cli-shell-api

From VyOS Wiki

Since release 6.2 (Mendocino) Vyatta has an API for working with configuration from shell scripts. Its binary is **/opt/vyatta/sbin/my_cli_shell_api** and has symbolic link at **/bin/cli-shell-api**. This page describes methods of the API as of Vyatta 6.4 (Oxnard) release.

For a less technical tutorial, see Shell script tutorial page.

Contents

Definitions

Active config is config, currently used by system.

Working config is config we are making during configuration session.

Effective config

The definition of "effective" is different under these two scenarios.

1. When used outside a config session, "effective" == "active". In other words, in such cases the effective config is the same as the running config.
2. When used during a config session, a config path (leading to either a "node" or a "value") is "effective" if ANY of the following is true.

- active && working

Path is in both active and working configs, i.e., unchanged.
- !active && working && committed
  Path is not in active, has been set in working, AND has already been committed, i.e., "commit" has successfully processed the addition/update of the path.
- active && !working && !committed
  Path is in active, has been deleted from working, AND has not been committed yet, i.e., "commit" (per priority) has not processed the deletion of the path yet, or it has been processed but failed.

Note: during commit, deactivate has the same effect as delete. So in such cases, as far as these functions are concerned, deactivated nodes don't exist.

Method reference

It is invoked in format:

```
cli-shell-api <method> <configuration path>
```

Currently API has the following methods:

Method	Arguments	Purpose
getSessionEnv	Session identifier	Returns environment variables needed for configuration session to work[1]
getEditEnv	Configuration path	Returns environment variables for edit level specified in argument.
getEditUpEnv	None	Returns environment variables for edit level above current edit level. Returns string "Already at the top level" when ran at the top edit level.
getEditResetEnv	None	Returns environment variables for the top level.
editLevelAtRoot	None	Returns 0 if current edit level is top level, 1 otherwise
getCompletionEnv	Command and component (e.g. "set service" or "edit firewall")	Returns environment variables needed for command completion to work.
getEditLevelStr	None	Returns current edit level.
markSessionUnsaved	None	Mark current configuration session unsaved.
unmarkSessionUnsaved	None	Reset session unsaved flag.
sessionUnsaved	None	Returns 0 when session unsaved flag is set, 1 otherwise.
setupSession	None	Initiate a configuration session. Needs environment to be set properly, see getSessionEnv.

sessionChanged	None	Returns 0 if configuration was changed from current session, 1 otherwise.
teardownSession	None	End current configuration session.
inSession	None	Returns 0 if configuration session is set up, 1 otheriwise.
exists	Configuration path	Returns 0 if specified configuration path exists (either in currently used or built during the session config), 1 otherwise.
existsActive	Configuration path	Returns 0 if specified node exists in the current active (running) configuration, 1 otherwise.
existsEffective	Configuration path	Returns 0 if specified path exists in effective config, 1 otherwise
isMulti	Configuration path	Returns 0 if specified node is a multi node (i.e. may have more than one value), 1 otherwise
isTag	Configuration path	Returns 0 if specified node is a tag node, 1 otherwise
Configuration path	???	
isLeaf	Configuration path	Returns 0 if specified node is a leaf node, 1 otherwise
getNodeType	Configuration path	Returns on of the following: **value** for value nodes, **leaf** for leaf nodes, **multi** for multi nodes, **tag** for tag nodes, **non-leaf** for the rest.
listNodes	Configuration path	Returns list of nodes under specified configuration path[2].
listActiveNodes	Configuration path	Returns list of nodes under specified configuration path that are present in currently used config.
listEffectiveNodes	Configuration path	Returns list of effective nodes under specified configuration path that are present in effective config.
returnValue	Configuration path	Returns value of a node under specified configuration path.
returnActiveValue	Configuration path	Returns value of a node under specified configuration path as present in currently used config.
returnEffectiveValue	Configuration path	Returns effective value of a node under specified configuration path as present in currently used config.
returnValues	Configuration path	Returns values of a multinode under specified configuration path[3].
returnActiveValues	Configuration path	Returns values of a multinode under specified configuration path as present in currently used config.
returnEffectiveValues	Configuration path	Returns effective values of a multinode under specified configuration path as present in currently used config.
validateTmplPath	Configuration path	Validate the path regardless of given value (e.g. "interfaces ethernet" is a valid path whereas "interfaces foobar" is not.). Returns 0 if path is valid, 1 otherwise.

validateTmplValPath	Configuration path	Validate configuration path with respect to value (e.g. "interfaces ethernet" is a valid path, whereas "interfaces foo" is not). Returns 0 if path is valid, 1 otherwise.
validateTmplValPath	Configuration path	Validate configuration path with respect to value (e.g. "interfaces ethernet eth0" is a valid path, whereas "interfaces ethernet foo0" is not). Returns 0 if path is valid, 1 otherwise.
showCfg	Configuration path (may be empty)	Shows configuration under specified path.
showConfig	Configuration path (may be empty)	Show configuration under specified path. Supports the following options: --show-active-only — show active configuration --show-working-only — show working configuration --show-show-defaults — include default value --show-hide-secrets — replace private information like passwords with "*" --show-context-diff — show "context diff" between two configs --show-commands — show output in "commands" --show-ignore-edit — don't use the edit level in environment --show-cfg1 <cfg1> --show-cfg2 <cfg2> — specify the two configs to be diffed (must specify both), values may be file names, "@ACTIVE" or "@WORKING"
loadFile	Path to config file	Load configuration file
getPreCommitHookDir	None	Returns path to pre-commit hooks directory
getPostCommitHookDir	None	Returns path to post-commit hooks directory
cfExists	Path to config file, configuration path	Returns 0 if specified configuration path exists in specified config file
cfReturnValue	Path to config file, configuration path	Returns value of specified node in specified file
cfReturnValues	Path to config file, configuration path	Returns values under specified path in specified file

Usage

Setting up the session

Before changing configuration and using most part of cli-shell-api methods you must set up session. Current best practice is to use process identifier ($PPID) as session identifier.

```
# Obtain session environment
session_env=$($SHELL_API getSessionEnv $PPID)

# Evaluate environment string
eval $session_env

# Setup the session
cli-shell-api setupSession
```

Then you can make sure session is set up:

```
cli-shell-api inSession
if [ $? -ne 0 ]; then
  echo "Something went wrong!"
fi
```

Don't forget to finish your session using cli-shell-api teardownSession. In a bash script make sure session is finished using something like:

```
function atexit() {
    cli-shell-api teardownSession
}
trap atexit EXIT
```

Configuration output

You can do configuration output even if session is not set up. Example:

```
# cli-shell-api showCfg firewall name TEST rule 10
   action reject
   source {
       address 172.16.0.0/24
   }
```

Working with multinode output

To work with output of listNodes, returnValues or similar methods you must eval it. Example:

```
node_list=$(cli-shell-api listNodes firewall name TEST)
eval "NODES=($node_list)"

for i in "${NODES[@]}";
  do
    cli-shell-api showCfg firewall name TEST rule $i
  done
```

Modifying configuration

Warning: You must set up a session before modifying configuration.

cli-shell-api itself does not have methods to modify configuration. It is done with separate commands that are in /opt/vyatta/sbin (${vyatta_sbindir}) and have names same to configuration mode commands with "my_" prefix. The only exception is command for saving configuration, which is /opt/vyatta/vyatta-save-config.pl (accepts config file name as its optional argument).

You may use the following snippet:

```
SET=${vyatta_sbindir}/my_set

DELETE=${vyatta_sbindir}/my_delete

COPY=${vyatta_sbindir}/my_copy

MOVE=${vyatta_sbindir}/my_move

RENAME=${vyatta_sbindir}/my_rename

ACTIVATE=${vyatta_sbindir}/my_activate

DEACTIVATE=${vyatta_sbindir}/my_activate

COMMENT=${vyatta_sbindir}/my_comment

COMMIT=${vyatta_sbindir}/my_commit

DISCARD=${vyatta_sbindir}/my_discard

SAVE=${vyatta_sbindir}/vyatta-save-config.pl
```

Example:

```
$SET interfaces ethernet eth0 address 10.0.0.1/24
$COMMIT
```

Script examples

- Search for nodes containing some string.

References

1. This and all other methods that return environment variables return a string suitable for "eval".
2. This and other listNodes* methods return strings that must be eval'ed into an array (e.g. values=$(cli-shell-api listNodes interfaces); eval "nodes=($values)")
3. Output of this and all other returnValues* methods is a string that must be eval'ed.

Retrieved from "https://wiki.vyos.net/index.php?title=Cli-shell-api&oldid=744"

Category: Development

- This page was last edited on 5 April 2015, at 23:26.
- Content is available under GNU Free Documentation License 1.3 or later unless otherwise noted.