Common

Library version:	RENAT 1.7.1
Library scope:	global
Named arguments:	supported

Introduction

Common library for RENAT

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

Table of Contents

Configuration file

Global configuration

There are 2 kinds of 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 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.

- device.yaml: contains global device information

```
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
```

- auth.yaml: contains authentication information

```
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
```

- template.yaml: contains devvice template information

```
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
        poller: renat
```

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 `case.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`

- `<testcase>/config/local.yaml: contains local data for a test case`

```
node:
    vmx11:
        device: vmx11
        snmp_polling: yes
    vmx12:
        device: vmx11
        snmp_polling: yes
    apollo:
        device: vmx11
        snmp_polling: yes

tester:
    tester01:
        type: ixnet
        ip: 10.128.32.70
        config: vmx_20161129.ixncfg

port-mapping:
    uplink01:
        device: vmx11
        port: ge-0/0/0
    downlink01:
        device: vmx12
        port: ge-0/0/2

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 · File Md5 · Fold Str · Follow Syslog And Trap · Get Config Path · Get File Without Error · Get Item Config Path · Get Item Name · 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	*name*, *mod*, *relative=False*	Changes file mod, likes Unix chmod `mod` is a string specifying the privilege mode `relative` is `False` or `True` Examples: Common.*Change Mod* tmp 0775
Cleanup Result	*ignore=^(log.html|output.xml|report.html)$*	Cleans up the result folder Deletes all files in current active folder that does not match the `ignore` expression and are older than the time the test has started. **Note**: The keyword only removes files but not folders
Convert Html To Pdf	*html_file*, *pdf_file*	Converts html file to pdf file
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` **Notes:** Keyword is matched partially. For example, `error` or `errorXXX` will be matched by `error` keyword.
Count Match Regexp	*regexp*, **pattern_list*	Count the number of `regex` found in `pattern_list` Examples: ${err_num}= *Count Match RegExp* .*error.* result/*.csv result/*.txt
Create Sequence	*start*, *end*, *interval*, *option=float*	Creates a list with number from `start` to `end` with `interval` Example: @{list}= *Create Sequence* 10 15 0.5 will create a list of [11.0, 11.5, 12.0, 12.5, 13.0, 13.5, 14.0, 14.5]
Csv Concat	*src_pattern*, *dst_name*, *has_header=None*	Concatinates CSV files vertically If the CSV files has header, set `has_header` to ${TRUE} Examples: | Commmon.*CSV Merge*	config/data0[3,4].csv	result/result2.csv	
Commmon.*CSV Merge*	config/data0[3,4].csv	result/result2.csv	has_header=${TRUE}	
Csv Merge	*src_pattern*, *dst_name*, *on_key=0*, *has_header=None*	Merges all CSV files horizontally by `on_key` key from `src_pattern` `on_key` is the order of key column that is used as key when merging the files. Default is zero. When `has_header` is not `None` (default value), it is the order of the row used to make the column name. Returns `False` if only one file was found, no merging happend Examples: | Common.*CSV Merge*	config/data0[3,4].csv	result/result2.csv	
Common.*CSV Merge*	config/data0[3,4].csv	result/result2.csv	has_header=${TRUE}	
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 `str_row` and `str_col` are used to specify necessary rows and columns. They are using the same format with slice for Python list. - : and : means all rows and columns - :2 and : means first 2 rows and all columns - : and 1,2 means all rows and 2nd and 3rd columns - 0:3 and 1 means 3 rows from the 1st one(0,1,2) and second column - 0:5:2 and 1 means 3 rows(0,3,5) and second column

Notes:

- Rows and columns are indexed from zero
- When ':' is used, the string has format: <start>:<stop> or <start>:<stop>:<step> For convenience, ':' means all the data, ':x' means first 'x' data

Examples:

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	:

Keyword	Arguments	Description
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
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 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 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 absolue 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:

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

Notes: For common use

- level 1: is default
- level 2: is debug mode
- level 3: is very informative mode

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
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>

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`
		
		``
		...
		node:
		  vmx11:
		    device: vmx11
		    snmp_polling: yes
		    mib: mib11.txt
		...
		``
		
		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.*MIB For Node*</td><td>vmx11</td></tr></table>

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` from `local.yaml` that has **ALL** tags defined by `tag_list`
		
		Tag was defined like this in local.yaml
		
		``
		vmx11:
		  device: vmx11
		  snmp_polling: yes
		  tag:
		    - tag1
		    - tag2
		``
		
		Examples:
		
		<table><tr><td>${test3}=</td><td>Common.*Node With Tag*</td><td>tag1</td><td>tag3</td></tr></table>

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
		
		``
		vmx11:
		  device: vmx11
		  snmp_polling: yes
		  tag:
		    - tag1
		    - tag2
		``
		
		Examples:
		
		<table><tr><td>${test3}=</td><td>Common.*Node Without Tag*</td><td>tag1</td><td>tag3</td></tr></table>

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 ${RENAT_BATCH} was defined, the keyword will print out the message and keeps running without pausing. Examples: <table><tr><td>Common.*Pause*</td><td>Waiting...</td><td>10s</td><td>error_on_timeout=${TRUE}</td><td>default input</td></tr><tr><td>Common.*Pause*</td><td>Waiting...</td><td>10s</td><td></td><td></td></tr></table>
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}= *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 varibles 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=127.0.0.1*	Post a message to Slack
Str 2 Seq	*str_index*, *size*	Returns a sequence from string format
		Samples:
		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

Altogether 48 keywords.
Generated by Libdoc on 2018-04-12 19:20:22.