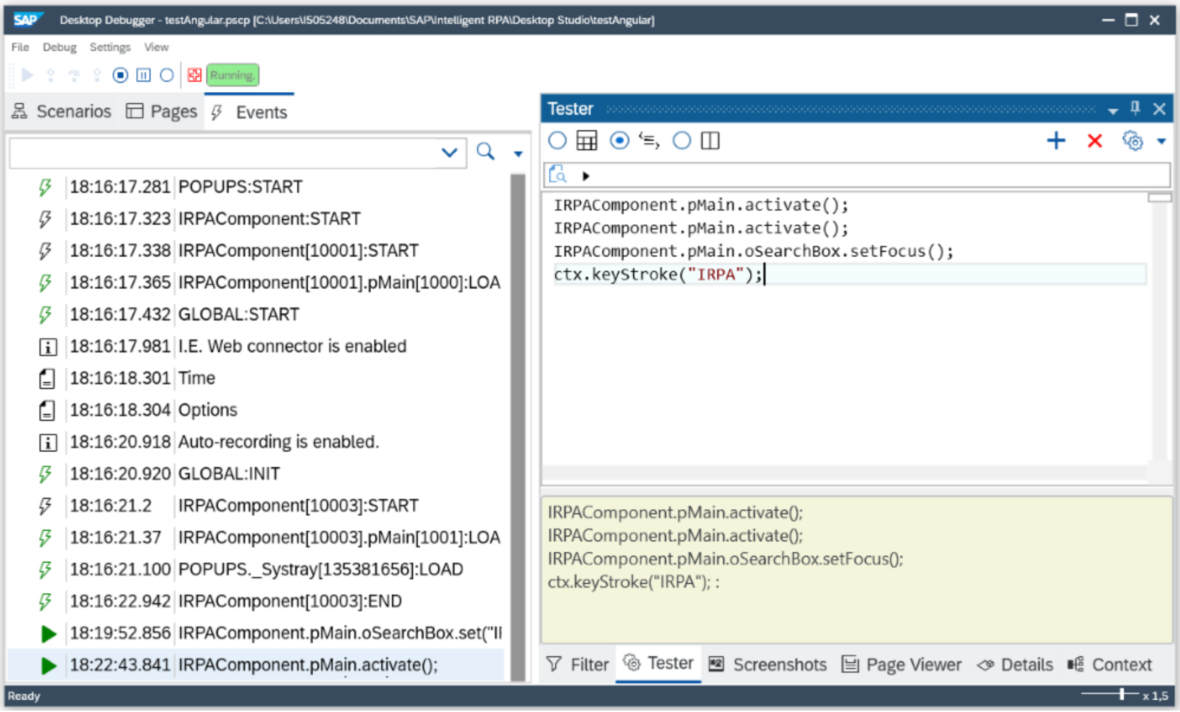
```
IRPAComponent.pMain.activate();
IRPAComponent.pMain.activate();
IRPAComponent.pMain.oSearchBox.setFocus();
ctx.keyStroke("IRPA");
```

> ⓘ Note
>
> On a web browser like Chrome, it is necessary to double activate then set focus on the item we want to send a keystroke sequence to.
>
> To avoid a simulated keystroke sequence, it should not be sent on the item directly.

As a result, a tooltip and a message are displayed.



```
ComponentService: fetched components
ComponentService: found components matching "IRPA"
```



```
18:16:17.281 POPUPS:START
18:16:17.323 IRPAComponent:START
18:16:17.338 IRPAComponent[10001]:START
18:16:17.365 IRPAComponent[10001].pMain[1000]:LOA
18:16:17.432 GLOBAL:START
18:16:17.981 I.E. Web connector is enabled
18:16:18.301 Time
18:16:18.304 Options
18:16:20.918 Auto-recording is enabled.
18:16:20.920 GLOBAL:INIT
18:16:21.2   IRPAComponent[10003]:START
18:16:21.37  IRPAComponent[10003].pMain[1001]:LOA
18:16:21.100 POPUPS._Systray[135381656]:LOAD
18:16:22.942 IRPAComponent[10003]:END
18:19:52.856 IRPAComponent.pMain.oSearchBox.set("II
18:22:43.841 IRPAComponent.pMain.activate();
```

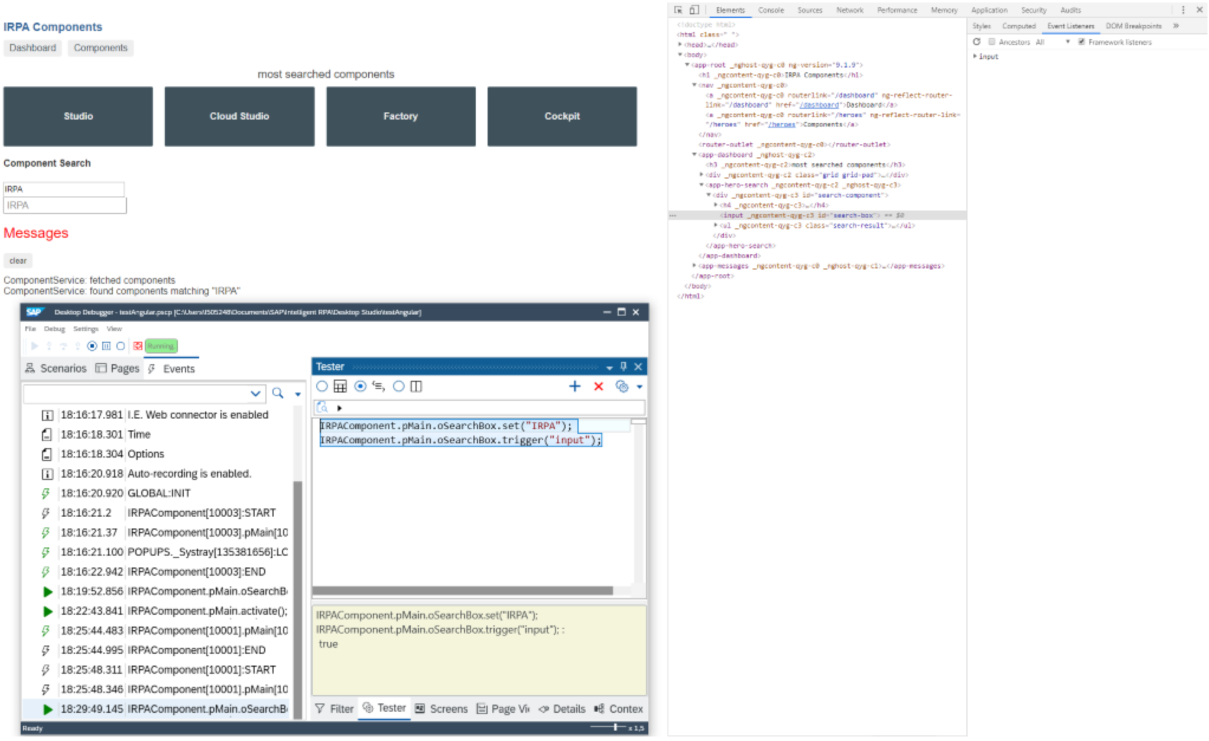
```
IRPAComponent.pMain.activate();
IRPAComponent.pMain.activate();
IRPAComponent.pMain.oSearchBox.setFocus();
ctx.keyStroke("IRPA");
```

```
IRPAComponent.pMain.activate();
IRPAComponent.pMain.activate();
IRPAComponent.pMain.oSearchBox.setFocus();
ctx.keyStroke("IRPA"); :
```

This solution can be applied to unlocked screens, however, it does not apply to machines with a locked screen and a disconnected Virtual Desktop Infrastructure. With a locked screen, `ctx.keystroke` is blocked.

Another way to resolve this behavior is to use events attached to the item:

```
IRPAComponent.pMain.oSearchBox.set("IRPA");
IRPAComponent.pMain.oSearchBox.trigger("input");
```



ComponentService: fetched components
ComponentService: found components matching "IRPA"

Desktop Debugger - testAngular.pscp [C:\Users\I505248\Documents\SAP\Intelligent RPA\Desktop Studio\testAngular]
File
Debug
Settings
View
Running
Scenarios
Pages
Events
Tester
Filter
Tester
Screens
Page Vi
Details
Contex
Ready

```
18:16:17.981 I.E. Web connector is enabled
18:16:18.301 Time
18:16:18.304 Options
18:16:20.918 Auto-recording is enabled.
18:16:20.920 GLOBAL:INIT
18:16:21.2   IRPAComponent[10003]:START
18:16:21.37  IRPAComponent[10003].pMain[10
18:16:21.100 POPUPS._Systray[135381656]:LC
18:16:22.942 IRPAComponent[10003]:END
18:19:52.856 IRPAComponent.pMain.oSearchB
18:22:43.841 IRPAComponent.pMain.activate();
18:25:44.483 IRPAComponent[10001].pMain[10
18:25:44.995 IRPAComponent[10001]:END
18:25:48.311 IRPAComponent[10001]:START
18:25:48.346 IRPAComponent[10001].pMain[10
18:29:49.145 IRPAComponent.pMain.oSearchB
```

```
IRPAComponent.pMain.oSearchBox.set("IRPA");
IRPAComponent.pMain.oSearchBox.trigger("input");
```

```
IRPAComponent.pMain.oSearchBox.set("IRPA");
IRPAComponent.pMain.oSearchBox.trigger("input"); :
 true
```

```
<!doctype html>
<html class=" ">
▶<head>…</head>
▼<body>
  ▼<app-root _nghost-qyg-c0 ng-version="9.1.9">
     <h1 _ngcontent-qyg-c0>IRPA Components</h1>
    ▼<nav _ngcontent-qyg-c0>
       <a _ngcontent-qyg-c0 routerlink="/dashboard" ng-reflect-router-
       link="/dashboard" href="/dashboard">Dashboard</a>
       <a _ngcontent-qyg-c0 routerlink="/heroes" ng-reflect-router-link=
       "/heroes" href="/heroes">Components</a>
     </nav>
     <router-outlet _ngcontent-qyg-c0></router-outlet>
    ▼<app-dashboard _nghost-qyg-c2>
       <h3 _ngcontent-qyg-c2>most searched components</h3>
      ▶<div _ngcontent-qyg-c2 class="grid grid-pad">…</div>
      ▼<app-hero-search _ngcontent-qyg-c2 _nghost-qyg-c3>
        ▼<div _ngcontent-qyg-c3 id="search-component">
          ▶<h4 _ngcontent-qyg-c3>…</h4>
...         <input _ngcontent-qyg-c3 id="search-box"> == $0
          ▶<ul _ngcontent-qyg-c3 class="search-result">…</ul>
          </div>
        </app-hero-search>
     </app-dashboard>
    ▶<app-messages _ngcontent-qyg-c0 _nghost-qyg-c1>…</app-messages>
   </app-root>
  </body>
</html>
```

As a result, the tooltip and the corresponding message are displayed. This solution can be applied to machines with a locked screen and disconnected Virtual Desktop Infrastructures, as long as the session is active.

A more complex web page may require the user to work directly in the browser by using the developer mode to find a solution.

For example, it is possible to simulate a specific mouse event on the page:

```
application.page.item.scriptItem( "dispatchEvent( new MouseEvent( 'mouseover',
{'bubbles': true, 'cancelable': true}))");
```

It is also possible to do it for a keyboard event, for input that only needs a specific key input.

## 1.2.1.2.4 Keep Screen Unlocked

A common best practice is to configure your environment to prevent the screen from locking.

This will allow mouse clicks and keystrokes to be used without getting stuck and sent to the screen lock instead of the targeted application.

## 1.2.1.2.5 Keep a Connection to the Virtual Desktop Infrastructure

If you are using a Virtual Desktop Infrastructure and mouse clicks and keystrokes are necessary, you can:

- Configure the Virtual Desktop Infrastructure to prevent the screen from locking.
- Directly connect to a Virtual Desktop Infrastructure from a local machine.

Eventually you may have a project that launches this connection from a local machine.

#### 1.2.1.2.6 Identify Another Functional Resolution

If all the resolutions previously suggested do not work, try to find a functional workaround. Another sequence of screen, another button to click on, any other way to validate your form may also work without needing a mouse click or a keystroke.

If none of the workarounds apply, it may become a blocker for your project.

It is important to track down those limitations early in your POC, design, or implementation. In order to do so, you can:

- Use action + delay + manual screen lock
- Perform regular tests on a disconnected Virtual Desktop Infrastructure or on a machine with a locked screen.

### 1.2.2 Best Practices for PDF Library

The PDF library enables data to be extracted from text-searchable PDF documents.

In some cases, the PDF library can't properly locate searched text in a document. Indeed, the order of the returned text elements may not correspond to the visual coordinates you use to render them.

To overcome such challenges, we recommend the following:

1. Use the getText method to retrieve the complete text of the PDF document.
2. Analyse the structure of the extracted index to localize the text elements you want to retrieve.
3. Based on the structure of the index, use an appropriate method or activity to retrieve the text element (getWordAfter or getWordBefore for instance).

> ⓘ Note
>
> You cannot apply a unique workaround here. In some cases, switching from getWordAfter to getWordBefore (or the opposite) can work. In more complex cases, you must dig deeper in the index structure and find index structure that remains stable.

## 1.2.3 Best Practices for Web Service Calls

### Context

When you use the `ajax.call` function to communicate with a web service, you receive a status from the server such as:

- 200.
- 300.
- And so on.

Sometimes, it seems that the call does not work and you receive a generic error message. This can be:

- The status only.
- A generic message.

It can be difficult to know where the issue comes from.

### Reasons

Web services can combine many different settings:

- Certificates on server/client side.
- SSL Certificates, which encrypt information, provide authentication and so on. They are necessary if you want to authenticate on the server.
- Tokens.

There are different types of web services:

- SOAP.
- ODATA.
- REST.

If you combine all these web services and settings, this can lead you to many different possibilities. So, there are many ways to configure the `ajax.call` method.

Moreover, you may be confronted with a misconfiguration of the servers or certificates that can lead to specific issues.

### 1.2.3.1 Causes and Resolutions

In this part, you can find the main causes of issues and how to resolve them.

In the following topics, you will learn more about:

- Client certificates.
- Server certificates.
- Cookies.
- Tokens.

## 1.2.3.1.1 Client Certificates

You use Client certificates to authenticate on a server. Sometimes you cannot connect to a server with a certificate.

You can use the option `ignoreClientCertificate` to avoid the certificate authentication.

This option uses a C++ implementation of the call.

Example:

```
ctx.ajax.call({
    url: url,
    method: 'GET',
    contentType: "application/json",
    headers: headers,
    ignoreClientCertificate: true,
    success: function(res, status, xhr) {
        rootData.xCSRF = xhr.headers["x-csrf-token"];

        var value = xhr.headers["Set-Cookie"];
        rootData.Cookies = value.split('; ')[0] + "; ";

        sc.endStep();
    },
    error: function(xhr, status, statusText) {
        ctx.log('error get')
        sc.endStep();
    }
});
```

## 1.2.3.1.2 Server Certificates

API may need such certificates. They are generally mandatory.

`ctx.ajax.call` is not able to force the call without those certificates.

Moreover, you cannot specify which certificates to use as you are not in a browser environment.

For more information on how to work around this issue, see: **How to deal with Ajax call issues** in Ajax Call [page 27]

### 1.2.3.1.3 Cookies

Cookies allow you to authenticate client requests and keep all session information.

If you need to use cookies in your calls, you must set a cookie to keep this information. You will then be able to use it in another call.

Example:

```
ctx.ajax.call({ //First call, get the cookies information
    url: "[url]",
    method: e.ajax.method.get,

    contentType: e.ajax.content.json,
    usePassport: true,
    ignoreClientCertificate:true,

    headers : {
        'Authorization': 'Basic [base 64 login:password]',
        'X-CSRF-Token': 'Fetch'
    },
    success: function(res, status, xhr) {

    ctx.wait(function(ev) {
        try {

            var csrf = xhr.headers['x-csrf-token'];
            var cookie = xhr.headers['Set-Cookie'];
            var myUrl = "https://domainName/sap/opu/odata/sap/xxx";

        ctx.wait(function(ev) {
            ctx.ajax.call({ //Second call, use the cookies in the headers object
            url: myUrl,
            method: e.ajax.method.post,

            ignoreClientCertificate:true,
            contentType: 'application/json;charset=utf-8',
            headers : {
             'Content-type': 'application/json;charset=utf-8',
             'X-CSRF-TOKEN': csrf,
            'Cookie': cookie
        },
        success: function(res, status, xhr) {

        },
        error: function(xhr,res,status) {
        }
      });
    }, 0);

    } catch (ex) {

   }
  }, 0);
},
});
```

## 1.2.3.1.4 Tokens

A token is a piece of data created by a server that contains information to identify a user and the token validity. If you use the token incorrectly in the Ajax call, it may lead to an issue. Like the cookies, the token needs two calls.

For instance, you make an API call in the Factory. You use the first call to get the token. The second call triggers the API call and it creates a job in the job queue. In this case, the calls are linked. The second call happens if the first one is successful.

Example:

```
ctx.ajax.call({ //First call, get the token

    url: serviceAccount.urls.accessTokenURL, //Url got from the service key “url
parameter +  /oauth/token?grant_type=client_credentials”

    method: e.ajax.method.get,

    contentType: e.ajax.content.json,

    header:{

    Accept: e.ajax.content.json,

    Authorization: 'Basic ' +
ctx.base64.encode(serviceAccount.credentials.clientID + ':' +
serviceAccount.credentials.clientSecret, false) // clientID and clientSecret got
from the service key

    },

    success: function(res, status, xhr) {

        ctx.log('Token generated');

        sc.localData.Token = 'Bearer ' + ctx.get(res, 'access_token');

        ctx.ajax.call({ // Second call

            url: serviceAccount.urls.apiTriggerURL, //Url from the API trigger
(end with “/run”)

            method: e.ajax.method.post,

               contentType: e.ajax.content.json,

               header:{

                 Accept: e.ajax.content.json,

                     'irpa-trigger-token':
serviceAccount.urls.irpaTriggerToken, //Token get from the API trigger

                 Authorization: sc.localData.Token

              },

            data:{

            //Payload of your API Trigger

              },
            success: function(res, status, xhr) {

                ctx.log('success');

                sc.endStep();

            return;

            },

        error: function(xhr, status, statusText) {

            //web service failed

            sc.setError(e.error.Fail, 'Web service failed: [' + status + ']' + 
statusText + ' - ' + xhr.responseText);

            sc.endScenario();

        return;

            }

        });

    },

    error: function(xhr, status, statusText) {

        //web service failed

        sc.setError(e.error.Fail, 'Web service failed: [' + status + ']' + 
statusText + ' - ' + xhr.responseText);

        sc.endScenario();

        return;

    }
});
```

## 1.2.3.2 Examples of the Best Practices

Here are some examples of the best practices for Web Service Calls.

As stated before, due to the complexity of the Ajax call, you may face different issues.

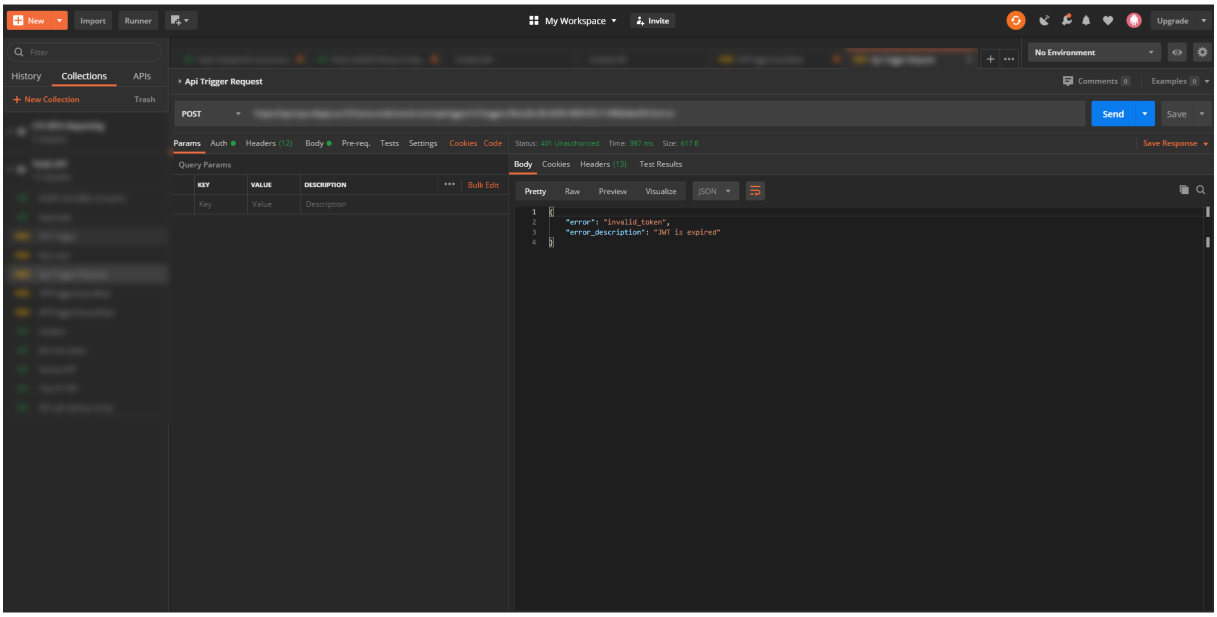
To assess these issues with the Ajax call, you can use multiple tools:

- Postman.
- cURL.
- Terminal commands.
- Ajax Call.

## 1.2.3.2.1 Postman

Postman is a tool that allows you to interrogate and test Web services and API calls.

This tool has multiple features and is easy to use.



Api Trigger Request

POST [redacted]

Status: 401 Unauthorized Time: 397 ms Size: 617 B

```
{
    "error": "invalid_token",
    "error_description": "JWT is expired"
}
```

## 1.2.3.2.2 cURL

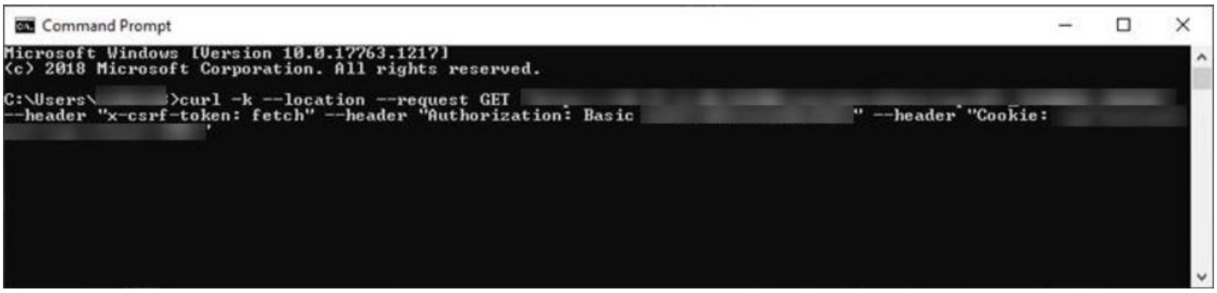
cURL is a command-line tool that you can use to get or send data including files using URL syntax.

On Windows 10 build 17063, cURL is included by default.

## 1.2.3.2.3 Terminal Commands

In the Command Prompt, you can test the cURL command during the assessment.

If the test is successful, the agent calls the cURL command through the terminal.



```
Microsoft Windows [Version 10.0.17763.1217]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\[redacted]>curl -k --location --request GET [redacted]
--header "x-csrf-token: fetch" --header "Authorization: Basic [redacted]" --header "Cookie: [redacted]
[redacted]'
```

# 1.2.3.2.4 Ajax Call

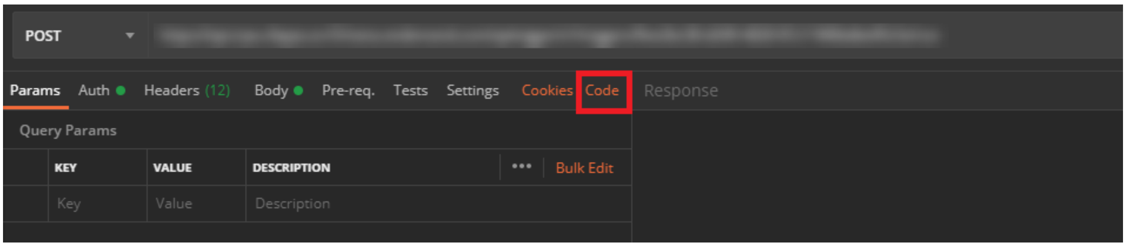
## How to configure your Ajax call function

If you do not know how to configure the Ajax call function, follow this process:

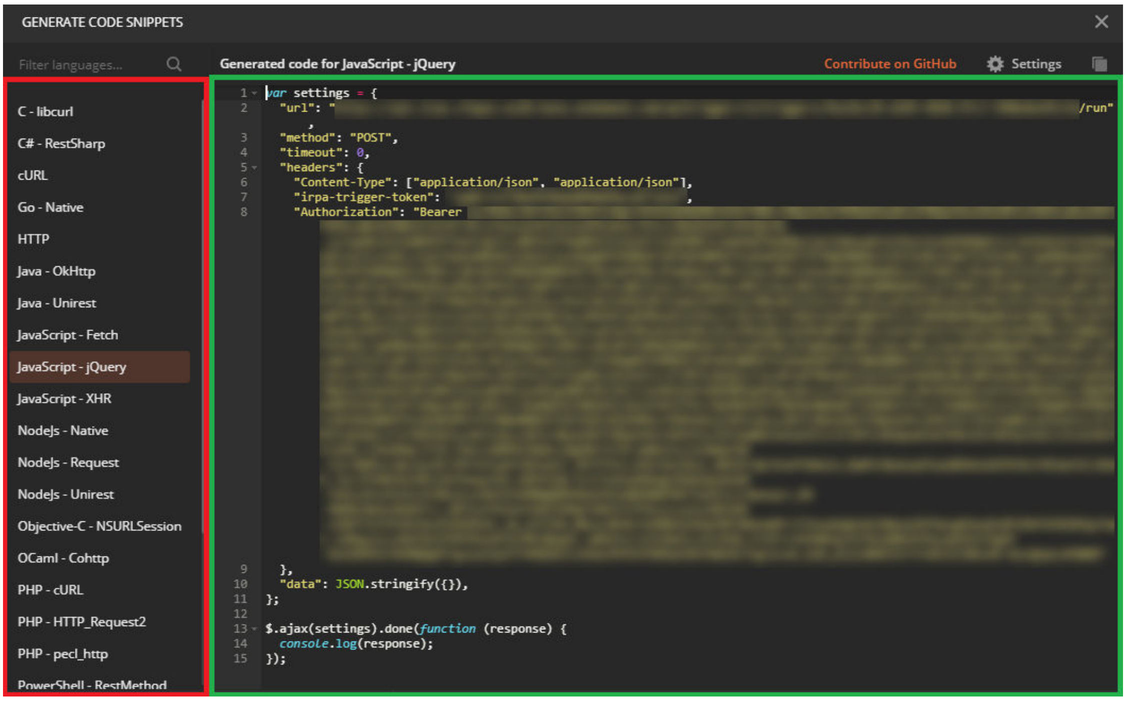
1. Make the request work on Postman.

> ⓘ Note
>
> If it does not work even with Postman, it means that you cannot make it work with the Ajax call function.

1. When it works, click the *Code* button.



2. Choose the programming language you work with.



Generated code for JavaScript - jQuery

```
var settings = {
  "url": "[redacted]/run"
      ,
  "method": "POST",
  "timeout": 0,
  "headers": {
    "Content-Type": ["application/json", "application/json"],
    "irpa-trigger-token": [redacted],
    "Authorization": "Bearer [redacted]
  },
  "data": JSON.stringify({}),
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

> ⓘ Note
>
> You can see different languages in the red box on the left. Here, we choose the JavaScript-jQuery language because the Ajax call function has been developed with the jQuery.ajax() method. In the green box, you notice the code generated by Postman.

2. Use this code in the Ajax call function.
   Copy and paste this code in the Desktop Studio and adjust the code if necessary. For more information, see: call

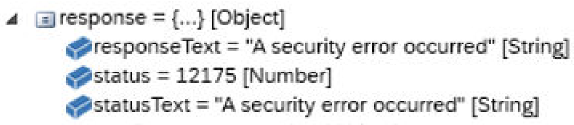
```
function getCall(success, error, localData){
    var result;
    var url = localData.fetchAPI = "";

    //API call to get csrf token and cookies
    ctx.ajax.call({
        method: e.ajax.method.get,
        url: url,
        contentType: e.ajax.content.json,
        headers: {
            'Authorization': localData.auth,
            'x-csrf-token' : 'fetch',
            'Cookie' : 'fetch',
            'Accept' : 'application/json'
        },
        ignoreClientCertificate: localData.clientCert,
        success: function(res, status, xhr) {

            if(res.d){
            result = xhr;
            success(result);
            } else {
                result = res;
                error(result);
            }

        },
        error: function(xhr, error, statusText) {

            result = res;
            error(result);

        }
    });
```

## How to deal with Ajax call issues

If you still face issues during the call (issues from the xhr object) after you performed the test in the previous process, try the following tests:

1. Try `ctx.ajax.call` with the option `ignoreClientCertificate` as true.

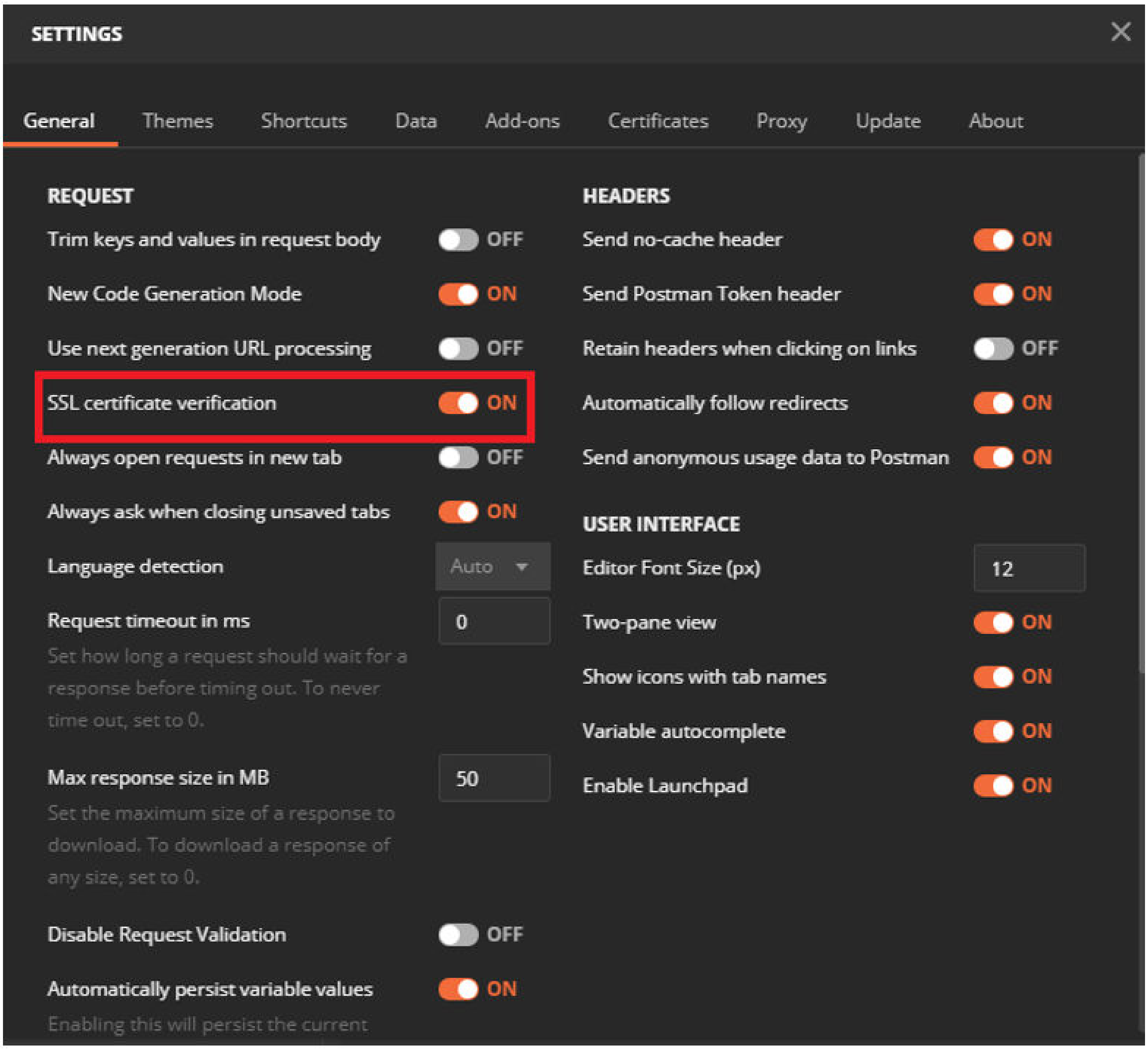
> ⓘ Note
>
> If it still does not work, check the error code, that can lead you to the origin of the error.



```
◢ response = {...} [Object]
    responseText = "A security error occurred" [String]
    status = 12175 [Number]
    statusText = "A security error occurred" [String]
```

.

2. Change the settings in Postman.



REQUEST

Trim keys and values in request body OFF

New Code Generation Mode ON

Use next generation URL processing OFF

SSL certificate verification ON

Always open requests in new tab OFF

Always ask when closing unsaved tabs ON

Language detection Auto

Request timeout in ms 0

Set how long a request should wait for a response before timing out. To never time out, set to 0.

Max response size in MB 50

Set the maximum size of a response to download. To download a response of any size, set to 0.

Disable Request Validation OFF

Automatically persist variable values ON

Enabling this will persist the current

HEADERS

Send no-cache header ON

Send Postman Token header ON

Retain headers when clicking on links OFF

Automatically follow redirects ON

Send anonymous usage data to Postman ON

USER INTERFACE

Editor Font Size (px) 12

Two-pane view ON

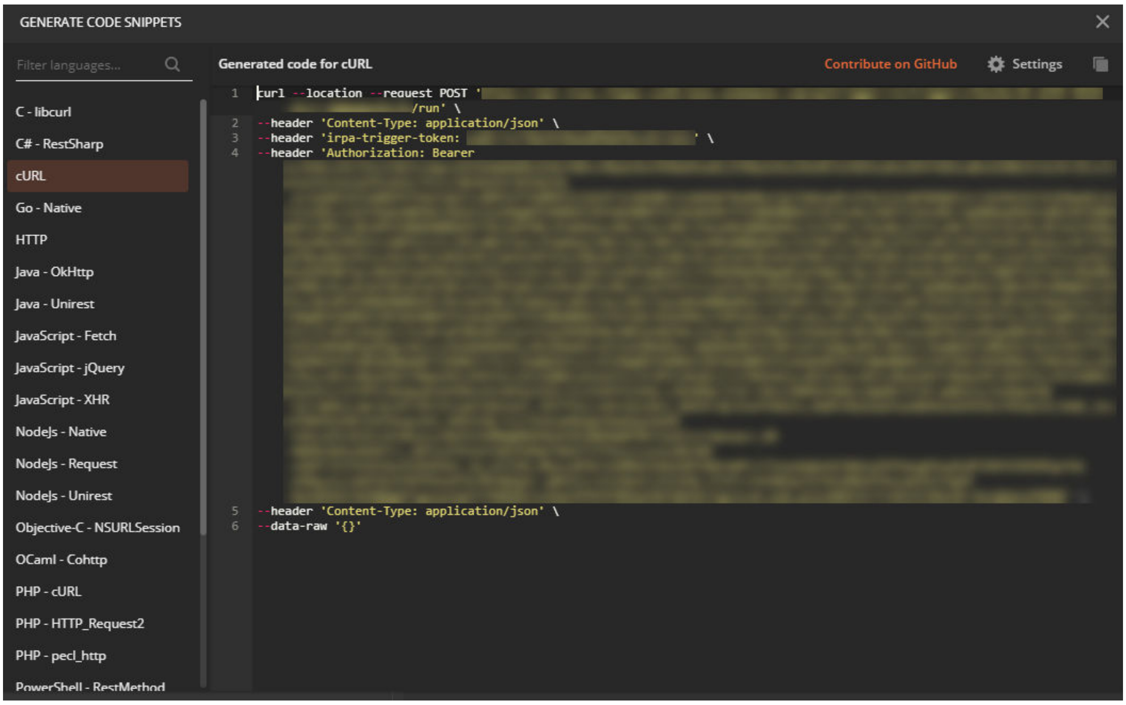
Show icons with tab names ON

Variable autocomplete ON

Enable Launchpad ON

By default, the *SSL certificate verification* is not enabled in Postman. This means that it forces the call even if the certificate is not available.
If the call on Postman does not work after you enabled this option, this means that this certificate is mandatory to make the call. As stated before, the Ajax call does not have this option.
One solution could be to use cURL.

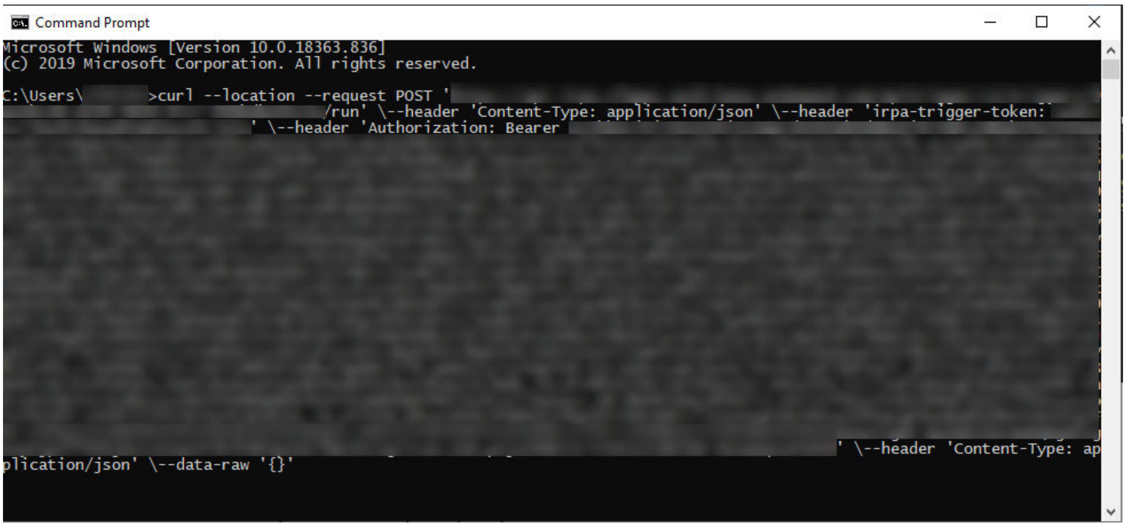
3. Try the cURL code given by Postman.
    - Follow the **How to configurate my Ajax call function** process above, from **step 1.a. to step 1.b.**, to retrieve the proper cURL code.



```
curl --location --request POST '[redacted]
/run' \
--header 'Content-Type: application/json' \
--header 'irpa-trigger-token: [redacted]' \
--header 'Authorization: Bearer [redacted]
--header 'Content-Type: application/json' \
--data-raw '{}'
```

- In this code, remove all characters **\** and replace all the simple quotation marks with double quotation marks.
- Run the cURL code on Command Prompt.
  Example:

```
curl --location --request GET "http://XXXX"
```



```
icrosoft Windows [Version 10.0.18363.836]
(c) 2019 Microsoft Corporation. All rights reserved.

:\Users\[redacted]>curl --location --request POST '
[redacted]/run' \--header 'Content-Type: application/json' \--header 'irpa-trigger-token:
[redacted]' \--header 'Authorization: Bearer [redacted]
[redacted]' \--header 'Content-Type: ap
plication/json' \--data-raw '{}'
```

If you have an error message that appears with the cURL code, do a search on the Internet. The error messages are usually very explicit. If the code is successful, you can integrate it in an SAP Intelligent RPA call.

- Implement this code in the Desktop Studio.

```
var cmdCurl = 'curl -k --location --request GET "http://XXXX"
ctx.exec(cmdCurl, 60, function(reqres) {
    var reqdata = ctx.json.parse(reqres.output);
});
```

> **Note**
>
> The option **-k** is used to ignore the client certificates.

## 1.2.4 Best Practices to Handle Important Amount of Data During Scenario Execution

It is developer's responsibility to ensure that big amount of data are not put in sc.data object during the scenario execution, because the sc.data object it sent back to the Cloud Factory.

As a workaround, if a big amount of data is generated and manipulated during the scenario, it can be emptied (partially or completely) in the last step of the scenario.

Example:

```
...
// during scenario execution
sc.data.bigData = ctx.fso.file.open(theBigFile);
...
// last step before ending the scenario
sc.data.bigData = undefined;
sc.endStep();
```

## 1.2.5 Best Practices for Captchas

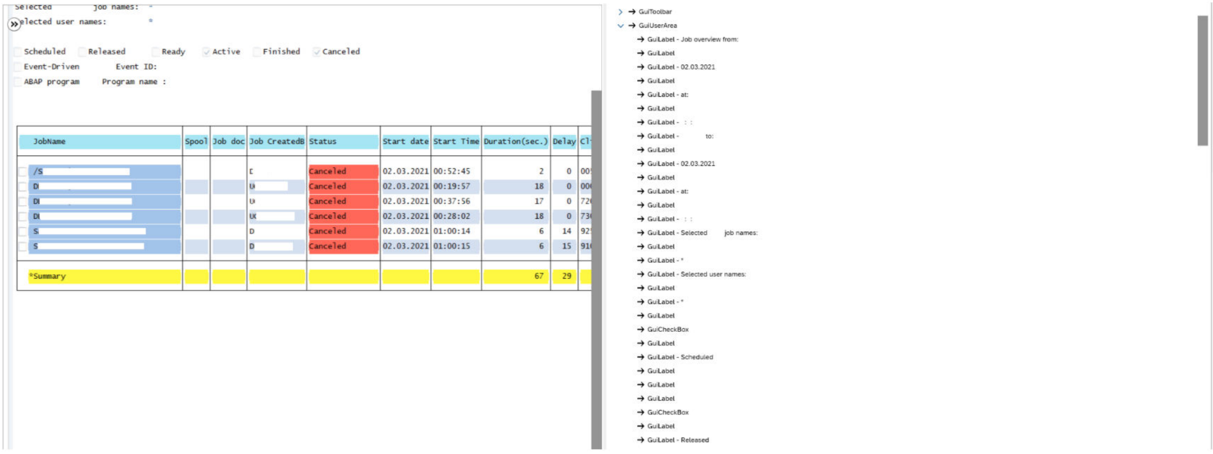
You can sometimes face captchas when you want to automate a website.

You can deal with captchas in two ways:

- You can try to solve captchas using an external image recognition library for instance (it may be a hard process development).
- You can also try to detect when the captcha is displayed and avoid this behavior. For example, a captcha may be displayed when the fields of a form are filled in too quickly. In this case, the best practice is to add a three-second delay between each step of your automation, for example.

## 1.2.6 Best Practices for Automating SAP GUI Screens with Low Hierarchies

When you capture a table with low hierarchies in the Desktop Studio, a lot of elements are displayed on the application tester.



Selected job names: *
Selected user names: *

□ Scheduled □ Released □ Ready ☑ Active □ Finished ☑ Canceled
□ Event-Driven Event ID:
□ ABAP program Program name :

| JobName | Spool | Job doc | Job CreatedB | Status | Start date | Start Time | Duration(sec.) | Delay | Cl |
|---|---|---|---|---|---|---|---|---|---|
| [redacted] | | | C | Canceled | 02.03.2021 | 00:52:45 | 2 | 0 | 00 |
| [redacted] | | | [redacted] | Canceled | 02.03.2021 | 00:19:57 | 18 | 0 | 00 |
| [redacted] | | | U | Canceled | 02.03.2021 | 00:37:56 | 17 | 0 | 72 |
| [redacted] | | | [redacted] | Canceled | 02.03.2021 | 00:28:02 | 18 | 0 | 73 |
| [redacted] | | | D | Canceled | 02.03.2021 | 01:00:14 | 6 | 14 | 92 |
| [redacted] | | | [redacted] | Canceled | 02.03.2021 | 01:00:15 | 6 | 15 | 91 |
| *Summary | | | | | | | 67 | 29 | |

- → GuiToolbar
- → GuiUserArea
  - → GuiLabel - Job overview from:
  - → GuiLabel
  - → GuiLabel - 02.03.2021
  - → GuiLabel
  - → GuiLabel - at:
  - → GuiLabel
  - → GuiLabel - : :
  - → GuiLabel - to:
  - → GuiLabel
  - → GuiLabel - 02.03.2021
  - → GuiLabel
  - → GuiLabel - at:
  - → GuiLabel
  - → GuiLabel - : :
  - → GuiLabel - Selected job names:
  - → GuiLabel
  - → GuiLabel - *
  - → GuiLabel - Selected user names:
  - → GuiLabel
  - → GuiLabel - *
  - → GuiLabel
  - → GuiCheckBox
  - → GuiLabel
  - → GuiLabel - Scheduled
  - → GuiLabel
  - → GuiLabel
  - → GuiLabel
  - → GuiCheckBox
  - → GuiLabel
  - → GuiLabel - Released

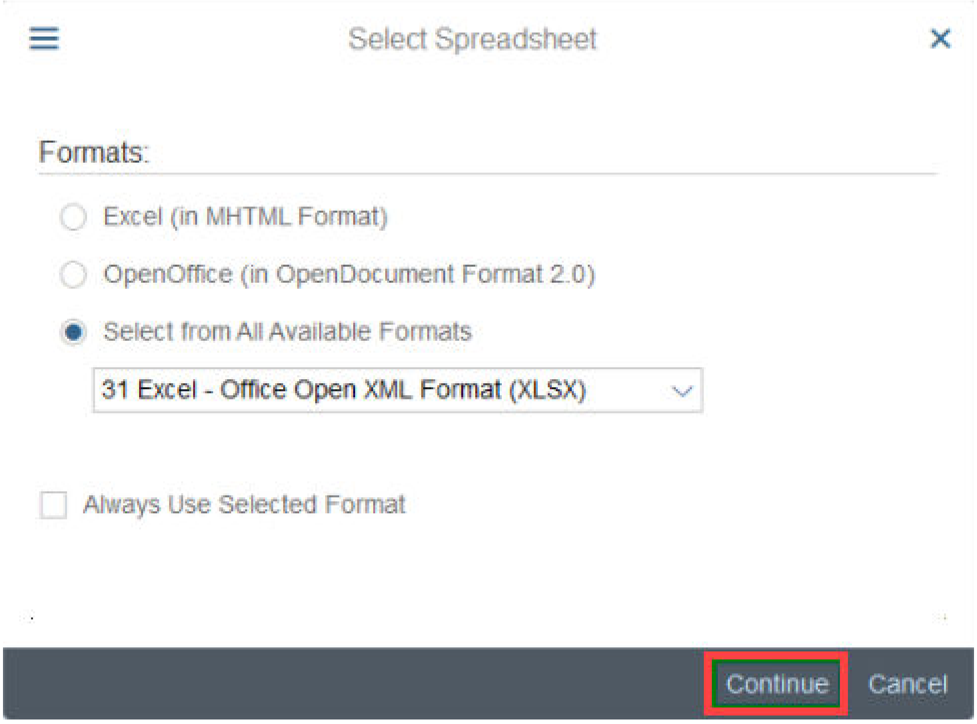
To recognize all the elements on the screen requires a lot of testing that can lead to a bad performance during the bot execution.

To handle this behavior, you can avoid reading the table in the Desktop 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* action on the screen you captured with the table.
   The first pop-up window appears.



Select Spreadsheet

Formats:

○ Excel (in MHTML Format)

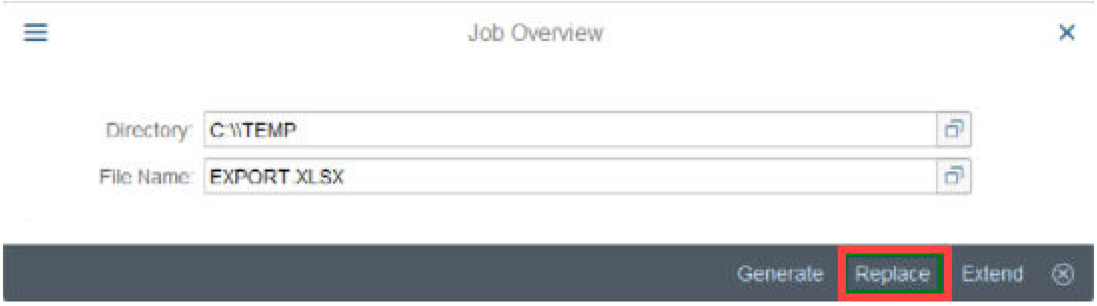
○ OpenOffice (in OpenDocument Format 2.0)

● Select from All Available Formats

31 Excel - Office Open XML Format (XLSX)

□ Always Use Selected Format

2. Click on *Continue*.
   The second pop-up window appears.



Job Overview

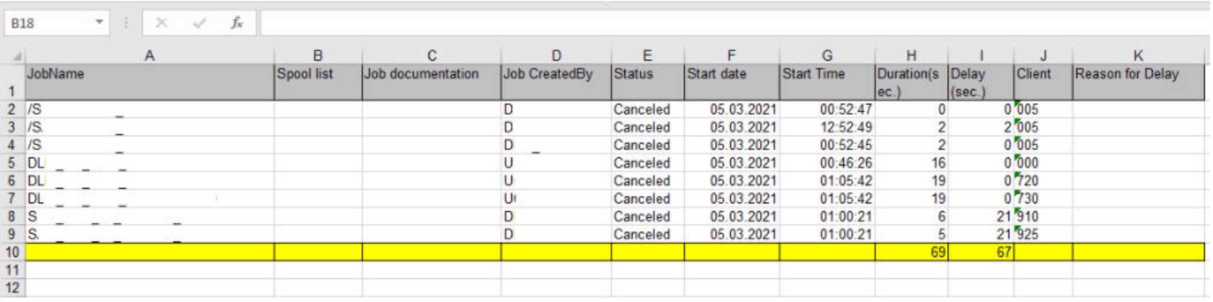
Directory: C:\\TEMP

File Name: EXPORT.XLSX

3. Click *Replace*.

## Result

The table is opened and saved in the Excel file.



B18

| | 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 | | | | | | | | | | | |

## 1.2.7 Best Practices for Managing Files

This section provides best practices for managing files.

### Configuration File that works for Different Users and Machines

A typical example would be that you need to place a file or folder locally. Asking end-users to create a folder manually for the bot is not very convenient. Also, it's hard to find a conventional path name that would work for all users in your organization.

You can avoid this by making your bot create these folders by default. Hence, if the bot can't find the required folder on the machine, it will create it. You can make it even more convenient by adding a condition for checking the folder's existence.

Sample Code

```
                // this will create a folder 'Config' under
'Desktop\Automation\ProjetName' only if this folder does not exist

                var configFile
=ctx.fso.folder.getSpecialFolder(e.shell.specialFolder.Desktop) + "\
\Automations\\"+ ctx.options.projectName + "\\config";
                if (!configFile.exist){
                ctx.fso.folder.create(configFile);
                }
```

### Working Files

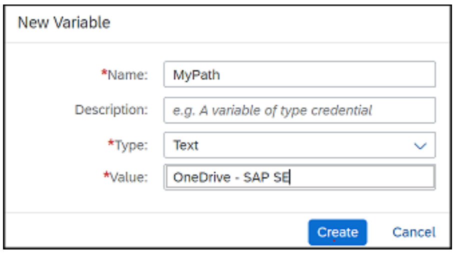
For temporary files used by the bot, Windows users may need some specific authorizations. The best practice is to create these files under the log folder of the project. This is the more convenient way to manage the working files.

Sample Code

```
                //this will create a file  'TempFiles' in your project folder
                ctx.fso.folder.create(ctx.options.path.log+"\\TempFiles\\")
```

### Managing Folder Path Using Factory Variables

When managing a common file/folder for a group of users who all have access to a shared repository. You can use the Cloud Factory text variable to store the path. then retrieve it in your script as in the example below.


New Variable

*Name: MyPath

Description: e.g. A variable of type credential

*Type: Text

*Value: OneDrive - SAP SE

For more details about Cloud Factory variables, refer to the Declaring Environment Variables from SAP Intelligent RPA Factory section.

{-} Sample Code

```
                var mypath;
                //
----------------------------------------------------------------
                //   Step: get_path_var
                //
----------------------------------------------------------------
                GLOBAL.step({ get_path_var: function(ev, sc, st) {
                var rootData = sc.data;
                ctx.workflow('Custom_getPathVar', 'c514ca1b-3d9d-445b-
a2e3-069c9a7b9397') ;

                // Describe functionality to be implemented in JavaSript
later in the project.

                // Declare setting path
                ctx.setting({ MyPath: {
                comment: "sc.localData.mytenant",
                server: true
                }});
                // Get setting
                ctx.settings.MyPath.get(function(code, label, setting) {
                if (code == e.error.OK) {
                // get value from setting.value
                mypath = setting.value;
                ctx.log(mypath);

                } else{
                ctx.log('Error during setting retrieval');
                sc.endStep(); // pSapGlobalCloud8L_man
                return;

                }
                sc.endStep(); // code_
                return;

                });


                }});
```

## Output Files

In case your scenario requires one or many output files, a good practice is to create the output folder locally.

{·} Sample Code

```
                    //this will create a folder 'output' under the path
'Desktop\Automations\ProjectName'


ctx.fso.folder.create(ctx.fso.folder.getSpecialFolder(e.shell.specialFolder.De
sktop) + "\\Automations\\"+ ctx.options.projectName + "\\output")
```

## 1.2.8 Best Practices for Chrome Extension

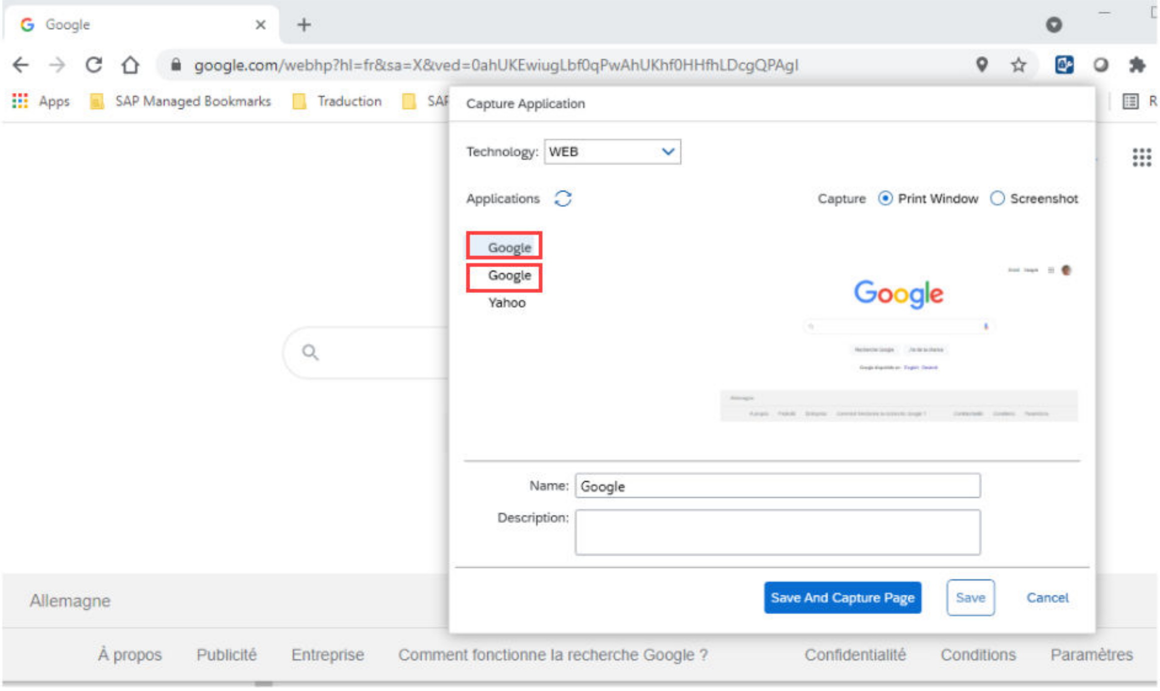
You can sometimes face an issue if you have enabled the *SAP Intelligent RPA Extension* for Chrome twice.

### Context

For more information on how to enable the web browser extension for Chrome, see Configuring Web Browsers.

If you have enabled the *SAP Intelligent RPA Extension* for Chrome twice, you can face the following issues:

- When you want to capture an application, you can see that the application appears twice on the *Capture Application* panel. This happens even if one of these applications is launched in the Chrome browser.

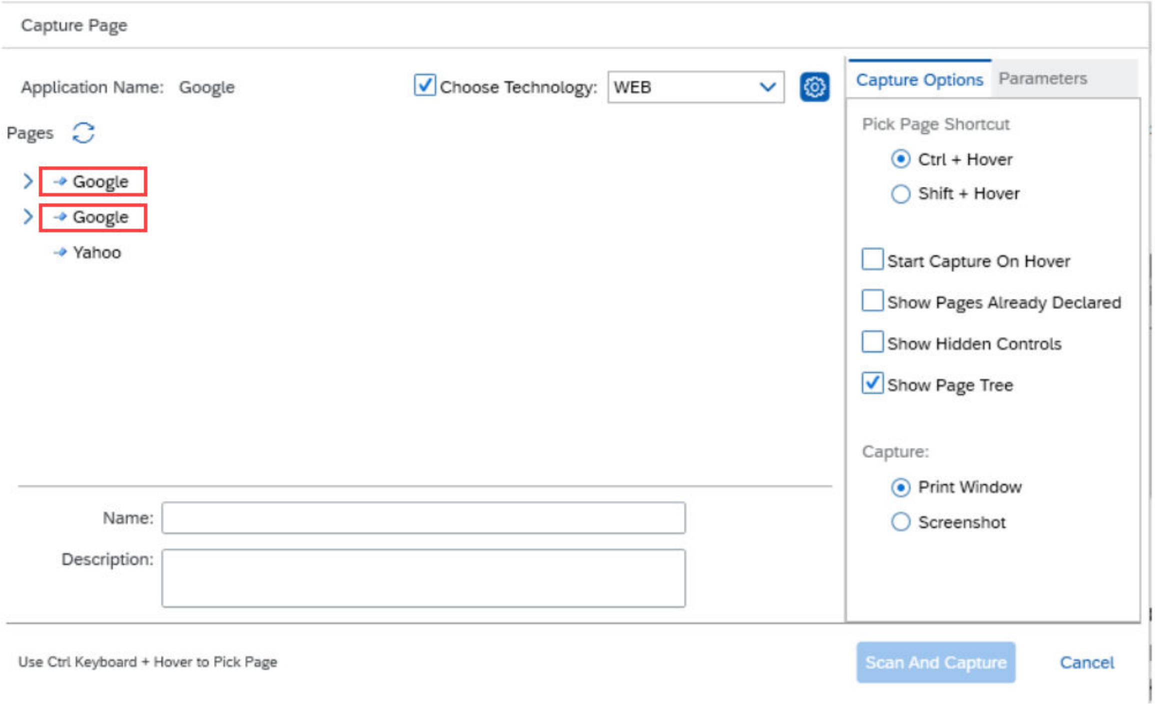


Technology: WEB

Capture ● Print Window ○ Screenshot

Name: Google

- When you want to capture pages from an application, you can see that the same page appears twice on the *Capture Page* panel.



Capture Page

Application Name: Google

☑ Choose Technology: WEB

Pages

- Google
- Google
- Yahoo

Name:

Description:

Capture Options | Parameters

Pick Page Shortcut

● Ctrl + Hover

○ Shift + Hover

□ Start Capture On Hover

□ Show Pages Already Declared

□ Show Hidden Controls

☑ Show Page Tree

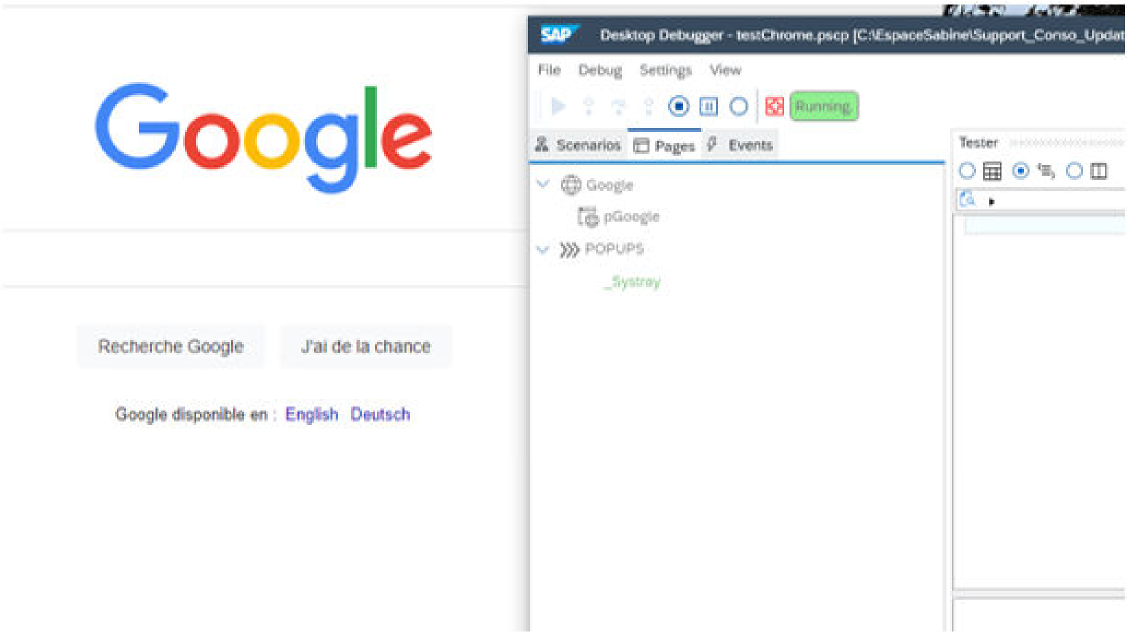
Capture:

● Print Window

○ Screenshot

Use Ctrl Keyboard + Hover to Pick Page

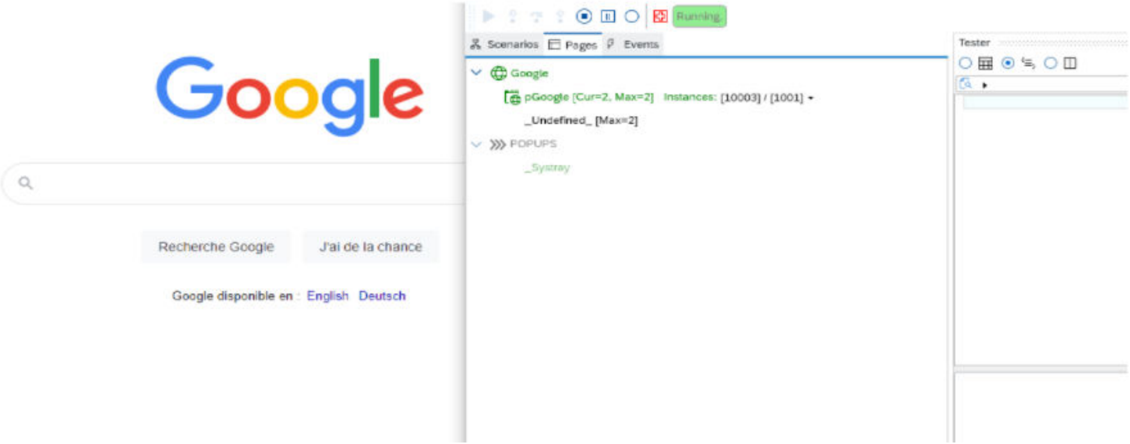
- When the Web application is launched in the Chrome Browse, the page and the application are not recognized.



Desktop Debugger - testChrome.pscp [C:\EspaceSabine\Support_Conso_Updat

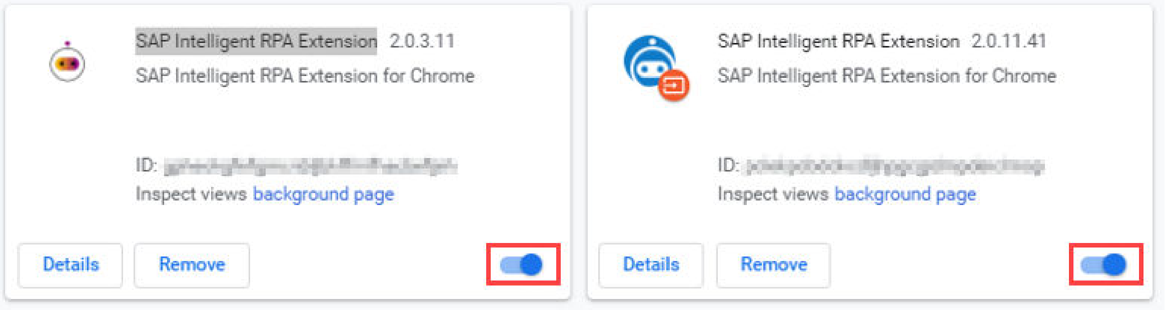
- Google
  - pGoogle
- POPUPS
  - _Systray

- To solve this problem, you can manually refresh the page. When the page is recognized you can see that the same page appears twice.



## Disabling One of the Extensions

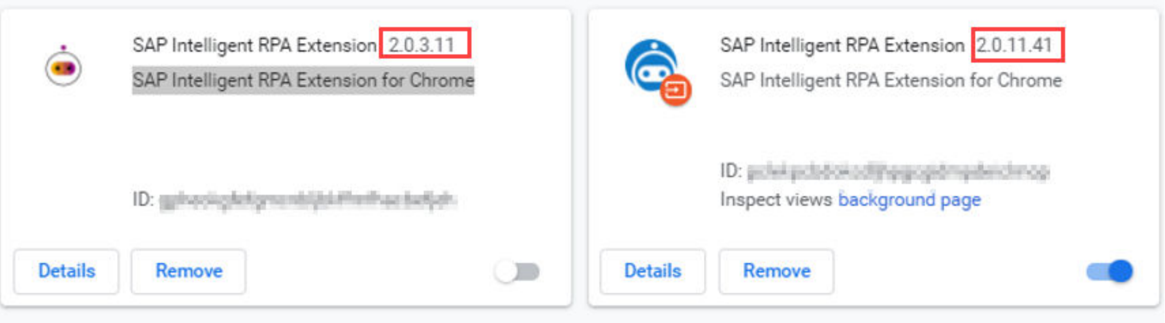
1. Enter **chrome://extensions/** in the Google search bar.
   You can notice that you have two SAP Intelligent RPA Extension for Chrome tiles.



SAP Intelligent RPA Extension 2.0.3.11
SAP Intelligent RPA Extension for Chrome
ID: [redacted]
Inspect views background page

SAP Intelligent RPA Extension 2.0.11.41
SAP Intelligent RPA Extension for Chrome
ID: [redacted]
Inspect views background page

2. Check the versions of the extensions and disable the one with the older version.



SAP Intelligent RPA Extension 2.0.3.11
SAP Intelligent RPA Extension for Chrome
ID: [redacted]

SAP Intelligent RPA Extension 2.0.11.41
SAP Intelligent RPA Extension for Chrome
ID: [redacted]
Inspect views background page

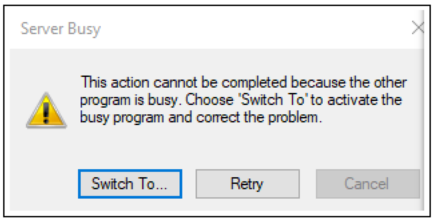
## Related Information

Configuring Web Browsers

## 1.2.9 Best Practices for managing Server Busy Pop-up

This section describes best practices for managing a server busy pop-up while executing an automation.

### Symptom



Server Busy

This action cannot be completed because the other program is busy. Choose 'Switch To' to activate the busy program and correct the problem.

- Workflow freeze. The process hangs and nothing is processed after that.
- The *Server Busy* pop-up often cannot be closed.
- Most of the time, agent systray is not reacting anymore.
- To stop the agent, it is necessary to kill the process.
- Sometime the *Server Busy* pop-up may disappear after a while and then process resume.

### Origin

The *Server Busy* pop-up is triggered when

- The target application is fully occupied and cannot answer to agent requests.
  - A long JavaScript is running on a web page
  - The page is downloading or uploading big set of data (downloading a file, searching data in backend, and many more.)
  - The application is slow or on an undersized environment
- The agent tries to perform an action on this same application in the same time (**click()** or **refresh()** or **...** )

Solution depends on the technical connector but basically, we need to be sure that the application is available before doing anything else.

In some cases, it is fully originated from the target application. It means that this pop-up could be triggered by simple manual action without any running Agent.

In that case, it is up to you to determine when this may happen, and how you avoid it manually. You will then have to mimic that in your workflow.

However, by experience, most of the time the popup will be triggered by the Agent itself (ctxtrun.exe), even it is related to your application unavailability.

The server busy popup should disappear after stopping the Agent.

If it comes from the Agent, then there are two possible technical origins that need to be identified.

- SDK
  - The SDK will ask the technical connector to perform actions and will request for a feedback.
    - Custom Polling and polling on item (item.wait())
    - Click on element
    - Execscript
  - Technical connector
    - The connector, beside executing requests from the SDK (see above), will mostly request the status of handled application
    - LOAD, wait of pages
    - Polling on pages
    - Start of application

### Reminder

In any case, it is essential that you identify the actions that may led to this popup. Most of the time, this popup will be steadily appearing on the same workflow steps.

Even if you request for help, we will ask you to provide such information.

We strongly advise you to use project traces to do so, refer to the 2776434, **SAP Knowledge Base Article**.

### How to Check the *Server Busy* pop-up is Connector Wise

- Identify the event that leads to the popup in your workflow, and all the surrounding actions, eventually using project traces. Refer to the 2776434, **SAP Knowledge Base Article**.
  - A click on a button
  - The LOAD of a specific page
  - Call of a script on a page
  - Download of a file
  - ...
- Launch the debugger without launching any workflow
- Manually replicate the action that would have been done by the workflow right before this popup appearance
- If nothing appears then this issue is not triggered by the connector itself
  - It means that the wait or LOAD **on pages** activities are not the one that trigger this popup.
  - It is safe to wait for those pages, and the probable cause of the issue should be related with requested actions from the agent.
- If the *Server Busy* pop-up appears:
  - If the application is captured with UIAutomation technology, then see the **Connector Wise** sub-section of the **Solution and Workaround** section.
  - Otherwise, open an incident on **CA-ML-IPA component**. It will be a possible blocker.

### How to Check the *Server Busy* pop-up is SDK Wise

- Identify the event that leads to the popup in your workflow, and all the surrounding actions, eventually using project traces. Refer to the 2776434, **SAP Knowledge Base Article**.
  - A click on a button
  - The LOAD of a specific page

    - Call of a script on a page
    - Download of a file
    - ...
- Launch the debugger without launching any workflow
- Copy paste actions (specific action and surrounding ones) from the workflow that seems to lead to this issue
    - Get rid of every scenario activity (endStep, endScenario, return, data management ...)
    - It will often look like a click on an item that lead to a page wait or LOAD handler, that leads to a specific action as other click or item polling or custom script execution.

In case of Server Busy pop-up, we identified multiple possible activities that leads to this, but it is not exhaustive, those are SDK activities. The SDK will expect a feedback from the technical Connector that may be hanging waiting for a feedback from the unavailable application:

- Custom polling on a page (ctx.polling)
- Item polling (app.page.myElement.wait)
- UIAutomation refresh()
- Web execscript

There are activities that will not request for such feedback and that should help finding a solution/workaround:

- Use the wait on a page as much as possible
    - A long wait must exist on an item to avoid item polling (item.wait)
- UIAutomation lockRefresh / unlockRefresh to prevent application flooding by the connector

## Solution and Workaround

Having an optimized application [page 61] is very important in this case. Optimizing an application will reduce the cost of requesting application, page, and item state.

That will then limit the pressure on potential complex application.

> ⓘ Note
>
> Best practices on recognition with UIAutomation is to avoid must exist / must not exist when possible. In this case we may have to use them because of the necessity to avoid polling on items. (ctx.polling / app.page.element.wait)

### SDK Wise

#### UIAutomation/Web Technology

Your page took long time to LOAD / is downloading or uploading data and you experience this issue when sending action to the page. This page potentially being not yet fully available or not loaded with complete data.

- Search for any element that will surely indicate full LOAD of your page
    - Title update
    - Pagination update
    - Item fully loaded

    - Item not grayed out anymore
    - Loading icon disappearance
    - ...
- Declare it as a MustExist on this page, with proper criteria indicating expected element state (ungrayed out, display none ...)
- Waiting for the related page will also wait for this element without SDK polling.
    - Doing an item.wait is SDK request
    - Having a must exist on the same item is only Connector duty (then no feedback will be requested by the SDK > no server busy popup)

**GUI Technology**

**Subpages**

Subpage may provoke this issue. You need then to avoid having GUI subpages. If a GUI subpage is not recognized, then SAPGUI will freeze which will then leads to a server busy popup.

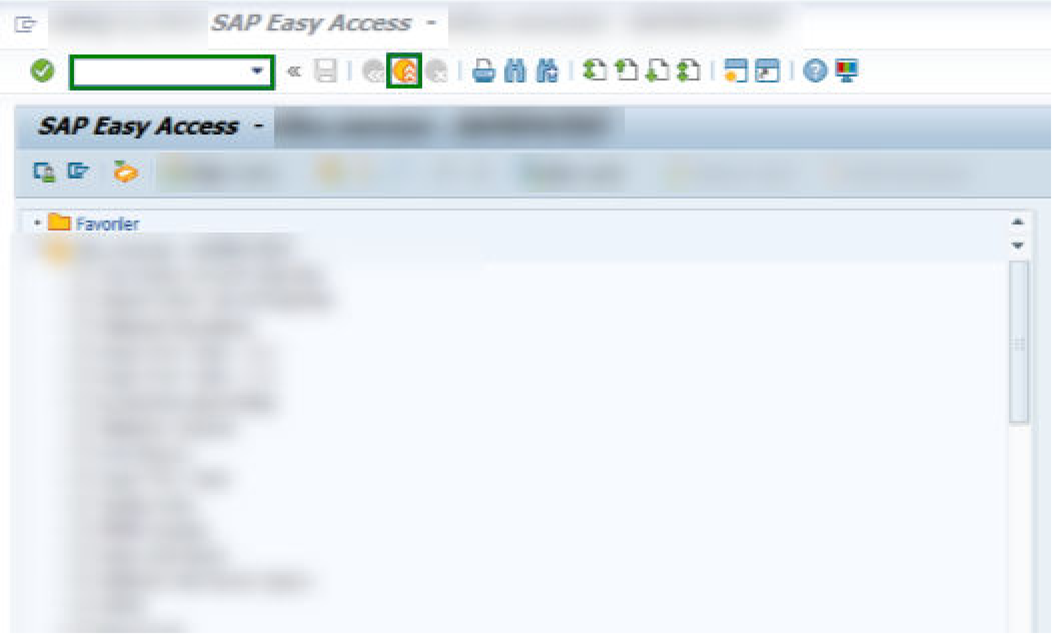
**Must/exist (passive optimization)**

Having too much must exist/must not exist elements will raise the cost of page recognition. It will have a direct impact on application availability and on the popup appearance. It is then advised to limit their number as much as possible.

**GUI Active optimization**

Despite getting rid of sub pages and having limited number of must exist and must not exist, you may still face the issue, and in that case, you will have to apply active optimization. This is done by the usage of specific activity and method in the workflow : **setBusyWaitTime**, **resetBusyWaitTime**, **lockRefresh**, and **unlockRefresh**.

- setBusyWaitTime and resetBusyWaitTime
  SAPGUI transaction may be occupied when loading pages or doing specific tasks. It is important to avoid performing an action while having a busy transaction. The Agent will always attempt to perform an action when the transaction is not busy. If the session is busy even after a declared timer, then a message session busy will appear.
  These two methods are used to Extend the Wait Period When the SAP Session is Busy
  The setBusyWaitTime method should be called in the first step of a workflow. Usually we use it on the page when handling the Easy Access page.



**SAPLOGON.pEasyAccess.setBusyWaitTime(//Time in milliseconds)**;
By default, the Busy Wait time is 1 second (1000ms) and can be modified. All the sessions opened during your process will use that time to handle every action made to the SAP GUI pages.
Each action done by the connector will check regularly up to the timer if SAP GUI is busy or not.
Always use resetBusyWaitTime when you finish your workflow as a best practice.

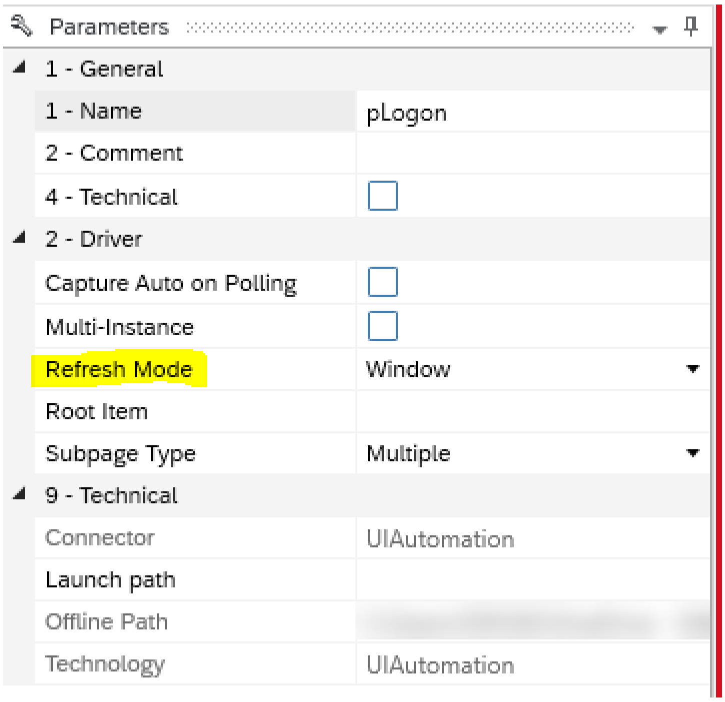
- **lockRefresh and unlockRefresh**
  UIAutomation and SAPGUI technical connector are working with polling mechanism. They will regularly refresh the state of handled application from the Agent point of view as well as for its content. The refresh is made by requesting those state directly to the application (and potentially stressing it).
  These refreshes may take time to be processed, and the polling mechanism may flood the application. When performing an action during a poll, server busy popup may occur.
  There is no way to slow down the polling mechanism with GUI technology (unlike UIAutomation).
  These two functions are used to stop the connector from refreshing the content of the page from Agent point of view. It will allow to have time to perform action while avoiding flooding handled application.
  Use it inside a wait listener when you know that the page is properly loaded
  app.page.wait(function(ev) {
  app.page.lockRefresh();
  **// >>> perform actions <<<**
  app.page.unlockRefresh();
  });
  Don’t forget the UnlockRefresh at the end of the step because the connector won’t see the next events as new page arrival.

> ⓘ Note
>
> Automatic polling may slow down every action made on a page, especially when this page cannot be optimized, has a lot of item, deep deepness, and a lot of must exist /must not exist

## Connector Wise

### UIAutomation

- If the popup is **not raised by the SDK**:
  - Try enclosing the page with lock and unlock refresh functions. Check if server busy error is not coming. (see how to do it above lockRefresh and unlockRefresh)
  - When the server busy popup comes up, wait for a few seconds to see if the pop up goes away automatically and bot execution proceeds. In that case, it may not be a blocker.
  - Deactivate automatic polling
    - In the studio, go to the page > parameters > set refreshmode= **no**.



Parameters

| | |
|---|---|
| 1 - General | |
| 1 - Name | pLogon |
| 2 - Comment | |
| 4 - Technical | □ |
| 2 - Driver | |
| Capture Auto on Polling | □ |
| Multi-Instance | □ |
| Refresh Mode | Window |
| Root Item | |
| Subpage Type | Multiple |
| 9 - Technical | |
| Connector | UIAutomation |
| Launch path | |
| Offline Path | [redacted] |
| Technology | UIAutomation |

- This mode will prevent automatic refresh of the page. Even if the page is modified (new content, new items), those updates will not be automatically be seen.
- Call mainpage.refresh or page.refresh in the code update state of the application from Agent point of view whenever you expect update from the page.

## 1.2.10 Asynchronous Events in For Loops

Sometimes you need to execute some actions several times in the workflow of your automation. Most of the time, you use a For loop directly in the code to do that. This method works well if you perform simple actions such as setting values in an Excel file or getting values from an array.

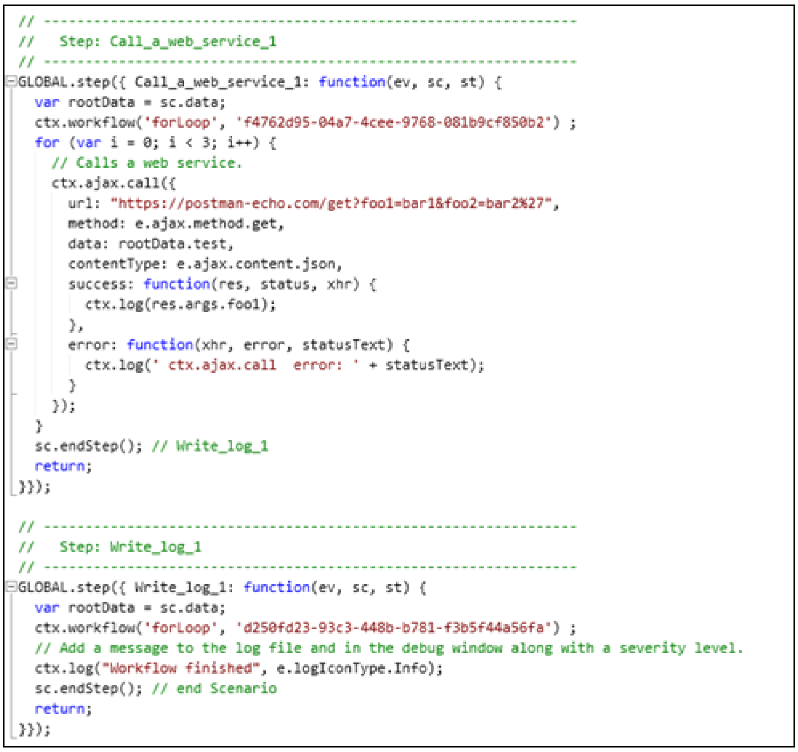
However, if you perform asynchronous tasks such as API calls or wait for elements, For loop doesn't wait till the end of asynchronous actions to execute the next iteration, and hence it doesn't work as expected. The next iteration is executed once the asynchronous action is performed. As a result, the step after the For loop can be executed before the end of all asynchronous actions. If the next step needs data from earlier asynchronous events, these data can't be set in time.

## API Calls in a For Loop

The following example demonstrates the behavior of performing API calls in a For loop:

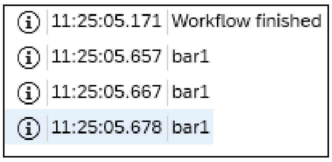
Sometimes you need to perform an API call several times. To do that, first, create your API call and surround it with a For loop, then add the next step (such as, a simple *Log* activity).

In the following code, you can expect the bot to call the API three times and print the result three times, and then print the Workflow finished log message.



```
// ----------------------------------------------------------------
//   Step: Call_a_web_service_1
// ----------------------------------------------------------------
GLOBAL.step({ Call_a_web_service_1: function(ev, sc, st) {
  var rootData = sc.data;
  ctx.workflow('forLoop', 'f4762d95-04a7-4cee-9768-081b9cf850b2') ;
  for (var i = 0; i < 3; i++) {
    // Calls a web service.
    ctx.ajax.call({
      url: "https://postman-echo.com/get?foo1=bar1&foo2=bar2%27",
      method: e.ajax.method.get,
      data: rootData.test,
      contentType: e.ajax.content.json,
      success: function(res, status, xhr) {
        ctx.log(res.args.foo1);
      },
      error: function(xhr, error, statusText) {
        ctx.log(' ctx.ajax.call  error: ' + statusText);
      }
    });
  }
  sc.endStep(); // Write_log_1
  return;
}});

// ----------------------------------------------------------------
//   Step: Write_log_1
// ----------------------------------------------------------------
GLOBAL.step({ Write_log_1: function(ev, sc, st) {
  var rootData = sc.data;
  ctx.workflow('forLoop', 'd250fd23-93c3-448b-b781-f3b5f44a56fa') ;
  // Add a message to the log file and in the debug window along with a severity level.
  ctx.log("Workflow finished", e.logIconType.Info);
  sc.endStep(); // end Scenario
  return;
}});
```

In the debugger, you can see that the last log message is printed before the results of all API calls. This behavior is due to the use of the For loop. The For loop doesn't wait till the end of the execution of the asynchronous event to perform the next iteration. After all the iterations, the *sc.endStep()* method is called, so that the next step is executed. However, this last step is executed before all API calls have finished their execution. That's the reason the order of all the logs isn't in the correct order in the debugger as expected.

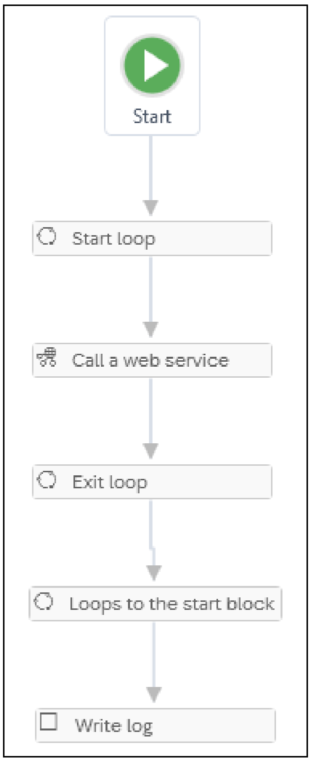


```
ⓘ 11:25:05.171 Workflow finished
ⓘ 11:25:05.657 bar1
ⓘ 11:25:05.667 bar1
ⓘ 11:25:05.678 bar1
```

## API Calls with Loop Activities

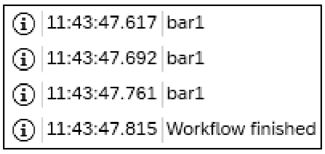
To avoid this behavior during the execution of your bot, you must use loop activities available in the Desktop Studio. These activities ensure that the next step is executed after the end of an asynchronous event.

You can directly add these loop activities to the Workflow Editor of the Desktop Studio. The previous example can be transformed as follows:



As mentioned earlier, each step is called after the previous one is executed. As a result, the *Exit loop* activity is executed once the *Call a web service* activity has finished its execution and gets the result from the API.

In the debugger, you can see that the order of all the logs is in the correct order as expected.



```
ⓘ 11:43:47.617 bar1
ⓘ 11:43:47.692 bar1
ⓘ 11:43:47.761 bar1
ⓘ 11:43:47.815 Workflow finished
```

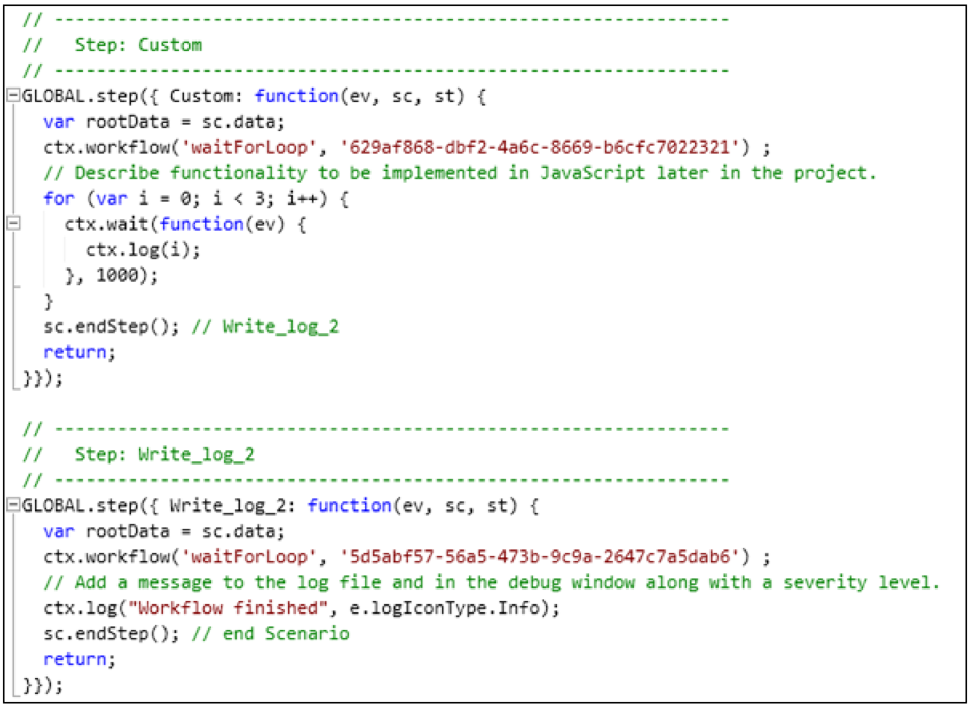
Thus, the implementation of loop activities ensures that all steps are executed in the correct order.

### Wait Activity in a Loop

The following example demonstrates the behavior of performing wait activity in a loop:

Sometimes you need to wait sometime before executing the next activity and the same behavior happens as mentioned above in *API Calls in a For Loop* section.

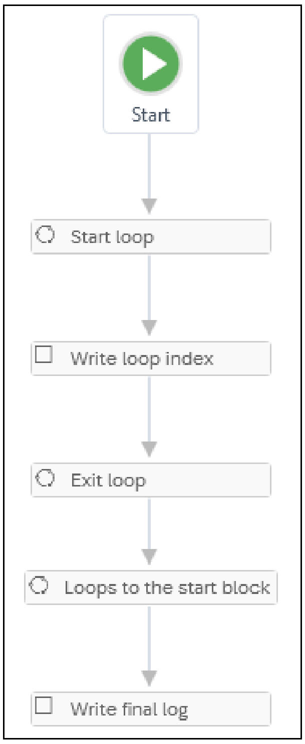
In the following code, you can see that the same error happens as *API Calls in a For Loop* section.



```
// ----------------------------------------------------------------
//   Step: Custom
// ----------------------------------------------------------------
GLOBAL.step({ Custom: function(ev, sc, st) {
  var rootData = sc.data;
  ctx.workflow('waitForLoop', '629af868-dbf2-4a6c-8669-b6cfc7022321') ;
  // Describe functionality to be implemented in JavaScript later in the project.
  for (var i = 0; i < 3; i++) {
    ctx.wait(function(ev) {
      ctx.log(i);
    }, 1000);
  }
  sc.endStep(); // Write_log_2
  return;
}});

// ----------------------------------------------------------------
//   Step: Write_log_2
// ----------------------------------------------------------------
GLOBAL.step({ Write_log_2: function(ev, sc, st) {
  var rootData = sc.data;
  ctx.workflow('waitForLoop', '5d5abf57-56a5-473b-9c9a-2647c7a5dab6') ;
  // Add a message to the log file and in the debug window along with a severity level.
  ctx.log("Workflow finished", e.logIconType.Info);
  sc.endStep(); // end Scenario
  return;
}});
```

## Wait Activity with Loop Activities

To execute all the steps in the correct order, you must change the previous code and use loop activities as follows:



Therefore, the implementation of loop activities ensures that all the index logs are printed before the final log.

# 1.3 Cloud Factory

The following Knowledge Base notes relating to the Cloud Factory are available:

- "Unable to create package. Package version already existing for 'ProjectName'" Error when trying to import a project into the Factory - 2792210
- "Factory could not be loaded insufficient_scope: Insufficient scope for this resource" - 2802932
- Nothing happens when launching an unattended scenario using Desktop Agent - 2796491
- How to hide machine or login information: 3056143

## 1.3.1 Insufficient Scope

The insufficient scope error is encountered when attempting to access the SAP Intelligent RPA Factory service.

### Context

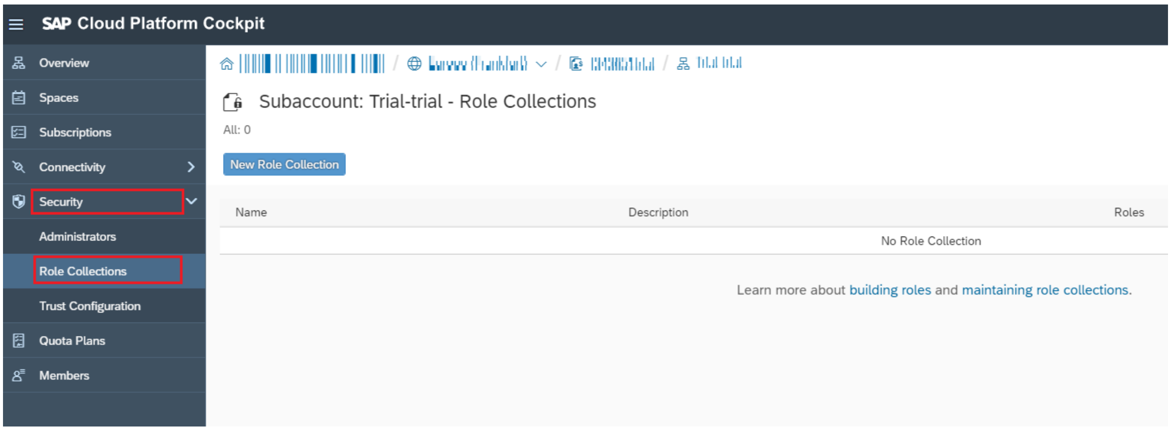
This error message means that the SAP Intelligent RPA onboarding process was not entirely completed. The correct roles were not configured and assigned.

To solve this problem, you need to add roles and users to these roles.

### Configure roles

1. In the *Security* tab of your subaccount, go to *Role Collections*.

> ⓘ Note
>
> You must have the Security Administrator role to access the *Security* tab in the SAP BTP (Business Technology Platform) Cockpit.



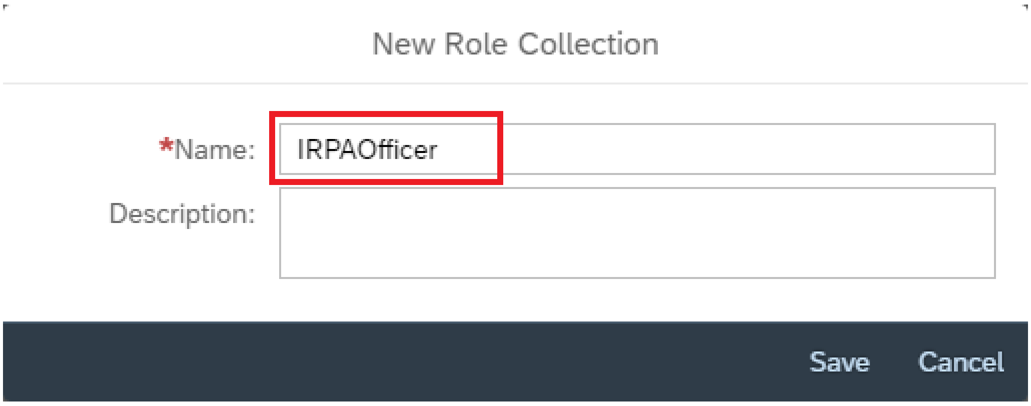
Subaccount: Trial-trial - Role Collections

All: 0

| Name | Description | Roles |
|---|---|---|
| | No Role Collection | |

Learn more about building roles and maintaining role collections.

2. Create a *New Role Collection*.

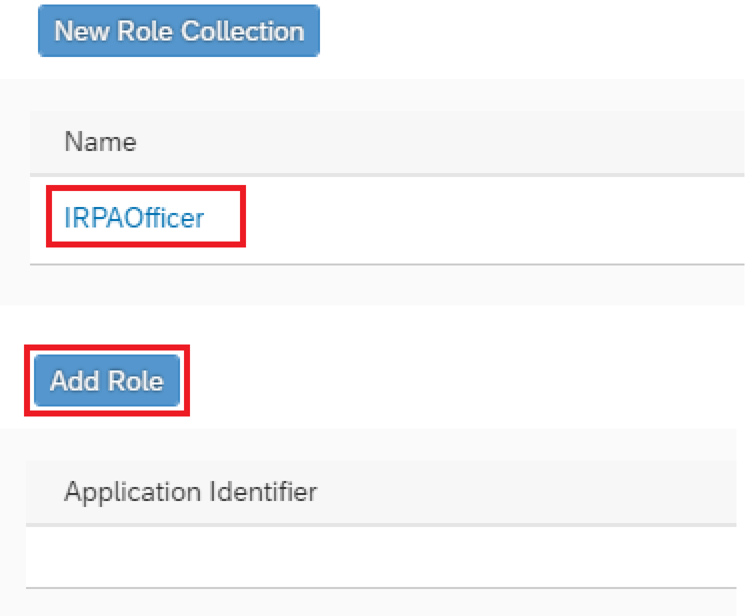


New Role Collection

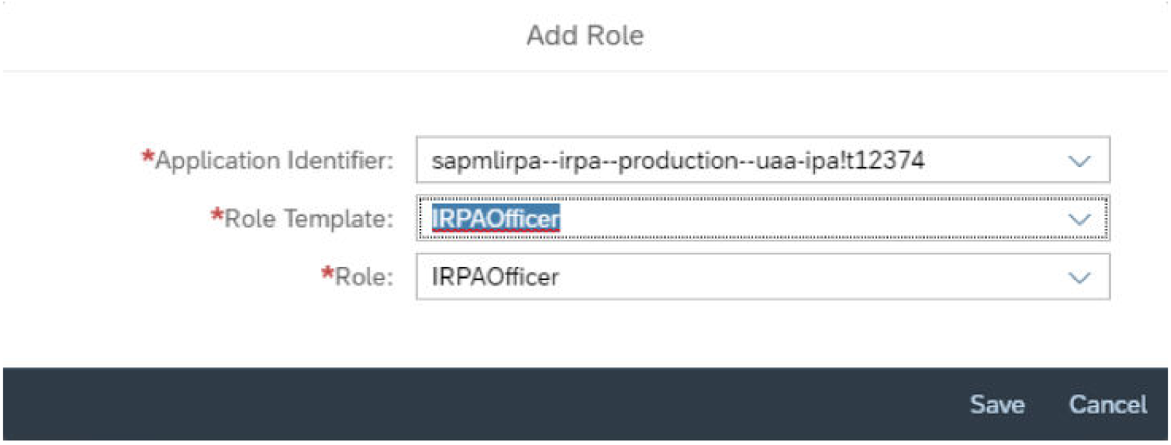
*Name: IRPAOfficer

Description:

3. Click the role collection you just created, and select *Add Role*.



4. In the *Application Identifier* dropdown list, select the *IRPAOfficer* role to create the administrator of the tenant.



Add Role

*Application Identifier: sapmlirpa--irpa--production--uaa-ipa!t12374

*Role Template: IRPAOfficer

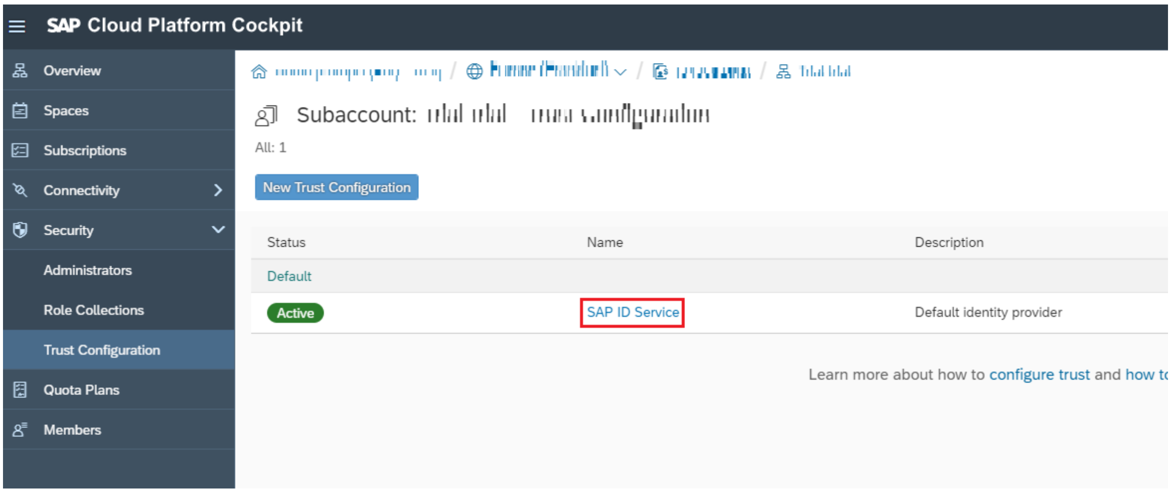
*Role: IRPAOfficer

5. Log out and log in again so that your new roles are taken into account.

## Assign roles to users

> ⓘ Note
>
> Repeat this operation for all:
>
> - The users who want to configure bots on the Cloud Factory: add the *IRPAOfficer* role to these users.
> - The users who want to run bots: add the *IRPAAgentUser* role (make sure this role is configured in the role collection).

1. Go to *Trust Configuration* and click on SAP ID Service.



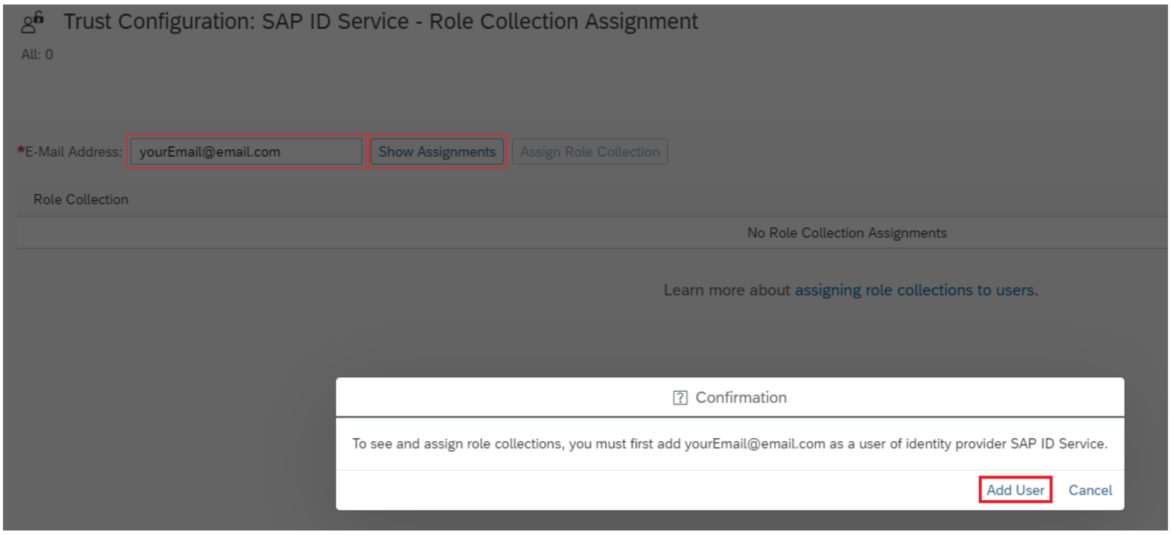
Subaccount:

All: 1

| Status | Name | Description |
|---|---|---|
| Default | | |
| Active | SAP ID Service | Default identity provider |

Learn more about how to configure trust and how t

2. Enter the e-mail address of your user, and click *Show Assignment*. If necessary click *Add User*.



Trust Configuration: SAP ID Service - Role Collection Assignment

All: 0

*E-Mail Address: yourEmail@email.com

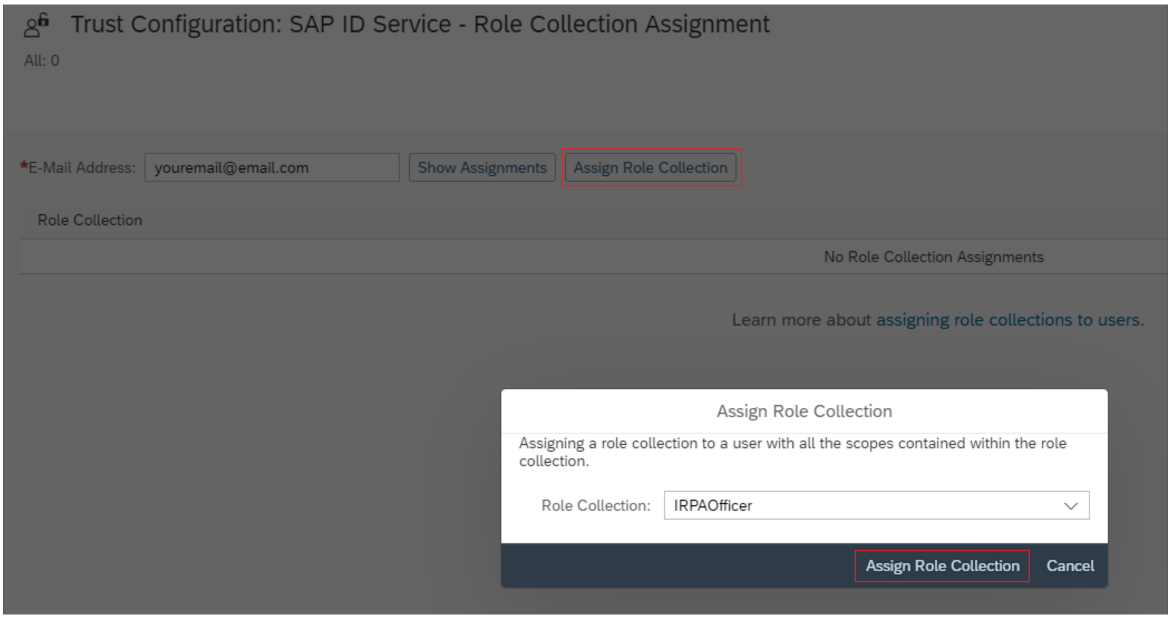
| Role Collection |
|---|
| No Role Collection Assignments |

Learn more about assigning role collections to users.

Confirmation

To see and assign role collections, you must first add yourEmail@email.com as a user of identity provider SAP ID Service.

3. Click *Assign Role Collection* and select the required role.


Trust Configuration: SAP ID Service - Role Collection Assignment

All: 0

*E-Mail Address: youremail@email.com

Show Assignments
Assign Role Collection
Role Collection
No Role Collection Assignments
Learn more about assigning role collections to users.
Assign Role Collection

Assigning a role collection to a user with all the scopes contained within the role collection.

Role Collection: IRPAOfficer

# 1.4 Cloud Studio

- Some activities are no longer editable after downgrading an SDK - 3025037
- How to configure Outlook to use SAP Intelligent RPA Outlook SDK - 3048085
- What do to if the Set Variable Value activity does not work with object sub properties -3145604

## 1.4.1 Cloud Studio and Desktop Studio Functionality Mapping

The functionalities of the Desktop Studio can be found in the Cloud Studio with the same or different nomenclature. The functionalities having similar or advanced features server the same purpose as in the Desktop Studio.

### The Context

#### Create Folder

In the Desktop Studio, Context is the data manager and used to manage all data arising during execution of a scenario. Similarly, the same functionality can be found in the Cloud Studio with a different nomenclature as Input/Output Parameters.

In the Desktop Studio, **Context > Create Folder** = In the Cloud Studio, Input/Output Parameters.

You can have the list data type. In the Cloud Studio, you must select the *List* checkbox to have the type presented as an array (for example a list).

To know more details about the managing data, refer to the Manage Data within an Automation section.

**Create Item**

In the Desktop Studio, **Context > Create Item** = In the Cloud Studio, Data Types (Predefined or Data type editor).

# 1.4.2 Set the For Each Control for Elements without Bounding Boxes

When you capture elements which don't have any bounding boxes, you cannot use the *For each* control.

## Context

When you capture a screen which have recurring elements such as several links, you can use the *For each* control. The *For each* control will apply to all recurring elements.

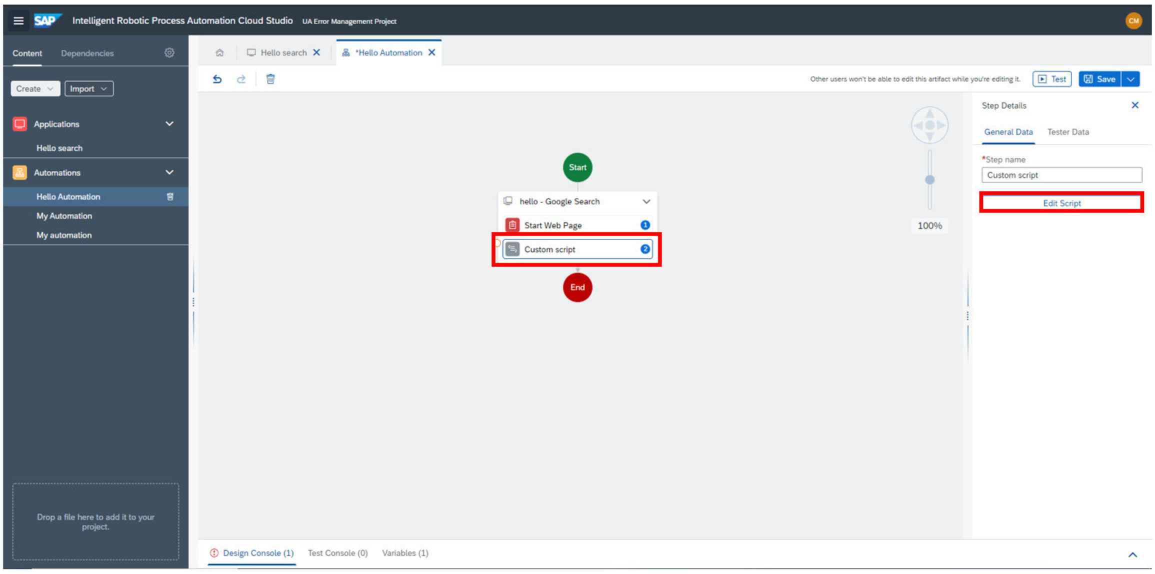
If you add activities to the *For each* control, they will apply to all the related recurring elements.

When you are working on captured screens, captured elements are sometimes not visible because they don't have any bounding boxes. Thus, you cannot use the *For each*.

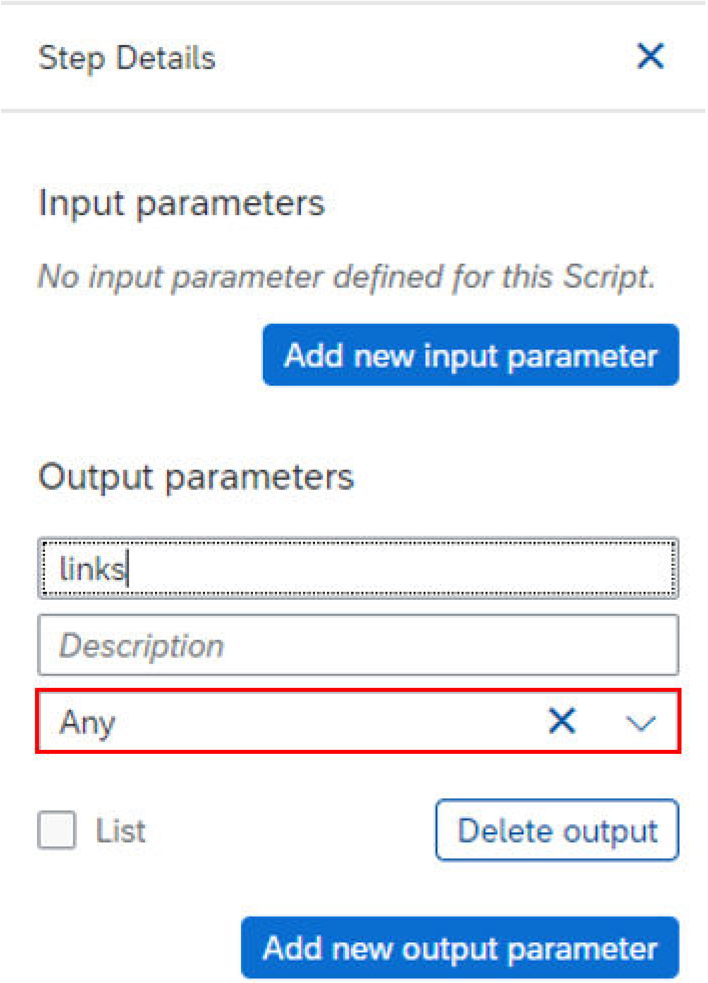
To solve this limitation, you can use the *Custom script* activity.

## Set the For each control for elements without bounding boxes

1. From the *Automation information* side panel, drag and drop a *Custom script* activity in your automation.
2. In the *Step Details* side panel, click *Edit script*.



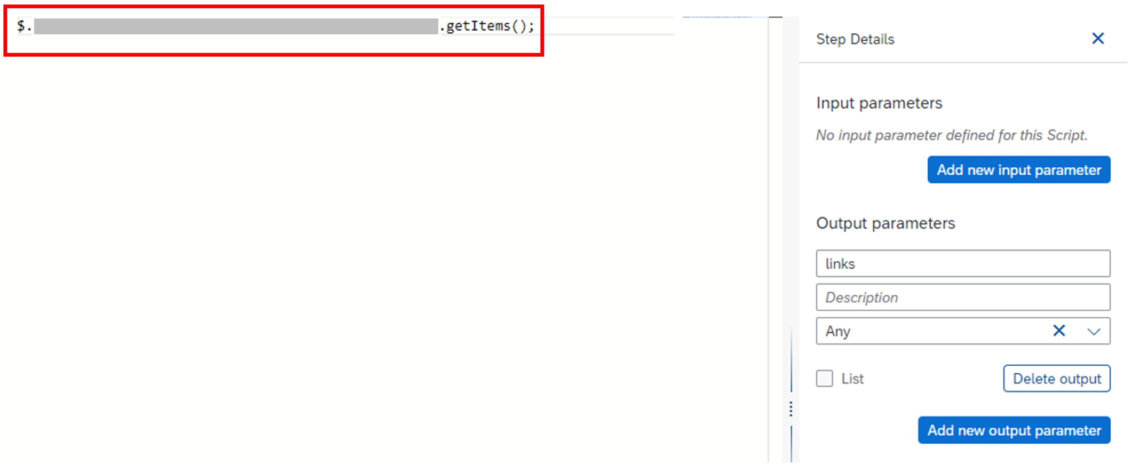
3. Select an output of type *Any*.



links

Any

4. In the central panel, add the `getItems` function at the end of the path of all the hidden elements (because they don't have any bounding boxes).

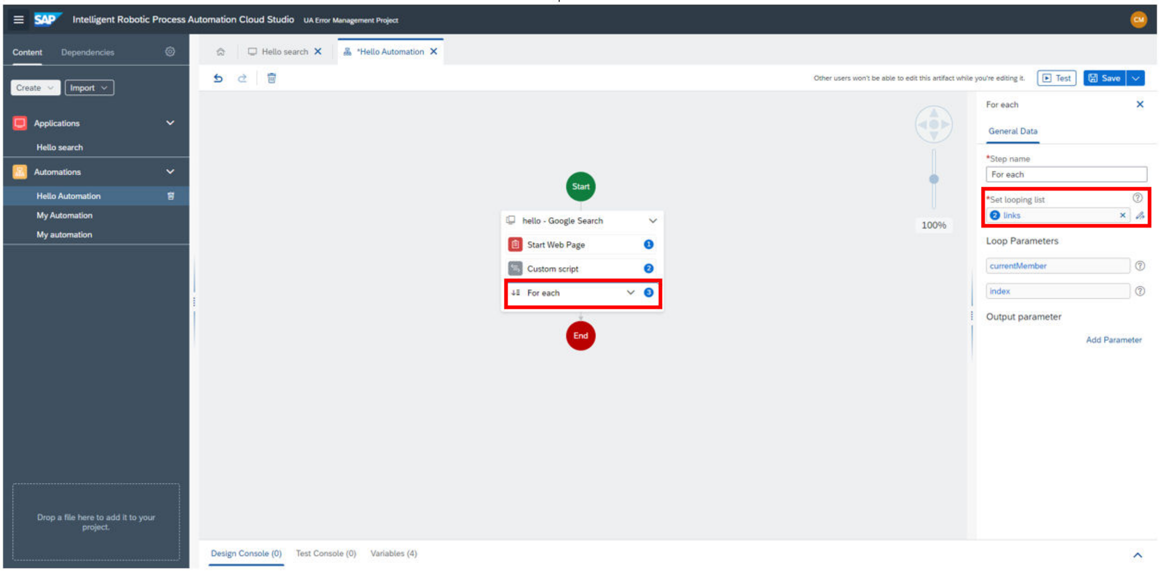


```
$.[redacted].getItems();
```

links

Any

5. Add the *For each* control to your automation.
6. In the *For each* side panel, select as input the previously set output in the *Set looping list* field.



*Step name: For each

*Set looping list: 2 links

currentMember

index

You can know use the *For each* control.

For more information, see For Each.

## 1.4.3 Unlock a Locked Session during the Automation Execution

When your bot is running in unattended mode, you sometimes need to unlock the session to perform activities. For example, keystroke and mouse click activities require to unlock the session to be performed. If you have set your Windows credentials in the Desktop Agent, the session is automatically unlocked and relocked when such activities must be performed.

> ⓘ Note
>
> It is recommended to use UI automation in scenarios where the machine does not get unlocked.

To make sure that the session unlocks and locks, you must first configure your Windows credentials in the Desktop Agent. For more information on how to store your Windows credentials, refer to Set Windows Password in Desktop Agent.

The Desktop Agent uses your credentials to automatically unlocks the session. Once the activity has been performed, the session automatically relocks 10 seconds after the action.

For more information, see How to Set Up an Unattended Bot.

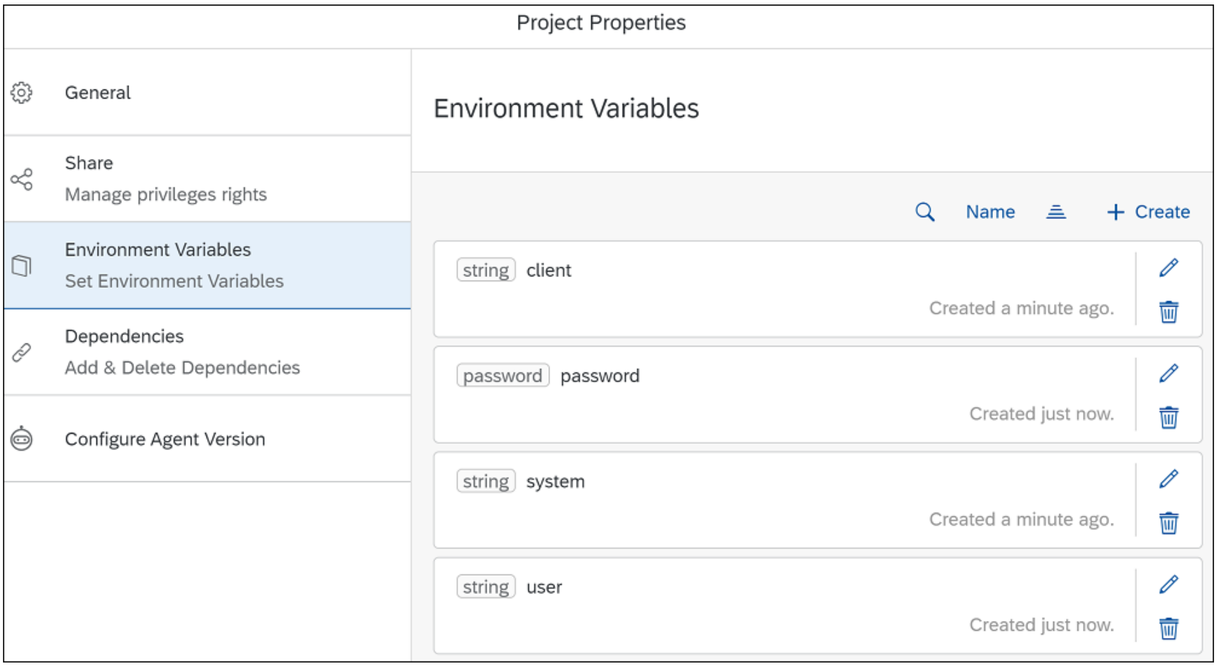
## 1.4.4 UI Application Running with Higher Privileges

Whenever you have any application that runs with higher privileges than the desktop agent (Admin mode), then the agent won't be able to interact with the application. It will be able to perform read-only activities such as highlighting, getting value of a field and so on, but however it won't be able to perform activities such as click on an element, set value to a field and so on. Therefore, you will experience technical issues if your application runs on a higher privilege such as 'Admin' mode.

## 1.4.5 Launch SAP Logon in Unattended Mode

### Prerequisites

You have created environment variables in a project, as described in Environment Variables.



Project Properties

General

Share
Manage privileges rights

Environment Variables
Set Environment Variables

Dependencies
Add & Delete Dependencies

Configure Agent Version

Environment Variables

string client
Created a minute ago.

password password
Created just now.

string system
Created a minute ago.

string user
Created just now.

You have created one or several automations in SAP Build Process Automation, as described in Create an Automation.

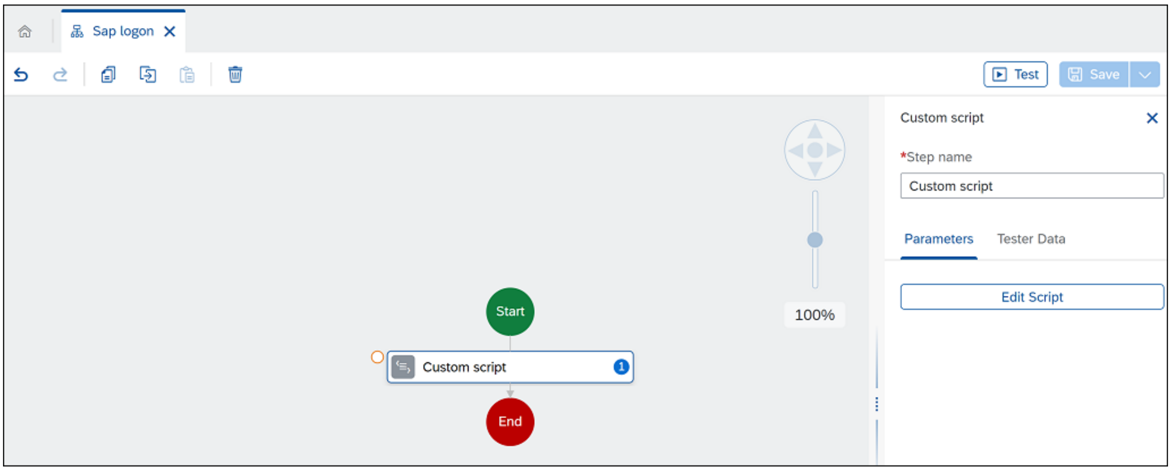
### Context

You can launch the SAP Logon application with an unattended bot using the custom script activity, environment variables, and scheduled trigger.

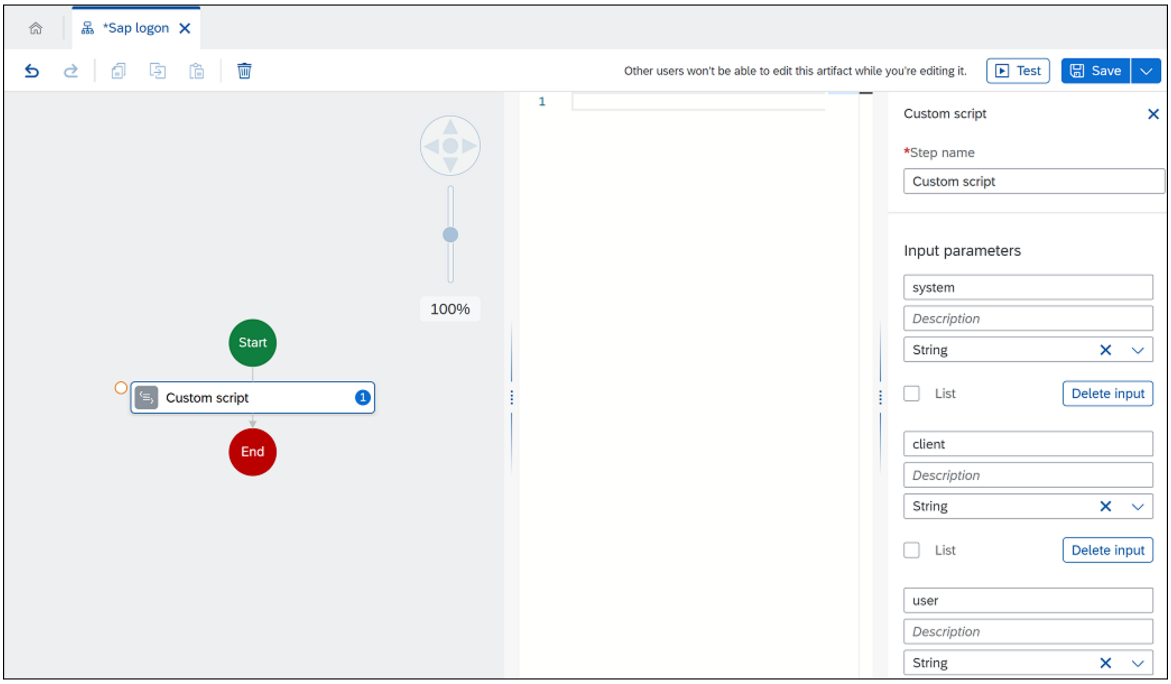
The following use case demonstrates how to launch the SAP Logon application in unattended mode.

## Procedure

1. In your automation, on the right-hand side panel, go to the *Controls* section.
2. Drag and drop *Custom Script* activity in your workflow.



3. Select *Custom Script* and click on *Edit Script*.

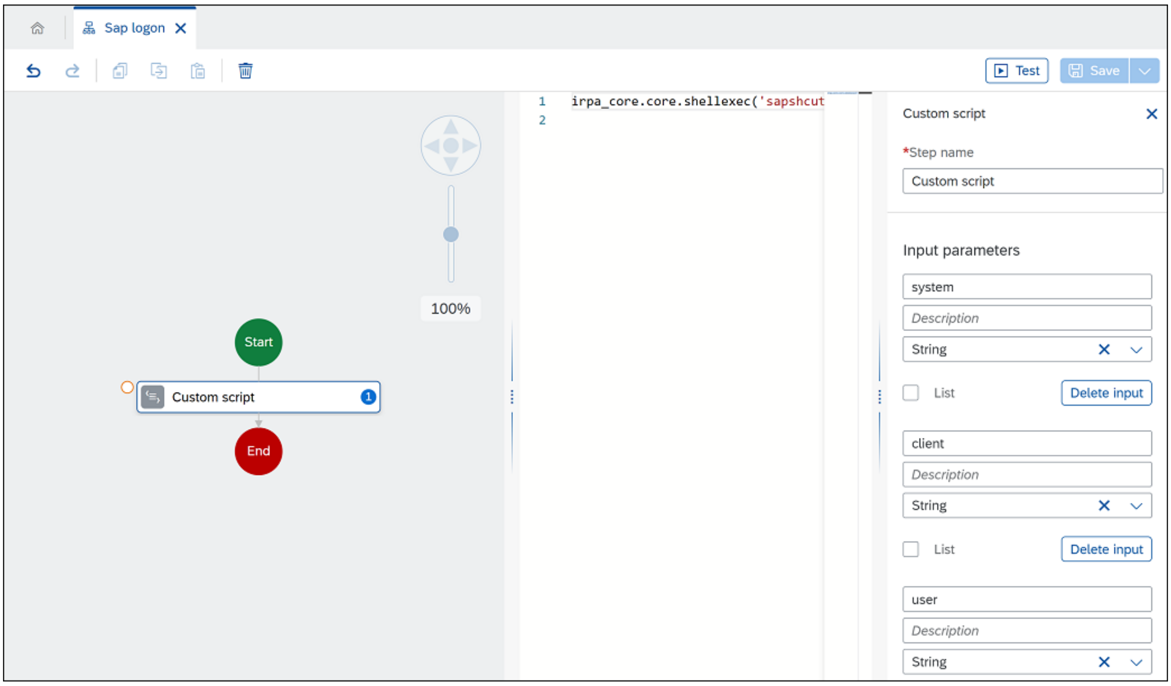


Other users won't be able to edit this artifact while you're editing it.

4. Add the variables that you want to use in the script so when you write the script, the binding between those variables and the input values are made. You need to use the *system*, the *client*, the *user*, and the *password* as input values. For the *password* input value, specify the type as “password” so that you can protect your password.
5. Write the following line inside the script:

```
irpa_core.core.shellexec(‘sapshcut.exe’, ‘-system=’ + system +  ‘ -client=’ +
client + ‘ -user=’ + user + ‘ -pw=’ + password);
```

This line makes the SAP Logon starts and logs in with the system, the client, the user, and the password you specified. You concatenate your variables to the line that gets executed on a shell by the automation.



```
irpa_core.core.shellexec('sapshcut
```

Custom script

*Step name: Custom script

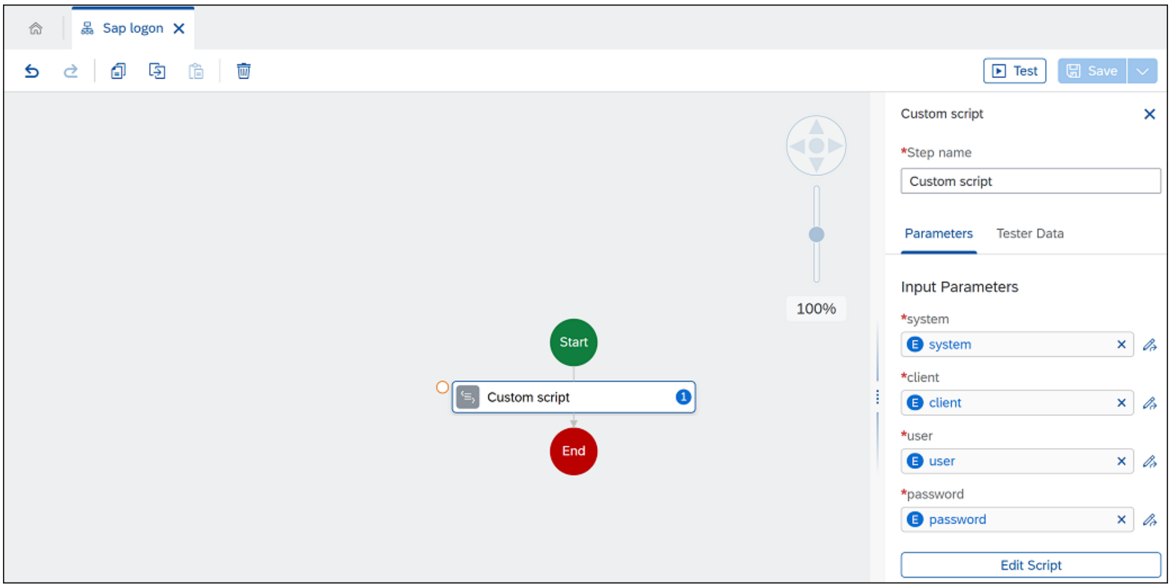
Input parameters

system — Description — String — □ List

client — Description — String — □ List

user — Description — String

6. Add the variables to the *Custom Script* activity. For every input parameter, write the name of the variable and then click on the variable that suggests the blue E before. Click on the close cross to close the edit tab.



Custom script

*Step name: Custom script

Input Parameters

*system
E system

*client
E client

*user
E user

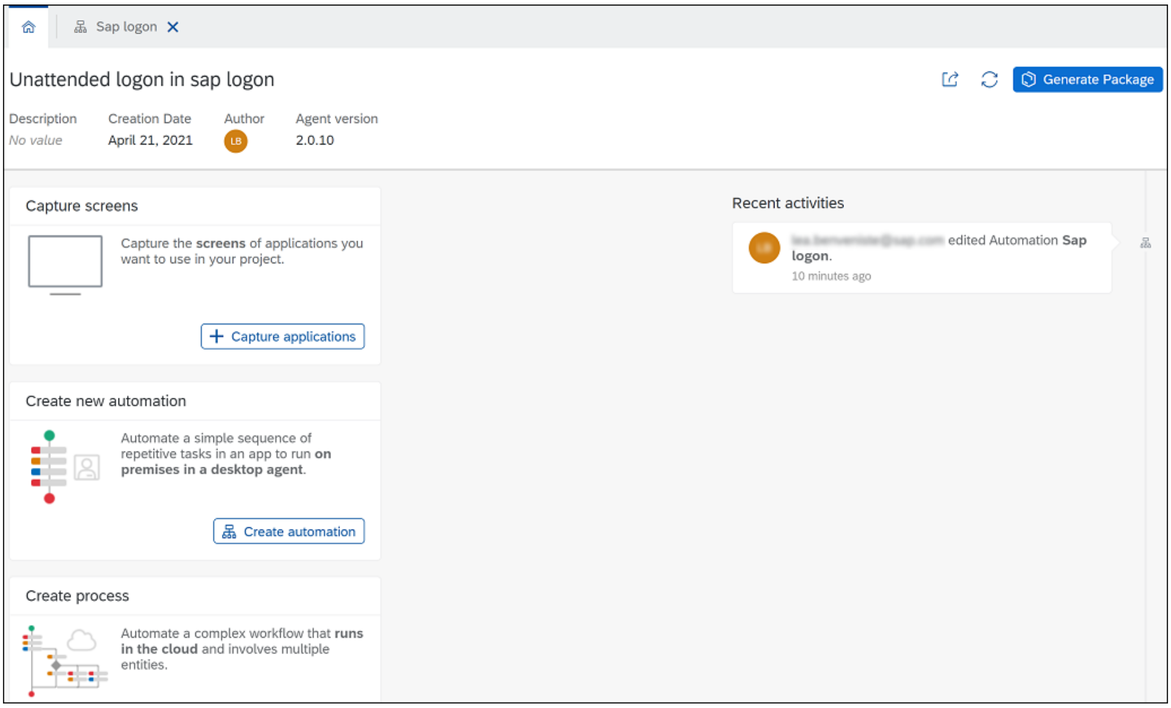
*password
E password

## Results

Now the automation to SAP logon is complete and you can setup the environment to launch this automation in unattended mode.

## Next Steps

1. Now that you finished designing your project, you need to create a package to add the project to your environment. For more information, see Generate Packages.



Unattended logon in sap logon

Description: *No value*
Creation Date: April 21, 2021
Author: LB
Agent version: 2.0.10

Capture screens

Capture the **screens** of applications you want to use in your project.

Create new automation

Automate a simple sequence of repetitive tasks in an app to run **on premises in a desktop agent**.

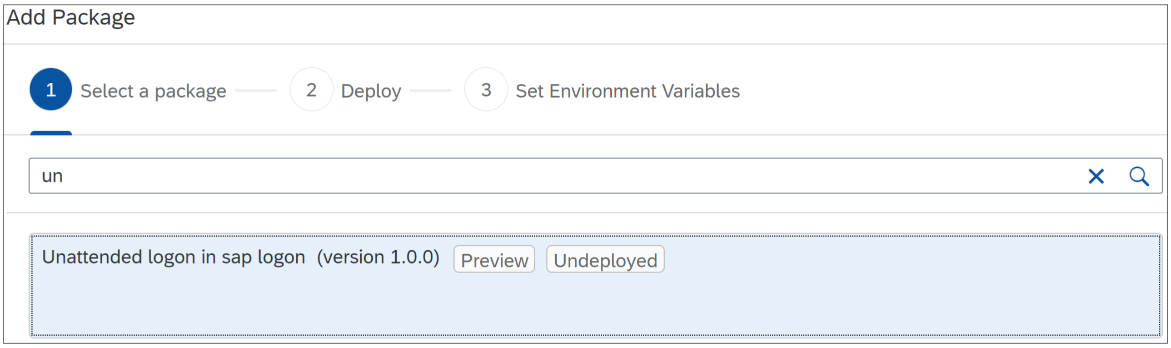
Create process

Automate a complex workflow that **runs in the cloud** and involves multiple entities.

Recent activities

[redacted] edited Automation **Sap logon**.
10 minutes ago

2. Once you generate the package, add the package to your environment. For more information, see Add Packages to an Environment.

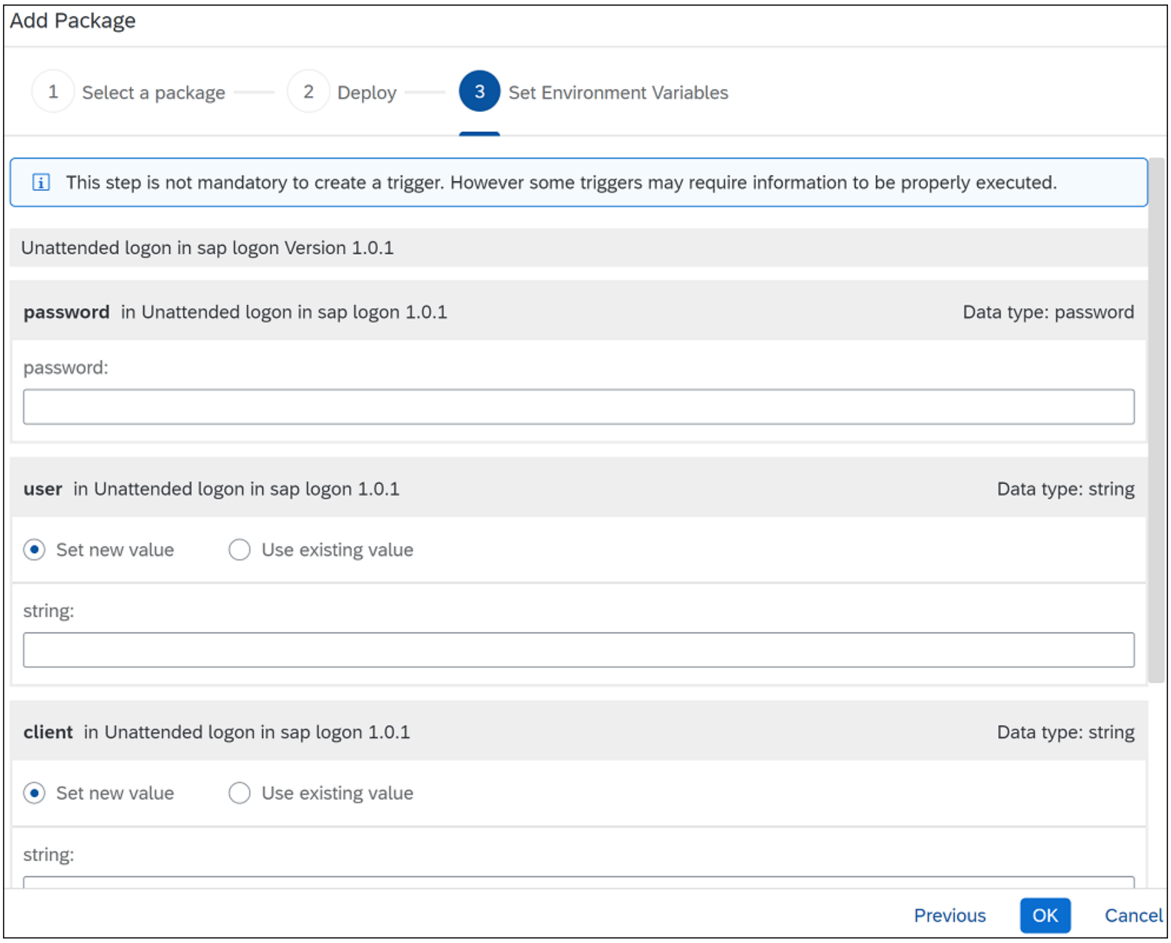


Add Package

un

Unattended logon in sap logon (version 1.0.0) Preview Undeployed

You can also set values to environment variables when deploying a package in the environment. For more information, see Environment Variables from the Cloud Studio.



Add Package

1 Select a package — 2 Deploy — 3 Set Environment Variables

This step is not mandatory to create a trigger. However some triggers may require information to be properly executed.

Unattended logon in sap logon Version 1.0.1

**password** in Unattended logon in sap logon 1.0.1 Data type: password

password:

**user** in Unattended logon in sap logon 1.0.1 Data type: string

● Set new value ○ Use existing value

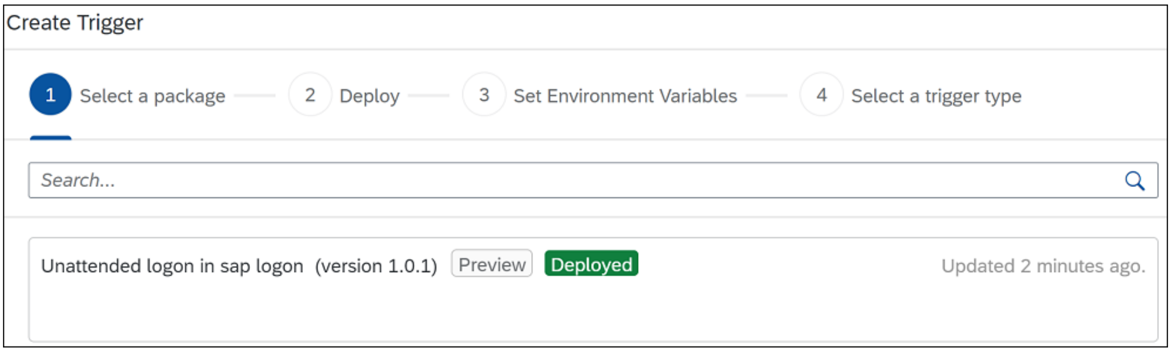
string:

**client** in Unattended logon in sap logon 1.0.1 Data type: string

● Set new value ○ Use existing value

string:

3. Add a trigger to define how and when the automation job will run. For more information, see Add Triggers to an Environment.



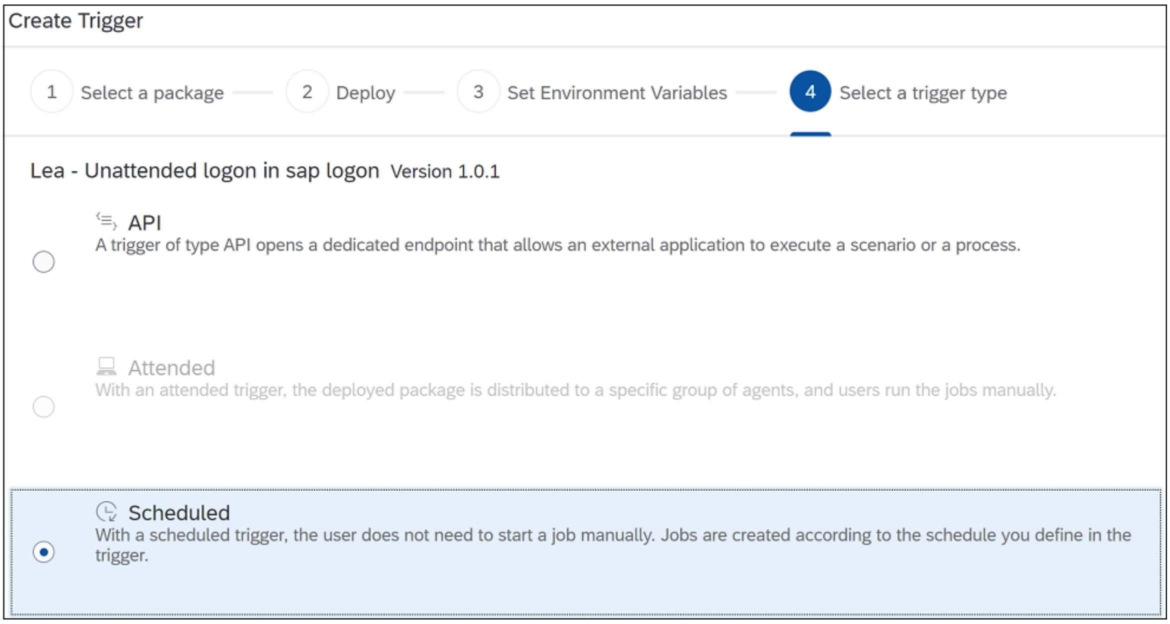
Create Trigger

1 Select a package — 2 Deploy — 3 Set Environment Variables — 4 Select a trigger type

Search...

Unattended logon in sap logon (version 1.0.1) Preview Deployed Updated 2 minutes ago.

As you want to run the automation in unattended mode, you must select the trigger type *Scheduled*. For more information, see Add a Scheduled Trigger.



Create Trigger

Lea - Unattended logon in sap logon Version 1.0.1

○ API
A trigger of type API opens a dedicated endpoint that allows an external application to execute a scenario or a process.

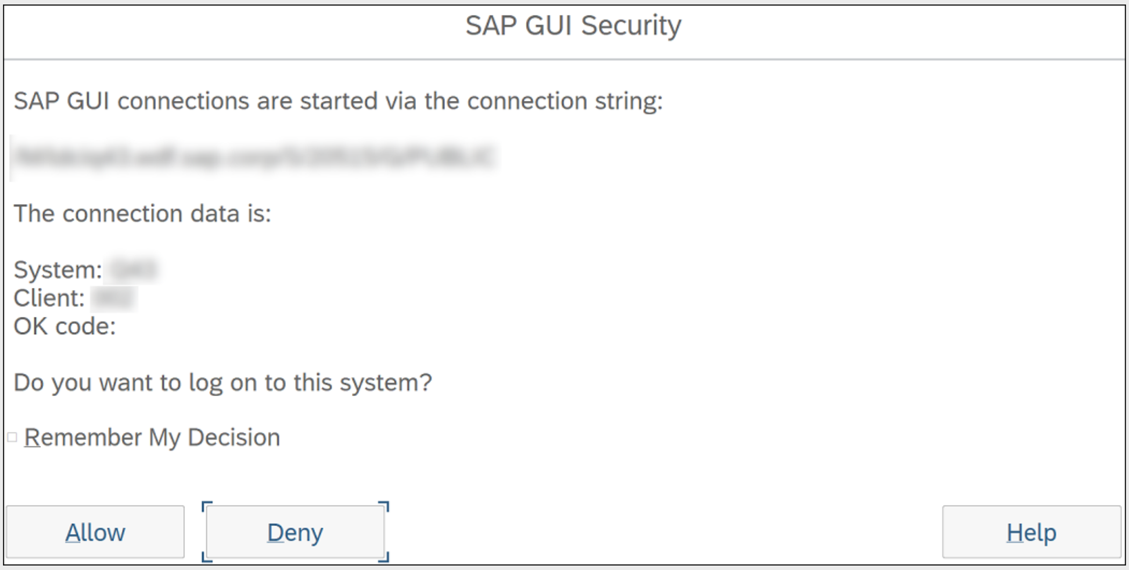
○ Attended
With an attended trigger, the deployed package is distributed to a specific group of agents, and users run the jobs manually.

● Scheduled
With a scheduled trigger, the user does not need to start a job manually. Jobs are created according to the schedule you define in the trigger.

4. Once you finalized the setup of your trigger, click ··· (*More options...*) button on the trigger tile and then select *Run now*.

### ⓘ Note

You must select your agent in *Unattended* mode directly in the agent systray. The first time you launch the bot, the following pop-up can appear.



SAP GUI Security

SAP GUI connections are started via the connection string:

[redacted]

The connection data is:

System: [redacted]
Client: [redacted]
OK code:

Do you want to log on to this system?

□ Remember My Decision

You can check the *Remember My Decision* checkbox and click the *Allow* button to not display the pop-up anymore.

## 1.4.6 Best Practices for Optimizing an Element

This section describes best practices for optimizing an element.

### Introduction

The automation of any kind of application (web, UIAutomation, and many more.) can only be done properly with a good recognition of pages and elements.

> ⓘ Note
>
> The recognition of your application and element is as important as the development of your workflow. If it is not done properly the automation may not work properly.

This includes multiple things like :

- Using the most unique criteria for each page and element.
- Ensure that the element is seen as unique by the connector (using the debugger for example).

Sometimes this is not enough, some side effects can be seen during the automation.

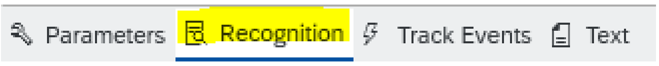
For example, if the following error message appears “Timeout waiting response from pilote” it means that you must optimize the recognition of the target element. Technically speaking, the connector send the action to the element but it takes too much time to give a response back.

### Recognition Tab

In SAP Intelligent RPA 1.0 we have the possibility to see how many tests are done by the connector to recognize a particular page or element.

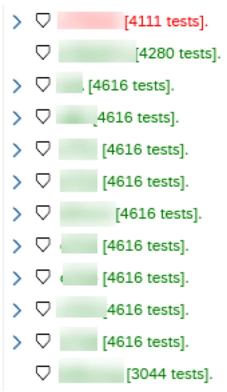
These tests can help us knowing if an optimization needs to be done or not.

In the Desktop Studio, on applications recognition side look at the right side after the parameters you should see the “Recognition” button.



By clicking on each page of an application you will see how many tests are done to recognize each element.

As you can see below, some of the elements have more than 4000 tests before being recognized.



[4111 tests].

[4280 tests].

[4616 tests].

4616 tests].

[4616 tests].

[4616 tests].

[4616 tests].

[4616 tests].

[4616 tests].

4616 tests].

[4616 tests].

[3044 tests].

## Number of Tests

What we want to do is to have less tests as possible and reach 1 test if possible.

To do so, we will have to optimize the recognition with advanced recognition capability.

> ⓘ Note
>
> Depending on the connector, it's not always necessary to optimize when you see a big number of tests.

Web: This connector is fast and having 500 – 1000~ is not a big number of tests, the automation can work fast and properly.

SAPGUI: When using an ID / name as a criterion for an element (having like 3000 tests is not an issue in this case as the connector already benefit of an internal optimization).

There is no "good" range of tests, it depends on the technology. As discussed above for web it can be 1000 and still working fine but for UIAutomation it would be too much.

For any other situation, the usage of advanced recognition must be implemented.

It's not mandatory at all but if you see a slow automation for example and you want to boost it, optimize the recognition is probably the best way to achieve it.

> ⓘ Note
>
> : There is a compromise between having less tests and the number of criteria used to recognize an element

For example, you will have a low number of tests by using the Items Pattern Method However, by doing this if the application's DOM change then you will have to update the criteria again.

It must be known by the developer before doing any development.

## Number of Elements in a Page

The optimization is also related to the number of elements recognize in your page.

There is not a maximum of element per page, but the good practice would be to have only useful elements:

- Interaction with inputs during the automation
- Elements used for the advanced recognition (Ancestor, Must exist, Must not exist, and many more)

Keep it as small as possible as the connector will test, in order, each element.

If the connector test pointless element before the element you want to interact with you will lose some precious time.

### Conclusion

As we saw together, when we talk about recognition optimization, you need to look at the recognition panel.

There is no “good” number of tests, each connector is not working the same regarding the recognition.

If an element needs to be optimized follow the advanced recognition methods.

As a best practice keep only the useful elements. Keeping an element which is not used during the automation or not used to recognize another element will slow the entire recognition of your page.

Now you know more about the optimization and the way to verify if an application is well recognized.

You will find more information about the recognition panel by following this link to the official documentation.

## 1.4.7 How Tests Work

Each time you perform a local test with Cloud Studio and Desktop Agent, several things are happening behind the scenes that could take some time during the initial test:

- The Agent will need to restart in "test mode" to enable the right communication channel between **Cloud Studio**, **Browser extension** and **Studio bridge**
- SDK packages added within your project will be:
  - downloaded from Cloud Studio
  - unpacked
  - loaded in the Agent memory
- Dependent packages (if any) will be downloaded, unpacked, and loaded whenever needed
- Project itself will be downloaded, unpacked and loaded in Agent memory
- As the content is downloaded from Cloud Studio and processed on Desktop Agent, the resources might be checked with regular antivirus or antimalware processes behind the scenes too.

Whenever you hit the test button without making any changes in the project, all resources are already within Desktop Agent and the test start instantly.

Starting with the version 2.0.20 of the Agent, improvements have been introduced:

- Prior this version, any change of the project stakeholders requires the agent to restart entirely.
- With this version, a change in an automation will not require the whole restart of the Agent. Waiting time between two tests will be significantly reduced.

However, to ensure project stability and integrity:

- If more than ten artifacts have been updated since the last execution, the Agent will restart entirely.
- Changes of other artifacts (such as Application Declaration) will still require the Agent to restart.

## 1.4.8 Execute an Automation from Another Automation

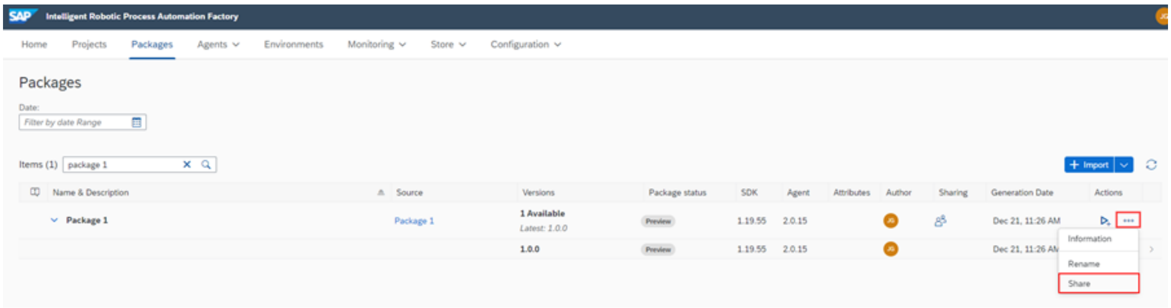
You can execute a given automation during the execution of another within the Cloud Studio. There are two possibilities for this, either using a direct dependency, or an API trigger.

### Using a direct dependency

This is the recommended option as it is the easiest to implement.

For example, let's say that you want to execute 'autoamtion 1' from 'package 1' in 'automation 2'.

1. Go to the *Packages* tab in the Cloud Factory
2. Find your 'package 1' and share it by clicking on the •••

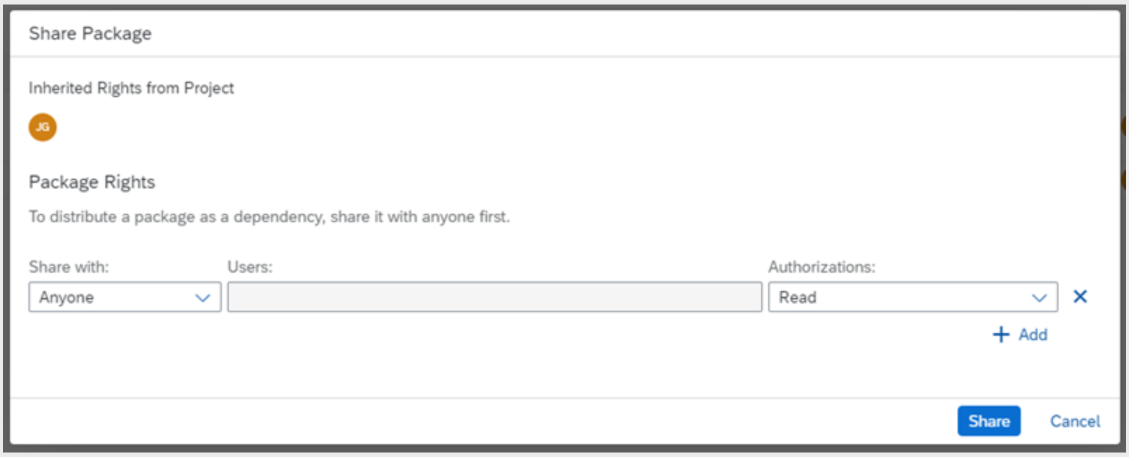


Packages

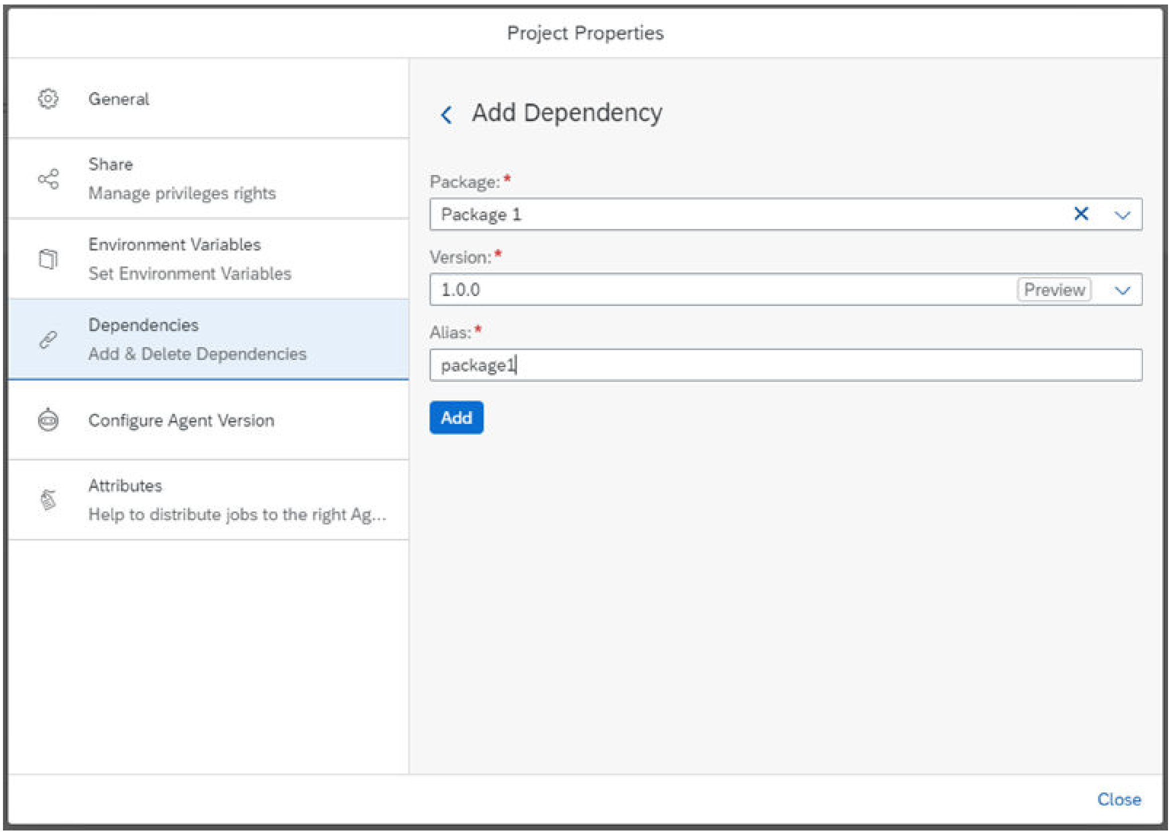
Date: Filter by date Range

Items (1) package 1

| Name & Description | Source | Versions | Package status | SDK | Agent | Attributes | Author | Sharing | Generation Date | Actions |
|---|---|---|---|---|---|---|---|---|---|---|
| Package 1 | Package 1 | 1 Available<br>Latest: 1.0.0 | Preview | 1.19.55 | 2.0.15 |  | JG |  | Dec 21, 11:26 AM | ••• |
|  |  | 1.0.0 | Preview | 1.19.55 | 2.0.15 |  | JG |  | Dec 21, 11:26 AM |  |

> ⓘ Note
>
> You need to share your package with *Anyone* choosing *Read* as the authorization as shown below:
>
> 
>
> Share Package
>
> Inherited Rights from Project
>
> JG
>
> Package Rights
>
> To distribute a package as a dependency, share it with anyone first.
>
> Share with: Anyone
>
> Users:
>
> Authorizations: Read
>
>

3. Open 'package 2' in the Cloud Studio and add 'package 1' as a dependency. Be sure to select the correct package version.



Project Properties

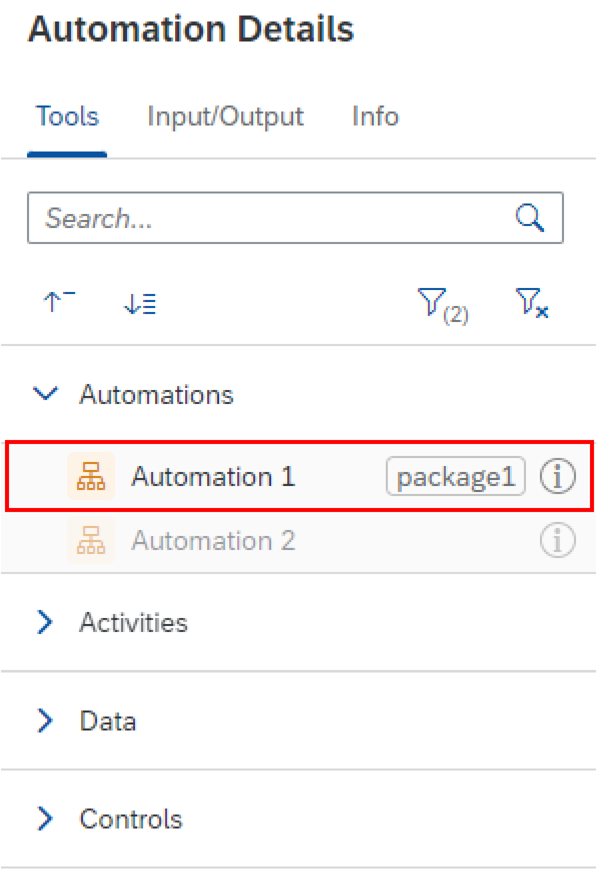
Add Dependency

Package: * Package 1

Version: * 1.0.0

Alias: * package1

4. The first package will now be available to execute in the list of automations in your project. You can then decide to execute the automations synchronously, or to execute it at later a time.



For more information see the page on Dependencies.

## Using an API Trigger

This option is more complex. This method works by adding a new job to the queue in the Cloud Factory. Thus, it will be executed later in your environment, but it won't be possible to execute synchronously.

You will need to create a specific API trigger, and then perform a call to a web service in order to execute the automation.

For more information see the page on how to add an API trigger and Execute an API Trigger.

# 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 an 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

THE BEST RUN SAP