

# Common

**Library version:** RENAT 0.1.14  
**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 a key 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.

Examples:

```
device:
  apollo:
    type: ssh-host
    description: main server
    ip: 10.128.3.101
  artemis:
    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, telnet or jump
  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 phrase 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)
  target:      another type (mandatory in case of access is jump)
```

**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.

Examples:

```
access-template:
  # template for an oridnary UNIX server access by SSH
  ssh-host:
    access: ssh
    auth: public-key
```

```

profile: default
prompt: \$
append:
init: unalias -a
# template for a Juniper router
juniper:
  access: telnet
  auth: plain-text
  profile: default
  prompt: "(#|>)"
  append: ' | no-more'
  init:
# template for a Cisco router
cisco:
  access: ssh
  auth: plain-text
  profile: default
  prompt: "@.*{#|>}"
  append:
  init:
# template for a Juniper router access through a SmartCS console server
jump-smartcs:
  access: jump
  access_base: telnet
  auth: plain-text
  profile: default
  prompt: "tty.*>"
  password_prompt: "Password:"
  target: juniper
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

Example:

```

auth:
  plain-text:
    default:
      user: user
      pass: xxxxxx
    flets:
      user: user
      pass: xxxxxx
  arbor:
    user: admin
    pass: xxxxxx

  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 necessary information for the test by yaml format

Example:

```
# 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 local information specific for this case
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 · Convert Xml · Count Keyword · Count Keyword Line · Count Match Regexp · Create Sequence · Csv Add · Csv Concat · Csv Create · Csv Merge · Csv Select · Current Usergroup · Current Username · 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 Multi Lines · Get Myid · Get Renat Path · Get Result Folder · Get Result Path · Get Test Device · Get Tmp Path · 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 · Screenshot · Send · Set Multi Item Variable · Set Result Folder · Slack · Start Display · Stop Next Run · Str 2 Seq · Version · Wait

## Keywords

Keyword	Arguments	Documentation
<b>Change Mod</b>	<i>name, mod, relative=True</i>	Changes file mod, likes Unix <code>chmod</code>  mod is a string specifying the privilege mode relative 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.																																
Close Display		<b>Note:</b> The keyword only removes files but not folders Closes the opened display																																
Convert Html To Pdf	html_file, pdf_file	Converts html file to pdf file																																
Convert Xml	style, src, dst	Converts XML by using XLS stylesheet  Predefined stylesheets are store in <i>tools/xls</i> under current active RENAT folder  Parameters: <ul style="list-style-type: none"><li>■ style: path to stylesheet</li><li>■ src: path to the XML source</li><li>■ dst: path to the output file</li></ul>																																
Count Keyword	keyword, *pattern_list	Count the keyword in files. Keyword is not case-sensitive																																
Count Keyword Line	keyword, *pattern_list	Count the number of lines contains the keyword  <b>Notes:</b> Keyword is matched partially. For example, <code>error</code> or <code>errorXXX</code> will be matched by <code>error</code> keyword.																																
Count Match Regexp	regexp, *pattern_list	Count the number of <code>regex</code> found in <code>pattern_list</code>  Examples: <div><code>\$(err_num)= <a href="#">Count Match RegExp</a> .*error.* result/*.csv result/*.txt</code></div>																																
Create Sequence	start, end, interval, option=float	Creates a list with number from <code>start</code> to <code>end</code> with <code>interval</code>  Example: <div><code>@{list}= <a href="#">Create Sequence</a> 10 15 0.5</code></div> will create a list of <code>[11.0, 11.5, 12.0, 12.5, 13.0, 13.5, 14.0, 14.5]</code>																																
Csv Add	pathname, *items	Add more data define by a list <i>items</i> to a existed CSV file  <b>Note::</b> do not check the consistency between item's number and header's number																																
Csv Concat	src_pattern, dst_name, input_header=None, result_header=True	Concatinates CSV files vertically If the CSV files has header, set <code>has_header</code> to <code>\$(TRUE)</code>  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></div>	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	Create a CSV file with headers defined by a list <i>header</i>  The CSV file is opend with <i>UTF-8</i> encoding mode																																
Csv Merge	src_pattern, dst_name, input_header=None, key=0, select_column=:, result_header=True	Merges all CSV files <code>horizontally</code> by <code>key</code> <code>key</code> from <code>src_pattern</code>  <code>input_header</code> defines whether the input files has header row or not. If <code>input_header</code> is <code>\$(NULL)</code> , the keyword assume that input files have no header and automatically define columns name. When <code>input_header</code> is not null (default is zero), the row define by <code>input_header</code> will be used as header and data is counted from the next row.  <code>select_column</code> is a string that define the output columns and <code>key</code> is the column name that used to merge. When <code>input_header</code> is <code>\$(NULL)</code> , <code>select_column</code> and <code>key</code> is the index of columns. Otherwise, they are <i>column name</i> .  The result header (column names) is decided by <code>result_header</code> ( <i>True</i> or <i>False</i> )  The keyword returns <code>False</code> if no file is found by the pattern  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></div>	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	Select part of the CSV file and write it to other file <code>str_row</code> and <code>str_col</code> are used to specify necessary rows and columns. They are using the same format with slice for Python list. <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> <b>Notes:</b> <ul style="list-style-type: none"><li>■ Rows and columns are indexed from zero</li><li>■ When ':' is used, the string has format: <code>&lt;start&gt;:&lt;stop&gt;</code> or <code>&lt;start&gt;:&lt;stop&gt;:&lt;step&gt;</code> For convenience, ':' means all the data, 'x' means first 'x' data</li></ul> Examples: <div><a href="#">CSV Select</a> result/data05.csv result/result3.csv 0,1,2 0,1</div>																																

		<div><div>CSV Select</div><div>result/data05.csv</div><div>result/result4.csv</div><div>:</div><div>0.1</div></div> <div><div>CSV Select</div><div>result/data05.csv</div><div>result/result5.csv</div><div>:2</div><div>:</div></div> <div><div>CSV Select</div><div>result/data05.csv</div><div>result/result6.csv</div><div>0:3</div><div>:</div></div> <div><div>CSV Select</div><div>result/data05.csv</div><div>result/result7.csv</div><div>0:5:2</div><div>:</div></div>								
Current Usergroup		Returns current usergroup								
Current Username		Returns current username								
Diff File	path1, path2, newline=True	Shows difference between files Returns the diff result (multi lines) path1 , path2 are absolute paths.								
Err	msg	Prints error msg to console								
Error Line Should Not Be Bigger Than	num, *pattern_list	Checks whether the number of lines that contains error be less than a number								
Error Should Not Be Bigger Than	num, *pattern_list	Checks whether the number of error be less than a number								
Explicit Run		skip the test case if global_variable RUN_ME is not defined Examples: <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> run.sh will bypass 00. Cabling by default. In other to run this test case \${FORCE} needs declared globally run.sh -X -v FORCE	00. Cabling		Common.Explicit Run		Log To Console	cabling...		
00. Cabling										
Common.Explicit Run										
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, default=None	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 Multi Lines	data, index	Returns multiple lines from text data using index index uses python rule.								
Get Myid		Returns ID uniq for this test case								
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. <b>Note:</b> the keyword only returns the name 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 <b>Notes:</b> Device number could less than node number								
Get Tmp Path		Returns temporary path								
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></table>	Common.Log	XXX	# this always be logged		Common.Log	AAA	level=2	# this will not be logged with common run.sh
Common.Log	XXX	# this always be logged								
Common.Log	AAA	level=2	# this will not be logged with common run.sh							

		<div>Common.<a href="#">Log</a> BBB level=2 # ./run.sh -v DEBUG:2 will log the message</div> <div><b>Notes:</b> 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></div>																
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. <i>Print</i> 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 <i>Run Keywords</i> of RobotFramework ( keyword and its arguments are separated by AND with the others.  Example: <table><tr><td><a href="#">Loop For Node Tag</a></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> <b>Note:</b> \$ in variable name must be escaped	<a href="#">Loop For Node Tag</a>	\\${node}	tag1		...	Switch	\\${node}	AND	...	Cmd	show system user	AND	...	Cmd	show system uptime	
<a href="#">Loop For Node Tag</a>	\\${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><a href="#">Merge Files</a></td><td>./result/*.csv</td><td>./result/test.csv</td></tr></table>	<a href="#">Merge Files</a>	./result/*.csv	./result/test.csv													
<a href="#">Merge Files</a>	./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 <div>... node:   vmx11:     device: vmx11     snmp_polling: yes     mib: mib11.txt ...</div> Default value is defined by mib keyword from global config/snmp-template.yaml for the type of the node  Example: <table><tr><td>\\${mib}=</td><td>Common.<a href="#">MIB For Node</a></td><td>vmx11</td></tr></table>	\\${mib}=	Common. <a href="#">MIB For Node</a>	vmx11													
\\${mib}=	Common. <a href="#">MIB For Node</a>	vmx11																
Node With Attr	attr_name, value	Returns a list of nodes which have attribute attr_name with value value																
Node With Tag	*tag_list	Returns list of node or webapp from local.yaml that has <b>ALL</b> tags defined by tag_list  Tag was defined like this in local.yaml <div>vmx11:   device: vmx11   snmp_polling: yes   tag:     - tag1     - tag2</div> Examples: <table><tr><td>\\${test3}=</td><td>Common.<a href="#">Node With Tag</a></td><td>tag1</td><td>tag3</td></tr></table>	\\${test3}=	Common. <a href="#">Node With Tag</a>	tag1	tag3												
\\${test3}=	Common. <a href="#">Node With Tag</a>	tag1	tag3															
Node Without Tag	*tag_list	Returns list of node from local.yaml that <b>does not has ANY</b> tags defined by tag_list  Tag was defined like this in local.yaml <div>vmx11:   device: vmx11   snmp_polling: yes   tag:     - tag1     - tag2</div> Examples: <table><tr><td>\\${test3}=</td><td>Common.<a href="#">Node Without Tag</a></td><td>tag1</td><td>tag3</td></tr></table>	\\${test3}=	Common. <a href="#">Node Without Tag</a>	tag1	tag3												
\\${test3}=	Common. <a href="#">Node Without Tag</a>	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.  Inf succeed, the keyword returns the input from user.  <b>Notes:</b> If the variable \${RENAT_BATCH} was defined, the keyword will print out the message and keeps running without pausing.  Examples: <table><tr><td>Common.<a href="#">Pause</a></td><td>Waiting...</td><td>10s</td><td>error_on_timeout=\${TRUE}</td><td>default input</td></tr><tr><td>Common.<a href="#">Pause</a></td><td>Waiting...</td><td>10s</td><td></td><td></td></tr></table>	Common. <a href="#">Pause</a>	Waiting...	10s	error_on_timeout=\${TRUE}	default input	Common. <a href="#">Pause</a>	Waiting...	10s								
Common. <a href="#">Pause</a>	Waiting...	10s	error_on_timeout=\${TRUE}	default input														
Common. <a href="#">Pause</a>	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: \${FOLDER}= <u>Random Name</u> capture_%05d 0 99																
Random Number	a=0, b=99	Returns a random number between [a,b]																
Renat Version		Returns RENAT version string																
Screenshot	file_path	Capture whole display to a file specified by file_path  <b>Notes:</b> This keyword saves the whole virtual screen(monitor), while the familiar WebApp.Screenshot Capture only saves the portion of the web browser. But in contrast, the WebApp.Screenshot Capture could do fullpage capture depending on the content of the browser.																
Send	sock, data, recv_buffer_size=1024, encode=utf-8	Sends bytes of data by socket sock and reicve the response  When recv_buffer_size is zero, the function does not execept a response from the remote.																
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  <b>Note:</b> Result folder should be set at the beginning 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																
Start Display		Starts a virtual display																
Stop Next Run	msg=Case run was stopped by user	Stops next run (specified by -n) using a .stop file with reason																
Str 2 Seq	str_index, size	Returns a sequence from string format  Examples: <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																
Wait	wait_time, size=10	Waits for wait-time and display the proress bar  wait_time used RF DateTime format.  Examples: <table><tr><td>Common.Wait</td><td>wait_time=30s</td><td>size=10</td></tr></table>	Common.Wait	wait_time=30s	size=10													
Common.Wait	wait_time=30s	size=10																

