Cinderdiags

# RunninG tests

The cli can be tested in Tempest or by a Python script without Tempest.

### TEMPEST

* Copy ./diags/tempest/test\_cinder\_diagnostics\_cli\_tool.py to any tempest directory
* Run the command to execute the test cases:
  + nosetests test\_cinder\_diagnostics\_cli\_tool.py

### without tempest

* python ./diags/cli/test/test\_cinder\_diagnostics\_cli\_tool.py

## conf\_reader.py

conf\_reader is the only module the cli cliff listers interact with. It parses the cli.conf file to identify cinder nodes and nova nodes.

#### software\_check()

Reader.software\_check() is called by the software.py cliff lister when the command ‘software-check’ is used. It creates ssh clients with ssh\_client.py for cinder nodes, nova nodes, or both (default) and uses pkg\_checks.py to check for a user provided software package and (optional) version number or checks for all default packages listed in constant.py.

software\_check() returns a list of dictionaries. Each dictionary is the check results for a package/node pair and must contain the following keys: “node”, “name”, “installed”, and “version”. Changing these requires changing software.py.

#### options\_check()

Reader.options\_check() is called by the options.py cliff lister when the command ‘options-check’ is used. It creates ssh clients with ssh\_client.py for cinder nodes. It then uses ssh\_client.py to copy cinder.conf files (location specified in cli.conf) from cinder nodes to a temp directory (specified in constant.py). Finally it uses hp3par\_wsapi\_checks.py to check all cinder nodes for user provided cinder.conf backend section or all backend sections in each node’s cinder.conf.

options\_check() returns a list of dictionaries. Each dictionary is the check results for a backend section/node pair.

To support non-3PAR arrays

Add the results of your options checks to the ‘checks’ list. options\_check() must return a list of dictionaries and all dictionaries must contain the following keys: “name”, “url”, “credentials”, “cpg”, “iscsi”, “node”, “driver”. Changing these requires changing options.py as well as hp3par\_wsapi\_checks.py.

## constant.py

constant.py contains all hardcoded values used in cinderdiags:

* DIRECTORY: The temp file location that cinder.conf files will be saved to once copied from the cinder nodes.
* TEST\_CLI\_CONFIG: The absolute path to cli.conf used for testing cli tool.
* CLI\_CONFIG: The path setup.py installs cli.conf to and conf\_reader.py reads cli.conf from. This must match data\_files in setup.py.
* NOVA\_PACKAGES: The default software packages to be checked for on nova nodes. Each package is a tuple: (‘name’, ‘minimum version number’).
* CINDER\_PACKAGES: The default software packages to be checked for on cinder nodes. Each package is a tuple: (‘name’, ‘minimum version number’).
* HP3PAR\_DRIVERS: The list of valid 3PAR driver modules. hp3par\_wsapi\_checks.py validates all volume\_drivers end with one of these strings.

## hp3par\_testclient.py

hp3par\_testclient.py is a dummy hp3parclient used for testing cli tool so that no connections to real 3PAR arrays are needed. It raises exceptions or returns values to simulate connecting to 3PAR web service API.

## hp3par\_wsapi\_check.py

hp3par\_wsapi\_check.py performs the options checks for 3PAR arrays by validating the 3PAR options in each cinder.conf backend section.

It parses and checks each node’s cinder.conf file and identifies 3PAR backend sections by “hp\_3par” being in the “volume\_driver” string.

It returns a list of dictionaries where each dictionary is the check results of one backend section:

* “name”: The option group name identified by configparser. (Example: [3PAR-ISCSI])
* “url”: Corresponds to the “hp3par\_api\_url” option. It is set to “fail” if hp3parclient is unable to connect to the url.
* “credentials”: Corresponds to the “ hp3par\_username” and “hp3par\_password” options. It is set to “fail” if hp3parclient is unable to login.
* “cpg”: Corresponds to the “hp3par\_cpg” option. It is set to “fail” if *any* CPG names are invalid for the array.
* “iscsi”: Corresponds to the “hp3par\_iscsi\_ips” option. It is set to “fail” if *any* iSCSI IP addresses are invalid.
* “node”: The node that cinder.conf was copied from.
* “driver”: Corresponds to the “volume\_driver” option. It fails if volume\_driver does not end in a valid 3par driver or if the path does not exist on the node. This check requires mlocate to be installed on the cinder node that is being checked. (This option is not 3par specific.)

## pkg\_checks.py

pkg\_checks.py checks for software packages installed on nodes by executing commands via SSH. get\_check\_type() determines the linux flavor of the node by inspecting the release information file. If it detects debian linux, dpkg\_check() looks for packages installed using apt-get. If it detects fedora, yum\_check() looks for packages installed using yum. If it detects suse, zypper\_check() looks for packages installed using zypper. In all cases, if the package is not found using one of these checks, pip\_check() checks for packages installed using pip. If no minimum version is provided, the version check is set to ‘N/A’.

If a specific package is not provided by the user, different default packages are checked according to node type (cinder or nova). To change default package checks, edit constant.py

The user or constant.py can provide alternatives for package names by putting ‘||’ between them. (Example: sg3-utils is also called sg3\_utils so the default name is “sg3-utils || sg3\_utils”.)

## ssh\_client.py

ssh\_client.py creates ssh clients to nodes using paramiko module.

get\_file() copies files from the node to the local machine and returns path on local machine.

execute() executes commands on nodes and returns string containing stdout and stderr generated by command.