Nidhogg Documentation

Release 3.3

Roland Wohlfahrt, Christian Assing

July 06, 2016

Contents

Note: NetApp Interface done right!

Contents:

CHAPTER 1

Get me started!

nidhogg.**get_netapp**(*url*, *username*, *password*)

Return the correct connection object to the filer.

You do not have to care if the filer is a cluster-mode or a seven-mode filer.

Note: Provided user must be authorized to use the Netapp API of the filer.

Parameters

- **url** (*str*) – hostname of the netapp filer
- **username** (*str*) – username to connect to the Netapp API.
- **password** (*str*) – password of the provided user

Returns Nidhogg instance

Return type *SevenMode* (if the filer is a seven-mode filer)

Return type *ClusterMode* (if the filer is a cluster-mode filer)

Example:

```
import nidhogg
filer = nidhogg.get_netapp("filer99.example.com", "<username>", "<password>")
filer.list_volumes()
```

nidhogg.**get_best_volume_by_size**(*volumes*, *filter_func=None*, ***kwargs*)

Return the best volume from the list of volumes with the biggest free size.

Apply filter function before if specified.

Parameters

- **volumes** (list of *Volume*) – list of volumes
- **filter_func** (*function*) – filter function applied before

Returns volume with the biggest free size

Return type *Volume*

nidhogg.**get_best_volume_by_quota**(*volumes*, *filter_func=None*, ***kwargs*)

Return the best volume from the list of volumes with the smallest quota ration.

Parameters

- **volumes** (list of *VolumeWithQuotaRatio*) – list of volumes
- **filter_func** (*function*) – filter function applied before

Returns volume with the smallest quota ratio (allocated quota size / volume size)

Return type *VolumeWithQuotaRatio*

CHAPTER 2

nidhogg API

class nidhogg.core.**Nidhogg**(*url*, *username*, *password*, *major*, *minor*, *http=<class 'nidhogg.http.NidhoggHttp'>*)

Bases: object

This is the base class for connecting to a NETAPP filer.

It provides functions that have 7-mode filers and cluster-mode filers in common.

Subclasses:

- *SevenMode*
- *ClusterMode*

apis

List of API commands available with the current credentials.

Returns list of API commands

Return type list of str or empty list

clustered

True if the filer is a cluster-mode filer, false otherwise.

Return type boolean

create_cifs_share(**args*, ***kwargs*)

See sub classes.

- Go to *create_cifs_share()* (SevenMode)
- Go to *create_cifs_share()* (ClusterMode)

create_qtree(*volume*, *qtree*, *mode=u'007'*)

Create a qtree on the specified volume.

Parameters

- **volume** (*str*) – name of the volume
- **qtree** – name of the qtree to be created
- **mode** (*str*) – initial file system permissions of the qtree

Raises NidhoggException if an error occurs

create_snapshot(*volume*, *name*)

Create a snapshot.

Parameters

- **volume** (*str*) – name of the volume
- **name** (*str*) – name of the snapshot

Raises NidhoggException if an error occurs

delete_cifs_acl(**args*, ***kwargs*)
See sub classes.

•Go to *delete_cifs_acl()* (SevenMode)

•Go to *delete_cifs_acl()* (ClusterMode)

delete_cifs_acls(**args*, ***kwargs*)
See sub classes.

•Go to *delete_cifs_acls()* (SevenMode)

•Go to *delete_cifs_acls()* (ClusterMode)

delete_cifs_share(*share_name*)
Delete the share with the given name.

Parameters **share_name** (*str*) – name of the share

Raises NidhoggException if an error occurs

delete_qtree(*volume*, *qtree*, *force=False*)
Delete a qtree on the specified volume.

Parameters

- **volume** (*str*) – name of the volume
- **qtree** (*str*) – name of the qtree to be deleted
- **force** (*bool*) – force deletion if true

Raises NidhoggException if an error occurs

delete_quota(**args*, ***kwargs*)
See sub classes.

•Go to *delete_quota()* (SevenMode)

•Go to *delete_quota()* (ClusterMode)

delete_snapshot(*volume*, *name*)
Delete a snapshot.

Parameters

- **volume** (*str*) – name of the volume
- **name** (*str*) – name of the snapshot

Raises NidhoggException if an error occurs

exists_qtree(*volume*, *qtree*)
Check if a qtree exits.

Parameters

- **volume** (*str*) – name of the volume
- **qtree** (*str*) – name of the qtree

Returns true, if the qtree exists

Return type bool

Raises NidhoggException if an error occurs

get_allocated_quota_ratio(**args*, ***kwds*)
Return the ratio *allocated quota size / volume size*.

Parameters

- **volume** (*str*) – name of the volume
- **volume_size_total** (*int*) – if specified we have the total size already from a previous API call

Returns ratio *allocated quota size / volume size*

Return type int

Raises NidhoggException if an error occurs

get_allocated_quota_size(**args*, ***kwds*)
Return the sum of all quotas of the specified volume.

Parameters **volume** (*str*) – name of the volume

Returns sum of all qtree quotas on this volume in byte

Return type int

Raises NidhoggException if an error occurs

get_quota(**args*, ***kwargs*)
See sub classes.

•Go to *get_quota()* (SevenMode)

•Go to *get_quota()* (ClusterMode)

get_snapmirror_status(**args*, ***kwargs*)
See sub classes.

•Go to *get_snapmirror_status()* (SevenMode)

•Go to *get_snapmirror_status()* (ClusterMode)

get_snapmirror_volume_status(**args*, ***kwargs*)
See sub classes.

•Go to *get_snapmirror_volume_status()* (SevenMode)

•Go to *get_snapmirror_volume_status()* (ClusterMode)

get_volumes(*filter_volume_names=[]*)
Return a list of snapable volumes of type *Volume*.

Parameters **filter_volume_names** (*list of str*) – consider only volumes that are in this list

Returns list of user home volumes

Return type list of *Volume*

Raises NidhoggException if an error occurs

get_volumes_with_quota_info(*filter_volume_names=[]*)
Return a list of snapable volumes of type *VolumeWithQuotaRatio*.

Parameters **filter_volume_names** (*list of str*) – consider only volumes that are in this list

Returns list of project home volumes

Return type list of *VolumeWithQuotaRatio*

Raises NidhoggException if an error occurs

has_forcegroup
Check if this cifs share feature is available.

Returns true, if feature is available

Return type bool

list_cifs_acls(**args*, ***kwargs*)
See sub classes.

- Go to *list_cifs_acls()* (SevenMode)
- Go to *list_cifs_acls()* (ClusterMode)

list_cifs_shares(**args*, ***kwargs*)
See sub classes.

- Go to *list_cifs_shares()* (SevenMode)
- Go to *list_cifs_shares()* (ClusterMode)

list_qtrees(**args*, ***kwargs*)
See sub classes.

- Go to *list_qtrees()* (SevenMode)
- Go to *list_qtrees()* (ClusterMode)

list_quotas(**args*, ***kwargs*)
See sub classes.

- Go to *list_quotas()* (SevenMode)
- Go to *list_quotas()* (ClusterMode)

list_snapable_volumes()
Return a list of *snapable* volumes.

That means, ignore volumes that are used as a snapmirror destination.

Returns list of snapable volumes

Return type list of *Volume*

Raises NidhoggException if an error occurs

list_snapshots(**args*, ***kwargs*)
See sub classes.

- Go to *list_snapshots()* (SevenMode)
- Go to *list_snapshots()* (ClusterMode)

list_volumes(**args*, ***kwargs*)
See sub classes.

- Go to *list_volumes()* (SevenMode)
- Go to *list_volumes()* (ClusterMode)

ontapi_version
ONTAPI version of the connected filer.

Returns ontapi version

Return type str

set_cifs_acl(**args*, ***kwargs*)
See sub classes.

- Go to *set_cifs_acl()* (SevenMode)
- Go to *set_cifs_acl()* (ClusterMode)

set_quota(**args*, ***kwargs*)
See sub classes.

- Go to *set_quota()* (SevenMode)
- Go to *set_quota()* (ClusterMode)

update_snapmirror(**args*, ***kwargs*)
See sub classes.

- Go to *update_snapmirror()* (SevenMode)
- Go to *update_snapmirror()* (ClusterMode)

update_snapmirror_with_snapshot(**args*, ***kwargs*)
See sub classes.

- Go to *update_snapmirror_with_snapshot()* (SevenMode)
- Go to *update_snapmirror_with_snapshot()* (ClusterMode)

volume_info(**args*, ***kwargs*)
See sub classes.

- Go to *volume_info()* (SevenMode)
- Go to *volume_info()* (ClusterMode)

vserver
Hostname of the connected filer.

Returns hostname

Return type str

vserver_fqdn
FQDN of the connected filer.

Returns FQDN of the filer

Return type str

exception nidhogg.core.**NidhoggException**
Bases: exceptions.Exception

Exception wrapper.

CHAPTER 3

nidhogg seven-mode details

class nidhogg.sevenmode.**SevenMode**(*url*, *username*, *password*, *major*, *minor*, *http=<class 'nidhogg.http.NidhoggHttp'>*)

Bases: *nidhogg.core.Nidhogg*

This class implements seven-mode filer specific API calls.

ACL_CHANGE = u'Change'
ACL permission constant for write access

ACL_FULL_CONTROL = u'Full Control'
ACL permission constant for full control

ACL_NO_ACCESS = u'No Access'
ACL permission constant for denying access

ACL_PERMISSIONS = [u'Full Control', u'Read', u'Change', u'No Access']
list of all permission constants

ACL_READ = u'Read'
ACL permission constant for read access

create_cifs_share(*volume*, *qtree*, *share_name*, *group_name=None*, *comment=None*, *umask=u'007'*)
Create a cifs share.

Parameters

- **volume** (*str*) – name of the volume
- **qtree** (*str*) – name of the qtree
- **share_name** (*str*) – name of the share
- **group_name** (*str*) – force group if specified
- **comment** (*str*) – description of the share
- **umask** (*str*) – file permission umask

Raises NidhoggException if an error occurs

delete_cifs_acl(*share_name*, *user_or_group*, *is_group=False*)
Delete cifs ACL of the specified user or group.

Parameters

- **share_name** (*str*) – name of the share
- **user_or_group** (*str*) – name of a user or group

- **is_group** (*bool*) – if true, param *user_or_group* specifies a unix group name

Raises NidhoggException if an error occurs

delete_cifs_acls(*share_name*)
Remove all cifs permssions.

Parameters **share_name** (*str*) – name of the share

Raises NidhoggException if an error occurs

delete_quota(*volume*, *qtree*)
Delete the quota of the specified volume and qtree.

Parameters

- **volume** (*str*) – name of the volume
- **qtree** (*str*) – name of the qtree

Raises NidhoggException if an error occurs

get_quota(*volume*, *qtree*)
Return the quota of the specified qtree on the given volume.

Parameters

- **volume** (*str*) – name of the volume
- **qtree** (*str*) – name of the qtree

Returns quota

Return type *Quota*

Raises NidhoggException if an error occurs

get_snapmirror_status(*volume=None*, *qtree=None*)
Get status of snapmirror replication pairs. If no params are provided, return all snapmirror status pairs.

Parameters

- **volume** (*str*) – name of source or destination volume
- **qtree** (*str*) – name of source or destination qtree

Returns list of all snapmirror pair status

Return type list of *SnapmirrorStatus* or empty list

Raises NidhoggException if an error occurs

get_snapmirror_volume_status(*volume*)
Get status of a snapmirror volume.

Parameters **volume** (*str*) – name of volume

Return type *SnapmirrorVolumeStatus*

Raises NidhoggException if an error occurs

list_cifs_acls(*share_name*)
Return ACL of the specified share.

Parameters **share_name** (*str*) – name of the share

Returns list of ACEs (access control entries)

Return type *ACE* or empty list

Raises NidhoggException if an error occurs

list_cifs_shares()
List all cifs shares.

Returns list of cifs shares

Return type list of *CifsShare* or empty list

Raises NidhoggException if an error occurs

list_qtrees(*volume*)
Return a list of qtrees of type *QTree*.

Parameters **volume** (*str*) – name of the volume

Returns list of qtrees

Return type list of *QTree* or empty list

Raises NidhoggException if an error occurs

list_quotas(*volume*)
Return a list of quota reports of the specified volume.

Parameters **volume** (*str*) – name of the volume

Returns list of quota reports

Return type *QuotaReport* or empty list

Raises NidhoggException if an error occurs

list_snapshots(*target_name*, *target_type=u'volume'*)
Return list of snapshots for given volume.

Parameters

- **target_name** (*str*) – name of the volume
- **target_type** (*str*) – type of the volume

Returns list of snapshots

Return type list of *Snapshot* or empty list

Raises NidhoggException if an error occurs

list_volumes(**args*, ***kwds*)
Return a list of volumes of type *Volume*.

Returns list of volumes

Return type list of *Volume* or empty list

Raises NidhoggException if an error occurs

set_cifs_acl(*share_name*, *user=u'everyone'*, *right=u'Read'*, *set_group_rights=False*)
Set a single ACL for the specifed share.

Parameters

- **share_name** (*str*) – name of the share
- **user** (*str*) – name of a user or unix group (if set_group_rights = True)
- **right** (*str*) – right to be set, value must be one of *ACL_PERMISSIONS*
- **set_group_rights** (*bool*) – if true, *user* param specifies a unix group name

Raises

- **NidhoggException** – if an error occurs
- **NidhoggException** – if wrong right was set

set_quota(*volume*, *qtree*, *quota_in_mb=1024*, *wait_til_finished=True*)
Set a quota in MiB (default = 1GiB) for the specified volume and qtree.

Parameters

- **volume** (*str*) – name of the volume
- **qtree** (*str*) – name of the qtree
- **quota_in_mb** (*int*) – quota in MiB
- **wait_til_finished** (*bool*) – if false, do not wait for resize operation

Raises

- **NidhoggException** – if an error occurs
- **NidhoggException** – if resize did not finish in time and we were waiting for it
- **NidhoggException** – if quotas are not enabled

update_snapmirror(*destination_volume*, *destination_qtree=None*, *source_filer=None*, *source_volume=None*, *source_qtree=None*)
Trigger the snapmirror replication.

If source_filer, source_volume and source_qtree (source location) are not specified (default), then the source in */etc/snapmirror.conf* for the destination path must be present.

Parameters

- **destination_volume** (*str*) – name of snapmirror destination volume
- **destination_qtree** (*str*) – name of snapmirror destination qtree
- **source_filer** (*str*) – hostname of source filer
- **source_volume** (*str*) – name of snapmirror source volume
- **source_qtree** (*str*) – name of snapmirror source qtree

Raises

- **NidhoggException** – if an error occurs
- **NidhoggException** – if source params are incomplete
- **NidhoggException** – if qtree params are used, but incomplete

update_snapmirror_with_snapshot(*name*, *destination_volume*, *destination_qtree=None*, *source_filer=None*, *source_volume=None*, *source_qtree=None*)
Update the named snapshot to the snapmirror destination.

Use the specified snapshot name also for the snapshot to be created on the destination server if possible.

If source_filer, source_volume and source_qtree (source location) are not specified (default), then the source in */etc/snapmirror.conf* for the destination path must be present.

Parameters

- **name** (*str*) – name of the snapshot
- **destination_volume** (*str*) – name of snapmirror destination volume

- **destination_qtree** (*str*) – name of snapmirror destination qtree
- **source_filer** (*str*) – hostname of source filer
- **source_volume** (*str*) – name of snapmirror source volume
- **source_qtree** (*str*) – name of snapmirror source qtree

Raises

- **NidhoggException** – if an error occurs
- **NidhoggException** – if source params are incomplete
- **NidhoggException** – if qtree params are used, but incomplete
- **NidhoggException** – if source contains no new data
- **NidhoggException** – if destination is busy

volume_info(**args*, ***kwds*)

Return basic information about the volume.

Parameters **volume** (*str*) – name of the volume

Returns volume

Return type *Volume*

Raises NidhoggException if an error occurs

CHAPTER 4

nidhogg cluster-mode details

class nidhogg.clustermode.**ClusterMode**(*url*, *username*, *password*, *major*, *minor*, *http=<class 'nidhogg.http.NidhoggHttp'>*)

Bases: *nidhogg.core.Nidhogg*

This class implements cluster-mode filer specific API calls.

ACL_CHANGE = u'change'
ACL permission constant for write access

ACL_FULL_CONTROL = u'full_control'
ACL permission constant for full control

ACL_NO_ACCESS = u'no_access'
ACL permission constant for denying access

ACL_PERMISSIONS = [u'full_control', u'read', u'change', u'no_access']
list of all permission constants

ACL_READ = u'read'
ACL permission constant for read access

create_cifs_share(*volume*, *qtree*, *share_name*, *group_name=None*, *comment=None*, *umask=u'007'*)
Create a cifs share.

Parameters

- **volume** (*str*) – name of the volume
- **qtree** (*str*) – name of the qtree
- **share_name** (*str*) – name of the share
- **group_name** (*str*) – force group name if provided (supported by cluster-mode filers with ontapi >= 1.30)
- **comment** (*str*) – description of the share
- **umask** (*str*) – file permission umask

Raises NidhoggException if an error occurs

delete_cifs_acl(*share_name*, *user_or_group*, *is_group=None*)
Delete cifs ACL of the specified user or group.

Parameters

- **share_name** (*str*) – name of the share

- **user_or_group** (*str*) – name of a user or group
- **is_group** (*None*) – not used for cluster-mode filers, specified here to be compatible with seven-mode method signature

Raises NidhoggException if an error occurs

delete_cifs_acls(*share_name*)
Remove all cifs permssions.

Parameters **share_name** (*str*) – name of the share

Raises NidhoggException if an error occurs

delete_quota(*volume*, *qtree*)
Delete the quota of the specified volume and qtree.

Parameters

- **volume** (*str*) – name of the volume
- **qtree** (*str*) – name of the qtree

Raises NidhoggException if an error occurs

get_quota(*volume*, *qtree*, *max_records=65536*)
Return the quota of the specified qtree on the given volume.

Parameters

- **volume** (*str*) – name of the volume
- **qtree** (*str*) – name of the qtree
- **max_records** (*int*) – limit returned records

Returns quota

Return type *Quota* or empty dict

Raises NidhoggException if an error occurs

get_snapmirror_status(*volume=None*, *qtree=None*)
Get status the snapmirror replication.

get_snapmirror_volume_status(*volume*)
Get status the snapmirror replication.

list_cifs_acls(*share_name*, *max_records=65536*)
Return ACL of the specified share.

Parameters

- **share_name** (*str*) – name of the share
- **max_records** (*int*) – limit returned records

Returns list of ACEs (access control entries)

Return type *ACE* or empty list

Raises NidhoggException if an error occurs

list_cifs_shares()
List all cifs shares.

Returns list of cifs shares

Return type list of *CifsShare* or empty list

Raises NidhoggException if an error occurs

list_qtrees(*volume*, *max_records=65536*)
Return a list of qtrees of type *QTree*.

Parameters

- **volume** (*str*) – name of the volume
- **max_records** (*int*) – limit returned records

Returns list of qtrees

Return type list of *QTree* or empty list

Raises NidhoggException if an error occurs

list_quotas(*volume*, *max_records=65536*)
Return a list of quota reports of the specified volume.

Parameters

- **volume** (*str*) – name of the volume
- **max_records** (*int*) – limit returned records

Returns list of quota reports

Return type *QuotaReport* or empty list

Raises NidhoggException if an error occurs

list_snapshots(*target_name*, *max_records=65536*)
Return list of snapshots for given volume.

Parameters

- **target_name** (*str*) – name of the volume
- **max_records** (*int*) – limit returned records

Returns list of snapshots

Return type list of *Snapshot* or empty list

Raises NidhoggException if an error occurs

list_volumes(**args*, ***kwds*)
Return a list of volumes of type *Volume*.

Parameters **max_records** (*int*) – limit returned records

Returns list of volumes

Return type list of *Volume* or empty list

Raises NidhoggException if an error occurs

set_cifs_acl(*share_name*, *user=u'everyone'*, *right=u'read'*, *set_group_rights=None*)
Set a single ACL for the specifed share.

Parameters

- **share_name** (*str*) – name of the share
- **user** (*str*) – name of a user or group
- **right** (*str*) – right to be set, value must be one of *ACL_PERMISSIONS*

- **set_group_rights** (*bool*) – if true, *user* param specifies a unix group name; if false, *user* param specifies a unix user name; if not defined, *user* param specifies a windows name

Raises

- **NidhoggException** – if an error occurs
- **NidhoggException** – if wrong right was set

set_quota(*volume*, *qtree*, *quota_in_mb=1024*, *wait_til_finished=True*)
Set a quota in MiB (default = 1GiB) for the specified volume and qtree.

Parameters

- **volume** (*str*) – name of the volume
- **qtree** (*str*) – name of the qtree
- **quota_in_mb** (*int*) – quota in MiB
- **wait_til_finished** (*bool*) – if false, do not wait for resize operation

Raises

- **NidhoggException** – if an error occurs
- **NidhoggException** – if resize did not finish in time and we were waiting for it
- **NidhoggException** – if quotas are not enabled

update_snapmirror(*destination_volume*, *destination_qtree=None*)
Trigger the snapmirror replication.

update_snapmirror_with_snapshot(*name*, *destination_volume*, *destination_qtree=None*)
Update the named snapshot to the snapmirror destination.

volume_info(**args*, ***kwds*)
Return basic information about the volume.

Parameters **volume** (*str*) – name of the volume

Returns volume

Return type *Volume*

Raises NidhoggException if an error occurs

nidhogg.clustermode.**MAX_RECORDS = 65536**
maximum records that can be retrieved via NETAPP API

CHAPTER 5

nidhogg data types

class nidhogg.compatible.**ACE**(***kwargs*)
Bases: *nidhogg.compatible.InitDict*

Data object representing an *access control entry*.

required_arguments = [u'share_name', u'permission', u'user_or_group', u'is_group', u'user_group_type']

class nidhogg.compatible.**Aggregate**(***kwargs*)
Bases: *nidhogg.compatible.InitDict*

Data object representing an aggregate.

required_arguments = [u'id', u'volume']

class nidhogg.compatible.**CifsShare**(***kwargs*)
Bases: *nidhogg.compatible.InitDict*

Data object representing a cifs share.

required_arguments = [u'path', u'share_name']

class nidhogg.compatible.**InitDict**(***kwargs*)
Bases: dict

Base class of the data object classes to enforce required keys in the dict.

class nidhogg.compatible.**QTree**(***kwargs*)
Bases: *nidhogg.compatible.InitDict*

Data object representing a qtree.

required_arguments = [u'qtree', u'status', u'security_style']

class nidhogg.compatible.**Quota**(***kwargs*)
Bases: *nidhogg.compatible.InitDict*

Data object representing a quota.

required_arguments = [u'disk_limit', u'file_limit', u'threshold', u'soft_disk_limit', u'soft_file_limit']

class nidhogg.compatible.**QuotaReport**(***kwargs*)
Bases: *nidhogg.compatible.InitDict*

Data object representing a quota report.

required_arguments = [u'disk_limit', u'file_limit', u'threshold', u'soft_disk_limit', u'soft_file_limit', u'quota_target'

class nidhogg.compatible.**SnapmirrorStatus**(***kwargs*)
Bases: *nidhogg.compatible.InitDict*

Data object representing a snapmirror status.

required_arguments = [u'source_location', u'destination_location', u'last_transfer_from', u'last_transfer_size', u'las

class nidhogg.compatible.**SnapmirrorVolumeStatus**(***kwargs*)
Bases: *nidhogg.compatible.InitDict*

Data object representing a snapmirror volume status.

required_arguments = [u'is_source', u'is_destination', u'is_transfer_in_progress', u'is_transfer_broken']

class nidhogg.compatible.**Snapshot**(***kwargs*)
Bases: *nidhogg.compatible.InitDict*

Data object representing a snapshot.

required_arguments = [u'name']

class nidhogg.compatible.**Volume**(***kwargs*)
Bases: *nidhogg.compatible.InitDict*

Data object representing a volume sortable by free size.

required_arguments = [u'name', u'state', u'size_total', u'size_used', u'size_available', u'files_used', u'files_total', u's

class nidhogg.compatible.**VolumeWithQuotaRatio**(***kwargs*)
Bases: *nidhogg.compatible.InitDict*

Data object representing a volume sortable by quota ratio.

required_arguments = [u'name', u'state', u'size_total', u'size_used', u'size_available', u'files_used', u'files_total', u's

CHAPTER 6

nidhogg helpers

6.1 nidhogg.http module

class nidhogg.http.**NidhoggHttp**(*url*, *username*, *password*)

Bases: object

Requests the Netapp API und converts the response into a dictionary.

invoke_request(*req*)

Request the Netapp API.

Parameters **req** (*dict*) – dictionary of request params

Returns Netapp API response

Return type str

parse_xml_reply(*xmlresponse*)

Convert XML reply into a dictionary.

Parameters **xmlresponse** (*str*) – Response from Netapp API.

Returns response

Return type dict

6.2 nidhogg.utils module

nidhogg.utils.**safe_get**(*d*, *key*)

Helper function.

Parameters

- **d** (*dict*) – dictionary
- **key** (*str*) – key to retrieve from dict

Returns value of specified key if not None, otherwise empty dict

nidhogg.utils.**underline_to_dash**(*d*)

Helper function to replace "_" to "-" in keys of specifed dictionary recursively.

Netapp API uses "-" in XML parameters.

Parameters **d** (*dict*) – dictionary of dictionaries or lists

Returns new dictionary

Return type dict

CHAPTER 7

CHANGELOG

7.1 v3.3

New *list_cifs_shares* method.

- See `list_cifs_shares()` (SevenMode)
- See `list_cifs_shares()` (ClusterMode)

PEP8 fixes

7.2 v3.2

setup.py fixes

7.3 v3.1

Travis testing

setup.py fixes

7.4 v3.00

First public release.

7.5 v2.14

Method `update_snapmirror()` (SevenMode) changed. Method `update_snapmirror_with_snapshot()` (SevenMode) changed. Param *source_filer*, *source_volume* and *source_qtree* introduced.

Update a qtree on a snapmirror destination. Connect to the destination filer, specify destination volume and qtree, source filer, volume and qtree and invoke command.

Attention: If *source_filer*, *source_volume* and *source_qtree* (source location) are not specified (default), then the source in */etc/snapmirror.conf* for the destination path must be present.

Example:

```
import nidhogg
dst = nidhogg.get_netapp("filer13.example.com", "<username>", "<password>")
dst.update_snapmirror_with_snapshot(
    name="userdir"
    destination_volume="sm_filer47_nidhoggtest",
    destination_qtree="nidhoggtest",
    source_filer="filer47.example.com",
    source_volume="nidhoggtest",
    source_qtree="nidhoggtest"
)
```

Method *get_snapmirror_volume_status()* (SevenMode) introduced. Get details about snapmirror status of the specified volume.

Example:

```
import nidhogg
dst = nidhogg.get_netapp("filer13.example.com", "<username>", "<password>")
dst.get_snapmirror_volume_status("sm_filer48_userhome_LCP")
>> {'is_source': False, 'is_destination': True, 'is_transfer_broken': False, 'is_transfer_in_pro
```

Waiting time for the quota resize operation to finish increased to 2 minutes.

- See *set_quota()* (SevenMode)
- See *set_quota()* (ClusterMode)

7.6 v2.13

Method *update_snapmirror_with_snapshot()* (SevenMode) introduced. Trigger the snapmirror replication using the named snapshot. Connect to the destination filer, specify snapshot name and destination volume and invoke command.

Example:

```
import nidhogg
filer = nidhogg.get_netapp("filer99.example.com", "<username>", "<password>")
filer.update_snapmirror_with_snapshot("nightly.1", "sq_filer99_test001", "smtest")
```

7.7 v2.12

Method *get_snapmirror_status()* (SevenMode) introduced. Check the status of snapmirror relations. Connect to the destination filer, specify volume of source or destination (optional) and qtree of source or destination (optional) and invoke command.

Example:

```
import nidhogg
filer = nidhogg.get_netapp("filer99.example.com", "<username>", "<password>")
# return status of all snapmirror relations
status_list = filer.get_snapmirror_status()
# return status of snapmirror relations of specified volume
status_list = filer.get_snapmirror_status("sq_filer99_test001")
# return status of snapmirror relations of specified volume and qtree
status_list = filer.get_snapmirror_status("sq_filer99_test001", "smtest")
```

7.8 v2.11

Method *update_snapmirror()* (SevenMode) introduced. Trigger the snapmirror replication. Connect to the destination filer, specify destination volume and qtree (optional) and invoke command.

Example:

```
import nidhogg
filer = nidhogg.get_netapp("filer99.example.com", "<username>", "<password>")
filer.update_snapmirror("sq_filer99_test001", "smtest")
```

7.9 v2.8

Param *local_volumes_only* removed from *list_volumes* (ClusterMode).

This 'feature' removed all volumes where the *owning_vserver != hostname* (hostname is derived from the connection string). So, if you connected to the filer via DNS alias, no volumes were found.

Originally it was used to filter volumes when connecting to a filer cluster. Not used in production mode.

- See *list_volumes()* (ClusterMode)

7.10 v2.7

Method *create_cifs_share()* (ClusterMode) now also uses param *group_name*. Cluster-mode filers with ONTAPI 1.3 supports "force group name".

Method *set_cifs_acl()* (ClusterMode) now sets also the correct *user-group-type* for the specified user or group:

- if param *set_group_rights* is True, *user-group-type* is "unix_group"
- if param *set_group_rights* is False, *user-group-type* is "unix_user"
- if param *set_group_rights* is None, *user-group-type* is "windows"

7.11 v2.6

Param *user_name* removed from *create_cifs_share*. Had no effect.

- See *create_cifs_share()* (SevenMode)
- See *create_cifs_share()* (ClusterMode)

CHAPTER 8

Purpose

This library is a wrapper interface to **Netapp filers** accross **versions and technologies** (sevenmode, vserver) using the native Netapp REST API. It provides a consistent interface for the most common operations.

CHAPTER 9

Installation

```
pip install nidhogg
```

CHAPTER 10

Usage

```
import nidhogg
filer = nidhogg.get_netapp("filer99.example.com", "<username>", "<password>")
filer.create_qtree("volume_name", "qtree_name")
```

CHAPTER 11

Planned further work

- support snapmirror methods for Netapp vserver mode
- add EMC Isilon wrapper

CHAPTER 12

Indices and tables

- genindex
- modindex

n

A

C

D

E

G

H

I

L

M

N

O

P

Q

R

S

U

V