

USER GUIDE 5.7

Contents

Navigate to a URL	9
Switch to another tab/window	9
Switch to alert/prompt	10
Set screen/window size - width * height	11
Click on text	11
Click on radio/checkbox	11
Click on Attribute value	12
Click based on Container	12
Click on repeated elements	12
Select/Choose	13
Verify text is on the page	14
Verify if an element is disabled/enabled/ (not) visible	14
Verify URL	15
Verify New Windows or Alerts	15
Verify text using xpath	15
Verify attribute of an xpath	15
Verify if a saved variable matches with some text	16
Verify CSS properties with xpath	16
Verify if a saved variable text is present on the page	17
Verify text between 2 variables	17
Expected Results (Verify Statements) Column to Force actions	18
Wait for given time	27
Get Todays date in a given format	27
Relative Date support	28
Creating variable	28
Referencing previously created variable	29
Saving a variable from label on page or xpath	29
Local variable	29
Enter/Click when the element is stale	29
Instruction to Verify variable is checked/unchecked	29
Optional Arguments	30
Provide spinner/progress bar information	32

No scroll option for a test case and project	32
Get element details	34
Full page screenshot for a project	35
Maintenance	35
Schedule in Edge Browsers	36
Serial Execution	37
Merge TestCase	38
Debug points	41
Mobile Browser Testing	43
Click on image icon with image uploaded as artifact	43
Multi Line Step Editor	44
Copy paste to and from excel	45
Post Execution Process for a Test Suite	46
Download and Upload	47
Automatic Disk Space Free-Up	48
NLP Scroll Changes	50
Steps to show screenshots during “AIQ Execution Screenshots OFF” mode	51
Remote execution and script generation	51
Zalenium/headful mode execution/generation	52
Use custom code from file	53
Actions with Proximity [Next to/before/after]	54
Script Execution	54
JavaScript	55
Set value for a Text box	55
Verify if a checkbox is checked	55
Verify if a checkbox is checked if in iframe	55
Verify number of elements	56
Get a values attribute	56
Find sum of numbers in a string	56
JavaScript for variable comparision	56
PythonScript for variable comparision	56
Python floating point comparision	56
Luhn Algorithm or Modulus 10 Algoritithm	57
Dropdown Validation	57
Find the difference between 2 dates	58
Get System Time	58

Table Instructions	59
Clicks	59
Enter text	59
Radio Button Selection	59
Checkbox Selection	59
DropDown Selection	60
Click/ Enter text based on row number	60
Get the table row count	60
Excel Functions	60
Summing up numbers	60
Multiplication	61
Division	61
Round	61
Greater than	61
Dates	61
Days between Dates	61
Currency Symbols as Prefix	61
Other Misc	62

NLP Commands - List

1. `_var{[excel formula]}` as `var_name`
2. `_xl{[excel formula]}` as `var_name`
3. Assert “text” is visible on the page
4. Choose “text”
5. Choose `{xpath: "address"}`
6. Click `_css("selector")`
7. Click `{xpath: "address"}`
8. Click on “text” radio
9. Begin block `block_name`
10. Begin script `_bash` with `${input_var}`
11. Begin script `_js` with `${input_var}`
12. Begin script `_py` with `${input_var}`
13. `break`
14. Choose `_css("selector")`
15. Click on “text” radio for “text”
16. Click on “exact attribute value”
17. Click on “text” after “text”
18. Click on “text” for “text”
19. Click on “text” before “text”
20. Click on `_xy{50,50}` of `_xpath{“//img”}`
21. Click on `_xy{20,30}` of `_css{#some_html_node_id}`
22. Create random variable `var_name`
23. Capture screenshot
24. Double Click on `_xy{20, 30}` of “Photo of Eiffel Tower”
25. Download file by clicking on “text”
26. End block
27. End script save as `${output_var_name}`
28. Enter “textbox”
29. Enter text in `_css("selector")`
30. Enter text in “text”
31. Enter text in `{xpath: "address"}`
32. Exec `_js{script}` with `${input_var}` returning `${output_var}`
33. Exec `_py{script}` with `${input_var}` returning `${output_var}`
34. Exec `_sh{script}` with `${input_var}` returning `${output_var}`
35. Exec "artifact file" with `${input_var}` returning `${output_var}`
36. Fill in “textbox”
37. Fill in text in `_css("selector")`
38. Fill in text in “text”
39. Fill in text in `{xpath: "address"}`
40. Go to site
41. Go to web page
42. Go to page

43. Go to
44. Go to website
45. enter "textbox" and save as xyz
46. Hit enter
47. Hit Down arrow key
48. hit up arrow key
49. Hit spacebar
50. Hit alt
51. Hit escape
52. Hit tab
53. Hover on `_css("selector")`
54. Hover on "text"
55. Hover on `{xpath: "address"}`
56. Hover over `_css("selector")`
57. Hover on `_xy{20,30}` of `_xpath{//*[id='some_html_node_id']}`
58. Navigate website
59. Navigate to website
60. Navigate to site
61. Navigate to web page
62. Navigate to page
63. Navigate site
64. Navigate web page
65. Navigate page
66. Navigate to
67. Navigate
68. Open website
69. Open site
70. Open web page
71. Open page
72. Press `_css("selector")`
73. Press `{xpath: "address"}`
74. Press on "text" radio
75. Press on "text" radio for "text"
76. Run `$(flow name)` for [number] times
77. Run `$(flow name)` for all rows
78. Save `_css("selector")` as `var_name`
79. Save `{xpath: "address"}` as `var_name`
80. Scroll up
81. Scroll down
82. Select `_css("selector")`
83. Select "text"
84. Select `{xpath: "address"}`
85. Set screen/window size - width * height
86. Set text in `_css("selector")`

87. Set text in "text"
88. Set text in {xpath: "address"}
89. Set screen to hd
90. Set screen to mobile phone
91. Set screen to tablet
92. Set screen to tablet landscape
93. Set screen to 1080
94. Set screen to 1080p
95. Set screen to 720
96. Set screen to 900
97. Set screen to Full HD
98. Set screen size - 200 * 200
99. Switch to 2nd tab
100. Switch to 2nd window
101. Switch to 3rd tab
102. Switch to 3rd window
103. Switch to alert and click on accept
104. Switch to alert and click on cancel
105. Switch to alert and click on leave
106. Switch to alert and click on ok
107. Switch to alert and click on stay
108. Switch to alert and save message as alert_set1 and click ok
109. Switch to confirm and click on accept
110. Switch to confirm and click on cancel
111. Switch to confirm and click on leave
112. Switch to confirm and click on ok
113. Switch to confirm and click on stay
114. Switch to new tab
115. Switch to new window
116. Switch to original tab
117. Switch to original window
118. Switch to prompt and click on accept
119. Switch to prompt and click on cancel
120. Switch to prompt and click on leave
121. Switch to prompt and click on ok
122. Switch to prompt and click on stay
123. Switch to window with title "title1"
124. Type in "textbox"
125. Type in text in _css("selector")
126. Type in text in "text"
127. Type in text in {xpath: "address"}
128. Take screenshot
129. Upload file to _css("selector")
130. Upload file to "text"

131. Upload file to {xpath: "address"}
132. Use custom code from [location]
133. Verify _css("selector") begins with "text" or begins with "text"
134. Verify _css("selector") begins with "text" or ends with "text"
135. Verify _css("selector") begins with "text"
136. Verify _css("selector") contains "text" or begins with "text"
137. Verify _css("selector") contains "text" or contains "text"
138. Verify _css("selector") contains "text" or ends with "text"
139. Verify _css("selector") contains "text"
140. Verify _css("selector") ends with "text" or ends with "text"
141. Verify _css("selector") ends with "text"
142. Verify _css("selector") is _css("selector")
143. Verify _css("selector") is _css("selector")
144. Verify _css("selector") is disabled
145. Verify _css("selector") is enabled
146. Verify _css("selector") is not visible
147. Verify _css("selector") is visible
148. Verify "text" is on the page
149. Verify {xpath: "address"} begins with "text" or begins with "text"
150. Verify {xpath: "address"} begins with "text" or ends with "text"
151. Verify {xpath: "address"} begins with "text"
152. Verify {xpath: "address"} contains "text" or begins with "text"
153. Verify {xpath: "address"} contains "text" or contains "text"
154. Verify {xpath: "address"} contains "text" or ends with "text"
155. Verify {xpath: "address"} contains "text"
156. Verify {xpath: "address"} ends with "text" or ends with "text"
157. Verify {xpath: "address"} ends with "text"
158. Verify {xpath: "address"} is {xpath: "address"}
159. Verify {xpath:"address"} background-color is #ffffff
160. Verify {xpath:"address"} color is #e01719
161. Verify {xpath:"address"} font-size 26px
162. Verify {xpath: "address"} is disabled
163. Verify {xpath: "address"} is enabled
164. Verify {xpath: "address"} is not visible
165. Verify {xpath: "address"} is visible
166. Verify alert exists
167. Verify pop up exists
168. Verify new tab exists
169. Verify the current url is "url"
170. Verify url is "url"
171. Verify variable \${var_name} is "text"
172. Verify \${var_name} is on the screen
173. Verify \${var_name} on the page
174. Verify new window exists

- 175. Wait [number] secs
- 176. Wait for [number] seconds
- 177. Wait for [number] secs

NLP Commands Detailed

Note: Whenever any action performed on a text doesn't work without quotes. Try once more with quotes.

Window/Tab/Alert

Navigate to a URL

Navigates to the url. This should always be the first step in any testcase, since each testcase is independent in Autonomiq. In addition you can also perform this during the course of a test step when you want to navigate to another page.

Open/Launch/Go to/navigate website **url**

open website | <https://www.wikipedia.org/>

Go to website | <https://twitter.com>

Launch website | <https://google.com>

Navigate website | <https://jsfiddle.com>

Test Steps	Data
Open site	http://google.com
Open web page	http://google.com
Open page	http://google.com
Navigate to web site	http://google.com
Navigate to site	http://google.com
Navigate to web page	http://google.com
Navigate to page	http://google.com
Navigate web site	http://google.com
Navigate site	http://google.com
Navigate web page	http://google.com
Navigate page	http://google.com
Navigate to	http://google.com
Navigate	http://google.com
Go to web site	http://google.com
Go to site	http://google.com
Go to web page	http://google.com
Go to page	http://google.com
Go to	http://google.com
Goto web site	http://google.com
Goto website	http://google.com
Goto site	http://google.com
Goto web page	http://google.com
Goto page	http://google.com
Goto	http://google.com

The url can be given from the data tab.

Switch to another tab/window

This command will move focus on to other pop-up windows if new window/tab comes-up

Switch to first/second/1/1st/2/2nd/3/3rd/original/new tab/window

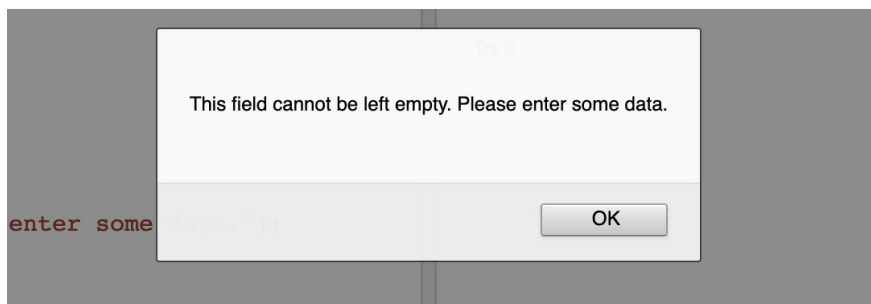
For a new window or tab

- **switch to new window**
- **switch to new tab**

This will get the focus back to the original window if a new window comes down
switch to original window

This will move focus to required tab/window if multiple window comes-up
switch to 3rd tab

Switch to alert/prompt



An alert/prompt is different from a normal window. Usual switch to window/tab will not work in this case. Use the below options as required.

Switch to alert/prompt/confirm and click on ok/accept/leave/cancel/stay

Based on what is displayed in your prompt you can choose one of the following commands

- **Switch to alert and click on accept**
- **Switch to alert and click on cancel**
- **Switch to alert and click on leave**
- **Switch to alert and click on ok**
- **Switch to alert and click on stay**
- **Switch to confirm and click on accept**
- **Switch to confirm and click on cancel**
- **Switch to confirm and click on leave**
- **Switch to confirm and click on ok**
- **Switch to confirm and click on stay**

Whenever an alert window is encountered, a variable will be created with variables as 'alert_message'. This variable can then be used to verify for specific message if needed.

Set screen/window size - width * height

If in case your window size is not coming up as required during automation, you can give the setting at the starting of the testcase after open window.

- **set screen size - 500 * 500**
- **set screen size - 500*500**
- **set window size - 500 * 500**
- **set window size - 500 * 500**

Click - Select - Choose

Click on text

Click/Press on/at "**element text**"

- **click on "login"**
- **click at "login"**
- **press on "login"**
- **press at "login"**

Click on radio/checkbox

Click/Press on/at "**text**" radio/checkbox

When you want to click a certain radio button or checkbox.

- **click on "yes" radio**

Click/Press on/at "**text**" radio/checkbox for "**element text**"

When you want to click a radio button or checkbox next to something. The following can be used.

- **click on "yes" radio for "are you over 18?"**

Click on Attribute value

Click also works on unique attribute values.

Click on “**exact attribute value**”

For example,

```
mouseleave:MWfikb;YMFC3:VKssTb">...</div>
▼<div class="FPdoLc VlLAE">
  ▼<center>
    <input value="Google Search" aria-label="Google Search" name="btnK" type="submit">
    <input value="I'm Feeling Lucky" aria-label="I'm Feeling Lucky" name="btnI" type="submit" jsaction="sf.lck">
  </center>
```

- **Click on “btnk”**

Note: This is case insensitive

Click based on Container

Click on “**identifier1**” for “**identifier2**”

Can be used in the following scenarios:

1. Where there are multiple repetitions of the same button, but there are unique identifiers for each which is present in the container surrounding it.

ex: Click on “READ MORE” for “Skate Victoria”

2. When there are multiple calendar pages (similar situation as above ex: **Click on “11” for “April 2019”**)

A screenshot of a webpage showing four event listings. Each listing includes a logo, event name, dates, location, and a 'READ MORE' button. The events are: World Team Trophy (11 Apr - 14 Apr, 2019, Fukuoka /JPN), Skate Victoria (09 Apr - 14 Apr, 2019, Sofia /BUL), Triglav Trophy & Narcisa Cup (08 Apr - 14 Apr, 2019, Jesenice /SLO), and ISU Judges Seminars in Pair Skating (18 Apr - 21 Apr, 2019, Berlin /GER).

A screenshot showing two calendar pages. The left page is for April 2019, and the right page is for May 2019. The date '11' in the April calendar is highlighted with a red box.

April 2019							May 2019						
Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat
-	-	1	2	3	4	5	6	-	-	1	2	3	4
7	8	9	10	11	12	13	5	6	7	8	9	10	11
14	15	16	17	18	19	20	12	13	14	15	16	17	18
21	22	23	24	25	26	27	19	20	21	22	23	24	25
28	29	30	-	-	-	-	26	27	28	29	30	31	-

Click on repeated elements

If repeated text is there, you can give the click based on proximity. [Actions based on proximity]

click at (x,y)

We can now give XY coordinate on any HTML node and ask system to click at that position. It uses the syntax `_xy{ }`

Example:



Below are the sample instructions that are supported:

- Click on `_xy{20, 30}` of `_css{#some_html_node_id}`
- Hover on `_xy{20, 30}` of `_xpath{//img[@id='some_html_node_id']}`
- Double Click on `_xy{20, 30}` of "Photo of Eiffel Tower"

Note: "Photo of Eiffel tower" is the label element for that image

Select/Choose

Select/choose "title of the dropdown"

In below cases, data to be selected should be sent as test data

- **select "country"**
- **choose "country"**

Select/choose "element text from list" in "title of the dropdown"

Here Passed is the value in the dropdown with Status as the title of the dropdown

- **select "Passed" in "Status"**

Enter

Enter/Type in/fill in/set text in “field”

In below cases, data to be entered should be sent as test data

- **enter “username”** **hanSong**
- **type in “username”** **hanSong**
- **fill in “username”** **hanSong**

Hover

Hover over/on “field”

hover over “username” / hover on “username”

In order to capture the full page screenshot of the application, Autonomiq captures multiple screenshots of the page and stitches them together. During this process, the page gets scrolled up and down to capture screenshots of different sections of the page. This might cause certain types of modals/pop ups (that might be on the page) to automatically close. Using hover will freeze the page from getting scrolled and hence can be used to deal with popups that close when scrolled.

Validations

Using Assert will ensure that the testcase fails and halts at the failed step

Using Verify will ensure that the test step fails but the script generation/execution will not halt. It will proceed to the next step.

Verify text is on the page

This helps to verify if a text is present on the page.

Verify “some text” is on the screen.

- **verify “login” is on the screen**

This step can also be used to dynamically wait until an element is loaded on the screen. Once the element is loaded, the validation gets done. If the element is not loaded within the timeout duration, validation will fail and the error message will state that the element was not found. By default, Autonomiq will wait for 8 seconds before timing out. You can provide a timeout option as shown below

Verify “some text” is on the screen --timeout 30

Verify if an element is disabled/enabled/ (not) visible

This method can be used to verify if the target element is visible/enabled etc

Verify {xpath: "**xpath**"} is **disabled/enabled/visible/not visible**

- **verify {xpath: "//img[@class='gb_Wa']"} is disabled**

Verify URL

The URL of the current window in focus can be verified with this command. This can be used when user navigates to another URL or when user switches to another window and is expected to verify the url,

Verify the current URL is "**url**"

- **verify the current URL is "https://www.test.com/"**

Verify New Windows or Alerts

This command can be used whenever user wants to check if new window appears/Pop-up appears. Based on this, a decision can be made if user should use 'Switch to'

Verify new window/tab/alert/pop up/pop-up exists

- **verify new window exists**
- **verify new alert exists**

Verify text using xpath

Verify {xpath: "**xpath**"} contains/begins with/ends with "**text**"

verify {xpath: "//img[@class='gb_Wa']"} begins with "google"

verify {xpath: "//img[@class='gb_Wa']"} begins with goo or ends with "gle"

Verify attribute of an xpath

This command can be used to verify the attribute of an element using given xpath.

Verify name/placeholder/data-toggle/value/class/type/any-attr of {xpath: "**xpath**"} is "**some value**"

- **verify name of {xpath: "//img[@class='gb_Wa']"} is "test"**

- **Verify {attribute:"xpath"} is "some text"**

✓ 2 **Verify {lang: "//select[@id="searchLanguage"]/option[@selected]} is en**

This can be used to check if checkbox is checked, if any attribute changes based on checkbox change.

Verify if a saved variable matches with some text

This command can be used to verify the value which was saved in some variable.

Verify variable \${variableName} is "someText"

- **verify variable \${var1} is "test"**

Verify CSS properties with xpath

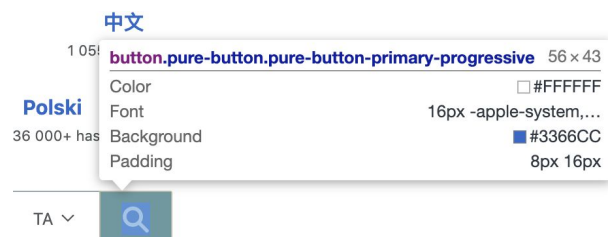
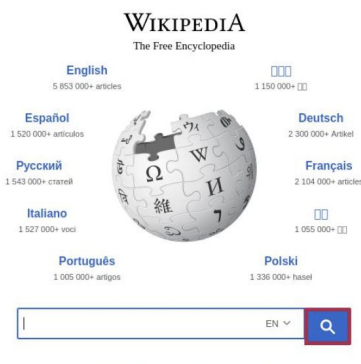
Any CSS attribute of an element can be validated to be expected value using the xpath of the element.

Verify {xpath: "xpath"}

✓ 2 **Verify {xpath: "//button[@type='submit']"} background-color is #3366cc**

width/height/font-family/text-align/font-size/display/color/background-color is "some value"

verify {xpath: : "//img[@class='gb_Wa']"} color is "#e01719"



Note: When validating colours, make sure that the value is in lower case

Verify if a saved variable text is present on the page

- verify variable `#{var1}` is on page
- verify variable `#{var1}` is visible

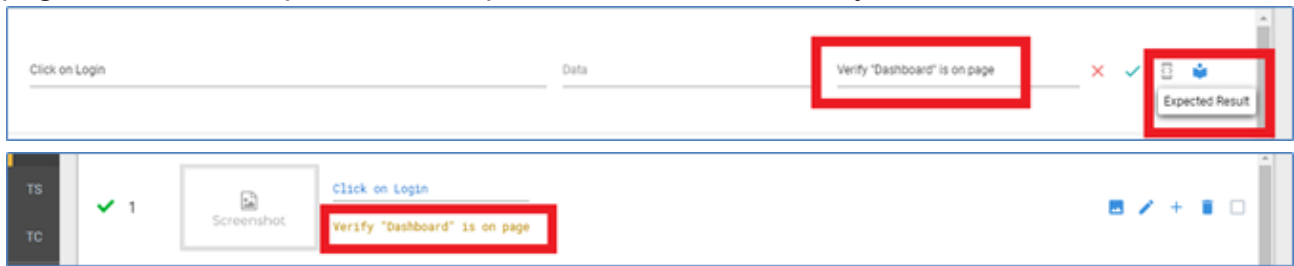
Verify text between 2 variables

Verify variable `#{var_1}` is `#{var_2}`

✓ 2 Verify variable `#{set1}` is `#{set2}`

Expected Results (Verify Statements) Column to Force actions

This column can be used with simple verify commands to check if a particular text is on the page after a click is performed. Expected Results can be only used with Click command.



In the above screenshot the tool will check if the text “Dashboard” is present on the page after the click (Normal Selenium Click). If the verify text condition is not satisfied the tool will now try a series of other clicks and stop if any of the resultant click works. The different clicks are listed below.

- Normal Selenium Click with Timeout
- Action Based Click or MoveAndClick
- Normal JS Click
- XPATH Based JS Click

Only simple Verify commands based on labels can be used in Expected Result. Other complex Verify commands which use URL, Variable, XPATH, CSS etc. are not supported.

Conditional Instructions

2 `_xl{${table_count} < 20}` as condition

3 if variable `${condition}` is "True", Click on English

These instructions are executed if certain conditions are true. Condition is similar to as 'verify'. Use this instead of verify if other instructions are to be performed after it.

if {condition}, some instruction to execute if condition is true

- if {current url is “https://...”}, enter username
- if {xpath:'xpath_'} is visible, Click on Submit
- if “homepage” is on the screen, continue



3

if {current url is "https://www.wikipedia.org/"}, Click on English



Looping Statements

Blocks can be used to loop through commands as many times as required.

begin block <block_name>{instruction1}{instruction2}....end block

```
begin block sample_block
open website
enter username
enter password
click on login
end block
```

run \${block_name} for (number) times

- **run \${sample_block} for 2 times**
run \${block_name} for (number/all) rows
- **run \${sample_block} for all rows**

username	password	first name	middle name	last name	lead source	No. of Employees
krishna@myattest.com	zeta1234	John	Roger	Doe	Web	5555555
krishna@myattest.com	zeta1234	John	Roger	Doe1	Web	5555555
krishna@myattest.com	zeta1234	John	Roger	Doe2	Web	5555555
krishna@myattest.com	zeta1234	John	Roger	Doe3	Web	5555555
krishna@myattest.com	zeta1234	John	Roger	Doe4	Web	5555555
krishna@myattest.com	zeta1234	John	Roger	Doe5	Web	5555555
krishna@myattest.com	zeta1234	John	Roger	Doe6	Web	5555555
krishna@myattest.com	zeta1234	John	Roger	Doe7	Web	5555555

If User wants to run through multiple data for input in each loop, then use separate data file with multiple data. After uploading the data file, associate/link it with the relevant testcase.

In this approach each row of data in the Excel/CSV will correspond to an iteration in the loop

run $\${blockname}$ until $\{condition\}$

The below command is similar to verify

- **run $\${sampleblock}$ until pop-up exists**

Nested Blocks

Nested blocks are blocks within blocks. You can have a single level of nesting, or you can even have multiple levels of nesting blocks

Run $\${block1}$ for all rows
Begin block block1
enter username
enter password
Run $\${block2}$
Begin block block2
enter username
enter password
end block
end block

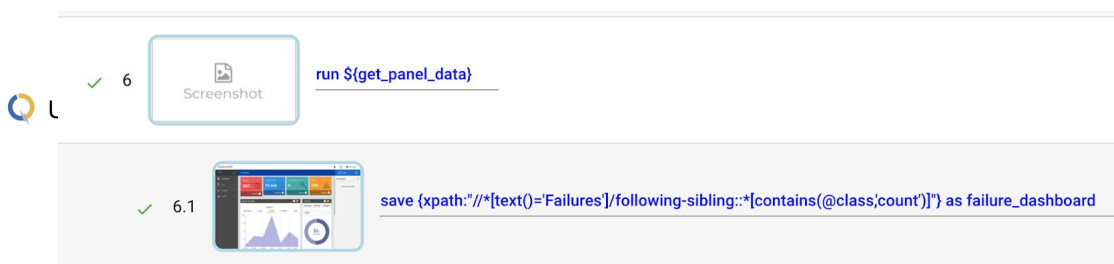
Modular approach - Flows

Flows similar to methods or functions where a user can create a block of code which can be reused across testcases.

run $\${flow_name}$

Once a flow is created, it can be used in a test step in the following way.

During generation for the first time, the steps will expand to display the steps inside the flow.



Nested flows

Flows inside a block is also supported. Here's a sample test case for nested flows

Test Steps	Data
open website	https://login.salesforce.com
run \${main_block} for 3 times	
begin block main_block	
enter username	
run \${loginbasic}	
run \${leadbasic}	
run \${logoutbasic}	
end block	

Decision Making Statement for Blocks/Flows

Note: Decision making statement (i.e) if and the else part will work only for block and flow statement

If statement

if statement is the most simple decision making statement. It is used to decide whether a certain statement or block of statements will be executed or not i.e if a certain condition is true then a block of statements is executed otherwise not.

Syntax

```
if(condition), run ${block}
Begin block blockname
    // Statements to execute if
    // condition is true
End block
```

The condition can be used with flow as below, we can call the flow or can create a block

```
if(condition), run ${flow}
    // Statements to execute if
    // condition is true
```

Example:

```
if {xpath: "//a[@class='page-title-action']"} is visible, run ${Create_User} for all rows
Begin block Create_User
click "Add New"
enter "Username"
enter "First Name"
enter "Last Name"
click on createusersub
end block
```

If the given xpath is visible then users will be created.

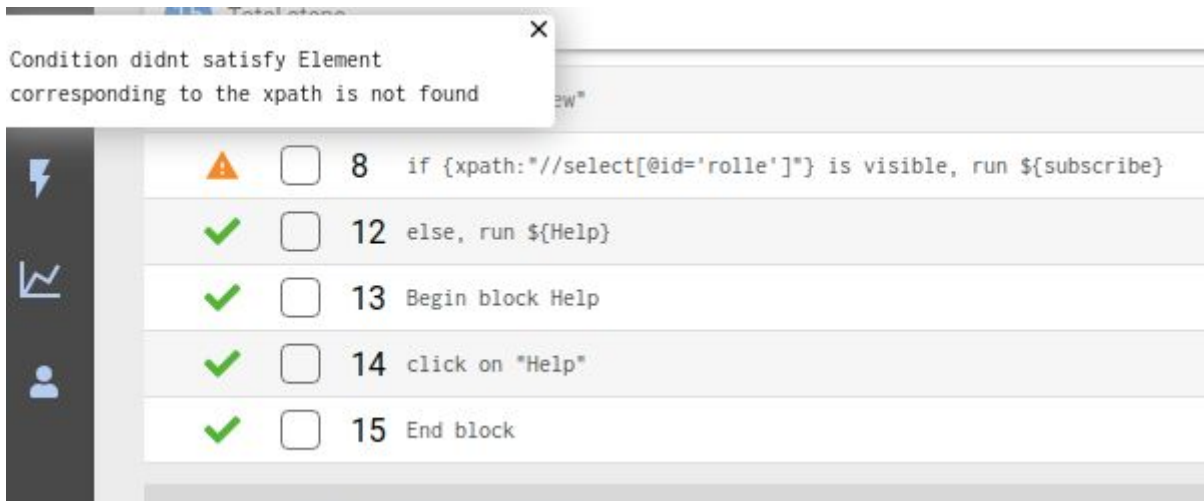
If-else statements

If-else statement, if a condition is true a block of statements will be executed and if the condition is false else part will be executed

Syntax

```
if (condition) run ${block}
Begin block block name
    // Executes this block if
    // condition is true
End Block
Else, run ${else_part}
Begin block else_part
    // Executes this block if
    // condition is false
End Block
```

Example:



In this example the if condition is not satisfied so else part is executed

Nested if statements

When an if else statement is present inside the body of another “if” or “else” then this is called nested if else.

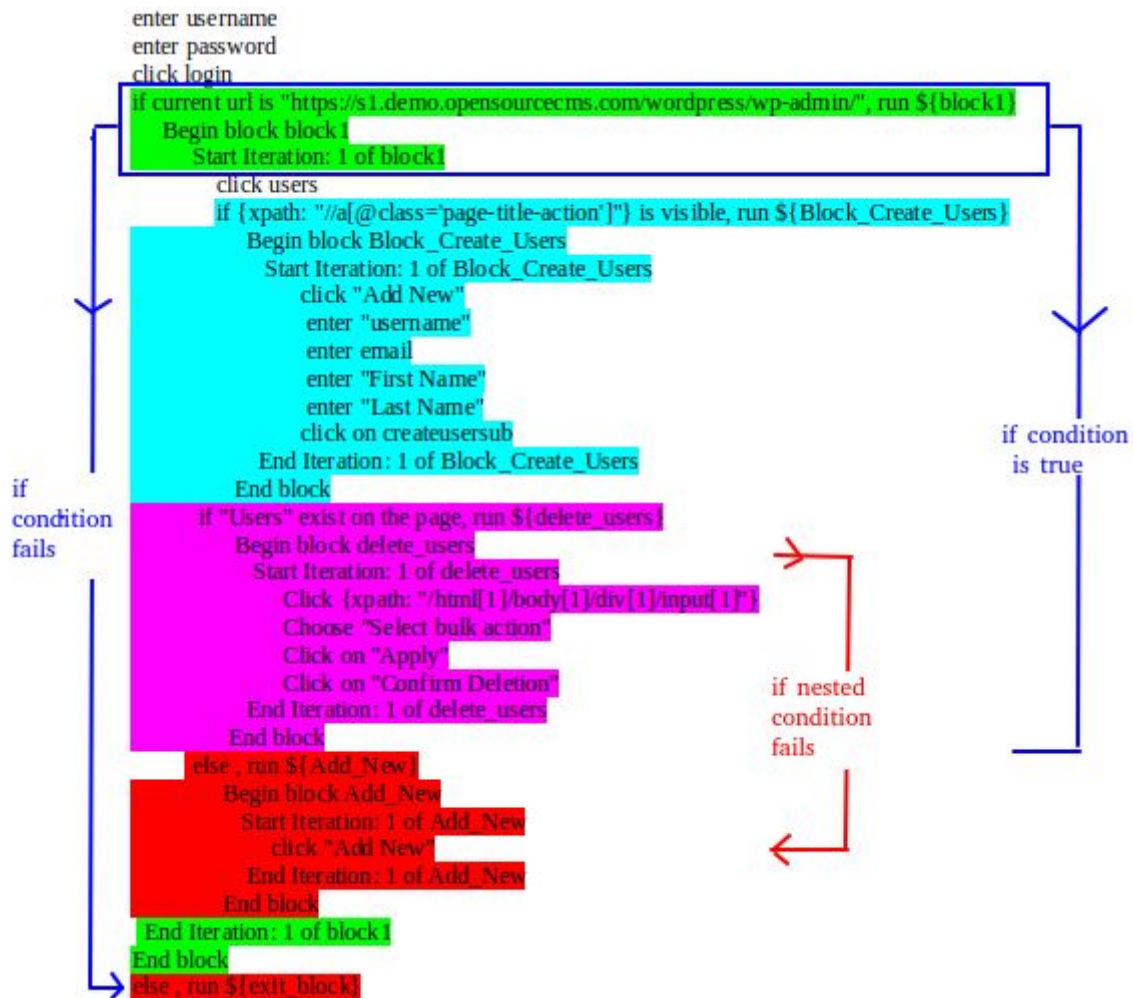
Syntax:

```

if (condition1), run ${block1}
Begin block block1
  //Nested if else inside the body of "if"
  if(condition2), run ${block2}
  Begin block block2
    //Statements inside the body of nested "if"
  End Block
  Else, ${else_part}
    // else_part is the flow here
  //Statements inside the body of nested "else"
Else, ${else_mainblock}
  //Statements inside the body of "else"

```

Example:



Else-if (elif)



The elif statement is useful when you need to check multiple conditions, nesting of if-else blocks can be avoided using else..if statement.

Note: instead of else if we need to mentioned as elif

Syntax:

```
if (condition1) run ${block1}
    //These statements would execute if the condition1 is true
elif(condition2) run ${block2}
    //These statements would execute if the condition2 is true and condition1 is
false
End block
.
.
Else, run ${else_block}
    //These statements would execute if all the conditions return false.
End block
End block
```

Example:

	<input type="checkbox"/>	2	if current url is "http://ninja.autonomiq.ai/ssignin", run \${login1}
	<input type="checkbox"/>	5	elif current url is "http://ninja.autonomiq.ai/signin", run \${login2}
	<input type="checkbox"/>	6	begin block login2
	<input type="checkbox"/>	7	enter username appuser
	<input type="checkbox"/>	8	enter password *****
	<input type="checkbox"/>	9	end block

In this example if condition is not satisfied so the block login1 is not executed, then the control moves to elif here the condition is satisfied and the block login2 is executed, when at least one condition is passed the else part will be skipped

Data Driven Parsing when condition satisfied

Data will be parsed when condition is satisfied

✓	<input type="checkbox"/>	1	open website http://ninja.autonomiq.ai
✓	<input type="checkbox"/>	2	run <code>#{block1}</code> <u>for all rows</u>
✓	<input type="checkbox"/>	3	Begin block block1
✓	<input type="checkbox"/>	4	if variable <u><code>#{username}</code></u> is test1 , run <code>#{block4}</code>
✓	<input type="checkbox"/>	5	begin block block4
✓	<input type="checkbox"/>	6	enter username test1

username	password
test1	pass1
test2	pass2

Above testcase run`#{block1}` for all rows will iterate through all rows in the data file. When condition matches for the current row that is running now, only that if block will get executed, Subsequent elif/else wont get executed, likewise if the condition did not match for other rows that if wont get executed.

If statements, continue

When the if condition is satisfied, with continue the control jumps to the beginning of the loop for the next iteration,

```
open website
Run #{check} for 2 times
Begin block check
enter username
enter password
click "log in"
end block
if "login" is on the screen, continue
```

Waits

Wait for given time

Wait for **number** seconds/secs

- wait for 3 seconds
- wait for 5 secs

Upload

Upload file should be uploaded to Artifacts section of Autonomiq. The field mentioned here will be the text in the upload search box.

Upload file to **[field]**

- upload file to "Resume" **resume.docx**

✓ 2 Upload file to {xpath:"//input[@type="file"]"} YH-F.jpg

Date Support



Get Today's date in a given format




This should be provided in the data tab.

{today, <format>}

- "Enter Date" + data tab should have {today, dd/mm/yy}
- "Enter Date" + data tab should have {today, mm/dd/yy}
- "Enter Date" + data tab should have {today, yy/mm/dd}

Relative Date support

Support is also provided for additions of days/months/years etc. This should be provided in data tab in UI. In uploaded file this should be present in the test data columned

Cache steps  Auto Scroll  **LIVE** ORIGINAL 

		Time
✓	<input type="checkbox"/>	1 open website http://example.com
✓	<input type="checkbox"/>	2 save variable as input_var 2
✓	<input type="checkbox"/>	3 save variable as var_month44 {Today + 2{mm}, mmmm}

Note: used format is not case-sensitive Here var_month44 store a month which is 2 months more than the current month

For running a job which should use today's date

- **{Today, MM/dd/yyyy}**

2 days ago from now

- **{Today - 2{dd}, MM/dd/yyyy}** undefined > 2 days later from now: **{Today + 2{dd}, MM/dd/yyyy}**

1 month ago from now

- **{Today - 1{mm}, MM/dd/yyyy}**

1 month later from now

- **{Today + 1{mm}, MM/dd/yyyy}**

1 year ago from now

- **{Today - 1{yy}, MM/dd/yyyy}**

1 year later from now

- **{Today + 1{yy}, MM/dd/yyyy}**

Using Variables

Creating variable

action and save it as **variable_name**

In this case, Instruction automatically get the value from the data tab and store it into my_name variable and they do act as is (this case, Enter username)

- Enter “username” and save it as my_name

Referencing previously created variable

`${varName}`

Note: Please refer to other sections also for other uses of variables.

Saving a variable from label on page or xpath

When saving variable, please note that only the text from that element or xpath will be saved.

save `{xpath:"xpath_value"}` as variable name

Local variable

Variable declared with _ variable name within the test step is a local variable which is accessible only within the test case.

`_var {" 123 "}` as _ variable name

eg: `_var{"123"}` as `_password`

Note: we cannot declare a local variable in the variables tab

Enter/Click when the element is stale

For Enterable and Clickable elements, the tool finds the element again if the variable (Safe_Click) is set as true. There are also changes for enter/click where the tool now handles the stale element exception. If the element is stale the tool now restarts the page and discovers the element again.

safe_click variable is true



Instruction to Verify variable is checked/unchecked

Step 1: get element details(click "text" checkbox) and save as variable

Step 2: verify variable `${variable.checked}` is "true"

Example test case:

open website

click "Remember me" checkbox

get element details(click "Remember me" checkbox) and save as check

verify variable \${check.checked} is "true"

✓ 2		<code>click "Remember me" checkbox</code>
✓ 3		<code>get element details(click "Remember me" checkbox) and save as check</code>
✓ 4		<code>verify variable \${check.checked} is "true"</code>

Variable ^	Value	Actions
<input type="checkbox"/> check	<pre>"rememberUn", "name": "rememberUn", "style": "", "type": "checkbox", "num_input_elems": 0, "attributes_input_elems": {}, "color_input_elems": {}, "text_content": ["Remember me"], "siblings": 2, "displayed": true, "enabled": true, "checked": true</pre>	<input type="checkbox"/> <input type="checkbox"/>

Other Misc

Optional Arguments

Optional arguments can be provided to a test step as described below -

Ignore Alert

By default, Autonomiq will check if a browser alert is present on the screen before interacting with any element on the screen. If an unhandled alert is present (alerts can be handled by – switch to alert and click on OK/Cancel), it'll purposely fail the test step with an error message stating that the alert is unhandled. If the user doesn't want for the test step to fail, they can use the ignoreAlert option as shown below

Click on **“login”** button **--ignoreAlert**

Dynamic Xpath

By default, Autonomiq caches xpaths for every test step so that subsequent script generations will be faster. However, if the user doesn't want to use the cached xpath for a certain step, they can provide the dynamicXpath option as shown below

Click on `${order_id}` --dynamicXpath

Note: If a certain xpath is not valid due to it being dynamic or an application change, it will be auto-healed which guarantees that the plain English step will not fail due to invalid xpaths.

Dealing with disabled elements (visually grayed out)

By default, Autonomiq will only interact with elements that are enabled. If the user wants to interact with a disabled element, they can use the Force option as shown below

Force click on "login"
where login button is grayed out.

Using Actions chain click

By default, Autonomiq uses selenium click and if selenium click fails, it'll switch to javascript click. However, if the user wants to specifically use action-chain click, they can provide it as shown below

Click on "login" --moveAndClick

Keyboard Operations

Following are the supported keyboard operations

Hit enter
Hit Down arrow key
hit up arrow key
Scroll up
Hit spacebar
Hit alt
Hit escape
Hit tab

Provide spinner/progress bar information

If the application under test has progress bars/spinners as a part of the UI design, Autonomiq provides the capability for users to specify the spinner information as a variable as shown below under “variables” tab. Once this information is provided, Autonomiq will dynamically wait until the progress bar/spinner disappears before proceeding with the next step.

Variable name : spinner_xpath

Variable value : xpath_of_the_spinner

No scroll option for a test case and project

When user does not require full screenshot of the test steps then No scroll option can be set true in the Variables tab, setting at variables tab will be applied at project level, since with no scroll option it is minimized taking full screenshot of the test steps this in turn will improve test case progress faster.

And also we have seen test steps behaving differently(actions not being performed) because of extensive scrolling. So with no scroll set at project level , test steps can behave properly

<input type="checkbox"/> Variable ^	Value	Actions
<input type="checkbox"/> no_scroll	true	⋮

No scroll option can also be applied to the specific test step

✓	3	Click on "California" --no scroll
---	---	-----------------------------------

Eg: Click on "California" --noscroll

When there is Hover and click scenario or drop-down selection scenario by giving --noscroll in the test step will not scroll the page for the respective test step.

Get element details

get element detail returns the object of the element attributes and store it in json format (i.e) get element detail can contain the element's tag name, attribute, visibility, enabled or not , size, location, the text of element, and store it in json format.

Eg1 : get element details(enter username) and save as xyz , xyz variable will have the element details

Variable ^	Value	Actions
<input type="checkbox"/> xyz	<pre>{ "tag_name": "input", "size": { "height": 32, "width": 268 }, "custom_tag": false, "location": { "x": 751, "y": 220 }, "attributes": { "aria-invalid": "false", "class": "MuiInput-input-52", "id": "username", "name": "username" } }</pre>	<input checked="" type="checkbox"/> <input type="checkbox"/>

✓ 2



get element details(enter username) and save as xyz

✓ 3



enter username

`${xyz.tag_name} -> ${xyz.tag_name}`

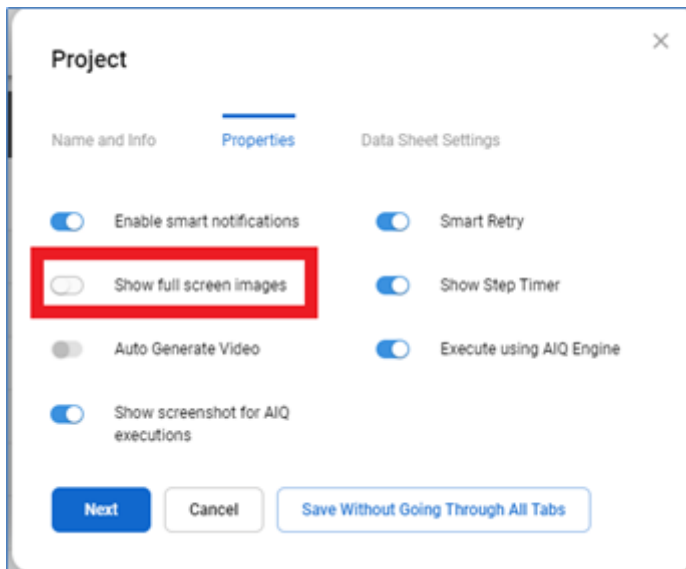
Full page screenshot for a project

Variable ^	Value	Actions
<input type="checkbox"/> use_full_screenshot	false	⋮

When user does not require full screenshot of the test steps then use_full_screenshot option can be set false in the Variables tab, setting at variables tab will be applied at project level, since with use_full_screenshot option it is minimized taking full screenshot of the test steps this in turn will improve test case progress faster.

or

When a user does not require a full screenshot of the test steps during generations or executions, then “Show full screen” images toggle button in Project Properties can be set to off. This will be applied at project level and since the tool will no longer scroll and take full screenshots of pages in test steps.

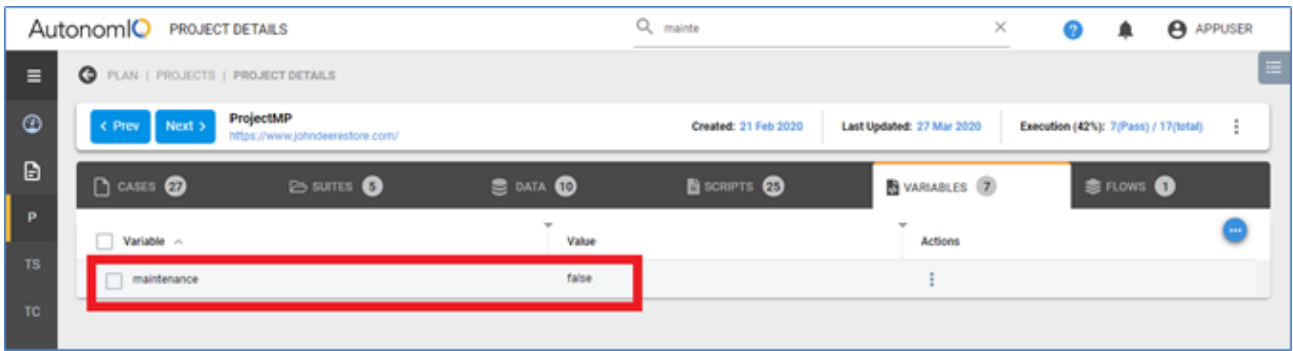


Maintenance

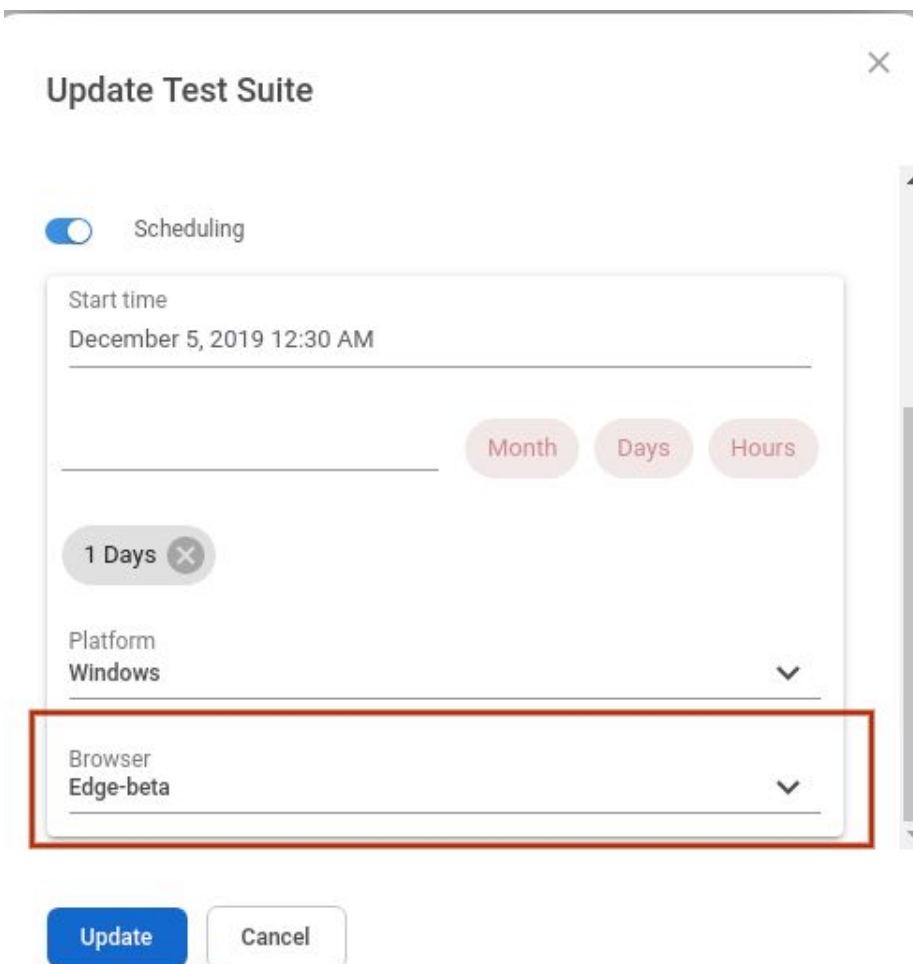
This feature self corrects XPATH and rediscovers the element if the older XPATH points to an invalid element.

Currently multiple XPATH's are generated while test cases are recorded or uploaded, and the same are passed on to the tools engine. As part of a new feature introduced the engine will validate the XPATHS and discard the ones which are pointing to incorrect elements. The engine will now retain only those XPATHS that are correct. If all the XPATHS are pointing to incorrect elements, then the engine does rediscovery and forms a new XPATH which may not have the attribute of old XPATH.

This feature can be turned off by specifying the maintenance variable as false, by default it is true.



Schedule in Edge Browsers



Now users have the option to execute/ scheduled suite with Edge browser.

Serial Execution

With serial execution mode, dependent test cases can be arranged and executed in the respective order where test cases will be executed one after the other

New Test Suite ×

Name and Info Schedule Test Cases **Arrange**

test

Search

<input type="checkbox"/> Case	Created	Last Run	Last Status
<input type="checkbox"/> New Test Case_23	Nov 20, 2019	Dec 19, 2019	N/A
<input type="checkbox"/> New Test Case_21	Nov 20, 2019	Nov 20, 2019	N/A
<input type="checkbox"/> New Test Case_22	Nov 20, 2019	Dec 5, 2019	N/A

Save Cancel

Execute Test



Suite Name

chk_repo

Platform

Linux

Browser

Chrome

Use Remote Driver

Execute

Complete suite

Filter by tag

Type to add a tag

Select Execution Mode

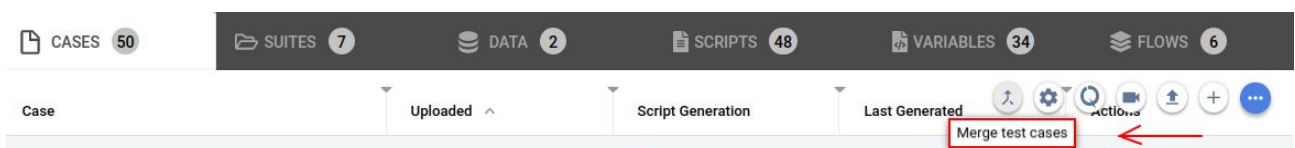
Serial

Execute

Cancel

Merge TestCase

This action offers the opportunity to merge multiple sets of test-cases in to one



Select the test cases to be merged

Merge Cases

Merge Arrange

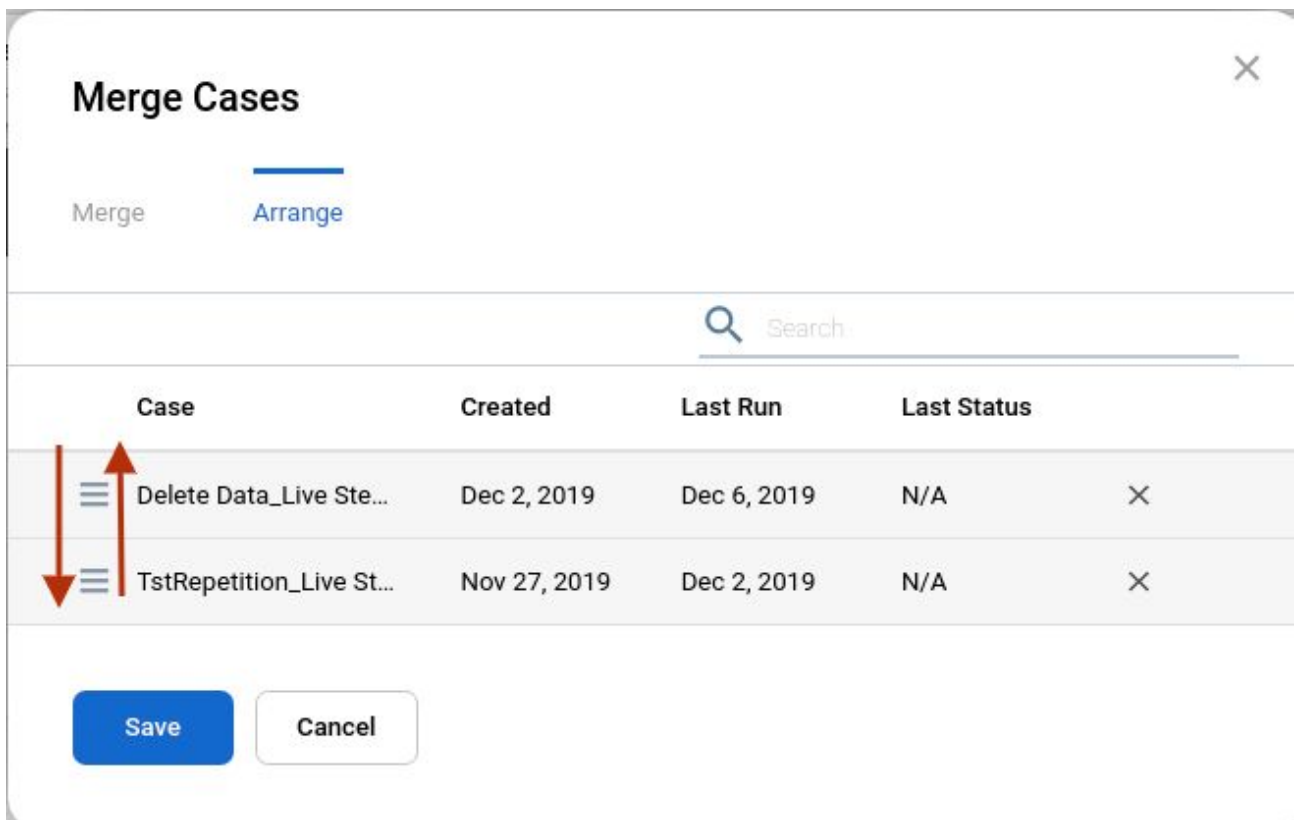
New Case Name
Test_Create_delete|

Search

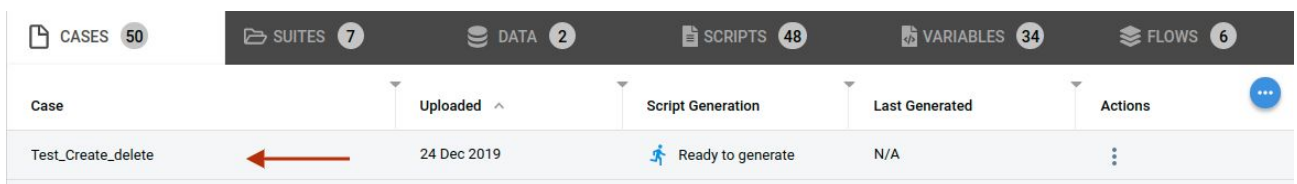
- Delete Data_Live Steps
- New Test Case_9
- TstRepetition_Live Steps (1)
- TstRepetition_Live Steps

Next Cancel

Arrange the order in a way the test case should be merged and then save



Merged test case created under Cases tab

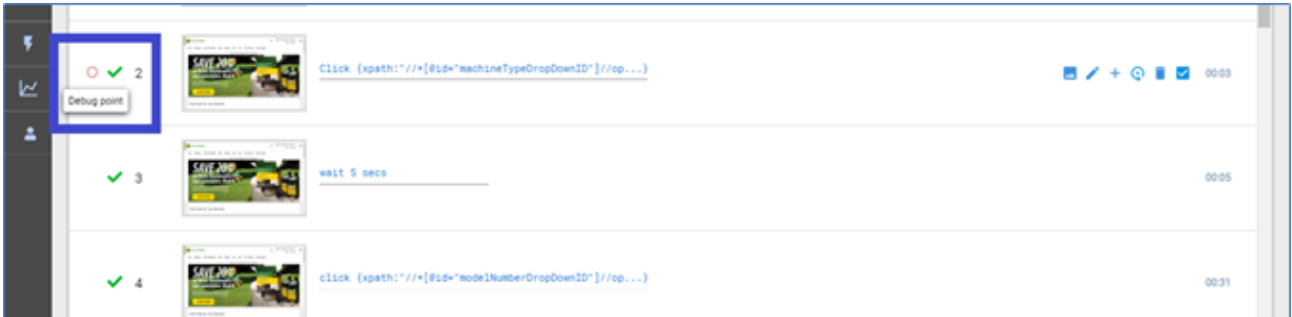


Note: Merged test case should retain the Data/ Artifact association will come in future release

Debug points

Users can stop the test case during generation at any step that the user pleases. Once stopped the user can make changes to the step where the test case stopped, or to the ones above it or the ones succeeding it. Debug Points can reduce time taken to create test cases significantly as the user doesn't have to start generation from the beginning in case a failure occurs

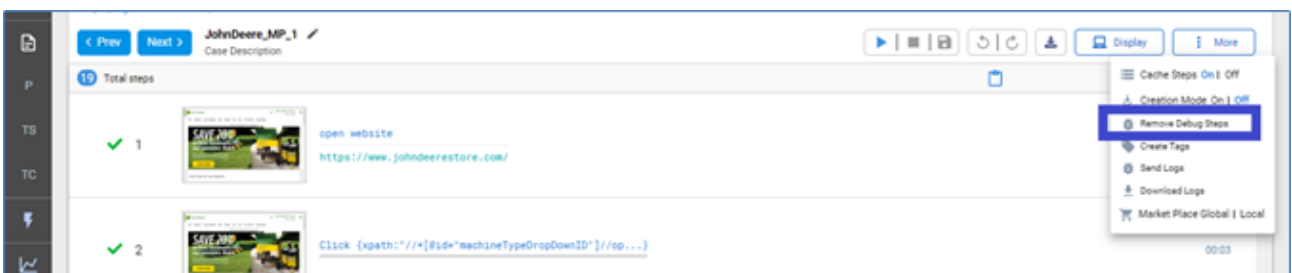
Debug Points are set from the step editor screen by clicking on the red circle that appears when a user hovers to the left of a test step.



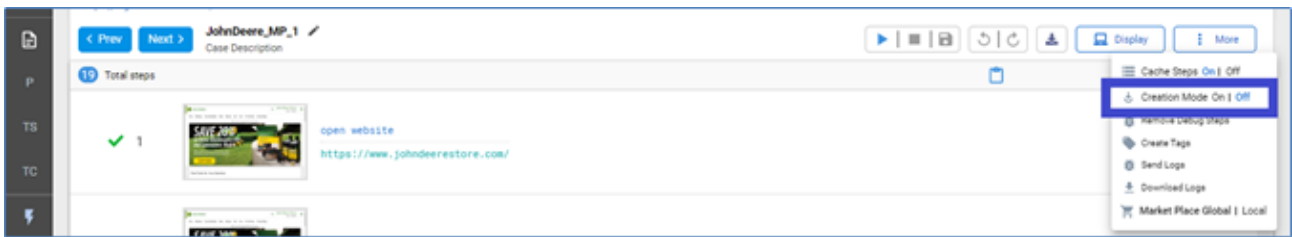
Debug Points can be set up for multiple or all steps, once selected a red dot signifies the existence of a Debug Point.



The selected Debug Points can be removed by clicking on red dots again. All the set Debug Points can be removed together by using the "Remove Debug Steps" option in More button.



Users also have the option to make the test case stop at each step during generation by using the Creation Mode in More button.



After a Debug Point is set and the test case stops at it, a Play button appears against the step. The user can validate the actions performed and continue generation by clicking on the Play button.

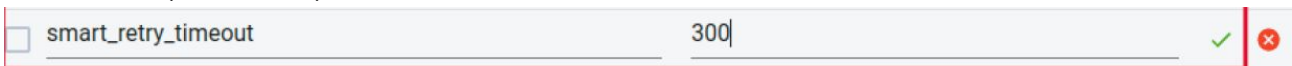


Once the Play button is clicked the test case will stop at the next Debug Point if set or if none are present will continue generation till the end.

If any step above a Debug point is changed the generation will restart from the modified step to preserve the integrity of the generation. Also, if no action is performed by the user once the test case stops at a Debug Point the tool will wait for 2 minutes before the debug expires and normal generation continues. To increase the debug time, you need to add a variable in the variable tab section.

Variable: **smart_retry_timeout**

Value : **300** (in seconds)



Debug Points also gives the capability to the user to update flows from a test case. Changes to steps in a flow can be made from the step editor screen. These changes are then visible from the Flows tab as well. This will improve test case creation times as the user doesn't have to navigate to Flows tab to make changes to a Flow.



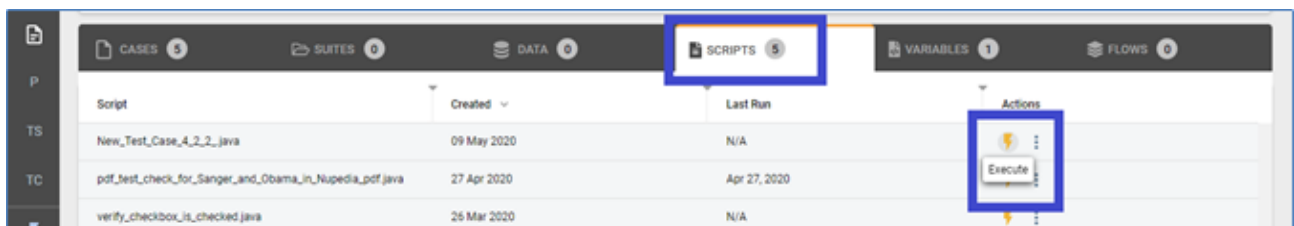
Debug Points must be used judiciously by the user and should not be used to alter previous steps when the driver state has changed. Also, in test cases where there are multiple iterations Debug Points can be set only for the first iteration. As it would be clear by now Debug points apply only to generations and not executions.

Mobile Browser Testing

Mobile Browser testing capability has been added via this release by leveraging Sauce Labs as a partner. The below Platform Browser combinations have been enabled as a Beta version.

- Windows 10, Chrome 80
- Windows 10, Firefox 74
- Windows 10, Edge 80
- Windows 10, IE 11.285
- Linux, Chrome 48
- Linux, Firefox 45
- macOS 10.13 (High Sierra), Safari 11.1

A user can access these options through the Scripts tab in the Project Details page by clicking on Execute.

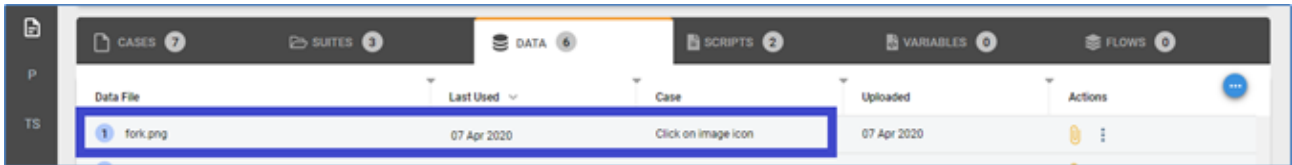


Mobile Browser testing cannot be executed via Suites currently, this feature will be rolled out in future releases.

Click on image icon with image uploaded as artifact

A user will now be able to click on an image icon on a webpage by using the same image that has been uploaded as an artifact. The image to be used for comparison and click is

uploaded as an artifact and associated with the relevant test case.



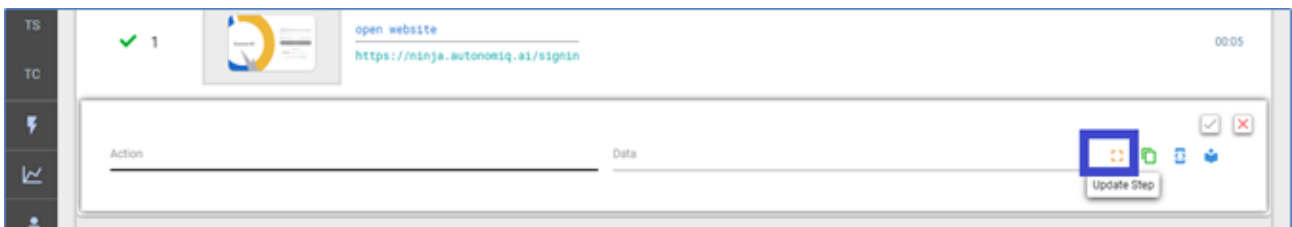
The artifact is then referenced in the test step to execute the click successfully.



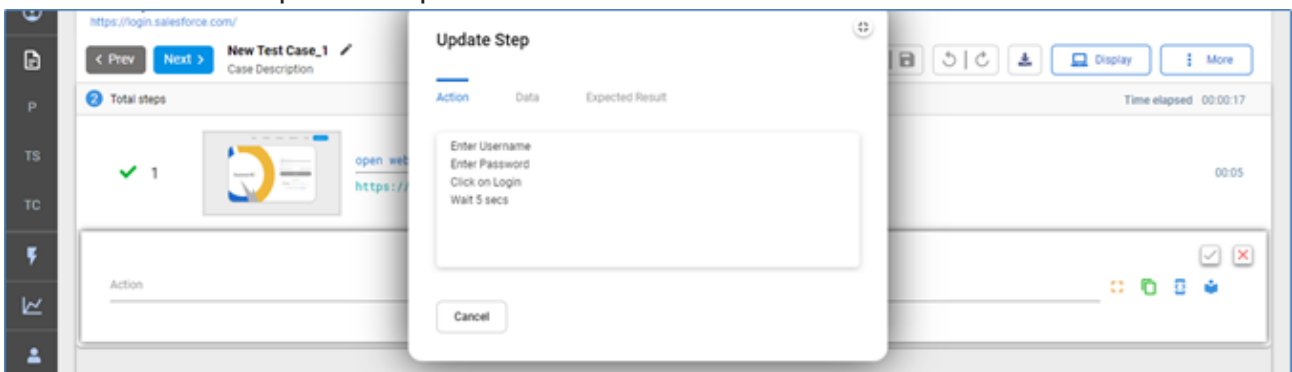
Multi Line Step Editor

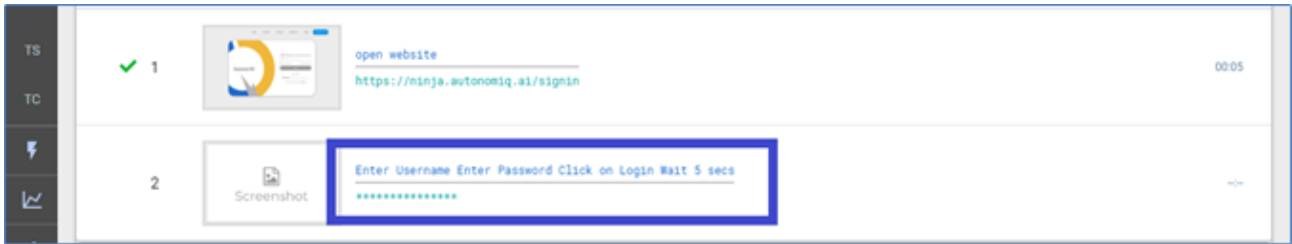
Multi Line Step Editor has been introduced to improve the tools online editing experience. Using the multi-line step editor multiple steps can be added to a test case in a seamless manner.

This feature can be accessed from the step editor page by clicking on the Update Step icon.



Information can be entered in Action, Date & Expected Results tab and the same will be converted as a compound step.





The test case can now be generated successfully.

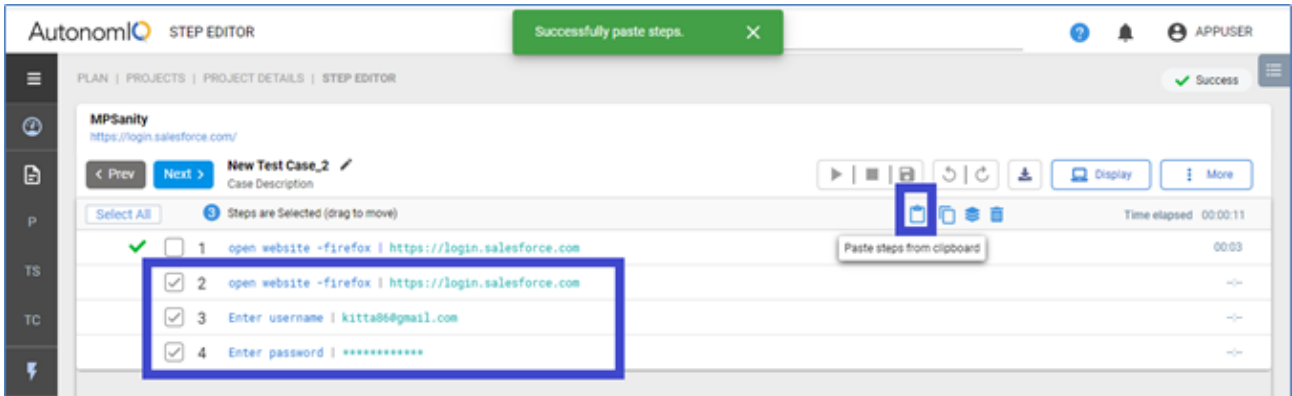
TS	✓	<input type="checkbox"/>	1	open website https://ninja.autonomiq.ai/signin	0005
TC	✓	<input type="checkbox"/>	2	Enter Username\Enter Password\Click on Login\Wait 5 secs *****	0016
	✓	<input type="checkbox"/>	2.1	Enter Username appuser	0003
	✓	<input type="checkbox"/>	2.2	Enter Password *****	0003
	✓	<input type="checkbox"/>	2.3	Click on Login	0002
	✓	<input type="checkbox"/>	2.4	Wait 5 secs	0005

Copy paste to and from excel

Test cases can be now copied and pasted directly onto the step editor page using this functionality. The new steps are appended at the end of the test case.

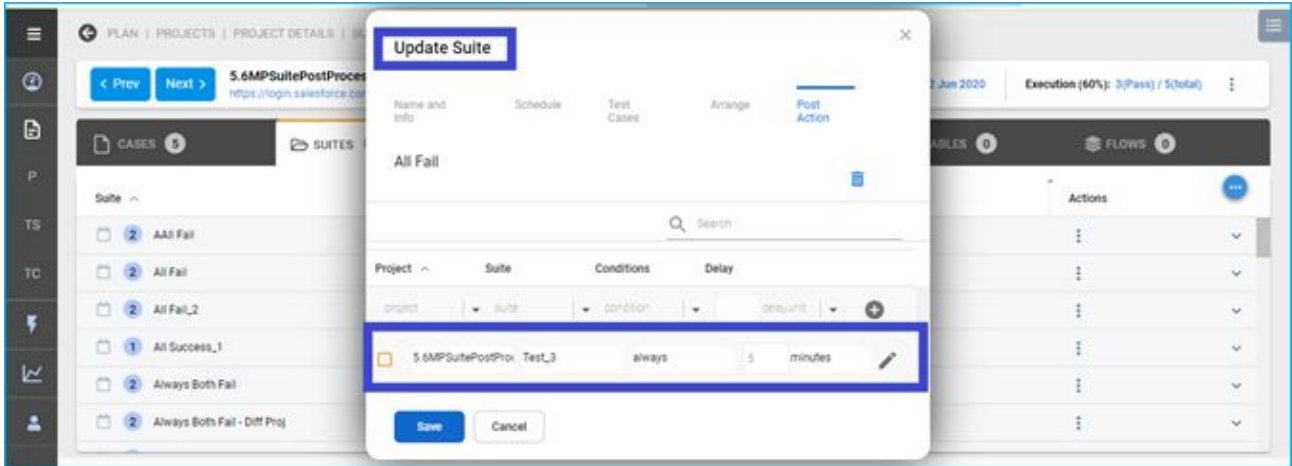
1	Test Steps	Data		
2	open website -firefox	https://login.salesforce.com		
3	Enter username	kitta86@gmail.com		
4	Enter password	Autonomiq12!		
5				

The copied steps can be pasted using the Clipboard icon on Step Editor page. The CTRL + V command can also be used for pasting the copied steps.

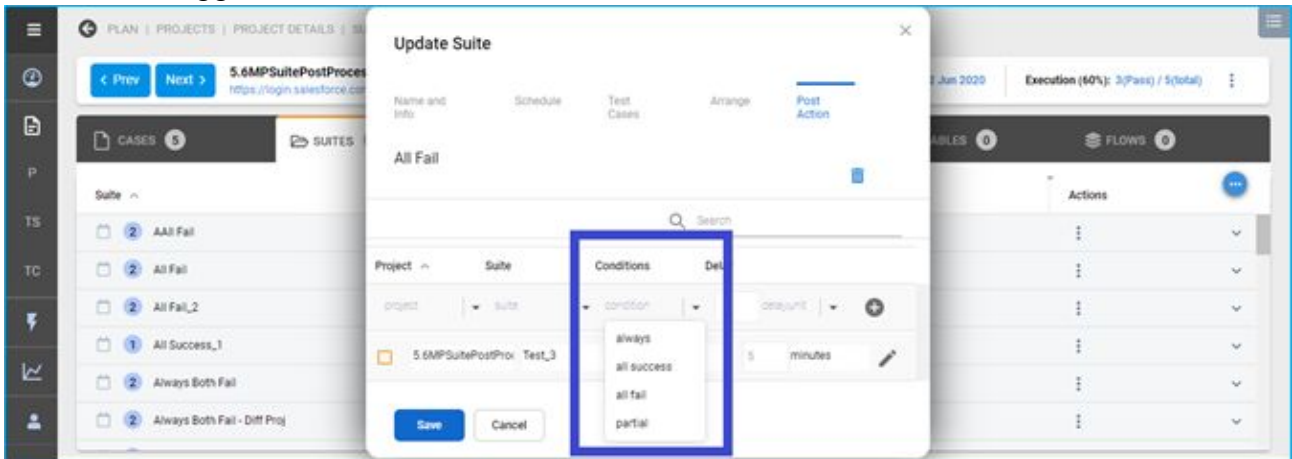


Post Execution Process for a Test Suite

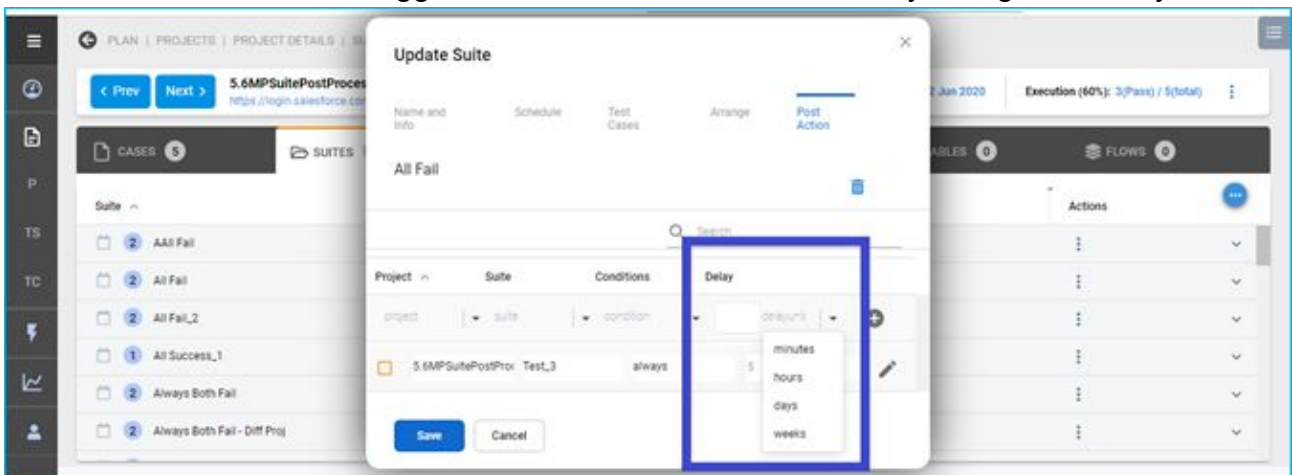
This new feature allows a test suite or suites to be triggered post the completion (Successful/Unsuccessful) of the parent test suite. A new tab has been added in the Update Suite modal to add post processing entries



Post processing suites can be from the same project or from different projects. Suites can be made to trigger based on different conditions as below.

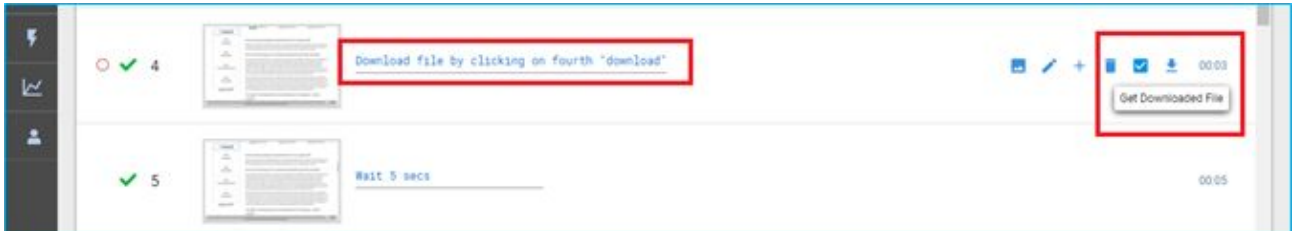


The suites can be also triggered at desired time intervals by using the Delay feature



Download and Upload

A user can now download a file through a test case by using the Download File command.



After the download is complete a new “Get Downloaded File” icon against the respective test step gets enabled. The user can download the file to his/her local machine by clicking on this icon.

The file that has been downloaded can be uploaded in subsequent steps by using the Upload File command as shown below. The file that was downloaded in Step 4 is being uploaded using the Upload File command. (Data column- **data:step 4**)



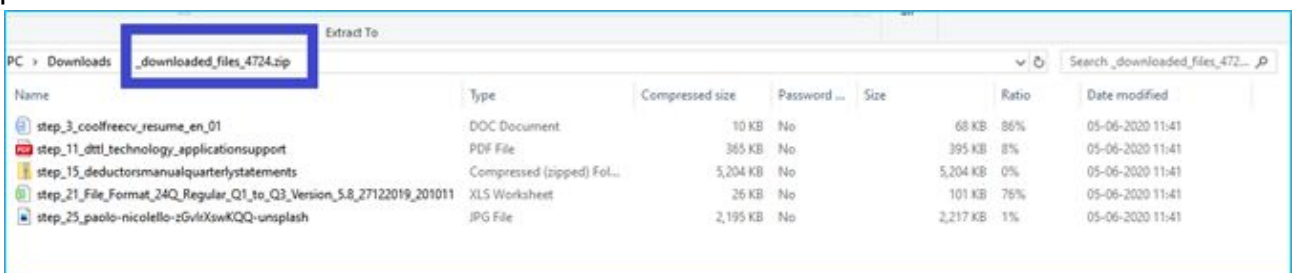
File downloaded in a test case can be also uploaded by specifying the file name as shown below.



The user can also download all the files that have been downloaded through a test case by clicking on the More button and then choosing “Get Downloaded Files”.



The downloaded files from the test case are available as a ZIP file with test step number prefixed to the file names.



Tool has a default wait timeout of 5 minutes for the download to start. The wait timeout can be modified by using **--downloadTimeout (Seconds)**.

Download file by clicking on "text" --downloadTimeout 600

Once a download starts the tool waits for 30mins (default) for the file download to complete. User can set the below **UI variable tab** to increase or decrease the wait time for file download complete

Variable: **download_complete_timeout**

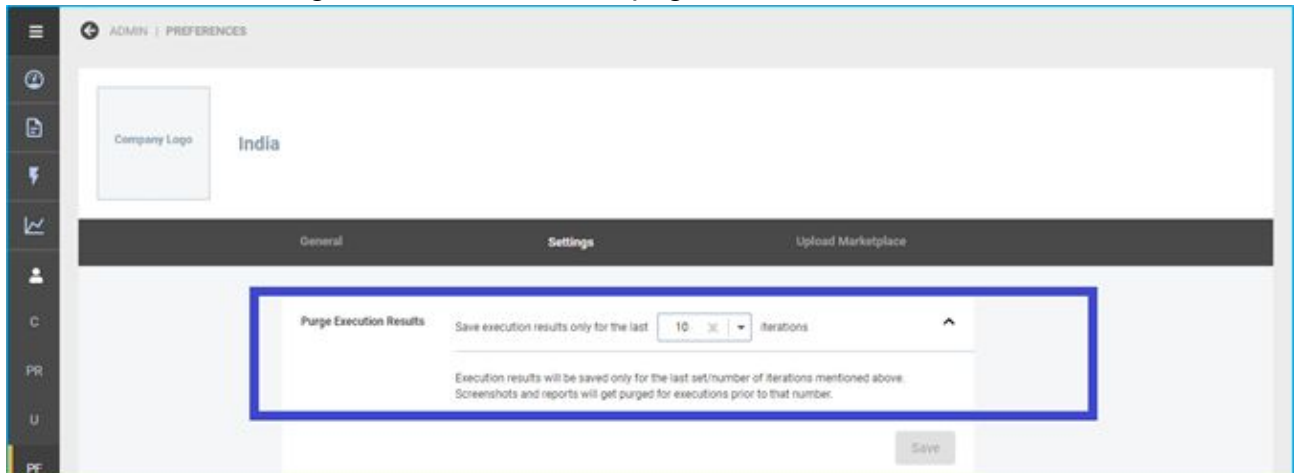
value: **<number of minutes>**

The upload and download are supported on the below browsers:

- Chrome (Local & Remote)
- Firefox (Local & Remote)
- Edge- (Remote) – Supported on the latest Chromium Edge browsers

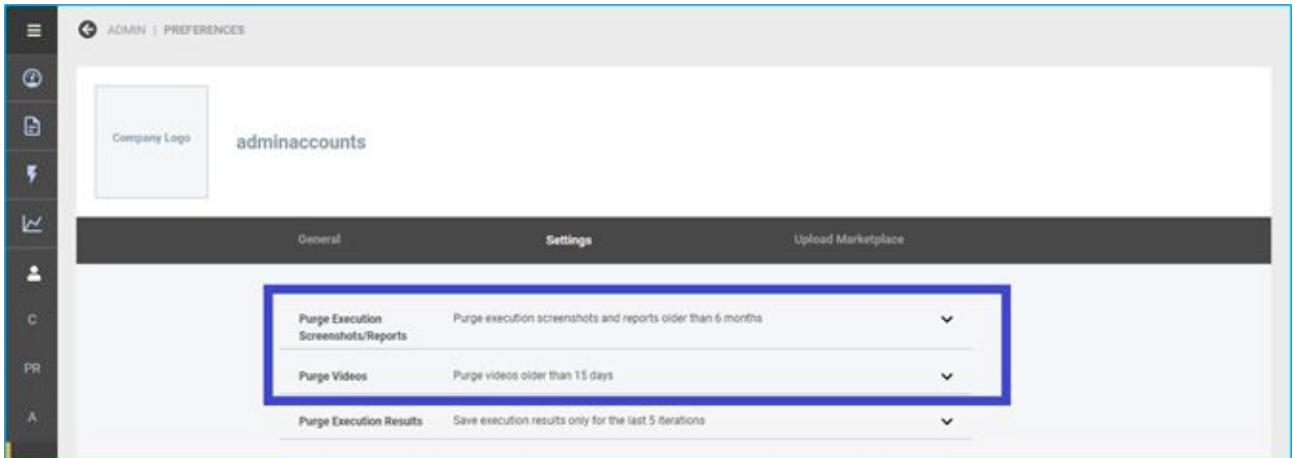
Automatic Disk Space Free-Up

Over a period the number of screenshots and reports from executions can clog the available disk space for a user. Now an admin user can manage the number of execution screen shots & reports that are stored through the Purge Execution Results setting. This is available under Settings on the Preferences page.



Any value between 5 and 20 can be set for the Purge Executions Results. So, if this setting is 5, then all execution screenshots and reports older than fifth execution will get purged thereby saving disk space.

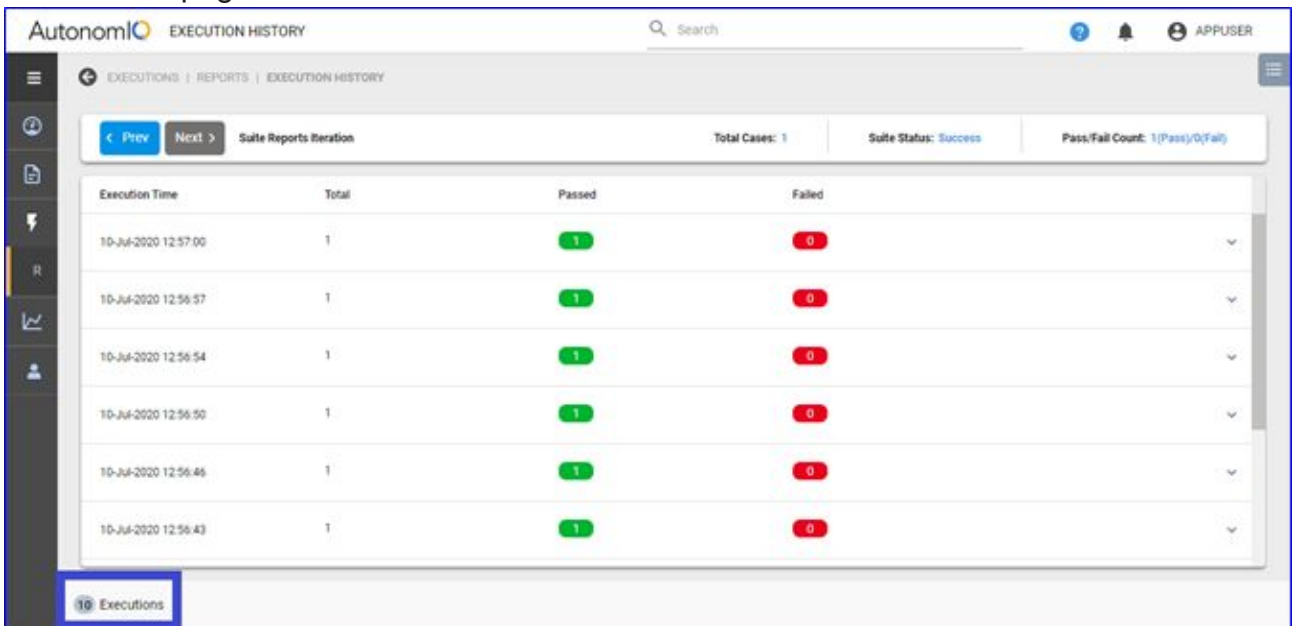
There are also two additional purge options that are available only to root user (appuser). Purge Execution Screenshots/Reports & Purge Videos options can be used to deleted screenshots/reports older than a period (Days/Months/Weeks/Year) or videos older than 15 days.

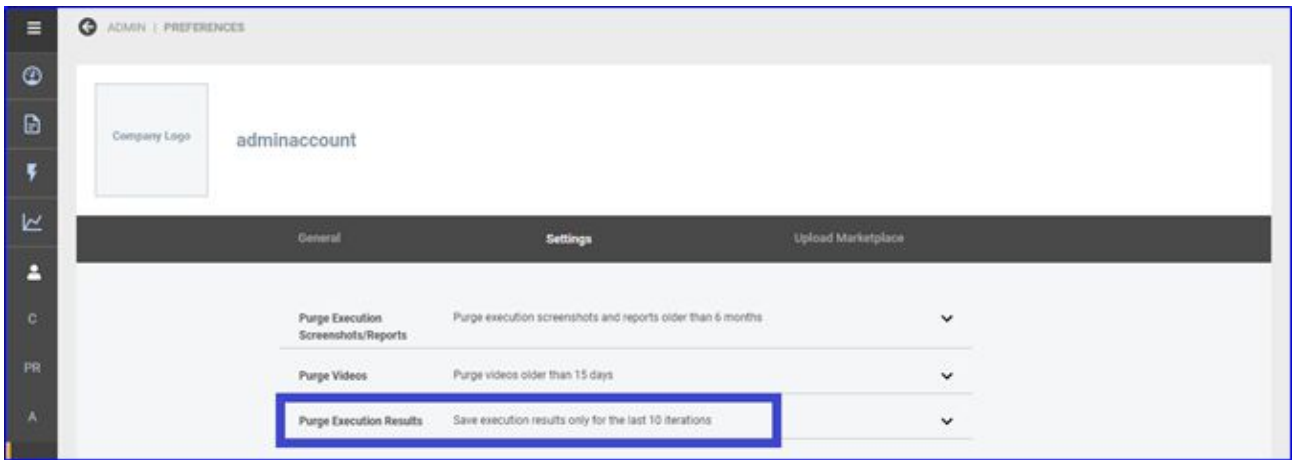


Any purge done using this option will be across accounts and therefore this feature should be exercised judiciously.

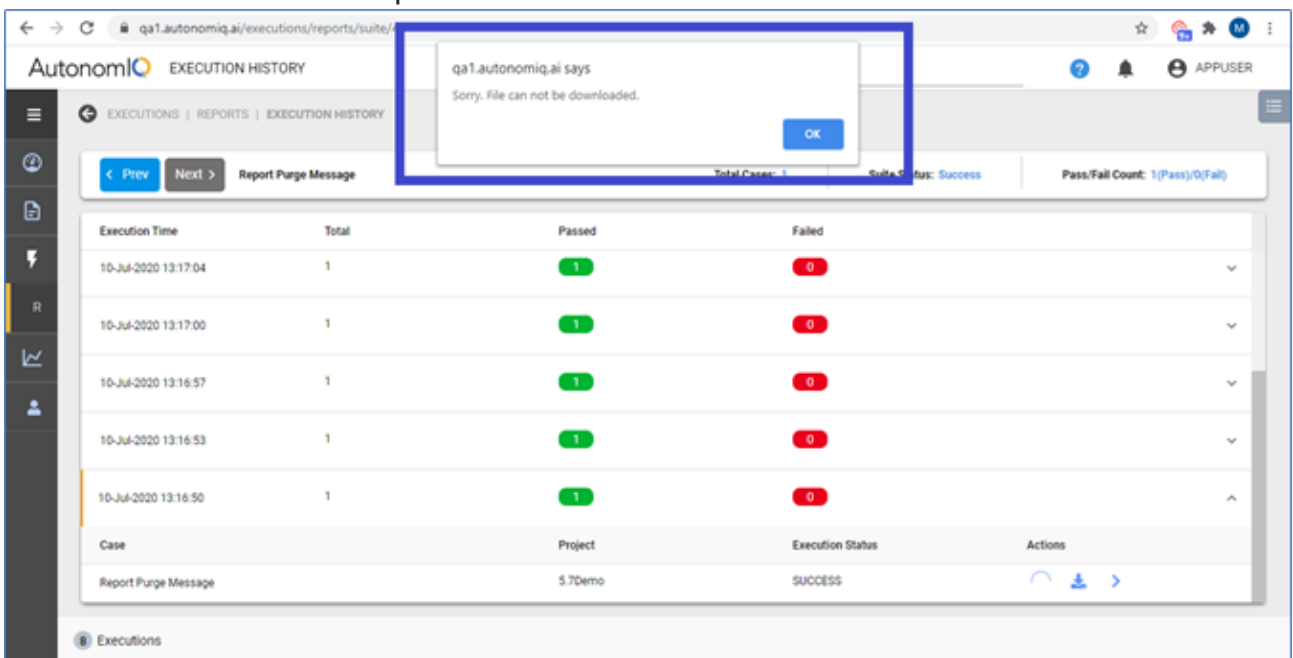
Report Iterations aligned with Purge Execution Results Setting

The number of versions maintained is matched with the Purge Execution Results setting Preferences page.





The Purge execution Results can have any value between 5 and 20 so a maximum of 20 report versions are possible to be maintained. If the report has been purged as per the purge execution setting a message will be displayed when the user clicks the Preview or Download button under a report iteration.



NLP Scroll Changes

Before **5.6 AIQ** version, NLP performed auto-scroll to search for elements on the webpage. In some cases, this was causing some dynamic elements present on the web page to vanish during scroll and this cascaded in test cases failing in subsequent steps. To avoid these failures auto-scroll has been turned off by default. Scroll could be turned on if the user desires by using the below variable.

Variable: page_scroll

Value: True or ON

Default: False

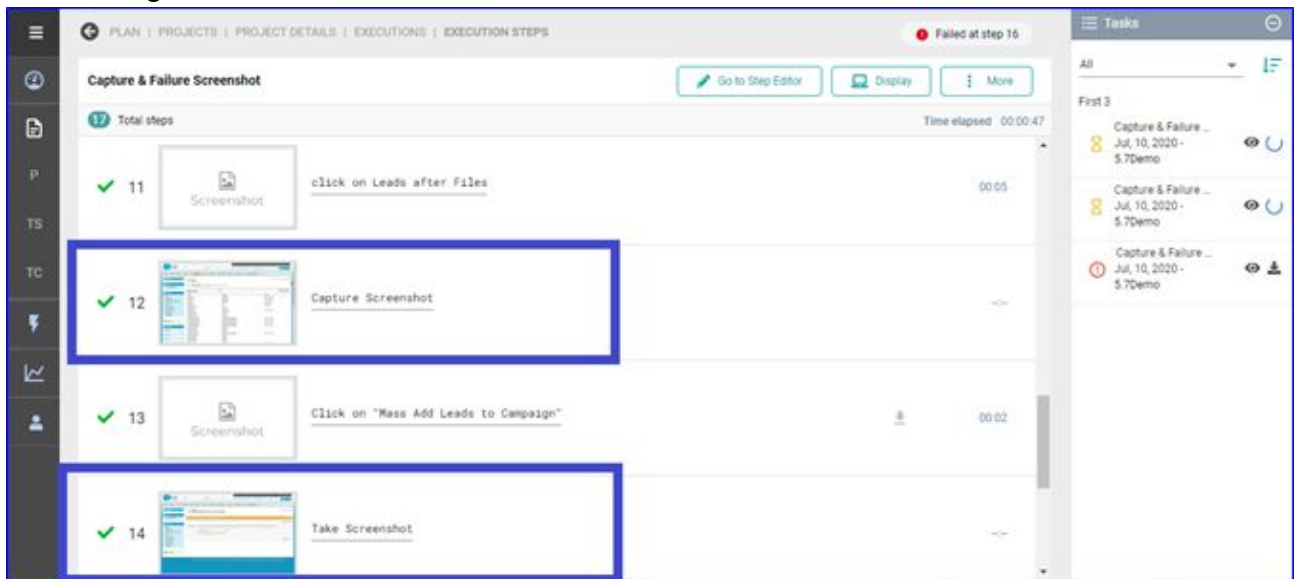
Steps to show screenshots during “AIQ Execution Screenshots OFF” mode

NLP steps:

Capture screenshot

Take Screenshot

Capture/Take Screenshot NLP will capture screenshots in-spite of setting screenshots OFF during executions.



The failed steps will also show the screenshots at “OFF” mode.



Remote execution and script generation

Using Autonomiq, script generations and executions can be done on a remote machine (machine that's different from the server on which Autonomiq is installed).

Variable name : use_remote_driver

Variable value : true

Variable name : capabilities

Variable value : { "url": "IP:PORT", "platform": "linux/windows",
"browser": "chrome/firefox/ie" }

Where IP – IP address of the remote machine
PORT - PORT on which selenium grid is running on the remote machine
Platform – remote machine (whether its windows or linux)
Browser – Browser on which you want to generate or execute the script

Zalenium/headful mode execution/generation

Execution or script generation videos can be recorded on AutonomIQ by setting the following variables under variables tab. This mode can also be used for live debugging.

Parameters to declare for headful mode generation

The below variables have to be declared to generate in headful mode.

Variable 1:

Variable name : use_remote_driver

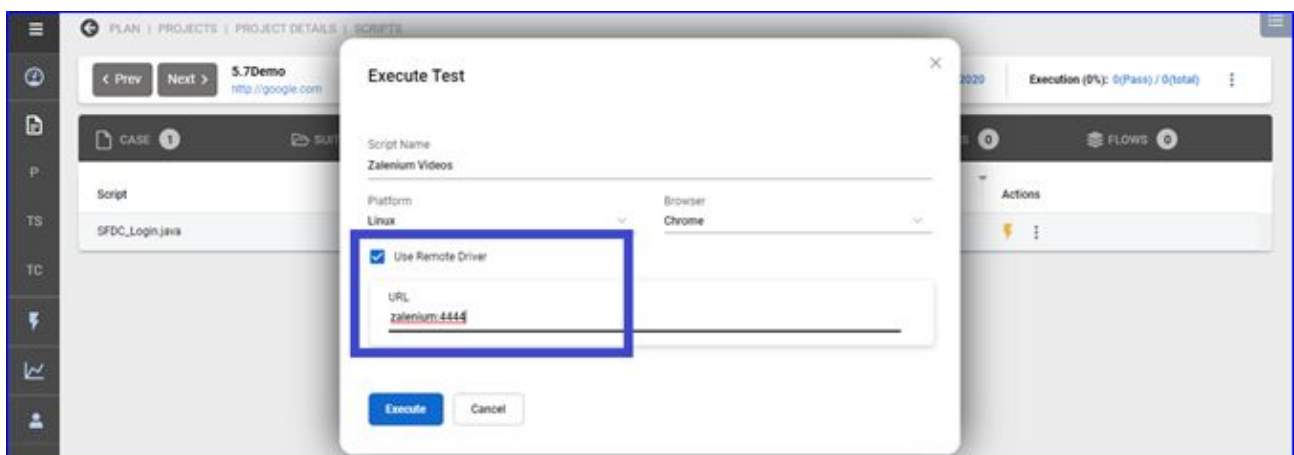
Variable value : true

Variable 2:

Variable name : capabilities

Variable value : {"url": "zalenium:4444", "platform": "linux", "browser": "chrome"}

Parameter to declare for headful mode execution

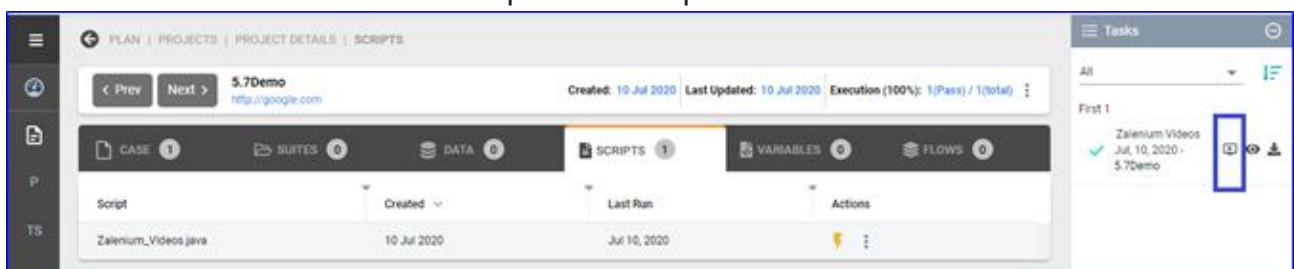


Url: zalenium:4444

To view the live generation/execution: url:4444/grid/admin/live

To view the recorded video after execution

These can be viewed from the task pane for scripts.



Use custom code from file

If a file is added to artifacts, then the user can use this code.

"use custom code from" + data tab should contain the name of the file
"Create random variable <var_name>" for a random alphanumeric string

- *used format is case-sensitive, so you should care about the name of a variable to use*

"Create random variable myprojectname" * how to use that variable in the following Instruction?
"Enter project name" + data tab should have "\${myprojectname}" value

- *above represent the format that we can use saved variable \${var_name}*

erred to a string * so in this case, they try to find element having given my_name string.

Test Data generation based on patterns

In the test data file, regex can be used to generate synthetic data. For example, following are some of the patterns -

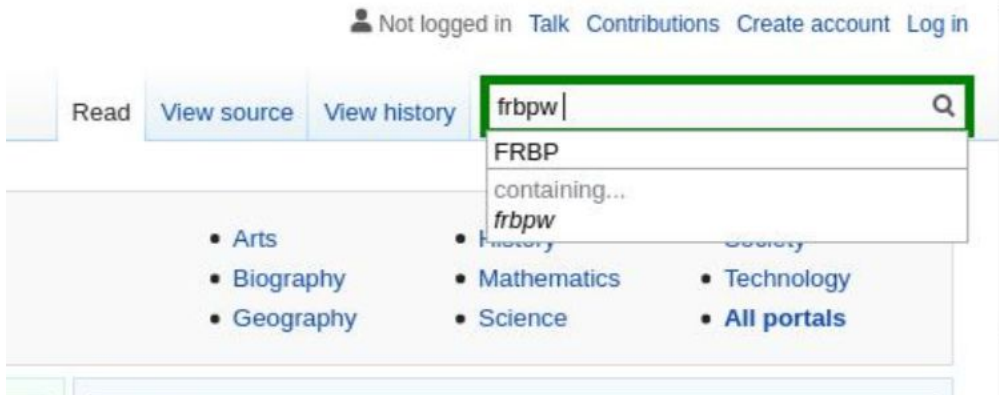
- `#{\d\d\d\d\d\d\d\d}` can generate 9446517410
- `#{[a-z][a-z][a-z][a-z][a-z]}` can generate lpwqh
- `#{[a-z][a-z][a-z][a-z][a-z]\d\d\d}` can generate sdkwf7809

✓ 2



enter search wikipedia

#{[a-z][a-z][a-z][a-z][a-z]}



Example

Actions with Proximity [Next to/before/after]

- click on "text" next to "text"
- click on "text" before "text"
- enter "text" after "text"

Script Execution

Autonomiq supports execution of Scripts - Python, JavaScript, Shellscript and Java. Simple actions can be performed using user written scripts.

This includes DOM handling and variable manipulation.

Exec `_js/_py/_bash`{script} with `$(input_variable_name)` returning `$(output_variable_name)`

if var = "testing", var_2 = "printing variable: testing"

- **exec `_py`{print('Modified variable: ' + aiq_1)} with `$(var1)` returning `$(var2)`**
- **exec `_js`{return 'Modified variable: ' + aiq_1} with `$(var1)` returning `$(var2)`**

- `exec _bash(echo 'Modified variable: ' + aiq_1) with ${var1} returning ${var2}`

Note:

- Any reference to input variable should be aiq_1 , aiq_2 and so on.
- Input variables can be numerous, they have to be comma separated
- Output variable support is currently only 1
- Here returning \${} is required with variable name. This is different from saving variable.
- Since the scripts being executed are separate, any kind of iframe navigation has to be handled within the script

Saving with Execution

Save `_js{return ${input_variable}}` as `output_variable`

- `save _js{return ${var}.replace("/", "/g")}` as `return_js`

JavaScript

Some examples of using JavaScript with Exec is given below

Set value for a Text box

```
exec _js{document.getElementById('someID').value=23}
```

Verify if a checkbox is checked

```
save _js{return document.getElementById("exampleCheck1").checked} as  
value
```

Verify if a checkbox is checked if in iframe

JS:

```
document.getElementsByName("iframe_name")[0].contentWindow.getElementById  
("exampleCheck1").checked
```

```
save _js{return
document.getElementById("iframeResult").contentWindow.document.getElementsB
yName("vehicle1")[0].checked} as value3
```

Note: If multiple iframe are present, repeat the code till the level you need to be

```
document.getElementById("iframeResult1").contentWindow.
document.getElementById("iframeResult1").contentWindow.document.getElementsByNam
e("vehicle1")[0].checked
```

Verify number of elements

```
Save _js{return document.getElementsByClassName("tfa-recent").length} as
varName
```

Get a values attribute

```
Save _js{return document.getElementsByClassName("tfa-recent").title} as varName
```

Find sum of numbers in a string

If num is "1|2|3" then sum will return 6

```
Exec _js{var
sum=0;aiq_1.split("|").forEach(function(val){sum=sum+parseInt(val)});return(su
m);} with ${num} returning ${sum}
```

JavaScript for variable comparision

```
exec _js{return 1 * aiq_1 > 1 * aiq_2 } with ${var_1}, ${var_2} returning ${var_5}
```

PythonScript for variable comparision

```
exec _py{print(int(aiq_1) > int(aiq_2)) } with ${var_1}, ${var_2} returning ${var_6}
```

Python floating point comparision

```
exec _py{print(float(aiq_1) > float(aiq_2)) } with ${var_1}, ${var_2} returning ${var_7}
```


Luhn Algorithm or Modulus 10 Algorithm

This will return true | false based on valid number given as input from variables.

```
Exec _js{var len = aiq_1.length;var sum = 0;for (var i = len-1; i >= 0; i--) {var d =
parseInt(aiq_1.charAt(i));if (i % 2 == (len)%2) { d *= 2; };if (d > 9) { d -= 9; };sum +=
d;};if(sum%10==0){return(true);}else{return(false);} }with ${num} returning ${status}
```

With further condition check on YYYYMM

```
Exec _js{var len = aiq_1.length; var init = aiq_1.substring(0,6);var sum = 0;var regex =
new RegExp("^\\d{4}(0[1-9]|1[0-2])$");for (var i = len-1; i >= 0; i--) {var d =
parseInt(aiq_1.charAt(i));if (i % 2 == (len)%2) { d *= 2; };if (d > 9) { d -= 9; };sum +=
d;};if(sum%10==0){if(regex.test(init)){console.log(true);}else{console.log(false);};}else{
console.log(false);} }with ${num3} returning ${status_fin}
```

Dropdown Validation

This will verify specific text is in dropdown or not.

1. Validate one value

Save {xpath: "//select[@id='year']"} as dropdown_values

save the text in the dropdown to a variable

_var{"1991"} as dropdown_validation

save the text that you want to make sure it exists or does not exists in dropdown
to a variable.

**Exec _js{return aiq_1.split("\n").includes(aiq_2)} with
\${dropdown_values},\${dropdown_validation} returning \${dropdown_result}**

run javascript to check 'dropdown_validation' is in 'dropdown_values'

Verify variable \${dropdown_result} is "True"

verify if dropdown_result is true or not.

3. Validate multiple values

Save {xpath: "//select[@id='year']"} as dropdown_values

`_var{"2017|2018|2019"} as dropdown_validation`

between multiple text, you should put "|".

```
Exec _js{return aiq_2.split("|").filter(x =>
!aiq_1.split("\n").includes(x)).length == 0; } with
${dropdown_values},${dropdown_validation} returning ${dropdown_result}
```

Verify variable `${dropdown_result}` is "True"

4. Return number of text that exists in dropdown

Save `{xpath: "//select[@id='year']"}` as `dropdown_values`

`_var{"2018|2020|2021"} as dropdown_validation`

```
Exec _js{return aiq_2.split("|").filter(x =>
aiq_1.split("\n").includes(x)).length; } with
${dropdown_values},${dropdown_validation} returning ${dropdown_result}
```

Verify variable `${dropdown_result}` is "2"

Shell

Find the difference between 2 dates

Here input variables are d1 and d2 and the difference in days is returned in diff

```
exec _bash{echo $(((date -d "${aiq_1}" '+%s') - $(date -d "${aiq_2}"
'+%s'))/86400)} with ${d1}, ${d2} returning ${diff}
```

Get System Time

- `Exec _bash{echo $(date +"%k")}` returning `{hr}`
- `Exec _bash{echo $(date +"%M")}` returning `{min}`

Other options:-

`%k` - hrs in 24 hr format

`%l` - hrs in 12hr format

`%M` - mins

`%S` - Seconds

(Ref: <https://www.cyberciti.biz/faq/linux-unix-formatting-dates-for-display/>)

Table Instructions

This will work for HTML tables with simple table-tr-td tags, not nested. This command identifies the cell below First Column for which Second Column's value matches.

Action on "**FirstColumnName**" Where **SecondColumnName** [operator]
SecondColumnValue

Clicks

Examples with different operators given below:

- Click on "Move" Where Winning % is 55.71
- Click on "Black" Where Black Elo is not undefined
- Click on "White" Where White Elo greater than 2700
- Click on "Move" Where Winning % lesser than 55
- Click on "White" Where White Elo begins with l
- Click on "Wins" Where Move starts with g
- Click on "Black" Where Date ends with ?
- Click on "White" Where Date contains 1994
- Click on "White" Where White Elo != undefined
- Click on "Move" where Winning % > 55
- Click on "Move" Where Winning % < 55
- Click on "Move" Where Winning % <= 55
- Click on "Move" Where Wins >= 70060
- Click on "Move" Where Wins = 70060

Enter text

Review is the column name here.

- Enter text in "Review" Where Location is Mysore

Radio Button Selection

Click on radio in "No" Where Location is Mysore

- No is the column name here.

Checkbox Selection

Click on checkbox in "Choose" Where Name is Anki

- Choose is the column name here.

DropDown Selection

Books is the column name here, the value to choose is given in testdata

- **Choose "Books" Where Name is SKM**

The Shinning is the value in the dropdown. The column in which this comes up need not be specified.

- **Choose "The Shinning" Where Name is Anki**

Click/ Enter text based on row number

Other similar commands are listed below

- **Click on "Move" Where row is 3**
- **Click on "Move" Where row is last**

Get the table row count

This will return the count of the number of rows in the table.

- **Get table row count as variable_Name**

Excel Functions

`_(xl|var){"excel formula"} as [variable_name]`

Summing up numbers

- `_xl{${var1}+${var1}} as sum1`
- `_xl{SUM(${var1},${var1})} as sum2`
-

Difference

- `_xl{${sum}-${var3}}`

Multiplication

- `_xl{17*3}` as var_float

Division

- `_xl{17/3}` as var_float

Round

- `_xl{ROUND(${var_float}, 1)}` as var_float_round

Greater than

- `_xl{${table_count} < 20}` as condition
 1. `_xl{ T(${var_1}) > T(${var_2}) }` as var_3 (This do TEXT comparison)
 2. `_xl{ 1 * T(${var_1}) > 1 * T(${var_2}) }` as var_4 (This do NUMERIC comparison)

Dates

- `_xl{TEXT(TODAY(), ""mm/dd/yyyy"")}` as var_date
- `_xl{TEXT(TODAY(), ""mmm-ddd"")}` as var_date2
- `_xl{TEXT(TODAY(), ""mmmm dddd"")}` as var_date3

Days between Dates

- `_xl{DAYS(${date1}.${date2})}` as date_diff

Currency Symbols as Prefix

The corresponding Currency symbol - in this case \$ - can be concatenated in the

✓ 16



`_xl{"$" & SUM({service_fee}, {cleaning_cost}, {tax}, {room_cost})} as actual_total_cost`

✓ 17



`Verify (xpath:"//*[@id='book_it_form']/div[2]/div[5]/div/div[2]/span/span") is ${actual_total_cost}`

beginning of the function

Other Misc

excel with numbers,

`"_xl{"$" & SUM("$18", "$12")} as var_sum",`

`"_xl{CEILING({var_float}, 1)} as var_float_ceil",`

`_xl{FLOOR({var_float})} as var_float_floor,`

`"_xl{MAXA("1", "2", "3", "4", "5")} as var_max",`

`"_xl{SMALL(["1", "2", "3", "4", "5"], 3)} as var_small",`

excel with string,

`"_xl{TRIM(" string with space ")} as var_str_trim",`

`"_xl{SUBSTITUTE("test test", "test", "testing")} as var_str_sub",`

`"_xl{SPLIT("111-222-333", "-")} as var_str_split",`

`"_xl{REGEXMATCH("https://test.com?params=testing", "https://test.com?")} as var_str_regex_match",`

`"_xl{REGEXREPLACE("test1 test2 test test", "test[0-9]", "test_with_number")} as var_str_regex_replace",`

Appendix

Data Backup

To take a manual backup , Copy “**aiq_data**” folder to the intended location outside the server.

Steps:

1. Create tar ball of aiq_data.
2. Calculate checksum of tar ball
3. Record checksum with date on shared file
4. Copy the data outside the server

5. Share the checksum with date to AIQ team