Apica

Desktop Application Check

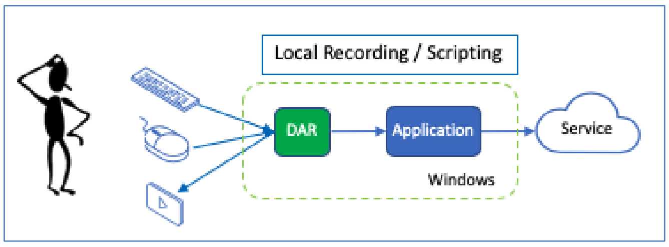
Recorder & Scripting

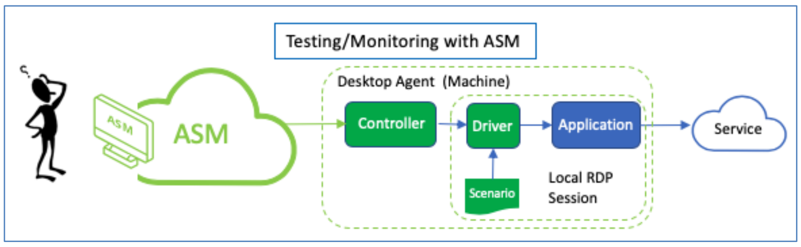
User Manual

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.

For shorthand and convenience, '**DAC**' is also the overall solution name what Apica uses for the manuals and titles and the umbrella term for the DAR, Agent, and Desktop Application Check modules.

2 Basic information

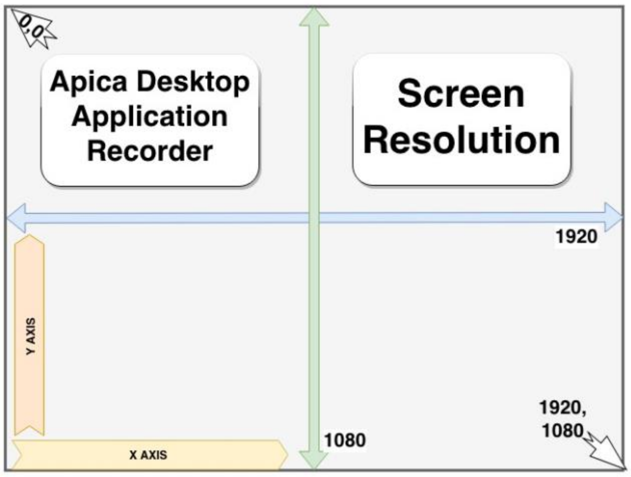
2.1 Screen resolution and screen coordinates

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 *assertTextAt*(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 located in the set default scenarios folder.

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. Apica recommends using 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 some_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"
```

In case the application you are monitoring locks up, and is left hanging, it may not be possible to start it the next time. In this case it could help to start the scenario by killing the old process (application_name.exe in the example below), using the *startApplication* command to start the Windows command prompt, with the following arguments (see command reference)

```
"application": "C:\\Windows\\System32\\cmd.exe",

"applicationArguments": "/c start taskkill.exe /f /im application_name.exe"
```

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.

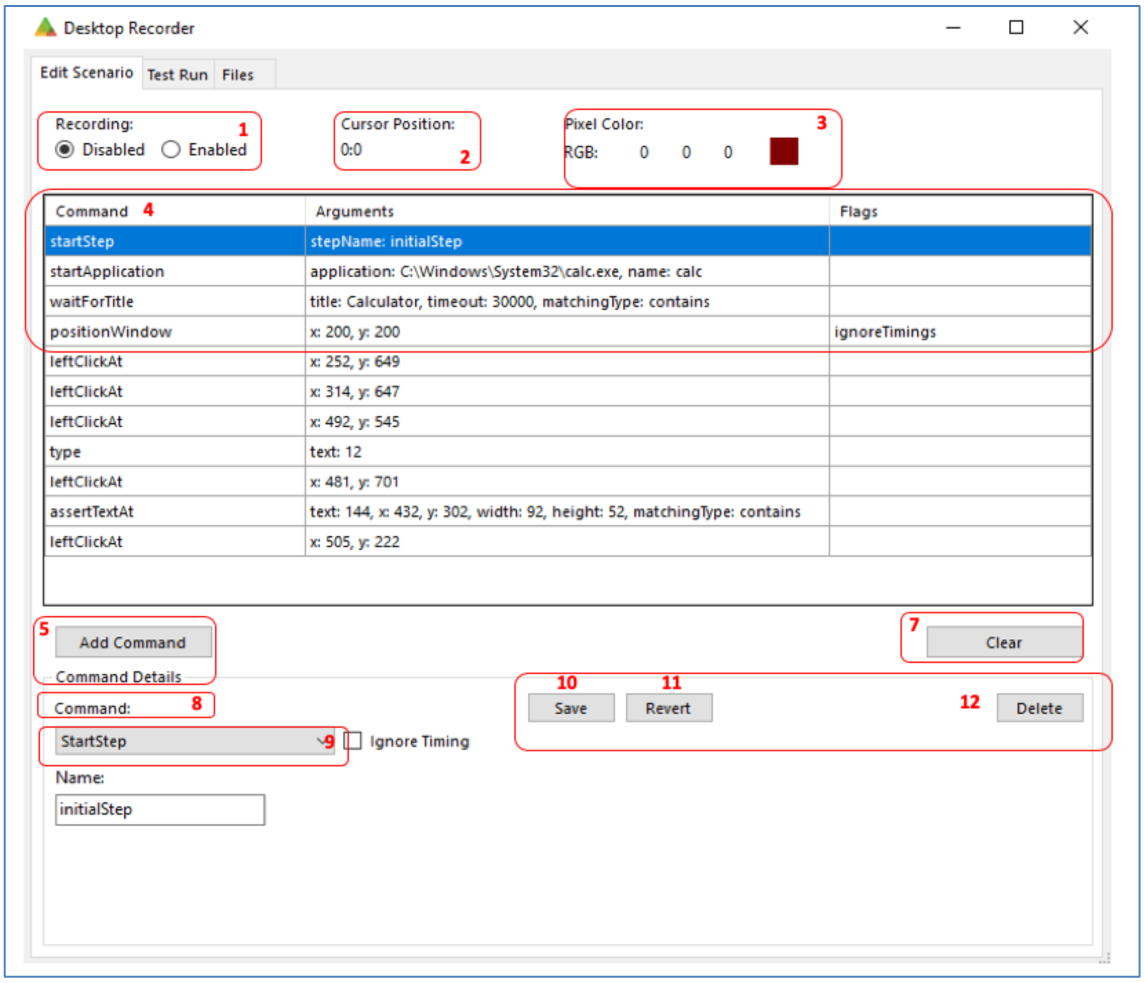
4 The Desktop Application Recorder User Interface

The main interface of the application has three tabs:

- Edit Scenario
- Test Run
- Files

4.1 Edit Scenario Tab

Recording and editing scenarios.



Recording: [✓ Disabled] Enabled

Cursor Position: 0:0

Pixel Color: RGB: 0 0 0 🟥

Command	Arguments	Flags
startStep	stepName: initialStep	
startApplication	application: C:\Windows\System32\calc.exe, name: calc	
waitForTitle	title: Calculator, timeout: 30000, matchingType: contains	
positionWindow	x: 200, y: 200	ignoreTimings
leftClickAt	x: 252, y: 649	
leftClickAt	x: 314, y: 647	
leftClickAt	x: 492, y: 545	
type	text: 12	
leftClickAt	x: 481, y: 701	
assertTextAt	text: 144, x: 432, y: 302, width: 92, height: 52, matchingType: contains	
leftClickAt	x: 505, y: 222	

Command Details

Command: StartStep

☐ Ignore Timing

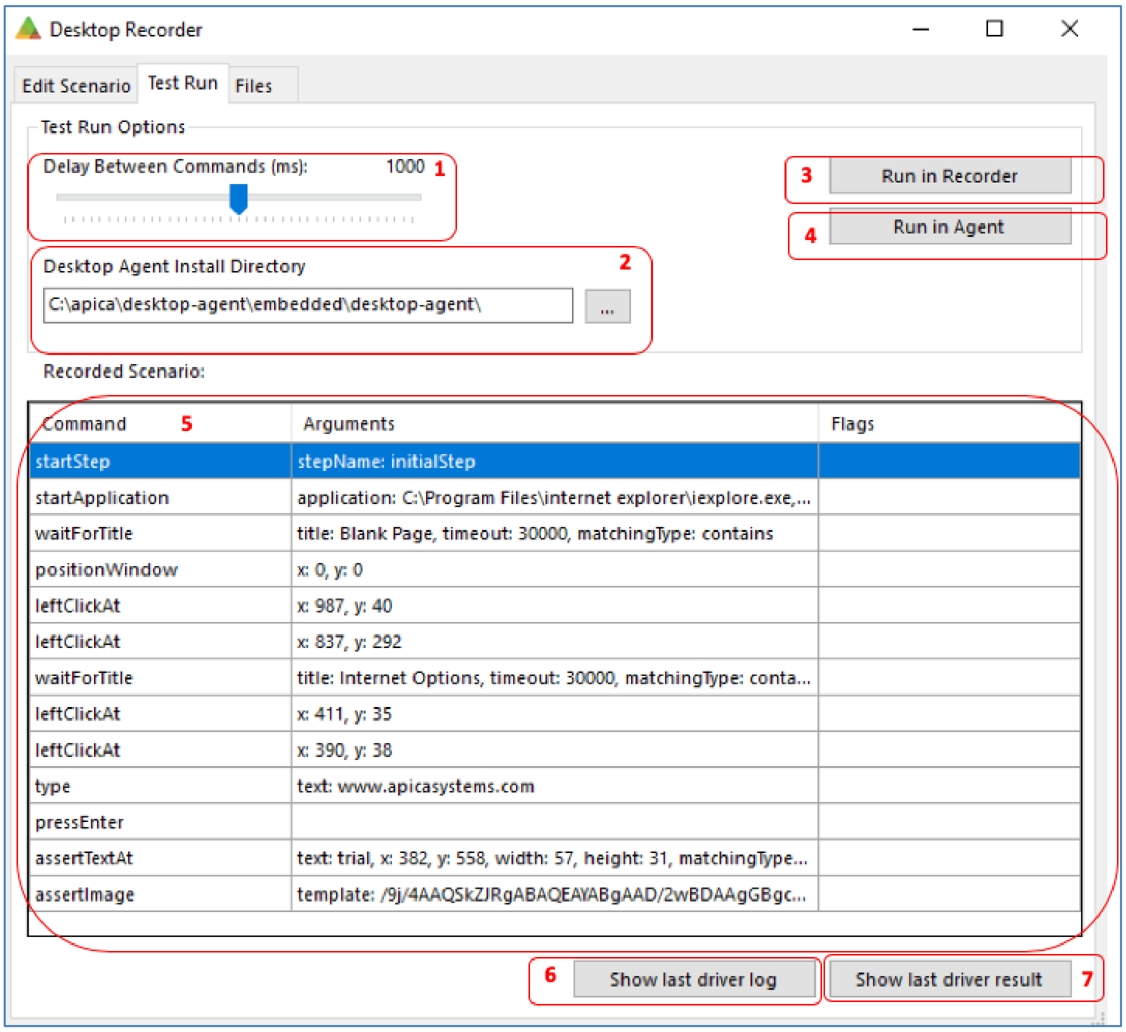
Name: initialStep

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.



Delay Between Commands (ms): 1000

Desktop Agent Install Directory

C:\apica\desktop-agent\embedded\desktop-agent\

Recorded Scenario:

Command	Arguments	Flags
startStep	stepName: initialStep	
startApplication	application: C:\Program Files\internet explorer\iexplore.exe,...	
waitForTitle	title: Blank Page, timeout: 30000, matchingType: contains	
positionWindow	x: 0, y: 0	
leftClickAt	x: 987, y: 40	
leftClickAt	x: 837, y: 292	
waitForTitle	title: Internet Options, timeout: 30000, matchingType: conta...	
leftClickAt	x: 411, y: 35	
leftClickAt	x: 390, y: 38	
type	text: www.apicasystems.com	
pressEnter		
assertTextAt	text: trial, x: 382, y: 558, width: 57, height: 31, matchingType...	
assertImage	template: /9j/4AAQSkZJRgABAQEAYABgAAD/2wBDAAgGBgc...	

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

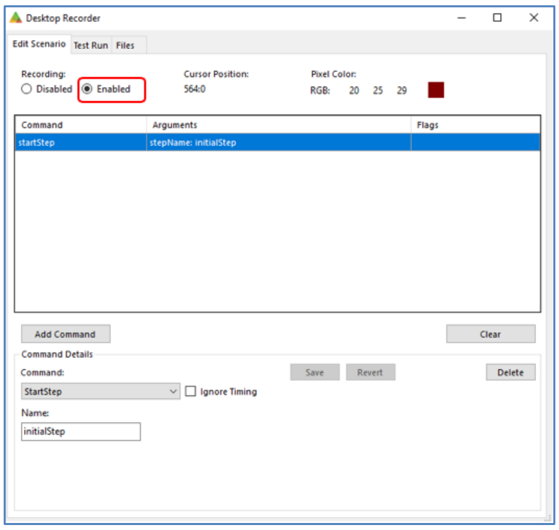
Opens (loads), or Saves, the scenario in JSON format. Files Guide:

	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*.

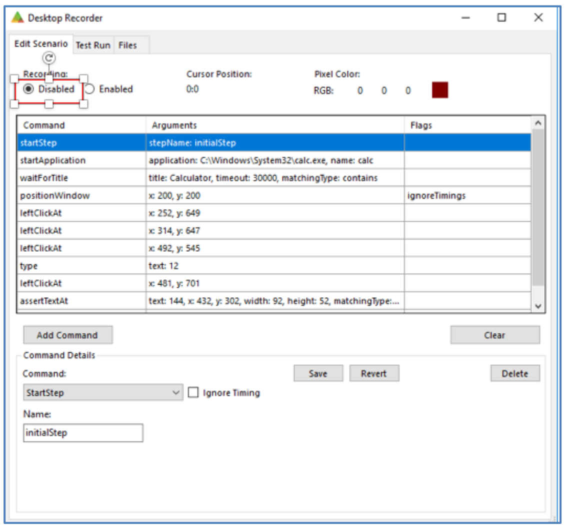
To start the recording select the "Enabled" radio button, under "Automatic Recording".



Recording: Disabled [✓ Enabled]
Cursor Position: 564:0
Pixel Color: RGB: 20 25 29 🟥

Command	Arguments	Flags
startStep	stepName: initialStep	

Command Details
Command: StartStep ☐ Ignore Timing
Name: initialStep



Recording: [✓ Disabled] Enabled
Cursor Position: 0:0
Pixel Color: RGB: 0 0 0 🟥

Command	Arguments	Flags
startStep	stepName: initialStep	
startApplication	application: C:\Windows\System32\calc.exe, name: calc	
waitForTitle	title: Calculator, timeout: 30000, matchingType: contains	
positionWindow	x: 200, y: 200	ignoreTimings
leftClickAt	x: 252, y: 649	
leftClickAt	x: 314, y: 647	
leftClickAt	x: 492, y: 545	
type	text: 12	
leftClickAt	x: 481, y: 701	
assertTextAt	text: 144, x: 432, y: 302, width: 92, height: 52, matchingType:...	

Command Details
Command: StartStep ☐ Ignore Timing
Name: initialStep

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

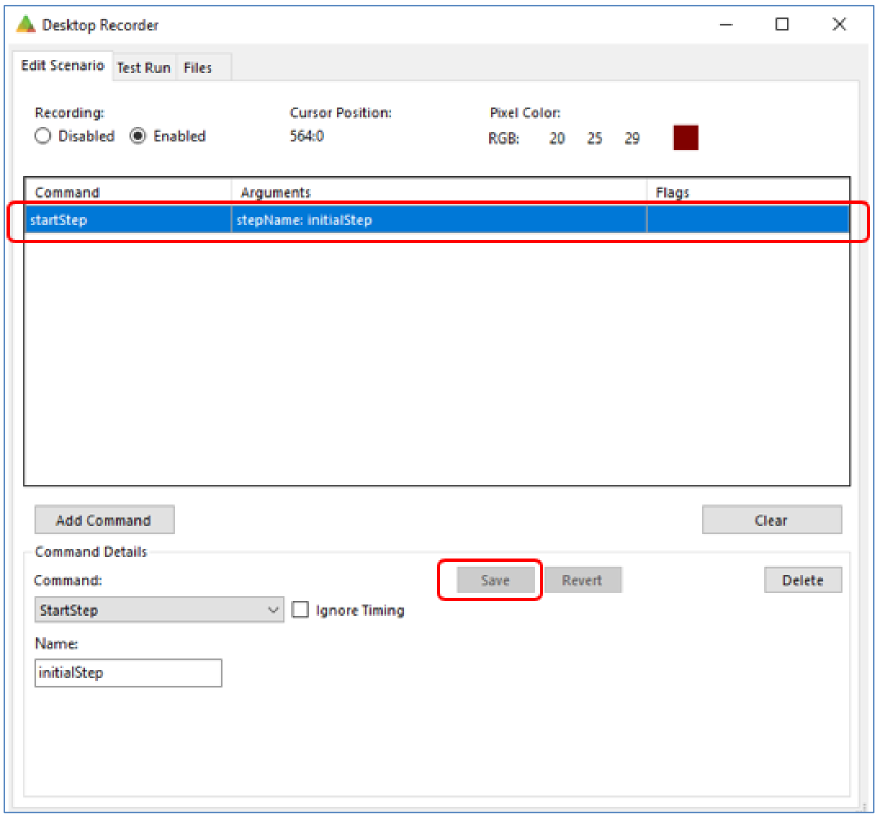
Note: Mouse-clicks on/in the Desktop Recorder are not recorded.

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.



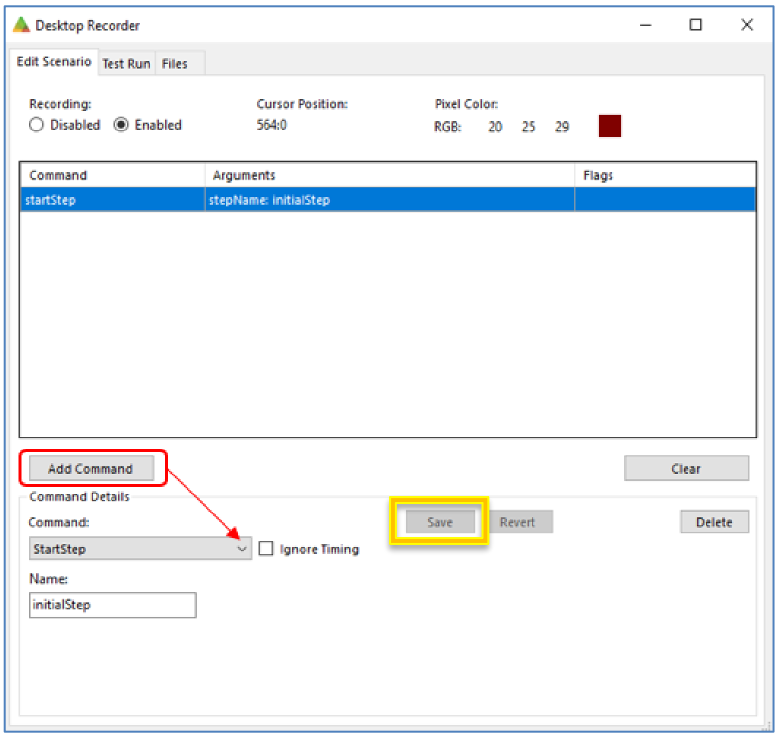
Recording: Disabled [✓ Enabled]
Cursor Position: 564:0
Pixel Color: RGB: 20 25 29 🟥

Command	Arguments	Flags
startStep	stepName: initialStep	

Command Details
Command: StartStep
[ ] Ignore Timing
Name: initialStep

6.1 Add

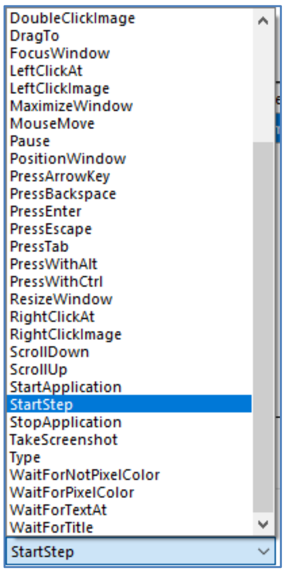
To add a new command, start by clicking "Add Command" and then select a command in the drop-down list labeled "Command".



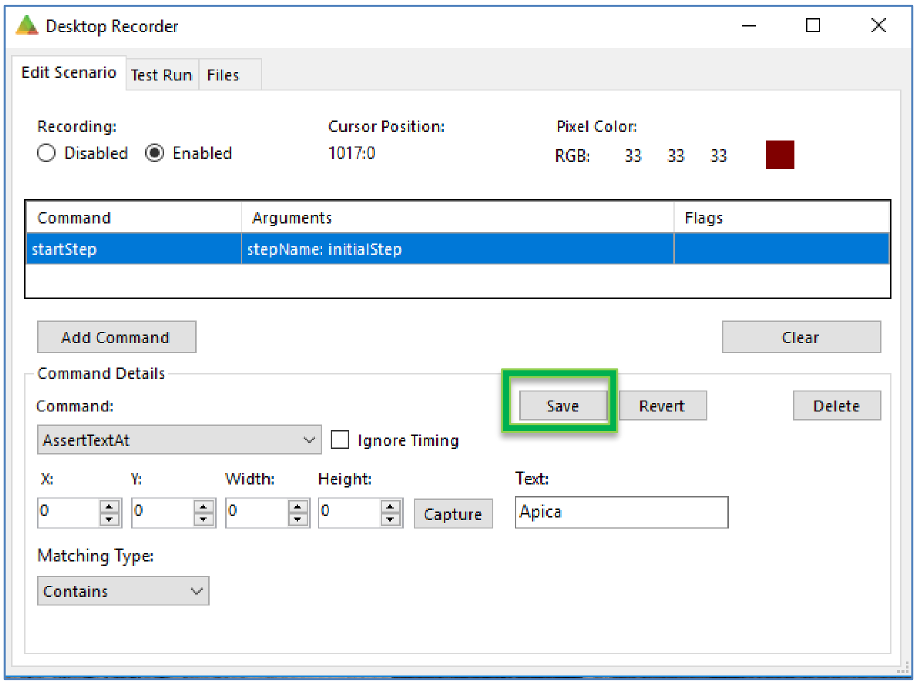
Command	Arguments	Flags
startStep	stepName: initialStep	

Command: StartStep

Name: initialStep



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


Desktop Recorder

Edit Scenario | Test Run | Files

Recording: ○ Disabled [✓ Enabled]

Cursor Position: 1017:0

Pixel Color: RGB: 33 33 33 🟥

Command	Arguments	Flags
startStep	stepName: initialStep	

Command Details

Command: AssertTextAt

☐ Ignore Timing

X: 0 Y: 0 Width: 0 Height: 0

Text: Apica

Matching Type: Contains

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

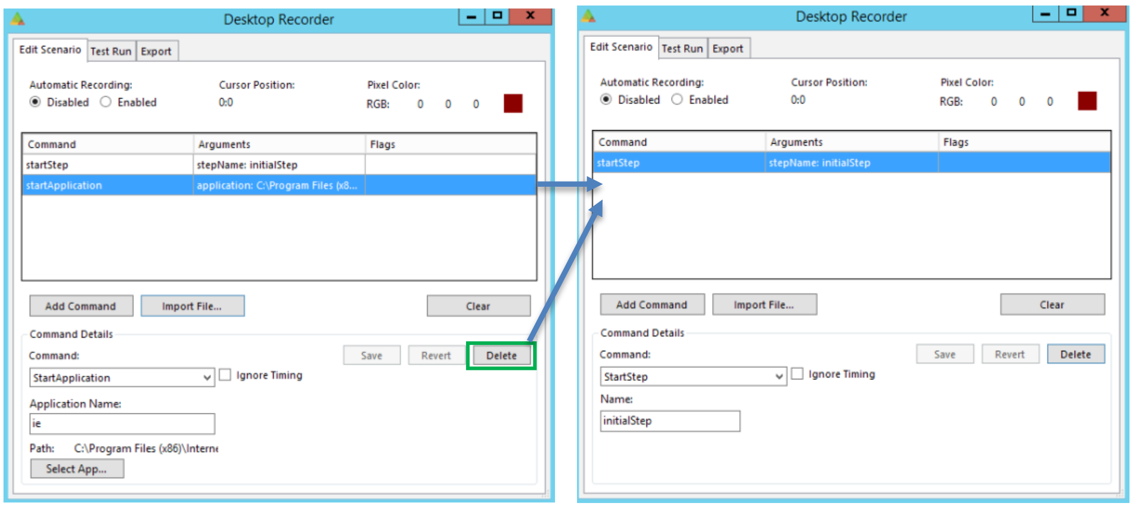
Once you have entered values for all required fields, click "Save" to add the command to your scenario.

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.



Automatic Recording: (● Disabled) ○ Enabled
Cursor Position: 0:0
Pixel Color: RGB: 0 0 0

Command	Arguments	Flags
startStep	stepName: initialStep	
startApplication	application: C:\Program Files (x8...	

Command Details
Command: StartApplication ☐ Ignore Timing
Application Name: ie
Path: C:\Program Files (x86)\Interne

Automatic Recording: (● Disabled) ○ Enabled
Cursor Position: 0:0
Pixel Color: RGB: 0 0 0

Command	Arguments	Flags
startStep	stepName: initialStep	

Command Details
Command: StartStep ☐ Ignore Timing
Name: initialStep

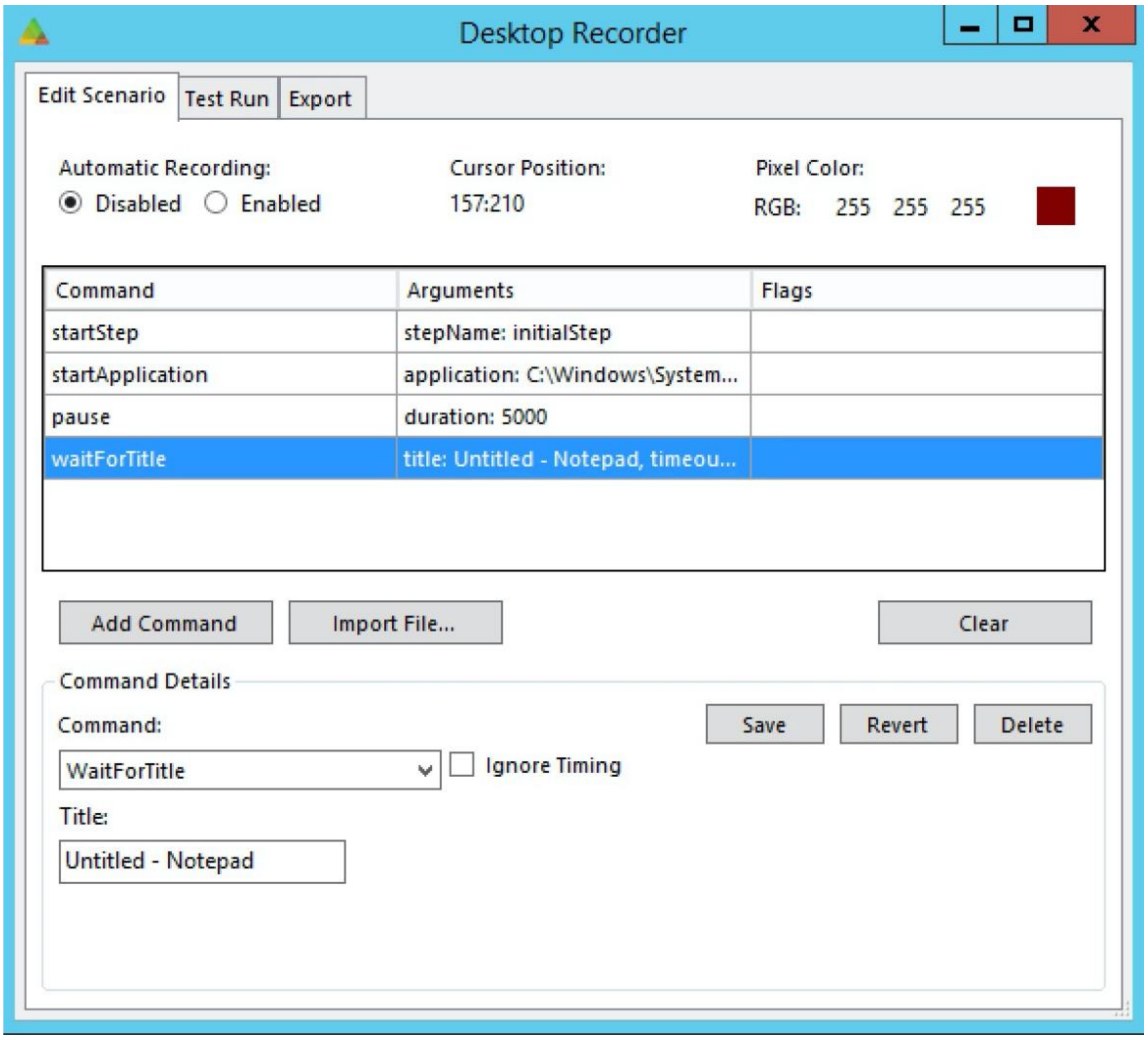
7 Validation and Flow Control

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 *matchingType* 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- Internet Explorer".



Automatic Recording: [● Disabled] ○ Enabled

Cursor Position: 157:210

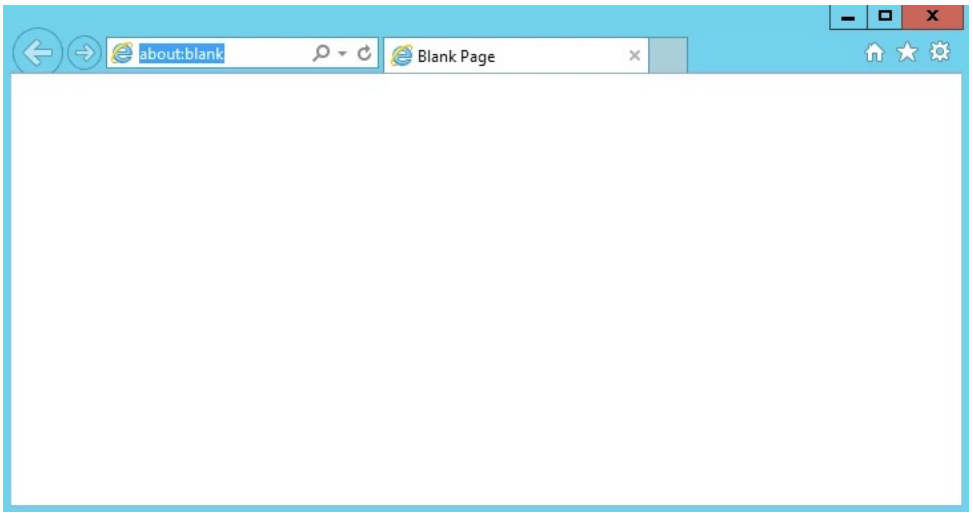
Pixel Color: RGB: 255 255 255 🟥

Command	Arguments	Flags
startStep	stepName: initialStep	
startApplication	application: C:\Windows\System...	
pause	duration: 5000	
waitForTitle	title: Untitled - Notepad, timeou...	

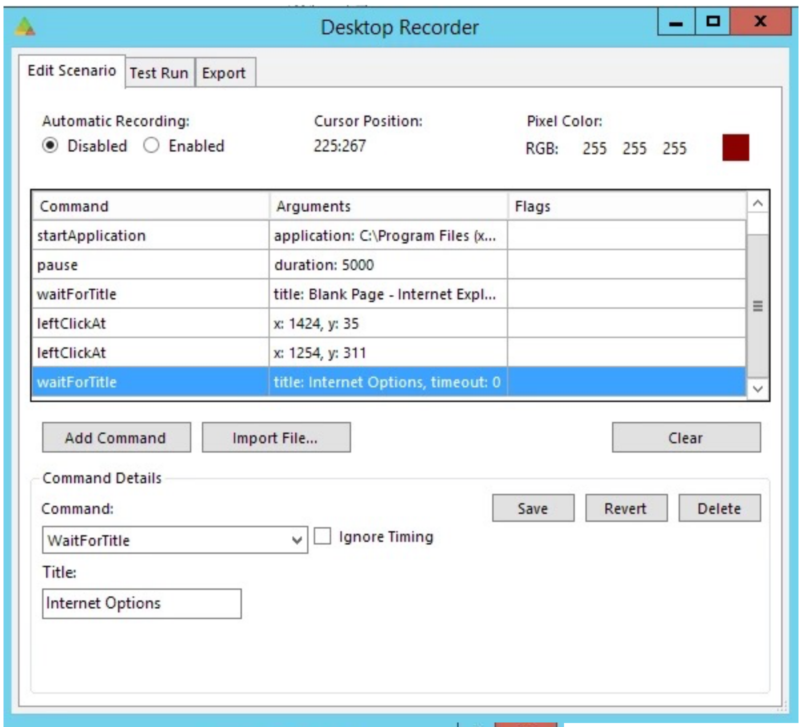
Command Details

Command: WaitForTitle ☐ Ignore Timing

Title: Untitled - Notepad



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



Automatic Recording: (●) Disabled ( ) Enabled

Cursor Position: 225:267

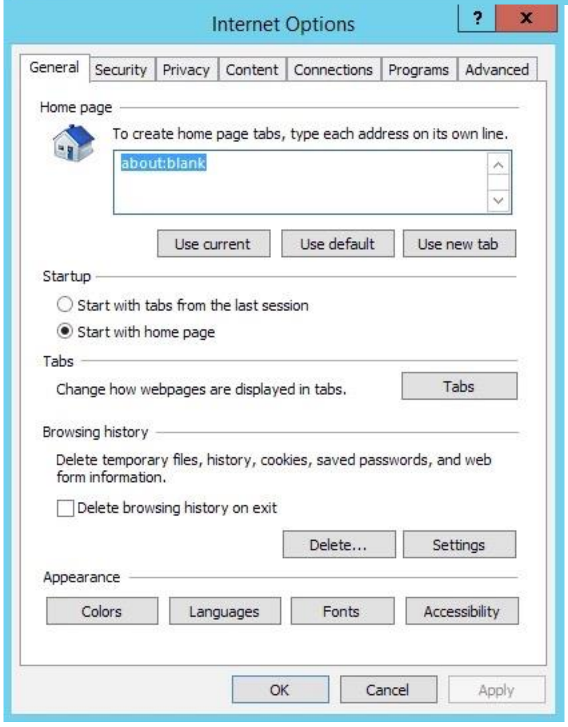
Pixel Color: RGB: 255 255 255 🟥

Command	Arguments	Flags
startApplication	application: C:\Program Files (x...	
pause	duration: 5000	
waitForTitle	title: Blank Page - Internet Expl...	
leftClickAt	x: 1424, y: 35	
leftClickAt	x: 1254, y: 311	
waitForTitle	title: Internet Options, timeout: 0	

Command Details

Command: WaitForTitle ☐ Ignore Timing

Title: Internet Options



Home page

To create home page tabs, type each address on its own line.

about:blank

Startup

( ) Start with tabs from the last session

(●) Start with home page

Tabs

Change how webpages are displayed in tabs.

Browsing history

Delete temporary files, history, cookies, saved passwords, and web form information.

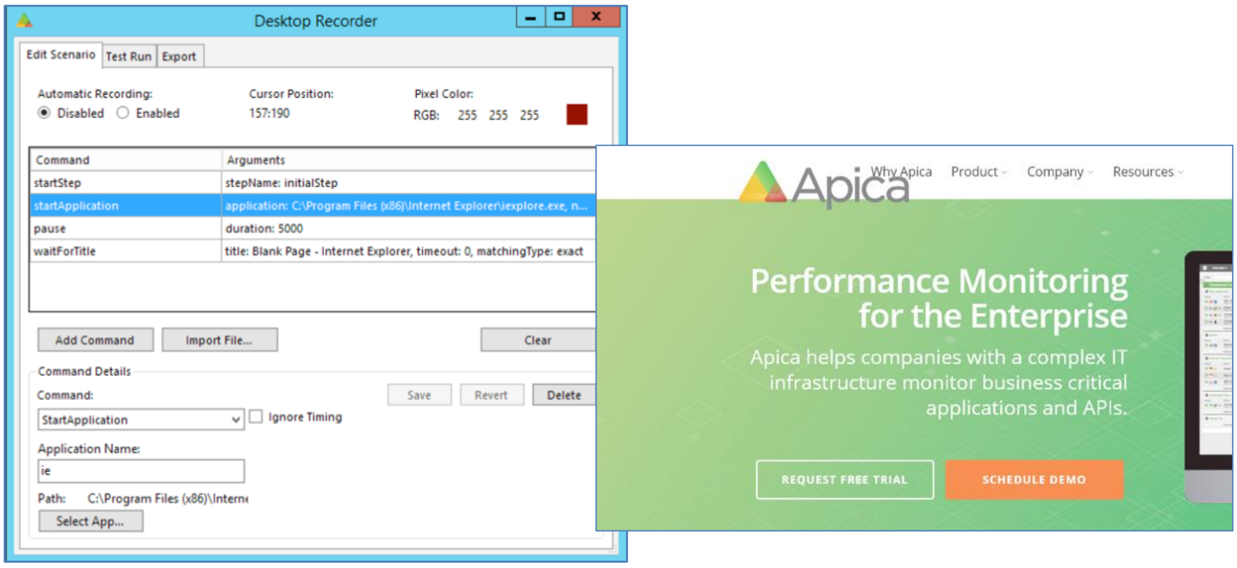
☐ Delete browsing history on exit

Appearance

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



Automatic Recording: (● Disabled) ○ Enabled

Cursor Position: 157:190

Pixel Color: RGB: 255 255 255 🟥

Command	Arguments
startStep	stepName: initialStep
startApplication	application: C:\Program Files (x86)\Internet Explorer\iexplore.exe, n...
pause	duration: 5000
waitForTitle	title: Blank Page - Internet Explorer, timeout: 0, matchingType: exact

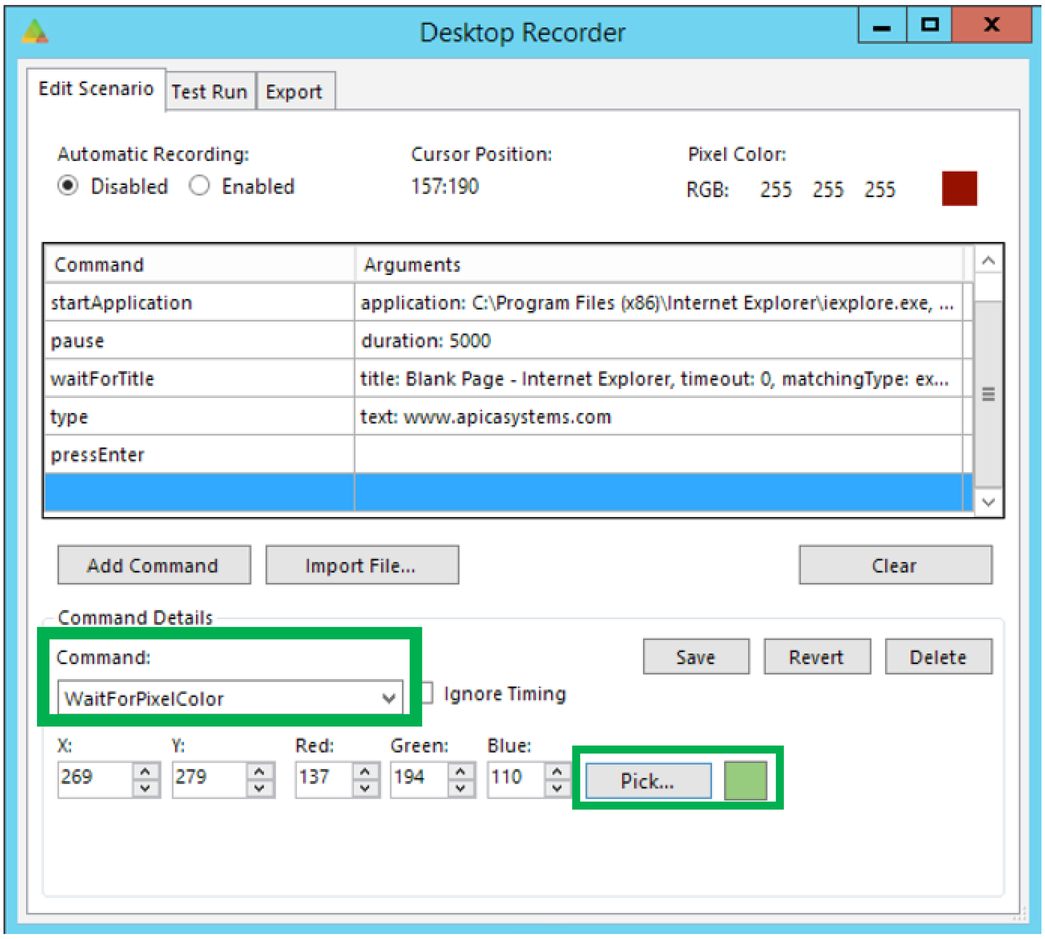
Command Details

Command: StartApplication ☐ Ignore Timing

Application Name: ie

Path: C:\Program Files (x86)\Interne

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



Automatic Recording: [✓ Disabled] Enabled

Cursor Position: 157:190

Pixel Color: RGB: 255 255 255 🟥

Command	Arguments
startApplication	application: C:\Program Files (x86)\Internet Explorer\iexplore.exe, ...
pause	duration: 5000
waitForTitle	title: Blank Page - Internet Explorer, timeout: 0, matchingType: ex...
type	text: www.apicasystems.com
pressEnter	

Command Details

Command: WaitForPixelColor

Ignore Timing

X:	Y:	Red:	Green:	Blue:
269	279	137	194	110

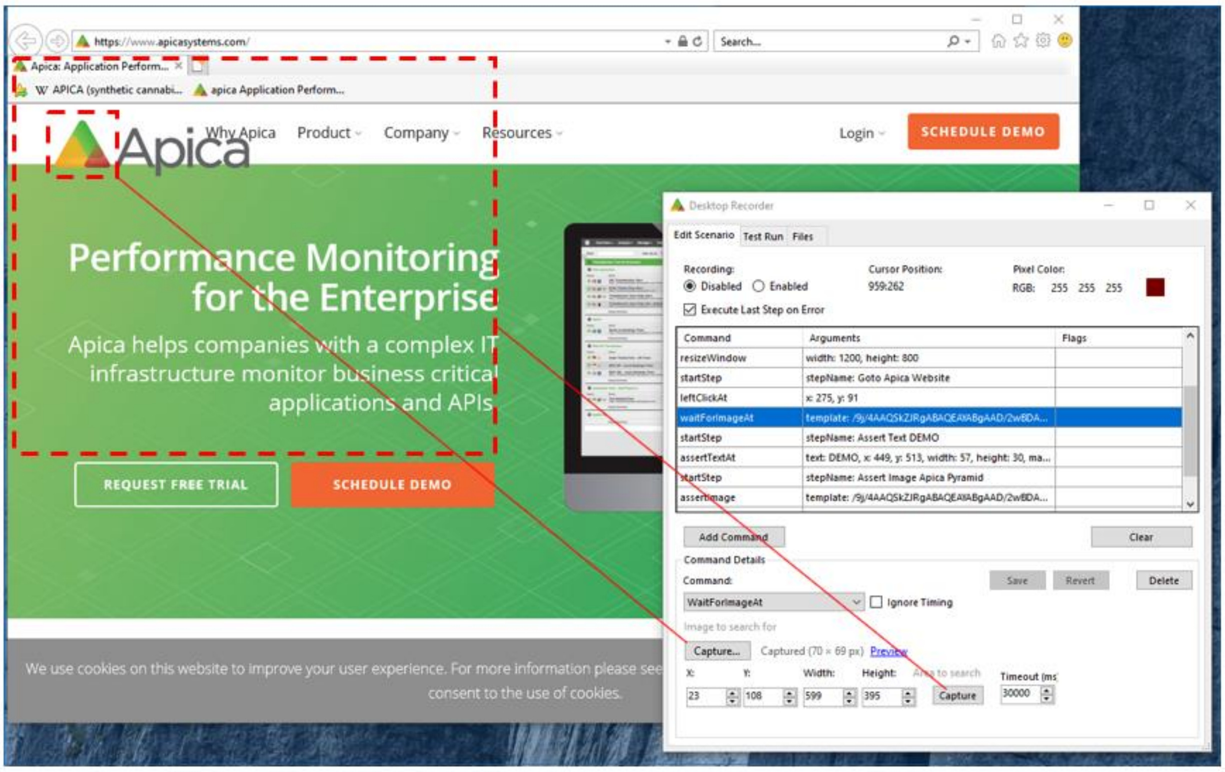
changes.

7.3 Wait for an Image to Appear on the Screen

If you need to wait for a specific image to appear on the screen, e.g. an icon, you can use *waitForImageAt* command.

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

Next, we add a "*waitForImageAt*" command, click "Capture" (image to search for) and then draw a rectangle around e.g. the Apica triangle. Then click the (area to search) “Capture” and draw a rectangle around the area where you expect the image to appear. This should be a bigger area than the image, to be able to work when the image changes position. When the scenario runs the scenario execution will pause while the command “polls” for the image to appear somewhere in the selected area, until it is found, or the command times out. To make sure that the script fails when an image is not detected an assertimage command should be used in combination with waitForImageAt.


https://www.apicasystems.com/
Apica: Application Perform...
W APICA (synthetic cannabi...
apica Application Perform...
Apica
Why Apica
Product
Company
Resources
Login
SCHEDULE DEMO
Performance Monitoring for the Enterprise
Apica helps companies with a complex IT infrastructure monitor business critical applications and APIs
REQUEST FREE TRIAL
SCHEDULE DEMO
We use cookies on this website to improve your user experience. For more information please see consent to the use of cookies.
Desktop Recorder
Edit Scenario
Test Run
Files
Add Command
Clear
Save
Revert
Delete
Capture...
Preview
Capture

Recording: [● Disabled] ○ Enabled
Cursor Position: 959:262
Pixel Color: RGB: 255 255 255
☑ Execute Last Step on Error

Command	Arguments	Flags
resizeWindow	width: 1200, height: 800	
startStep	stepName: Goto Apica Website	
leftClickAt	x: 275, y: 91	
waitForImageAt	template: /9j/4AAQSkZJRgABAQEAYABgAAD/2wBDA...	
startStep	stepName: Assert Text DEMO	
assertTextAt	text: DEMO, x: 449, y: 513, width: 57, height: 30, ma...	
startStep	stepName: Assert Image Apica Pyramid	
assertImage	template: /9j/4AAQSkZJRgABAQEAYABgAAD/2wBDA...	

Command Details
Command: WaitForImageAt ☐ Ignore Timing
Image to search for
Captured (70 × 69 px)
X: 23 Y: 108 Width: 599 Height: 395 Area to search
Timeout (ms 30000

7.4 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.

7.5 assertText

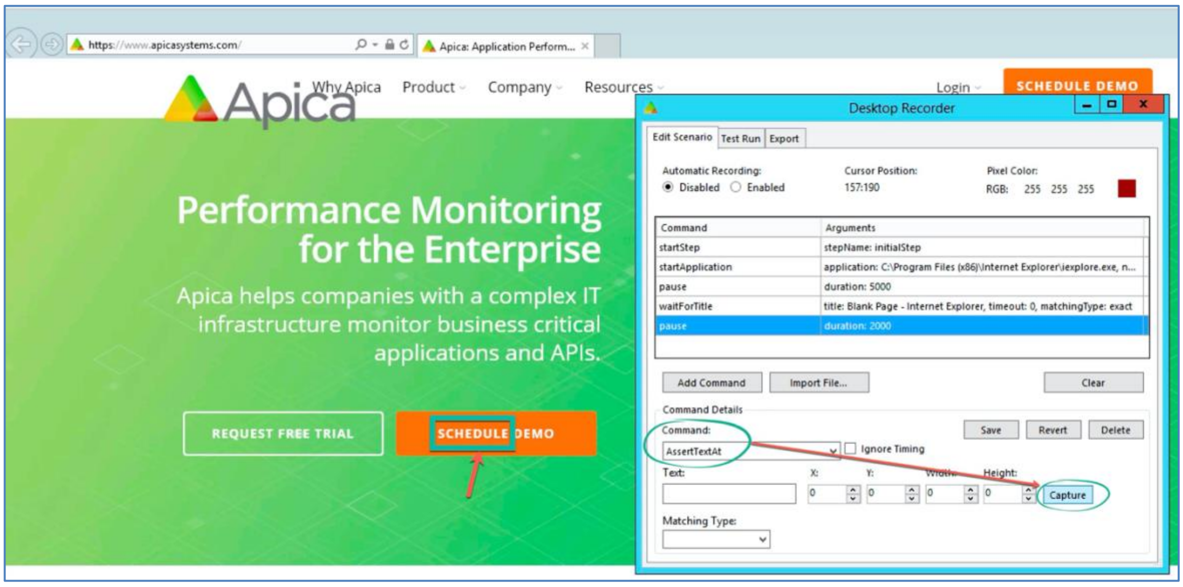
The *assertText* command captures a screenshot of the currently focused application window, feeds the image to the Tesseract OCR engine which returns any text detected. The *Text* argument is then compared to the text returned by Tesseract. If the returned text matches, the command is successful. You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

7.6 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 the returned text matches, the command is successful. You can also use the *matchingType* argument to select matching type: exact, glob (global expressions), or contains.

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, make 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 this 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 "Schedule" in the SCHEDULE DEMO button), when the mouse button is released the WIDTH,HEIGHT,X and Y arguments should be updated automatically.

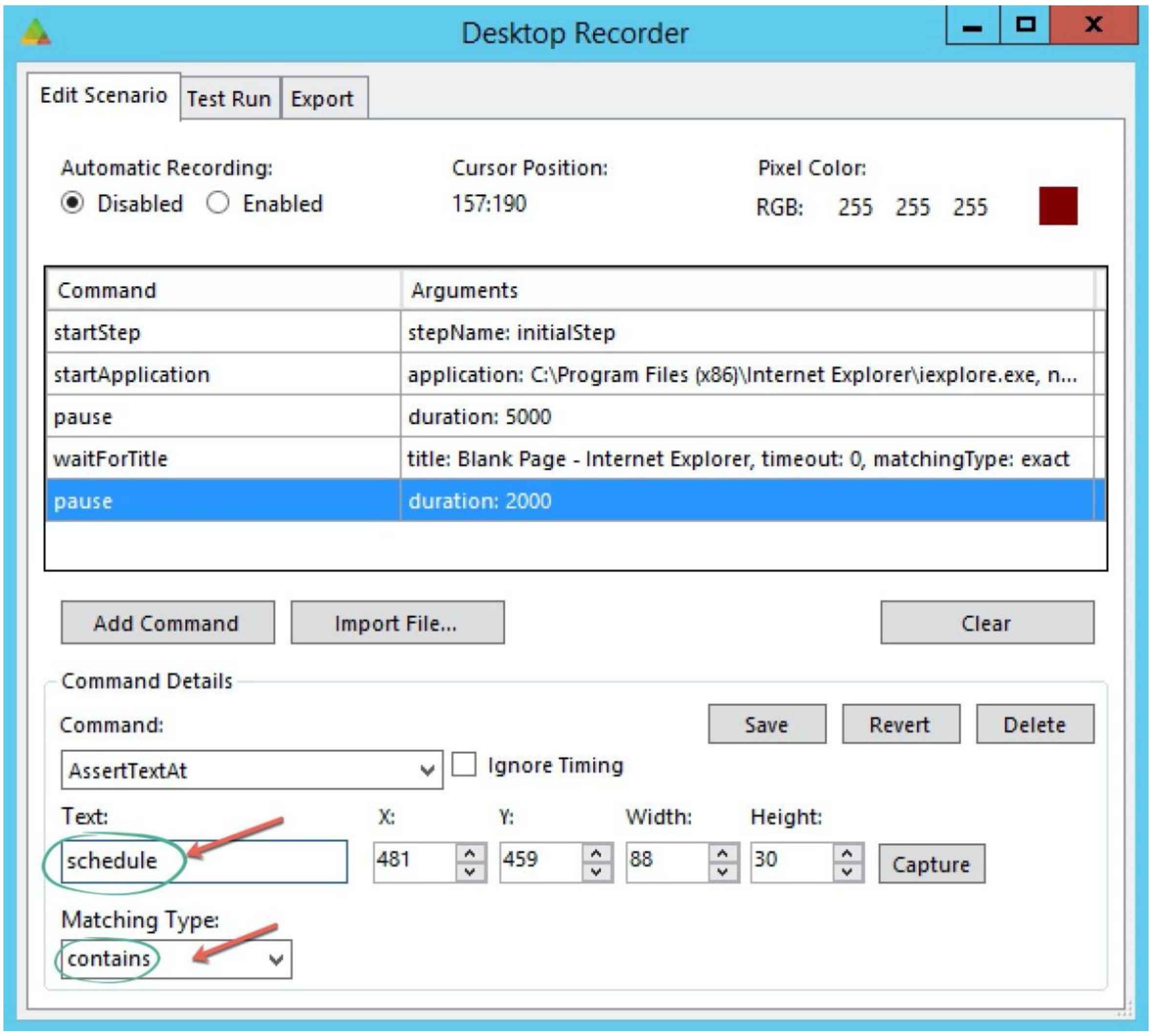

https://www.apicasystems.com/
Apica: Application Perform...
Apica
Why Apica
Product
Company
Resources
Login
SCHEDULE DEMO
Performance Monitoring for the Enterprise
Apica helps companies with a complex IT infrastructure monitor business critical applications and APIs.
REQUEST FREE TRIAL
SCHEDULE DEMO
Desktop Recorder
Edit Scenario
Test Run
Export
Add Command
Import File...
Clear
Save
Revert
Delete
Capture

Automatic Recording: [✓ Disabled] Enabled
Cursor Position: 157:190
Pixel Color: RGB: 255 255 255

Command	Arguments
startStep	stepName: initialStep
startApplication	application: C:\Program Files (x86)\Internet Explorer\iexplore.exe, n...
pause	duration: 5000
waitForTitle	title: Blank Page - Internet Explorer, timeout: 0, matchingType: exact
pause	duration: 2000

Command Details
Command: AssertTextAt
Ignore Timing
Text:
X: 0
Y: 0
Width: 0
Height: 0
Matching Type:

Next, in the input field labeled "Text", we enter a string of text (e.g. “schedule”) 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.



Automatic Recording: [✓ Disabled] Enabled

Cursor Position: 157:190

Pixel Color: RGB: 255 255 255 🟥

Command	Arguments
startStep	stepName: initialStep
startApplication	application: C:\Program Files (x86)\Internet Explorer\iexplore.exe, n...
pause	duration: 5000
waitForTitle	title: Blank Page - Internet Explorer, timeout: 0, matchingType: exact
pause	duration: 2000

Command: AssertTextAt

Ignore Timing: ☐

Text:	X:	Y:	Width:	Height:
schedule	481	459	88	30

Matching Type: contains

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

8 Assert that Image is Present on Screen

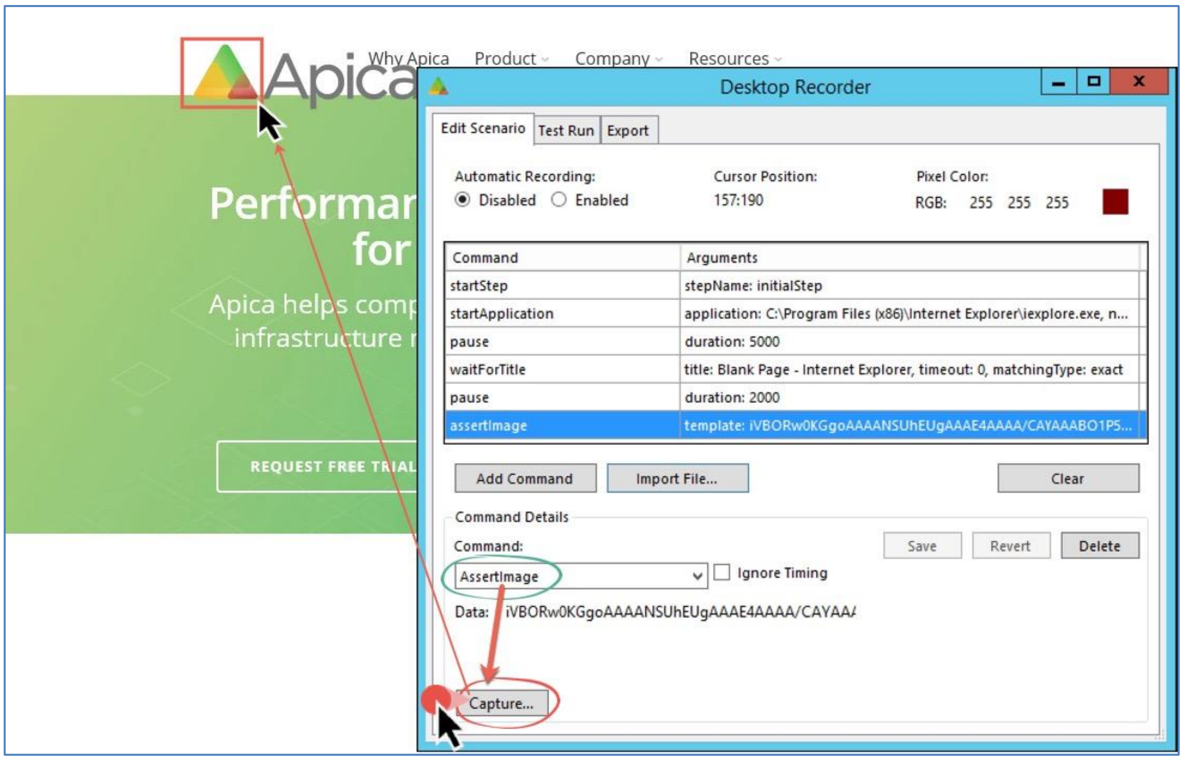
8.1 assertImage

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 loading www.apicasystems.com. Next we add an *assertImage* command and click "Capture", this adds the Desktop Recorder 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. When the scenario runs, the scenario execution will be stopped if we cannot detect the Apica logo on the screen.



Automatic Recording: (● Disabled) ○ Enabled

Cursor Position: 157:190

Pixel Color: RGB: 255 255 255

Command	Arguments
startStep	stepName: initialStep
startApplication	application: C:\Program Files (x86)\Internet Explorer\iexplore.exe, n...
pause	duration: 5000
waitForTitle	title: Blank Page - Internet Explorer, timeout: 0, matchingType: exact
pause	duration: 2000
assertImage	template: iVBORw0KGgoAAAANSUhEUgAAAE4AAAA/CAYAAABO1P5...

Command Details

Command: AssertImage ☐ Ignore Timing

Data: iVBORw0KGgoAAAANSUhEUgAAAE4AAAA/CAYAA

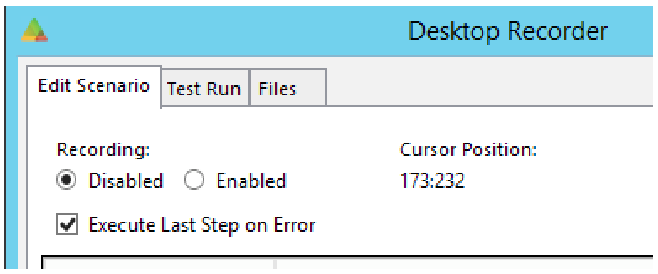
9 Pause Scenario Execution

9.1 pause

In some cases, you need to pause the execution of the script for a certain amount of time were the ”waitfor” commands cannot be used. In these cases use the *pause* command to pause the scenario execution for X amount of millisecond

9.2 Execute Last Step on Error

Check this box to force the agent/recorder to run the last step of the scenario, even if the test failed due to a scenario error. If this box is checked, the last step of the scenario may be used to clean-up, or to make sure sessions are reset as expected.



Recording: [✓ Disabled] Enabled

Cursor Position: 173:232

✓ Execute Last Step on Error

10 Variables

The following commands may be used to store values in variables for scenario Flow Control.

- *store*
- *storeTextAt*
- *storeEval*
- *storePixelColor*
- *storeTitle*

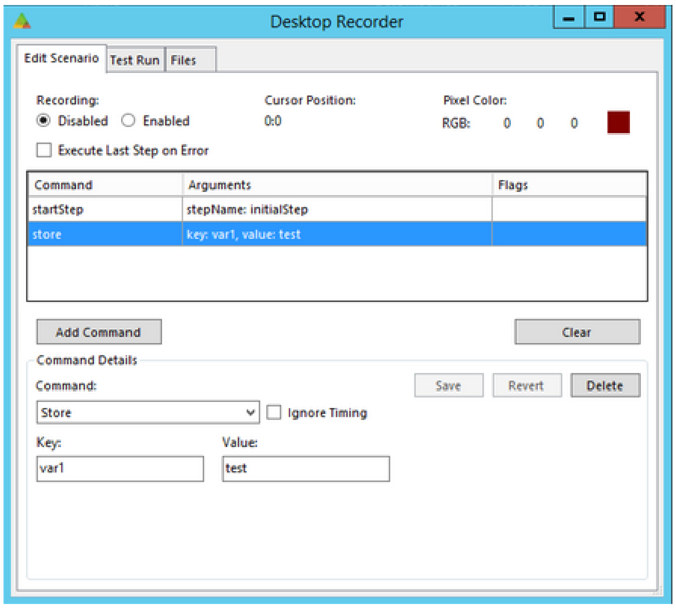
These variables can be referred to in place of string arguments for other commands in the recorder, which are then replaced at runtime with the actual value stored in the corresponding variable.

- ${var_name}

Note: The exception is as the argument for the commands, "label" and "startStep", which are not supported.

10.1 Store a (predefined) string in a Variable

Use the *store* command to store a string in a variable, for later use in the scenario:



Recording: [✓ Disabled] Enabled

Cursor Position: 0:0

Pixel Color: RGB: 0 0 0

Execute Last Step on Error

Command	Arguments	Flags
startStep	stepName: initialStep	
store	key: var1, value: test	

Command Details

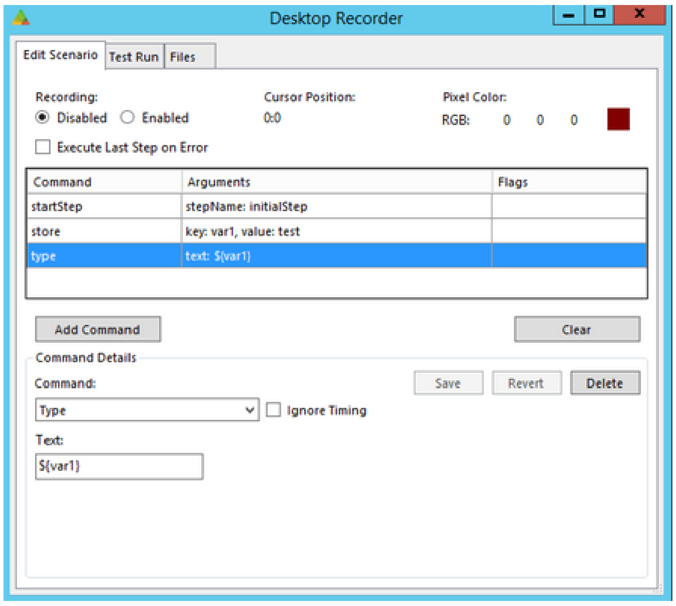
Command: Store

Ignore Timing

Key: var1

Value: test

Then refer to it like this:



Desktop Recorder

Edit Scenario | Test Run | Files

Recording: [✓ Disabled] Enabled
Cursor Position: 0:0
Pixel Color: RGB: 0 0 0 🟥

☐ Execute Last Step on Error

Command	Arguments	Flags
startStep	stepName: initialStep	
store	key: var1, value: test	
type	text: ${var1}	

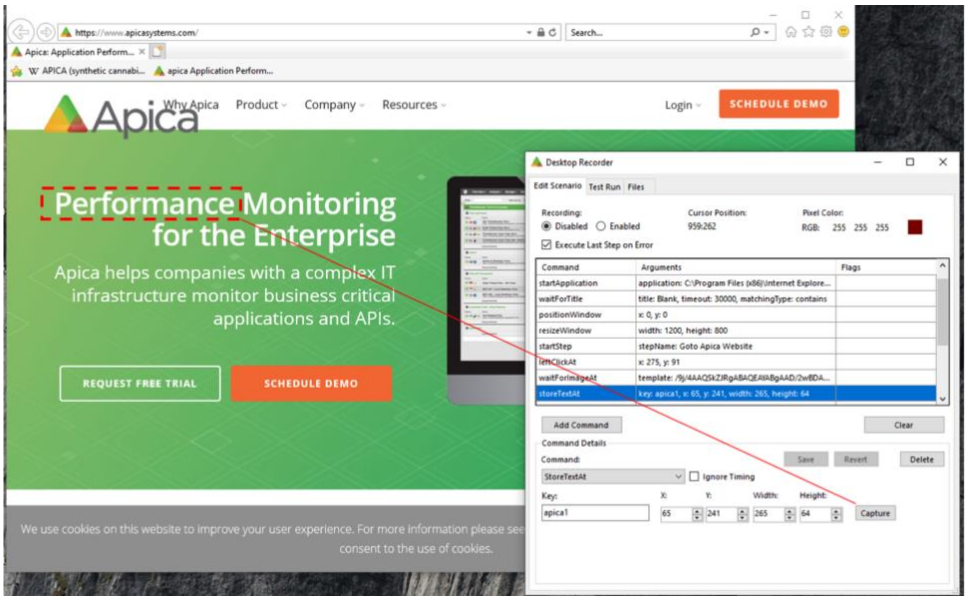
Command Details

Command: Type ☐ Ignore Timing

Text: ${var1}

10.2 Store an extracted string in a variable

Use the *storeTextAt* command to capture a string from the screen and store it in a variable. The capture part works the same way as for the *assertTextAt* and *waitForTextAt* commands.



Desktop Recorder

Edit Scenario | Test Run | Files

Recording: [✓ Disabled] Enabled
Cursor Position: 959:262
Pixel Color: RGB: 255 255 255 🟥

☑ Execute Last Step on Error

Command	Arguments	Flags
startApplication	application: C:\Program Files (x86)\Internet Explore...	
waitForTitle	title: Blank, timeout: 30000, matchingType: contains	
positionWindow	x: 0, y: 0	
resizeWindow	width: 1200, height: 800	
startStep	stepName: Goto Apica Website	
leftClickAt	x: 275, y: 91	
waitForImageAt	template: /9j/4AAQSkZJRgABAQEAYABgAAD/2wBDA...	
storeTextAt	key: apica1, x: 65, y: 241, width: 265, height: 64	

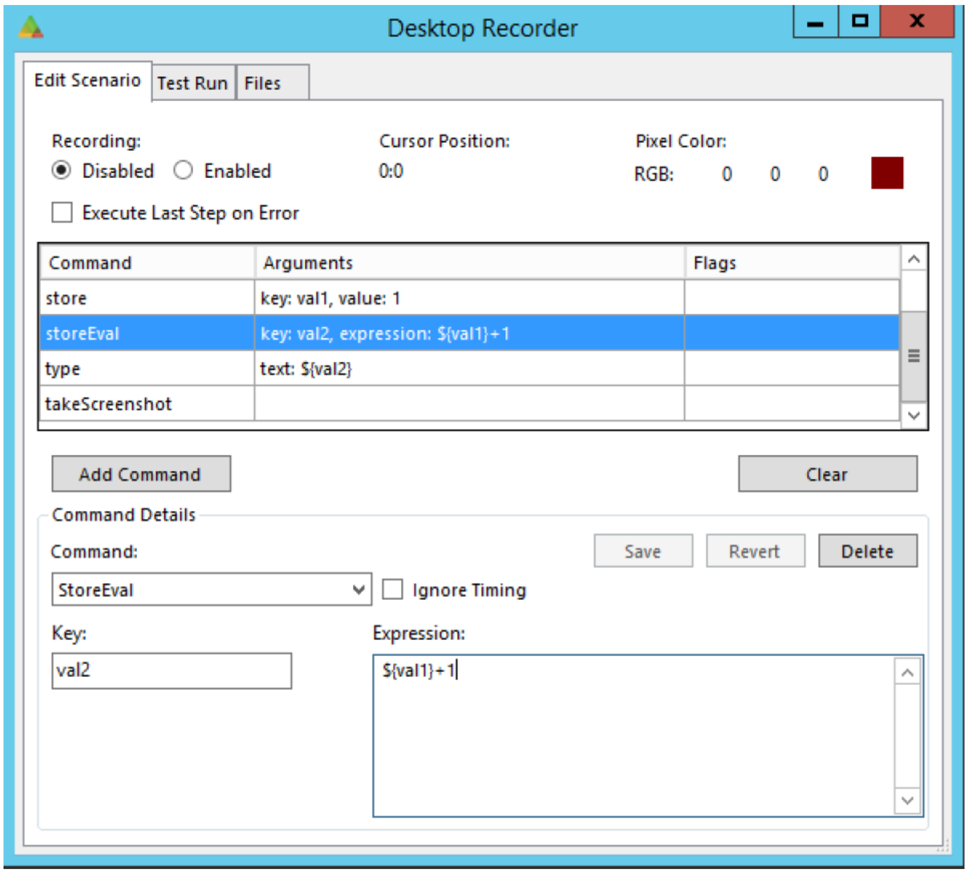
Command Details

Command: StoreTextAt ☐ Ignore Timing

Key: apica1 X: 65 Y: 241 Width: 265 Height: 64

10.3 storeEval

The *storeEval* command runs a JavaScript snippet, and then stores the result of evaluating the snippet in a new variable. This variable can then be used in the subsequent calls within the script as any other value.



Recording: [✓ Disabled] Enabled

Cursor Position: 0:0

Pixel Color: RGB: 0 0 0 🟥

☐ Execute Last Step on Error

Command	Arguments	Flags
store	key: val1, value: 1	
storeEval	key: val2, expression: ${val1}+1	
type	text: ${val2}	
takeScreenshot		

Command Details

Command: StoreEval ☐ Ignore Timing

Key: val2

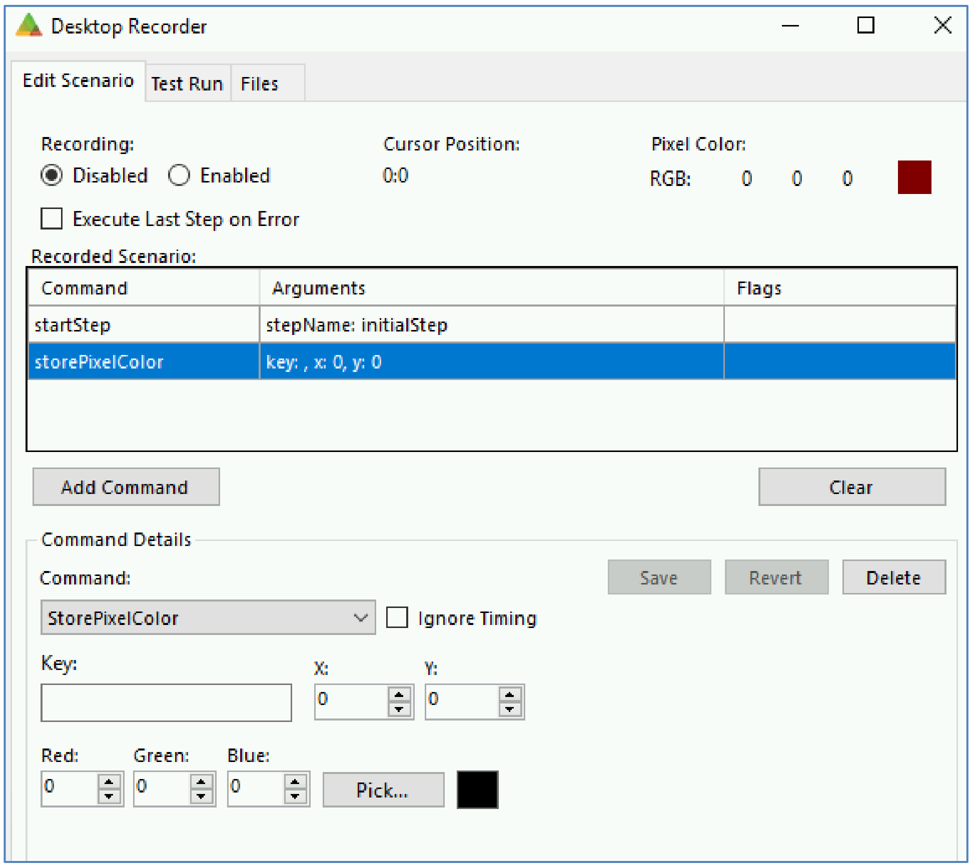
Expression: ${val1}+1

In this example we simply add 1 to the value of a previously stored variable (val1), and store the new value in another variable (val2)

At the end of the scenario we type the value stored in val2, which is 2 (the sum of the equation 1+1)

10.4 storePixelColor

Store the pixel color (RGB) at target coordinates. Note that the RGB value is stored as three separate variables: $key_red, $key_green, $key_blue.



Recording: [✓ Disabled] Enabled

Cursor Position: 0:0

Pixel Color: RGB: 0 0 0

☐ Execute Last Step on Error

Recorded Scenario:

Command	Arguments	Flags
startStep	stepName: initialStep	
storePixelColor	key: , x: 0, y: 0	

Command Details

Command: StorePixelColor ☐ Ignore Timing

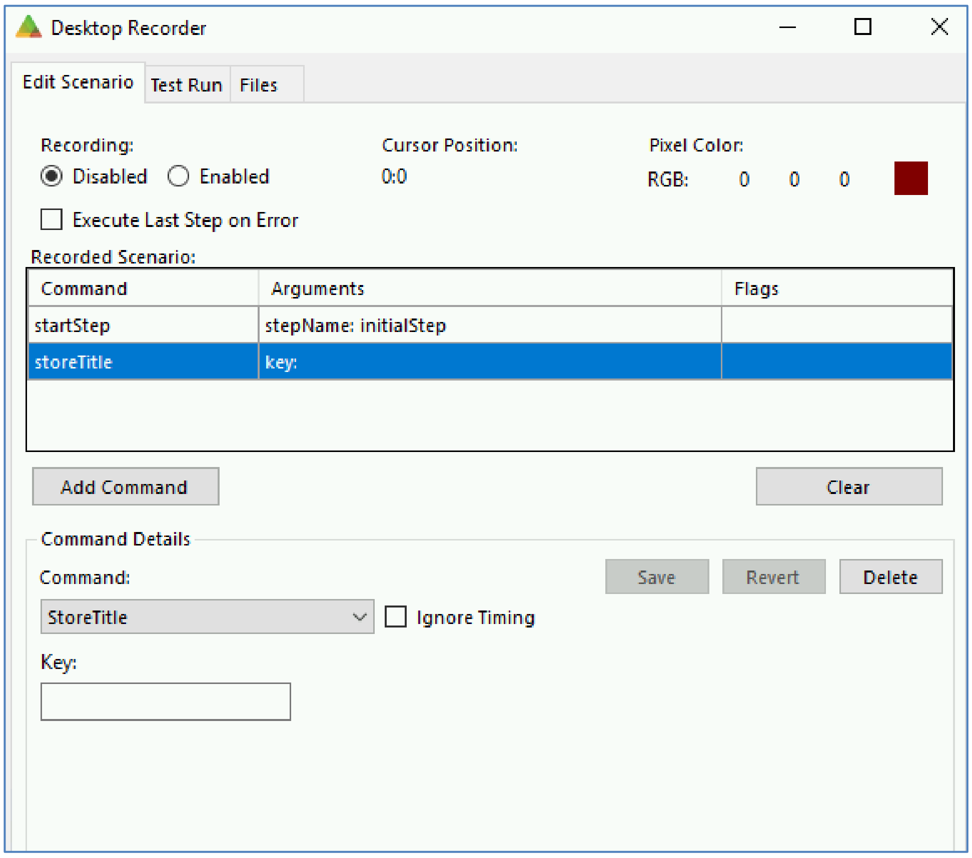
Key:

X: 0 Y: 0

Red: 0 Green: 0 Blue: 0

10.5 storeTitle

Store the title of the currently focused application window.



Recording: [● Disabled] ○ Enabled

Cursor Position: 0:0

Pixel Color: RGB: 0 0 0 🟥

☐ Execute Last Step on Error

Recorded Scenario:

Command	Arguments	Flags
startStep	stepName: initialStep	
storeTitle	key:	

Command Details

Command: StoreTitle ☐ Ignore Timing

Key:

10.6 JavaScripting in onCondition and storeEval commands

Note that *onCondition* and *storeEval* commands, both execute JavaScript code. If you have a string stored in a “dac” variable, that you wish to use in a JavaScript, you need to store it first as a string in your JavaScript code:

```
var s = "" + ${MyString}
```

If your variable contains an int or a bool value on the other hand, you can just use the dac var as is. In this example the variable demo_button contains the result of storeAssertImageAt

```
if ( ${demo_button}==true ) {return true;} else {return false;}
```

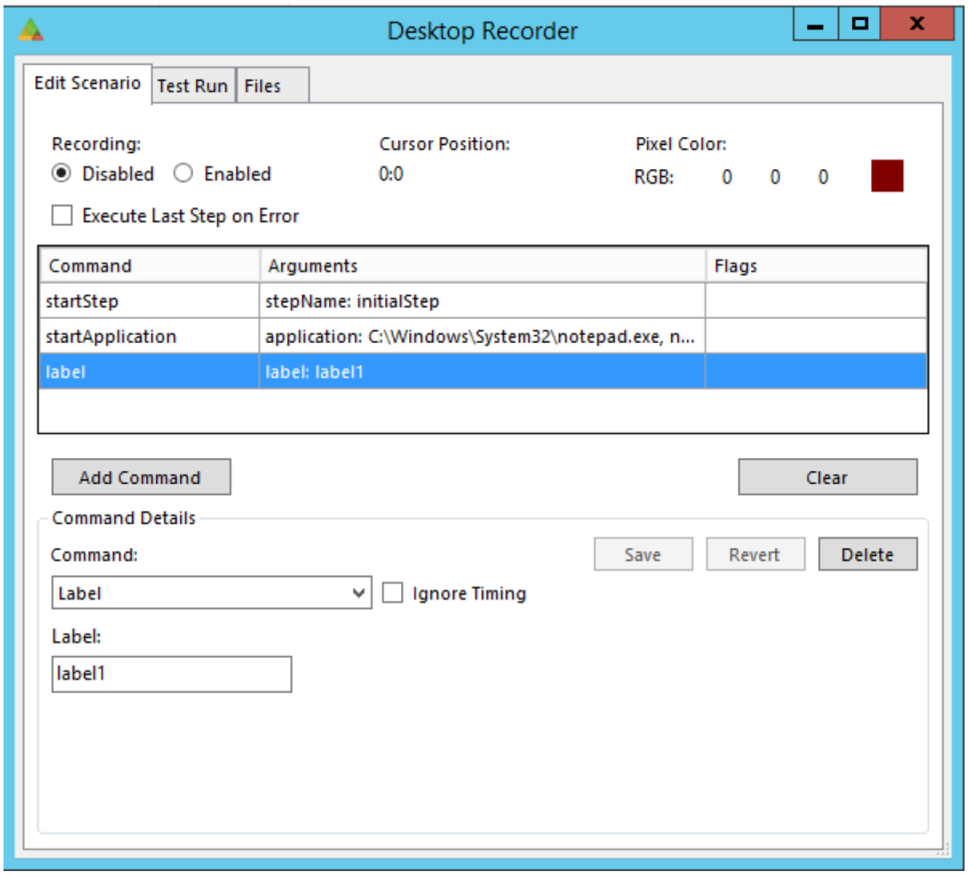
Or

```
${demo_button}==true
```

You cannot change the value of a recorder variable from inside a JavaScript, any change made to the value of a recorder var inside a script will be lost when the script is done executing. Instead, use *storeEval* to store the output of a script in a new (or overwrite an existing) recorder variable

11 Flow control

The label command is used to define labels or points in the scenario.



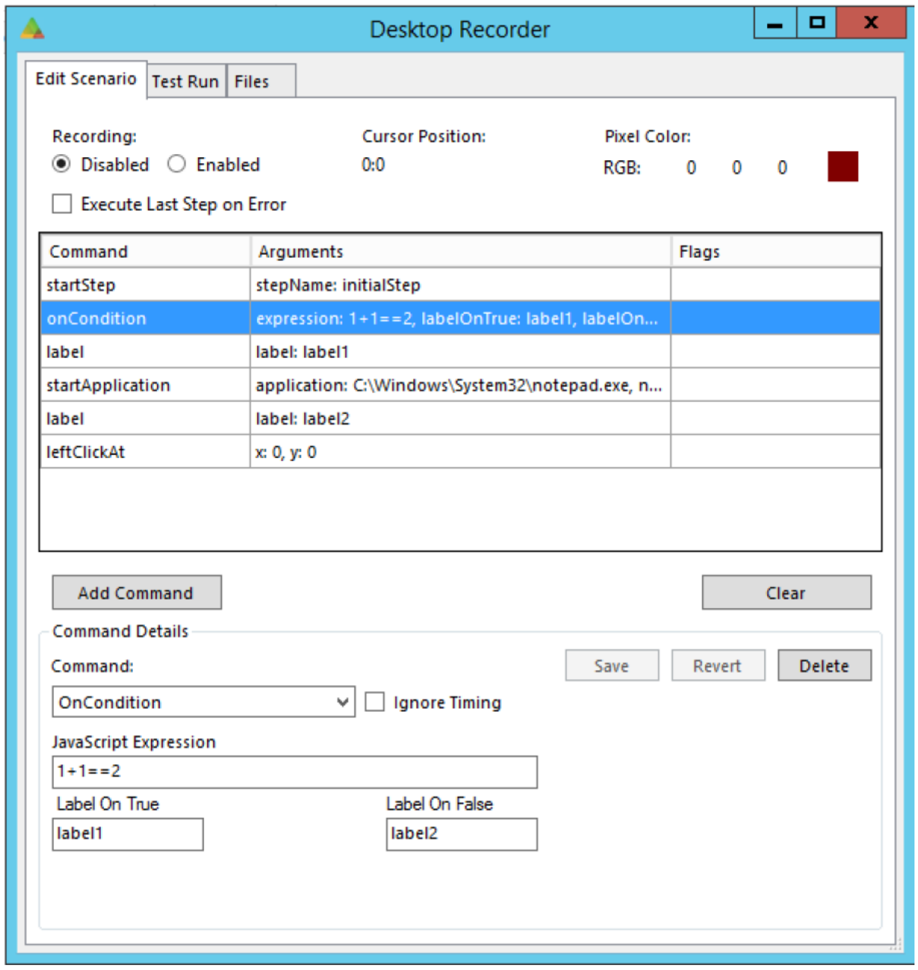
Recording: [✓ Disabled] Enabled
Cursor Position: 0:0
Pixel Color: RGB: 0 0 0 🟥
Execute Last Step on Error

Command	Arguments	Flags
startStep	stepName: initialStep	
startApplication	application: C:\Windows\System32\notepad.exe, n...	
label	label: label1	

Command Details
Command: Label
Ignore Timing
Label: label1

Note: You cannot use variables when defining labels.

You can then jump to one of these points, based on the result of evaluating a JavaScript snippet, using the *onCondition* command, or by jumping to a specific label using the *goto* command.



Recording: (● Disabled) ○ Enabled

Cursor Position: 0:0

Pixel Color: RGB: 0 0 0 🟥

☐ Execute Last Step on Error

Command	Arguments	Flags
startStep	stepName: initialStep	
onCondition	expression: 1+1==2, labelOnTrue: label1, labelOn...	
label	label: label1	
startApplication	application: C:\Windows\System32\notepad.exe, n...	
label	label: label2	
leftClickAt	x: 0, y: 0	

Command Details

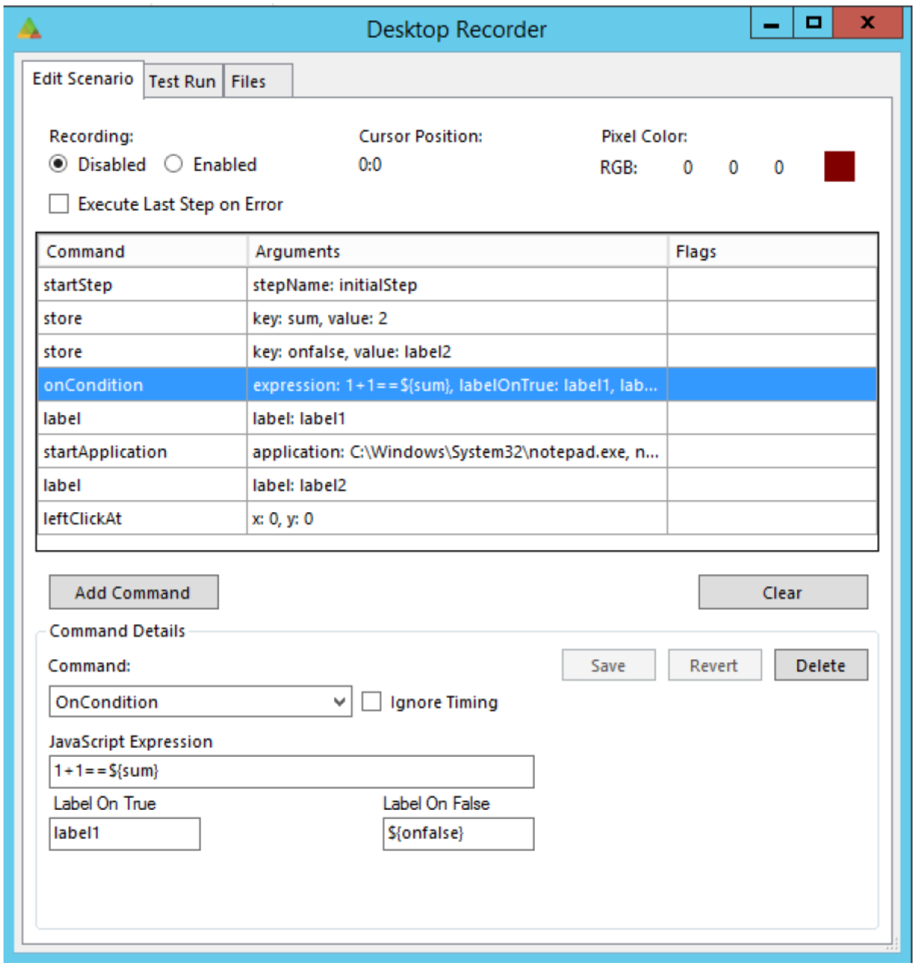
Command: OnCondition ☐ Ignore Timing

JavaScript Expression: 1+1==2

Label On True: label1

Label On False: label2

Note that you can use variables in both expression and labelOnTrue/labelOnFalse *arguments. Also, both arguments are actually optional, if either label is omitted the result will be that the scenario continues as usual.*



Recording: [✓ Disabled] Enabled

Cursor Position: 0:0

Pixel Color: RGB: 0 0 0 🟥

Execute Last Step on Error

Command	Arguments	Flags
startStep	stepName: initialStep	
store	key: sum, value: 2	
store	key: onfalse, value: label2	
onCondition	expression: 1+1==${sum}, labelOnTrue: label1, lab...	
label	label: label1	
startApplication	application: C:\Windows\System32\notepad.exe, n...	
label	label: label2	
leftClickAt	x: 0, y: 0	

Command Details

Command: OnCondition

Ignore Timing

JavaScript Expression: 1+1==${sum}

Label On True: label1

Label On False: ${onfalse}

12 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 '*'.

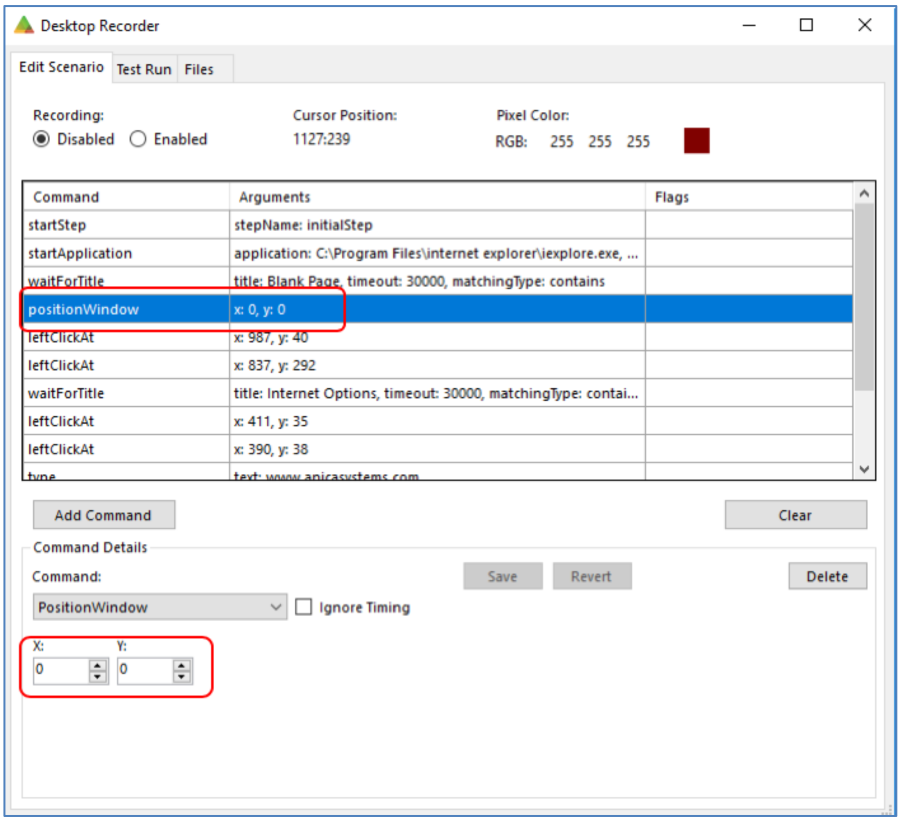
The matching type - glob works as follows:

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

13 Window Management

13.1 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.



Desktop Recorder

Edit Scenario | Test Run | Files

Recording: [✓ Disabled] Enabled

Cursor Position: 1127:239

Pixel Color: RGB: 255 255 255 🟥

Command	Arguments	Flags
startStep	stepName: initialStep	
startApplication	application: C:\Program Files\internet explorer\iexplore.exe, ...	
waitForTitle	title: Blank Page, timeout: 30000, matchingType: contains	
positionWindow	x: 0, y: 0	
leftClickAt	x: 987, y: 40	
leftClickAt	x: 837, y: 292	
waitForTitle	title: Internet Options, timeout: 30000, matchingType: contai...	
leftClickAt	x: 411, y: 35	
leftClickAt	x: 390, y: 38	
type	text: www.apicasystems.com	

Command Details

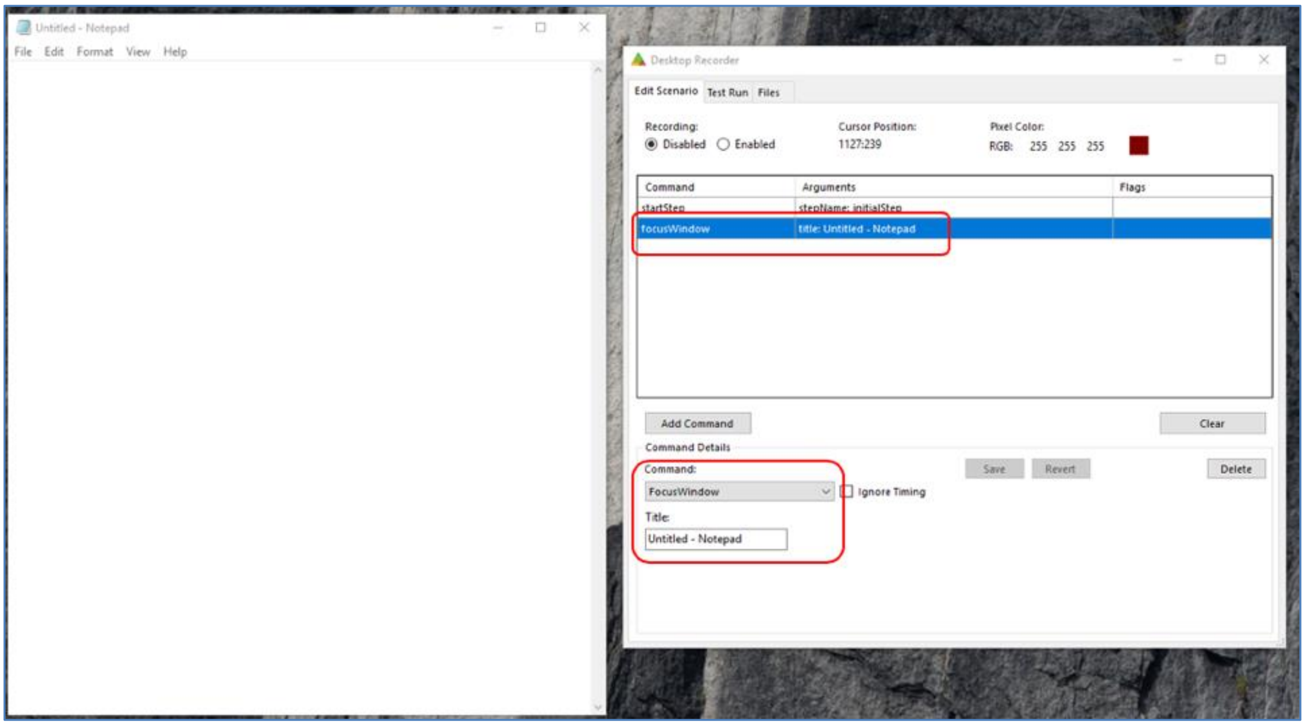
Command: PositionWindow

Ignore Timing: ☐

X: 0

Y: 0

13.2 Focus Window



Recording: (● Disabled) ○ Enabled
Cursor Position: 1127:239
Pixel Color: RGB: 255 255 255 🟥

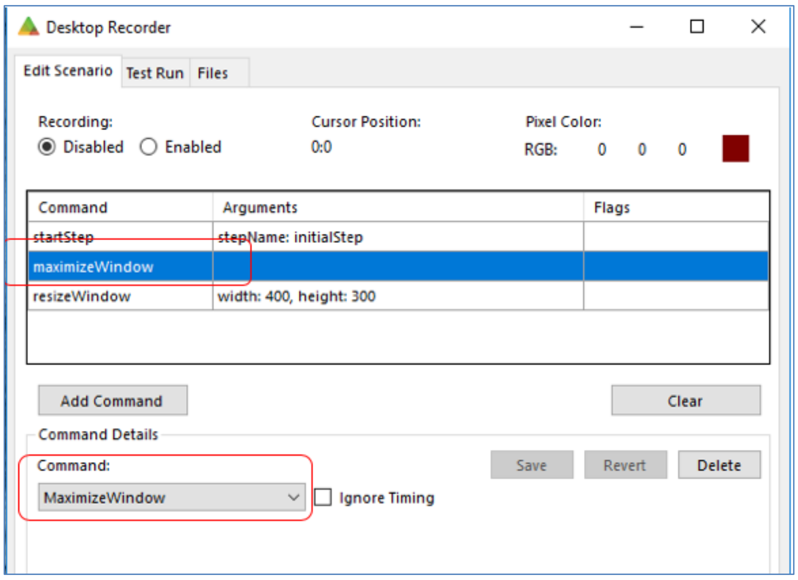
Command	Arguments	Flags
startStep	stepName: initialStep	
focusWindow	title: Untitled - Notepad	

Command Details
Command: FocusWindow ☐ Ignore Timing
Title: Untitled - Notepad

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

13.3 Maximize Window

Use *maximizeWindow* to maximize the currently focused application window.



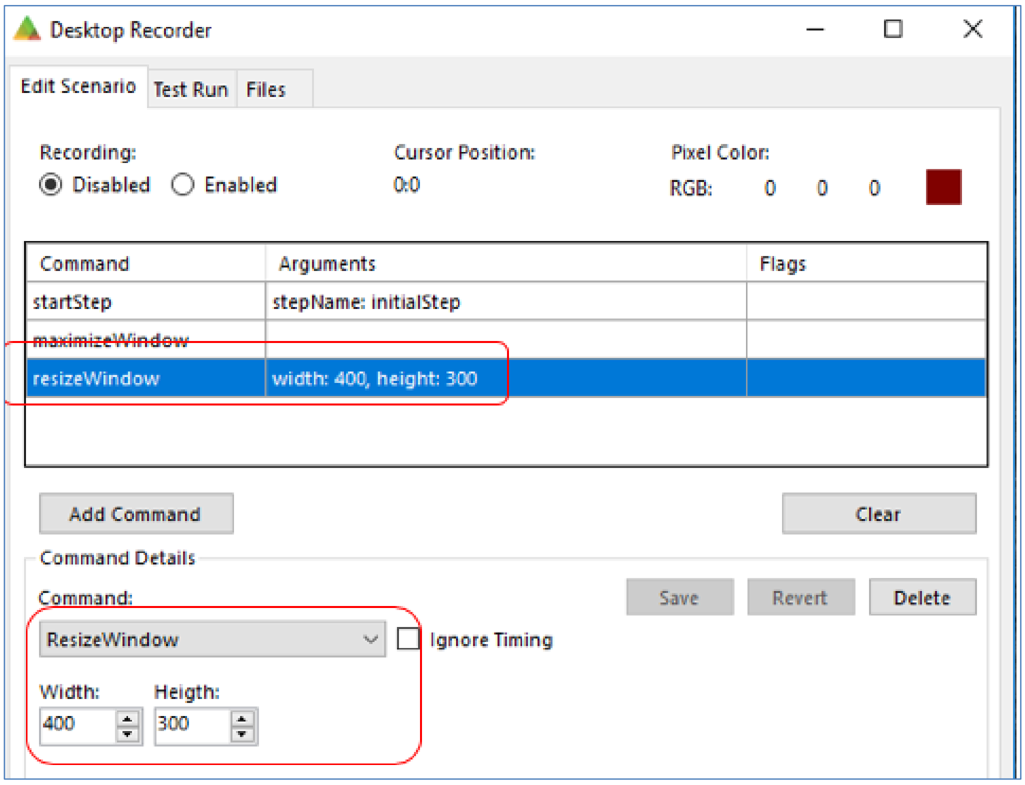
Recording: (● Disabled) ○ Enabled
Cursor Position: 0:0
Pixel Color: RGB: 0 0 0 🟥

Command	Arguments	Flags
startStep	stepName: initialStep	
maximizeWindow		
resizeWindow	width: 400, height: 300	

Command Details
Command: MaximizeWindow ☐ Ignore Timing

13.4 Resize Window

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



Recording: [✓ Disabled] Enabled

Cursor Position: 0:0

Pixel Color: RGB: 0 0 0

Command	Arguments	Flags
startStep	stepName: initialStep	
~~maximizeWindow~~		
resizeWindow	width: 400, height: 300	

Command Details

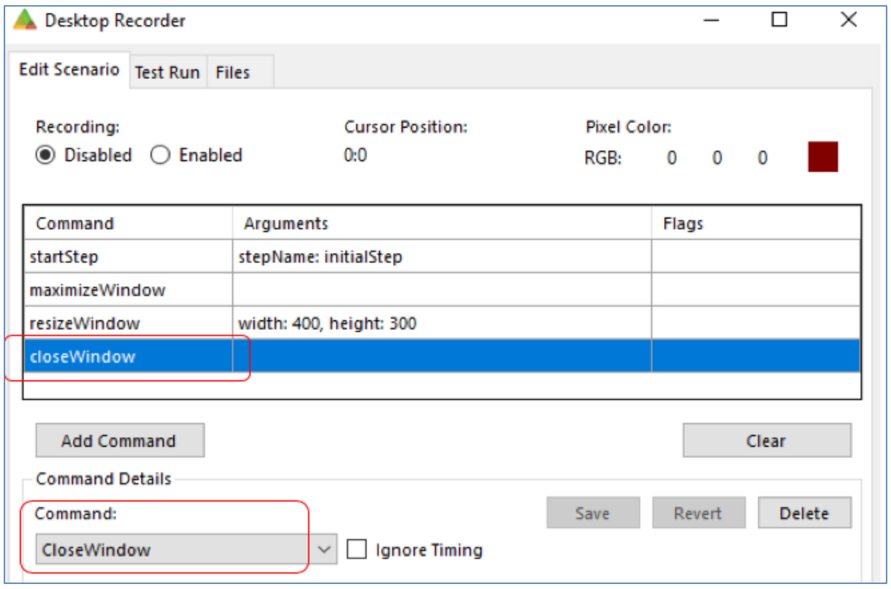
Command: ResizeWindow

Ignore Timing

Width: 400 Heigth: 300

13.5 Close Window

Use *closeWindow* to close the currently focused application window.



Recording: [✓ Disabled] Enabled

Cursor Position: 0:0

Pixel Color: RGB: 0 0 0

Command	Arguments	Flags
startStep	stepName: initialStep	
maximizeWindow		
resizeWindow	width: 400, height: 300	
closeWindow		

Command Details

Command: CloseWindow

Ignore Timing

14 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.

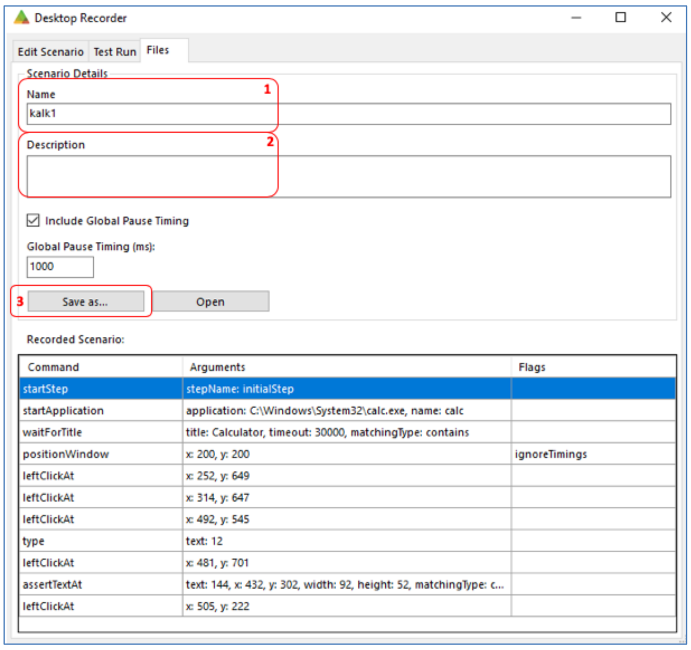
15 Open/Save Scenarios

15.1 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.



Name: kalk1

Description:

☑ Include Global Pause Timing

Global Pause Timing (ms): 1000

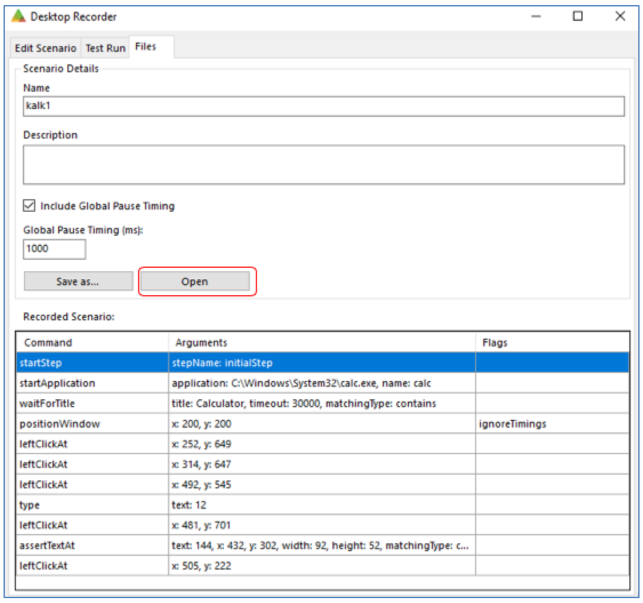
Recorded Scenario:

Command	Arguments	Flags
startStep	stepName: initialStep	
startApplication	application: C:\Windows\System32\calc.exe, name: calc	
waitForTitle	title: Calculator, timeout: 30000, matchingType: contains	
positionWindow	x: 200, y: 200	ignoreTimings
leftClickAt	x: 252, y: 649	
leftClickAt	x: 314, y: 647	
leftClickAt	x: 492, y: 545	
type	text: 12	
leftClickAt	x: 481, y: 701	
assertTextAt	text: 144, x: 432, y: 302, width: 92, height: 52, matchingType: c...	
leftClickAt	x: 505, y: 222	

15.2 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.



Name: kalk1

Description:

☑ Include Global Pause Timing

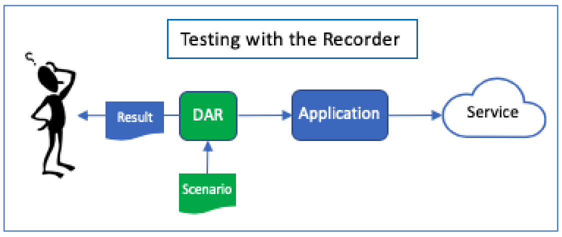
Global Pause Timing (ms): 1000

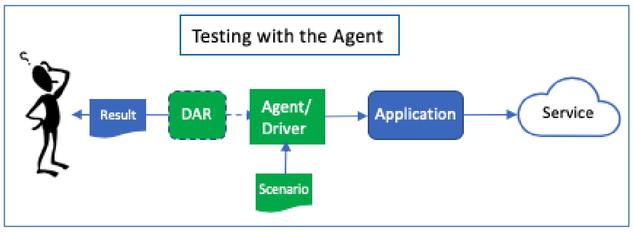
Recorded Scenario:

Command	Arguments	Flags
startStep	stepName: initialStep	
startApplication	application: C:\Windows\System32\calc.exe, name: calc	
waitForTitle	title: Calculator, timeout: 30000, matchingType: contains	
positionWindow	x: 200, y: 200	ignoreTimings
leftClickAt	x: 252, y: 649	
leftClickAt	x: 314, y: 647	
leftClickAt	x: 492, y: 545	
type	text: 12	
leftClickAt	x: 481, y: 701	
assertTextAt	text: 144, x: 432, y: 302, width: 92, height: 52, matchingType: c...	
leftClickAt	x: 505, y: 222	

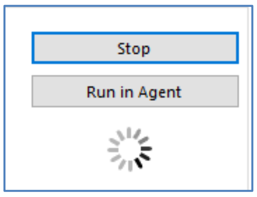
16 Test Run a Scenario

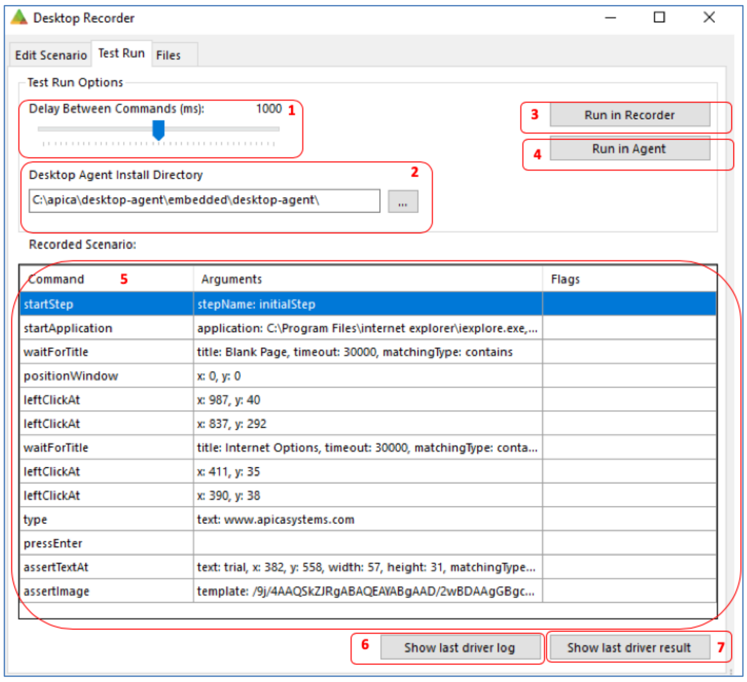
You can test either with the local Desktop Recorder or you can run it on the Agent.





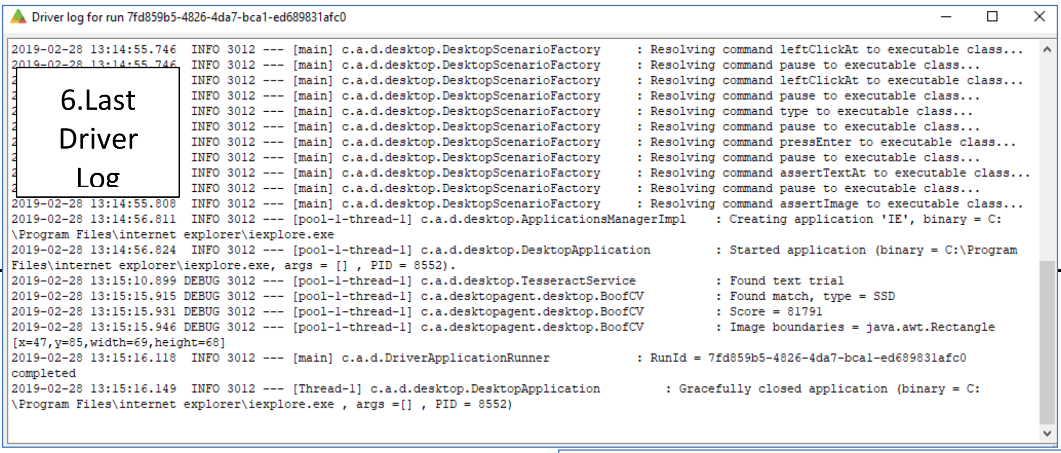
1. To test your scenario, Select the "Test Run" tab.
2. Click "Run in Recorder" (3) or Run in Agent” (4) to start the test.
3. When running in recorder, use the slider labeled "Delay Between Commands (ms)" (1) to set a delay, in milliseconds, between each command.
4. When “Run in Agent” is used, the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



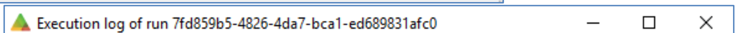


C:\apica\desktop-agent\embedded\desktop-agent\

Command	Arguments	Flags
startStep	stepName: initialStep	
startApplication	application: C:\Program Files\internet explorer\iexplore.exe,...	
waitForTitle	title: Blank Page, timeout: 30000, matchingType: contains	
positionWindow	x: 0, y: 0	
leftClickAt	x: 987, y: 40	
leftClickAt	x: 837, y: 292	
waitForTitle	title: Internet Options, timeout: 30000, matchingType: conta...	
leftClickAt	x: 411, y: 35	
leftClickAt	x: 390, y: 38	
type	text: www.apicasystems.com	
pressEnter		
assertTextAt	text: trial, x: 382, y: 558, width: 57, height: 31, matchingType...	
assertImage	template: /9j/4AAQSkZJRgABAQEAYABgAAD/2wBDAAgGBgc...	



```
2019-02-28 13:14:55.746  INFO 3012 --- [main] c.a.d.desktop.DesktopScenarioFactory     : Resolving command leftClickAt to executable class...
2019-02-28 13:14:55.746  INFO 3012 --- [main] c.a.d.desktop.DesktopScenarioFactory     : Resolving command pause to executable class...
                         INFO 3012 --- [main] c.a.d.desktop.DesktopScenarioFactory     : Resolving command leftClickAt to executable class...
                         INFO 3012 --- [main] c.a.d.desktop.DesktopScenarioFactory     : Resolving command pause to executable class...
                         INFO 3012 --- [main] c.a.d.desktop.DesktopScenarioFactory     : Resolving command type to executable class...
                         INFO 3012 --- [main] c.a.d.desktop.DesktopScenarioFactory     : Resolving command pause to executable class...
                         INFO 3012 --- [main] c.a.d.desktop.DesktopScenarioFactory     : Resolving command pressEnter to executable class...
                         INFO 3012 --- [main] c.a.d.desktop.DesktopScenarioFactory     : Resolving command pause to executable class...
                         INFO 3012 --- [main] c.a.d.desktop.DesktopScenarioFactory     : Resolving command assertTextAt to executable class...
                         INFO 3012 --- [main] c.a.d.desktop.DesktopScenarioFactory     : Resolving command pause to executable class...
2019-02-28 13:14:55.808  INFO 3012 --- [main] c.a.d.desktop.DesktopScenarioFactory     : Resolving command assertImage to executable class...
2019-02-28 13:14:56.811  INFO 3012 --- [pool-1-thread-1] c.a.d.desktop.ApplicationsManagerImpl    : Creating application 'IE', binary = C:
\Program Files\internet explorer\iexplore.exe
2019-02-28 13:14:56.824  INFO 3012 --- [pool-1-thread-1] c.a.d.desktop.DesktopApplication         : Started application (binary = C:\Program
Files\internet explorer\iexplore.exe, args = [] , PID = 8552).
2019-02-28 13:15:10.899 DEBUG 3012 --- [pool-1-thread-1] c.a.d.desktop.TesseractService           : Found text trial
2019-02-28 13:15:15.915 DEBUG 3012 --- [pool-1-thread-1] c.a.desktopagent.desktop.BoofCV          : Found match, type = SSD
2019-02-28 13:15:15.931 DEBUG 3012 --- [pool-1-thread-1] c.a.desktopagent.desktop.BoofCV          : Score = 81791
2019-02-28 13:15:15.946 DEBUG 3012 --- [pool-1-thread-1] c.a.desktopagent.desktop.BoofCV          : Image boundaries = java.awt.Rectangle
[x=47,y=85,width=69,height=68]
2019-02-28 13:15:16.118  INFO 3012 --- [main] c.a.d.DriverApplicationRunner            : RunId = 7fd859b5-4826-4da7-bca1-ed689831afc0
completed
2019-02-28 13:15:16.149  INFO 3012 --- [Thread-1] c.a.d.desktop.DesktopApplication        : Gracefully closed application (binary = C:
\Program Files\internet explorer\iexplore.exe , args =[] , PID = 8552)
```



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.

17 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.

18 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.

19 Appendix - Testing a scenario through the API

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.

19.1 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.

19.2 Get JobStatus

```
(get) http://<host>:8080/job/<jobId>

Expected response

{

    "id": "<jobId>",

    "status": "finished"  // or running, failed etc.

}
```

19.3 Get a result

```
(get) http://<host>:8080/result/<jobId>

Header

Accept:application/xml
```

Expected response

```
{A result in XML format}
```