

Common

Library version: RENAT 0.1.9
Library scope: global
Named arguments: supported

Introduction

Common library for RENAT

It loads config files and create necessary variables. The file should be the 1st library included from any test case.

Table of Contents

- [Configuration file](#)
- [Variables](#)
- [Shortcuts](#)
- [Keywords](#)

Configuration file

Global configuration

There are 2 important configuration files. The global configuration files (aka master files) include device information, authentication etc that are used for all the test cases in the suite. The local configuration file `local.yaml` includes information about nodes, tester ports etc. that are used in a specific test case.

At the beginning, the module makes a local copy the master files and initialize necessary variables.

The RENAT framework utilized the YAML format for its configurations file.

The master files folder is defined by `renat-master-folder` in `$RENAT_PATH/config/config.yaml`. Usually, users do not need to modify the master files. The most common case is when new device is deployed, the `device.yaml` need to be update so that device could be used in the test cases.

1. device.yaml: contains global device information

Each device information is store under `device` block and has the following format:

```
<node_name>
  type:      <device type>
  description: <any useful description>
  ip:        <the IPv4 address of the device>
```

Where `<node_name>` is the name of the device. It could be the name of a switch, router or a web appliance box and should be uniq between the devices. `<description>` is any useful information and `<ip>` is the IP that RENAT uses to access the device.

`<type>` is important because it will be used as the ky of the `access_template` in template file. Usually users do not need to invent a new type but should use the existed type. When a new platform need to be supported, a new type will be introduced with the correspon template and authentication information.

Samples:

```
device:
  apollo:
    type: ssh-host
    description: main server
    ip: 10.128.3.101
  artermis:
    type: ssh-host
    description: second server
    ip: 10.128.3.91
  vmx11:
    type: juniper
    description: r1
    ip: 10.128.64.11
  vmx12:
    type: juniper
    description: r2
    ip: 10.128.64.12
```

2. template.yaml: contains device template information

The template file contains information about how to access to the device and how it should polling information (SNMP only for now). Each template has the following format:

`<type>`: access: `<ssh or telnet>` auth: `<plaint-text or public-key>` profile: `<authentication profile name>` prompt: `<a regular expression for the PROMPT of the CLI device>` (optional) login_prompt: `<a login PROMPT for CLI device>` (optional) password_prompt: `<a PROMPT for asking password of CLI device>` (optional) append: `<a pharase to append automatically for every CLI command that executes>` on this device (optional) init: `<an array of command that will be executed automatically after a sucessful login of CLI device>` (optional)

Note: Becareful about the prompt field. Usually RENAT will wait until it could see the prompt in its output. A wrong prompt will halt the system until it is timed out.

Samples:

```
access-template:
  ssh-host:
    access: ssh
    auth: public-key
    profile: default
    prompt: \$
    append:
    init: unalias -a
  juniper:
    access: telnet
    auth: plain-text
```

```

profile: default
prompt: "(#|>) "
append: ' | no-more'
init:
cisco:
  access: ssh
  auth: plain-text
  profile: default
  prompt: "\\@.*(#|>) "
  append:
  init:
snmp-template:
  juniper:
    mib: ./mib-Juniper.json
    community: public
    poller: renat
  cisco:
    mib: ./mib-Cisco.json
    community: public

```

3. auth.yaml: contains authentication information

The file contains authentication information that system uses when access to a device. Each authentication type has following format:

```

plain-text
<profile>
  user: <user name>
  password: <password>

```

or

```

public-key:
<profile>:
  user: <user name>
  key: <public key path>

```

Where <profile> is the name of the authentication profile specified in the `access template` of the device

Sample:

```

auth:
  plain-text:
    default:
      user: user
      pass: nttXXX
    flets:
      user: user
      pass: lpcoXXXX
    arbor:
      user: admin
      pass: nttXXX

  public-key: # for Public Key authentication
    default:
      user: robot
      key: /home/user/.ssh/robot_id_rsa
    test:
      user: jenkins
      key: /var/lib/jenkins/.ssh/id_rsa

```

Local Configuration

Local configuration (aka `local.yaml`) was used by a test case of its sub test cases. Test cases could includes several test cases (the sub level is not limited). The local configuration is defined by `local.yaml` in the `config` folder of each test case. If a test case does not has the `local.yaml` in its `config` folder, it will use the `local.yaml` file in its parent test case and so on. This will help users to share the test information for related test case without having the same `local.yaml` for each test case (**Note:** this feature is enabled from RENAT 0.1.4). The `local.yaml` that is really used for the test is called `active local.yaml`.

When user used the wizard `item.sh` to create a new test case, they have the ability to crete new `local.yaml` or not. `local.yaml` could be edited and inserted new information later to hold more informations for the test case.

When a test is run, it will display its current active `local.yaml`

The local configuration file of each test item is stored in the `config` folder of the item as `'local.yaml'`

Usually the `local.yaml` has following parts:

- CLI node information: started by `node` keyword
- WEB node information: started by `webapp` keyword
- Tester device information: started by `tester` keyword
- Default information: automatically created and started by `default` keyword
- And other neccessary information for the test by yaml format

Sample:

```

# CLI node
node:
  vmx11:
    device: vmx11
    snmp_polling: yes
  vmx12:
    device: vmx11
    snmp_polling: yes
  apollo:
    device: vmx11
    snmp_polling: yes

```

```
# web application information
webapp:
  arbor-sp-a:
    device: arbor-sp-a
    proxy:
      http: 10.128.8.210:8080
      ssl: 10.128.8.210:8080
      socks: 10.128.8.210:8080

# Tester information
tester:
  tester01:
    type: ixnet
    ip: 10.128.32.70
    config: vmx_20161129.ixncfg

# Other user information|
port-mapping:
  uplink01:
    device: vmx11
    port: ge-0/0/0
  downlink01:
    device: vmx12
    port: ge-0/0/2

# Default information
default:
  ignore_dead_node: yes
  terminal:
    width: 80
    height: 32
  result_folder: result
```

Variables

The module automatically create `GLOBAL` & `LOCAL` variable for other libraries. It also creates global list variables `GLOBAL`, `LOCAL` and `NODE` that could be accessed from Robot Framework test cases.

The `GLOBAL` variable holds all information defined by the master files and `LOCAL` variable holds all variables defined by active `local.yaml`. And `NODE` is a list that hold all active nodes defined in the `local.yaml`.

Users could access to the information of a key in `local.yaml` by `$(LOCAL['key'])`, information of a node by `$(LOCAL['node']['vmx11'])` or simply `$NODE['vmx']`. When a keyword need a list of current node, `@{NODE}` could be used.

Notes: By default, RENAT will stop and raise an exception if connection to a node is failed. But if `ignore_dead_node` is defined as `yes` (default) is the current active `local.yaml`, RENAT will omit an warning but keep running the test and remove the node from its active node list.

Shortcuts

Change Mod · Cleanup Result · Convert Html To Pdf · Count Keyword · Count Keyword Line · Count Match Regexp · Create Sequence · Csv Concat · Csv Merge · Csv Select · Diff File · Err · Error Line Should Not Be Bigger Than · Error Should Not Be Bigger Than · Explicit Run · File Md5 · Fold Str · Follow Syslog And Trap · Get Config Path · Get Config Value · Get File Without Error · Get Item Config Path · Get Item Name · Get Myid · Get Renat Path · Get Result Folder · Get Result Path · Get Test Device · Is Stable · Keyword Line Should Not Be Bigger Than · Keyword Should Not Be Bigger Than · Load Plugin · Log · Log To Console · Loop For Node Tag · Md 5 · Merge Files · Mib For Node · Node With Attr · Node With Tag · Node Without Tag · Pause · Ping Until Ok · Random Name · Random Number · Renat Version · Set Multi Item Variable · Set Result Folder · Slack · Str 2 Seq · Version

Keywords

Keyword	Arguments	Documentation
Change Mod	<i>name, mod, relative=False</i>	Changes file mod, likes Unix chmod <code>mod</code> is a string specifying the privilege mode <code>relative</code> is <code>False</code> or <code>True</code> Examples: <div>Common.Change Mod tmp 0775</div>
Cleanup Result	<i>ignore=^(log.html output.xml report.html)\$</i>	Cleans up the result folder Deletes all files in current active folder that does not match the <code>ignore</code> expression and are older than the time the test has started. Note: The keyword only removes files but not folders
Convert Html To Pdf	<i>html_file, pdf_file</i>	Converts html file to pdf file
Count Keyword	<i>keyword, *pattern_list</i>	Count the keyword in files. Keyword is not case-sensitive
Count Keyword Line	<i>keyword, *pattern_list</i>	Count the number of lines contains the <code>keyword</code> Notes: Keyword is matched partially. For example, <code>error</code> or <code>errorXXX</code> will be matched by <code>error</code> keyword.
Count Match Regexp	<i>regexp, *pattern_list</i>	Count the number of <code>regex</code> found in <code>pattern_list</code> Examples: <div>\$(err_num)= Count Match RegExp .*error.* result/*.csv result/*.txt</div>
Create Sequence	<i>start, end, interval, option=float</i>	Creates a list with number from <code>start</code> to <code>end</code> with <code>interval</code> Example: <div></div>

		<div>@{list}= Create Sequence 10 15 0.5</div> <div>will create a list of [11.0, 11.5, 12.0, 12.5, 13.0, 13.5, 14.0, 14.5]</div>																																
Csv Concat	<div>src_pattern, dst_name,</div> <div>input_header=None, result_header=True</div>	<div>Concatinates CSV files vertically If the CSV files has header, set has_header to \${TRUE}</div> <div>Examples:</div> <table><tr><td>Common.CSV Concat</td><td>config/data0[3,4].csv</td><td>result/result2.csv</td><td></td></tr><tr><td>Common.CSV Concat</td><td>config/data0[3,4].csv</td><td>result/result2.csv</td><td>has_header=\${TRUE}</td></tr></table>	Common. CSV Concat	config/data0[3,4].csv	result/result2.csv		Common. CSV Concat	config/data0[3,4].csv	result/result2.csv	has_header=\${TRUE}																								
Common. CSV Concat	config/data0[3,4].csv	result/result2.csv																																
Common. CSV Concat	config/data0[3,4].csv	result/result2.csv	has_header=\${TRUE}																															
Csv Merge	<div>src_pattern, dst_name,</div> <div>input_header=None, key=0,</div> <div>select_column=:, result_header=True</div>	<div>Merges all CSV files horizontally by key key from src_pattern</div> <div>input_header defines whether the input files has header row or not. If input_header is \${NULL}, the keyword assume that input files have no header and automatically define columns name. When input_header is not null (default is zero), the row define by input_header will be used as header and data is counted from the next row.</div> <div>select_column is a string that define the output columns and key is the column name that used to merge. When input_header is \${NULL}, select_column and key is the index of columns. Otherwise, they are column name.</div> <div>The result header (column names) is decided by result_header (True or False)</div> <div>The keyword returns False if no file is found by the pattern</div> <div>Examples:</div> <table><tr><td>Common.CSV Merge</td><td>config/data0[3,4].csv</td><td>result/result2.csv</td><td></td></tr><tr><td>Common.CSV Merge</td><td>config/data0[3,4].csv</td><td>result/result2.csv</td><td>input_header=0</td></tr><tr><td>Common.CSV Merge</td><td>src_pattern=\${RESULT_FOLDER}/balance*.csv</td><td>input_header=0</td><td></td></tr><tr><td>...</td><td>dst_name=\${RESULT_FOLDER}/result.csv</td><td>result_header=\${FALSE}</td><td></td></tr><tr><td>...</td><td>key=Stat Name</td><td>select_column=Valid Frames Rx.</td><td></td></tr><tr><td>Common.CSV Merge</td><td>src_pattern=\${RESULT_FOLDER}/balance*.csv</td><td>input_header=\${NULL}</td><td></td></tr><tr><td>...</td><td>dst_name=\${RESULT_FOLDER}/result.csv</td><td>result_header=\${FALSE}</td><td></td></tr><tr><td>...</td><td>key=0</td><td>select_column=5</td><td></td></tr></table>	Common. CSV Merge	config/data0[3,4].csv	result/result2.csv		Common. CSV Merge	config/data0[3,4].csv	result/result2.csv	input_header=0	Common. CSV Merge	src_pattern=\${RESULT_FOLDER}/balance*.csv	input_header=0		...	dst_name=\${RESULT_FOLDER}/result.csv	result_header=\${FALSE}		...	key=Stat Name	select_column=Valid Frames Rx.		Common. CSV Merge	src_pattern=\${RESULT_FOLDER}/balance*.csv	input_header=\${NULL}		...	dst_name=\${RESULT_FOLDER}/result.csv	result_header=\${FALSE}		...	key=0	select_column=5	
Common. CSV Merge	config/data0[3,4].csv	result/result2.csv																																
Common. CSV Merge	config/data0[3,4].csv	result/result2.csv	input_header=0																															
Common. CSV Merge	src_pattern=\${RESULT_FOLDER}/balance*.csv	input_header=0																																
...	dst_name=\${RESULT_FOLDER}/result.csv	result_header=\${FALSE}																																
...	key=Stat Name	select_column=Valid Frames Rx.																																
Common. CSV Merge	src_pattern=\${RESULT_FOLDER}/balance*.csv	input_header=\${NULL}																																
...	dst_name=\${RESULT_FOLDER}/result.csv	result_header=\${FALSE}																																
...	key=0	select_column=5																																
Csv Select	<div>src_file, dst_file, str_row=:, str_col=:</div> <div>has_header=None</div>	<div>Select part of the CSV file and write it to other file str_row and str_col are used to specify necessary rows and columns. They are using the same format with slice for Python list.</div> <div><div>▪ : and : means all rows and columns</div><div>▪ :2 and : means first 2 rows and all columns</div><div>▪ : and 1,2 means all rows and 2nd and 3rd columns</div><div>▪ 0:3 and 1 means 3 rows from the 1st one(0,1,2) and second column</div><div>▪ 0:5:2 and 1 means 3 rows(0,3,5) and second column</div></div> <div>Notes:</div> <div><div>▪ Rows and columns are indexed from zero</div><div>▪ When ':' is used, the string has format: <start>:<stop> or <start>:<stop>:<step> For convenience, ':' means all the data, 'x' means first 'x' data</div></div> <div>Examples:</div> <table><tr><td>CSV Select</td><td>result/data05.csv</td><td>result/result3.csv</td><td>0,1,2</td><td>0,1</td></tr><tr><td>CSV Select</td><td>result/data05.csv</td><td>result/result4.csv</td><td>:</td><td>0,1</td></tr><tr><td>CSV Select</td><td>result/data05.csv</td><td>result/result5.csv</td><td>:2</td><td>:</td></tr><tr><td>CSV Select</td><td>result/data05.csv</td><td>result/result6.csv</td><td>0:3</td><td>:</td></tr><tr><td>CSV Select</td><td>result/data05.csv</td><td>result/result7.csv</td><td>0:5:2</td><td>:</td></tr></table>	CSV Select	result/data05.csv	result/result3.csv	0,1,2	0,1	CSV Select	result/data05.csv	result/result4.csv	:	0,1	CSV Select	result/data05.csv	result/result5.csv	:2	:	CSV Select	result/data05.csv	result/result6.csv	0:3	:	CSV Select	result/data05.csv	result/result7.csv	0:5:2	:							
CSV Select	result/data05.csv	result/result3.csv	0,1,2	0,1																														
CSV Select	result/data05.csv	result/result4.csv	:	0,1																														
CSV Select	result/data05.csv	result/result5.csv	:2	:																														
CSV Select	result/data05.csv	result/result6.csv	0:3	:																														
CSV Select	result/data05.csv	result/result7.csv	0:5:2	:																														
Diff File	<div>path1, path2, newline=True</div>	<div>Shows difference between files</div> <div>Returns the diff result (multi lines) path1 , path2 are absolute paths.</div>																																
Err	<div>msg</div>	<div>Prints error msg to console</div>																																
Error Line Should Not Be Bigger Than	<div>num, *pattern_list</div>	<div>Checks whether the number of lines that contains error be less than a number</div>																																
Error Should Not Be Bigger Than	<div>num, *pattern_list</div>	<div>Checks whether the number of error be less than a number</div>																																
Explicit Run		<div>skip the test case if global_variable RUN_ME is not defined</div> <div>Sample scenario:</div> <table><tr><td>00. Cabling</td><td></td></tr><tr><td>Common.Explicit Run</td><td></td></tr><tr><td>Log To Console</td><td>cabling...</td></tr></table> <div>run.sh will bypass 00. Cabling by default. In other to run this test case \${FORCE} needs declared globally run.sh -X -v FORCE</div>	00. Cabling		Common. Explicit Run		Log To Console	cabling...																										
00. Cabling																																		
Common. Explicit Run																																		
Log To Console	cabling...																																	
File Md5	<div>path</div>	<div>Returns MD5 hash of a file</div> <div>path is an absolute path</div>																																
Fold Str	<div>str</div>	<div>Folds a string by adding Non-Width-Space char (0x200b) at 6th char</div>																																
Follow Syslog And Trap	<div>pattern, log_file_name=syslog-trap.log,</div> <div>delay_str=1s</div>	<div>Pauses the execution and wait for the pattern is matched if the file log_file_name located in the current result folder.</div>																																

		<p>By default the <code>log_file_name</code> is <code>./result/syslog-trap.log</code> which is created by Follow Syslog and Trap keyword.</p> <p>The keyword should be in tests between <i>Follow Syslog adn Trap Start</i> and <i>Follow Syslog and Trap Stop</i> keywords.</p>																
Get Config Path		Returns absolute path of RENAT config folder path																
Get Config Value	<code>key</code> , <code>base=default</code>	Returns value of a key for renat configuration with this other LOCAL[base][key] > GLOBAL[base][key] > None																
Get File Without Error	<code>file_path</code>	Get content of the file and return null string if the file does not exist																
Get Item Config Path		Returns absolute path of current item config folder																
Get Item Name		Returns the name of the running item																
Get Myid																		
Get Renat Path		Returns the absolute path of RENAT folder																
Get Result Folder		Returns current result folder name. Default is <code>result</code> in current test case. Note: the keyword only returns the <code>name</code> of the result folder not its absloue path.																
Get Result Path		Returns absolute path of the current result folder																
Get Test Device		Return a list of all test device that is used in this test Notes: Device number could less than node number																
Is Stable	<code>seq</code> , <code>threshold</code> , <code>percentile=90</code>	Checks if the value sequence is stable or not																
Keyword Line Should Not Be Bigger Than	<code>num</code> , <code>keyword</code> , <code>*pattern_list</code>	Checks whether the number of line containing the keyword be less than a number																
Keyword Should Not Be Bigger Than	<code>num</code> , <code>keyword</code> , <code>*pattern_list</code>	Checks whether the number of keyword be less than a number																
Load Plugin		Load plugin in renat/plugin folder																
Log	<code>msg</code> , <code>level=1</code>	<p>Logs <code>msg</code> to the current log file (not console)</p> <p>The <code>msg</code> will logged only if the level is bigger than the global level <code>\$(DEBUG)</code> which could be defined at runtime. If <code>\$(DEBUG)</code> is not defined, it will be considered as the default <code>level</code> as 1.</p> <p>Examples:</p> <table><tr><td>Common.Log</td><td>XXX</td><td># this always be logged</td><td></td></tr><tr><td>Common.Log</td><td>AAA</td><td>level=2</td><td># this will not be logged with common run.sh</td></tr><tr><td>Common.Log</td><td>BBB</td><td>level=2</td><td># ./run.sh -v DEBUG:2 will log the message</td></tr></table> <p>Notes: For common use</p> <ul style="list-style-type: none">■ level 1: is default■ level 2: is debug mode■ level 3: is very informative mode	Common. Log	XXX	# this always be logged		Common. Log	AAA	level=2	# this will not be logged with common run.sh	Common. Log	BBB	level=2	# ./run.sh -v DEBUG:2 will log the message				
Common. Log	XXX	# this always be logged																
Common. Log	AAA	level=2	# this will not be logged with common run.sh															
Common. Log	BBB	level=2	# ./run.sh -v DEBUG:2 will log the message															
Log To Console	<code>msg</code> , <code>level=1</code>	<p>Logs a message to console</p> <p>See Common.<i>Print</i> for more details about debug level</p>																
Loop For Node Tag	<code>var</code> , <code>tags</code> , <code>*keywords</code>	<p>Repeatedly executes RF <code>keyword</code> for nodes that has tag <code>tags</code></p> <p>multi tags are separated by <code>:</code> keywords has same meaning with <code>keywords</code> used by <i>Run Keywords</i> of RobotFramework (<code>keyword</code> and its arguments are separated by <code>AND</code> with the others.</p> <p>Example:</p> <table><tr><td>Loop For Node Tag</td><td><code>\\${node}</code></td><td>tag1</td><td></td></tr><tr><td>...</td><td><i>Switch</i></td><td><code>\\${node}</code></td><td>AND</td></tr><tr><td>...</td><td><i>Cmd</i></td><td>show system user</td><td>AND</td></tr><tr><td>...</td><td><i>Cmd</i></td><td>show system uptime</td><td></td></tr></table> <p>Note: <code>\$</code> in variable name must be escaped</p>	Loop For Node Tag	<code>\\${node}</code>	tag1		...	<i>Switch</i>	<code>\\${node}</code>	AND	...	<i>Cmd</i>	show system user	AND	...	<i>Cmd</i>	show system uptime	
Loop For Node Tag	<code>\\${node}</code>	tag1																
...	<i>Switch</i>	<code>\\${node}</code>	AND															
...	<i>Cmd</i>	show system user	AND															
...	<i>Cmd</i>	show system uptime																
Md 5	<code>str</code>	Returns MD5 hash of a string																
Merge Files	<code>path_name</code> , <code>file_name</code>	<p>Merges all the text files defined by <code>path_name</code> to <code>file_name</code></p> <p>Example:</p> <table><tr><td>Merge Files</td><td><code>./result/*.csv</code></td><td><code>./result/test.csv</code></td></tr></table>	Merge Files	<code>./result/*.csv</code>	<code>./result/test.csv</code>													
Merge Files	<code>./result/*.csv</code>	<code>./result/test.csv</code>																
Mib For Node	<code>node</code>	<p>Returns the mib file name for this <code>node</code> mib file is define by <code>mib</code> keyword under the <code>node</code> in <code>local.yaml</code></p> <pre>... node: vmx11: device: vmx11 snmp_polling: yes mib: mib11.txt ...</pre> <p>Default value is defined by <code>mib</code> keyword from global <code>config/snmp-template.yaml</code> for the <code>type</code> of the node</p> <p>Example:</p> <table><tr><td><code>\\${mib}=</code></td><td>Common.MIB For Node</td><td><code>vmx11</code></td></tr></table>	<code>\\${mib}=</code>	Common. MIB For Node	<code>vmx11</code>													
<code>\\${mib}=</code>	Common. MIB For Node	<code>vmx11</code>																
Node With Attr	<code>attr_name</code> , <code>value</code>	Returns a list of nodes which have attribute <code>attr_name</code> with value <code>value</code>																

Node With Tag	*tag_list	Returns list of node or webapp from local.yaml that has ALL tags defined by tag_list Tag was defined like this in local.yaml <pre>vmx11: device: vmx11 snmp_polling: yes tag: - tag1 - tag2</pre> Examples: <table><tr><td>\$(test3)=</td><td>Common.</td><td>Node With Tag</td><td>tag1</td><td>tag3</td></tr></table>	\$(test3)=	Common.	Node With Tag	tag1	tag3											
\$(test3)=	Common.	Node With Tag	tag1	tag3														
Node Without Tag	*tag_list	Returns list of node from local.yaml that does not has ANY tags defined by tag_list Tag was defined like this in local.yaml <pre>vmx11: device: vmx11 snmp_polling: yes tag: - tag1 - tag2</pre> Examples: <table><tr><td>\$(test3)=</td><td>Common.</td><td>Node Without Tag</td><td>tag1</td><td>tag3</td></tr></table>	\$(test3)=	Common.	Node Without Tag	tag1	tag3											
\$(test3)=	Common.	Node Without Tag	tag1	tag3														
Pause	msg=, time_out=3h, error_on_timeout=True, default_input=	Displays the message msg and pauses the test execution and wait for user input In case of error_on_timeout is True(default), the keyword will raise an error when timeout occurs. Otherwise, it will continue the test. Notes: If the variable \${RENAT_BATCH} was defined, the keyword will print out the message and keeps running without pausing. Examples: <table><tr><td>Common.</td><td>Pause</td><td>Waiting...</td><td>10s</td><td>error_on_timeout=\${TRUE}</td><td>default input</td></tr><tr><td>Common.</td><td>Pause</td><td>Waiting...</td><td>10s</td><td></td><td></td></tr></table>	Common.	Pause	Waiting...	10s	error_on_timeout=\${TRUE}	default input	Common.	Pause	Waiting...	10s						
Common.	Pause	Waiting...	10s	error_on_timeout=\${TRUE}	default input													
Common.	Pause	Waiting...	10s															
Ping Until Ok	node, wait_str=5s, extra=-c 3	Ping a node until it gets response. Then wait for more wait_str Default extra option is -c 3																
Random Name	base, a=0, b=99	Returns a random name by a base and a random number between [a,b] Example: <table><tr><td>\$(FOLDER)=</td><td>Random Name</td><td>capture_%05d</td><td>0</td><td>99</td></tr></table>	\$(FOLDER)=	Random Name	capture_%05d	0	99											
\$(FOLDER)=	Random Name	capture_%05d	0	99														
Random Number	a=0, b=99	Returns a random number between [a,b]																
Renat Version		Returns RENAT version string																
Set Multi Item Variable	*vars	Set multiple variables to be suite variable at the same time Suite variables (or item variable) could be access anywhere in all the item scenario.																
Set Result Folder	folder	Sets the result folder to folder and return the old result folder. The result folder contains all output files from the test likes tester ouput, config file ... folder is a folder name that under current test case folder The system will create a new folder if it does not exist and set its mode to 0775 Note: Result folder should be set at the begining of the test. Changing result folder only has effect on up comming connection																
Slack	msg, channel=#automation_dev, user=renat, host=10.128.3.103:4713	Post a message to Slack																
Str 2 Seq	str_index, size	Returns a sequence from string format Samples: <table><tr><td>Str2Seq</td><td>::</td><td>5</td><td># (0,1,2,3,4)</td></tr><tr><td>Str2Seq</td><td>:2</td><td>5</td><td># (0,1)</td></tr><tr><td>Str2Seq</td><td>1:3</td><td>5</td><td># (1,2)</td></tr><tr><td>Str2Seq</td><td>0:5:2</td><td>5</td><td># (0,2,4)</td></tr></table>	Str2Seq	::	5	# (0,1,2,3,4)	Str2Seq	:2	5	# (0,1)	Str2Seq	1:3	5	# (1,2)	Str2Seq	0:5:2	5	# (0,2,4)
Str2Seq	::	5	# (0,1,2,3,4)															
Str2Seq	:2	5	# (0,1)															
Str2Seq	1:3	5	# (1,2)															
Str2Seq	0:5:2	5	# (0,2,4)															
Version		Returns the current version of RENAT																

