User Guide

NLP Commands - List

1. _var{[excel formula]} as [var_name]
2. _xl{[excel formula]} as [var_name]
3. [Action] and save it as [var_name]
4. Assert [text] is visible on the page
5. Assert image of "text", "text" is on the page
6. Begin block [block_name]
7. Begin script _bash with ${input_var}
8. Begin script _js with ${input_var}
9. Begin script _py with ${input_var}
10. Choose _css("[selector]")
11. Choose [text]
12. Choose {xpath: "[address]"}
13. Click _css("[selector]")
14. Click {xpath: "[address]"}
15. Click on radio next to [text]
16. Click on [exact attribute value]
17. Create random variable [var_name]
18. End block
19. End script save as ${output_var_name}
20. Enter text in _css("[selector]")
21. Enter text in [text]
22. Enter text in {xpath: "[address]"}
23. Exec _js{[script]} with ${input_var} returning ${output_var}
24. Exec _py{[script]} with ${input_var} returning ${output_var}
25. Exec _sh{[script]} with ${input_var} returning ${output_var}
26. Exec "[artifact file]" with ${input_var} returning ${output_var}
27. Fill in text in _css("[selector]")
28. Fill in text in [text]
29. Fill in text in {xpath: "[address]"}
30. Go to website
31. Hover on _css("[selector]")
32. Hover on [text]
33. Hover on {xpath: "[address]"}
34. Hover over _css("[selector]")
35. Hover over [text]
36. Hover over {xpath: "[address]"}
37. Launch website
38. Market:
39. Navigate website
40. Open website
41. Press _css("[selector]")
42. Press {xpath: "[address]"}
43. Press on radio next to [text]
44. Run ${flow name} for [number] times
45. Run ${flow name} for all rows

46. Save _css("[selector]") as [var_name]
47. Save {xpath: "[address]"} as [var_name]
48. Select _css("[selector]")
49. Select [text]
50. Select {xpath: "[address]"}
51. Set screen/window size - [width] * [height]
52. Set text in _css("[selector]")
53. Set text in [text]
54. Set text in {xpath: "[address]"}
55. Switch to 2nd tab
56. Switch to 2nd window
57. Switch to 3rd tab
58. Switch to 3rd window
59. Switch to alert and click on accept
60. Switch to alert and click on cancel
61. Switch to alert and click on leave
62. Switch to alert and click on ok
63. Switch to alert and click on stay
64. Switch to confirm and click on accept
65. Switch to confirm and click on cancel
66. Switch to confirm and click on leave
67. Switch to confirm and click on ok
68. Switch to confirm and click on stay
69. Switch to new tab
70. Switch to new window
71. Switch to original tab
72. Switch to original window
73. Switch to prompt and click on accept
74. Switch to prompt and click on cancel
75. Switch to prompt and click on leave
76. Switch to prompt and click on ok
77. Switch to prompt and click on stay
78. Type in text in _css("[selector]")
79. Type in text in [text]
80. Type in text in {xpath: "[address]"}
81. Upload file to _css("[selector]")
82. Upload file to [text]
83. Upload file to {xpath: "[address]"}
84. Use custom code from [location]
85. Verify _css("[selector]") begins with [text] or begins with [text]
86. Verify _css("[selector]") begins with [text] or ends with [text]
87. Verify _css("[selector]") begins with [text]
88. Verify _css("[selector]") contains [text] or begins with [text]
89. Verify _css("[selector]") contains [text] or contains [text]
90. Verify _css("[selector]") contains [text] or ends with [text]
91. Verify _css("[selector]") contains [text]
92. Verify _css("[selector]") ends with [text] or ends with [text]
93. Verify _css("[selector]") ends with [text]

94. Verify _css("[selector]") is _css("[selector]")
95. Verify _css("[selector]") is _css("[selector]")
96. Verify _css("[selector]") is disabled
97. Verify _css("[selector]") is enabled
98. Verify _css("[selector]") is not visible
99. Verify _css("[selector]") is visible
100. Verify [text] is on the page
101. Verify {xpath: "[address]"} begins with [text] or begins with [text]
102. Verify {xpath: "[address]"} begins with [text] or ends with [text]
103. Verify {xpath: "[address]"} begins with [text]
104. Verify {xpath: "[address]"} contains [text] or begins with [text]
105. Verify {xpath: "[address]"} contains [text] or contains [text]
106. Verify {xpath: "[address]"} contains [text] or ends with [text]
107. Verify {xpath: "[address]"} contains [text]
108. Verify {xpath: "[address]"} ends with [text] or ends with [text]
109. Verify {xpath: "[address]"} ends with [text]
110. Verify {xpath: "[address]"} is {xpath: "[address]"}
111. Verify {xpath: "[address]"} is {xpath: “[address]"}
112. Verify {xpath:”[address]”} background-color is #ffffff
113. Verify {xpath:”[address]”} color is #e01719
114. Verify {xpath:”[address]”} font-size 26px
115. Verify {xpath: "[address]"} is disabled
116. Verify {xpath: "[address]"} is enabled
117. Verify {xpath: "[address]"} is not visible
118. Verify {xpath: "[address]"} is visible
119. Verify alert is exists
120. Verify pop up is exists
121. Verify tab is exists
122. Verify the current url is [url]
123. Verify url is [url]
124. Verify variable ${var_name} is [text]
125. Verify window is exists
126. Wait [number] secs
127. Wait for [number] seconds
128. Wait for [number] secs
129. Wait until _css("[selector]") is exists
130. Wait until _css("[selector]") is visible
131. Wait until [text] is exists
132. Wait until [text] is visible
133. Wait until {xpath: "[address]"} is exists
134. Wait until {xpath: "[address]"} is visible

NLP Commands Detailed

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

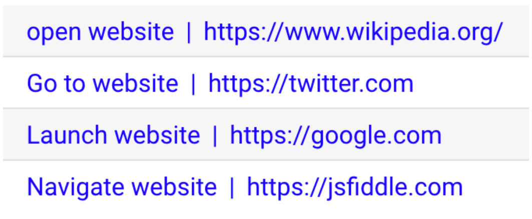
Window/Tab/Alert

Navigate to a URL

Navigates to the url. This should always be the first step in any testcase, since each testcase in 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]

The url can be given from the data tab.



open website	https://www.wikipedia.org/
Go to website	https://twitter.com
Launch website	https://google.com
Navigate website	https://jsfiddle.com

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

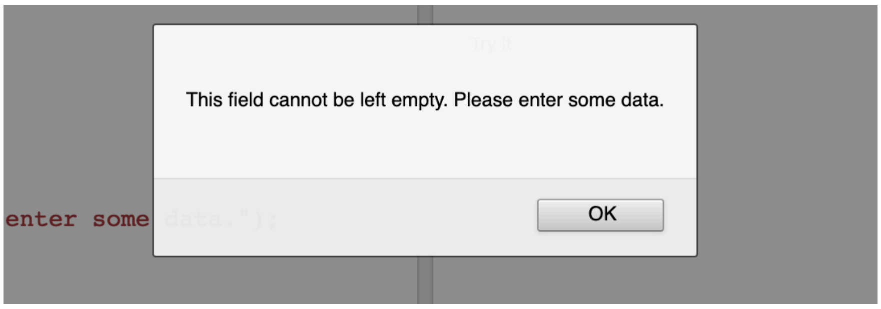
For a new window or tab

- **"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


This field cannot be left empty. Please enter some data.

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 win 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 user name"**
- **"click at user name"**
- **"press on password"**
- **"press at password"**

Click on radio/checkbox

Click/Press on/at [radio/checkbox] next to [element text]

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

- **"click on the radio next to username"**
- **"press at the radio next to password"**

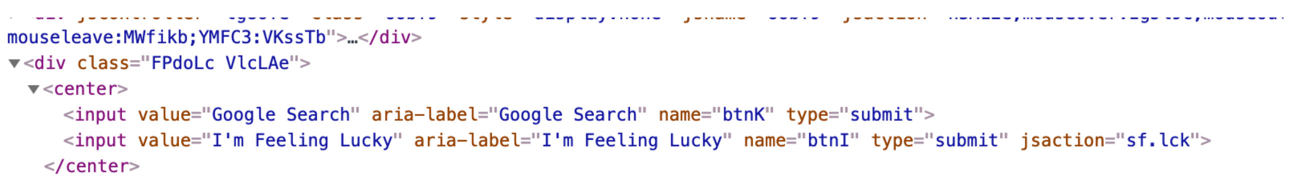
Click on Attribute value

Click also works on unique attribute values.

Click on [exact attribute value]

For example,

- **Click on btnk**



```
[illegible]
mouseleave:MWfikb;YMFC3:VKssTb">…</div>
▼<div class="FPdoLc VlcLAe">
  ▼<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>
```

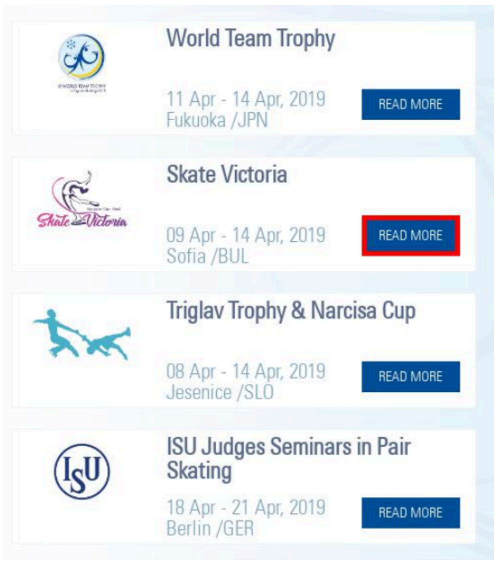
Note: This is case insensitive

Click based on Container

Click on [identifier1] for [identifier2]

Can be used in the following scenarios:

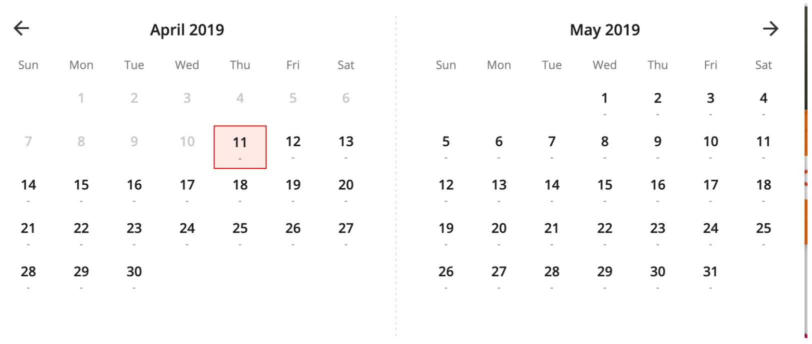
1. Where there are multiple repetitions of the same button, but there are unique identifiers for each which is present in the container surrounding it.
   ex: Click on READ MORE for Skate Victoria
2. When there are multiple calendar pages (similar situation as above
   ex: Click on 13 for May 2019


World Team Trophy
11 Apr - 14 Apr, 2019
Fukuoka /JPN
READ MORE

Skate Victoria
09 Apr - 14 Apr, 2019
Sofia /BUL
READ MORE

Triglav Trophy & Narcisa Cup
08 Apr - 14 Apr, 2019
Jesenice /SLO
READ MORE

ISU Judges Seminars in Pair Skating
18 Apr - 21 Apr, 2019
Berlin /GER
READ MORE


April 2019

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

May 2019

Sun	Mon	Tue	Wed	Thu	Fri	Sat
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
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]

Select/Choose

Select/choose [element text]

In below cases data to be selected should be sent from data tab

- **"select username"**
- **"choose password"**

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]

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

Hover

Hover over/on [field]

"hover over username" / "hover on username"

Validations

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

Using Verify will ensure that the testcase fails but the 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. This works of part of a phrase also.

Verify [some text] is on the screen.

- **"verify 'login' is on the screen"**

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 image of something is on page

This command is for verify some image is on the page.

Verify image of "something" is on the page.

- **"verify image of 'person', 'man' is on the page”**
-

Verify text using xpath

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

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

Verify if xpath matches with given xpath

Presently verify command supports elements which can be found with xpath

Verify {xpath: "xpath"} is [xpath]

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

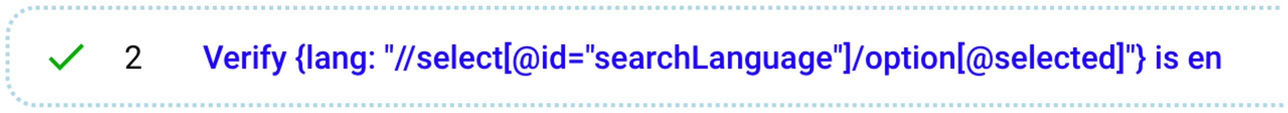
Verify attribute of an xpath

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

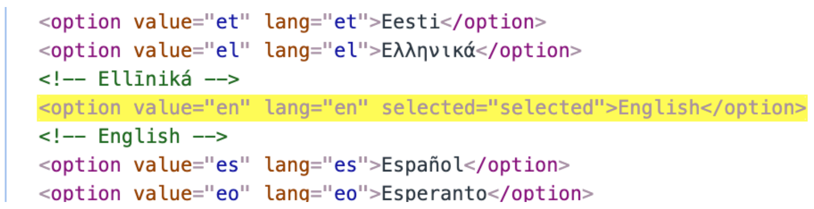
Verify name/placeholder/data-toggle/value/class/type/any-attr of {xpath: "xpath"} is "some value"

- **"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



```
<option value="et" lang="et">Eesti</option>
<option value="el" lang="el">Ελληνικά</option>
<!-- Ellīniká -->
<option value="en" lang="en" selected="selected">English</option>
<!-- English -->
<option value="es" lang="es">Español</option>
<option value="eo" lang="eo">Esperanto</option>
```

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"

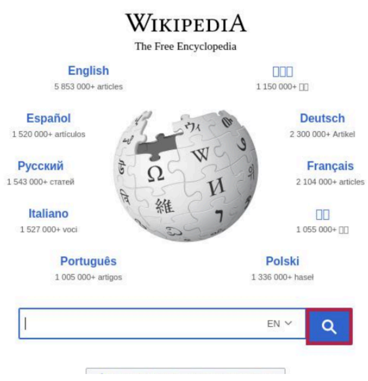
- **"verify variable ${var1} is "test"**

Verify CSS properties with xpath

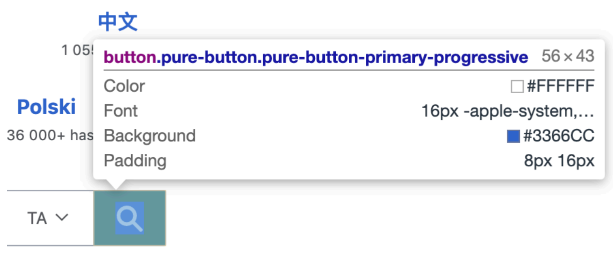
Any CSS attribute of an element can be validated to be expected value using the xpath of the element.

Verify {xpath: "xpath"} width/height/font-family/text-align/font-size/display/color/background-color is "some value"

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



e='submit']"} background-color is #3366cc


button.pure-button.pure-button-primary-progressive 56 × 43
Color □#FFFFFF
Font 16px -apple-system,...
Background ■#3366CC
Padding 8px 16px

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 ${var1} is on page
- "verify ${var1} is visible

2 Verify ${something} is on page

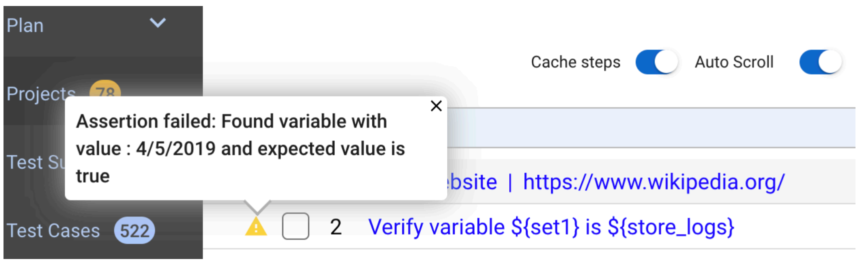
Verify text between 2 variables

Verify variable ${var_1} is ${var_2}

- "Verify variable ${set1} is ${set2}"

✓ 2 Verify variable ${set1} is ${set2}

Sample failed case:

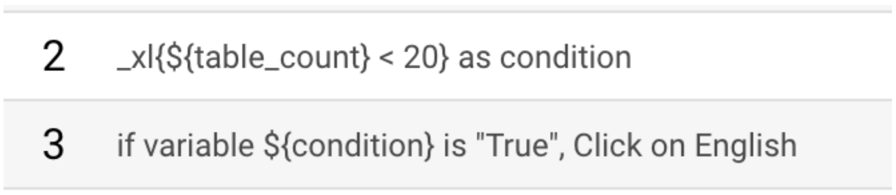


Assertion failed: Found variable with value : 4/5/2019 and expected value is true

bsite | https://www.wikipedia.org/

2 Verify variable ${set1} is ${store_logs}

Conditional Instructions

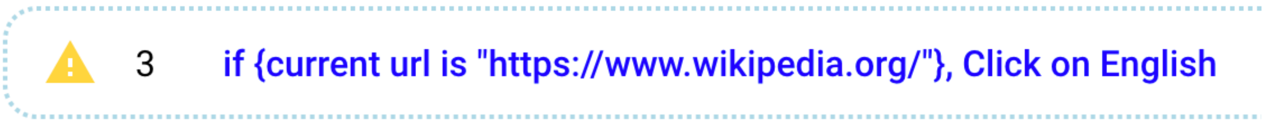


2	_xl{${table_count} < 20} as condition
3	if variable ${condition} is "True", Click on English

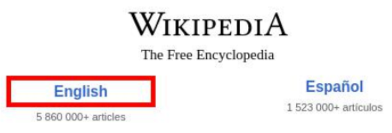
These instructions are executed if certain conditions are true. Condition is similar to as 'verify'. Use this instead of verify if other instrucitons are to be performed after it.

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

- **if {current url is “https://..."}, enter userrname**
- **if {xpath:'xpath_'} is visible, Click on Submit**
-



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



Looping Statements

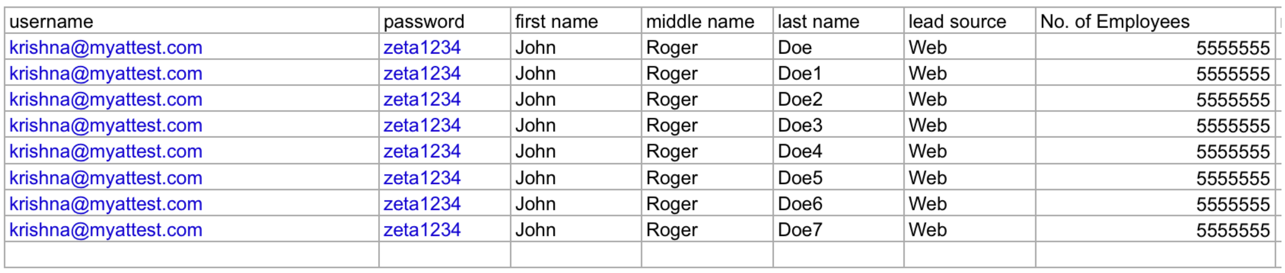
Blocks can be used to loop through commands as many times as required.

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**
-



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

run ${block_name} for (number/all) rows

- **run ${sample_block} for all rows**

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 a loop

run ${block*name} until {condition}*

The below command is similar to verify

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

Note: While generating, only the first loop is used. During generation, all the data will be used and looped through.

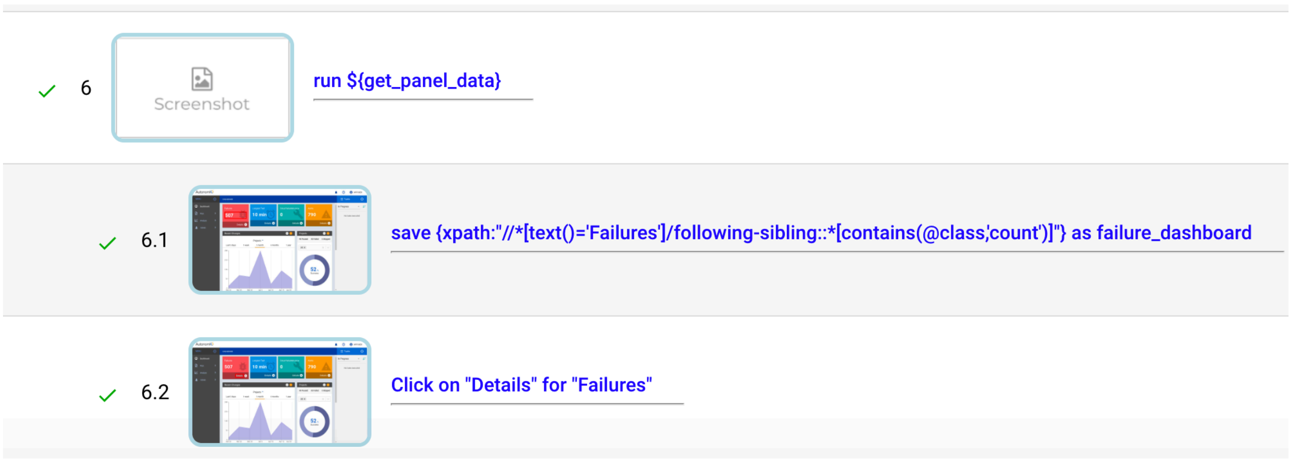
Modular approach - Flows

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.

Note: Presently support is for adding and cloning a flow, User cannot delete a created flow. User can clone and edit the steps if needed


✓ 6 Screenshot run ${get_panel_data}

✓ 6.1 save {xpath:"//*[text()='Failures']/following-sibling::*[contains(@class,'count')]"} as failure_dashboard

✓ 6.2 Click on "Details" for "Failures"

Waits

Wait for given time

Wait for {number} seconds/secs

- **"wait for 3 seconds"**
- **"wait for 5 secs"**

Wait until a condition is met

Wait until [element] is visible/exists

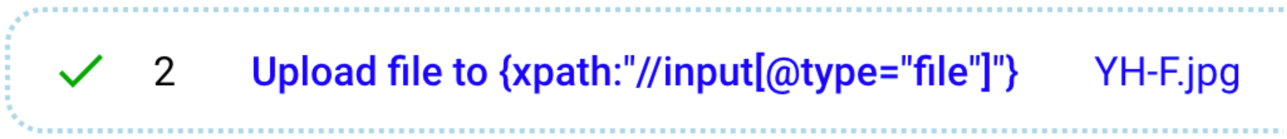
- **"wait until username is visible"**
- **"wait until username is exists"**

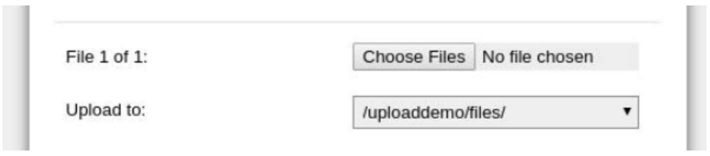
Upload

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

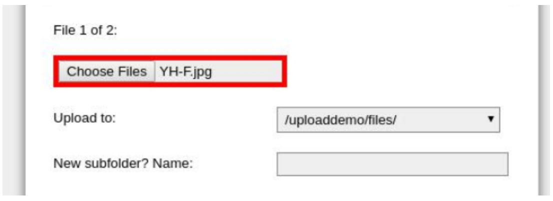
Upload file to [field]

- **" upload file to 'select xls,xlsx or csv file' "**


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


File 1 of 1: Choose Files No file chosen

Upload to: /uploaddemo/files/


File 1 of 2:

Choose Files YH-F.jpg

Upload to: /uploaddemo/files/

New subfolder? Name:

Date Support

Get Todays date in a given format

This should be provided in the data tab.

{today, <format>}

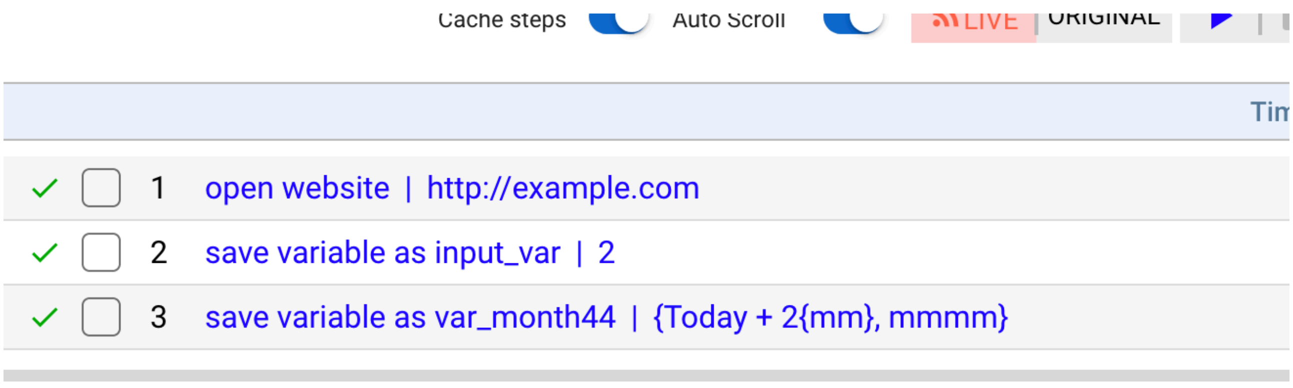
- **"Enter Date" + data tab should have {today, dd/mm/yy}**
- **"Enter Date" + data tab should have {today, mm/dd/yy}**
- **"Enter Date" + data tab should have {today, yy/mm/dd}**
-

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

Note: used format is not case-sensitive

Here var_month44 store a month which is 2 months more than the current month



1 open website | http://example.com
2 save variable as input_var | 2
3 save variable as var_month44 | {Today + 2{mm}, mmmm}

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. If the element/xpath does not point to any text

save {xpath:"xpath_value"} as [variable name]

save [element text] as [variable name]

Other Misc

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 my*project*name" * how to use that variable in the following Instruction? "Enter project name" + data tab should have "${my*project*name}" 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.

Random number support

In the data tab, "#{AnomTest\d\d\d\d\d\d\d}" is transformed into below AnomTest1657483

(not precisely correct, 'cause It's randomly generated every time)

Actions with Proximity [Next to/before/after]

- **"click on edit next (to) Donald"**
- **"click on radio before <some text>"**
- **"enter user name after <some 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/sh/java{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 sh(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 me numerous, they have to be comma seperated
- 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 seperate, 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.getElementsByName("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.getElementsByName("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(sum);} with ${num} returning ${sum}**

Luhn Algorithm or Modulus 10 Algortithm

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.

2. 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"

3. 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

%I - 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 Columns 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 I"**
- **"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

Dates

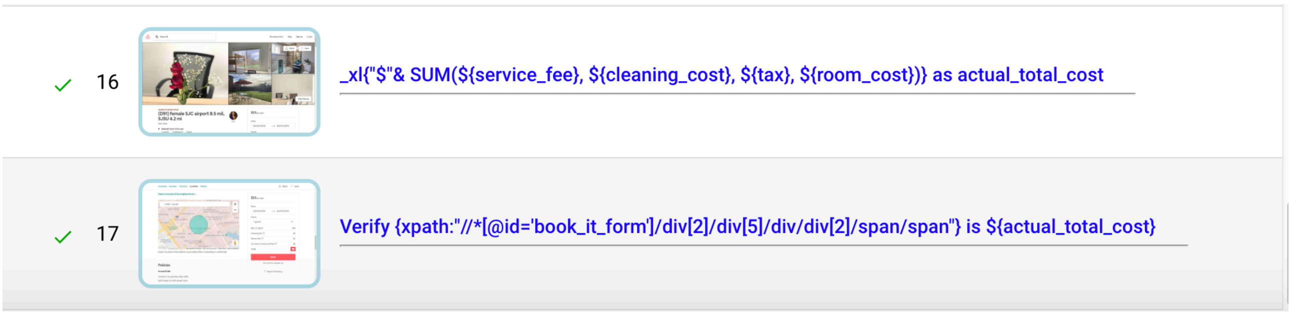
- _xl{TEXT(TODAY(), ""mm/dd/yyyy"")} as var_date
- _xl{TEXT(TODAY(), ""mmm-ddd"")} as var_date2
- _xl{TEXT(TODAY(), ""mmmm dddd"")} as var_date3
-

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 beginning of the function



✓	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}

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",
```