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 Logging · **C**sv Snapshot · **G**et All Test Result · **G**et All Views All Csv Logs · **G**et All Views Csv Log · **G**et Csv Log · **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 · **G**et View All Csv Logs · **G**et View Csv Log · **L**ink Up Down By Index · **L**ink Up Down By Name · **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 .*

		▪ 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 Logging	*self*, *enabled=True*, **views*	Toggles enable/disable CSV loggin for a view Parameters: ▪ *views*: is a list of views. *None* means all views. Result files will have format <View name>.index.csv, when index is automatically increased everytime the view is disable and re-enable again in the same test. Samples: Tester.*CSV Logging* ${TRUE} Flow Statistics Sleep 10s Tester.*CSV Logging* ${FALSE} Flow Statistics **Note**: Long time enable fof CSV loggin could returns in very big file
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 Samples: Tester.*CSV Snapshot* snapshot03_ # collect all views Tester.*CSV Snapshot* snapshot03_ Port Statistics Flow Statistics # collect specific views **Note**: the name of result file will be modified so *space* will be replaced by *underbar*. Depending on the traffic status, the available views could be varied. For example, the view *Flow Statistics* is not available when there is no traffic.
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 All Views All Csv Logs	*self*, *prefix=*	Gets all CSV logs for all available views
Get All Views Csv Log	*self*, *prefix=*	Gets the newest CSV log of all available views
Get Csv Log	*self*, *prefix=*, *index=-1*, **views*	Gets all CSV log for a specific views or all from current test folder Parameters: ▪ *views* is a list of views. *None* is all views. ▪ *prefix* will be appended automatically to the beginning of the result ▪ *range*: number of files from the newest data. *-1* for only 1 newest and *:* for all files. Samples: Tester.*Get CSV Log* single_ -1 # get the newest CSV logging data for all views Tester.*Get CSV Log* all_ : Flow Statistics # get all CSV logging data for one view
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. **Note**: the name of the result files are modified so that *space* will become *underbar*, *hyphen* will be deleted.
Get View All Csv Logs	*self*, *view*, *prefix=*	Gets the newest CSV log file of **ALL** available views
Get View Csv Log	*self*, *view*, *prefix=*	Gets the newest CSV log file of the specific view
Link Up Down By Index	*self*, *port_index=0*, *state=up*	Simulates a LinkUpDown by port index Parameters: ▪ *port_index*: zero-started port index ▪ *state*: is *up* or *down* Samples: Tester.*Link Up Down By Index* 0 down Sleep 5s Tester.*Link Up Down By Index* 0 up
Link Up Down By Name	*self*, *port_name*, *state=up*	Simulates a LinkUpDown by port name Parameters: ▪ *port_index*: zero-started port index ▪ *state*: is *up* or *down* Samples: Tester.*Link Up Down By Name* Ethernet - 001 down Sleep 5s Tester.*Link Up Down By Name* Ethernet - 001 up
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) ▪ `media` : *copper* or *fiber* (**Note**: no default value) See Common for more details about the yaml configuration files.
Load Traffic	*self*, *wait_time=2m*, *wait_time2=2m*, *apply=True*, *protocol=True*, *force=True*, *tx_mode=interleaved*	
Loss From File	*self*, *file_name=Flow_Statistics.csv*, *index=0*	Returns `packet loss` by miliseconds and *delta frame*. Parameters: ▪ *file_name*: flow information (csv format). Default is `Flow_Statistics.csv` ▪ *index*: row index of the result(counted from zero)

		- *index*: row index of the result(counted from zero) Samples: | ${LOSS} | ${DELTA}= | Tester.*Loss From File* | Flow_Statistics.csv | | | ${LOSS} | ${DELTA}= | Tester.*Loss From File* | Flow_Statistics.csv | index=1 | **Note**: The calculation should be performed when traffic is stopped. The calculation supposed traffic is configured by frame per second.
Ping	*self*, *dst_ip*, *src_port_index=0*, *src_intf_index=0*	Ping from Ixia to `dst_ip` The keyword return the output string as it is. The return could be - 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> Parameters: ■ src_port_index: index of Ixia port (starts from 0) ■ src_intf_index: index of interface insides the port (starts from 0) Examples: | Tester.*Ping* | 1.1.1.1 | 0 | 0 | | Tester.*Ping* | 1.1.1.1 | | |
Regenerate	*self*	Regenerates **all** flow of current traffic items
Reset Config	*self*	Clears current config and creates new blank config
Run Quicktest	*self*, *test_index=0*, *wait_until_finish=True*	Runs the Quicktest and wait until it finishes **Warning**: it could take a long time to finish a quicktest
Set All Traffic Item	*self*, *enabled=True*	Enables/Disables **all** traffic items at once
Set Bgp Items	*self*, *port_index*, *neighbor_index*, *route_range_index*, *is_enable*	Enables/Disables BGP entry by a set of port,neighbor,route_range index 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: | Tester.*Set BGP Items* | 0 | * | * | ${FALSE} | | Tester.*Set BGP Items* | 0 | * | * | ${TRUE} |
Set Bgp Neighbor	*self*, **indexes*, ***kwargs*	Enables/Disables BGP entry by neighbor index `kwargs` contains following parameters: ■ indexes: is a list of index of BGP neighbor (index is started from zero) ■ vport_index: is the target vport index ■ enabled: TRUE or FALSE Examples: | Tester.*Set BGP Item* | 0 | 1 | vport_index=0 | enabled=${FALSE} \| \| Tester.*Set BGP Item* \| 0 \| 1 \| vport_index=1 \| enabled=${TRUE} |
Set Capture Port	*self*, *data_mode=True*, *control_mode=True*, *port_index=0*	Capture packets for follow `port` `port_index`: is a index of current test port (start from 0) `data_mode`: capture data packets and save in <intf>_HW.cap file `control_mode`: capture controls packets and save in <intf>_SW.cap file **Note**: `control_mode` saves all control packets and `data_mode` only saves data packets. **Note**: `control_mode` saves all control packets and `data_mode` only saves data packet Examples: | Tester.*Set Capture Port* | 0 | | | | Tester.*Set Capture Port* | control_mode=${TRUE} | 0 | 1 |
Set Traffic Item	*self*, **items*, ***kwargs*	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`) **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: | Set Traffic Item | Traffic Item 1 | Traffic Item 2 | | Set Traffic Item | @{item_list} |

		Set Traffic Item | Traffic Item 1 | enabled = ${FALSE}
Should Be Pingable	*self*, *dst_ip*, *src_port_index=0*, *src_intf_index=0*	Ping from Ixia and raise an error if ping fails The keyword return *True* if succeeds
Start Capture	*self*, *wait_time=30s*	Start packet capture Target ports are set by the configuration file or by [Set Capture] keyword
Start Protocol	*self*, *wait_time=1m*	Starts all protocols and wait for `wait_time` Default `wait_time` is 1 minute. Make sure `wait_time` is big engouh to start all protocols.
Start Test Composer	*self*, *script_name=Main_Procedure*, *run_num=1*, *wait_for_test=True*, *parameter=*, *wait=10s*	Run a test composer script. The test composer script should be included in an Ixia Network configuration file and loaded properly with *Load Config* Parameters: ▪ `script_name` is the name of the script to run. Default value is `Main_Procedure`. ▪ `wait_for_test`: if `${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: Tester.*Start Test Composer* | parameter=XXX YYY Tester.*Get Test Composer Result* | result_file=script1.log Tester.*Start Test Composer* | parameter={{VAR1 AAA} {VAR2 BBB}} Tester.*Get Test Composer Result* | result_file=script1.log
Start Traffic	*self*, *wait_time=30s*	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.
Stop All Protocols	*self*, *wait_time=30s*	Stop all running protocols
Stop And Save Capture	*self*, *prefix=*, *wait_until_finish=True*, *monitor_interval=5s*	Stop current capture and save the resuls to folder specified by `path` Captured files will be saved in current `result` folder with `prefix` appended in their names. Examples: Tester.*Start Capture* | | Sleep | 10s Tester.*Stop And Save Capture* | ${RESULT_FOLDER}/capture.zip
Stop Quicktest	*self*, *test_index=0*	Stops a running test
Stop Test Composer	*self*, *wait=10s*	Stop a running composer Do nothing when a test composer has already stopped or no composer has been prepared.
Stop Traffic	*self*, *stop_protocol=False*, *wait_time=10s*	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
Wait Until Connected	*self*, *timeout_str=5m*	Waits until ports become enabled and connected