

# Common

**Library version:** RENAT 0.1.10  
**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 · Close Display · Convert Html To Pdf · Count Keyword · Count Keyword Line · Count Match Regexp · Create Sequence · Csv Add · Csv Concat · Csv Create · 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 Csv · 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 · Start Display · Str 2 Seq · Version · Wait

## Keywords

Keyword	Arguments	Documentation
<b>Change Mod</b>	<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.<a href="#">Change Mod</a> tmp 0775</div>
<b>Cleanup Result</b>	<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.  <b>Note:</b> The keyword only removes files but not folders
<b>Close Display</b>		Closes the opened display
<b>Convert Html To Pdf</b>	<i>html_file, pdf_file</i>	Converts html file to pdf file
<b>Count Keyword</b>	<i>keyword, *pattern_list</i>	Count the keyword in files. Keyword is not case-sensitive
<b>Count Keyword Line</b>	<i>keyword, *pattern_list</i>	Count the number of lines contains the <code>keyword</code>  <b>Notes:</b> Keyword is matched partially. For example, <code>error</code> or <code>errorXXX</code> will be matched by <code>error</code> keyword.
<b>Count Match Regexp</b>	<i>regexp, *pattern_list</i>	Count the number of <code>regex</code> found in <code>pattern_list</code>  Examples: <div>\$(err_num)= <a href="#">Count Match RegExp</a> *.error.* result/*.csv result/*.txt</div>
<b>Create Sequence</b>	<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>@{list}= <a href="#">Create Sequence</a> 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 Add	pathname, *items	<div>Add more data define by a list <i>items</i> to a existed CSV file</div> <div><b>Note::</b> do not check the consistency between item's number and header's number</div>																																
Csv Concat	src_pattern, dst_name, input_header=None, result_header=True	<div>Concatinates CSV files vertically If the CSV files has header, set <code>has_header</code> to <code>\$(TRUE)</code></div> <div>Examples:</div> <table><tr><td>Common.<a href="#">CSV Concat</a></td><td>config/data0[3,4].csv</td><td>result/result2.csv</td><td></td></tr><tr><td>Common.<a href="#">CSV Concat</a></td><td>config/data0[3,4].csv</td><td>result/result2.csv</td><td>has_header=\$(TRUE)</td></tr></table>	Common. <a href="#">CSV Concat</a>	config/data0[3,4].csv	result/result2.csv		Common. <a href="#">CSV Concat</a>	config/data0[3,4].csv	result/result2.csv	has_header=\$(TRUE)																								
Common. <a href="#">CSV Concat</a>	config/data0[3,4].csv	result/result2.csv																																
Common. <a href="#">CSV Concat</a>	config/data0[3,4].csv	result/result2.csv	has_header=\$(TRUE)																															
Csv Create	pathname, *header	<div>Create a CSV file with headers defined by a list <i>header</i></div> <div>The CSV file is opend with <i>UTF-8</i> encoding mode</div>																																
Csv Merge	src_pattern, dst_name, input_header=None, key=0, select_column=., result_header=True	<div>Merges all CSV files horizontally by <i>key</i> key from <i>src_pattern</i></div> <div><i>input_header</i> defines whether the input files has header row or not. If <i>input_header</i> is <code>\$(NULL)</code>, the keyword assume that input files have no header and automatically define columns name. When <i>input_header</i> is not null (default is zero), the row define by <i>input_header</i> will be used as header and data is counted from the next row.</div> <div><i>select_column</i> is a string that define the output columns and <i>key</i> is the column name that used to merge. When <i>input_header</i> is <code>\$(NULL)</code>, <i>select_column</i> and <i>key</i> is the index of columns. Otherwise, they are <i>column name</i>.</div> <div>The result header (column names) is decided by <i>result_header</i> (<i>True</i> or <i>False</i>)</div> <div>The keyword returns <i>False</i> if no file is found by the pattern</div> <div>Examples:</div> <table><tr><td>Common.<a href="#">CSV Merge</a></td><td>config/data0[3,4].csv</td><td>result/result2.csv</td><td></td></tr><tr><td>Common.<a href="#">CSV Merge</a></td><td>config/data0[3,4].csv</td><td>result/result2.csv</td><td>input_header=0</td></tr><tr><td>Common.<a href="#">CSV Merge</a></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.<a href="#">CSV Merge</a></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. <a href="#">CSV Merge</a>	config/data0[3,4].csv	result/result2.csv		Common. <a href="#">CSV Merge</a>	config/data0[3,4].csv	result/result2.csv	input_header=0	Common. <a href="#">CSV Merge</a>	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. <a href="#">CSV Merge</a>	src_pattern=\$(RESULT_FOLDER)/balance*.csv	input_header=\$(NULL)		...	dst_name=\$(RESULT_FOLDER)/result.csv	result_header=\$(FALSE)		...	key=0	select_column=5	
Common. <a href="#">CSV Merge</a>	config/data0[3,4].csv	result/result2.csv																																
Common. <a href="#">CSV Merge</a>	config/data0[3,4].csv	result/result2.csv	input_header=0																															
Common. <a href="#">CSV Merge</a>	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. <a href="#">CSV Merge</a>	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	src_file, dst_file, str_row=., str_col=., has_header=None	<div>Select part of the CSV file and write it to other file <i>str_row</i> and <i>str_col</i> are used to specify necessary rows and columns. They are using the same format with slice for Python list.</div> <div><ul style="list-style-type: none"><li>▪ : and : means all rows and columns</li><li>▪ :2 and : means first 2 rows and all columns</li><li>▪ : and 1,2 means all rows and 2nd and 3rd columns</li><li>▪ 0:3 and 1 means 3 rows from the 1st one(0,1,2) and second column</li><li>▪ 0:5:2 and 1 means 3 rows(0,3,5) and second column</li></ul></div> <div><b>Notes:</b></div> <div><ul style="list-style-type: none"><li>▪ Rows and columns are indexed from zero</li><li>▪ When ':' is used, the string has format: &lt;start&gt;:&lt;stop&gt; or &lt;start&gt;:&lt;stop&gt;:&lt;step&gt; For convenience, ':' means all the data, 'x' means first 'x' data</li></ul></div> <div>Examples:</div> <table><tr><td><a href="#">CSV Select</a></td><td>result/data05.csv</td><td>result/result3.csv</td><td>0,1,2</td><td>0,1</td></tr><tr><td><a href="#">CSV Select</a></td><td>result/data05.csv</td><td>result/result4.csv</td><td>:</td><td>0,1</td></tr><tr><td><a href="#">CSV Select</a></td><td>result/data05.csv</td><td>result/result5.csv</td><td>:2</td><td>:</td></tr><tr><td><a href="#">CSV Select</a></td><td>result/data05.csv</td><td>result/result6.csv</td><td>0:3</td><td>:</td></tr><tr><td><a href="#">CSV Select</a></td><td>result/data05.csv</td><td>result/result7.csv</td><td>0:5:2</td><td>:</td></tr></table>	<a href="#">CSV Select</a>	result/data05.csv	result/result3.csv	0,1,2	0,1	<a href="#">CSV Select</a>	result/data05.csv	result/result4.csv	:	0,1	<a href="#">CSV Select</a>	result/data05.csv	result/result5.csv	:2	:	<a href="#">CSV Select</a>	result/data05.csv	result/result6.csv	0:3	:	<a href="#">CSV Select</a>	result/data05.csv	result/result7.csv	0:5:2	:							
<a href="#">CSV Select</a>	result/data05.csv	result/result3.csv	0,1,2	0,1																														
<a href="#">CSV Select</a>	result/data05.csv	result/result4.csv	:	0,1																														
<a href="#">CSV Select</a>	result/data05.csv	result/result5.csv	:2	:																														
<a href="#">CSV Select</a>	result/data05.csv	result/result6.csv	0:3	:																														
<a href="#">CSV Select</a>	result/data05.csv	result/result7.csv	0:5:2	:																														
Diff File	path1, path2, newline=True	<div>Shows difference between files</div> <div>Returns the diff result (multi lines) <i>path1</i> , <i>path2</i> are absolute paths.</div>																																
Err	msg	Prints error <i>msg</i> to console																																
Error Line Should Not Be Bigger Than	num, *pattern_list	Checks whether the number of lines that contains <i>error</i> be less than a number																																
Error Should Not Be Bigger Than	num, *pattern_list	Checks whether the number of <i>error</i> be less than a number																																
Explicit Run		<div>skip the test case if global_variable <i>RUN_ME</i> is not defined</div> <div>Sample scenario:</div> <table><tr><td>00. Cabling</td><td></td></tr><tr><td>Common.<a href="#">Explicit Run</a></td><td></td></tr><tr><td>Log To Console</td><td>cabling...</td></tr></table> <div><i>run.sh</i> will bypass 00. Cabling by default. In other to run this test case <i>\$(FORCE)</i> needs declared globally <i>run.sh -X -v FORCE</i></div>	00. Cabling		Common. <a href="#">Explicit Run</a>		Log To Console	cabling...																										
00. Cabling																																		
Common. <a href="#">Explicit Run</a>																																		
Log To Console	cabling...																																	

File Md5	path	Returns MD5 hash of a file  path is an absolute path																
Fold Str	str	Folds a string by adding Non-Width-Space char (0x200b) at 6th char																
Follow Syslog And Trap	pattern, log_file_name=syslog-trap.log, delay_str=1s	Pauses the execution and wait for the pattern is matched if the file log_file_name located in the current result folder.  By default the log_file_name is ./result/syslog-trap.log which is created by Follow Syslog and Trap keyword.  The keyword should be in tests between Follow Syslog adn Trap Start and Follow Syslog and Trap Stop keywords.																
Get Config Path		Returns absolute path of RENAT config folder path																
Get Config Value	key, base=default	Returns value of a key for renat configuration with this other LOCAL[base][key] > GLOBAL[base][key] > None																
Get File Without Error	file_path	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 result in current test case.  Note: the keyword only returns the name of the result folder not its absoulue 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	seq, threshold, percentile=90	Checks if the value sequence is stable or not																
Keyword Line Should Not Be Bigger Than	num, keyword, *pattern_list	Checks whether the number of line containing the keyword be less than a number																
Keyword Should Not Be Bigger Than	num, keyword, *pattern_list	Checks whether the number of keyword be less than a number																
Load Plugin		Load plugin in renat/plugin folder																
Log	msg, level=1	Logs msg to the current log file (not console)  The msg will logged only if the level is bigger than the global level \${DEBUG} which could be defined at runtime. If \${DEBUG} is not defined, it will be considered as the default level as 1.  Examples: <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> Notes: For common use <ul style="list-style-type: none"><li>level 1: is default</li><li>level 2: is debug mode</li><li>level 3: is very informative mode</li></ul>	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 Csv	csv_file, index=False, border=0	Logs a content of csv_file into default log.html  index, border are table attributes																
Log To Console	msg, level=1	Logs a message to console  See Common.Print for more details about debug level																
Loop For Node Tag	var, tags, *keywords	Repeatedly executes RF keyword for nodes that has tag tags  multi tags are separated by : keywords has same meaning with keywords used by Run Keywords of RobotFramework ( keyword and its arguments are separated by AND with the others.  Example: <table><tr><td>Loop For Node Tag</td><td>\${node}</td><td>tag1</td><td></td></tr><tr><td>...</td><td>Switch</td><td>\${node}</td><td>AND</td></tr><tr><td>...</td><td>Cmd</td><td>show system user</td><td>AND</td></tr><tr><td>...</td><td>Cmd</td><td>show system uptime</td><td></td></tr></table> Note: \$ in variable name must be escaped	Loop For Node Tag	\${node}	tag1		...	Switch	\${node}	AND	...	Cmd	show system user	AND	...	Cmd	show system uptime	
Loop For Node Tag	\${node}	tag1																
...	Switch	\${node}	AND															
...	Cmd	show system user	AND															
...	Cmd	show system uptime																
Md 5	str	Returns MD5 hash of a string																
Merge Files	path_name, file_name	Merges all the text files defined by path_name to file_name  Example: <table><tr><td>Merge Files</td><td>./result/*.csv</td><td>./result/test.csv</td></tr></table>	Merge Files	./result/*.csv	./result/test.csv													
Merge Files	./result/*.csv	./result/test.csv																
Mib For Node	node	Returns the mib file name for this node mib file is define by mib keyword under the node in local.yaml  <pre>... node: vmx11:</pre>																

		<pre>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> <pre>\$(mib)= Common.MIB For Node vmx11</pre>																
Node With Attr	<code>attr_name, value</code>	Returns a list of nodes which have attribute <code>attr_name</code> with value <code>value</code>																
Node With Tag	<code>*tag_list</code>	<p>Returns list of <code>node</code> or <code>webapp</code> from <code>local.yaml</code> that has <b>ALL</b> tags defined by <code>tag_list</code></p> <p>Tag was defined like this in <code>local.yaml</code></p> <pre>vmx11:   device: vmx11   snmp_polling: yes   tag:     - tag1     - tag2</pre> <p>Examples:</p> <pre>\$(test3)= Common.Node With Tag tag1 tag3</pre>																
Node Without Tag	<code>*tag_list</code>	<p>Returns list of <code>node</code> from <code>local.yaml</code> that <b>does not has ANY</b> tags defined by <code>tag_list</code></p> <p>Tag was defined like this in <code>local.yaml</code></p> <pre>vmx11:   device: vmx11   snmp_polling: yes   tag:     - tag1     - tag2</pre> <p>Examples:</p> <pre>\$(test3)= Common.Node Without Tag tag1 tag3</pre>																
Pause	<code>msg=, time_out=3h, error_on_timeout=True, default_input=</code>	<p>Displays the message <code>msg</code> and pauses the test execution and wait for user input</p> <p>In case of <code>error_on_timeout</code> is <code>True</code>(default), the keyword will raise an error when timeout occurs. Otherwise, it will continue the test.</p> <p><b>Notes:</b> If the variable <code>\$(RENAT_BATCH)</code> was defined, the keyword will print out the message and keeps running without pausing.</p> <p>Examples:</p> <pre>Common.Pause Waiting... 10s error_on_timeout=\${TRUE} default input Common.Pause Waiting... 10s</pre>																
Ping Until Ok	<code>node, wait_str=5s, extra=-c 3</code>	Ping a <code>node</code> until it gets response. Then wait for more <code>wait_str</code> Default <code>extra</code> option is <code>-c 3</code>																
Random Name	<code>base, a=0, b=99</code>	<p>Returns a random name by a <code>base</code> and a random number between [a,b]</p> <p>Example:</p> <pre>\$(FOLDER)= Random Name capture_%05d 0 99</pre>																
Random Number	<code>a=0, b=99</code>	Returns a random number between [a,b]																
Renat Version		Returns RENAT version string																
Set Multi Item Variable	<code>*vars</code>	<p>Set multiple variables to be <i>suite variable</i> at the same time</p> <p>Suite variables (or item variable) could be access anywhere in all the item scenario.</p>																
Set Result Folder	<code>folder</code>	<p>Sets the result folder to <code>folder</code> and return the old result folder. The result folder contains all output files from the test likes tester ouput, config file ...</p> <p><code>folder</code> is a folder name that under current test case folder</p> <p>The system will create a new folder if it does not exist and set its mode to <code>0775</code></p> <p><b>Note:</b> Result folder should be set at the begining of the test. Changing result folder only has effect on up comming connection</p>																
Slack	<code>msg, channel=#automation_dev, user=renat, host=10.128.3.103:4713</code>	Post a message to Slack																
Start Display		Starts a virtual display																
Str 2 Seq	<code>str_index, size</code>	<p>Returns a sequence from string format</p> <p>Samples:</p> <table><tr><td><code>Str2Seq</code></td><td>::</td><td>5</td><td># (0,1,2,3,4)</td></tr><tr><td><code>Str2Seq</code></td><td>:2</td><td>5</td><td># (0,1)</td></tr><tr><td><code>Str2Seq</code></td><td>1:3</td><td>5</td><td># (1,2)</td></tr><tr><td><code>Str2Seq</code></td><td>0:5:2</td><td>5</td><td># (0,2,4)</td></tr></table>	<code>Str2Seq</code>	::	5	# (0,1,2,3,4)	<code>Str2Seq</code>	:2	5	# (0,1)	<code>Str2Seq</code>	1:3	5	# (1,2)	<code>Str2Seq</code>	0:5:2	5	# (0,2,4)
<code>Str2Seq</code>	::	5	# (0,1,2,3,4)															
<code>Str2Seq</code>	:2	5	# (0,1)															
<code>Str2Seq</code>	1:3	5	# (1,2)															
<code>Str2Seq</code>	0:5:2	5	# (0,2,4)															
Version		Returns the current version of RENAT																
Wait	<code>wait_time, size=10</code>	<p>Waits for <code>wait-time</code> and display the proress bar</p> <p><code>wait_time</code> used RF <i>DateTime</i> format.</p> <p>Examples:</p> <pre>Common.Wait wait_time=30s size=10</pre>																

