Desktop Application Monitoring

Recorder & Scripting

User Manual

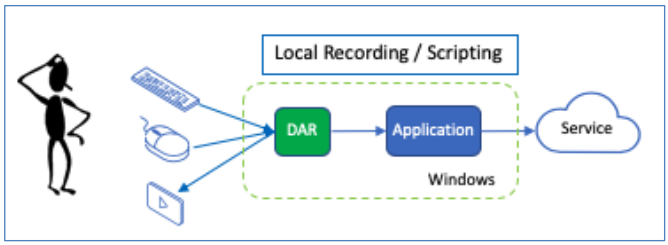
8 May 2019

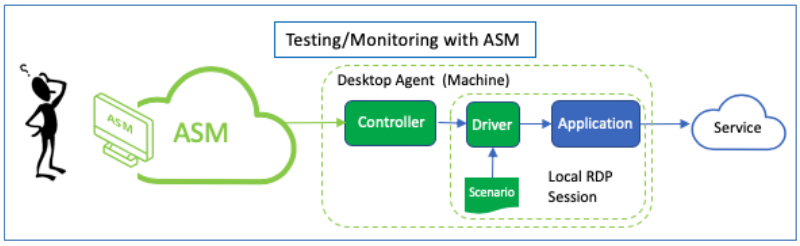
Table of Contents

1 Introduction

Apica's Desktop Application Monitoring is a set of applications and services that measure Microsoft Windows desktop applications. This document is the User's Guide to recording and scripting user scenarios, to be used to monitor desktop Application availability and performance.

The Desktop Application Recorder (DAR) is the Windows Desktop Application part of the solution. There is also the Desktop Application Agent ("Desktop Agent"), used for executing the checks, and typically installed on the customer premises (a so-called "private agent").





Finally, there is the monitoring part of the solution: A Desktop Application Check (DAC) which is the deployed Scenario that is run from the Desktop Application Agent for long term monitoring of the targeted Windows application. This is the check that ASM will report the metrics back as part of the analytics.

2 Basic information

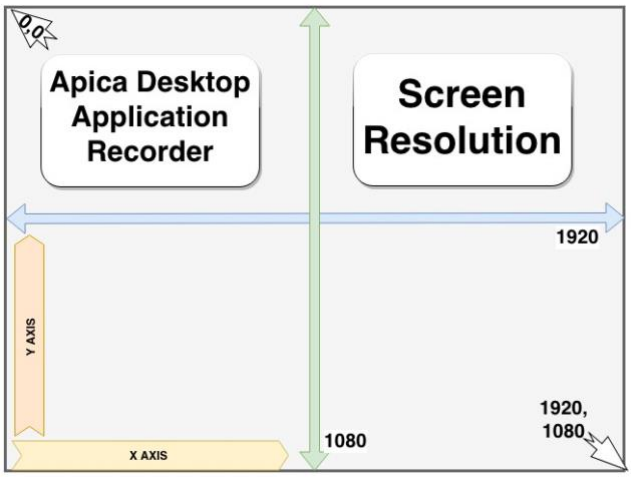
2.1 Screen resolution and screen coordinates

Capturing the desktop application often replies on where the cursor/mouse position is when recording the sequence of events. Many of the commands use the position of the cursor as arguments, e.g. leftClickAt(x,y). So, it is essential to use the same screen resolution for both recording and executing the scenarios.

Default Resolution: The default screen resolution for the Desktop Application Agent (which executes the scenarios) is 1920 x 1080, so Apica recommends this when recording scenarios.

The grid is laid out as follows:

- The upper left corner has
- coordinates x=0, y=0.
- The lower right corner has coordinates x=1920, y=1080.
- Where
    - X is horizontal (left/right)
    - Y is vertical (up/down).
- When selecting an area, e.g. with the command assetTextAt(x,y,width,height) the x,y point is in the upper left corner of the area/box.



2.2 Location of files

Scenarios are the recorded series of DAR-captured steps. These scenarios are stored in a directory, specified in a properties file.

For the developer installation, the default directory location for scenario files is “C:/apica/asm-desktop-agent/embedded/asm-desktop-agent/scenarios/”.

This location is set in the “C:/apica/asm-desktop-agent/embedded/asm-desktop-agent/application.properties” file.

You can change the location of the scenarios by editing this properties file.

Note: It’s only possible for the Desktop Application Agent to test-run scenarios that are located in the set default location.

2.3 Best practices

When starting an application (to be tested):

- **Avoid using click-on-icon**, neither on the desktop nor on the taskbar.
- Avoid the start-menu.

Why: the agent uses a remote desktop session to run the test, and that user/desktop may not have the necessary icon in place (or the same place). Use the startApplication command instead.

Sync the Window Size: After starting the application, Apica recommends either maximizing the window, or positionWindow(x,y) in order to have the coordinates be the same as what was captured in the recording to what is played back during the test session.

Note: Apica recommends avoiding Exact type matches when matching text because its match is very strict and does not accept (as an example) any extra spaces before/after. Apica recommends using "contains" as a preferred match type.

Remember to assert that the application interface has been completely rendered before you continue with the next command. For instance, trying to assert that a certain text string is present on the screen too quickly may fail, purely because it has not been rendered yet. It is recommended that you e.g. use the waitForPixel command first and assert after this.

When using the startApplication command to start the Windows command prompt, use the following arguments (see command reference)

- "application": "C:\\Windows\\System32\\cmd.exe",
- "applicationArguments": "/c start cmd"

When using the startApplication command to run a bat script, use the following arguments (see command reference)

- "application": "C:\\Windows\\System32\\cmd.exe",
- "applicationArguments": "/c start some_script.bat"

When using the command "assertImage", try selecting an image with many colors and variations as monotoned images are more difficult to match. Apica recommends a size of around 100x100 pixels.

3 Installing the application

The Desktop Application Monitoring installation guide is provided separately from this manual.

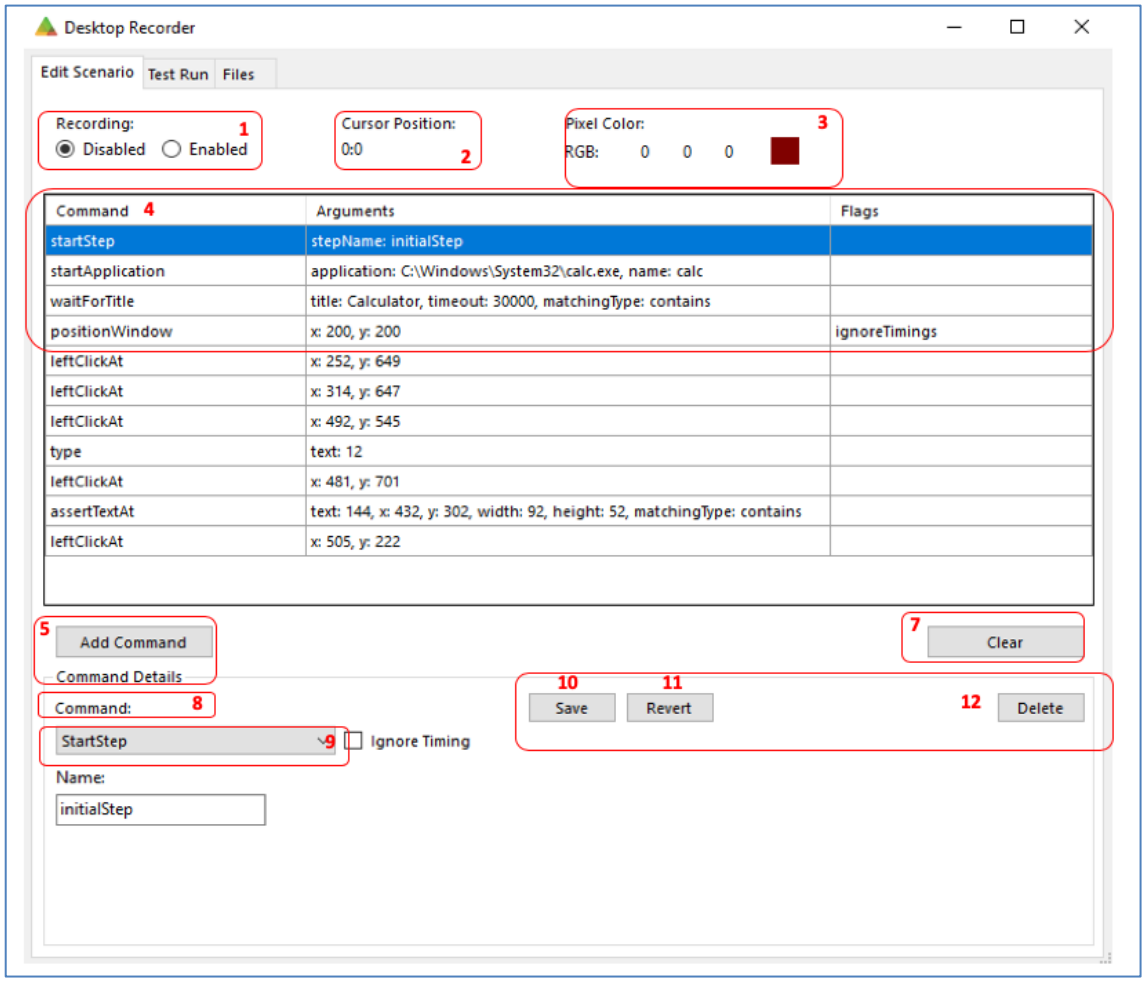
4 The Desktop Application Recorder User Interface

The main interface of the application has three tabs:

- Edit Scenario
- Test Run
- Files

4.1 Edit Scenario Tab

Recording and editing scenarios.

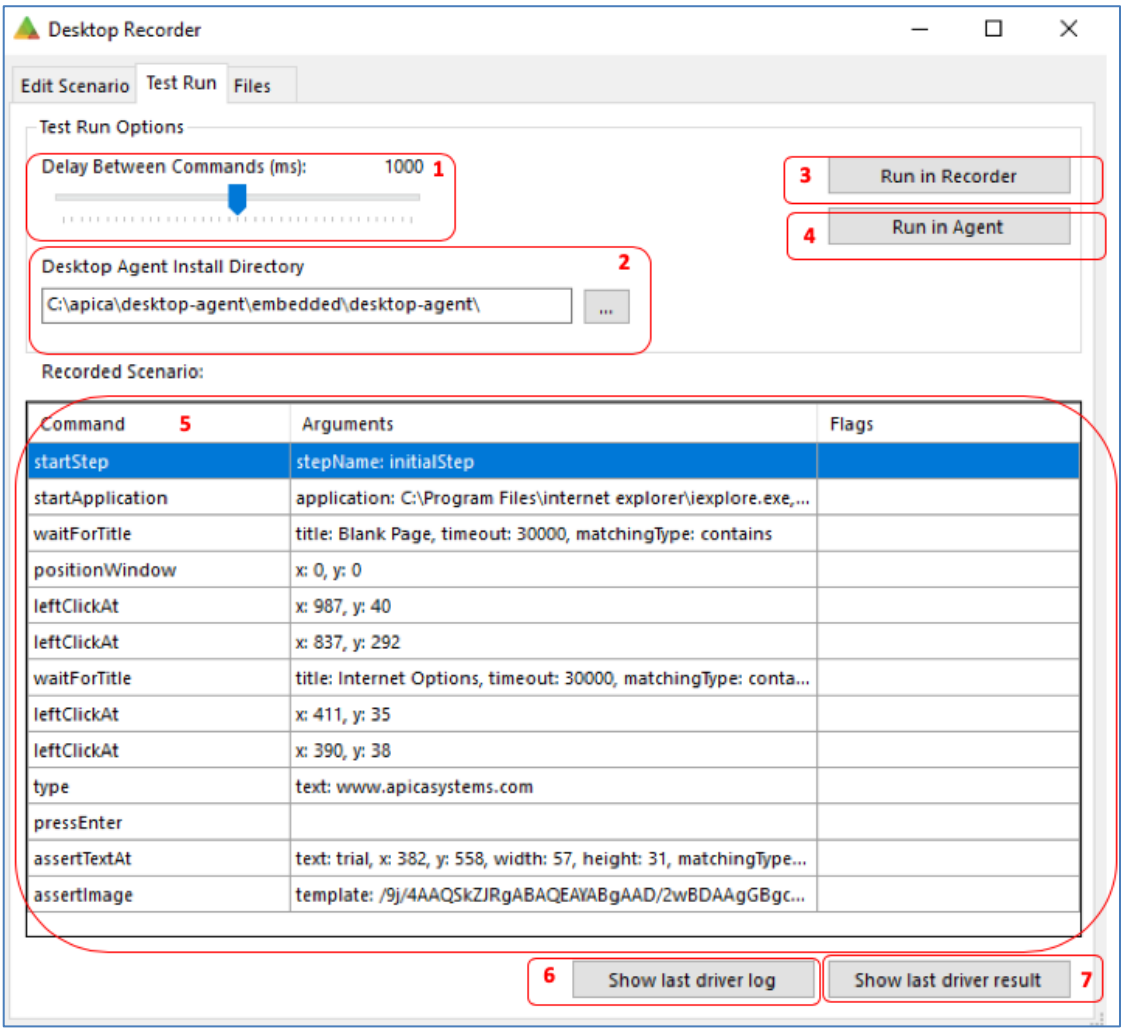


Edit Scenario Guide:

#	Item	Explanation
1	Recording	Disable or Enable recording
2	Cursor Position	While automatic recording is enabled, the position of the mouse cursor is displayed here
3	Pixel Color	While automatic recording is enabled, the RGB value of the pixel at mouse cursor position is displayed here.
4	Scenario Table	The name of the recorded command Arguments: argument values Displays enabled flags (if any)
5	Add Command Button	Adds a new command to the scenario
6	Import File Button	Import a scenario from file
7	Clear Button	Clear/Delete all commands
8	Command Details	Shows details of the currently selected command, or a new command
9	Command Drop-Down	Shows the currently selected command, when adding a new command, you may select type of command here
10	Save Button	Click here to save any changes made to the currently selected command, or to save a new command
11	Revert Button	If you made changes to the currently selected command, click here to revert to the original arguments
12	Delete	Removes the selected command from the scenario

4.2 Test Run Tab

Testing/verifying a new recording or imported scenario.



Test Run Guide:

	Item	Explanation
1	Delay between command slider	Use the slider to set a delay (in ms) between each command in the scenario, the default delay is 1000 ms.
2	Installation Directory	The scenario directory is located here.
3	Run in Recorder	Start a test run of your scenario through the recorder. Note that only a subset of the commands can be played back (executed) by the desktop recorder, to enable full command support you must use ”Run in Agent”
4	Run in Agent	Start a test run of your scenario through the real Desktop Agent. See Test Run for more information.
5	Recorded Scenario Table	- The name of the recorded command - Arguments: Recorded argument values - Displays enabled flags (if any)
6,7	Show last driver log Show last driver result	- Show log and result

4.3 Files Tab

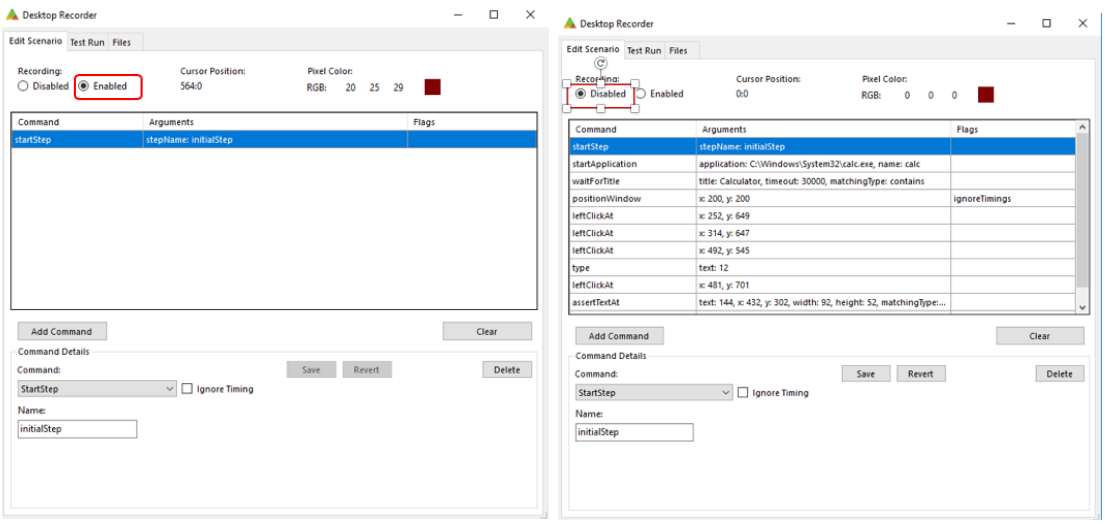
Opens (loads), or Saves, the scenario in JSON format. Files Guide:

	Item	Explanation
1	Name	Name of scenario
2	Description	Field for optional scenario description
3	Include Global Pause Timing	Adds a pause command in between each command in the scenario, with a delay that corresponds to the delay configured in the "Test Run" tab.
4	Save as… / Open	Opens a traditional windows file dialog.

5 Recording a scenario

The Desktop Recorder can record the following actions automatically: leftClickAt, rightClickAt, doubleClickAt, mouseMove & dragTo (click and drag), type, pressEnter, pressEscape, pressBackspace, pressTab.

To start the recording select the “Enabled” radio button, under "Automatic Recording".



When you are done select the checkbox labeled "Disabled" to stop the recording.

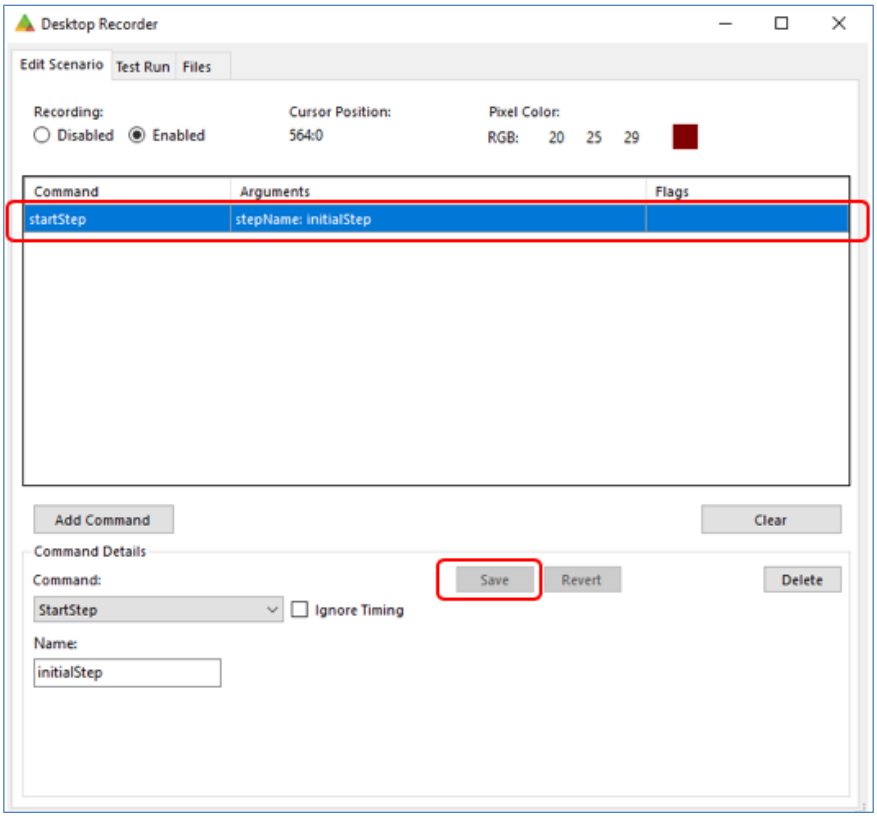
Note: Mouse-clicks on/in the Desktop Recorder are not recorded.

The following commands are recordable:

- leftClickAt
- rightClickAt
- doubleClickAt
- mouseMove
- dragTo
- type
- pressEscape
- presenter
- pressBackspace
- pressTab

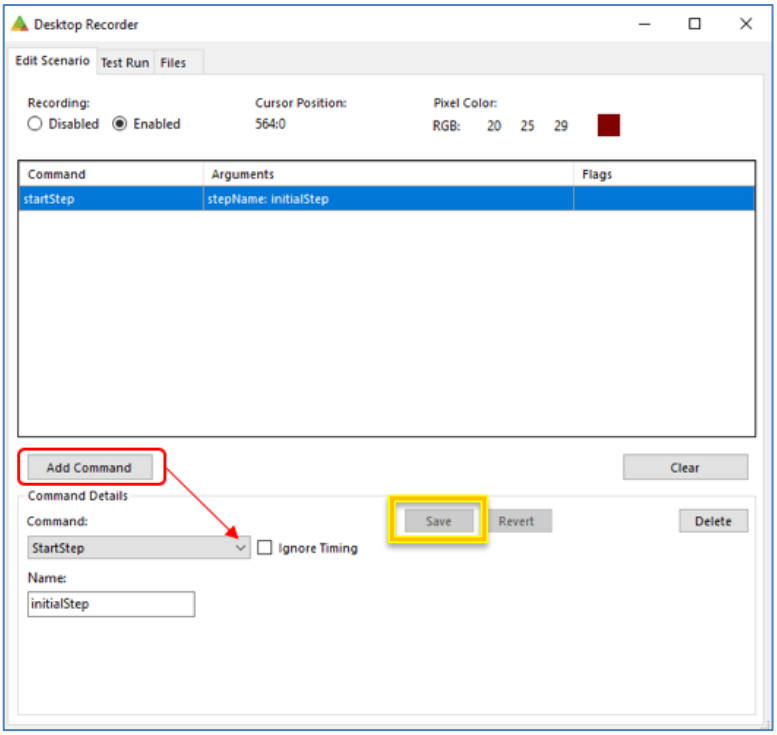
6 Add/Edit/Delete Command

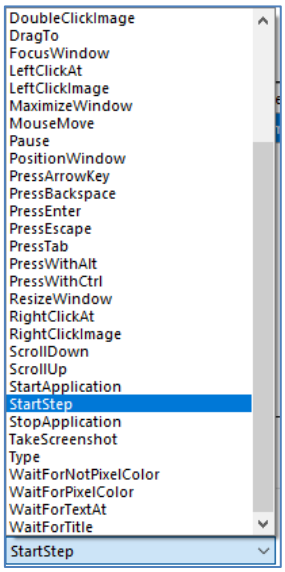
When you open the Desktop Recorder you will be presented with a new scenario, which contains a single "startStep" command. You may change the name of the step by selecting the command and then entering a new name in the text box labeled "Name", save the command by clicking the "Save" button.



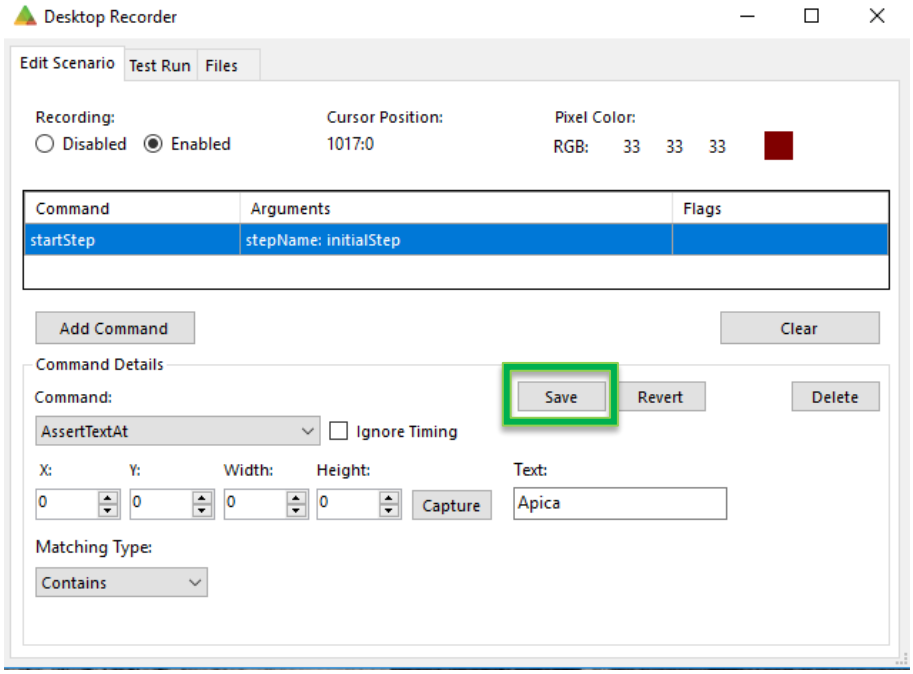
6.1 Add

To add a new command, start by clicking "Add Command" and then select a command in the drop-down list labeled "Command".





You will be presented with several input fields (arguments) which must be filled out before you can use the save for that command.



To see the how the different commands work, see the command reference list.

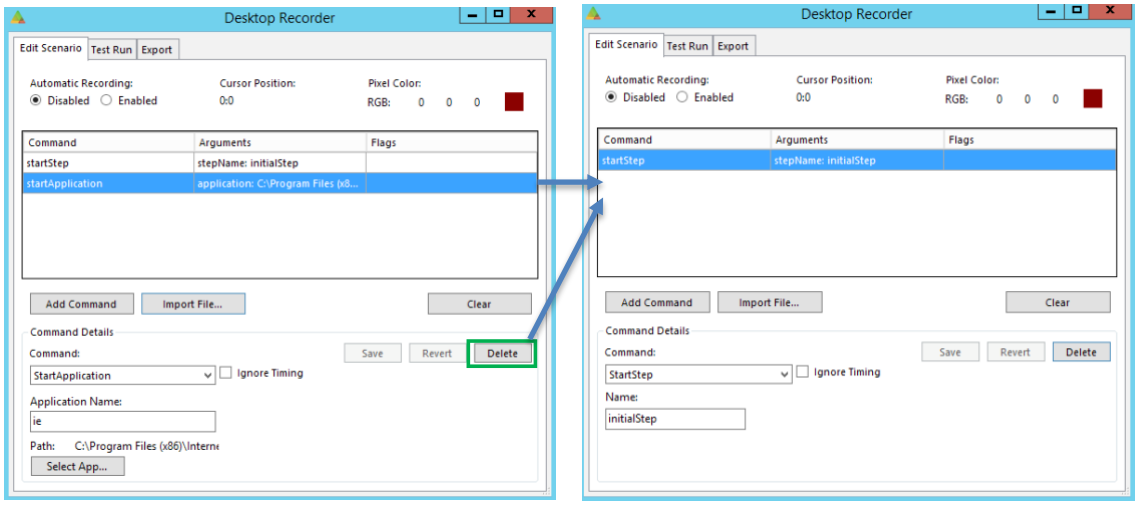
Once you have entered values for all required fields, click "Save" to add the command to your scenario.

6.2 Edit

To edit a command, select the command in the list, edit one or more arguments and then click "Save" to save the changes or "Revert" to revert the changes.

6.3 Delete

To delete a command, select the command in the list and click "Delete" to remove it from the scenario.



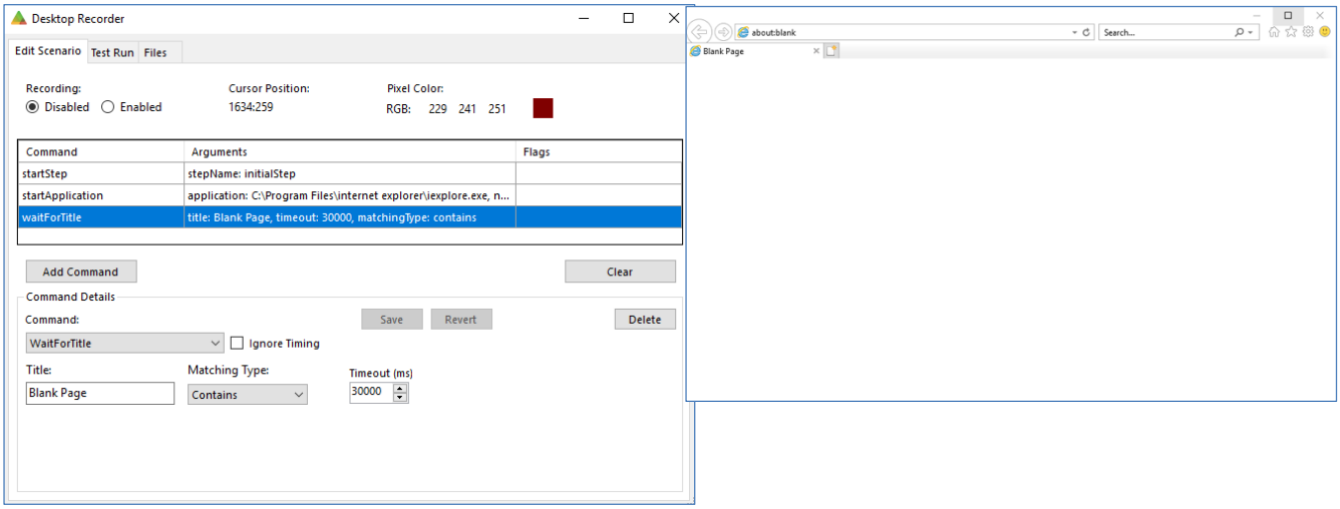
7 Validation and Flow Control

7.1 Wait for Applications to Start / Windows to Appear

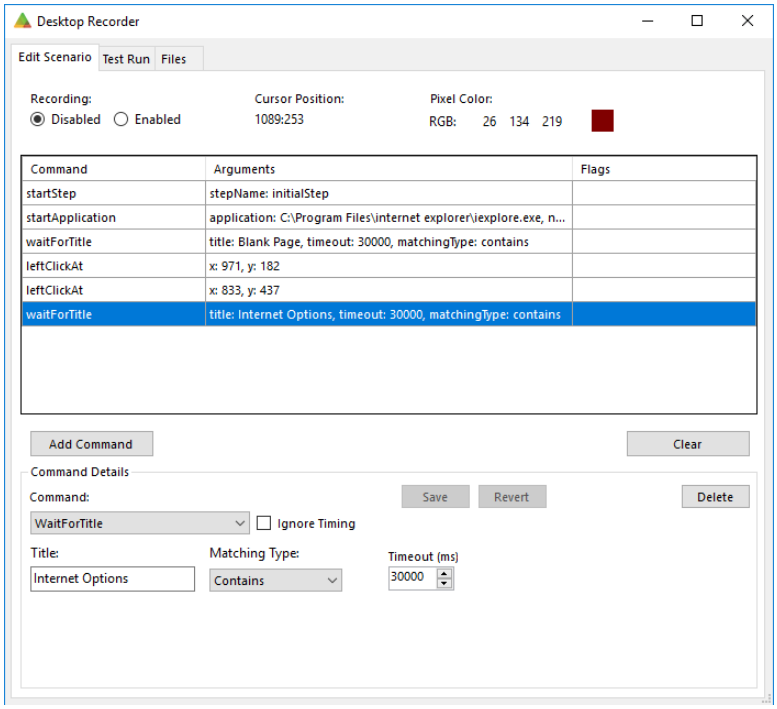
A scenario typically starts with a startApplication command which starts an instance of the application that the user wishes to test. Whenever you start an application that spawns a new window—or whenever you switch between windows (using the focusWindow command)—you should be able to use the waitForTitle command to wait for a new window (with a title that matches the "Title" argument) to appear. If an application opens pop-up windows within the application, these windows are also typically labeled, and you should be able to use the waitForTitle command to pause the scenario execution until these windows appear.

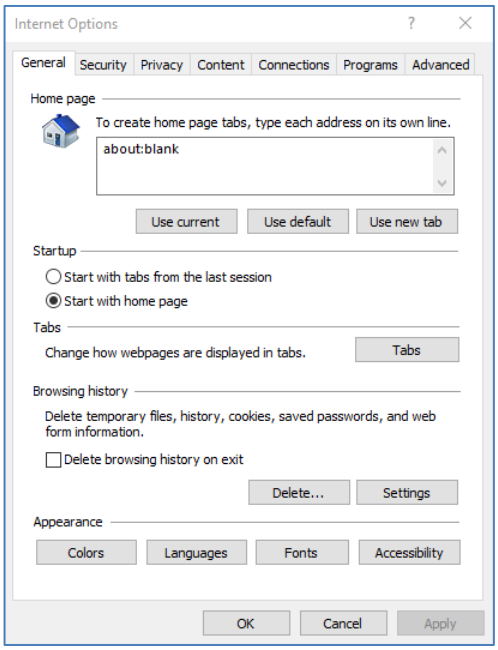
You can also use the *Matching Type* argument to select matching type: exact, glob (global expressions), or “contains”.

In the following example, we start by opening Internet Explorer, and then we wait for the title of the currently focused window to match "Blank Page", using the “contains” option, with a timeout set to 30 seconds.



Next, we open "Internet Options" and wait for the title of the currently focused window to match "Internet Options"

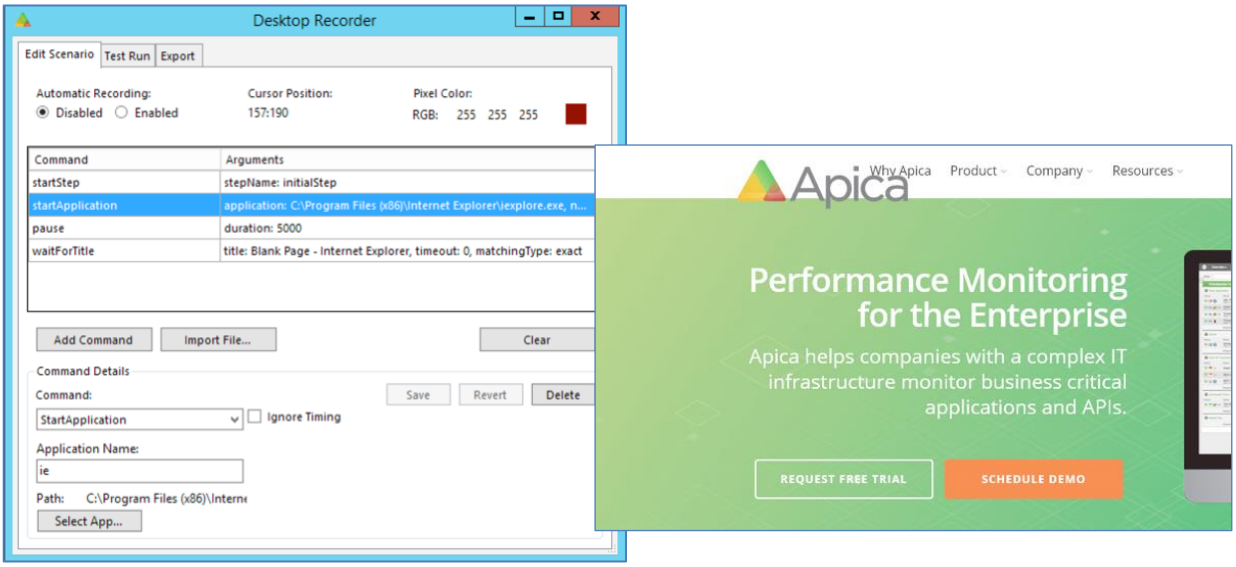




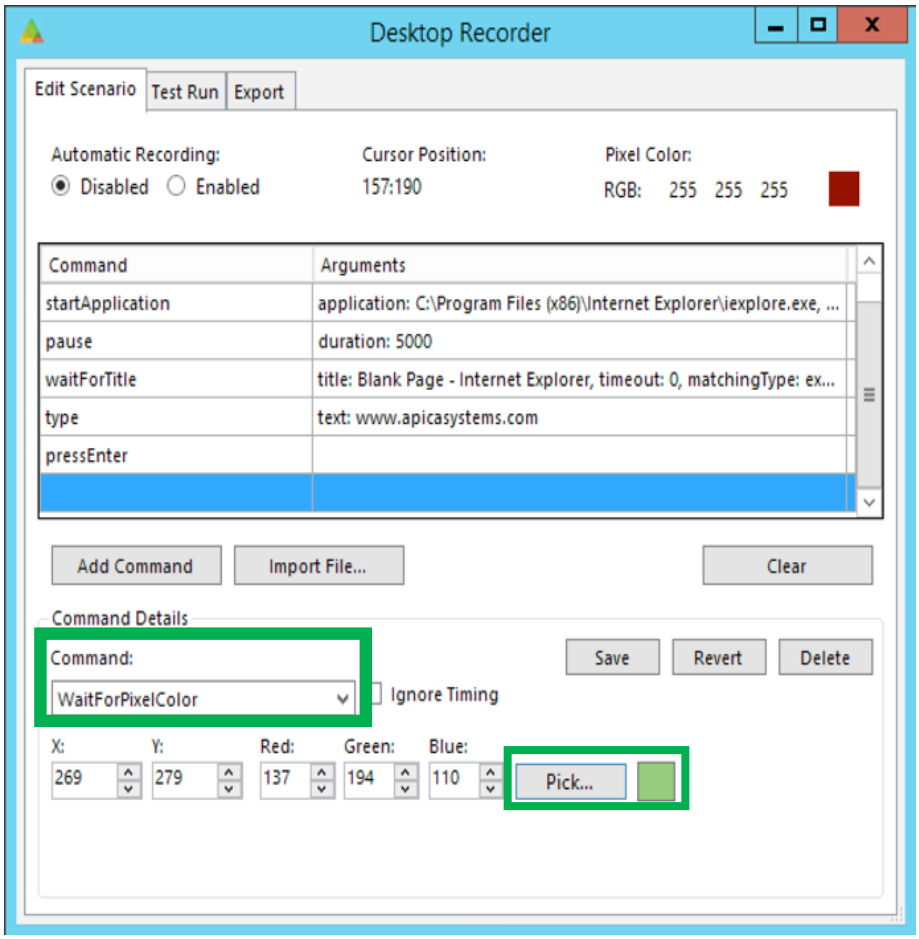
7.2 Wait for Non-Text Content to Appear on Screen

If you need to wait for some generic content to appear on the screen, you may be able to use waitForPixelColor command to wait for the RGB value of a specific pixel on the screen, to match a pre-defined value captured by the Desktop Recorder.

In this example, we start by opening Internet Explorer and then navigate to www.apicasystems.com



Next, we add a "WaitForPixelColor" command, click "Pick" and then click on the green background of the apicasystems.com site. When the scenario runs (after pressing <enter> to load apicasystems.com), the scenario execution will pause



until the background color changes.

7.3 Assert that Text is Present on the Screen

You can use assertText or assertTextAt to assert that a string of text is present on the screen. Note that the recorder is not able to re-play (execute) these commands. To test them, you need to check the "Use full command support" checkbox.

assertText

The *assertText* command captures a screenshot of the currently focused application *window*, feeds the image to the OCR engine (Tesseract) which returns any text detected. The *Text* argument is then compared to the text returned by Tesseract. If it matches, the command is successful. You can also use the *Matching Type* argument to select matching type: exact, glob (global expressions), or “contains”.

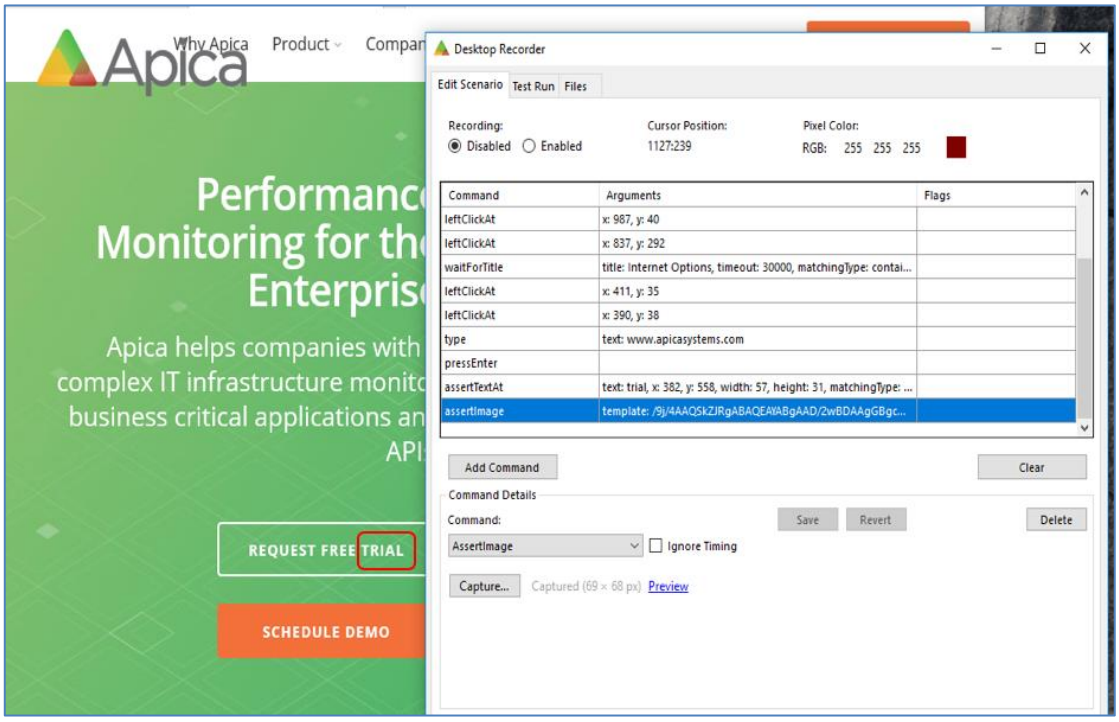
assertTextAt

The *assertTextAt* command captures a screenshot of a specific *part of the screen*, feeds the image to the Tesseract OCR engine which returns any text detected. The *Text* argument is then compared to the text returned by Tesseract. If it matches, the command is successful. You can also use the Matching Type argument to select matching type: exact, glob (global expressions), or “contains”.

Example of assertTextAt: To select the part of the screen that will be passed to Tesseract start by clicking the "Capture" button. A transparent overlay is added on top of the screen.

Click and hold the left mouse button and drag the mouse to select a rectangle on the screen, making sure that the text you wish to detect is completely within this rectangle, once you release the mouse, the arguments in command details should be updated automatically.

In the following example, we start by opening Internet Explorer and then navigate to www.apicasystems.com. Next, we add an "*AssertTextAt*" command and click "Capture", this adds an overlay on top of the desktop, much like the snipping tool in Windows. Then, we select a part of the screen that covers the text we wish to detect (in this case we selected the word ‘TRIAL’ in the “REQUEST FREE TRIAL” button), when the mouse button is released the WIDTH, HEIGHT, X and Y arguments should be updated automatically.



Next, in the input field labeled "Text", we enter a string of text to match against and the type of match (exact/contains/glob). When the scenario runs with this condition, it will stop executing if we cannot detect that text within the specified coordinates.

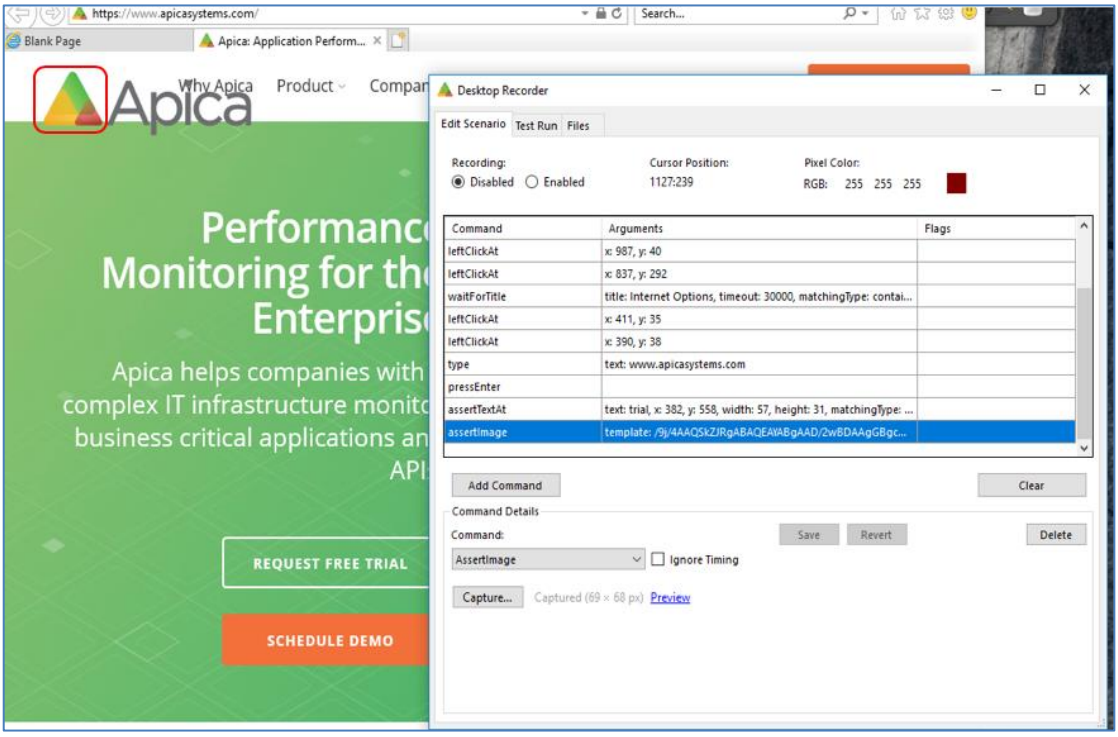
Note that these commands work best when you try to match a single word or string of text, and *that matches are case insensitive*

7.4 Assert that Image is Present on Screen

Use the “AssertImage”: command to stress that a particular image must be present on the screen, as a condition. First, you must capture the image to assert. So, start by clicking the "Capture" button. A transparent overlay is added on top of the screen, click and hold the left mouse button, and drag the mouse to select a rectangle on the screen, once you release the left mouse button the arguments should be updated automatically. Apica recommends a size of around 100x100 pixels.

Important: The captured image must be distinctive and not found on any other part of the screen that is being evaluated.

In this example we start by opening Internet Explorer and load www.apicasystems.com. Next, we add an "AssertImage" command and click "Capture", this adds an overlay on top of the desktop, much like the snipping tool in Windows. Then, we select the part of the screen that we wish to use for verification. In this case, we select the Apica logo. So, when the scenario runs, if we cannot detect the Apica logo on the screen, it will stop executing.



7.5 Pause Scenario Execution

In some cases, the script/scenario can run faster than the application can respond, causing false execution errors. In these cases, if there is a reason to wait for the application to render a response, use the “Pause” command to pause the scenario execution for X milliseconds.

8 Matching text

For any command where we match text (assertTitle, waitFortitle, assertText, assertTextAt, waitForTextAt), the Desktop Application Recorder includes the option to select the type of matching performed (exact/contains/glob). Again: Text matches for these types are *case in-sensitive*.

Exact: The returned text must match the recorded value exactly.

Apica recommends avoiding Exact type matches when matching text because its match is very strict and does not accept (as an example) any extra spaces before/after. Apica recommends using “contains” as a preferred match type.

Contains: The returned text must contain the recorded value.

Glob: (Global expressions) The returned text must match the recorded value based on a specified pattern of single character using a question mark '?' or multiple/no characters using an asterisk '*'.

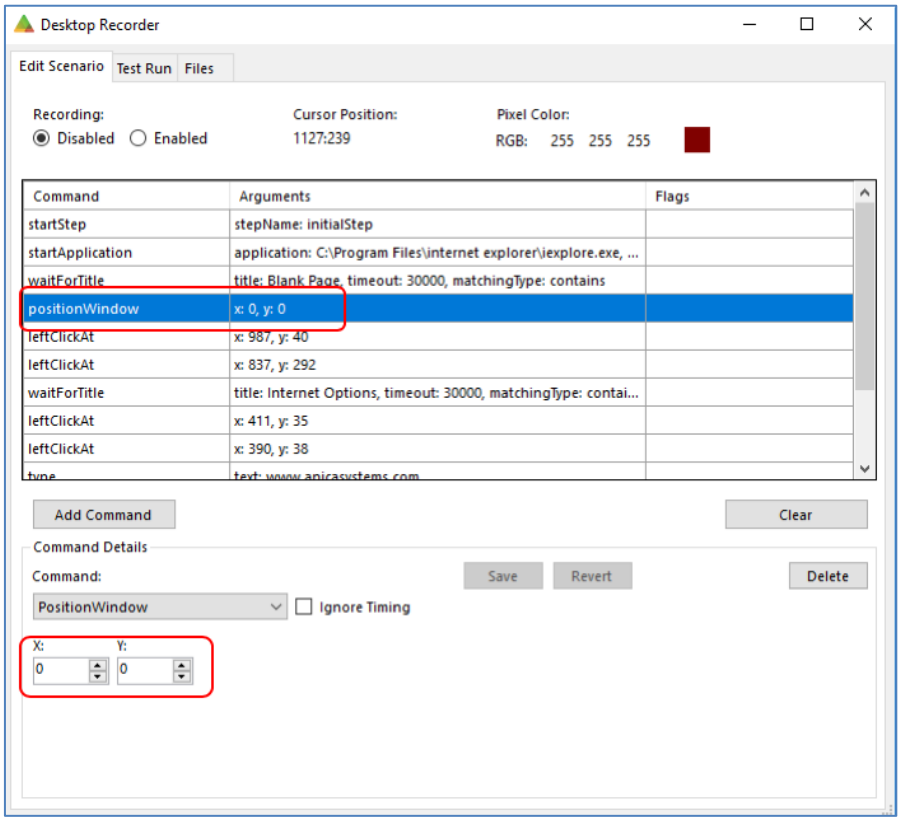
The matching type - glob works as follows:

Wildcard	Description	Example	Matches	Does not match
*	matches any number of any characters including none	Law*	Law, Laws, or Lawyer	GrokLaw, La, or aw
		Law	Law, GrokLaw, or Lawyer.	La, or aw
		*obot	Robot, obot, mobot	I am a robot
		* robot	I am a robot	Obot, I am a obot
?	matches any single character	?at	Cat, cat, Bat, bat	at

9 Window Management

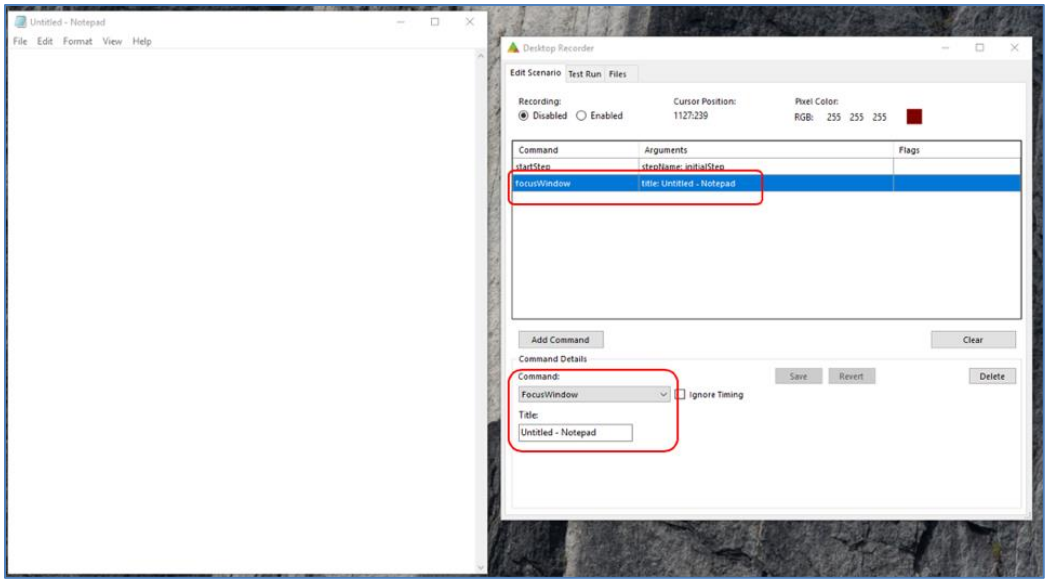
9.1 Position Window

Use PositionWindow to position the currently focused application window at coordinates X,Y. To place the window in the upper left window, enter coordinates 0,0.



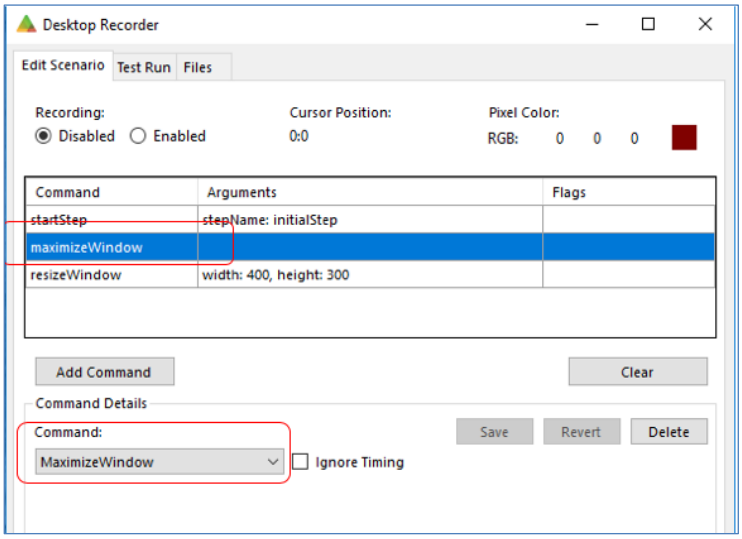
9.2 Focus Window

Use FocusWindow to switch focus to another window defined by Window title. Example: switch focus to the Notepad window:



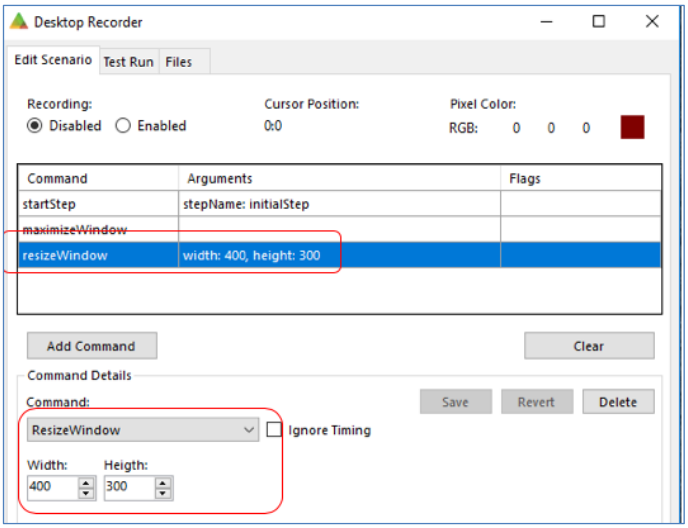
9.3 Maximize Window

Use MaximizeWindow to maximize the currently-focused application window.



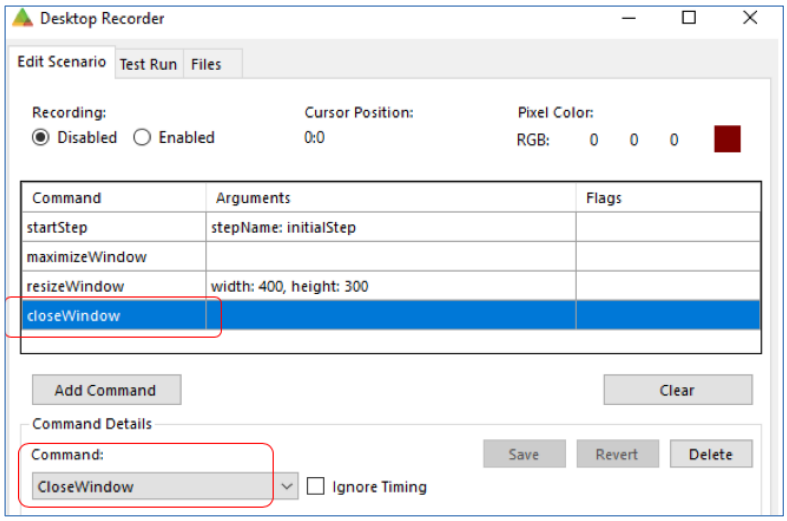
9.4 Resize Window

Use ResizeWindow to resize the currently focused application window to a selected WIDTH, and HEIGHT.



9.5 Close Window

Use CloseWindow to close the currently focused application window.



10 Ignore Timing

The Ignore Timing checkbox next to Commands is a flag that can optionally be toggled On/Checked or Off/Unchecked. When this box is checked, the flag is enabled and the duration of the command (the time it took to execute the command) won't be included in the total duration of the test.

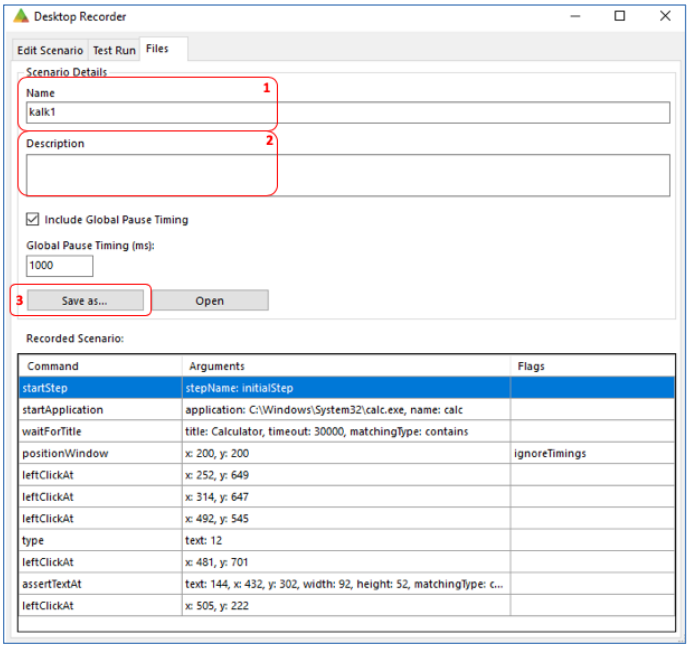
11 Open/Save Scenarios

11.1 Save a Scenario

To save your scenario to file, start by selecting the "Files" tab.

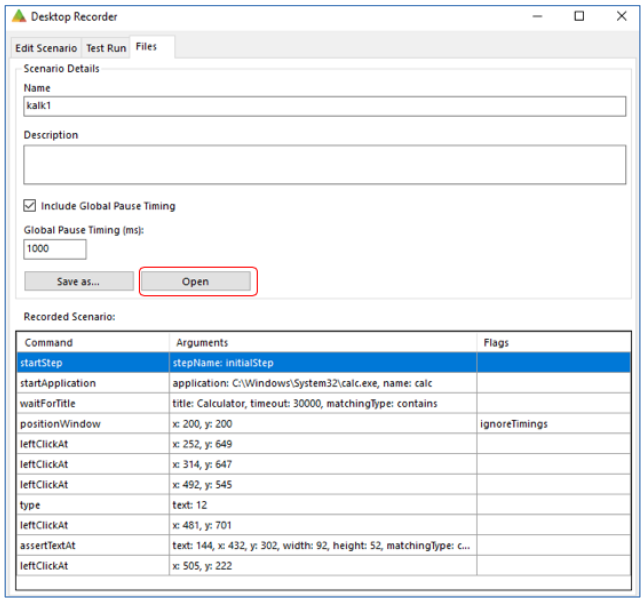
Enter a name for your scenario in the input field labeled "Name" (1). You may also add a Description (2). Click "Save as..." (3) to continue.

Browse to a location and click "Save" to save the scenario to file.



11.2 Open (loading) a scenario

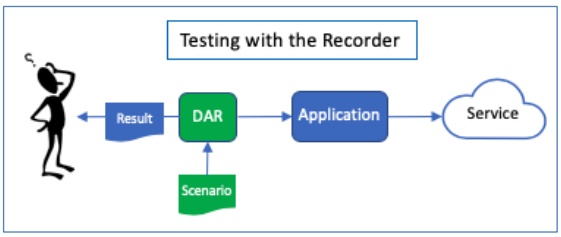
The scenarios are stored in JSON format. You can load a previously-created scenario from a file by clicking the "Open" button.

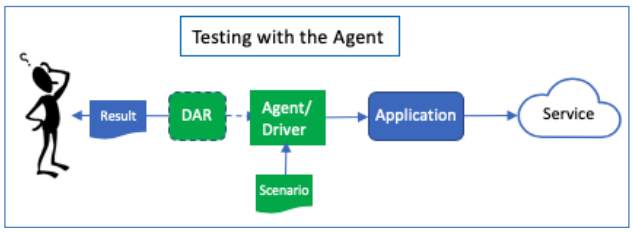


Browse to and select your scenario file and click "Open" to load it into the Desktop Application Recorder.

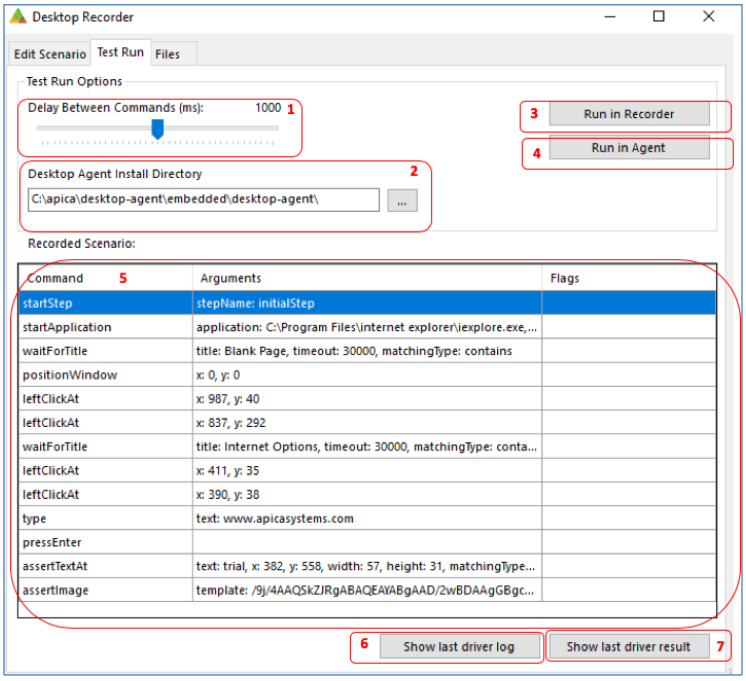
12 Test Run a Scenario

You can test either with the local Desktop Recorder or you can run it on the Agent.

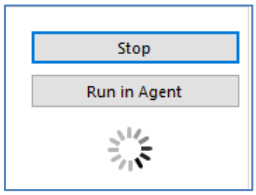


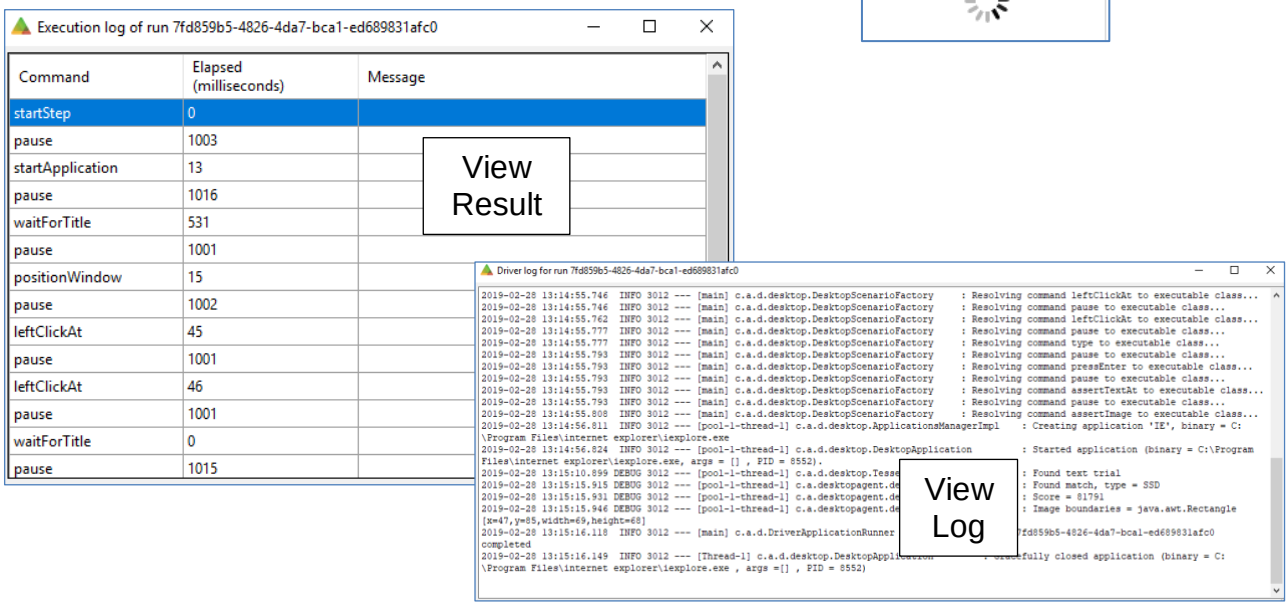


1. To test your scenario, Select the "Test Run" tab.
2. Click "Run in Recorder" (3) or Run in Agent" (4) to start the test.
3. When running in recorder, use the slider labeled "Delay Between Commands (ms)" (1) to set a delay, in milliseconds, between each command.
4. When "Run in Agent" is used, the delays saved in the scenario is used.



5. While a test is running, a “Stop” button appears, replacing “Run in…”. Click this to stop the run.
6. After a test “Run in Agent”, you can view the Result (7) and the Log (6).





Note that only the following commands will be executed (re-played) when using the “Run in Recorder” option:

“Run in Recorder” Only Options

startApplication	stopApplication	pause
positionWindow	focusWindow	maximizeWindow
resizeWindow	closeWindow	leftClickAt
rightClickAt	doubleClickAt	mouseMove
dragTo	type	pressEscape
pressEnter	pressBackspace	pressTab
assertTitle	waitForTitle	assertPixelColor
assertNotPixelColor	waitForPixelColor	waitForNotPixelColor

The “Run in Recorder” option is typically used in the beginning of the scripting, to verify mouse and keyboard commands.

13 Command Reference

Group	Command	Description	Arguments	Notes	Recorder Support: recording	Recorder Support: replaying
Applications	startApplication	Start application	application, name, applicationArguments	application is path to binary. name creates a reference to the started application. applicationArguments - comma-separated list of arguments as single string (e.g. "--argumentOne,--argumentTwo,--argumentThree=value")	Manually	+
	stopApplication	Quit application	name	name is reference to started application. This command will then close it.	Manually	+
Windows	focusWindow	Switch focus to target window X	title		Manually	
	closeWindow	Close the current window.			Manually	+
	positionWindow	Position current window at location X,Y	x, y		Manually	+
	resizeWindow	Resize current window to WIDTH:HEIGHT	width, height		Manually	+
	maximizeWindow	Maximize the current window.			Manually	+

	assertTitle	Assert that the current window title matches X	title, matchingType (optional, possible values: 'exact' (default) | 'glob' | 'contains')		Manually	+
	waitForTitle	Wait for current window title to match X	title, timeout (optional, in milliseconds), matchingType (optional, possible values: 'exact' (default) | 'glob' | 'contains')		Manually	+
Mouse	moveMouse	Move mouse pointer to coordinates X,Y	x, y		+	+
	dragTo	Hold left mouse and move pointer to X,Y	x, y		+	+
	leftClickAt	Left click at coordinates X,Y	x, y		+	+
	rightClickAt	Right click at coordinates X,Y	x, y		+	+
	doubleClickAt	Double click at coordinates X,Y	x, y		+	+
	scrollUp	Scroll up X notches.	notches		Manually	
	scrollDown	Scroll down X notches.	notches		Manually	
Keyboard	type	Simulate keystrokes for each character in STRING	text		+	+
	pressWithAlt	Press key with alt. Key must be alphanumeric.	key		Manually	
	pressWithCtrl	Press key with ctrl. Key must be alphanumeric.	key		Manually	
	pressEnter	Presses enter.			+	+
	pressBackspace	Presses backspace.			+	+
	pressEscape	Presses escape.			+	+
	pressTab	Presses tab.			+	+

	pressFunctionKey	Presses a function key	key	Key is a number between 1 and 12 and corresponds to F1-F12.		+
	pressArrowKey	Presses an arrow key	direction	direction is one of: up | down | left | right	Manually	
Flow Control	pause	Pause for X ms	duration		Manually	+
Text	assertText	Assert that TEXT is present in active window	text, matchingType (optional, possible values: 'exact' (default) | 'glob' | 'contains')		Manually	
	assertTextAt	Assert TEXT is present within X:Y:WIDTH:HEIGHT	text, x, y, width, height, matchingType (optional, possible values: 'exact' (default) | 'glob' | 'contains')		Manually	
	waitForTextAt	Wait until TEXT is present within X:Y:WIDTH:HEIGHT, or until timeout expires	text,x,y,width,height,timeout (optional, in milliseconds), matchingType (optional, possible values: 'exact' (default) | 'glob' | 'contains')			
Image Recognition	assertImage	Assert that the IMAGE (base 64 image data) is present in active window	template, scoreType (optional, possible values: SSD (default) | NCC)		Manually	
	leftClickImage	Locate IMAGE (base 64 image data) on the screen and perform a left click in the center	template, scoreType (optional, possible values: SSD (default) | NCC)		Manually	

	rightClickImage	Locate IMAGE (base 64 image data) on the screen and perform a right click in the center	template, score Type (optional, possible values: SSD (default) | NCC)		Manually	
	doubleClickImage	Locate IMAGE (base 64 image data) on the screen and perform a double click in the center	template, score Type (optional, possible values: SSD (default) | NCC)		Manually	
Screenshots	takeScreenshot	Takes a screenshot.			Manually	
Misc.	startStep	Create a new group of commands	stepName		Manually	
Validation	assertPixelColor	Assert that pixel color (RGB value) at coordinates X,Y matches RED,GREEN,BLUE	x, y, red, green, blue		Manually	+
	assertNotPixelColor	Assert that pixel color (RGB value) at coordinates X,Y does not match RED,GREEN,BLUE	x, y, red, green, blue		Manually	+
	waitForPixelColor	Wait until the pixel color (RGB value) at coordinates X,Y matches RED,GREEN,BLUE	x, y, red, green, blue, timeout		Manually	+
	waitForNotPixelColor	Wait until the pixel color (RGB value) at coordinates X,Y does not match RED,GREEN,BLUE	x, y, red, green, blue, timeout		Manually	+

14 Scenario Format

The format of an exported/saved scenario is JSON, and looks as follows:

```
{
    "name": "Example Scenario Name",
    "description": "A simple description",
    "global_pause": 1000,
    "include_global_pause_timings": false,
    "variables": {},
    "commands": [
      {
         "command":"startApplication",
         "args":{
            "name":"notepad",
            "application":"C:\\Windows\\system32\\notepad.
exe"
         },
         "ignore_timing":true
      },
        {
            "command": "type",
            "args": {
                "text": "typing!"
            },
            "ignore_timing": false
        }
    ]
}
```

A scenario file can be edited manually. Before uploading, Apica recommends testing it in the Desktop Application Recorder, or at least verify the JSON formatting.

15 Appendix - Testing a scenario through the command line

(go to the installation directory of the Desktop Agent)

```
> java -Djava.awt.headless=false -jar driver-0.jar --
scenarioId=<name_of_scenario> --runId=abc123
```

This will run the scenario using the Agent, and show the console window during execution, which might be useful for debugging.

16 Appendix - Testing a scenario through the API

NOTE: 19-05-07 NOT YET SUPPORTED

When the Desktop Agent has been installed as a service (see installation manual), you can command it to execute checks (i.e. dispatching a job) through the API (example: using Postman).

Important: This will initiate the run through the service (which will initiate a new local RDP session etc.). After finishing the job, all RDP sessions will be terminated. You MUST follow the complete installation guide before this will work.

16.1 Start a job

```
(post) http://<host>:8080/job
Headers
Accept:application/json
Content-Type:application/json
Body
{"debug": true, "format_version": 0, "job_timeout": 60, "scenario_id":
"<name-of-scenario>.json",  "screenshots": true}
Expected response
{
    "id": "8b2e9be586714bd5896ff10de52ea99e"    // The jobId
}
```

Note: If the command results in a 500-error, it may be because the scenario does not exist.

16.2 Get JobStatus

```
(get) http://<host>:8080/job/<jobId>
Expected response
{
    "id": "<jobId>",
    "status": "finished"  // or running, failed etc.
}
```

16.3 Get a result

```
(get) http://<host>:8080/result/<jobId>
Header
Accept:application/xml
```

Expected response

{A result in XML format}