Router

Library version:	RENAT 1.7.1
Library scope:	test suite
Named arguments:	supported

Introduction

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

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

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

- Juniper module
- Cisco module
- GR module

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

Router.*Xrun*	Load Config
Router.*Load Config*	

Shortcuts

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

Keywords

Keyword	Arguments	Documentation

Cmd	*command=*, *prompt=*	Runs the command `command` and waits until the prompt defined for this router. This keyword is identical to `VChannel.Cmd`
		Examples:
		Router.*Cmd* | set system login user testtest authentication plain-text-password | password: | # wait for *password:*
		Router.*Cmd* | Renat2017 | password: | # wait for *password:*
		Router.*Cmd* | Renat2017 | | # wait for default prompt
		The above sample creates an output likes this:
		`user@vmx11# set system login user testtest authentication plain-text-password`
		`New password:Renat2017`
		`Retype new password:Renat2017`
		
		`[edit]`

Cmd And Wait For	*command*, *keyword*, *interval=30s*, *max_num=10*, *error_with_max_num=True*	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.

Exec File	*file_name*, *vars=*, *comment=#* , *step=False*, *str_error=syntax,rror*	Executes commands listed in `file_name` Lines started with `comment` character is considered as comments
		`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 *error*
		A sample for command list with Jinja2 template:
		`show interface {{ LOCAL['extra']['line1'] }}`
		`show interface {{ LOCAL['extra']['line2'] }}`
		
		`{% 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`

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)---------->
Get Ip		Returns the IP address of current node Examples: ${router_ip}= | Router.*Get IP* |
Read		Executes command `read` for the current vchannel coressponded to this router
Snap	*name*, **cmd_list*	Remembers the result of a list of command defined by `cmd_list` 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
Stop Screen Mode		Stop the screen mode
Switch	*name*	Changes the current channel of this router to `name` Rerturns old node name **Note:** This is identical to VChannel.Switch Examples: Router.*Switch* | vmx11 | Router.*Cmd* | show version |
Write	*cmd_str*, *wait_str=1s*, *start_screen_mode=False*	Executes command `write` for the current vchannel coressponded to this router
Xrun	*cmd*, **args*, ***kwargs*	Runs the vendor independent keywords. 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 12 keywords.
Generated by Libdoc on 2018-04-12 19:20:23.