RENAT

Library version:	RENAT 0.1.11
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 to control L1 switches, includes mod_calient mod, mod_ntm mod
Router:
Library provides keywords to control routers, includes mod_juniper mod , mod_cisco mod and mod_gr mod
Tester:
Library provides keywords to control testers, includes mod_ixnet , mod_ixload and mod_ixbps
WebApp:
Common library for web application, includes 2 child libraries: Samurai and Arbor
Hypervisor:
Library provides keywords to control Hypervisor, included mod_vmware
LabKeyword:
Common lab keywords

Others

Changes:
Changes information

Choose each libraries for detail infomration and samples about keywords.

Shortcuts

Keywords

Keyword	Arguments	Documentation

Altogether 0 keywords.
Generated by Libdoc on 2018-12-06 13:39:53.

🔍

Common

Library version:	RENAT 0.1.11
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
- Variables
- Shortcuts
- Keywords

Configuration file

Global configuration

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

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

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

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

1. device.yaml: contains global device information

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

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

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

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

Samples:

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

2. template.yaml: contains device template information

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

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

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

Samples:

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

3. auth.yaml: contains authentication information

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

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

or

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

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

Sample:

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

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

Local Configuration

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

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

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

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

Usually the `local.yaml` has following parts:

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

Sample:

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

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

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

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

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

Variables

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

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

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

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

Shortcuts

Change Mod · **C**leanup Result · **C**lose Display · **C**onvert Html To Pdf · **C**ount Keyword · **C**ount Keyword Line · **C**ount Match Regexp · **C**reate Sequence · **C**sv Add · **C**sv Concat · **C**sv Create · **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 · **E**xplicit Run · **F**ile Md5 · **F**old Str · **F**ollow Syslog And Trap · **G**et Config Path · **G**et Config Value · **G**et File Without Error · **G**et Item Config Path · **G**et Item Name · **G**et Myid · **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**oad Plugin · **L**og · **L**og Csv · **L**og To Console · **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**creenshot · **S**et Multi Item Variable · **S**et Result Folder · **S**lack · **S**tart Display · **S**tr 2 Seq · **V**ersion · **W**ait

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
Close Display		Closes the opened display
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:

<table>
<tr><td></td><td></td><td>
<table><tr><td>@{list}=</td><td>Create Sequence</td><td>10</td><td>15</td><td>0.5</td></tr></table>
will create a list of `[11.0, 11.5, 12.0, 12.5, 13.0, 13.5, 14.0, 14.5]`
</td></tr>
<tr><td>**Csv Add**</td><td>*pathname*, **items*</td><td>Add more data define by a list *items* to a existed CSV file
Note:: do not check the consistency between item's number and header's number</td></tr>
<tr><td>**Csv Concat**</td><td>*src_pattern*, *dst_name*, *input_header=None*, *result_header=True*</td><td>Concatinates CSV files vertically If the CSV files has header, set `has_header` to `${TRUE}`
Examples:
<table>
<tr><td>Commmon.CSV Concat</td><td>config/data0[3,4].csv</td><td>result/result2.csv</td><td></td></tr>
<tr><td>Commmon.CSV Concat</td><td>config/data0[3,4].csv</td><td>result/result2.csv</td><td>has_header=${TRUE}</td></tr>
</table>
</td></tr>
<tr><td>**Csv Create**</td><td>*pathname*, **header*</td><td>Create a CSV file with headers defined by a list *header*
The CSV file is opend with *UTF-8* encoding mode</td></tr>
<tr><td>**Csv Merge**</td><td>*src_pattern*, *dst_name*, *input_header=None*, *key=0*, *select_column=:*, *result_header=True*</td><td>Merges all CSV files `horizontally` by `key` key from `src_pattern`
`input_header` defines whether the input files has header row or not. If `input_header` is `${NULL}`, the keyword assume that input files have no header and automatically define columns name. When `input_header` is not null (default is zero), the row define by `input_header` will be used as header and data is counted from the next row.
`select_column` is a string that define the output columns and `key` is the column name that used to merge. When `input_header` is `${NULL}`, `select_column` and *key* is the index of columns. Otherwise, they are *column name*.
The result header (column names) is decided by `result_header` (*True* or *False*)
The keyword returns `False` if no file is found by the pattern
Examples:
<table>
<tr><td>Common.CSV Merge</td><td>config/data0[3,4].csv</td><td>result/result2.csv</td><td></td></tr>
<tr><td>Common.CSV Merge</td><td>config/data0[3,4].csv</td><td>result/result2.csv</td><td>input_header=0</td></tr>
<tr><td>Common.CSV Merge</td><td>src_pattern=${RESULT_FOLDER}/balance*.csv</td><td>input_header=0</td><td></td></tr>
<tr><td>...</td><td>dst_name=${RESULT_FOLDER}/result.csv</td><td>result_header=${FALSE}</td><td></td></tr>
<tr><td>...</td><td>key=Stat Name</td><td>select_column=Valid Frames Rx.</td><td></td></tr>
<tr><td>Common.CSV Merge</td><td>src_pattern=${RESULT_FOLDER}/balance*.csv</td><td>input_header=${NULL}</td><td></td></tr>
<tr><td>...</td><td>dst_name=${RESULT_FOLDER}/result.csv</td><td>result_header=${FALSE}</td><td></td></tr>
<tr><td>...</td><td>key=0</td><td>select_column=5</td><td></td></tr>
</table>
</td></tr>
<tr><td>**Csv Select**</td><td>*src_file*, *dst_file*, *str_row=:*, *str_col=:*, *has_header=None*</td><td>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:
<table>
<tr><td>CSV Select</td><td>result/data05.csv</td><td>result/result3.csv</td><td>0,1,2</td><td>0,1</td></tr>
<tr><td>CSV Select</td><td>result/data05.csv</td><td>result/result4.csv</td><td>:</td><td>0,1</td></tr>
<tr><td>CSV Select</td><td>result/data05.csv</td><td>result/result5.csv</td><td>:2</td><td>:</td></tr>
<tr><td>CSV Select</td><td>result/data05.csv</td><td>result/result6.csv</td><td>0:3</td><td>:</td></tr>
<tr><td>CSV Select</td><td>result/data05.csv</td><td>result/result7.csv</td><td>0:5:2</td><td>:</td></tr>
</table>
</td></tr>
<tr><td>**Diff File**</td><td>*path1*, *path2*, *newline=True*</td><td>Shows difference between files
Returns the diff result (multi lines) `path1`, `path2` are absolute paths.</td></tr>
<tr><td>**Err**</td><td>*msg*</td><td>Prints error `msg` to console</td></tr>
<tr><td>**Error Line Should Not Be Bigger Than**</td><td>*num*, **pattern_list*</td><td>Checks whether the number of lines that contains `error` be less than a number</td></tr>
<tr><td>**Error Should Not Be Bigger Than**</td><td>*num*, **pattern_list*</td><td>Checks whether the number of `error` be less than a number</td></tr>
<tr><td>**Explicit Run**</td><td></td><td>skip the test case if global_variable RUN_ME is not defined
Sample scenario:
<table>
<tr><td>00. Cabling</td><td></td></tr>
<tr><td>Common.Explicit Run</td><td></td></tr>
<tr><td>Log To Console</td><td>cabling...</td></tr>
</table>
`run.sh` will bypass `00. Cabling` by default. In other to run this test case *${FORCE}* needs declared globally `run.sh -X -v FORCE`
</td></tr>
</table>

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

<table>
<tr><th></th><th></th><th></th></tr>
<tr><td></td><td></td><td>   device: vmx11
    snmp_polling: yes
    mib: mib11.txt
...

Default value is defined by <code>mib</code> keyword from global <code>config/snmp-template.yaml</code> for the <code>type</code> of the node

Example:<table><tr><td>${mib}=</td><td>Common.MIB For Node</td><td>vmx11</td></tr></table></td></tr>
<tr><td><b>Node With Attr</b></td><td><i>attr_name</i>, <i>value</i></td><td>Returns a list of nodes which have attribute <code>attr_name</code> with value <code>value</code></td></tr>
<tr><td><b>Node With Tag</b></td><td><i>*tag_list</i></td><td>Returns list of <code>node</code> or <code>webapp</code> from <code>local.yaml</code> that has <b>ALL</b> tags defined by <code>tag_list</code>

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></td></tr>
<tr><td><b>Node Without Tag</b></td><td><i>*tag_list</i></td><td>Returns list of <code>node</code> from <code>local.yaml</code> that <b>does not has ANY</b> tags defined by <code>tag_list</code>

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></td></tr>
<tr><td><b>Pause</b></td><td><i>msg=</i>, <i>time_out=3h</i>, <i>error_on_timeout=True</i>, <i>default_input=</i></td><td>Displays the message <code>msg</code> and pauses the test execution and wait for user input

In case of <code>error_on_timeout</code> is True(default), the keyword will raise an error when timeout occurs. Otherwise, it will continue the test.

<b>Notes:</b> If the variable <code>${RENAT_BATCH}</code> 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></td></tr>
<tr><td><b>Ping Until Ok</b></td><td><i>node</i>, <i>wait_str=5s</i>, <i>extra=-c 3</i></td><td>Ping a <code>node</code> until it gets response. Then wait for more <code>wait_str</code> Default <code>extra</code> option is <code>-c 3</code></td></tr>
<tr><td><b>Random Name</b></td><td><i>base</i>, <i>a=0</i>, <i>b=99</i></td><td>Returns a random name by a <i>base</i> and a random number between [a,b]

Example:<table><tr><td>${FOLDER}=</td><td>Random Name</td><td>capture_%05d</td><td>0</td><td>99</td></tr></table></td></tr>
<tr><td><b>Random Number</b></td><td><i>a=0</i>, <i>b=99</i></td><td>Returns a random number between [a,b]</td></tr>
<tr><td><b>Renat Version</b></td><td></td><td>Returns RENAT version string</td></tr>
<tr><td><b>Screenshot</b></td><td><i>file_path</i></td><td>Capture whole display to a file specified by <code>file_path</code>

<b>Notes</b>: This keyword saves the whole virtual screen(monitor), while the familiar WebApp.<i>Screenshot Capture</i> only saves the portion of the web browser. But in contrast, the WebApp.<i>Screenshot Capture</i> could do <i>fullpage capture</i> depending on the content of the browser.</td></tr>
<tr><td><b>Set Multi Item Variable</b></td><td><i>*vars</i></td><td>Set multiple varibles to be <i>suite variable</i> at the same time

Suite variables (or item variable) could be access anywhere in all the item scenario.</td></tr>
<tr><td><b>Set Result Folder</b></td><td><i>folder</i></td><td>Sets the result folder to <code>folder</code> and return the old result folder. The result folder contains all output files from the test likes tester ouput, config file ...

<code>folder</code> 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 <i>0775</i>

<b>Note:</b> Result folder should be set at the begining of the test. Changing result folder only has effect on up comming connection</td></tr>
<tr><td><b>Slack</b></td><td><i>msg</i>, <i>channel=#automation_dev</i>, <i>user=renat</i>, <i>host=10.128.3.103:4713</i></td><td>Post a message to Slack</td></tr>
<tr><td><b>Start Display</b></td><td></td><td>Starts a virtual display</td></tr>
<tr><td><b>Str 2 Seq</b></td><td><i>str_index</i>, <i>size</i></td><td>Returns a sequence from string format

Samples:<table><tr><td>Str2Seq</td><td>::</td><td>5</td><td># (0,1,2,3,4)</td></tr><tr><td>Str2Seq</td><td>:2</td><td>5</td><td># (0,1)</td></tr><tr><td>Str2Seq</td><td>1:3</td><td>5</td><td># (1,2)</td></tr><tr><td>Str2Seq</td><td>0:5:2</td><td>5</td><td># (0,2,4)</td></tr></table></td></tr>
<tr><td><b>Version</b></td><td></td><td>Returns the current version of RENAT</td></tr>
<tr><td><b>Wait</b></td><td><i>wait_time</i>, <i>size=10</i></td><td>Waits for <i>wait-time</i> and display the proress bar</td></tr>
</table>

wait_time used RF *DateTime* format.

Examples:

Common.*Wait*	wait_time=30s	size=10

Altogether 58 keywords.
Generated by Libdoc on 2018-12-06 13:39:38.

VChannel

Library version:	RENAT 0.1.11
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*
- *Connections*
- *Shortcuts*
- *Keywords*

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

Change Log · Change Prompt · Close · Close All · Cmd · Cmd And Wait For · Cmd And Wait For Regex · Cmd More · Cmd Yesno · Connect · Connect All · Current Prompt · Exec File · Flush All · Get Channel · Get Channels · Get Current Channel · Get Current Name · Get Ip · Log · Read · Reconnect · Set Log Separator · Snap · Snap Diff · Start Screen Mode · Stop Screen Mode · Switch · Write

Keywords

<table>
<tr><th>Keyword</th><th>Arguments</th><th>Documentation</th></tr>
<tr><td><b>Change Log</b></td><td><i>log_file</i>, <i>mode=w</i></td><td>Stops current log file and create a new log file.
Every log from that point will be saved to the new log file Return old log filename</td></tr>
<tr><td><b>Change Prompt</b></td><td><i>str_prompt</i></td><td>Changes the current prompt of the channel
Returns previous prompt. User should change the prompt <code>before</code> execute the new command that expects to see new prompt.
Example:
<table>
<tr><td>Router.<i>Switch</i></td><td>vmx11</td><td></td></tr>
<tr><td>${prompt}=</td><td>VChannel.<i>Change Prompt</i></td><td>%</td></tr>
<tr><td>VChannel.<i>Cmd</i></td><td>start shell</td><td></td></tr>
<tr><td>VChannel.<i>Cmd</i></td><td>ls</td><td></td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td></td><td><table><tr><td>VChannel.Change Prompt</td><td>${prompt}</td><td></td></tr><tr><td>Vchannel.Cmd</td><td>exit</td><td></td></tr></table></td></tr>
<tr><td><b>Close</b></td><td></td><td>Closes current connection and returns the active channel name</td></tr>
<tr><td><b>Close All</b></td><td></td><td>Closes all current sessions and flush out all log files.
Current node name was reset to `None`</td></tr>
<tr><td><b>Cmd</b></td><td><i>command=</i>, <i>prompt=None</i>, <i>timeout=None</i>, <i>match_err= (unknown command.|syntax error, expecting <command&gt;.)</i></td><td>Executes a `command` and wait until for the prompt.
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`
<i>timeout</i> is the timeout for this <i>Cmd</i>. If <i>timeout</i> is not define, the local <i>vchannel/cmd-timeout</i> or global <i>vchannel/cmd-timeout</i> will be used.
The keyword returns error when the output matches the `match_err` and the default config value <i>cmd-auto-check</i> is `True`
Output will be automatically logged to the channel current log file.
See Common for details about the config files.</td></tr>
<tr><td><b>Cmd And Wait For</b></td><td><i>command</i>, <i>keyword</i>, <i>interval=30s</i>, <i>max_num=10</i>, <i>error_with_max_num=True</i></td><td>Execute a command and expect `keyword` occurs in the output. If not wait for `interval` and repeat the process again
After `max_num`, if `error_with_max_num` is `True` then the keyword will fail. Ortherwise the test continues.</td></tr>
<tr><td><b>Cmd And Wait For Regex</b></td><td><i>command</i>, <i>pattern</i>, <i>interval=30s</i>, <i>max_num=10</i>, <i>error_with_max_num=True</i></td><td>Execute a command and expect `pattern` occurs in the output. If not wait for `interval` and repeat the process again
When the keyword contains `not:` at the beginning, the matching logic is revsersed.
After `max_num`, if `error_with_max_num` is `True` then the keyword will fail. Ortherwise the test continues.</td></tr>
<tr><td><b>Cmd More</b></td><td><i>cmd=</i>, <i>wait_prompt=.*---\(more.*\)---</i>, <i>press_key=</i> , <i>prompt=None</i></td><td>Execute a command and press <i>press_key</i> when <i>wait_prompt</i> is displayed until the prompt</td></tr>
<tr><td><b>Cmd Yesno</b></td><td><i>cmd</i>, <i>ans=yes</i>, <i>question=? [yes,no]</i> , <i>timeout=5s</i></td><td>Executes a `cmd`, waits for `question` and answers that by `ans`</td></tr>
<tr><td><b>Connect</b></td><td><i>node</i>, <i>name</i>, <i>log_file</i>, <i>timeout=20m</i>, <i>w=80</i>, <i>h=32</i>, <i>mode=w</i></td><td>Connects to the node and create a VChannel instance
Login information is automatically extracted from yaml configuration. By defaullt a virtual terminal (vty100) with size 80x64 is attachted to this channel.
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.
Multi sessions to the same node could be open with different names. Use <i>Switch</i> to change the current active session by its name
Examples:<table><tr><td>Connect</td><td>vmx11</td><td>vmx11</td><td>vmx11.log</td><td></td><td></td></tr><tr><td>Connect</td><td>vmx11</td><td>vmx11</td><td>vmx11.log</td><td>80</td><td>64</td></tr></table>See `Common` for more detail about the yaml config files.</td></tr>
<tr><td><b>Connect All</b></td><td><i>prefix=</i></td><td>Connects to <b>all</b> nodes that are defined in active `local.yaml`.
A prefix `prefix` was appended to the alias name of the connection. A new log file by `<alias>.log` was automatiocally created.
See <i>Common</i> for more detail about active `local.yaml`</td></tr>
<tr><td><b>Current Prompt</b></td><td></td><td>Return current prompt</td></tr>
<tr><td><b>Exec File</b></td><td><i>file_name</i>, <i>vars=</i>, <i>comment=#</i> , <i>step=False</i>, <i>str_error=syntax,rror</i></td><td>Executes commands listed in `file_name` Lines started with `comment` character is considered as comments
`file_name` is a file located inside the `config` folder of the test case.
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`.
`vars` has the format: `var1=value1,var2=value2`
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 <i>error</i>
A sample for command list with Jinja2 template:
<pre>show interface {{ LOCAL['extra']['line1'] }}
show interface {{ LOCAL['extra']['line2'] }}</pre></td></tr>
</table>

		{% for i in range(2) %} show interface et-0/0/{{ i }} {% endfor %} Examples: Router.*Exec File* cmd.lst Router.*Exec File* step=${TRUE} str_error=syntax,error **Note:** Comment in the middle of the line is not supported For example if `comment` is "# " # 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
Flush All		Flushes all remain data into the logger
Get Channel	*name*	Returns a channel by its `name`
Get Channels		Returns all current vchannel instances
Get Current Channel		Returns the current active channel
Get Current Name		Returns the current active channel's name
Get Ip		Returns the IP address of current node Examples: ${router_ip}= Router.*Get IP*
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. 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 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
Snap	*name*, **cmd_list*	Remembers the result of a list of command defined by `cmd_list` Use this keyword with *Snap Diff* to get the difference between the command's result. The a new snapshot will overrride the previous result. 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. Difference is in `context diff` format
Start Screen Mode		Starts the `screen mode`. 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` 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 Returns the current *channel_id*, *local_channel_id* and the output of current terminal. **Notes:** There is no assurance that the output of previous *Write* command will be in the retur output because keywords like Logger.*Log All* will update every channels. Examples: VChannel.*Switch* vmx12
Write	*str_cmd=*, *str_wait=0s*, *start_screen_mode=False*	Sends `str_cmd` to the target node and return after `str_wait` time. If `start_screen_mode` is `True`, the channel will be shifted to `Screen Mode`. Default value of `screen_mode` is False. 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. 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. Parameters: ▪ `str_cmd`: the command ▪ `str_wait`: time to wait after apply the command ▪ `start_screen_mode`: whether start the `screen mode` right after writes the command

Special input likes Ctrl-C etc. could be used with global variable ${CTRL-<char>}

Returns the output after writing the command the the channel.

When *str_wait* is not *0s*, the keyword `read` and `return the output` after waiting *str_wait*. Otherwise, the keyword return without any output.

Notes: This is a non-blocking command.

Examples:

VChannel.*Write*	monitor interface traffic	start_screen_mode=${TRUE}
VChannel.*Write*	${CTRL_C}	# simulates Ctrl-C

Altogether 29 keywords.
Generated by Libdoc on 2018-12-06 13:39:42.

Logger

Library version:	RENAT 0.1.11
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.

Notes: log file is not updated pararelly. Anytime a terminal is switched to, it will update its log file.

Shortcuts

Log · Log All · Switch

Keywords

<table>
<tr><th>Keyword</th><th>Arguments</th><th>Documentation</th></tr>
<tr><td><b>Log</b></td><td><i>msg</i>, <i>with_time=False</i>, <i>mark=***</i></td><td>Inserts a message <code>msg</code> to the current <i>VChannel</i> log file. A default mark of <code>***</code> will be added at the beginning ant the end of this message.
Example:<table><tr><td>Logger.<i>Log</i></td><td>START TRAFFIC FROM HERE</td><td>${TRUE}</td><td></td></tr><tr><td>Logger.<i>Log</i></td><td>START TRAFFIC FROM HERE</td><td>${False}</td><td>===</td></tr></table></td></tr>
<tr><td><b>Log All</b></td><td><i>msg</i>, <i>with_time=False</i>, <i>mark=***</i></td><td>Inserts a message <code>msg</code> to current <b>all</b> VChannel log files.
A default <code>mark</code> of <code>***</code> and newline will be added at the beggining and the end of this message.
Example:<table><tr><td>Logger.<i>Log All</i></td><td>START TRAFFIC FROM HERE</td><td>${TRUE}</td><td></td></tr><tr><td>Logger.<i>Log All</i></td><td>START TRAFFIC FROM HERE</td><td>${TRUE}</td><td>===</td></tr></table>The log file will look likes this:
user@vmx12>

*** 06:01PM on August 13, 2017: START TRAFFIC FROM HERE ***

=== 06:01PM on August 13, 2017: START TRAFFIC FROM HERE ===

configure</td></tr>
<tr><td><b>Switch</b></td><td><i>name</i></td><td>Switches the current VChannel instance to <code>name</code>. <code>name</code> is the name of the VChannel (usually is the node name defined in the current active <code>local.yaml</code>).
Example:<table><tr><td>Logger.<i>Switch</i></td><td>vmx11</td></tr></table></td></tr>
</table>

Altogether 3 keywords.
Generated by Libdoc on 2018-12-06 13:39:38.

OpticalSwitch

Library version:	RENAT 0.1.11
Library scope:	test suite
Named arguments:	supported

Introduction

A library provides control for L1 Optical Switch

Unlike other device, there is no *Switch* keywork with optical switch. Usually user only need to care about the interfaces not the ports of the switches.

Shortcuts

Add · **C**lear By File · **C**lose All · **C**onnect All · **D**elete · **G**et Connection Info · **L**oad From File · **S**ave To File

Keywords

Keyword	Arguments	Documentation
Add	*dev1*, *intf1*, *dev2*, *intf2*, *direction=bi*, *force=False*	Adds a connection. See details in each module help
Clear By File	*file_name=*, *comment=#*	Clears all x-connections defined in the *connection file* Default *connection file* is defined in `optic/connection` of `config/local.yaml`
Close All		Close all connections
Connect All		Connect to all L1 switch and read all neccesary information
Delete	*dev1*, *intf1*, *dev2*, *intf2*, *direction=bi*, *force=False*	Deletes a connection. See details in each module help
Get Connection Info	*dev*, *intf*	Returns connection information. See details in each module help.

Load From File	*file_name=*, *force=True*, *comment=#*	Loads the connection file and set the connections
		`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.
		The `connection file` supports `jinja2` template language. Besides, `#` is the default comment char which could be changed
		The format of `connection file` follows:
		- each connection is described by 1 line
		- `source` and `destination` are separated by ` *- or > , which mean* `bidirection` or `unidirection` (unidirection connects `source tx` to `dest rx`
		Connection file sample:
		`device1:port1 - device2:port2`
		`device1:port3 > device2:port`
		Examples:
		OpticalSwitch.*Load From File*
		OpticalSwitch.*Load From File* | save1.conn

Save To File	*file_name*	Saves the current connection of all devices in this test. By default, all interfaces of the devices are save. If a connection file is given, only interfaces specified in the connection file are saved Examples: OpticalSwitch.*Save To File* | save1.conn

Altogether 8 keywords.
Generated by Libdoc on 2018-12-06 13:39:41.

calient

Library scope: global
Named arguments: supported

Introduction

A library provides control for Calient Optical Switch

Table of Contents

- *Master file*
- *Connection file Format*
- *Shortcuts*
- *Keywords*

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. X-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

Add · Delete · Get Connection Info

Keywords

<table>
<tr><th>Keyword</th><th>Arguments</th><th>Documentation</th></tr>
<tr><td><b>Add</b></td><td><i>self</i>, <i>dev1</i>, <i>intf1</i>, <i>dev2</i>, <i>intf2</i>, <i>direction=bi</i>, <i>force=False</i></td><td>Adds x-connection between <code>dev1:intf1</code> and <code>dev2:intf2</code>
<code>direction</code> is <code>bi</code> for bi-direction or <code>uni</code> for uni-direction. If <code>direction</code> is <code>uni</code>, the tx of <code>dev 1:port 1</code> will be connected to <code>dev 2:port 2</code>.
With <code>force</code> mode, existed connection that use those ports will be deleted. Without <code>force</code> mode, an existed connection will make the keyword fails
Examples:
<table><tr><td>OpticalSwitch.<i>Add</i></td><td>mx2008-31-33</td><td>xe-3/0/0</td><td>mx2008-31-33</td><td>xe-3/0/1</td><td>bi</td><td>${TRUE}</td></tr></table><b>Note</b>: when <code>force</code> is <code>False</code> but the current ports is owned by the same connection endpoints, keyword will succeed.
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.</td></tr>
<tr><td><b>Delete</b></td><td><i>self</i>, <i>dev1</i>, <i>intf1</i>, <i>dev2</i>, <i>intf2</i>, <i>direction=bi</i>, <i>force=False</i></td><td>Deletes the connection between <code>dev1:intf1 - dev2:intf2</code>
Examples:
<table><tr><td>OpticalSwitch.<i>Delete</i></td><td>mx2008-31-33</td><td>xe-3/0/1</td><td>mx2008-31-33</td><td>xe-3/0/1</td><td>uni</td></tr></table></td></tr>
<tr><td><b>Get Connection Info</b></td><td><i>self</i>, <i>dev</i>, <i>intf</i></td><td>Returns information of the optic switch port that connected to <code>dev:intf</code>. The information is in jason format.
Examples:
<table><tr><td>OpticalSwitch.<i>Get Connection Info</i></td><td>mx2008-31-33</td><td>xe-3/0/1</td></tr></table>return information looks like below:
<pre>result = {u'outoc': u'NOHW', u'outopwdh': u'-20.0', u'inos': u'OOS',
u'outalias': u'', u'inowner': u'TRANSIT', u'outopwct': u'-23.0', u'inpower':
u'-3.4', u'inas': u'IS', u'outpower': u'-4.8', u'outas': u'OOS-NP', u'inopt':
u'-17.0', u'inopth': u'13.0', u'incircuit': u'3.3.1>3.3.2', u'inalias': u'',</pre></td></tr>
</table>

		u'inoc': u'NOHW', u'inoptc': u'-20.0', u'outos': u'OOS', u'port': u'3.3.1', u'outowner': u'NONE', u'outcircuit': u''}

Altogether 3 keywords.
Generated by Libdoc on 2018-12-06 13:39:46.

g4ntm

Library scope:	global
Named arguments:	supported

Introduction

A library provides control for Telescent Network Topology Management (NTM) robot patch.

Shortcuts

Add · **D**elete · **G**et Connection Info

Keywords

Keyword	Arguments	Documentation
Add	*self*, *dev1*, *intf1*, *dev2*, *intf2*, *direction=bi*, *force=False*	
Delete	*self*, *dev1*, *intf1*, *dev2*, *intf2*, *direction=bi*, *force=False*	Deletes the connection between `dev1:intf1 - dev2:intf2`
Get Connection Info	*self*, *dev*, *intf*	Returns information about the connection by router/interface

Altogether 3 keywords.
Generated by Libdoc on 2018-12-06 13:39:47.

Router

Library version:	RENAT 0.1.11
Library scope:	test suite
Named arguments:	supported

Introduction

A class provides keywords for router control. An instance of Router class automatically assigned methods of a VChannel class (**Note**: this is not an inheritance but rather 1-to-1 relation)

See VChannel for more details about *VChannel*.

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.*Switch*	vmx12
Router.*Xrun*	Load Config
Router.*Load Config*	

Shortcuts

Follow Mib · Xrun

Keywords

Keyword	Arguments	Documentation
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`. Stableness is checked by SNMP polling result. The MIB list is define by `mib` in `node` section Parameter: - `wait_time(1)`: the time before the evaluation starting - `interval_time(2)`: interval between SNMP polling time - `threshold`: below this value is evaluated as `stable` - `len(3)`: the size of the evaluation window (number of values that are used in each valuation) - `percentile`: real useful percentage of data (ignore top `100-percentile` percent) - `max_len(4)`: maximum waiting `lend` for this checking time sequence: --(1)--|-(2)-|-----|-----|----|-----|-----| <--------(3)----------> poll poll <--------(3)----------> <--------------------(4)---------->
Xrun	*cmd*, **args*, ***kwargs*	Runs the vendor independent keywords. Parametes: - `cmd`: a keyword - `args`: other argumemts Examples: Router.*Xrun* | Flap Interface | ge-0/0/0 This keyword will then actually calling the correspond keyword for the device type.

Altogether 2 keywords.
Generated by Libdoc on 2018-12-06 13:39:39.

cisco

Library scope:	global
Named arguments:	supported

Introduction

Documentation for test library `cisco`.

Shortcuts

Get User · **G**et Version

Keywords

Keyword	Arguments	Documentation
Get User	*self*	Return the current login user
Get Version	*self*	return router version information

Altogether 2 keywords.
Generated by Libdoc on 2018-12-06 13:39:43.

gr

Library scope:	global
Named arguments:	supported

Introduction

Provides keywords for Hitachi GR platform

Shortcuts

Get 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 2018-12-06 13:39:43.

juniper

Library scope:	global
Named arguments:	supported

Introduction

Provides keywords for Juniper platform

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

Shortcuts

Create 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 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` 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. 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. 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`
		`time_format` decides the time part of the output. Example result are :
		`May 24 04:14:25`
		`May  4 04:14:25`
		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 ▪ `src_file` is a absolute path insides the router ▪ `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 `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` `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=(error:)*	Loads configuration to a router. Usable `mode` is `set`, `override`, `merge` and `replace` `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. 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 ... If the loading has no error that match the `error_match`, the configuration will be commited. 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 2018-12-06 13:39:43.

WebApp

Library version:	RENAT 0.1.11
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. The webapp device has following format:

```
<test node name>:
   device:    <device name>
   proxy:
      http:     <proxy for http>
      https:    <proxy for http>
      ssl:      <proxy for http>
```

Where `<device name>` is defined in master `device.yaml`, `proxy` section could be optional.

In order to download files from link, a *dowload-path* and MIME of that download need to be listed in *profile* section of the application setting. Default *download-path* is the current active result folder.

Samples:

```
...
webapp:
   samurai-1:
      device: samurai-b
```

profile: auto-save-mime: application/octet-stream

```
      proxy:
         http:   10.128.8.210:8080
         ssl:    10.128.8.210:8080
   arbor-1:
      device: arbor-sp-a
```

profile: auto-save-mime: application/xml,application/octet-stream

```
      proxy:
         http:   10.128.8.210:8080
         ssl:    10.128.8.210:8080
...
```

Selenium2Library keywords still could be used along with this library like this:

Click Link	//a[contains(.,'ユーザ設定')]
Sleep	2s
Click Link	Home設定
Sleep	2s
Samurai.Capture Screenshot	

See Selenium2Library for more details.

The module *Samurai* and *Arbor* based on this module. See Arbor, Samurai for details about keywords of each application.

Shortcuts

Capture Screenshot · **C**lose · **C**lose All · **G**et Verbose · **M**ark Element · **O**pen Ff With Profile · **R**eset Capture Counter · **S**et Ajax Wait · **S**et Capture Counter · **S**et Capture Format · **S**et Verbose · **S**witch · **V**erbose Capture · **W**ait Until Element Changes

Keywords

<table>
<tr><th>Keyword</th><th>Arguments</th><th>Documentation</th></tr>
<tr><td><b>Capture Screenshot</b></td><td><i>filename=None</i>, <i>extra=</i></td><td>
<p>Captures the current screen to file</p>
<p>Using the internal counter for filename if <code>filename</code> is not specified. In this case, the filename is defined by a pre-set format. <i>Set Capture Format</i> could be used to change the current format.</p>
<p>An extra information will be add to the filename if <code>extra</code> is defined</p>
<p>Examples:</p>
<table>
<tr><td>Samurai.<i>Capture Screenshot</i></td><td></td><td># samurai_0000000001.png</td></tr>
<tr><td>Samurai.<i>Capture Screenshot</i></td><td>extra=_list</td><td># samurai_0000000002_<i>list</i>.png</td></tr>
<tr><td>Arbor.<i>Capture Screenshot</i></td><td></td><td># arbor_0000000001.png</td></tr>
<tr><td>Arbor.<i>Capture Screenshot</i></td><td>extra=_xxx</td><td># arbor_0000000001_<i>xxx</i>.png</td></tr>
<tr><td>Samurai.<i>Capture Screenshot</i></td><td>filename=1111.png</td><td># 1111.png</td></tr>
</table>
</td></tr>
</table>

Close		Close the web application
Close All		Closes all current opened applications
Get Verbose		Get current verbose mode
Mark Element	*xpath*	Marking an element to check its stattus later
Open Ff With Profile	*app*, *name*	
Reset Capture Counter		Resets the counter of the screen capture
Set Ajax Wait	*wait_time=2s*	Set the ajax wait time
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 The format does not include the default prefix `.png` The default format is `<mod>_%010d`. `mod` could be `samurai` or `arbor` See https://docs.python.org/2/library/string.html#format-specification-mini-language for more details about the format string. Examples: Samurai.*Set Capture Format* ${case}_%010d # ${case} is a predefined variable
Set Verbose	*verbose=False*	Set current verbose mode to `verbose`
Switch	*name*	Switches the current browser to `name`
Verbose Capture	**args*, ***kwargs*	Capture screenshot if verbose mode is `True` otherwise do nothing
Wait Until Element Changes	*interval=5s*, *timeout=180s*, *error_on_timeout=False*	Wait until the marked element has been changed

Altogether 14 keywords.
Generated by Libdoc on 2018-12-06 13:39:48.

Samurai

Library version:	RENAT 0.1.11
Library scope:	test suite
Named arguments:	supported

Introduction

A library provides functions to control Samurai application

The library utilize *SeleniumLibrary* 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 connec to 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.

Currently the module support Samurai 09/14/16

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 and how to configure the `local.yaml` file.

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

Shortcuts

Add Policy · Add Policy Group · Add User · Capture Screenshot · Change Policy View Group · Click All Elements · Close · Close All · Close Window · Connect · Connect All · Delete Policy · Delete Policy Group · Delete User · Edit Mitigation Controller · Edit Policy · Get Mitigation List · Get Verbose · Left Menu · Login · Logout · Make Item Map · Mark Element · Open Ff With Profile · Reconnect · Reset Capture Counter · Select Items In Table · Select Window · Set Ajax Wait · Set Capture Counter · Set Capture Format · Set Verbose · Show Detail Mitigation · Show Policy Basic · Show Policy Detection · Show Policy Display · Show Policy Mitigation · Show Policy Mo · Show Policy Monitor · Show Policy Notify · Show Policy Nw Monitor · Show Policy Traffic · Start Mitigation · Stop Mitigation · Switch · Update Mitigation Controller Info · Verbose Capture · Wait Until Element Changes

Keywords

<table>
<tr><th>Keyword</th><th>Arguments</th><th>Documentation</th></tr>
<tr><td><b>Add Policy</b></td><td><i>**policy</i></td><td>
Adds a new Samurai policy

`policy` is a map containing the below information to create the new policy.

<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><i>test001</i></td></tr>
<tr><td>basic_alias</td><td>alias name of the policy</td><td></td><td><i>test001</i></td></tr>
<tr><td>basic_port_id</td><td>another alias</td><td></td><td></td></tr>
<tr><td>basic_facing</td><td>`customer` or `backbone`</td><td></td><td><i>customer</i></td></tr>
<tr><td>basic_intf_list</td><td>list of router and interface pair, separated by comma</td><td>yes</td><td><i>10.128.18.31:xe-0/0/0.1</i></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 (`incoming` or `outgoing`)</td><td></td><td><i>incoming</i></td></tr>
<tr><td>traffic_enabled</td><td>Enable traffic monitoring or not</td><td>yes</td><td><i>${TRUE}</i> or <i>${FALSE}</i></td></tr>
<tr><td>detection_enabled</td><td>Enable detection or not</td><td>yes</td><td><i>${TRUE}</i> or <i>${FALSE}</i></td></tr>
<tr><td>detection_direction</td><td>change detect direction fo all attack type</td><td>`incomming,<i>outgoing,</i>`both`
|| <i>both:check</i></td><td></td></tr>
<tr><td>mitigation_enabled</td><td>Enable Mitigation or not</td><td>yes</td><td><i>${TRUE}</i> or <i>${FALSE}</i></td></tr>
<tr><td>mitigation_zone_name</td><td>Name of the zone for mitigation</td><td></td><td><i>zone001</i></td></tr>
<tr><td>mitigation_zone_prefix</td><td>Prefixes that could mitigate</td><td></td><td><i>1.1.1.1/32</i></td></tr>
<tr><td>mitigation_thr_bps</td><td>Upper limit (bps)</td><td></td><td><i>800,000,000</i></td></tr>
<tr><td>mitigation_thr_pps</td><td>Upper limit (pps)</td><td></td><td><i>54,000,000</i></td></tr>
<tr><td>mitigation_auto_enabled</td><td>Enable
automitgation or not</td><td></td><td><i>${TRUE}</i> or <i>${FALSE}</i></td></tr>
<tr><td>mitigation_auto_level</td><td>Automitgation level</td><td></td><td>0:overLow 1:overMedium 2:High</td></tr>
<tr><td>mitigation_auto_time</td><td>Automitigation detect attack time (min)</td><td></td><td>default is 15</td></tr>
<tr><td>mitigation_mo_enabled</td><td>Using Arbor TMS MO or not</td><td>yes</td><td><i>${TRUE}</i> or <i>${FALSE}</i></td></tr>
<tr><td>mitigation_auto_stop_enabled</td><td>Enable</td><td></td><td><i>${TRUE}</i> or <i>${FALSE}</i></td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td></td><td>
<table>
<tr><td>mitigation_auto_stop_enabled</td><td>Enable automitgation stop or not</td><td></td><td><i>${TRUE} or ${FALSE}</i></td></tr>
<tr><td>mitigation_auto_stop_level</td><td>Automitgation level</td><td></td><td>0:overLow 2:High</td></tr>
<tr><td>mitigation_auto_stop_time</td><td>Automitigation stop detect attack time (min)</td><td></td><td>default is 15</td></tr>
<tr><td>mitigation_device_list</td><td>Devices used for TMS, separated by comma</td><td></td><td><i>ArborSP-A</i></td></tr>
<tr><td>mitigation_mo_name</td><td>MO name, separated by comma</td><td></td><td><i>OCN12(ALU)_LOOSE</i></td></tr>
<tr><td>mitigation_comm_list</td><td>commna separated peer/community list</td><td></td><td><i>1.10(180.0.1.10)/2914:666,1.11(180.0.1.11)/2914:777</i></td></tr>
<tr><td>nw_monitor_gre1</td><td>1st GRE address for NW monitor</td><td></td><td><i>210.0.1.1</i></td></tr>
<tr><td>nw_monitor_gre2</td><td>2nd GRE address for NW monitor</td><td></td><td><i>210.0.1.1</i></td></tr>
<tr><td>nw_monitor_ce1</td><td>1st CE address for NW monitor</td><td></td><td><i>210.0.1.2</i></td></tr>
<tr><td>nw_monitor_ce2</td><td>2nd CE address for NW monitor</td><td></td><td><i>210.0.1.2</i></td></tr>
<tr><td>nw_monitor_pe1</td><td>1st PE for NW monitor (list)</td><td></td><td><i>edge01hige-MX2020-15(118.23.176.244)</i></td></tr>
<tr><td>nw_monitor_pe2</td><td>2nd PE for NW monitor (list)</td><td></td><td><i>edge01hige-MX2020-15(118.23.176.244)</i></td></tr>
<tr><td>event_name</td><td>name of the message event to make</td><td></td><td><i>info1</i></td></tr>
<tr><td>event_addr</td><td>address to send the events</td><td></td><td><i>user@mail.com</i></td></tr>
<tr><td>view_group</td><td>user group that could view this policy, separated by comma</td><td>yes</td><td><i>SuperGroup,test_group_007</i></td></tr>
</table>
<p>Example:</p>
<table>
<tr><td>Samurai.Switch</td><td>samurai-1</td><td></td></tr>
<tr><td>Samurai.Add Policy</td><td>name=${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=${TRUE}</td><td></td></tr>
<tr><td>...</td><td>mitigation_zone_name=test_zone001</td><td>mitigation_zone_prefix=1.1.1.1/32</td></tr>
<tr><td>...</td><td>mitigation_device_list=ArborSP-A,ArborSP-B</td><td></td></tr>
<tr><td>...</td><td>mitigation_mo_enabled=${TRUE}</td><td></td></tr>
<tr><td>...</td><td>mitigation_mo_name=N000000012_LOOSE</td><td></td></tr>
<tr><td>...</td><td>mitigation_comm_list=1.10(180.0.1.10)/2914:666,1.11(180.0.1.11)/2914:777</td><td></td></tr>
<tr><td>...</td><td>event_name=test</td><td>event_addr=user@mail.com</td></tr>
<tr><td>...</td><td>view_group=SuperGroup</td><td></td></tr>
</table>
</td></tr>
<tr><td><b>Add Policy Group</b></td><td><i>group_name</i>, <i>policy_list=*</i>, <i>limit_bps=4000000000</i>, <i>limit_pps=2700000</i></td><td>
<p>Add a new policy group</p>
<p><code>group_name</code> is the name of the new group. <code>policy_list</code> is a comma separated of existed policy that should be bound to this policy. An asterisk for this parameter (<code>*</code>) means <i>all of the existed policy</i>. <code>limit_bps</code> and <code>limit_pps</code> are the mitigation capacity threshold of this group.</p>
</td></tr>
<tr><td><b>Add User</b></td><td><i>group</i>, <i>**user_info</i></td><td>
<p>Adds user to the current group <code>user_info</code> is a dictionary contains user information that has following keys: <code>name</code>, <code>password</code>, <code>privilege</code> and <code>policy</code></p>
<p><code>privilege</code> is existed privilege that has been created (e.g: <i>system_admin</i>.</p>
<p><code>policy</code> could be <code>*</code> for all current policies or a list of policy names that are binded to this user.</p>
<p><code>group</code> is the user group. <code>Dot(.)</code> means current group</p>
<p>Examples:</p>
<table>
<tr><td>Samurai.Add User</td><td>OCNDDoS</td><td>name=user000</td><td>password=Test12345678</td></tr>
<tr><td>...</td><td>privilege=system_admin</td><td>policy=*</td><td></td></tr>
<tr><td>Samurai.Add User</td><td>OCNDDoS</td><td>username=user001</td><td>password=Test12345678</td></tr>
<tr><td>...</td><td>privilege=system_admin</td><td>policy=OCN11,OCN12</td><td></td></tr>
</table>
</td></tr>
<tr><td><b>Capture Screenshot</b></td><td><i>filename=None</i>, <i>extra=</i></td><td>
<p>Captures the current screen to file</p>
<p>Using the internal counter for filename if <code>filename</code> 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.</p>
<p>An extra information will be add to the filename if <code>extra</code> is defined</p>
<p>Examples:</p>
<table>
<tr><td>Samurai.Capture Screenshot</td><td></td><td># samurai_0000000001.png</td></tr>
<tr><td>Samurai.Capture Screenshot</td><td>extra=_list</td><td># samurai_0000000002_<i>list</i>.png</td></tr>
<tr><td>Arbor.Capture Screenshot</td><td></td><td># arbor_0000000001.png</td></tr>
<tr><td>Arbor.Capture Screenshot</td><td>extra=_xxx</td><td># arbor_0000000001_xxx.png</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td></td><td><table><tr><td>Arbor.Capture Screenshot</td><td>extra=_xxx</td><td># arbor_0000000001_xxx.png</td></tr><tr><td>Samurai.Capture Screenshot</td><td>filename=1111.png</td><td># 1111.png</td></tr></table></td></tr>
<tr><td><b>Change Policy View Group</b></td><td><i>name</i>, <i>*group_name</i></td><td>Changes the groups that could see this policy
`name` is the policy name. `group_name` is a list of policies
Example:
<table><tr><td>Samurai.Change Policy View Group</td><td>super_admin</td><td>test_group001</td></tr></table></td></tr>
<tr><td><b>Click All Elements</b></td><td><i>xpath</i></td><td>Click all element in current page defined by `xpath`
Returns the number of elements that have been clicked</td></tr>
<tr><td><b>Close</b></td><td></td><td>Close the web application</td></tr>
<tr><td><b>Close All</b></td><td></td><td>Closes all current opened applications</td></tr>
<tr><td><b>Close Window</b></td><td></td><td>Closes the current window</td></tr>
<tr><td><b>Connect</b></td><td><i>app</i>, <i>name</i></td><td>Opens a web browser and connects to application and assigns a `name`.
- <i>app</i> is the name of the application (e.ge. samurai-1)
- <i>name</i> is the name of the browser
If not defined in `local.yaml` those following key will have defaut values:
<table><tr><td>browser</td><td>firefox</td><td>optional</td></tr><tr><td>login_url</td><td>/</td><td>optiona</td></tr><tr><td>proxy:</td><td></td><td>optional</td></tr><tr><td>http: 10.128.8.210:8080</td><td>optional</td><td></td></tr><tr><td>ssl: 10.128.8.210:8080</td><td>optional</td><td></td></tr><tr><td>socks: 10.128.8.210:8080</td><td>optional</td><td></td></tr></table></td></tr>
<tr><td><b>Connect All</b></td><td></td><td>Connects to all applications defined in `local.yaml`
The name of the connection will be the same of the <i>webapp</i> name</td></tr>
<tr><td><b>Delete Policy</b></td><td><i>*policy_names</i></td><td>Deletes poilcies by their names
Returned the number of deleted users
<b>Notes:</b> If the policy does not exists, the system will not report any error.
Examples:
<table><tr><td>Samurai.Delete Policy</td><td>test001</td><td>test002</td></tr></table></td></tr>
<tr><td><b>Delete Policy Group</b></td><td><i>*group_list</i></td><td>Deletes policy groups
See <i>Select Items In Table</i> for more detail about how to choose <i>group_list</i>
Returns the number of deleted policy groups Example:
<table><tr><td>Samurai.Delete Policy Group</td><td>test_group001</td><td>test_group002</td></tr></table></td></tr>
<tr><td><b>Delete User</b></td><td><i>group</i>, <i>*user_list</i></td><td>Deletes user from the user group
`group` is the user group. And `.` means current group Returns the number of deleted users
Examples:
<table><tr><td>Samurai.Delete User</td><td>SuperGroup</td><td>user001</td><td>user002</td></tr><tr><td>Samurai.Delete User</td><td>.</td><td>user002</td><td></td></tr></table></td></tr>
<tr><td><b>Edit Mitigation Controller</b></td><td><i>controller</i>, <i>**config</i></td><td>Change the setting of the mitigation control
- `control`: name of the mitigation controller
- `config`: configuration need to be changed. Currently only `tms_group` is configurable with the following format: `groupname1:action1,groupname2:action2`. `groupname` is currently set TMS group name and action could be <i>click</i>,<i>check</i> or <i>uncheck</i>.
Example:
<table><tr><td>Samurai.Edit Mitigation Controller</td><td>controller=vSP-A</td><td>tms_group=Logical0_SOCN_IPv4:uncheck</td></tr></table></td></tr>
<tr><td><b>Edit Policy</b></td><td><i>**policy</i></td><td>Edits a Samurai policy
`policy` contains information about the policy. See <i>Add Policy</i> for more details about `policy` format</td></tr>
<tr><td><b>Get Mitigation List</b></td><td><i>status=実行中</i></td><td>Gets current mitigation list
Return current active mitgation name, ID and the number of them
Example:
<table><tr><td>${MITI} ${IDS} ${NUM}=</td><td>Samurai.Get Mitigation List</td></tr></table></td></tr>
<tr><td><b>Get Verbose</b></td><td></td><td>Get current verbose mode</td></tr>
<tr><td><b>Left Menu</b></td><td><i>menu</i>, <i>locator=None</i>, <i>ignore_first_element=True</i></td><td>Chooses the left panel menu by its displayed name
When `locater` is not null, the keyword will return a list of text attribute of all elements specified by the `locator`. `locator` could be a xpath or a predefined string.
`locator` predefined strings are: `MITIGATE_REALTIME`, `MITIGATE_LIST`, `DETECT_LIST`
For example, a xpath `//div[@id='infoareain2']/*//td[1]/a` means the list of <i>link</i> of all elements in a 1st column of a table insides a `div` with id `infoareain2`.
Examples:
<table><tr><td>Samurai.Left Menu</td><td>Traffic</td><td></td><td></td></tr><tr><td>Samurai.Left Menu</td><td>Detection</td><td></td><td></td></tr><tr><td>Samurai.Left Menu</td><td>ポリシー管理</td><td></td><td></td></tr><tr><td>@{LIST}=</td><td>Samurai.Left Menu</td><td>Active Mitigation</td><td>//div[@id='infoareain2']/*//td[1]/a</td></tr></table></td></tr>
<tr><td><b>Login</b></td><td></td><td>Logs-in into the application</td></tr>
</table>

		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* The map is a dictionary from *item* to the *WebElement* Items name found by xpath are used as keys
Mark Element	*xpath*	Marking an element to check its stattus later
Open Ff With Profile	*app*, *name*	
Reconnect		Reconnects to the server
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 xpath points to the column that used as key and xpath2 is the relative xpath contains the target column. item_list is a list of item and its action that need to check. Item in the list could be a regular expresion with the format re:<regular expression>|action. The default action for the item could be click`*(default),*`check or uncheck The keyword is called with assuming that the table is already visible. Returns the tupple of all items and selected items **Note:** Non-width-space (\u200b) will be take care by the keyword. **Note:** if the first item_list is * then the keywork will try to click a link named *すべてを選択*.
Select Window	*title*	Selects a window by its title
Set Ajax Wait	*wait_time=2s*	Set the ajax wait time
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 The format does not include the default prefix .png The default format is <mod>_%010d. mod could be samurai or arbor See https://docs.python.org/2/library/string.html#format-specification-mini-language for more details about the format string. Examples: Samurai.*Set Capture Format* ${case}_%010d # ${case} is a predefined variable
Set Verbose	*verbose=False*	Set current verbose mode to verbose
Show Detail Mitigation	*id*	Shows detail information of a mitigation
Show Policy Basic	*policy_name*	Makes the virtual browser show basic setting of the policy *name*. A following Samurai.*Capture Screenshot* is necessary to capture the result.
Show Policy Detection	*policy_name*	Shows the detction pannel of *policy_name* policy
Show Policy Display	*policy_name*	Make a virtual browser show the display setting of a policy **Notes**: Depending on the setting of the policy, MO panel may not be existed. In this case, if *strict* is True, then the keyword will fail.
Show Policy Mitigation	*policy_name*	Make the virtual browser show the mitigation setting of a policy
Show Policy Mo	*policy_name*, *strict=False*	Make the virtual browser show the MO setting of a policy Automatically expand the MO section of other devices. **Notes**: Depending on the setting of the policy, MO panel may not be existed. In this case, if *strict* is True, then the keyword will fail.
Show Policy Monitor	*policy_name*	
Show Policy Notify	*policy_name*	Make a virtual browser show the mitigation setting of a policy
Show Policy Nw Monitor	*policy_name*, *strict=False*	Make a virtual browser show the NW monitor setting of a policy **Notes**: Depending on the setting of the policy, MO panel may not be existed. In this case, if *strict* is True, then the keyword will fail.
Show Policy Traffic	*policy_name*	Makes the virtual browser show the traffic setting of the policy *name*. 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 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 Returns mitigation id and selected arbor device Example: ${id} ${device}= Samurai.*Start Mitigation* 211.1.12.1/32 mitigation by RENAT SP-A ${TRUE}
Stop Mitigation	*id*, *raise_error=True*	Stops a mitigation by its ID The keyword will raise an error if *raise_error* is True. Otherwise it will ignore any errors. Example: Samurai.*Stop Mitigation* 700
Switch	*name*	Switches the current browser to name
Update	*controller*, *wait=10s*	Updates information of *controller*

Update Mitigation Controller Info	*controller*, *wait=10s*	Updates information of *controller*
Verbose Capture	**args*, ***kwargs*	Capture screenshot if verbose mode is `True` otherwise do nothing
Wait Until Element Changes	*interval=5s*, *timeout=180s*, *error_on_timeout=False*	Wait until the marked element has been changed

Altogether 48 keywords.
Generated by Libdoc on 2018-12-06 13:39:49.

Arbor

Library version:	RENAT 0.1.11
Library scope:	test suite
Named arguments:	supported

Introduction

A library provides functions to control Arbor application

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

See WebApp for common keywords of web applications.

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

Notes: From 0.1.10, move from *Selenium2Library* to *SeleniumLibrary*

Shortcuts

Capture Screenshot · Clean Status Msg · Close · Close All · Commit · Connect · Connect All · Detail First Mitigation · Get Status Msg · Get Verbose · Login · Logout · Mark Element · Menu · Open Ff With Profile · Reconnect · Reset Capture Counter · Set Ajax Wait · Set Capture Counter · Set Capture Format · Set Verbose · Show All Mitigations · Show Detail Countermeasure · Show Detail First Mitigation · Show Detail Mitigation · Show Detail Mitigation With Order · Switch · Verbose Capture · Wait Until Element Changes

Keywords

<table>
<tr><th>Keyword</th><th>Arguments</th><th>Documentation</th></tr>
<tr><td><b>Capture Screenshot</b></td><td><i>filename=None</i>, <i>extra=</i></td><td>Captures the current screen to file

Using the internal counter for filename if <code>filename</code> is not specified. In this case, the filename is defined by a pre-set format. <i>Set Capture Format</i> could be used to change the current format.

An extra information will be add to the filename if <code>extra</code> is defined

Examples:
<table>
<tr><td>Samurai.<i>Capture Screenshot</i></td><td></td><td># samurai_0000000001.png</td></tr>
<tr><td>Samurai.<i>Capture Screenshot</i></td><td>extra=_list</td><td># samurai_0000000002_<i>list</i>.png</td></tr>
<tr><td>Arbor.<i>Capture Screenshot</i></td><td></td><td># arbor_0000000001.png</td></tr>
<tr><td>Arbor.<i>Capture Screenshot</i></td><td>extra=_xxx</td><td># arbor_0000000001_<i>xxx</i>.png</td></tr>
<tr><td>Samurai.<i>Capture Screenshot</i></td><td>filename=1111.png</td><td># 1111.png</td></tr>
</table></td></tr>
<tr><td><b>Clean Status Msg</b></td><td></td><td>Disposes any alert or status messgae at the top of the current page</td></tr>
<tr><td><b>Close</b></td><td></td><td>Close the web application</td></tr>
<tr><td><b>Close All</b></td><td></td><td>Closes all current opened applications</td></tr>
<tr><td><b>Commit</b></td><td><i>strict=True</i></td><td>Commit the current changes

If <i>strict</i> is <code>${TRUE}</code> then the keyword fails if there is not change to commit. Otherwise, it does nothing</td></tr>
<tr><td><b>Connect</b></td><td><i>app</i>, <i>name</i></td><td>Opens a web browser and connects to application and assigns a <code>name</code>.

Extra information could be added to the <code>webapp</code> sections likes <code>login_url</code>, <code>browser</code> or <code>profile_dir</code>. Default values are:
<table>
<tr><td>browser</td><td>firefox</td></tr>
<tr><td>login_url</td><td>/</td></tr>
<tr><td>profile_dir</td><td>./config/samurai.profile</td></tr>
</table></td></tr>
<tr><td><b>Connect All</b></td><td></td><td>Connects to all applications defined in <code>local.yaml</code>

The name of the connection will be the same of the <i>webapp</i> name</td></tr>
<tr><td><b>Detail First Mitigation</b></td><td></td><td></td></tr>
<tr><td><b>Get Status Msg</b></td><td></td><td>Get current status message

Return null string if no message exists</td></tr>
<tr><td><b>Get Verbose</b></td><td></td><td>Get current verbose mode</td></tr>
<tr><td><b>Login</b></td><td></td><td>Logs into the Arbor application</td></tr>
<tr><td><b>Logout</b></td><td></td><td>Logs-out the current application, the browser remains</td></tr>
<tr><td><b>Mark Element</b></td><td><i>xpath</i></td><td>Marking an element to check its stattus later</td></tr>
<tr><td><b>Menu</b></td><td><i>order</i>, <i>wait=2s</i>, <i>capture_all=False</i>, <i>prefix=menu_</i>, <i>suffix=.png</i>,</td><td>Access to Arbor menu</td></tr>
</table>

<table>
<tr><td></td><td><i>partial_match=False</i></td><td>Parameters
<ul>
<li>order is the list of top menu items separated by '/'</li>
<li>wait is the wait time after the last item is clicked</li>
<li>if capture_all is True then a screenshot is captured for each menu item automtically. In this case, the image file is appended by</li>
</ul>
prefix and suffix.
<ul>
<li>by default, the system try to match the menu item in full, when partial_match is True, partial match is applied.</li>
</ul>
Samples:
<table>
<tr><td>Arbor.Menu</td><td>order=Alerts/Ongoing</td><td></td></tr>
<tr><td>Arbor.Capture Screenshot</td><td></td><td></td></tr>
<tr><td>Arbor.Menu</td><td>order=Alerts/All Alerts</td><td></td></tr>
<tr><td>Arbor.Capture Screenshot</td><td></td><td></td></tr>
<tr><td>Arbor.Menu</td><td>order=System/Status/Deployment Status</td><td></td></tr>
<tr><td>Arbor.Capture Screenshot</td><td></td><td></td></tr>
<tr><td>Arbor.Menu</td><td>order=System/Status/Signaling Status/Appliance Status</td><td>partial_match=${TRUE}</td></tr>
<tr><td>Arbor.Capture Screenshot</td><td></td><td></td></tr>
</table>
</td></tr>
<tr><td><b>Open Ff With Profile</b></td><td><i>app, name</i></td><td></td></tr>
<tr><td><b>Reconnect</b></td><td></td><td>Reconnect to server if necessary</td></tr>
<tr><td><b>Reset Capture Counter</b></td><td></td><td>Resets the counter of the screen capture</td></tr>
<tr><td><b>Set Ajax Wait</b></td><td><i>wait_time=2s</i></td><td>Set the ajax wait time</td></tr>
<tr><td><b>Set Capture Counter</b></td><td><i>value=0</i></td><td>Sets the counter of the screen capture to value</td></tr>
<tr><td><b>Set Capture Format</b></td><td><i>format</i></td><td>Sets the format for the screen capture file
The format does not include the default prefix .png The default format is <mod>_%010d. mod could be samurai or arbor
See https://docs.python.org/2/library/string.html#format-specification-mini-language for more details about the format string.
Examples:
<table>
<tr><td>Samurai.Set Capture Format</td><td>${case}_%010d</td><td># ${case} is a predefined variable</td></tr>
</table>
</td></tr>
<tr><td><b>Set Verbose</b></td><td><i>verbose=False</i></td><td>Set current verbose mode to verbose</td></tr>
<tr><td><b>Show All Mitigations</b></td><td></td><td>Shows all mitigations</td></tr>
<tr><td><b>Show Detail Countermeasure</b></td><td><i>name, *method_list</i></td><td>Shows detail informatin about a countermeasure
<i>name</i> is used to search the the mitigation and <i>method_list</i> is a list of countermeasures that are listed in Arbor Countermeasures panel
Example:
<table>
<tr><td>${NAME}</td><td>${ID}=</td><td>Show Detail First Mitigation</td><td></td></tr>
<tr><td>Arbor.Show Detail Countermeasure</td><td>${NAME}</td><td>DNS Malformed</td><td></td></tr>
<tr><td>Arbor.Capture Screenshot</td><td></td><td></td><td></td></tr>
<tr><td>Sleep</td><td>10s</td><td></td><td></td></tr>
<tr><td>Arbor.Show Detail Countermeasure</td><td>${NAME}</td><td>Zombie Detection</td><td>HTTP Malformed</td></tr>
<tr><td>Arbor.Capture Screenshot</td><td></td><td></td><td></td></tr>
</table>
</td></tr>
<tr><td><b>Show Detail First Mitigation</b></td><td></td><td>Shows details about the 1st mitigation on the list
The keyword returns the <i>mitigation ID</i> and its name</td></tr>
<tr><td><b>Show Detail Mitigation</b></td><td><i>search_str</i></td><td>Shows detail information of a mitigation by its <i>search_str</i>
<b>Note</b>: the result could include multi mitigations</td></tr>
<tr><td><b>Show Detail Mitigation With Order</b></td><td><i>order</i></td><td>Shows details about the <i>order</i>(th) mitigation in the current list
<i>order</i> is counted from 1. The keyword returns the mitigation_id and its name
Example:
<table>
<tr><td>${NAME}</td><td>${ID}=</td><td>Arbor.Show Detail Mitigation With Order</td><td>3</td></tr>
<tr><td>Log To Console</td><td>${NAME}:${ID}</td><td></td><td></td></tr>
<tr><td>Arbor.Capture Screenshot</td><td></td><td></td><td></td></tr>
</table>
</td></tr>
</table>

Switch	*name*	Switches the current browser to `name`
Verbose Capture	**args*, ***kwargs*	Capture screenshot if verbose mode is `True` otherwise do nothing
Wait Until Element Changes	*interval=5s*, *timeout=180s*, *error_on_timeout=False*	Wait until the marked element has been changed

Altogether 29 keywords.
Generated by Libdoc on 2018-12-06 13:39:50.

Tester

Library version:	RENAT 0.1.11
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 items, start and stop traffic flows. It also could generate traffic reports and support QuickTest for IxNetwork.

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

```
tester:
  tester01:
    device: ixnet03_8009
    config: vmx_20161129.ixncfg
    real_port:
      -  description: to egde router
```

chassis: 10.128.32.71

```
           card: 6
           port: 11
      -  description: to backbone router
```

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

Close All · Connect · Connect All · Switch

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 2018-12-06 13:39:40.

ixload

Library scope:	global
Named arguments:	supported

Introduction

provides functions for IxLoad

To use IxLoad module, a IxLoad TCL server should be started properly.

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

Close · **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 A `prefix` will be automatically added to the file names. Currently the follow data will be downloaded to the local machine ▪ HTTP_Server.csv ▪ HTTP Client.csv ▪ HTTP Client - Per URL.csv ▪ HTTP Server - Per URL.csv ▪ L2-3 Stats for Client Ports.csv ▪ L2-3 Stats for Server Ports.csv ▪ L2-3 Throughput Stats.csv ▪ Port CPU Statistics.csv Extra files could be add by `more_file` which is a comma separated filename string 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` `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 Returns the elapsed time in seconds

Altogether 7 keywords.
Generated by Libdoc on 2018-12-06 13:39:45.

ixnet

Library scope:	global
Named arguments:	supported

Introduction

provides functions for IxNetwork

To use IxNetwork module, a IxNetwork TCL server should be started properly.

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

Add 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 · **C**sv Snapshot · **G**et All Test Result · **G**et Quicktest List · **G**et Quicktest Result · **G**et Quicktest Result Path · **G**et Test Composer Result · **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**egenerate · **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 Test Composer · **S**tart Traffic · **S**top All Protocols · **S**top And Save Capture · **S**top Quicktest · **S**top Test Composer · **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 real-port information from active local config - time_out is the wait time until port is connected (default is 2m) - learn_time is the time waiting for arp to be learned (default is 2m) Sample of local config |tester: tester: device: ixnet03_8009 config: quicktest.ixncfg real-port: - chassis: 10.128.4.41 card: 4 port: 3 ip: 10.100.11.2 mask: 24 gw: 10.100.11.1 - chassis: 10.128.4.41 card: 4 port: 4 ip: 10.100.14.2 mask: 24 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 Type could be one of following: rfc2544throughput, rfc2544frameLoss, rfc2544back2back. Use Tester.*Load Config* to load a customized quicktest When clear_all is True, any existed quicktests will be cleared. Transmit mode tx_mode takes following values: interleaved (default) or sequential. The mode should be identical with the transmit mod of the ports. **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*, *refresh=True*	Applies the current traffic configuration refresh: Refreshed the learned information before apply the traffic or not **Note:** This is a blocking command
Change Frame Rate	*self*, *value*, *pattern=.**, *flow_pattern=.**	Changes the frame rate Parameter: - value: value to set. Depends on the current configuration, this could be percent line rate or bit per second etc. - pattern*: a regular expression to identify* `traffic item name, default is everything .* - flow_pattern: a regular expression to identify flow group inside the item
Change Frame Rate Dynamic	*self*, *value*, *pattern=.**	Changes the traffic flow rate on-fly No need to stop the running traffic to change the rate Parameter: - value: value to set. Depend on the current configuration, this could be percent line rate or bit per second etc. - pattern: a regular expression to identify traffic item name, default is everything .*

Change Frame Size	*self*, *type*, *value*, *pattern=.**, *flow_pattern=.**	Changes the frame size Parameter: ▪ `type`: could be `fixed size`, `increment_from`,`increment_step` or `increment_to` ▪ `value`: value to set ▪ `pattern`: a regular expression to identify traffic item name, default is everything `.*` ▪ `flow_pattern`: a regular expression to identify `flow group` inside the item
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*
Csv Snapshot	*self*, *prefix=snapshot_*, **views*	Get current CSV snapshot Parameters: ▪ *prefix*: prefix that be added to the filename. Default is `snapshot_` ▪ *views*: list of target views (eg: `Port Statistics`, `Flow Statistics` ...). If *view* is `None`, all current available views will be target
Get All Test Result	*self*, *prefix=stat_*	Collects all Ixia traffic data after traffic is stopped. 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 `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 `test_index` is a index of the current Quicktest. `-1` means that last one.
Get Test Composer Result	*self*, *result_file=composer.log*	Get the result of test composer script
Get Test Report	*self*, *local_name=ixnet_report.pdf*, *enable_all=True*	Generates and get report of the current active test in PDF format `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 Currently, supported views are: `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` Result were store as CSV files in `result` folder. If there is no valid data, view will be silently ignored 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*, *wait_time3=30s*	loads traffic configuration, applies and start protocol if necessary. 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. 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. Parameters: ▪ `apply`: applies traffic when `True` otherwise ▪ `protocol`: starts all protocols when `True` otherwise ▪ `force`: force to reclaim the ports when `True` otherwise ▪ `wait_time`: wait time after applying protocols ▪ `wait_time2`: maximum wait time befor all ports become available. In common case, this is calculated automatically so user does not need to change this value. ▪ `wait_time3`: default waiting time after config file is loaded (30s) More information about ports could be define in `real_port` section like this: # tester information tester:    tester:     device: ixnet03_8009     config: bgp.ixncfg     real-port:       - chassis: 10.128.4.41         card: 4         port: 7         media: fiber         tx_mode: interleaved Configurable port parameters ares: ▪ `tx_mode`: *sequential* or *interleaved*(default)

<table>
<tr><td></td><td></td><td>- media : <i>copper</i> or <i>fiber</i> (<b>Note</b>: no default value)

See Common for more details about the yaml configuration files.</td></tr>
<tr><td><b>Load Traffic</b></td><td><i>self</i>, <i>wait_time=2m</i>, <i>wait_time2=2m</i>, <i>apply=True</i>, <i>protocol=True</i>, <i>force=True</i>, <i>tx_mode=interleaved</i></td><td></td></tr>
<tr><td><b>Loss From File</b></td><td><i>self</i>, <i>file_name=Flow_Statistics.csv</i>, <i>index=0</i></td><td>Returns packet loss by miliseconds and delta frame.

Parameters:

- <i>file_name</i>: flow information (csv format). Default is Flow_Statistics.csv
- <i>index</i>: row index of the result(counted from zero)

The calculation should be performed when traffic is stopped. The calculation supposed traffic is configured by frame per second</td></tr>
<tr><td><b>Ping</b></td><td><i>self</i>, <i>dst_ip</i>, <i>src_port_index=0</i>, <i>src_intf_index=0</i></td><td>Ping from Ixia to dst_ip

The keyword return the output string as it is. The return could be

<pre>
- Port <portName>: ping failed: port not assigned
- Response received from <sourceIp>/unknown . Sequence Number <sequenceNumber>
- Ping request to <destinationIp>/unknown ip failed: <GenericPingError>/<error>: <genericError>unknown reason
- Error: Couldn't find any source interface for Send Ping to <destinationIp> on <portName> Id <id>
- Error: Couldn't find any source IP for Send Ping to <destinationIp> on <portName> Id <id>
</pre>
Parameters:

- src_port_index: index of Ixia port (starts from 0)
- src_intf_index: index of interface insides the port (starts from 0)

Examples:
<table>
<tr><td>Tester.<i>Ping</i></td><td>1.1.1.1</td><td>0</td><td>0</td></tr>
<tr><td>Tester.<i>Ping</i></td><td>1.1.1.1</td><td></td><td></td></tr>
</table></td></tr>
<tr><td><b>Regenerate</b></td><td><i>self</i></td><td>Regenerates <b>all</b> flow of current traffic items</td></tr>
<tr><td><b>Reset Config</b></td><td><i>self</i></td><td>Clears current config and creates new blank config</td></tr>
<tr><td><b>Run Quicktest</b></td><td><i>self</i>, <i>test_index=0</i>, <i>wait_until_finish=True</i></td><td>Runs the Quicktest and wait until it finishes

<b>Warning</b>: it could take a long time to finish a quicktest</td></tr>
<tr><td><b>Set All Traffic Item</b></td><td><i>self</i>, <i>enabled=True</i></td><td>Enables/Disables <b>all</b> traffic items at once</td></tr>
<tr><td><b>Set Bgp Items</b></td><td><i>self</i>, <i>port_index</i>, <i>neighbor_index</i>, <i>route_range_index</i>, <i>is_enable</i></td><td>Enables/Disables BGP entry by a set of port,neighbor,route_range index

Parameters:

- port_index: index of the port
- neighbor_index: index of the neighbor or *
- route_range_index: index of the route range or *
- is_enable: ${TRUE} or ${FALSE}

Note

Examples:
<table>
<tr><td>Tester.<i>Set BGP Items</i></td><td>0</td><td>*</td><td>*</td><td>${FALSE}</td></tr>
<tr><td>Tester.<i>Set BGP Items</i></td><td>0</td><td>*</td><td>*</td><td>${TRUE}</td></tr>
</table></td></tr>
<tr><td><b>Set Bgp Neighbor</b></td><td><i>self</i>, <i>*indexes</i>, <i>**kwargs</i></td><td>Enables/Disables BGP entry by neighbor index

kwargs contains following parameters:

- indexes: is a list of index of BGP neighbor (index is started from zero)
- vport_index: is the target vport index
- enabled: TRUE or FALSE

Examples:
<table>
<tr><td>Tester.<i>Set BGP Item</i></td><td>0</td><td>1</td><td>vport_index=0</td><td>enabled=${FALSE}</td></tr>
<tr><td>Tester.<i>Set BGP Item</i></td><td>0</td><td>1</td><td>vport_index=1</td><td>enabled=${TRUE}</td></tr>
</table></td></tr>
<tr><td><b>Set Capture Port</b></td><td><i>self</i>, <i>data_mode=True</i>, <i>control_mode=True</i>, <i>port_index=0</i></td><td>Capture packets for follow port

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

<b>Note</b>: control_mode saves all control packets and data_mode only saves data packets.

<b>Note</b>: control_mode saves all control packets and data_mode only saves data packet

Examples:
<table>
<tr><td>Tester.<i>Set Capture Port</i></td><td>0</td><td></td><td></td></tr>
<tr><td>Tester.<i>Set Capture Port</i></td><td>control_mode=${TRUE}</td><td>0</td><td>1</td></tr>
</table></td></tr>
<tr><td><b>Set Traffic Item</b></td><td><i>self</i>, <i>*items</i>, <i>**kwargs</i></td><td>Enables/Disables some traffic items items

Parameters:

- items: a list of Ixia traffic item name
- enabled: False or True ,the mode to set traffic item to, default is True (enabled)

<b>Note</b>: traffic item could be specified by ::<num> format. In this case the num is the order of traffic item</td></tr>
</table>

<table>
<tr><td></td><td></td><td>Note: traffic item could be specified by #<num> format. In this case the num is the order of traffic item count from zero.
Returns True if all items are set coordinately or otherwise
Examples:
<table><tr><td>Set Traffic Item</td><td>Traffic Item 1</td><td>Traffic Item 2</td></tr><tr><td>Set Traffic Item</td><td>@{item_list}</td><td></td></tr><tr><td>Set Traffic Item</td><td>Traffic Item 1</td><td>enabled = ${FALSE}</td></tr></table></td></tr>
<tr><td>**Should Be Pingable**</td><td>*self*, *dst_ip*, *src_port_index=0*, *src_intf_index=0*</td><td>Ping from Ixia and raise an error if ping fails
The keyword return *True* if succeeds</td></tr>
<tr><td>**Start Capture**</td><td>*self*, *wait_time=30s*</td><td>Start packet capture
Target ports are set by the configuration file or by [Set Capture] keyword</td></tr>
<tr><td>**Start Protocol**</td><td>*self*, *wait_time=1m*</td><td>Starts all protocols and wait for wait_time
Default wait_time is 1 minute. Make sure wait_time is big engouh to start all protocols.</td></tr>
<tr><td>**Start Test Composer**</td><td>*self*, *script_name=Main_Procedure*, *run_num=1*, *wait_for_test=True*, *parameter=*, *wait=10s*</td><td>Run a test composer script.
The test composer script should be included in an Ixia Network configuration file and loaded properly with *Load Config*
Parameters:
- script_name is the name of the script to run. Default value is Main_Procedure.
- wait_for_test: if ${TRUE} then wait until the script finishes.
- parameter: parameter that is passed to the script. Parameter could be in 2 formats: {{VAR1 VALUE1} {VAR2 VALUE2}} or simply as VALUE1 VALUE2.
The script must prepare *VAR1* and *VAR2* properly by *Test parameter*. See Ixia Network anout composer script for more details.
- wait: wait time before go to next keyword
Examples:
<table><tr><td>Tester.*Start Test Composer*</td><td>parameter=XXX YYY</td></tr><tr><td>Tester.*Get Test Composer Result*</td><td>result_file=script1.log</td></tr><tr><td>Tester.*Start Test Composer*</td><td>parameter={{VAR1 AAA} {VAR2 BBB}}</td></tr><tr><td>Tester.*Get Test Composer Result*</td><td>result_file=script1.log</td></tr></table></td></tr>
<tr><td>**Start Traffic**</td><td>*self*, *wait_time=30s*</td><td>Starts the current traffic settiing and wait for wait_time.
Note: This is a asynchronus action. After called, the keyword finishes immediatly but it will take a while before traffic starts
By default the keyword will wait for 30 seconds.</td></tr>
<tr><td>**Stop All Protocols**</td><td>*self*, *wait_time=30s*</td><td>Stop all running protocols</td></tr>
<tr><td>**Stop And Save Capture**</td><td>*self*, *prefix=*, *wait_until_finish=True*, *monitor_interval=5s*</td><td>Stop current capture and save the resuls to folder specified by path
Captured files will be saved in current result folder with prefix appended in their names.
Examples:
<table><tr><td>Tester.*Start Capture*</td><td></td></tr><tr><td>Sleep</td><td>10s</td></tr><tr><td>Tester.*Stop And Save Capture*</td><td>${RESULT_FOLDER}/capture.zip</td></tr></table></td></tr>
<tr><td>**Stop Quicktest**</td><td>*self*, *test_index=0*</td><td>Stops a running test</td></tr>
<tr><td>**Stop Test Composer**</td><td>*self*, *wait=10s*</td><td>Stop a running composer
Do nothing when a test composer has already stopped or no composer has been prepared.</td></tr>
<tr><td>**Stop Traffic**</td><td>*self*, *stop_protocol=False*, *wait_time=10s*</td><td>Stops the current traffic and wait for wait_time
Parameters:
- stop_protocol: if True also stops all running protocols
- wait_time: time to wait after apply the command</td></tr>
<tr><td>**Wait Until Connected**</td><td>*self*, *timeout_str=5m*</td><td>Waits until ports become enabled and connected</td></tr>
</table>

Altogether 41 keywords.
Generated by Libdoc on 2018-12-06 13:39:44.

ixbps

Library scope:	global
Named arguments:	supported

Introduction

provides functions for Ixia Breaking Point

Breaking Point testers setting is set by `tester` section in `local.yaml`. Users need to specify the physical ports used by the test by its card and port number.

Setting example:

```
ixbps01:
   device: ixbps01
   config: test.bpt
   real-port:
      - card:  1
        port:  0
      - card:  1
        port:  1
```

Notes: Ignore the *self* parameters when using those keywords. The module requires Breaking Point Python library installedd properly to work.

Shortcuts

Cleanup Tests · Close · Get Card Config · Get Card Mode · Get Test Report · Load Config · Set Card Config · Start Test · Stop Test · Wait Until Finish

Keywords

Keyword	Arguments	Documentation
Cleanup Tests	*self*	Cleans up running test and release their ports
Close	*self*	Closes the connection to the BP box
Get Card Config	*self*, *slot_num*	Get card configuration for *slot_num* Parameter: - *slot_num* is *all* or an integer start from 1 Result is a json formatted string contains the informaition for specific slot or *all* slots.
Get Card Mode	*self*, *slot_num*	Gets the *mode* of a specific slot
Get Test Report	*self*, *report_name=result*, *format=csv*	Gets and saves the test report to local disk The report will be in PDF format. If `export_csv` is `True` then test results are also exported by CSV format.
Load Config	*self*, *config_name=*, *force=True*	Loads test configuration config_name is defined in `local.yaml` or specific by user in the main scenario.
Set Card Config	*self*, *slot*, *action=mode*, *param=ixload*	Changes the configuration of BPS card Parameters: - slot: slot number - action: `mode` or `perfacc` - param: depending on *action* Values of *param*: - if *action* is `mode` then *param* should be `ixload`, `bp` or `bpl23` (BreakingPoint L2/3) - if *action* is `perfacc` then *param* should be `${TRUE}` or `${FALSE}`
Start Test	*self*, *test_name=None*, *force=True*	Starts a test by its name The system automatically reserve the ports defined in `local.yaml`. The reserved ports are released when the test is stopped `test_name` is the name of the testmodel saved in the configuration. If `test_name` is *None*, the 1st testmodel will be used. If `force` is *True* then all running tests and their reserved ports will be released.
Stop Test	*self*, *wait=5s*	Stops a running test
Wait Until Finish	*self*, *interval=30s*, *timeout=30m*, *verbose=False*	Waits until the test finished or timeout **Notes**: This is a blocking keyword

Altogether 10 keywords.
Generated by Libdoc on 2018-12-06 13:39:46.

Hypervisor

Library version:	RENAT 0.1.11
Library scope:	test suite
Named arguments:	supported

Introduction

A module controls Hypervisors

A hypervisor is declared in `local.yaml` like this:

```
# esxi information
hypervisor:
  esxi-server:
     device: esxi-3-15
```

Notes: Currently support VMWare(Esxi) only

Shortcuts

Capture Mks Screenshot · **C**lick Mks · **C**lose · **C**lose All · **C**onnect · **C**onnect All · **G**et Mks Ticket · **G**et Vm Id · **G**et Vm List · **G**et Vm Power State · **O**pen Console · **P**ower Off · **P**ower On · **R**eset Capture Counter · **S**end Mks Cmd · **S**end Mks Key · **S**et Capture Format · **S**witch · **X**run

Keywords

Keyword	Arguments	Documentation
Capture Mks Screenshot	**args*, ***kwargs*	
Click Mks	**args*, ***kwargs*	
Close		Closes and disconnects from a hypervisor
Close All		Closes all current opend hypervisor connection
Connect	*hyper*, *name*	Connects to a Hypervisor
Connect All	*prefix=*	Connect to **all** hypervisor listed in local config yaml
Get Mks Ticket	**args*, ***kwargs*	
Get Vm Id	**args*, ***kwargs*	
Get Vm List	**args*, ***kwargs*	
Get Vm Power State	**args*, ***kwargs*	
Open Console	**args*, ***kwargs*	
Power Off	**args*, ***kwargs*	
Power On	**args*, ***kwargs*	
Reset Capture Counter	**args*, ***kwargs*	
Send Mks Cmd	**args*, ***kwargs*	
Send Mks Key	**args*, ***kwargs*	
Set Capture Format	**args*, ***kwargs*	
Switch	*name*	Switch the current hypervisor to a new one
Xrun	*cmd*, **args*, ***kwargs*	

Altogether 19 keywords.
Generated by Libdoc on 2018-12-06 13:39:51.

vmware

Library scope:	global
Named arguments:	supported

Introduction

provides function for VMware ESXI

Athough this module provides some keywords for interact with the console (WebConsole), they only suppport very primitive actions. Do not use this to accomplish complex tasks. Instead, using VChannel through a SSH/Telnet channel.

Shortcuts

Capture Mks Screenshot · **C**lick Mks · **G**et Mks Ticket · **G**et Vm Id · **G**et Vm List · **G**et Vm Power State · **O**pen Console · **P**ower Off · **P**ower On · **R**eset Capture Counter · **S**end Mks Cmd · **S**end Mks Key · **S**et Capture Format

Keywords

<table>
<tr><th>Keyword</th><th>Arguments</th><th>Documentation</th></tr>
<tr><td><b>Capture Mks Screenshot</b></td><td><i>self</i>, <i>filename=None</i>, <i>extra=</i></td><td>Captures the current web console to <i>filename</i>
If <i>filename</i> is `None`, the captured filename will be decided by the current `format` with an auto-increment counter and a `extra` at the end.
Example:
<table>
<tr><td><i>Hypervisor</i>.Capture MKS Screenshot</td><td># will create a file console_0000000001.png</td><td></td></tr>
<tr><td><i>Hypervisor</i>.Capture MKS Screenshot</td><td>xxx.png</td><td># will create a file xxx.png</td></tr>
<tr><td><i>Hypervisor</i>.Capture MKS Screenshot</td><td># will create a file console_0000000002.png</td><td></td></tr>
</table></td></tr>
<tr><td><b>Click Mks</b></td><td><i>self</i>, <i>xoffset</i>, <i>yoffset</i></td><td>Click on the MKS console at <i>xoffset</i>,<i>yoffset</i> coordinate
<b>Notes</b>: The coordinate (0,0) is at the left corner of the console screen</td></tr>
<tr><td><b>Get Mks Ticket</b></td><td><i>self</i>, <i>vm_name</i></td><td>Returns a MSK ticket for WebConsole</td></tr>
<tr><td><b>Get Vm Id</b></td><td><i>self</i>, <i>vm_name</i></td><td>Returns a VMID of a VM</td></tr>
<tr><td><b>Get Vm List</b></td><td><i>self</i></td><td>Returns current VM name list of the hypervisor</td></tr>
<tr><td><b>Get Vm Power State</b></td><td><i>self</i>, <i>vm_name</i></td><td>Get vm power status
Return `on` of `off`</td></tr>
<tr><td><b>Open Console</b></td><td><i>self</i>, <i>vm_name</i>, <i>width=None</i>, <i>height=None</i></td><td>Opens a web console for a VM <i>vm_name</i>
Returns the <i>width</i> and <i>height</i> of the console Examples:
<table>
<tr><td>Hypervisor.<i>Set Capture Format</i></td><td>console_%010d</td><td></td></tr>
<tr><td><i>Open Console</i></td><td>${VM_NAME}</td><td></td></tr>
<tr><td>Hypervisor.<i>Capture MKS Screenshot</i></td><td></td><td></td></tr>
<tr><td><i>Send MKS Cmd</i></td><td>root</td><td></td></tr>
<tr><td><i>Send MKS Cmd</i></td><td>password</td><td>wait=10s</td></tr>
<tr><td><i>Send MKS Cmd</i></td><td>ls</td><td></td></tr>
<tr><td>Hypervisor.<i>Capture MKS Screenshot</i></td><td></td><td></td></tr>
<tr><td><i>Send MKS Key</i></td><td>${CTRL_L}</td><td></td></tr>
<tr><td><i>Send MKS Cmd</i></td><td>whoami</td><td></td></tr>
<tr><td>Hypervisor.Capture MKS Screenshot`</td><td></td><td></td></tr>
</table></td></tr>
<tr><td><b>Power Off</b></td><td><i>self</i>, <i>vm_name</i>, <i>graceful=True</i></td><td>Shutdowns a VM
If <i>graceful</i> is True, a graceful shutdown is tried before a power off.
<b>Note</b>: if VMware tools is not install on the VM, graceful shutdown is not available</td></tr>
<tr><td><b>Power On</b></td><td><i>self</i>, <i>vm_name</i></td><td>Power on a VM</td></tr>
<tr><td><b>Reset Capture Counter</b></td><td><i>self</i></td><td></td></tr>
<tr><td><b>Send Mks Cmd</b></td><td><i>self</i>, <i>cmd</i>, <i>wait=2s</i></td><td>Sends command to current web console and wait for a while
By default, <i>wait</i> time is `2s` and the keyword will automaticall add a `Newline` char after sending the <i>cmd</i></td></tr>
<tr><td><b>Send Mks Key</b></td><td><i>self</i>, <i>key</i>, <i>wait=1s</i></td><td>Sends key strokes to current web console
Special Ctrl char could be used as `${CTRL_A}` to `${CTRL_Z}`
Examples:</td></tr>
</table>

		Send MKS Key ${CTRL_L}
Set Capture Format	*self*, *format*	Set console capture format Initialized format is `'vmware_%010d'`

Altogether 13 keywords.
Generated by Libdoc on 2018-12-06 13:39:52.