# RENAT

| | |
|---|---|
| **Library version:** | RENAT 0.1.7 |
| **Library scope:** | global |
| **Named arguments:** | supported |

## Introduction

Document for RENAT framework

All in one pdf renat.pdf

# Libraries

RENAT includes following libraries:

Common:
    Common library of RENAT
VChannel:
    Library controls connection to targets (servers, routers, ...)
Logger:
    Library provides enhanced loggging keywords
Optical:
    Library provides keywords for controlling L1 Switch(Calient)
Router:
    Library provides keywords to control routers, includes mod_juniper mod , mod_cisco mod and mod_gr mod
Tester:
    Library provieds keywors to control testers, includes mod_ixnet and mod_ixload
WebApp:
    Common library for web application, includes 2 child libraries: Samurai and Arbor

# Others

Readme:
    Release information

Choose each libraries for detail infomration and samples about keywords.

## Shortcuts

## Keywords

| Keyword | Arguments | Documentation |
|---|---|---|

Altogether 0 keywords.
Generated by Libdoc on 2018-03-20 02:58:15.

# Common

| | |
|---|---|
| **Library version:** | RENAT 0.1.7 |
| **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

**C**hange Mod · **C**leanup Result · **C**onvert Html To Pdf · **C**ount Keyword · **C**ount Keyword Line · **C**ount Match Regexp · **C**reate Sequence · **C**sv Concat · **C**sv Merge · **C**sv Select · **D**iff File · **E**rr · **E**rror Line Should Not Be Bigger Than · **E**rror Should Not Be Bigger Than · **F**ile Md5 · **F**old Str · **F**ollow Syslog And Trap · **G**et Config Path · **G**et File Without Error · **G**et Item Config Path · **G**et Item Name · **G**et Renat Path · **G**et Result Folder · **G**et Result Path · **G**et Test Device · **I**s Stable · **K**eyword Line Should Not Be Bigger Than · **K**eyword Should Not Be Bigger Than · **L**og · **L**oop For Node Tag · **M**d 5 · **M**erge Files · **M**ib For Node · **N**ode With Attr · **N**ode With Tag · **N**ode Without Tag · **P**ause · **P**ing Until Ok · **R**andom Name · **R**andom Number · **R**enat Version · **S**et Multi Item Variable · **S**et Result Folder · **S**tr 2 Seq · **V**ersion

## Keywords

| Keyword | Arguments | Documentation |
|---|---|---|
| **Change Mod** | *name*, *mod*, *relative=False* | Changes file mod, likes Unix chmod<br>`mod` is a string specifying the privilege mode `relative` is `False` or `True`<br>Examples:<br>Common.*Change Mod* \| tmp \| 0775 |
| **Cleanup Result** | *ignore=^(log.html\|output.xml\|report.html)$* | Cleans up the result folder<br>Deletes all files in current active folder that does not match the `ignore` expression and are older than the time the test has started.<br>**Note**: The keyword only removes files but not folders |
| **Convert Html To Pdf** | *html_file*, *pdf_file* | Converts html file to pdf file |
| **Count Keyword** | *keyword*, **pattern_list* | Count the keyword in files. Keyword is not case-sensitive |
| **Count Keyword Line** | *keyword*, **pattern_list* | Count the number of lines contains the `keyword`<br>**Notes:** Keyword is matched partially. For example, `error` or *`errorXXX* will be matched by *`error* keyword. |
| **Count Match Regexp** | *regexp*, **pattern_list* | Count the number of `regex` found in `pattern_list`<br>Examples:<br>${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`<br>Example:<br>@{list}= \| *Create Sequence* \| 10 \| 15 \| 0.5<br>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 `${TRUE}`<br>Examples:<br>Commmon.*CSV Merge* \| config/data0[3,4].csv \| result/result2.csv \|<br>Commmon.*CSV Merge* \| config/data0[3,4].csv \| result/result2.csv \| has_header=${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`<br>`on_key` is the order of key column that is used as key when merging the files. Default is zero.<br>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<br>Examples:<br>Common.*CSV Merge* \| config/data0[3,4].csv \| result/result2.csv \|<br>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.<br>▪ : and : means all rows and columns<br>▪ :2 and : means first 2 rows and all columns<br>▪ : and 1,2 means all rows and 2nd and 3rd columns<br>▪ 0:3 and 1 means 3 rows from the 1st one(0,1,2) and second column<br>▪ 0:5:2 and 1 means 3 rows(0,3,5) and second column |

| | | |
|---|---|---|
| | | **Notes:**<br>▪ Rows and columns are indexed from zero<br>▪ When ':' is used, the string has format: <start>:<stop> or <start>:<stop>:<step> For convenience, ':' means all the data, ':x' means first 'x' data<br>Examples:<br>*CSV Select* result/data05.csv result/result3.csv 0,1,2 0,1<br>*CSV Select* result/data05.csv result/result4.csv : 0,1<br>*CSV Select* result/data05.csv result/result5.csv :2 :<br>*CSV Select* result/data05.csv result/result6.csv 0:3 :<br>*CSV Select* result/data05.csv result/result7.csv 0:5:2 : |
| **Diff File** | *path1*, *path2*, *newline=True* | Shows difference between files<br>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<br>`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.<br>By default the *log_file_name* is *./result/syslog-trap.log* which is created by *Follow Syslog and Trap* keyword.<br>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.<br>**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<br>**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 |
| **Log** | *msg* | Logs `msg` to the current log file |
| **Loop For Node Tag** | *var*, *tags*, **keywords* | Repeatly executes RF `keyword` for nodes that has tag `tags`<br>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.<br>Example:<br>*Loop For Node Tag* \${node} tag1<br>... *Switch* \${node} AND<br>... *Cmd* show system user AND<br>... *Cmd* show system uptime<br>**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`<br>Example:<br>*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`<br>```<br>...<br>node:<br>  vmx11:<br>  device: vmx11<br>  snmp_polling: yes<br>  mib: mib11.txt<br>...<br>```<br>Default value is defined by `mib` keyword from global `config/snmp-template.yaml` for the `type` of the node<br>Example:<br>`${mib}= Common.MIB For Node vmx11` |
| **Node With Attr** | *attr_name*, *value* | Returns a list of nodes which have attribute `attr_name` with value `value` |
| **Node With Tag** | **tag_list* | Returns list of `node` from `local.yaml` that has **ALL** tags defined by `tag_list`<br>Tag was defined like this in local.yaml<br>```<br>vmx11:<br>  device: vmx11<br>  snmp_polling: yes<br>  tag:<br>  - tag1<br>  - tag2<br>```<br>Examples:<br>`${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`<br>Tag was defined like this in local.yaml<br>```<br>vmx11:<br>  device: vmx11<br>  snmp_polling: yes<br>  tag:<br>  - tag1<br>  - tag2<br>```<br>Examples:<br>`${test3}= Common.Node Without Tag tag1 tag3` |
| **Pause** | *msg=*, *time_out=1s*, *default_input=*, *error_on_timeout=False* | Displays the message `msg` and pauses the test execution and wait for user input<br>In case of `error_on_timeout` is False(default), the keyword will return with `default_input` and the test will be continued wihout error. Otherwise, the keyword will raise an error and stop.<br>If the variable `${RENAT_BATCH}` was defined, the keyword will print out the message and keeps running without pausing.<br>Examples:<br>`Common.Pause Waiting... 10s default error_on_timeout=${TRUE}`<br>`Common.Pause Waiting... 10s` |
| **Ping Until Ok** | *node*, *wait_str=5s*, *extra=-c 3* | Ping a `node` until it gets response. Then wait for more `wait_str` Default `extra` option is `-c 3` |
| **Random Name** | *base*, *a=0*, *b=99* | Returns a random name by a *base* and a random number between [a,b]<br>Example:<br>`${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<br>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 ...<br>`folder` is a folder name that under current test case folder<br>The system will create a new folder if it does not exist and set its mode to *0775*<br>**Note:** Result folder should be set at the begining of the test. Changing result folder only has effect on up comming connection |
| **Str 2 Seq** | *str_index*, *size* | Returns a sequence from string format |

| | | |
|---|---|---|
| | | Samples:<br>*Str2Seq* :: 5 # (0,1,2,3,4)<br>*Str2Seq* :2 5 # (0,1)<br>*Str2Seq* 1:3 5 # (1,2)<br>*Str2Seq* 0:5:2 5 # (0,2,4) |
| **Version** | | Returns the current version of RENAT |

Altogether 45 keywords.
Generated by Libdoc on 2018-03-20 02:58:08.

# VChannel

| | |
|---|---|
| **Library version:** | RENAT 0.1.7 |
| **Library scope:** | test suite |
| **Named arguments:** | supported |

## Introduction

A basic library that provides Terminal connection to routers/hosts

`VChannel` is a core RENAT library that maintains input/output to nodes with an attached virtual terminal. It encapsulates the SSH/Telnet connections behind and provides common usage of access and execute commands to the nodes. Each channel instance has its own log file and a virtual terminal.

### Table of Contents



## Device, Node and Channel

RENAT has 3 types of connection target. `Device`, `Node` and `Channel`.

### Device

Each device stands for a real physical box that has its own IP address and is defined in the master file `device.yaml`. Users do not directly use `device` in keywords.

### Node

Node is a logical instance of a `device`. It could stand for a logical instance of a router or just a virtual terminal to the router. Nodes were defined in `local.yaml` of the test case. Several nodes could point to a same device.

### Channel

Each channel holds a session to a node. Each channel has its own log file and a virtual terminal. Any command used by *Cmd*,*Write* or *Read* will be logged to the log file. Each channel is identified by a name when it is created with *Connect* keyword and is released with *Close* keyword.

**Notes:** multi sessions to a same device could be done with predefined multi nodes to same device in the `local.yaml` file or by using multi *Connect* with different *name*.

## Connections

The library provides a channel to a target node. Each channel is attached with a virtual terminal. Input and output to the node are made through this virtual terminal. This will help to provide the output looks like the output when operator is using the real terminal.

When keywords *Read*, *Write*, *Cmd* are used, if the connection is not available anymore, the system will try to reconnect to the host with the information provided in the 1st connect. It will try `max_retry_for_connect` times and wait for `interval_between_retry` seconds between retries. The values of `max_retry_for_connect` and `interval_between_retry` are defined in `./config/config.yaml`

Usually when RENAT could not make the connections to the target, the system will raise an exception. But if the `ignore_dead_node` is defined as `yes` in the current active `local.yaml`, the system will ignore the dead node, remove it from the global variable `LOCAL['node']` and `NODE` and keep running the test.

## Shortcuts

**C**hange Log · **C**hange Prompt · **C**lose · **C**lose All · **C**md · **C**md Yesno · **C**onnect · **C**onnect All · **F**lush All · **G**et Channel · **G**et Channels · **G**et Current Channel · **L**og · **R**ead · **R**econnect · **S**et Log Separator · **S**tart Screen Mode · **S**top Screen Mode · **S**witch · **W**rite

## Keywords

| Keyword | Arguments | Documentation |
|---|---|---|
| **Change Log** | *log_file*, *mode=w* | Stops current log file and create a new log file.<br>Every log from that point will be saved to the new log file Return old log filename |
| **Change Prompt** | *str_prompt* | Changes the current prompt of the channel<br>Returns previous prompt. User should change the prompt `before` execute the new command that expects to see new prompt.<br>Example: |

| | | |
|---|---|---|
| Router.*Switch* | vmx11 | |
| ${prompt}= | VChannel.*Change Prompt* | % |
| VChannel.*Cmd* | start shell | |
| VChannel.*Cmd* | ls | |
| VChannel.*Change Prompt* | ${prompt} | |

| | | |
|---|---|---|
| | | Vchannel.*Cmd* exit |
| **Close** | | Closes current connection and reset the channel name |
| **Close All** | | Closes all current sessions and flush out all log files.<br>Current node name was reset to None |
| **Cmd** | *command*, *prompt=*, *match_err=(unknown command.\|syntax error, expecting <command>.)* | Executes a command and wait until for the prompt.<br>This is a blocking keyword. Execution of the test case will be postponed until the prompt appears. If prompt is a null string (default), its value is defined in the ./config/template.yaml<br>Output will be automatically logged to the channel current log file.<br>See Common for details about the config files. |
| **Cmd Yesno** | *cmd*, *ans=yes*, *question=? [yes,no]* | Executes a str_cmd, waits for question and answers that by ans |
| **Connect** | *node*, *name*, *log_file*, *timeout=20m*, *w=80*, *h=32*, *mode=w* | Connects to the node and create a VChannel instance<br>Login information is automatically extracted from yaml configuration. By defaullt a virtual terminal (vty100) with size 80x64 is attachted to this channel.<br>If a login was successful, VChannel will create a log file name log_file for the connection in the current result folder of the test case. This log file will contain any command input/output executed on this channel.<br>Multi sessions to the same node could be open with different names. Use *Switch* to change the current active session by its name<br>Examples:<br>*Connect* vmx11 vmx11 vmx11.log<br>*Connect* vmx11 vmx11 vmx11.log 80 64<br>See Common for more detail about the yaml config files. |
| **Connect All** | *prefix=* | Connects to **All** nodes that are defined in active local.yaml.<br>A prefix prefix was appended to the alias name of the connection. A new log file by <alias>.log was automatiocally created.<br>See *Common* for more detail about active local.yaml |
| **Flush All** | | |
| **Get Channel** | *name* | Returns a channel by its name |
| **Get Channels** | | Returns all current vchannel instances |
| **Get Current Channel** | | Returns the current active channel |
| **Log** | *msg* | Writes the log message msg to current log file of the channel |
| **Read** | *silence=False* | Returns the current output of the virtual terminal and automatically logs to file.<br>In normal mode this will return the **unread** output only, not all the content of the screen. |
| **Reconnect** | *name* | Reconnects to the name node using existed information<br>The only difference is that the mode of the log file is set to `a+` by default |
| **Set Log Separator** | *sep=* | Set a separator between the log of read, write or cmd keywords |
| **Start Screen Mode** | | Starts the screen mode.<br>In the screen mode, the output is just the same with the real terminal. It means that any real-time application likes top will be captured as-is. Consecutive *read* from this VChannel instance may produce redundancy ouput. |
| **Stop Screen Mode** | | Stops the screen mode and returns to normal mode<br>In screen mode, *Write* does not return any thing and no output is logged. In normal mode, escape sequences are not processed by the virtual terminal. |
| **Switch** | *name* | Switches the current active channel to name. There only one active channel at any time<br>Examples:<br>VChannel.*Switch* vmx12 |
| **Write** | *str_cmd*, *str_wait=1s*, *start_screen_mode=False* | Sends str_cmd to the target node and return after str_wait time.<br>If start_screen_mode is True, the channel will be shifted to Screen Mode. Default value of screen_mode is False.<br>In normal mode, a new line char will be added automatically to the str_cmd and the command return the output it could get at that time from the terminal and also logs that to the log file.<br>In screen Mode, if it is necessary you need to add the new line char by your own and the ouput is not be logged or returned from the keyword.<br>Parameters: |

| | | |
|---|---|---|
| | | ▪ str_cmd: the command<br>▪ str_wait: time to wait after apply the command<br>Special key likes Ctrl-C etc. could be used with global variable ${CTRL-<char>}<br>Returns the output after writing the command the the channel.<br>**Notes:** This is a non-blocking command.<br>Examples:<br>VChannel.*Write* \| monitor interface traffic \| start_screen_mode=${TRUE}<br>VChannel.*Write* \| ${CTRL_C} \| # simulates Ctrl-C |

Altogether 20 keywords.
Generated by Libdoc on 2018-03-20 02:58:11.

# Logger

| | |
|---|---|
| **Library version:** | RENAT 0.1.7 |
| **Library scope:** | test suite |
| **Named arguments:** | supported |

## Introduction

Provides advanced logging functions. Every Logger instance has one VChannel object and the is synchronized with the current active VChannel.

## Shortcuts

**L**og · **L**og All · **S**witch

## Keywords

| Keyword | Arguments | Documentation |
|---|---|---|
| **Log** | *msg*, *with_time=False*, *mark=**** | Inserts a message `msg` to the current *VChannel* log file. A default mark of `***` will be added at the beginning ant the end of this message.<br>Example:<br>Logger.*Log* \| START TRAFFIC FROM HERE \| ${TRUE} \|<br>Logger.*Log* \| START TRAFFIC FROM HERE \| ${False} \| **===** |
| **Log All** | *msg*, *with_time=False*, *mark=**** | inserts a message `msg` to current **all** VChannel log files.<br>A default `mark` of `***` and newline will be added at the beggining and the end of this message.<br>Example:<br>Logger.*Log All* \| START TRAFFIC FROM HERE \| ${TRUE} \|<br>Logger.*Log All* \| START TRAFFIC FROM HERE \| ${TRUE} \| **===**<br>The log file will look likes this:<br>user@vmx12><br>*** 06:01PM on August 13, 2017: START TRAFFIC FROM HERE ***<br>=== 06:01PM on August 13, 2017: START TRAFFIC FROM HERE ===<br>configure |
| **Switch** | *name* | Switches the current VChannel instance to `name`. `name` is the name of the VChannel (usually is the node name defined in the current active `local.yaml`).<br>Example:<br>Logger.*Switch* \| vmx11 |

Altogether 3 keywords.
Generated by Libdoc on 2018-03-20 02:58:09.

# OpticalSwitch

| | |
|---|---|
| **Library version:** | RENAT 0.1.7 |
| **Library scope:** | test suite |
| **Named arguments:** | supported |

## Introduction

A library provides control for L1 Optical Switch (currently Calient)

`OpticalSwitch` is a RENAT library that provides control for L1 optical switch. Currently the library only supports Calient.

### Table of Contents



## Master file

The L1 switch provides a mechanism to remotely connect device interface. Each device interface has been wired to L1 switch already. The connection was described in the master file located specific by *calient-master-path* in the configuration file *renat/config/config.yaml*.

The master file includes several Calients in each tab. The column meaning and order is trivial.

## Connection file Format

Keywords *Load From File*, *Clear By File* and *Save To File* use the x-connection file. The connection has following rules:

Connection files are text files and have the following format:

```
# this is the comment
device1,interface1,-,device2,interface2
device1,interface1,>,device2,interface2
```

The separator `-` means a bidirection connection and `>` means a unidirection connection. For a unidirection connection, `device1/interface1` TX will be connected to `device2/interface2` RX.

**Note:** The separator character must be surrounded by spaces or commas.

The connection file also support jinja2 template format. After the template is evaluated, comment could be used by `comment char`

There is no need to specify which L1 switch for the x-connection. The system will automatically find the appropriate switch.

## Shortcuts

**A**dd · **C**lear By File · **C**lose All · **C**onnect All · **D**elete · **G**et Connection Info · **L**oad From File · **R**ead Map · **S**ave To File

## Keywords

| Keyword | Arguments | Documentation |
|---|---|---|
| **Add** | *dev1*, *intf1*, *dev2*, *intf2*, *direction=bi*, *force=False* | Adds x-connection between `dev1:intf1` and `dev2:intf2`<br>`direction` is `bi` for bi-direction or `uni` for uni-direction. If `direction` is `uni`, the tx of `dev 1:port 1` will be connected to `dev 2:port 2`.<br>With `force` mode, existed connection that use those ports will be deleted. Without `force` mode, an existed connection will make the keyword fails<br>Examples:<br>`OpticalSwitch.Add` `mx2008-31-33` `xe-3/0/0` `mx2008-31-33` `xe-3/0/1` `bi` `${TRUE}`<br>**Note**: when `force` is `False` but the current ports is owned by the same connection endpoints, keyword will succeed.<br>**Note:** For a bidirection connection, 2 single uni-direction connection will be made instead of 1 bi-direction connection. This will make the link could be simulated tx/rx failure later. |
| **Clear By File** | *file_name=*, *comment=#* | Clears all x-connections defined in the *connection file*<br>Default *connection file* is defined in `optic/connection` of `config/local.yaml` |
| **Close All** | | |
| **Connect All** | | |
| **Delete** | *dev1*, *intf1*, *dev2*, *intf2*, *direction=bi* | Deletes the connection between `dev1:intf1 - dev2:intf2`<br>Examples:<br>`OpticalSwitch.Delete` `mx2008-31-33` `xe-3/0/1` `mx2008-31-33` `xe-3/0/1` `uni` |
| **Get Connection** | *dev*, *intf* | Returns information of the optic switch port that connected to `dev:intf`. The information is in jason |

| | | |
|---|---|---|
| **Info** | | format.<br>Examples:<br>OpticalSwitch.*Get Connection Info* \| mx2008-31-33 \| xe-3/0/1<br>return information looks like below:<br>result = {u'outoc': u'NOHW', u'outopwdh': u'-20.0', u'inos': u'OOS',<br>u'outalias': u'', u'inowner': u'TRANSIT', u'outopwct': u'-23.0', u'inpower':<br>u'-3.4', u'inas': u'IS', u'outpower': u'-4.8', u'outas': u'OOS-NP', u'inopt':<br>u'-17.0', u'inopth': u'13.0', u'incircuit': u'3.3.1>3.3.2', u'inalias': u'',<br>u'inoc': u'NOHW', u'inoptc': u'-20.0', u'outos': u'OOS', u'port': u'3.3.1',<br>u'outowner': u'NONE', u'outcircuit': u''} |
| **Load From File** | *file_name=*, *force=True*, *comment=#* | Loads the connection file and set the connections<br>`filename` is the name of the connection file under the current config folder. If `filename` is empty, the value of `optic/connection` from `config/local.yaml` will be used.<br>The `connection file` supports `jinja2` template language. Besides, `#` is the default comment char which could be changed<br>The format of `connection file` follows:<br>▪ each connection is described by 1 line<br>▪ `source` and `destination` are separated by `` ` - *or* > `, *which mean* `` `bidirection `` or `unidirection` (unidirection connects `source tx` to `dest rx`<br>Connection file sample:<br>device1:port1 - device2:port2<br>device1:port3 > device2:port<br>Examples:<br>OpticalSwitch.*Load From File*<br>OpticalSwitch.*Load From File* \| save1.conn |
| **Read Map** | | Reads the master port map file<br>Make lower for all informations. |
| **Save To File** | *file_name* | Saves the current connection of all devices in this test.<br>By default, all interfaces of the devices are save. If a connection file is given, only interfaces specified in the connection file are saved<br>Examples:<br>OpticalSwitch.*Save To File* \| save1.conn |

Altogether 9 keywords.
Generated by Libdoc on 2018-03-20 02:58:10.

# Router

| | |
|---|---|
| **Library version:** | RENAT 0.1.7 |
| **Library scope:** | test suite |
| **Named arguments:** | supported |

## Introduction

A class provides keywords for router controll. Usual command could be executed via VChannel. This class provides the vendor independent commands

Device's `type` is defined in master `device.yaml`. The system will load appropriate modules for each device.

Details about keywords provided by modules could be found in document of each module likes:

- Juniper module
- Cisco module
- GR module

Keywords provides by above module could be executed through *Xrun* keyword or directly called from `Router`. Examples:

| | |
|---|---|
| Router.*Xrun* | Load Config |
| Router.*Load Config* | |

## Shortcuts

**C**md · **E**xec File · **F**ollow Mib · **F**ollow Syslog · **G**et Ip · **R**ead · **S**nap · **S**nap Diff · **S**top Screen Mode · **S**witch · **W**rite · **X**run

## Keywords

| Keyword | Arguments | Documentation |
|---|---|---|
| **Cmd** | *str_cmd=*, *str_prompt=* | Runs the command `str_cmd` and waits until the prompt defined for this router. This keyword is identical to `VChannel.Cmd`<br><br>Examples:<br><br>Router.*Cmd* \| set system login user testtest authentication plain-text-password \| password: \| # wait for *password:*<br>Router.*Cmd* \| Renat2017 \| password: \| # wait for *password:*<br>Router.*Cmd* \| Renat2017 \| \| # wait for default prompt<br><br>The above sample creates an output likes this:<br><br>`user@vmx11# set system login user testtest authentication plain-text-password`<br>`New password:Renat2017`<br>`Retype new password:Renat2017`<br><br>`[edit]` |
| **Exec File** | *file_name*, *vars=*, *comment=#* , *step=False*, *str_error=syntax,rror* | Executes commands listed in `file_name` Lines started with `comment` character is considered as comments<br><br>`file_name` is a file located inside the `config` folder of the test case.<br><br>This command file could be written in Jinja2 format. Default usable variables are `LOCAL` and `GLOBAL` which are identical to `Common.LOCAL` and `Common.GLOBAL`. More variables could be supplied to the template by `vars`.<br><br>`vars` has the format: `var1=value1,var2=value2`<br><br>If `step` is `True`, after very command the output is check agains an error list. And if a match is found, execution will be stopped. Error list is define by `str_err`, that contains multi regular expression separated by a comma. Default value of `str_err` is *error*<br><br>A sample for command list with Jinja2 template:<br><br>`show interface {{ LOCAL['extra']['line1'] }}`<br>`show interface {{ LOCAL['extra']['line2'] }}`<br><br>`{% for i in range(2) %}`<br>`show interface et-0/0/{{ i }}`<br>`{% endfor %}`<br><br>Examples:<br><br>Router.*Exec File* \| cmd.lst \|<br>Router.*Exec File* \| step=${TRUE} \| str_error=syntax,error<br><br>**Note:** Comment in the middle of the line is not supported For example if `comment` is "# "<br><br>`# this is comment line <-- this line will be ignored` |

| | | |
|---|---|---|
| | | ## this is not an comment line, and will be enterd to the router cli, but the router might ignore this |
| **Follow Mib** | *node_list*, *wait_time=10s*, *interval_time=5s*, *len=12*, *percentile=80*, *threshold=75*, *max_len=300*, *factor=1* | Waits until all the nodes defined in `node_list` become `stable`.<br>Stableness is checked by SNMP polling result. The MIB list is define by `mib` in `node` section Parameter:<br>▪ `wait_time(1)`: the time before the evaluation starting<br>▪ `interval_time(2)`: interval between SNMP polling time<br>▪ `threshold`: below this value is evaluated as `stable`<br>▪ `len(3)`: the size of the evaluation window (number of values that are used in each valuation)<br>▪ `percentile`: real useful percentage of data (ignore top `100-percentile` percent)<br>▪ `max_len(4)`: maximum waiting `lend` for this checking<br>time sequence: --(1)--\|-(2)-\|-----\|-----\|----\|-----\|-----\|<br><--------(3)----------> poll poll<br><--------(3)----------><br><----------(4)----------> |
| **Follow Syslog** | *pattern* | Watches syslog of this device on Apollo server and block the excuation until the `pattern` is matched |
| **Get Ip** | | Returns the IP address of current node Examples:<br>${router_ip}= Router.*Get IP* |
| **Read** | | Executes command `read` for the current vchannel coressponded to this router |
| **Snap** | *name*, **cmd_list* | Remembers the result of a list of command defined by `cmd_list`<br>Use this keyword with *Snap Diff* to get the difference between the command's result. The a new snapshot will overrride the previous result.<br>Each snap is identified by its `name` |
| **Snap Diff** | *name* | Executes the comman that have been executed before by `name` snapshot and return the difference.<br>Difference is in `context diff` format |
| **Stop Screen Mode** | | Stop the screen mode |
| **Switch** | *name* | Changes the current channel of this router to `name`<br>Rerturns old node name **Note:** This is identical to VChannel.Switch<br>Examples:<br>Router.*Switch* vmx11<br>Router.*Cmd* show version |
| **Write** | *cmd_str*, *wait_str=1s*, *start_screen_mode=False* | Executes command `write` for the current vchannel coressponded to this router |
| **Xrun** | *cmd*, **args*, ***kwargs* | Runs the vendor independent keywords.<br>Parametes:<br>▪ cmd: a keyword<br>▪ args: other argumemts<br>Examples:<br>Router.*Xrun* Flap Interface ge-0/0/0<br>This keyword will then actually calling the correspond keyword for the device type. |

Altogether 12 keywords.
Generated by Libdoc on 2018-03-20 02:58:09.

# cisco

| | |
|---|---|
| **Library scope:** | global |
| **Named arguments:** | supported |

## Introduction

Provides keywords for Cisco platform

## Shortcuts

**G**et Version

## Keywords

| Keyword | Arguments | Documentation |
|---|---|---|
| **Get Version** | *self* | return router version information |

Altogether 1 keywords.
Generated by Libdoc on 2018-03-20 02:58:12.

# gr

| | |
|---|---|
| **Library scope:** | global |
| **Named arguments:** | supported |

## Introduction

Provides keywords for Hitachi GR platform

## Shortcuts

**G**et Chassis Serial · **G**et Version

## Keywords

| Keyword | Arguments | Documentation |
|---|---|---|
| **Get Chassis Serial** | *self* | Returns the serial number of the chassis |
| **Get Version** | *self* | return router version information |

Altogether 2 keywords.
Generated by Libdoc on 2018-03-20 02:58:12.

# juniper

**Library scope:** global
**Named arguments:** supported

## Introduction

Provides keywords for Juniper platform

**Notes:** Ignore the *self* parameters when using those keywords.

## Shortcuts

**C**reate Best Path Select Data · **D**isable Interface · **E**nable Interface · **F**lap Interface · **G**et Chassis Serial · **G**et Cli Mode · **G**et Config · **G**et Current Datetime · **G**et File · **G**et Intf Addr · **G**et Route Number · **G**et Version · **L**ink Status · **L**oad Config · **N**umber Of Bgp Neighbor · **N**umber Of Ospf Neighbor · **N**umber Of Ospf3 Neighbor

## Keywords

| Keyword | Arguments | Documentation |
|---|---|---|
| **Create Best Path Select Data** | *self*, *route_content*, *output_excel=best.xlsx* | Creates the matrix of best path selection<br>Provides the test described in *smb://10.128.3.91/SharePoint01/31_VerificationRoom/31_13_検証環境セット/BGP-Best-Path-SelectionのAll-in-One設定_20161118改良/* The test uses predefined Ixia config and follows predefined steps |
| **Disable Interface** | *self*, *intf* | Disables an interface `intf` |
| **Enable Interface** | *self*, *intf* | Enables an interface `intf` |
| **Flap Interface** | *self*, *intf*, *time_str=10s* | Simulates an interface flap for interface `intf`<br>Disables the interface and wait for a while before turning it up again |
| **Get Chassis Serial** | *self* | Returns the serial number of the chassis |
| **Get Cli Mode** | *self* | Returns current mode of the CLI.<br>Return value is `config` for configuration mode or `command` for command mode |
| **Get Config** | *self*, *dst_name=* | Gets the current configuration file of the router to current `result` folder. Wait for `str_timeout` to finish the download, default `str_timeout` is 10 seconds. Increases this value if the config file is large.<br>Default `dst_name` is `juniper.conf.gz` |
| **Get Current Datetime** | *self*, *time_format=%H:%M:%S*, *delta_time=0s*, *dir=+*, ***kwargs* | Returns the current date time with vendor format `delta_time` will be added or subtracted to current time, default is `0s`<br>`time_format` decides the time part of the output. Example result are :<br>`May 24 04:14:25`<br>`May  4 04:14:25`<br>**Note:** The date part is padded by space, and the result is allways 15 characters |
| **Get File** | *self*, *src_file*, *dst_file=* | Gets a file from router<br>▪ `src_file` is a absolute path insides the router<br>▪ `dst_file` is a file name under `result` folder |
| **Get Intf Addr** | *self*, *intf_name*, *family=inet* | Returns the tuple of address and netmask of an interface<br>`family` should be `inet` or `inet6` If the address is not set, (",") will be returned. |
| **Get Route Number** | *self*, *table=inet.0* | Returns number of active route in the `table`<br>`table` could be `inet.0` or `inet.6` |
| **Get Version** | *self* | Returns router version information |
| **Link Status** | *self*, *if_name* | Returns link physical status as string (aka: "up down", "up up") |
| **Load Config** | *self*, *mode=set*, *config_file=*, *confirm=0s*, *vars=*, *err_match=( syntax \| error )* | Loads configuration to a router. Usable `mode` is `set`, `override`, `merge` and `replace`<br>`set` mode uses configuration that contains `set` command. Mode `override`, `merge` and `replace` use ordinary JunOS configuration file with appropriate mode. `config_file` is a configuration file inside the `config` folder of the current test case.<br>Config file could includes jinja2 template. The template will be evalued with *LOCAL*, *GLOBAL* and varibles defined by *vars*. The *vars* has the format: var1=value1,var2=value2 ...<br>If the loading has no error that match the `error_match`, the configuration will be commited.<br>The keywordl waits for `confirm` seconds before rollback the commited configuration. A zero value indicates an immediatly commit |

| | | |
|---|---|---|
| **Number Of Bgp Neighbor** | *self*, *state=Established* | Returns number of BGP neighbor in `state` state |
| **Number Of Ospf Neighbor** | *self*, *state=Full* | Returns number of OPSF neighbors with status `state` |
| **Number Of Ospf3 Neighbor** | *self*, *state=Full* | Returns number of OPSFv3 neighbors with status `state` |

Altogether 17 keywords.
Generated by Libdoc on 2018-03-20 02:58:12.

# WebApp

| | |
|---|---|
| **Library version:** | RENAT 0.1.7 |
| **Library scope:** | test suite |
| **Named arguments:** | supported |

## Introduction

A library provides common keywords for web applications (aka Samurai, Arbor TMS)

The library utilize *Selenium2Library* and adds more functions to control Samurai application easily.

The *WebApp* uses the configuration in `local.yaml` in `webapp` section:

```
...
webapp:
  samurai-1:
    device: samurai-b
    profile: samurai.profile
  arbor-1:
    device: arbor-sp-a
    profile: samurai.profile
...
```

*Selenium2Library* keywords still could be used along with this library. See Selenium2Library for more details.

See Arbor, Samurai for details about keywords of each application.

## Shortcuts

**C**apture Screenshot · **C**onnect All · **R**eset Capture Counter · **S**et Capture Counter · **S**et Capture Format

## Keywords

| Keyword | Arguments | Documentation |
|---|---|---|
| **Capture Screenshot** | *filename=None*, *extra=* | Captures the current screen to file<br>Using the internal counter for filename if `filename` is not specified. In this case, the filename is defined by a pre-set format. *Set Capture Format* could be used to change the current format.<br>An extra information will be add to the filename if `extra` is defined<br>Examples:<br>Samurai.*Capture Screenshot* \| \| # samurai_0000000001.png<br>Samurai.*Capture Screenshot* \| extra=_list \| # samurai_0000000002_*list*.png<br>Arbor.*Capture Screenshot* \| \| # arbor_0000000001.png<br>Arbor.*Capture Screenshot* \| extra=_xxx \| # arbor_0000000001_*xxx*.png<br>Samurai.*Capture Screenshot* \| file_name=1111.png \| # 1111.png |
| **Connect All** | | Connects to all applications defined in `local.yaml`<br>The name of the connection will be the same of the *webapp* name |
| **Reset Capture Counter** | | Resets the counter of the screen capture |
| **Set Capture Counter** | *value=0* | Sets the counter of the screen capture to `value` |
| **Set Capture Format** | *format* | Sets the format for the screen capture file<br>The format does not include the default prefix `.png` The default format is `<mod>_%010d`. `mod` could be `samurai` or `arbor`<br>See https://docs.python.org/2/library/string.html#format-specification-mini-language for more details about the format string.<br>Examples:<br>Samurai.*Set Capture Format* \| ${case}_%010d \| # ${case} is a predefined variable |

Altogether 5 keywords.
Generated by Libdoc on 2018-03-20 02:58:14.

# Samurai

| | |
|---|---|
| **Library version:** | RENAT 0.1.7 |
| **Library scope:** | test suite |
| **Named arguments:** | supported |

## Introduction

A library provides functions to control Samurai application

The library utilize *Selenium2Library* and adds more functions to control Samurai application easily. Without other furthur mentions, all of the concepts of `user`, `user group` are Samurai concepts.

By default, RENAT will try to connecto all Samurai nodes defined in active `local.yaml` at the beginning of the test and disconnect from them at the end of the test automatically. Usually user does not need to use `Connect All` and `Close` explicitly.

Currently, this module supposed that Samurai is used in Japanese locale. When Samurai module has error, it tried to make the last snapshot in `result/selenium-screenshot-x.png`. Checking this capture will help to understand the reason of the error.

Some keywords of Samurai is using `xpath` to identify elements. See *Selenium2Library* for more details about xpath.

See WebApp for common keywords of web applications.

*Selenium2Library* keywords still could be used together within this library. See Selenium2Library for more details.

## Shortcuts

**A**dd Policy · **A**dd Policy Group · **A**dd User · **C**apture Screenshot · **C**hange Policy View Group · **C**lick All Elements · **C**lose · **C**lose All · **C**onnect · **C**onnect All · **D**elete Policy · **D**elete Policy Group · **D**elete User · **E**dit Policy · **L**eft Menu · **L**ogin · **L**ogout · **M**ake Item Map · **R**eset Capture Counter · **S**elect Items In Table · **S**et Capture Counter · **S**et Capture Format · **S**how Policy Basic · **S**how Policy Mitigation · **S**how Policy Mo · **S**how Policy Monitor · **S**tart Mitigation · **S**top Mitigation · **S**witch

## Keywords

<table>
<tr><th>Keyword</th><th>Arguments</th><th>Documentation</th></tr>
<tr><td>Add Policy</td><td>**policy</td><td>
Adds a new Samurai policy<br><br>
<code>policy</code> is a map containing the below information to create the new policy.<br><br>
<table>
<tr><th>key</th><th>meaning</th><th>mandatory</th><th>sample</th></tr>
<tr><td>name</td><td>name of the policy</td><td>yes</td><td>test001</td></tr>
<tr><td>basic_alias</td><td>alias name of the policy</td><td></td><td>test001</td></tr>
<tr><td>basic_port_id</td><td>another alias</td><td></td><td></td></tr>
<tr><td>basic_facing</td><td><code>customer</code> or <code>backbone</code></td><td></td><td>customer</td></tr>
<tr><td>basic_intf_list</td><td>list of router and interface pair, separated by comma</td><td>yes</td><td>10.128.18.31:xe-0/0/0.1</td></tr>
<tr><td>basic_cidr_list</td><td>list of CIDR separate by comma</td><td></td><td></td></tr>
<tr><td>basic_option_filter</td><td>optinal filter</td><td></td><td></td></tr>
<tr><td>basic_direction</td><td>direction of the traffic (<code>incoming</code> or <code>outgoing</code>)</td><td></td><td>Incoming</td></tr>
<tr><td>traffic_enabled</td><td>Enable traffic monitoring or not</td><td>yes</td><td>${True}</i> or <i>${False}</td></tr>
<tr><td>detection_enabled</td><td>Enable detection or not</td><td>yes</td><td>${True}</i> or <i>${False}</td></tr>
<tr><td>mitigation_zone_name</td><td>Name of the zone for mitigation</td><td></td><td>zone001</td></tr>
<tr><td>mitigation_zone_prefix</td><td>Prefixes that could mitigate</td><td></td><td>1.1.1.1/32</td></tr>
<tr><td>mitigation_thr_bps</td><td>Upper limit (bps)</td><td></td><td>800,000,000</td></tr>
<tr><td>mitigation_thr_pps</td><td>Upper limit (pps)</td><td></td><td>54,000,000</td></tr>
<tr><td>mitigation_mo_enabled</td><td>Using Arbor TMS MO or not</td><td>yes</td><td>${True}</i> or <i>${False}</td></tr>
<tr><td>mitigation_device_list</td><td>Devices used for TMS, separated by comma</td><td></td><td>ArborSP-A</td></tr>
<tr><td>mitigation_mo_name</td><td>MO name, separated by comma</td><td></td><td>OCN12(ALU)_LOOSE</td></tr>
<tr><td>mitigation_comm_list</td><td>commna separated peer/community list</td><td>yes</td><td>1.10(180.0.1.10)/2914:666,1.11(180.0.1.11)/2914:777</td></tr>
<tr><td>nw_monitor_gre1</td><td>1st GRE address for NW monitor</td><td></td><td>210.0.1.1</td></tr>
<tr><td>nw_monitor_gre2</td><td>2nd GRE address for NW monitor</td><td></td><td>210.0.1.1</td></tr>
<tr><td>nw_monitor_ce1</td><td>1st CE address for NW monitor</td><td></td><td>210.0.1.2</td></tr>
<tr><td>nw_monitor_ce2</td><td>2nd CE address for NW monitor</td><td></td><td>210.0.1.2</td></tr>
<tr><td>nw_monitor_pe1</td><td>1st PE for NW monitor (list)</td><td></td><td>edge01hige-MX2020-15(118.23.176.244)</td></tr>
<tr><td>nw_monitor_pe2</td><td>2nd PE for NW monitor (list)</td><td></td><td>edge01hige-MX2020-15(118.23.176.244)</td></tr>
<tr><td>event_name</td><td>name of the message event to make</td><td></td><td>info1</td></tr>
<tr><td>event_addr</td><td>address to send the events</td><td></td><td>user@mail.com</td></tr>
<tr><td>view_group</td><td>user group that could view this policy, separated by comma</td><td>yes</td><td>SuperGroup,test_group_007</td></tr>
</table>
<br>
Example:<br><br>
<table>
<tr><td>Samurai.Switch</td><td>samurai-1</td><td></td></tr>
<tr><td>Samurai.Add Policy</td><td>name=${POLICY_NAME}</td><td>basic_alias=${POLICY_NAME}</td></tr>
<tr><td>...</td><td>basic_facing=customer</td><td>basic_intf_list=10.128.18.31:xe-0/0/0.1</td></tr>
<tr><td>...</td><td>basic_cidr_list=1.1.1.0/24</td><td>basic_direction=incoming</td></tr>
<tr><td>...</td><td>traffic_enabled=${TRUE}</td><td></td></tr>
<tr><td>...</td><td>detection_enabled=${TRUE}</td><td></td></tr>
</table>
</td></tr>
</table>

| | | | | |
|---|---|---|---|---|
| | | ... | mitigation_zone_name=test_zone001 | mitigation_zone_prefix=1.1.1.1/32 |
| | | ... | mitigation_device_list=ArborSP-A,ArborSP-B | |
| | | ... | mitigation_mo_enabled=${TRUE} | |
| | | ... | mitigation_mo_name=N000000012_LOOSE | |
| | | ... | mitigation_comm_list=1.10(180.0.1.10)/2914:666,1.11(180.0.1.11)/2914:777 | |
| | | ... | event_name=test | event_addr=user@mail.com |
| | | ... | view_group=SuperGroup | |

| | | |
|---|---|---|
| **Add Policy Group** | *group_name*, *policy_list=**, *limit_bps=4000000000*, *limit_pps=2700000* | Add a new policy group<br>`group_name` is the name of the new group. `policy_list` is a comma separated of existed policy that should be bound to this policy. An asterisk for this parameter (`*`) means *all of the existed policy*. `limit_bps` and `limit_pps` are the mitigation capacity threshold of this group. |
| **Add User** | *group*, ***user_info* | Adds user to the current group `user_info` is a dictionary contains user information that has following keys: `name`, `password`, `privilege` and `policy`<br>`privilege` is existed privilege that has been created (e.g: *system_admin*.<br>`policy` could be `*` for all current policies or a list of policy names that are binded to this user.<br>`group` is the user group. `Dot(.)` means current group<br>Examples:<br>Samurai.*Add User* \| OCNDDoS \| name=user000 \| password=Test12345678<br>... \| privilege=system_admin \| policy=*<br>Samurai.*Add User* \| OCNDDoS \| username=user001 \| password=Test12345678<br>... \| privilege=system_admin \| policy=OCN11,OCN12 |
| **Capture Screenshot** | *filename=None*, *extra=* | Captures the current screen to file<br>Using the internal counter for filename if `filename` is not specified. In this case, the filename is defined by a pre-set format. *Set Capture Format* could be used to change the current format.<br>An extra information will be add to the filename if `extra` is defined<br>Examples:<br>Samurai.*Capture Screenshot* \| \| # samurai_0000000001.png<br>Samurai.*Capture Screenshot* \| extra=_list \| # samurai_0000000002_*list*.png<br>Arbor.*Capture Screenshot* \| \| # arbor_0000000001.png<br>Arbor.*Capture Screenshot* \| extra=_xxx \| # arbor_0000000001_*xxx*.png<br>Samurai.*Capture Screenshot* \| file_name=1111.png \| # 1111.png |
| **Change Policy View Group** | *name*, **group_name* | Changes the groups that could see this policy<br>`name` is the policy name. `group_name` is a list of policies<br>Example:<br>Samurai.*Change Policy View Group* \| super_admin \| test_group001 |
| **Click All Elements** | *xpath* | Click all element in current page defined by `xpath`<br>Returns the number of elements that have been clicked |
| **Close** | | Closes the current active browser |
| **Close All** | | Closes all current opened applications |
| **Connect** | *app*, *name* | Opens a web browser and connects to application and assigns a `name`.<br>If not defined in `local.yaml` those following key will have defaut values:<br>browser \| firefox \| optional<br>login_url \| / \| optiona<br>proxy: \| \| optional<br>http: 10.128.8.210:8080 \| optional \|<br>ssl: 10.128.8.210:8080 \| optional \|<br>socks: 10.128.8.210:8080 \| optional \|<br>profile_dir \| ./config/samurai.profile \| optional |
| **Connect All** | | Connects to all applications defined in `local.yaml`<br>The name of the connection will be the same of the *webapp* name |
| **Delete Policy** | **policy_names* | Deletes poilcies by their names<br>Returned the number of deleted users<br>**Notes:** If the policy does not exists, the system will not report any error.<br>Examples:<br>Samurai.*Delete Policy* \| test001 \| test002 |
| **Delete Policy Group** | **group_list* | Deletes policy groups<br>Returns the number of deleted policy groups Example:<br>Samurai.*Delete Policy Group* \| test_group001 \| test_group002 |
| **Delete User** | *group*, **user_list* | Deletes user from the user group<br>`group` is the user group. And `.` means current group Returns the number of deleted users<br>Examples:<br>Samurai.*Delete User* \| SuperGroup \| user001 \| user002<br>Samurai.*Delete User* \| . \| user002 \| |

| | | |
|---|---|---|
| **Edit Policy** | ***policy* | Edits a Samurai policy<br>`policy` contains information about the policy. See *Add Policy* for more details about `policy` format |
| **Left Menu** | *menu* | Chooses the left panel menu by its displayed name<br>Return a list of 1st meaningful column Example:<br>Samurai.*Left Menu* \| Traffic<br>Samurai.*Left Menu* \| Detection<br>Samurai.*Left Menu* \| ポリシー管理 |
| **Login** | | Logs-in into the application<br>User and password is set by the template and authentication methods in the master files |
| **Logout** | | Logs-out the current application, the browser remains |
| **Make Item Map** | *xpath* | Makes a item/webelement defined *xpath*<br>The map is a dictionary from *item* to the *WebElement* Items name found by `xpath` are used as keys |
| **Reset Capture Counter** | | Resets the counter of the screen capture |
| **Select Items In Table** | *xpath*, *xpath2*, **item_list* | Checks items in Samurai table by xpath<br>`xpath` points to the column that used as key and `xpath2` is the relative xpath contains the checkbox column.<br>`item_list` is a list of item that need to check. Item in the list could be a regular expresion with the format `reg=<regular expression>`.<br>The keyword is called with assuming that the table is already visible.<br>Returns the tupple of all items and selected items<br>**Note:** Non-width-space (\u200b) will be take care by the keyword.<br>**Note:** if the first item_list is `*` then the keywork will try to click a link named *すべてを選択*. |
| **Set Capture Counter** | *value=0* | Sets the counter of the screen capture to `value` |
| **Set Capture Format** | *format* | Sets the format for the screen capture file<br>The format does not include the default prefix `.png` The default format is `<mod>_%010d`. `mod` could be `samurai` or `arbor`<br>See https://docs.python.org/2/library/string.html#format-specification-mini-language for more details about the format string.<br>Examples:<br>Samurai.*Set Capture Format* \| ${case}_%010d \| # ${case} is a predefined variable |
| **Show Policy Basic** | *policy_name* | Makes the virtual browser show basic setting of the policy *name*.<br>A following Samurai.*Capture Screenshot* is necessary to capture the result. |
| **Show Policy Mitigation** | *policy_name* | Make the virtual browser show the mitigation setting of a policy<br>A following Samurai.*Capture Screenshot* is necessary to capture the result. |
| **Show Policy Mo** | *policy_name* | Make the virtual browser show the MO setting of a policy<br>Automatically expand the MO section of other devices if necessary.<br>A following Samurai.*Capture Screenshot* is necessary to capture the result. |
| **Show Policy Monitor** | *policy_name* | Make a virtual browser show the mitigation setting of a policy<br>A following Samurai.*Capture Screenshot* is necessary to capture the result. |
| **Start Mitigation** | *policy*, *prefix*, *comment=mitigation started by RENAT*, *device=None*, *force=False* | Starts a mitigation with specific `prefix`<br>`device` is used for matching real device name configured by Samurai If `force` is `TRUE` then the keyword will fail if selected device does not contain `device`<br>Returns mitigation `id` and selected `arbor device`<br>Example:<br>${id} \| ${device}= \| Samurai.*Start Mitigation* \| 211.1.12.1/32 \| mitigation by RENAT \| SP-A \| ${TRUE} |
| **Stop Mitigation** | *id* | Stops a mitigation by its ID<br>Example:<br>Samurai.*Stop Mitigation* \| 700 |
| **Switch** | *name* | Switches the current browser to `name` |

Altogether 29 keywords.
Generated by Libdoc on 2018-03-20 02:58:14.

# Arbor

| | |
|---|---|
| **Library version:** | RENAT 0.1.7 |
| **Library scope:** | test suite |
| **Named arguments:** | supported |

## Introduction

A library provides functions to control Arbor application

The library utilize *Selenium2Library* and adds more functions to control Arbor application easily.

See WebApp for common keywords of web applications.

*Selenium2Library* keywords still could be used along with this library. See Selenium2Library for more details.

## Shortcuts

**C**apture Screenshot · **C**lose · **C**lose All · **C**onnect · **C**onnect All · **D**etail First Mitigation · **L**ogin · **L**ogout · **R**eset Capture Counter · **S**et Capture Counter · **S**et Capture Format · **S**et Count · **S**how All Mitigations · **S**how Detail Mitigation · **S**witch

## Keywords

| Keyword | Arguments | Documentation |
|---|---|---|
| **Capture Screenshot** | *filename=None*, *extra=* | Captures the current screen to file<br>Using the internal counter for filename if `filename` is not specified. In this case, the filename is defined by a pre-set format. *Set Capture Format* could be used to change the current format.<br>An extra information will be add to the filename if `extra` is defined<br>Examples:<br>Samurai.*Capture Screenshot* \| \| # samurai_0000000001.png<br>Samurai.*Capture Screenshot* \| extra=_list \| # samurai_0000000002_*list*.png<br>Arbor.*Capture Screenshot* \| \| # arbor_0000000001.png<br>Arbor.*Capture Screenshot* \| extra=_xxx \| # arbor_0000000001_*xxx*.png<br>Samurai.*Capture Screenshot* \| file_name=1111.png \| # 1111.png |
| **Close** | | Closes the current active browser |
| **Close All** | | Closes all current opened applications |
| **Connect** | *app*, *name* | Opens a web browser and connects to application and assigns a `name`.<br>Extra information could be added to the `webapp` sections likes `login_url`, `browser` or `profile_dir`. Default values are:<br>browser \| firefox<br>login_url \| /<br>profile_dir \| ./config/samurai.profile |
| **Connect All** | | Connects to all applications defined in `local.yaml`<br>The name of the connection will be the same of the *webapp* name |
| **Detail First Mitigation** | | Shows details about the 1st mitigation on the list |
| **Login** | | Logged-into the Arbor application |
| **Logout** | | Logs-out the current application, the browser remains |
| **Reset Capture Counter** | | Resets the counter of the screen capture |
| **Set Capture Counter** | *value=0* | Sets the counter of the screen capture to `value` |
| **Set Capture Format** | *format* | Sets the format for the screen capture file<br>The format does not include the default prefix `.png` The default format is `<mod>_%010d`. `mod` could be `samurai` or `arbor`<br>See https://docs.python.org/2/library/string.html#format-specification-mini-language for more details about the format string.<br>Examples:<br>Samurai.*Set Capture Format* \| ${case}_%010d \| # ${case} is a predefined variable |
| **Set Count** | *counter=0* | Sets current counter to `counter` |
| **Show All Mitigations** | | Shows all mitigations |
| **Show Detail** | *id* | |

| | | |
|---|---|---|
| **Mitigation** | | Shows detail information for a mitigation |
| **Switch** | *name* | Switches the current browser to `name` |

Altogether 15 keywords.
Generated by Libdoc on 2018-03-20 02:58:15.

# Tester

| | |
|---|---|
| **Library version:** | RENAT 0.1.7 |
| **Library scope:** | test suite |
| **Named arguments:** | supported |

## Introduction

A class provides keywords for controlling testers and traffic generators. It could load predefined traffic file, manipulate traffic item, start and stop traffic flows. It also could generate traffic reports ...

Tester information is stored in the active `local.yaml` likes this:

```
tester:
  tester01:
    device: ixnet03_8009
    config: vmx_20161129.ixncfg
    real_port:
      - chassis: 10.128.32.71
        card: 6
        port: 11
      - chassis: 10.128.32.71
        card: 6
        port: 9
```

where `device` is the tester defined in the master `device.yaml` file. If `real_port` does not exist, port remapping will not take place. Otherwise, port remapping will use the `real_port` information to reassign all existed ports and map to Ixia ports.

In this case, the order will be the order when user created the port in Ixia GUI. **Note:** User can always confirm the created order by `clear sorting` in Ixia GUI

Examples:

| | |
|---|---|
| Tester.*Connect All* | |
| Tester.*Switch* | tester01 |
| Tester.*Load And Start Traffic* | |
| *Sleep* | 30s |
| Tester.*Stop Traffic* | |

Time format used in this module is same with `time string` format of Robot Framework. For more details about this, see DateTime library of Robot Framework.

**Note:** See IxNet module, IxLoad module for details about keyword of each module.

## Shortcuts

**C**lose All · **C**onnect · **C**onnect All · **S**witch

## Keywords

| Keyword | Arguments | Documentation |
|---|---|---|
| **Close All** | | Closes all connections |
| **Connect** | *name* | Connect to the tester `name` |
| **Connect All** | | Connects to all testers |
| **Switch** | *name* | Switchs the current tester to `name` |

Altogether 4 keywords.
Generated by Libdoc on 2018-03-20 02:58:10.

# ixload

| | |
|---|---|
| **Library scope:** | global |
| **Named arguments:** | supported |

## Introduction

provides functions for IxLoad

RENAT runs a virtual IxLoad client locally in the background that connects to a Windows App server. Keywords from test case will send control messages to the client, which in turn will control the test ports.

Different to IxNetwork, an IxLoad test case usually stops within predefined time before `Stop Traffic` was called.

**Notes:** Ignore the *self* parameters when using those keywords.

## Shortcuts

**C**lose · **C**ollect Data · **G**et Test Report · **L**oad Config · **L**oad Traffic · **S**tart Traffic · **S**top Traffic

## Keywords

| Keyword | Arguments | Documentation |
|---|---|---|
| **Close** | *self* | Disconnects the current tester client |
| **Collect Data** | *self*, *prefix=*, *more_file=*, *ignore_not_found=True* | Collects all result data and save them to the current active `result` folder<br>A `prefix` will be automatically added to the file names.<br>Currently the follow data will be downloaded to the local machine<br>▪ HTTP_Server.csv<br>▪ HTTP Client.csv<br>▪ HTTP Client - Per URL.csv<br>▪ HTTP Server - Per URL.csv<br>▪ L2-3 Stats for Client Ports.csv<br>▪ L2-3 Stats for Server Ports.csv<br>▪ L2-3 Throughput Stats.csv<br>▪ Port CPU Statistics.csv<br>Extra files could be add by `more_file` which is a comma separated filename string<br>When `ignore_not_found` is True, the keyword will not terminate even when the expected file is not found. |
| **Get Test Report** | *self*, *prefix=* | Get the test report(PDF) and put it into the active result folder |
| **Load Config** | *self*, *config_name=* | Loads the test traffic defined by `config_name`<br>`file_path` is the path of the test file on the **remote** App server A path to a remote network drive could be use to load a config file on Renat server. |
| **Load Traffic** | *self*, *file_path* | |
| **Start Traffic** | *self* | Starts the test traffic |
| **Stop Traffic** | *self* | Stops the current running test<br>Returns the elapsed time in seconds |

Altogether 7 keywords.
Generated by Libdoc on 2018-03-20 02:58:13.

# ixnet

**Library scope:** global
**Named arguments:** supported

## Introduction

provides functions for IxNetwork

RENAT will connect to the App server and control the test ports. Test files and result will be insde the RENAT server.

In order to run RENAT test case with *IxLoad*, the *TCLServer* must be activated with *Administrator* privileges on the Ixia App server.

**Notes:** Ignore the *self* parameters when using those keywords.

## Shortcuts

**A**dd Port · **A**dd Quicktest · **A**pply Traffic · **C**hange Frame Rate · **C**hange Frame Rate Dynamic · **C**hange Frame Size · **C**lose · **C**ollect All Data · **C**ollect Data · **G**et All Test Result · **G**et Quicktest List · **G**et Quicktest Result · **G**et Quicktest Result Path · **G**et Test Report · **G**et Test Result · **L**oad And Start Traffic · **L**oad Config · **L**oad Traffic · **L**oss From File · **P**ing · **R**eset Config · **R**un Quicktest · **S**et All Traffic Item · **S**et Bgp Items · **S**et Bgp Neighbor · **S**et Capture Port · **S**et Traffic Item · **S**hould Be Pingable · **S**tart Capture · **S**tart Protocol · **S**tart Traffic · **S**top All Protocols · **S**top And Save Capture · **S**top Quicktest · **S**top Traffic · **W**ait Until Connected

## Keywords

| Keyword | Arguments | Documentation |
|---|---|---|
| **Add Port** | *self*, *force=True*, *time_out=2m*, *learn_time=2m* | Add ports using the information from active local config<br>• `time_out` is the wait time until port is connected (default is 2m)<br>• `learn_time` is the time waiting for arp to be learned (default is 2m)<br>Sample of local config \|tester:<br>`tester:`<br>`  device: ixnet03_8009`<br>`  config: quicktest.ixncfg`<br>`  real_port:`<br>`  - chassis: 10.128.4.41`<br>`  card: 4`<br>`  port: 3`<br>`  ip: 10.100.11.2`<br>`  mask: 24`<br>`  gw: 10.100.11.1`<br>`  - chassis: 10.128.4.41`<br>`  card: 4`<br>port: 4<br>`  ip: 10.100.14.2`<br>`  mask: 24`<br>`  gw: 10.100.14.1` |
| **Add Quicktest** | *self*, *name*, *test_type=rfc2544throughput*, *tx_mode=interleaved*, *clear_all=True* | Create a new Quicktest with default value<br>Type could be one of following: `rfc2544throughput`,*rfc2544frameLoss,rfc2544back2back* `Use Tester.Load Config` to load a customized quicktest<br>When `clear_all` is True, any existed quicktests will be cleared.<br>Transmit mode `tx_mode` takes following values: `interleaved` (default) or `sequential`. The mode should be identical with the transmit mod of the ports.<br>**Notes**: The keyword **does not** create necessary ports. It should be used with a existed configuration by Tester.*Load Config* or Tester.*Add Port* keyword. |
| **Apply Traffic** | *self* | Applies the current traffic configuration<br>**Note:** This is a blocking command |
| **Change Frame Rate** | *self*, *value*, *pattern=.** | Changes the frame rate<br>Parameter:<br>• `value`: value to set. Depends on the current configuration, this could be `percent line rate` or `bit per second` etc.<br>• `traffic_pattern`*: a regular expression to identify traffic item name, default is everything `.\**<br>` |
| **Change Frame Rate Dynamic** | *self*, *value*, *pattern=.** | Changes the traffic flow rate on-fly<br>No need to stop the running traffic to change the rate<br>Parameter:<br>• `value`: value to set. Depend on the current configuration, this could be `percent line rate` or `bit per second` etc.<br>• `pattern`*: a regular expression to identify traffic item name, default is everything `.\**<br>` |
| **Change Frame Size** | *self*, *type*, *value*, *pattern=.** | Changes the frame size<br>Parameter: |

| | | |
|---|---|---|
| | | ▪ `type` : could be `fixed size` , `increment_from`,`increment_step` or `increment_to`<br>▪ `value` : value to set<br>▪ traffic_pattern*: a regular expression to identify traffic item name, default is everything* `.*` |
| **Close** | *self* | Disconnects the current tester client |
| **Collect All Data** | *self*, *prefix=stat_* | Deprecated. Use |
| **Collect Data** | *self*, *view*, *prefix=stat_* | Depricated. Use *Get Test Result* |
| **Get All Test Result** | *self*, *prefix=stat_* | Collects all Ixia traffic data after traffic is stopped.<br>Results are CSV files that are stored in `result` folder. The prefix `prefix` is appended to the original view name |
| **Get Quicktest List** | *self* | Returns current loaded Quicktest list |
| **Get Quicktest Result** | *self*, *test_index=-1*, *prefix=*, *enable_all=True* | Get the result.csv file from the latest Quicktests<br>`test_index` is a index of the current Quicktest. `-1` means that last one. |
| **Get Quicktest Result Path** | *self*, *test_index=-1* | Returns the path of the newest run of a Quicktest<br>`test_index` is a index of the current Quicktest. `-1` means that last one. |
| **Get Test Report** | *self*, *local_name=ixnet_report.pdf*, *enable_all=True* | Generates and get report of the current active test in PDF format<br>`local_name` : name of the report on local machine. Default is `ixnet_report.pdf` |
| **Get Test Result** | *self*, *view*, *prefix=stat_* | Collects traffic data of a `view` and export to a CSV file in `result` folder<br>Currently, supported views are:<br>`Port Statistics` , `Global Protocol Statistics` , `BGP Aggregated Statistics` , `BGP Aggregated State Counts` , `OSPF Aggregated Statistics` , `OSPF Aggregated State Counts` , `OSPFv3 Aggregated Statistics` , `OSPFv3 Aggregated State Counts` , `L2-L3 Test Summary Statistics` , `Flow Statistics` , `Flow Detective` , `Data Plane Port Statistics` , `User Defined Statistics` , `Traffic Item Statistics`<br>Result were store as CSV files in `result` folder. If there is no valid data, view will be silently ignored<br>The prefix `prefix` is appended to the view name for the CSV file. |
| **Load And Start Traffic** | *self*, *wait_time1=10s*, *wait_time2=10s* | Combines *Load Traffic* and *Start Traffic* to one keyword. |
| **Load Config** | *self*, *config_name=*, *wait_time=2m*, *wait_time2=2m*, *apply=True*, *protocol=True*, *force=True*, *tx_mode=interleaved* | loads traffic configuration, applies and start protocol if necessary.<br>The config file name was defined in the *`local.yaml* which is a Ixia Network configuration file and located in the *config* folder of the test.<br>The keyword remap the vports to real port when data is specified in the local configuration file. For some reasons, the txMode is cleared when remapping happens. Use `tx_mode` to set the TxMode of the remapped ports.<br>Parameters:<br>▪ `apply` : applies traffic when `True` otherwise<br>▪ `protocol` : starts all protocols when `True` otherwise<br>▪ `force` : force to reclaim the ports when `True` otherwise<br>▪ `tx_mode` : *sequential* or *interleaved*(default)<br>▪ `wait_time` : wait time after applying protocols<br>▪ `wait_time2` : maximum wait time befor all ports become available. In<br>common case, this is calculated automatically so user does not need to change this value.<br>See Common for more details about the yaml configuration files. |
| **Load Traffic** | *self*, *wait_time=2m*, *wait_time2=2m*, *apply=True*, *protocol=True*, *force=True*, *tx_mode=interleaved* | |
| **Loss From File** | *self*, *file_name=Flow_Statistics.csv*, *tx_frame_i=3*, *frame_delta_i=5*, *time1_i=23*, *time2_i=24* | Returns `packet loss` by miliseconds and delta frame.<br>The calculation should be performed when traffic is stopped. The calculation supposed traffic is configured by frame per second |
| **Ping** | *self*, *dst_ip*, *src_port_index=0*, *src_intf_index=0* | Ping from Ixia to `dst_ip`<br>The keyword return the output string as it is. The return could be<br>`Port <portName>: ping failed: port not assigned`<br>`Response received from <sourceIp>/unknown . Sequence Number <sequenceNumber>`<br>`Ping request to <destinationIp>/unknown ip failed: <GenericPingError>/<error>: <genericError>unknown reason`<br>`Error: Couldn't find any source interface for Send Ping to <destinationIp> on <portName> Id <id>`<br>Error: Couldn't find any source IP for Send Ping to <destinationIp> on <portName> Id <id><br>Parameters:<br>▪ src_port_index: index of Ixia port (starts from 0)<br>▪ src_intf_index: index of interface insides the port (starts from 0)<br>Examples:<br>Tester.Ping 1.1.1.1 0 0<br>Tester.Ping 1.1.1.1 |

| | | |
|---|---|---|
| **Reset Config** | *self* | Clears current config and creates new blank config |
| **Run Quicktest** | *self*, *test_index=0*, *wait_until_finish=True* | Runs the Quicktest and wait until it finishes<br>**Warning**: it could take a long time to finish a quicktest |
| **Set All Traffic Item** | *self*, *enabled=True* | Enables/Disables all traffic items at once |
| **Set Bgp Items** | *self*, *port_index*, *neighbor_index*, *route_range_index*, *is_enable* | Enables/Disables BGP entry by a set of port,neighbor,route_range index<br>Parameters:<br>▪ port_index: index of the port<br>▪ neighbor_index: index of the neighbor or *<br>▪ route_range_index: index of the route range or `*'<br>▪ is_enable: ${TRUE} or ${FALSE}<br>Note<br>Examples: |
| **Set Bgp Neighbor** | *self*, **indexes*, ***kwargs* | Enables/Disables BGP entry by neighbor index<br>kwargs contains following parameters:<br>▪ indexes: is a list of index of BGP neighbor (index is started from zero)<br>▪ vport_index: is the target vport index<br>▪ enabled: TRUE or FALSE<br>Examples:<br>Tester.*Set BGP Item* \| 0 \| 1 \| vport_index=0 \| enabled=${FALSE}<br>Tester.*Set BGP Item* \| 0 \| 1 \| vport_index=1 \| enabled=${TRUE} |
| **Set Capture Port** | *self*, *data_mode=True*, *control_mode=True*, *port_index=0* | Capture packets for follow port<br>port_index: is a index of current test port (start from 0) data_mode: capture data packets and save in <intf>_HW.cap file control_mode: capture controls packets and save in <intf>_SW.cap file<br>**Note**: control_mode saves all control packets and data_mode only saves data packets.<br>**Note**: control_mode saves all control packets and data_mode only saves data packet<br>Examples:<br>Tester.*Set Capture* \| 0<br>Tester.*Set Capture* \| control_mode=${TRUE} \| 0 \| 1 |
| **Set Traffic Item** | *self*, **items*, ***kwargs* | Enables/Disables some traffic items items<br>Parameters:<br>▪ items: a list of Ixia traffic item name<br>▪ enabled: False or True ,the mode to set traffic item to, default is True (enabled)<br>**Note**: traffic item could be specified by ::<num> format. In this case the num is the order of traffic item count from zero.<br>Returns True if all items are set coordinately or otherwise<br>Examples:<br>Set Traffic Item \| Traffic Item 1 \| Traffic Item 2<br>Set Traffic Item \| @{item_list}<br>Set Traffic Item \| Traffic Item 1 \| enabled = ${FALSE} |
| **Should Be Pingable** | *self*, *dst_ip*, *src_port_index=0*, *src_intf_index=0* | Ping from Ixia and raise an error if ping fails<br>The keyword return *True* if succeeds |
| **Start Capture** | *self*, *wait_time=30s* | Start packet capture<br>Target ports are set by the configuration file or by [Set Capture] keyword |
| **Start Protocol** | *self*, *wait_time=1m* | Starts all protocols and wait for wait_time<br>Default wait_time is 1 minute. Make sure wait_time is big engouh to start all protocols. |
| **Start Traffic** | *self*, *wait_time=30s* | Starts the current traffic settiing and wait for wait_time.<br>**Note:** This is a asynchronus action. After called, the keyword finishes immediatly but it will take a while before traffic starts<br>By default the keyword will wait for 30 seconds. |
| **Stop All Protocols** | *self*, *wait_time=30s* | Stop all running protocols |
| **Stop And Save Capture** | *self*, *prefix=*, *wait_until_finish=True*, *monitor_interval=5s* | Stop current capture and save the resuls to folder specified by path<br>Captured files will be saved in current result folder with prefix appended in their names.<br>Examples:<br>Tester.*Start Capture*<br>Sleep \| 10s<br>Tester.*Stop And Save Capture* \| ${RESULT_FOLDER}/capture.zip |
| **Stop Quicktest** | *self*, *test_index=0* | Stops a running test |

| | | |
|---|---|---|
| **Stop Traffic** | *self*, *stop_protocol=False*, *wait_time=10s* | Stops the current traffic and wait for `wait_time`<br>Parameters:<br>▪ stop_protocol: if `True` also stops all running protocols<br>▪ wait_time: time to wait after apply the command |
| **Wait Until Connected** | *self*, *timeout_str=5m* | Waits until ports become enabled and connected |

Altogether 36 keywords.
Generated by Libdoc on 2018-03-20 02:58:13.