[Browser and web automation (robocorp.com)](https://robocorp.com/docs/development-guide/browser)

# Desktop automation and RPA

Despite the ever-growing popularity of [web-based applications](https://robocorp.com/docs/development-guide/browser) in the workplace, many business processes still involve desktop applications for reasons of legacy, security, or hardware needs. Being able to control these types of applications programmatically opens up a world of automation possibilities.

Desktop automation allows your robot to accomplish tasks acting like a human operator, directly controlling a desktop interface. This includes operations like opening and closing applications, simulating mouse movements and clicks, triggering keyboard keys and shortcuts, taking screenshots.

Compared to browser automation, desktop automation is a more varied and complex field. However, the main idea stays the same across operating systems and access methods: you need a way to point the robot to specific parts of the desktop screen, and once a target is identified, the robot can be instructed to interact with it by clicking on it, typing on it, dragging it, etcetera.

The Robocorp stack supports different desktop automation approaches, which differ mostly in how the "targets" for the robots are identified and managed.

**RPA.Desktop**: Desktop is a cross-platform library for navigating and interacting with desktop environments. It can be used to automate applications through the same interfaces that are available to human users. For image-based desktop automation, you should use the cross-platform RPA.Desktop library.

* Mouse and keyboard input emulation
* Starting and stopping applications
* Finding elements through image template matching
* Scraping text from given regions
* Taking screenshots
* Clipboard management

**~~RPA.Desktop.Windows~~**~~: deprecated and will be no longer maintained~~

**RPA.Windows**: RPA.Desktop.Windows를 대체

* available via **rpaframework-windows** package (rpaframework-windows=2.0.0 필요?)

## Image template matching based automation

In image template-based desktop automation, you provide the robot with screenshots of the interface's parts that it needs to interact with, like a button or input field. The images are saved together with your automation code. The robot will compare the image to what is currently displayed on the screen and find its target.

Using this same technique, you can also find a specific part of the interface on the screen and then add an offset in pixels, telling the robot, for example, to "click 200 pixels on the right" of the image that you are providing.

This technique enables automating environments like **Citrix** and other remote terminals where you don't have access to the target machine itself, but effectively only to a "video stream" of the desktop.

Our [VS Code extensions](https://robocorp.com/docs/developer-tools/visual-studio-code/overview) provides a set of UI tools to take, manage, and store image-based locators in your automation projects.

### Common challenges of image locator-based desktop automation

When using this approach, these are some of the challenges you should be aware of:

* **System settings can impact the recognition of the images**: How the interface elements look on a screen depends on system settings like color schemes, transparency, and system fonts. Images taken on a system might end up looking different than the target system, and the robot might not recognize them, stopping the process.
* **Screen resolution is a factor**: A different screen resolution might cause elements on the screen to move around or change in size.
* **Different versions of the same operating system can differ visually**: Operating systems provide the general guidelines of how the interface elements are drawn on the screen. If the operating system is updated, image templates might stop being recognized.

To mitigate this type of issues and make your automation less fragile, we recommend:

* sticking to default settings for fonts and colors
* using accessibility options to reduce visual effects like shadows and transparencies
* if possible, using the target machine to take the locator images to ensure that all settings are the same.

## Textual locator-based desktop automation

### Application UI items identifiers

If you are automating a Microsoft Windows application, instead of using images, you can try to target the actual UI elements within it, referring to them by their identifiers.

To get to the identifiers, you can [**inspect the running application using Accessibility Insights**](https://robocorp.com/docs/development-guide/desktop/how-to-find-user-interface-elements-using-locators-and-keyboard-shortcuts-in-windows-applications#inspecting-windows-applications-with-accessibility-insights)**.**

[**Accessibility Insights Downloads**](https://accessibilityinsights.io/downloads/)

Microsoft recommends [Accessibility Insights](https://accessibilityinsights.io/) for viewing the UI automation properties. Legacy tools such as [Inspect.exe](https://docs.microsoft.com/en-us/windows/win32/winauto/inspect-objects) can also be used.

**Web**: Accessibility Insights for Web helps developers find and fix accessibility issues in web apps and sites. This browser extension for Chrome and the new Microsoft Edge runs on Windows, MacOS, and Linux computers.,

Chrome, Edge에 extension 설치 완료

**Windows**: Accessibility Insights for Windows helps developers find and fix accessibility issues in Windows apps. This desktop app runs on Windows computers..

Accessibility Insights for Windows (from MS) 설치 완료

**Android** (생략)

**MiPlatform Browser:**

ActiveX 타입 및 exe 타입

**Exe 타입**(‘전용브라우저’라 힘) 데스크탑 애플리케이션과 유사해 보여, 어려울 듯,

Accessibility Insight로는 어떨지?

한국투자증권 사례

**ActiveX 타입**(기존 브라우저 활용)은 그래도 가능하지 싶음.

Exe 타입의 전용 브라우저 내용 전체를 component화하여 브라우저에 표시

url 표시되므로 scraping하면 될 수 있겠음

[**Accessibility Insights 사용법**](https://accessibilityinsights.io/downloads/)

<https://www.youtube.com/watch?v=BIu9ONdMGGg> 참조

Live inspect: allow to verify the element in an app have the right UI properties simply by hovering over them. The elements are displayed in UIA tree. Each element control type and name are prominently visible along with the properties that are most relevant for the control type.

Alternatively, you can see all properties that have values or choose the specific properties you want to see.

Fast Pass: helps developers find common high-impact accessibility issues in less than 5 minutes. Recommended running a Fast Pass prior to each code check-in. FastPass begins with automated checks. Within seconds, Accessibility Insights checks for compliance with dozens of accessibility requirements.

Trouble shooting:

## OCR locators

Another available option is to create locators using OCR ([Optical Character Recognition](https://en.wikipedia.org/wiki/Optical_character_recognition)). Using this approach, you can find elements on the screen by their textual content. For example, you could find the "Send" button by telling the robot to click wherever on the screen the "Send" text appears.

This approach is similar to the image template-based one and shares some of the same weaknesses:

* You have to make sure that the text appears only once on the screen, so choose your targets wisely.
* Test that the OCR engine can find and correctly recognize the text: it might be "defeated" by system settings like opacity, shadows, low contrast, etc.

In general, OCR locators are quite fragile and should be used as a last resort. The primary use case for OCR capabilities in desktop automation is reading text information from specified screen regions.

## Using keyboard shortcuts

In most desktop applications, a lot can be accomplished by using keyboard shortcuts, and since we can control the keyboard, we can use them directly. Check the documentation for the application you are automating to see what is available.

Check this [Windows desktop application robot](https://robocorp.com/docs/development-guide/desktop/windows-desktop-application-robot) article for an example.

Here's a video of automating a video game using keyboard shortcuts:

## Which libraries should you use?

The RPA Framework set of open-source libraries that Robocorp develops supports all of the approaches we talked about.

For image-based desktop automation, you should use the cross-platform [RPA.Desktop](https://robocorp.com/docs/libraries/rpa-framework/rpa-desktop) library, which provides:

* Mouse and keyboard input emulation
* Starting and stopping applications
* Finding elements through image template matching
* Scraping text from given regions
* Taking screenshots
* Clipboard management
* OCR selector support

For textual locator-based desktop automation using identifiers in Microsoft Windows, you can use the [RPA.Desktop.Windows](https://robocorp.com/docs/libraries/rpa-framework/rpa-desktop-windows) library, which also provides Windows-specific keywords to open and close applications.

## Additional options for macOS automation

If you are automating macOS applications, in addition to the functionality provided by the RPA.Desktop library, you can take advantage of the automation capabilities that are included with the operating system itself. For example, using the [Run keyword from the Operating system Robot Framework library](https://robocorp.com/docs/libraries/built-in/operatingsystem/keywords#run) which allows you to run any arbitrary command on your Mac, you can run [AppleScript](https://developer.apple.com/library/archive/documentation/AppleScript/Conceptual/AppleScriptLangGuide/introduction/ASLR_intro.html) instructions via the [osascript](https://ss64.com/osx/osascript.html" \t "_blank) command, or trigger [Automator](https://support.apple.com/en-gb/guide/automator/welcome/mac) workflows.

## Desktop automation in Control Room

Once you have your desktop automation working locally and in the target system, you are ready to benefit from the orchestration and hosting features of [Control Room](https://robocorp.com/docs/control-room).

Control Room provides you a centralized place to manage your robot code. You can then [trigger robots, get reports and traceability, set up access control](https://robocorp.com/docs/control-room), and many other features.

Particularly relevant to desktop automation is [Robocorp Workforce Agent](https://robocorp.com/docs/control-room/configuring-workforce/overview). Once you have identified the machine that the automation will run (it could be the same computer you used for development, or, more likely, another physical or virtual machine), just [install Robocorp Workforce Agent](https://robocorp.com/docs/control-room/configuring-workforce/installation) on it, and you will be able to [trigger and schedule the robot on that machine from Control Room](https://robocorp.com/docs/control-room/configuring-workforce/running-process-locally).

Also, by using the [Assistant](https://robocorp.com/docs/control-room/operating-assistants) of Control Room, you can set up attended desktop automation workflows where robots and human operators can work together in accomplishing a task.

## Desktop automation robot examples

Here are complete robot examples demonstrating desktop automation with the Robocorp stack:

* [Simple Windows Calculator Robot](https://robocorp.com/portal/robot/robocorp/example-windows-calculator)  
  A very simple robot that just interacts with Windows 10 Calculator in different ways.
* [Windows Desktop App Robot](https://robocorp.com/docs/development-guide/desktop/windows-desktop-application-robot)  
  This software robot opens the Spotify desktop application, searches for the given song, and plays the song. The robot demonstrates the basic Windows-automation capabilities of the RPA Framework, using keyboard navigation.
* [Travel Directions Desktop Automation Robot On Mac OS](https://robocorp.com/portal/robot/robocorp/example-desktop-image-template-matching)  
  This example robot demonstrates the use of image templates and keyboard shortcuts to find travel directions between two random locations on Earth using the macOS Maps application. Also, it demonstrates the use of desktop automation and [browser automation](https://robocorp.com/docs/development-guide/browser) combined in one robot.
* [Desktop Automation With Image Recognition And OCR](https://robocorp.com/portal/robot/robocorp/example-desktop-image-ocr)  
  This robot demonstrates automating a desktop application with image recognition and OCR, interacting with the open-source accounting software GnuCash.

Learn more about the libraries mentioned on this page:

* [RPA.Desktop](https://robocorp.com/docs/libraries/rpa-framework/rpa-desktop)
* [RPA.Desktop.OperatingSystem](https://robocorp.com/docs/libraries/rpa-framework/rpa-desktop-operatingsystem)
* [RPA.Desktop.Windows](https://robocorp.com/docs/libraries/rpa-framework/rpa-desktop-windows)

# [How to find user interface elements using locators and keyboard shortcuts in Windows applications (robocorp.com)](https://robocorp.com/docs/development-guide/desktop/how-to-find-user-interface-elements-using-locators-and-keyboard-shortcuts-in-windows-applications#inspecting-windows-applications-with-accessibility-insights)

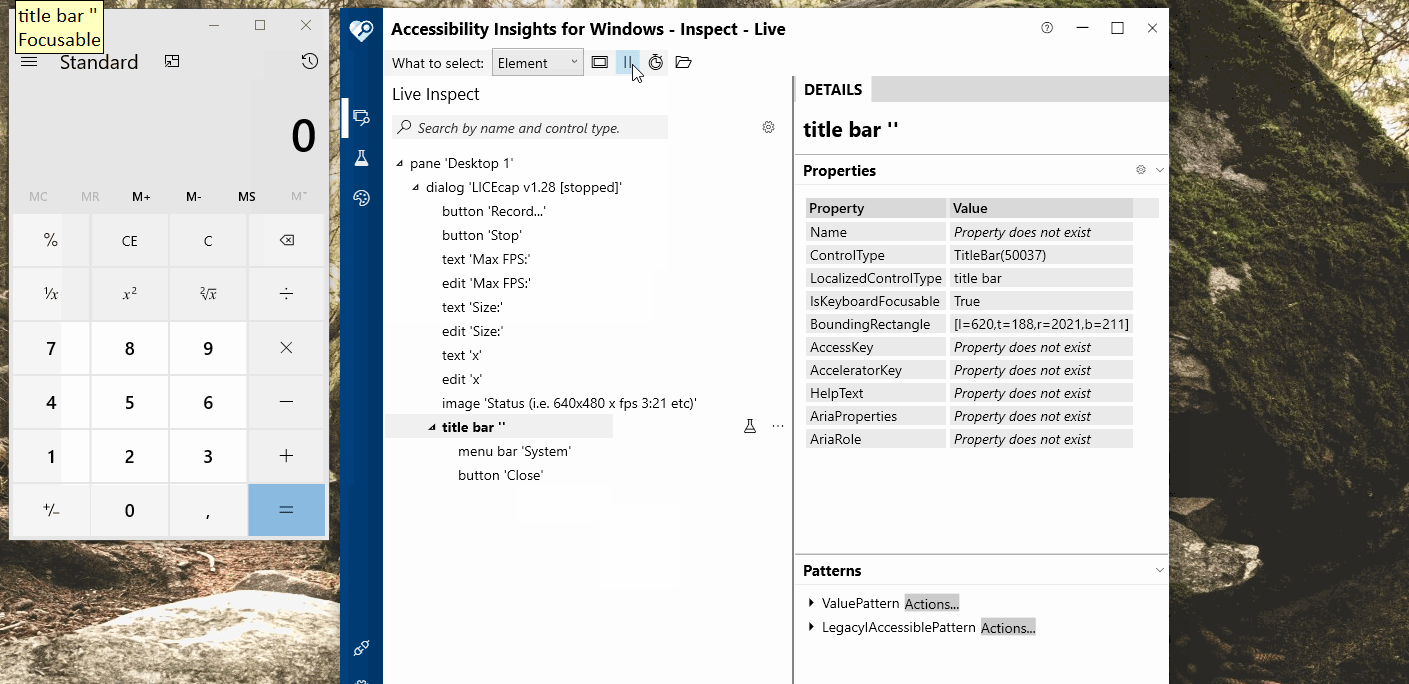
# How to find user interface elements using locators and keyboard shortcuts in Windows applications

## Available tools for inspecting Windows applications

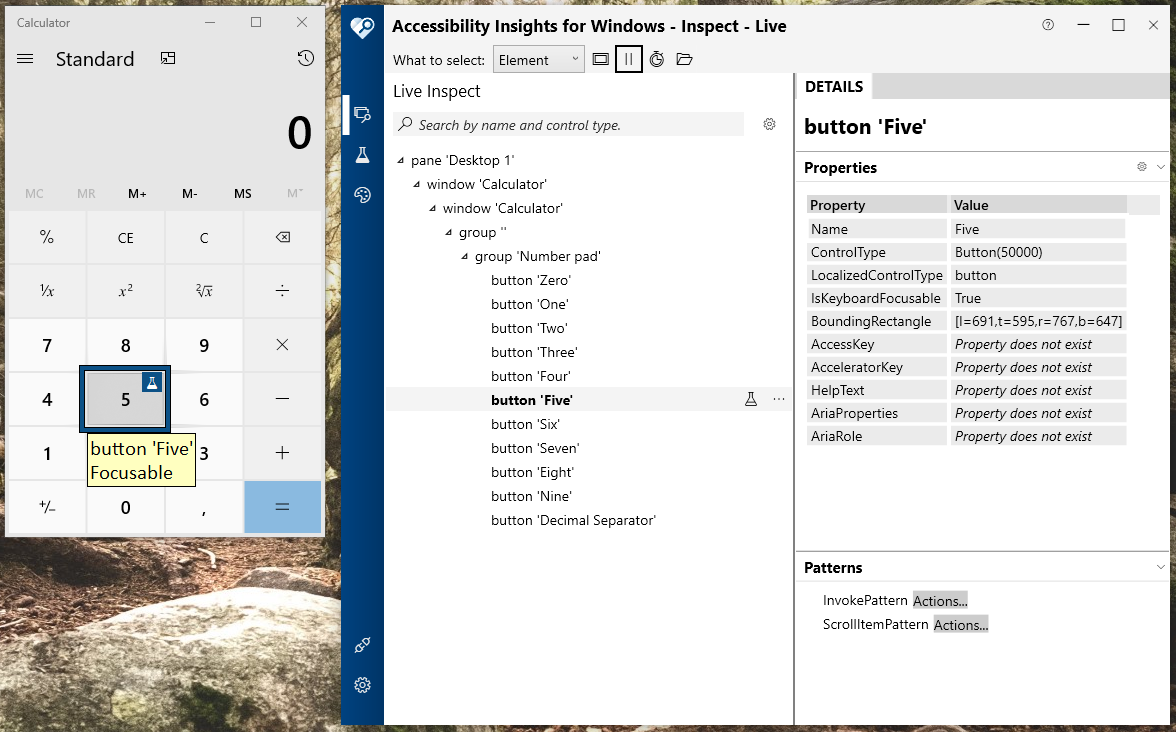
One way to automate [Windows](https://www.microsoft.com/en-us/windows) applications is to target UI components with their identifiers. Microsoft recommends [Accessibility Insights](https://accessibilityinsights.io/) for viewing the UI automation properties. Legacy tools such as [Inspect.exe](https://docs.microsoft.com/en-us/windows/win32/winauto/inspect-objects) can also be used.

## Inspecting Windows applications with Accessibility Insights

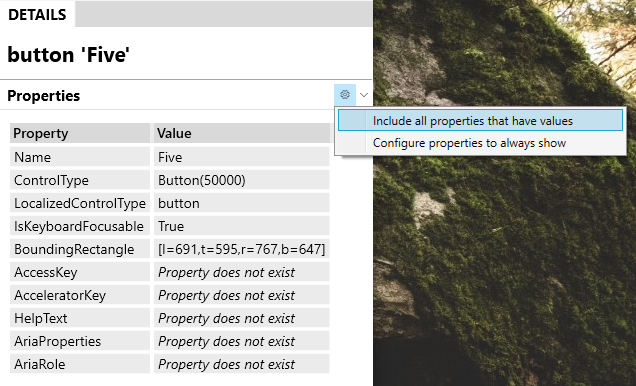
After installing and launching [Accessibility Insights for Windows](https://accessibilityinsights.io/en/downloads), inspecting Windows applications is straight-forward. Using the Windows Calculator as an example, hovering over the application displays the properties of the UI components.



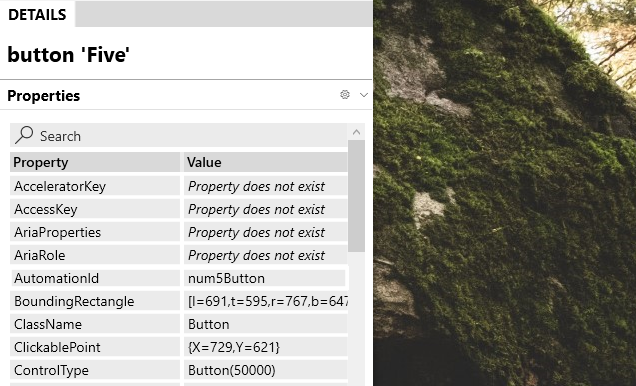
By default, Accessibility Insights displays only a few properties, including the accessible Name of the UI component in the DETAILS pane. In this case, the name of the button is Five. Using localized names for automation is not the most robust option since the labels change based on Windows language settings.



To see more properties, click on the settings icon and select Include all properties that have values:



This will include the AutomationId property. In this case, the value of that property is num5Button:



You can use the value of the AutomationId property in your robot script. Here we are using the [Mouse Click](https://robocorp.com/docs/libraries/rpa-framework/rpa-desktop-windows/keywords#mouse-click) keyword from the [RPA.Desktop.Windows](https://robocorp.com/docs/libraries/rpa-framework/rpa-desktop-windows) library, prefixing the automation ID with id::

\*\*\* Settings \*\*\*

Library RPA.Desktop.Windows

\*\*\* Keywords \*\*\*

Open The Calculator

Open Executable calc.exe Calculator

\*\*\* Keywords \*\*\*

Click The Five Button Using The AutomationId Property Value

Mouse Click id:num5Button

\*\*\* Tasks \*\*\*

Automate The Calculator

Open The Calculator

Click The Five Button Using The AutomationId Property Value

## Accessing UI components and functionality using keyboard shortcuts

If the application supports keyboard shortcuts (see [Windows desktop application robot](https://robocorp.com/docs/development-guide/desktop/windows-desktop-application-robot)), it is recommended to use those whenever possible.

This is an example of a robot that controls a browser using only the RPA.Desktop.Windows library and the [available keyboard shortcuts](https://pywinauto.readthedocs.io/en/latest/code/pywinauto.keyboard.html):

Robocorp also supports [browser automation](https://robocorp.com/docs/development-guide/browser) solutions such as [Selenium](https://www.selenium.dev/) and [Playwright](https://playwright.dev/).

\*\*\* Settings \*\*\*

Library RPA.Desktop.Windows

\*\*\* Keywords \*\*\*

Open the browser

Open From Search Firefox Firefox

\*\*\* Keywords \*\*\*

Open new private window

Send Keys To Input ^+P

\*\*\* Keywords \*\*\*

Focus address bar

Send Keys To Input ^L

\*\*\* Keywords \*\*\*

Type Robocorp docs URL and press enter

Send Keys To Input https://robocorp.com/docs

\*\*\* Keywords \*\*\*

Open new tab

Send Keys To Input ^T

\*\*\* Keywords \*\*\*

Type Robot Framework URL and press enter

Send Keys To Input https://robotframework.org

\*\*\* Keywords \*\*\*

Open search dialog and search for Robocorp docs

Send Keys To Input ^Frobohub

\*\*\* Keywords \*\*\*

Zoom in

Send Keys To Input ^{VK\_ADD}^{VK\_ADD}^{VK\_ADD}

\*\*\* Tasks \*\*\*

Automate desktop web browser

Open the browser

Open new private window

Focus address bar

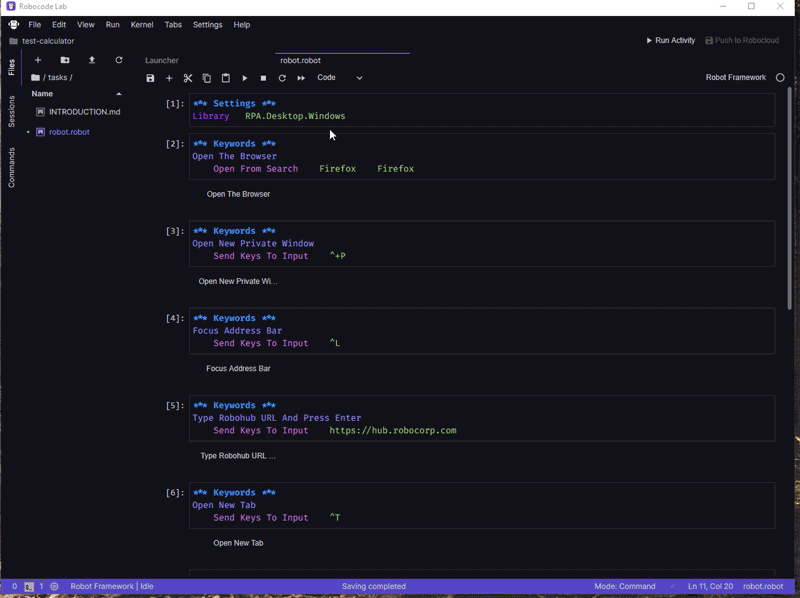
Type Robocorp docs URL and press enter

Open new tab

Type Robot Framework URL and press enter

Open search dialog and search for Robocorp docs

Zoom in

See the robot in action:

## When all else fails: Image-based locators

Sometimes UI components in windows applications do not have an ID that could be used to target them, or they can not be accessed by keyboard shortcuts. In these cases it is still possible to locate them using [image-based locators](https://robocorp.com/docs/developer-tools/robocorp-lab/locating-and-targeting-UI-elements).

## Conclusion

Windows desktop applications can be automated using UI component IDs, keyboard shortcuts, or image-based locators.

Learn more about the libraries mentioned on this page:

* [RPA.Desktop.Windows](https://robocorp.com/docs/libraries/rpa-framework/rpa-desktop-windows)

2020년 2월 19일