

VChannel

Library version: RENAT 0.1.8g
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 Yesno · Connect · Connect All · Flush All · Get Channel · Get Channels · Get Current Channel · Log · Read · Reconnect · Set Log Separator · Start Screen Mode · Stop Screen Mode · Switch · Write

Keywords

Keyword	Arguments	Documentation															
Change Log	<i>log_file, mode=w</i>	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															
Change Prompt	<i>str_prompt</i>	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.Switch</td><td>vmx11</td><td></td></tr><tr><td>\${prompt}=</td><td>VChannel.Change Prompt</td><td>%</td></tr><tr><td>VChannel.Cmd</td><td>start shell</td><td></td></tr><tr><td>VChannel.Cmd</td><td>ls</td><td></td></tr><tr><td>VChannel.Change Prompt</td><td>\${prompt}</td><td></td></tr></table>	Router. Switch	vmx11		\${prompt}=	VChannel. Change Prompt	%	VChannel. Cmd	start shell		VChannel. Cmd	ls		VChannel. Change Prompt	\${prompt}	
Router. Switch	vmx11																
\${prompt}=	VChannel. Change Prompt	%															
VChannel. Cmd	start shell																
VChannel. Cmd	ls																
VChannel. Change Prompt	\${prompt}																

		Vchannel. Cmd	exit														
Close		Closes current connection and returns the active channel name															
Close All		Closes all current sessions and flush out all log files. Current node name was reset to <code>None</code>															
Cmd	<i>command, prompt=, match_err=(unknown command./syntax error, expecting <command>.)</i>	Executes a <code>command</code> and wait until for the prompt. This is a blocking keyword. Execution of the test case will be postponed until the prompt appears. If <code>prompt</code> is a null string (default), its value is defined in the <code>./config/template.yaml</code> Output will be automatically logged to the channel current log file. See Common for details about the config files.															
Cmd Yesno	<i>cmd, ans=yes, question=? [yes,no]</i>	Executes a <code>cmd</code> , waits for <code>question</code> and answers that by <code>ans</code>															
Connect	<i>node, name, log_file, timeout=20m, w=80, h=32, mode=w</i>	Connects to the node and create a VChannel instance Login information is automatically extracted from yaml configuration. By default a virtual terminal (vty100) with size 80x64 is attached to this channel. If a login was successful, VChannel will create a log file name <code>log_file</code> 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 Switch 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 <code>Common</code> for more detail about the yaml config files.				Connect	vmx11	vmx11	vmx11.log			Connect	vmx11	vmx11	vmx11.log	80	64
Connect	vmx11	vmx11	vmx11.log														
Connect	vmx11	vmx11	vmx11.log	80	64												
Connect All	<i>prefix=</i>	Connects to all nodes that are defined in active <code>local.yaml</code> . A prefix <code>prefix</code> was appended to the alias name of the connection. A new log file by <code><alias>.log</code> was automatically created. See <code>Common</code> for more detail about active <code>local.yaml</code>															
Flush All																	
Get Channel	<i>name</i>	Returns a channel by its <code>name</code>															
Get Channels		Returns all current vchannel instances															
Get Current Channel		Returns the current active channel															
Log	<i>msg</i>	Writes the log message <code>msg</code> to current log file of the channel															
Read	<i>silence=False</i>	Returns the current output of the virtual terminal and automatically logs to file. In <code>normal mode</code> this will return the unread output only, not all the content of the screen.															
Reconnect	<i>name</i>	Reconnects to the <code>name</code> node using existed information The only difference is that the mode of the log file is set to <code>`a+`</code> by default															
Set Log Separator	<i>sep=</i>	Set a separator between the log of <code>read</code> , <code>write</code> or <code>cmd</code> keywords															
Start Screen Mode		Starts the <code>screen mode</code> . In the <code>screen mode</code> , the output is just the same with the real terminal. It means that any real-time application likes <code>top</code> will be captured as-is. Consecutive read from this VChannel instance may produce redundancy output.															
Stop Screen Mode		Stops the <code>screen mode</code> and returns to <code>normal mode</code> In <code>screen mode</code> , Write does not return any thing and no output is logged. In <code>normal mode</code> , escape sequences are not processed by the virtual terminal.															
Switch	<i>name</i>	Switches the current active channel to <code>name</code> . There only one active channel at any time Examples: <table><tr><td>VChannel.Switch</td><td>vmx12</td></tr></table>				VChannel. Switch	vmx12										
VChannel. Switch	vmx12																
Write	<i>str_cmd, str_wait=0s, start_screen_mode=False</i>	Sends <code>str_cmd</code> to the target node and return after <code>str_wait</code> time. If <code>start_screen_mode</code> is <code>True</code> , the channel will be shifted to <code>Screen Mode</code> . Default value of <code>screen_mode</code> is <code>False</code> . In <code>normal mode</code> , a <code>new line</code> char will be added automatically to the <code>str_cmd</code> and the command return the output it could get at that time from the terminal and also logs that to the log file. In <code>screen Mode</code> , if it is necessary you need to add the <code>new line</code> char by your own and the output 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 with no output.

Notes: This is a non-blocking command.

Examples:

VChannel. Write	monitor interface traffic	start_screen_mode=\${TRUE}
VChannel. Write	<code>\${CTRL_C}</code>	# simulates Ctrl-C

Altogether 20 keywords.

Generated by [Libdoc](#) on 2018-05-09 01:01:24.

