Nutanix REST Client Documentation

Release 1.5.0

Ross Davies

Jun 10, 2021

Contents

CHAPTER 1

Installation Guide

The Nutanix API Client is available through the Python Package Index.

The code is available on gitlab at https://gitlab.com/nutanix-se/python/ntnx-api-library and can optionally be built and installed from source.

1.1 Python Package Index Installation

```
$ pip install ntnx_api
```

Or

```
$ easy_install ntnx_api
```

1.2 Source Code Installation

```
$ mkdir nutanix
$ cd nutanix
$ git clone https://gitlab.com/nutanix-se/python/ntnx-api-library
$ cd ntnx-api-library
$ python setup.py install
```

Or to build HTML documentation from source:

```
$ cd docs/
$ make html
```

This creates a _build/ directory under docs/.

CHAPTER 2

Quick Start Guide

This guide is intended to give users a basic idea of REST Client usage through examples.

2.1 Before You Begin

You should already have the Nutanix Prism REST Client installed.

This includes installing REST Client package dependencies.

See *Installation Guide* for more information.

2.2 Starting A Session

To verify the REST Client package is installed and importable, try executing the following in a Python interpreter:

```
>>> from ntnx_api.client import ApiClient
>>> ntnx_api = ApiClient(
>>>   connection_type = 'pe',
>>>   ip_address='1.1.1.1'
>>>   username='admin'
>>>   password='nutanix/4u'
>>> )
```

If that succeeds without an ImportError, you are ready to start using the client.

```
>>> from ntnx_api import prism
>>> ntnx_cluster = prism.Cluster(api_client=ntnx_api)
>>> clusters = ntnx_clusters.get_clusters()
>>> print(len(clusters))
```

CHAPTER 3

Contributing Guide

Create a private fork of the project

On master branch, increment version number

Set __version__ = '1.5.0' in ntnx_api/__version.py

Update docs/changelog.rst with new version & changes

Make changes to the ntnx_api code

Install local developer requirements

pip install -r test-requirements.txt

Run flake8 locally to verify package

```
$ tox -e flake8
```

Run unit test locally to verify package. You may have to update the IP addresses in the test included in ntnx_api/test/ with an ip address of your local nutanix cluster / prism central.

```
$ tox -e py3
```

Push changes

```
$ git push origin master
```

Create tag for version

```
$ git tag 1.5.0 -m "Release v1.5.0"
```

Push tag

```
$ git push origin 1.5.0
```

Submit merge request back to develop branch of main project

CHAPTER 4

API Glossary

Contents:

4.1 ntnx_api.client

4.1.1 ApiClient

class ntnx_api.client.**PrismApi**(**args*, ***kwargs*)

A class to represent a connection to an Nutanix API on either Prism Element or Prism Central

param ip_address IPv4 address or resolvable DNS entry for Nutanix API host

type ip_address str

param username Username to authenticate to API (default='admin')

type username str, optional

param password Password for user to authenticate to API (default='nutanix/4u')

type password str, optional

param port The port where the Nutanix API listens (default='9440')

type port str, optional

param validate_certs The port where the Nutanix API listens (default='9440')

type validate_certs bool, optional

raises `ValueError` - If the connection type is not set - If the connection type is not valid

raises *NutanixError* - If the API is not reachable

New in version 1.1.0: This class supersedes the originally released class *ApiClient*. To interact with the Prism Element or Prism Central APIs please use this class instead of the deprecated class.

request(*uri*, *api_version=None*, *payload=None*, *params=None*, *response_code=None*, *timeout=120*, *method=None*)
Perform HTTP request for REST API.

Parameters

- **uri** (*str*) – URI of resource to interact with
- **api_version** (*str('v1', 'v2.0', 'v3'), optional*) – Version of API to use. Affects the base of the called URI.
- **payload** (*dict, optional*) – Data to be used in the body of request
- **params** (*dict, optional*) – Data to be used in the query string of request
- **response_code** (*int, optional*) – Expected response code (default=200)
- **timeout** (*int, optional*) – Number of seconds to wait before the API call times out (default=120)
- **method** (*str('GET', 'POST', 'PUT', 'DELETE'), optional*) – Method for the API request. If payload=None default=GET, if payload!=None default=POST

Returns Result of API query

Return type ResponseDict or ResponseList

setup()
Connect to rest API and store variables neccessary for optimal operation of the class.

Returns *True* if successful;, *False* otherwise

Return type bool

test()
Test that a query can be performed against the defined API connection

Returns *True* if connected, *False* otherwise

Return type bool

upload(*uri*, *file_path*, *header_dict=None*, *api_version=None*, *params=None*, *response_code=200*, *timeout=120*, *method='POST'*)
Perform HTTP upload request for REST API.

Parameters

- **uri** (*str*) – URI of resource to interact with
- **file_path** (*str*) – PAth to the file to be uploaded
- **header_dict** (*dict, optional*) – Dict of additional headers to add to this request.
- **api_version** (*str('v1', 'v2.0', 'v3'), optional*) – Version of API to use. Affects the base of the called URI.
- **params** (*dict, optional*) – Data to be used in the query string of request
- **response_code** (*int, optional*) – Expected response code (default=200)
- **timeout** (*int, optional*) – Number of seconds to wait before the API call times out (default=120)
- **method** (*str('GET', 'POST', 'PUT', 'DELETE'), optional*) – Method for the API request. If payload=None default=GET, if payload!=None default=POST

Returns Result of API query

Return type ResponseDict or ResponseList

class ntnx_api.client.**ApiClient**(*connection_type*, *ip_address=None*, *username='admin'*, *password='nutanix/4u'*, *port='9440'*, *validate_certs=False*, *environment=None*, *api_key=None*)

A class to represent a connection to an Nutanix API

param connection_type Identifier for the type of Nutanix API to connect to

type connection_type str('pe','pc','era','karbon','kps','frame','foundation')

param ip_address IPv4 address or resolvable DNS entry for Nutanix API host

type ip_address str

param username Username to authenticate to API (default='admin')

type username str, optional

param password Password for user to authenticate to API (default='nutanix/4u')

type password str, optional

param port The port where the Nutanix API listens (default='9440')

type port str, optional

param validate_certs The port where the Nutanix API listens (default='9440')

type validate_certs bool, optional

param environment Environment string for Xi Frame

type environment str, optional

param api_key API Key string for Karbon Platform Services (KPS)

type api_key str, optional

raises `ValueError` - If the connection type is not set - If the connection type is not valid

raises *NutanixError* - If the API is not reachable

Deprecated since version 1.1.0: This ApiClient class is being deprecated in favor of separate classes for teach Nutanix endpoint type being connected with. This will simplify future development and use and allow for more granular changes based on the requirements of the endpoints API. Prism Element or Central API connection management has moved to *PrismApi*

request(*uri*, *api_version=None*, *payload=None*, *params=None*, *response_code=200*, *timeout=120*, *method=None*)

Perform HTTP request for REST API.

Parameters

- **uri** (*str*) – URI of resource to interact with
- **api_version** (*str('v1', 'v2.0', 'v3'), optional*) – Version of API to use. Affects the base of the called URI.
- **payload** (*dict, optional*) – Data to be used in the body of request
- **params** (*dict, optional*) – Data to be used in the query string of request
- **response_code** (*int, optional*) – Expected response code (default=200)
- **timeout** (*int, optional*) – Number of seconds to wait before the API call times out (default=120)

- **method** (*str('GET', 'POST', 'PUT', 'DELETE'), optional*) – Method for the API request. If payload=None default=GET, if payload!=None default=POST

Returns Result of API query

Return type ResponseDict or ResponseList

test()
Test that a query can be performed against the defined API connection

Returns *True* if connected, *False* otherwise

Return type bool

class ntnx_api.client.**NutanixError**(*reason*)
Exception type raised by Nutanix API.

Parameters **reason** (*str*) – A message describing why the error occurred.

Variables **reason** (*str*) – A message describing why the error occurred.

class ntnx_api.client.**NutanixRestHTTPError**(*target*, *rest_version*, *data*, *params*, *response*)
Exception raised as a result of non-200 response status code.

Parameters

- **target** (*str*) – IP or DNS name of the array that received the HTTP request.
- **rest_version** (*str*) – The REST API version that was used when making the request.
- **data** (*str*) – The REST API data that was used when making the request.
- **params** (*str*) – The REST API parameters that was used when making the request.
- **response** (requests.Response) – The response of the HTTP request that caused the error.

Variables

- **target** (*str*) – IP or DNS name of the array that received the HTTP request.
- **rest_version** (*str*) – The REST API version that was used when making the request.
- **code** (*int*) – The HTTP response status code of the request.
- **headers** (*dict*) – A dictionary containing the header information. Keys are case-insensitive.
- **reason** (*str*) – The textual reason for the HTTP status code (e.g. "BAD REQUEST").
- **text** (*str*) – The body of the response which may contain a message explaining the error.

Note: The error message in text is not guaranteed to be consistent across REST versions, and thus should not be programmed against.

4.2 ntnx_api.prism

4.2.1 Config

class ntnx_api.prism.**Config**(*api_client*)
A class to represent the configuration of the Nutanix Prism Instance

Parameters **api_client** (ntnx.client.ApiClient) – Initialized API client class

accept_elua(*name*, *title*, *company*, *clusteruuid=None*)
Accept the Nutanix ELUA

Parameters

- **name** (*str*) – Your name
- **title** (*str*) – Your job title
- **company** (*str*) – Name of your company
- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the ntnx.client.ApiClient *connection_type* is set to *pc*.

add_auth_dir(*name*, *directory_url*, *domain*, *username*, *password*, *recursive=False*, *directory_type='LDAP'*, *connection_type='LDAP'*, *clusteruuid=None*)
Add authentication directory for a specific cluster

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the ntnx.client.ApiClient *connection_type* is set to *pc*.
- **name** (*str, optional*) – Directory name
- **directory_url** (*str*) – ldap/ldaps URL to connect to the domain including the port your LDAP target is listening on. eg ldap://192.168.1.10:384
- **domain** (*str*) – Fully qualified name of the domain. eg nutanix.local
- **username** (*str*) – Username to authenticate to the domain
- **password** (*str*) – Password for user to authenticate to the domain
- **recursive** (*bool, optional*) – Whether to search for nested groups
- **directory_type** (*str('ACTIVE_DIRECTORY', 'OPEN_LDAP'), optional*) – Type of directory
- **connection_type** (*str('LDAP'), optional*) – Type of connection

add_auth_dir_role_mapping(*directory*, *directory_entities*, *directory_entity_type*, *cluster_admin=False*, *user_admin=False*, *clusteruuid=None*)
Add authentication role mapping for a named authentication directory on a specific cluster. If either *cluster_admin* or *user_admin* is not set the role granted to this user is *ROLE_CLUSTER_VIEWER*.

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the ntnx.client.ApiClient *connection_type* is set to *pc*.
- **directory** (*str*) – Name of directory.
- **directory_entities** (*list of str*) – List of users/groups to add.
- **directory_entity_type** (*str('USER', 'GROUP')*) – Type of directory entity being added.
- **cluster_admin** (*bool, optional*) – Whether to grant user *Cluster Admin* privilege
- **user_admin** (*bool, optional*) – Whether to grant user *User Admin* privilege

add_dns(*dns_server*, *clusteruuid=None*)
Add dns server to a specific cluster

Parameters **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

add_local_user(*username*, *password*, *firstname*, *lastname*, *email*, *enabled=True*, *cluster_admin=False*, *user_admin=False*, *language='en-US'*, *clusteruuid=None*)
Add local user on a specific cluster. User is added with cluster viewer rights. Additional rights have to be added after the user is created

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.
- **username** (*str*) – Username
- **password** (*str*) – Password
- **firstname** (*str, optional*) – First name of user
- **lastname** (*str, optional*) – Last name of user
- **email** (*str, optional*) – Email address for user
- **cluster_admin** (*bool, optional*) – Whether to grant user *Cluster Admin* privilege
- **user_admin** (*bool, optional*) – Whether to grant user *User Admin* privilege
- **language** (*str('en-US','zh-CN','ja-JP'), optional*) – Localization region for user account *(default='en-US')*
- **enabled** (*bool, optional*) – Enable user account *(default=True)*

add_ntp(*ntp_server*, *clusteruuid=None*)
Add ntp server to a specific cluster

Parameters **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

add_proxy(*name*, *address*, *port*, *proxy_types*, *username=None*, *password=None*, *clusteruuid=None*)
Add proxy configuration for a specific cluster

Parameters **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

change_ui_admin_password(*admin_password*, *ssh_user='nutanix'*, *ssh_password='nutanix/4u'*, *clusteruuid=None*)
Change the password for the 'admin' UI user account. This is not exposed via the API so paramiko is used to establish an ssh session to the CVM. If ssh is blocked or key-based authentication is enabled (cluster-lockdown) then this will not work.

Parameters

- **admin_password** (*str*) – The new admin password to be set for the prism admin user account. See https://portal.nutanix.com for password complexity requirements.
- **ssh_user** – The user with ssh access to the nutanix cluster (default='nutanix')

- **ssh_password** (*str, optional*) – The password for the user with ssh access to the nutanix cluster (default='nutanix/4u')
- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

get_alert_config(*clusteruuid=None*)
Get alert configuration for specified cluster

Parameters **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

get_auth_config(*clusteruuid=None*)
Retrieve authentication data for a specific cluster

Parameters **clusteruuid** (*str, optional :returns: A list of dictionaries describing the authentication configuration of the cluster.*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns A list of authentication config.

Return type ResponseList

get_auth_dir_role_mappings(*clusteruuid=None*)
Get all authentication role mappings for a specific cluster

Parameters **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

get_auth_dirs(*clusteruuid=None*)
Get authentication directories for a specific cluster

Parameters **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

get_auth_types(*clusteruuid=None*)
Get authentication types for a specific cluster

Parameters **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

get_categories()
Retrieve data for all categories.

Note: Will only return data when *connection_type=='pc'*

Deprecated since version 1.5.0: The *Config* is being reorganized and this function has been moved ot a class dedicated to the management of Categories *Categories*.

get_category_key_usage(*category*, *key*)
Retrieve data for all vms or hosts belonging to a specific category & key.

parameter category Category name

type category str

parameter key Key name

type key str

Note: Will only return data when *connection_type=='pc'*

Deprecated since version 1.5.0: The `Config` is being reorganized and this function has been moved ot a class dedicated to the management of Categories `Categories`.

get_category_keys(*category*)
Retrieve data for all keys belonging to a specific category.

param category Category name

type category str

Note: Will only return data when *connection_type=='pc'*

Deprecated since version 1.5.0: The `Config` is being reorganized and this function has been moved ot a class dedicated to the management of Categories `Categories`.

get_dns(*clusteruuid=None*)
Retrieve dns servers configured for a specific cluster

Parameters **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns A list of the clusters dns servers

Return type ResponseList

get_local_users(*clusteruuid=None*)
Get local users on a specific cluster

Parameters **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

get_ntp(*clusteruuid=None*)
Retrieve ntp servers configured for a specific cluster

Parameters **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns A list of the clusters ntp servers

Return type ResponseList

get_project_usage(*project_name*)
Retrieve vms that belong to a specific project.

param project_name Project name

type project_name str

Note: Will only return data when *connection_type=='pc'*

Deprecated since version 1.5.0: The *Config* is being reorganized and this function has been moved ot a class dedicated to the management of Projects *Projects*.

get_projects()
Retrieve data for all projects.

Note: Will only return data when *connection_type=='pc'*

Deprecated since version 1.5.0: The *Config* is being reorganized and this function has been moved ot a class dedicated to the management of Projects *Projects*.

get_protection_rules()
Retrieve data for all protection rules.

Note: Will only return data when *connection_type=='pc'*

get_proxy(*clusteruuid=None*)
Retrieve proxy configured for a specific cluster

Parameters **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns A list of the clusters dns servers

Return type ResponseList

get_pulse(*clusteruuid=None*)
Get pulse config for a specific cluster

Parameters **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns A dictionary describing pulse configuration from the specified cluster.

Return type ResponseList

get_smtp(*clusteruuid=None*)
Get smtp config for a specific cluster

Parameters **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns A dictionary describing smtp configuration from the specified cluster.

Return type ResponseList

get_ui_2048_game(*clusteruuid=None*)
Get UI 2048 game status for a specific cluster

Parameters **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns A dict with the defined UI 2048 game setting *{'status': 'true'}* or None

Return type ResponseDict

get_ui_animation(*clusteruuid=None*)
Get UI animated background particles status for a specific cluster

Parameters **clusteruuid**(*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns A dict with the defined UI particle animation setting *{'status': 'true'}* or None

Return type ResponseDict

get_ui_banner(*clusteruuid=None*)
Get UI text (title/blurb) for a specific cluster

Parameters **clusteruuid**(*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns A dict with the defined UI banner *{'status': 'true', 'content':'blah blah'}* or None

Return type ResponseDict

get_ui_color(*clusteruuid=None*)
Get UI color 1 and color 2 for a specific cluster

Parameters **clusteruuid**(*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns A dict with the defined UI colors *{'color1': '#CC6164', 'color2':'#FFD055'}* or None

Return type ResponseDict

get_ui_config(*clusteruuid=None*)
Get the configuration data for a clusters Prism Element user interface

Parameters **clusteruuid**(*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

get_ui_text(*clusteruuid=None*)
Get UI text (title/blurb) for a specific cluster

Parameters **clusteruuid**(*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns A dict with the defined UI text *{'title': 'blah', 'blurb':'blah blah'}* or None

Return type ResponseDict

remove_alert_config(*clusteruuid=None*)
Reset alert configuration for specified cluster

Parameters **clusteruuid**(*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

remove_auth_dir(*name*, *clusteruuid=None*)
Remove authentication directory for a specific cluster

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.
- **name** (*str, optional*) – Directory name

remove_auth_dir_role_mapping(*directory*, *directory_entities*, *directory_entity_type*, *cluster_admin=False*, *user_admin=False*, *clusteruuid=None*)
Delete authentication role mapping for a named authentication directory on a specific cluster

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.
- **directory** (*str*) – Name of directory.
- **directory_entities** (*list of str*) – List of users/groups to add.
- **directory_entity_type** (*str('USER', 'GROUP')*) – Type of directory entity being added.
- **cluster_admin** (*bool, optional*) – Whether to grant user *Cluster Admin* privilege
- **user_admin** (*bool, optional*) – Whether to grant user *User Admin* privilege

remove_dns(*dns_server*, *clusteruuid=None*)
Remove dns server from a specific cluster

Parameters **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

remove_local_user(*username*, *clusteruuid=None*)
Remove local user on a specific cluster

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.
- **username** (*str*) – Username

remove_ntp(*ntp_server*, *clusteruuid=None*)
Remove ntp server from a specific cluster

Parameters **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

remove_proxy(*name*, *clusteruuid=None*)
Remove proxy configuration from a specific cluster

Parameters **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

remove_smtp(*clusteruuid=None*)
Remove smtp config for a specific cluster

Parameters **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

set_auth_dir(*name*, *directory_type*, *directory_url*, *domain*, *username*, *password*, *recursive=False*, *connection_type='LDAP'*, *force=False*, *clusteruuid=None*)

Set authentication directory for a specific cluster

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.
- **name** (*str, optional*) – Directory name
- **directory_url** (*str*) – ldap/ldaps URL to connect to the domain including the port your LDAP target is listening on. eg ldap://192.168.1.10:384
- **domain** (*str*) – Fully qualified name of the domain. eg nutanix.local
- **username** (*str*) – Username to authenticate to the domain
- **password** (*str*) – Password for user to authenticate to the domain
- **recursive** (*bool, optional*) – Whether to search for nested groups *(default: False)*
- **directory_type** (*str('ACTIVE_DIRECTORY', 'OPEN_LDAP'), optional*) – Type of directory
- **connection_type** (*str('LDAP'), optional*) – Type of connection *(default: 'LDAP')*
- **force** (*bool, optional*) – Force directory update. Use this to update the password of the auth domain user.

Returns *updated* if changed, *added* if created or None otherwise

Return type str

set_auth_dir_role_mapping(*directory*, *directory_entities*, *directory_entity_type*, *cluster_admin=False*, *user_admin=False*, *clusteruuid=None*)

Create or update authentication role mapping for a named authentication directory on a specific cluster.

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.
- **directory** (*str*) – Name of directory.
- **directory_entities** (*list of str*) – List of users/groups to add.
- **directory_entity_type** (*str('USER', 'GROUP')*) – Type of directory entity being added.
- **cluster_admin** (*bool, optional*) – Whether to grant user *Cluster Admin* privilege
- **user_admin** (*bool, optional*) – Whether to grant user *User Admin* privilege

Returns *updated* if changed, *added* if created or None otherwise

Return type str

set_dns(*clusteruuid=None*, *dns_servers=None*)
Set dns servers for a specific cluster

Parameters **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns *updated* if changed, *added* if created or None otherwise

Return type str

set_local_user(*username*, *password*, *firstname*, *lastname*, *email*, *enabled=True*, *cluster_admin=False*, *user_admin=False*, *language='en-US'*, *clusteruuid=None*)
Create or update local user on a specific cluster. User is added with cluster viewer rights. Additional rights have to be added after the user is created

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.
- **username** (*str*) – Username
- **password** (*str*) – Password
- **firstname** (*str, optional*) – First name of user
- **lastname** (*str, optional*) – Last name of user
- **email** (*str, optional*) – Email address for user
- **cluster_admin** (*bool, optional*) – Email address for user
- **user_admin** (*bool, optional*) – Whether to grant user *User Admin* priviliges
- **language** (*str('en-US','zh-CN','ja-JP'), optional*) – Localization region for user account *(default='en-US')*
- **enabled** (*bool, optional*) – Enable user account *(default=True)*

Returns *updated* if changed, *added* if created or None otherwise

Return type str

set_ntp(*clusteruuid=None*, *ntp_servers=None*)
Set ntp servers for a specific cluster

Parameters **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns *updated* if changed, *added* if created or None otherwise

Return type str

set_proxy(*address*, *port*, *clusteruuid=None*, *name='proxy'*, *username=''*, *password=''*, *http=True*, *https=False*, *socks=False*)
Set proxy configuration for a specific cluster

Parameters **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns *updated* if changed, *added* if created or None otherwise

Return type str

set_pulse(*enable*, *email_address_list=None*, *email_nutanix=False*, *clusteruuid=None*)
Set UI animated background particles status for a specific cluster

Parameters **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns *updated* if changed, *added* if created or None otherwise

Return type str

set_smtp(*address*, *port*, *mode=None*, *from_email_address='do-not-reply@nutanix.cluster'*, *username=None*, *password=None*, *force=False*, *clusteruuid=None*)
Set smtp config for a specific cluster

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.
- **address** (*str*) – SMTP server IP address or FQDN
- **port** (*int*) – SMTP server port
- **mode** (*str('tls', 'ssl', None), optional*) – SMTP connection mode
- **from_email_address** (*str, optional*) – Email address to send alerts from *(default: do-not-reply@nutanix.cluster)*
- **username** (*str, optional*) – Username to authenticate to the SMTP server
- **password** (*str, optional*) – Password for user to authenticate to the SMTP server
- **force** (*bool, optional*) – Force update regardless of differences *(default=False)*

Returns *updated* if changed, *added* if created or None otherwise

Return type str

set_ui_2048_game(*status*, *clusteruuid=None*)
Set UI 2048 game status for a specific cluster

Parameters **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns *updated* if changed, *added* if created or None otherwise

Return type str

set_ui_animation(*status*, *clusteruuid=None*)
Set UI animated background particles status for a specific cluster

Parameters **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns *updated* if changed, *added* if created or None otherwise

Return type str

set_ui_banner(*status*, *content*, *clusteruuid=None*)
Set UI welcome banner (title/blurb) for a specific cluster

Parameters **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the ntnx.client.ApiClient *connection_type* is set to *pc*.

Returns *updated* if changed, *added* if created or None otherwise

Return type str

set_ui_color(*color1*, *color2*, *clusteruuid=None*)
Set UI color 1 and color 2 for a specific cluster

Parameters **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the ntnx.client.ApiClient *connection_type* is set to *pc*.

Returns *updated* if changed, *added* if created or None otherwise

Return type str

set_ui_text(*title*, *blurb*, *clusteruuid=None*)
Set UI text (title/blurb) for a specific cluster

Parameters **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the ntnx.client.ApiClient *connection_type* is set to *pc*.

Returns *updated* if changed, *added* if created or None otherwise

Return type str

update_alert_config(*email_list*, *enable=True*, *enable_default=True*, *enable_digest=True*, *nutanix_default_email='nos-alerts@nutanix.com'*, *clusteruuid=None*)
Update alert configuration for specified cluster

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the ntnx.client.ApiClient *connection_type* is set to *pc*.
- **email_list** (*list of str*) –
- **enable** (*bool, optional*) –
- **enable_default** (*bool, optional*) –
- **enable_digest** (*bool, optional*) –
- **nutanix_default_email** (*str, optional*) –

update_auth_dir(*name*, *directory_type*, *directory_url*, *domain*, *username*, *password*, *recursive=False*, *connection_type='LDAP'*, *clusteruuid=None*)
Update authentication directory for a specific cluster

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the ntnx.client.ApiClient *connection_type* is set to *pc*.
- **name** (*str, optional*) – Directory name
- **directory_url** (*str*) – ldap/ldaps URL to connect to the domain including the port your LDAP target is listening on. eg ldap://192.168.1.10:384
- **domain** (*str*) – Fully qualified name of the domain. eg nutanix.local

- **username** (*str*) – Username to authenticate to the domain
- **password** (*str*) – Password for user to authenticate to the domain
- **recursive** (*bool, optional*) – Whether to search for nested groups
- **directory_type** (*str('ACTIVE_DIRECTORY', 'OPEN_LDAP'), optional*) – Type of directory
- **connection_type** (*str('LDAP'), optional*) – Type of connection

update_auth_dir_role_mapping(*directory*, *directory_entities*, *directory_entity_type*, *cluster_admin=False*, *user_admin=False*, *clusteruuid=None*)

Update authentication role mapping for a named authentication directory on a specific cluster

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.
- **directory** (*str*) – Name of directory.
- **directory_entities** (*list of str*) – List of users/groups to add.
- **directory_entity_type** (*str('USER', 'GROUP')*) – Type of directory entity being added.
- **cluster_admin** (*bool, optional*) – Whether to grant user *Cluster Admin* privilege
- **user_admin** (*bool, optional*) – Whether to grant user *User Admin* privilege

update_local_user(*username*, *password*, *firstname*, *lastname*, *email*, *enabled=True*, *cluster_admin=False*, *user_admin=False*, *language='en-US'*, *clusteruuid=None*)

Update local user on a specific cluster. User is added with cluster viewer rights. Additional rights have to be added after the user is created

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.
- **username** (*str*) – Username
- **password** (*str*) – Password
- **firstname** (*str, optional*) – First name of user
- **lastname** (*str, optional*) – Last name of user
- **email** (*str, optional*) – Email address for user
- **cluster_admin** (*bool, optional*) – Whether to grant user *Cluster Admin* privilege
- **user_admin** (*bool, optional*) – Whether to grant user *User Admin* privilege
- **language** (*str('en-US','zh-CN','ja-JP'), optional*) – Localization region for user account *(default='en-US')*
- **enabled** (*bool, optional*) – Enable user account *(default=True)*

update_proxy(*name*, *address*, *port*, *proxy_types*, *username=None*, *password=None*, *clusteruuid=None*)

Add proxy configuration for a specific cluster

Parameters **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

update_pulse (*enable*, *email_address_list=None*, *email_nutanix=False*, *clusteruuid=None*)
Get pulse config for a specific cluster

Parameters **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

update_smtp (*address*, *from_email_address*, *port*, *secure_mode=None*, *username=None*, *password=None*, *clusteruuid=None*)
Update smtp config for a specific cluster

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.
- **address** (*str*) –
- **from_email_address** (*str*) –
- **port** (*int*) –
- **secure_mode** (*str, optional*) –
- **username** (*str, optional*) –
- **password** (*str, optional*) –

4.2.2 Cluster

class ntnx_api.prism.**Cluster** (*api_client*)
A class to represent a Nutanix Cluster

Parameters **api_client** (`ntnx.client.ApiClient`) – Initialized API client class

get (*clusteruuid=None*, *refresh=False*)
Retrieve data for a specific cluster

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.
- **refresh** (*bool, optional*) – Refresh data for this cluster if it already exists.

Returns A dictionary describing the configuration of the cluster.

Return type ResponseDict

get_all_uuids ()
Retrieve a list of all clusters.

Returns A list of dictionaries describing the configuration of each cluster.

Return type ResponseList

Note: Will return all registered clusters when *connection_type=='pc'*

Note: Will only return one cluster when *connection_type=='pe'*

get_ha(*clusteruuid=None*)
Retrieve HA data for a specific cluster

Parameters **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns A dictionary describing the HA configuration of the cluster.

Return type ResponseDict

Note: Cluster HA configuration will only present for cluster running the AHV hypervisor.

search_name(*name*, *clusteruuid=None*)
Retrieve data for a specific cluster, in a specific cluster by host cluster

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.
- **name** (*str, optional*) – A host name to search for.

Returns A dictionary describing the found cluster.

Return type ResponseDict

search_uuid(*uuid*, *clusteruuid=None*)
Retrieve data for a specific cluster, in a specific cluster by host uuid

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.
- **uuid** (*str, optional*) – A cluster uuid to search for.

Returns A dictionary describing the found cluster.

Return type ResponseDict

4.2.3 Hosts

class ntnx_api.prism.**Hosts**(*api_client*)
A class to represent a Nutanix Clusters Hosts

Parameters **api_client** (`ntnx.client.ApiClient`) – Initialized API client class

get(*clusteruuid=None*)
Retrieve data for each host in a specific cluster

Parameters **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns A list of dictionaries describing each host from the specified cluster.

Return type ResponseList

get_categories (*uuid*, *refresh=False*)
Retrieve the categories assigned to the specified host if connected to a prism central

Parameters

- **uuid** (*str*) – The UUID of a host.
- **refresh** (*bool, optional*) – Whether to refresh the class dataset (default=False).

Returns A dictionary with all categories for the specified host.

Return type ResponseDict

get_metadata (*refresh=False*)
Retrieve metadata for each host from the connected PC instance

Returns A list of dictionaries describing each vm from the specified cluster.

Return type ResponseList

get_project (*uuid*, *refresh=False*)
Retrieve the project assigned to the specified host if connected to a prism central

Parameters

- **uuid** (*str*) – The UUID of a host.
- **refresh** (*bool, optional*) – Whether to refresh the class dataset (default=False).

Returns A string containing the project name.

Return type str

search_ip (*ip_address*, *clusteruuid=None*, *refresh=False*)
Retrieve data for a specific host, in a specific cluster by ip_address. The CVM, Hypervisor and IPMI IP addresses will be tested

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.
- **ip_address** (*str, optional*) – A host name to search for.
- **refresh** (*bool, optional*) – Whether to refresh the class dataset (default=False).

Returns A dictionary describing the found host.

Return type ResponseDict

search_name (*name*, *clusteruuid=None*, *refresh=False*)
Retrieve data for a specific host, in a specific cluster by host name

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

- **name** (*str, optional*) – A host name to search for.
- **refresh** (*bool, optional*) – Whether to refresh the class dataset (default=False).

Returns A dictionary describing the found host.

Return type ResponseDict

search_uuid(*uuid*, *clusteruuid=None*, *refresh=False*)
Retrieve data for a specific host, in a specific cluster by host uuid

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.
- **uuid** (*str, optional*) – A host uuid to search for.

Returns A dictionary describing the found host.

Return type ResponseDict

4.2.4 Vms

class ntnx_api.prism.**Vms**(*api_client*)
A class to represent a Nutanix Clusters Virtual Machines

Parameters **api_client** (`ntnx.client.ApiClient`) – Initialized API client class

add_disks(*vm_uuid: str*, *disks: list = None*, *add_cdrom: bool = False*, *wait: bool = True*, *clusteruuid: str = None*)
Add disks from a dict to an existing virtual machine.

Parameters

- **vm_uuid** (*str*) – The uuid for the VM to be have disks added.
- **disks** (*list, optional*) – A list of vdisks dicts to be added to this VM (default='null').

  The dictionary format per-vDisk is as follows::

  – bus (str, optional, default='scsi'). The bus to use for the vDisk. Choice of 'scsi', 'ide', 'pci', 'sata', 'spapr', 'nvme'

  – size_gb (int, optional). Size of vDisk in GB. Use this when creating a new disk or cloning from an image. Can be used to increase the size of vDisk created from an image

  – storage_container_name (str, optional). Name of Storage Container. Only used when creating a new vDisk. Mutually exclusive with "storage_container_uuid"

  – storage_container_uuid (str, optional). UUID of Storage Container. Only used when creating a new vDisk. Mutually exclusive with "storage_container_name"

  – image_name (str, optional). Name of Image to clone from. Only used when creating a new vDisk from an existing image. Mutually exclusive with "image_uuid"

  – image_uuid (str, optional). UUID of Image to clone from. Only used when creating a new vDisk from an existing image. Mutually exclusive with "image_name"

- volume_group_name (str, optional). Name of Volume Group to attach. Only used when attaching an existing volume group. Mutually exclusive with "volume_group_uuid"
- volume_group_uuid (str, optional). UUID of Volume Group to attach. Only used when attaching an existing volume group. Mutually exclusive with "volume_group_name"
- flash_mode (bool, optional, default=False). True or False
- label (str, optional). Unknown

Examples;

1. Add a single virtual disk - [{'bus': 'scsi', 'storage_container_name': 'default', 'size_gb': 50, },]
2. Add multiple virtual disk - [{'bus': 'scsi', 'storage_container_name': 'default', 'size_gb': 50, }, {'bus': 'scsi', 'storage_container_name': 'default', 'size_gb': 200, },]
3. Add a single virtual disk with flash mode enabled - [{'bus': 'scsi', 'storage_container_name': 'default', 'size_gb': 15, 'flash_mode': True},]
4. Add a single virtual disk from an image - [{'bus': 'scsi', 'image_name': 'centos8', },]
5. Add a single virtual disk from an image with a new minimum size - [{'bus': 'scsi', 'image_name': 'centos8', 'size_gb': 500, },]
6. Add a volume group - [{'bus': 'scsi', 'volume_group_name': 'volume_group_database1', },]

- **add_cdrom** (*bool, optional*) – Whether to add a cdrom drive to this VM (default=True)
- **wait** (*bool, optional*) – Wait for the task to complete. (defaults=True)
- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns True or False to indicate whether the disk was successfully added.

Return type bool

add_nics(*vm_uuid: str*, *nics: list = None*, *wait: bool = True*, *clusteruuid: str = None*)

Add one or more nics to an existing virtual machine.

Parameters

- **vm_uuid** (*str*) – The uuid for the VM to have the nic attached.
- **nics** (*list, optional*) – A list of NIC dicts to be added to this VM (default='null').

  The dictionary format per-NIC is as follows::

  - network_name (str, optional). The name of the network or port group to attach the NIC onto. Mutually exclusive with "network_uuid".
  - network_uuid (str, optional). The uuid of the network or port group to attach the NIC onto. Mutually exclusive with "network_name".

- adaptor_type (str, optional, default='e1000'). The network adaptor type to use for the NIC. Choice of 'e1000', 'e1000e', 'pcnet32', 'vmxnet', 'vmxnet2', 'vmxnet3',
- connect (bool, optional, default=True). Whether to connect the NIC to the network.
- mac_address (str, optional, default=None). A user-defined MAC address to use for this NIC.
- ipam (bool, optional, default=False). Whether to use AHV IPAM to automatically provide an IP address.
- requested_ip_address (str, optional). A user-defined IP address to use in conjunction with AHV IPAM. Requires 'ipam' to also be set to True.

Examples;

1. Create a simple NIC - [{'network_name': 'vm network',},]
2. Create a NIC with AHV IPAM - [{'network_name': 'vm network', 'ipam': True, },]
3. Create multiple NICs with mixed configuration - [{'network_name': 'vm network 1', }, {'network_name': 'vm network 2', 'ipam': True, },]
4. Create a NIC with AHV IPAM and a defined IP address - [{'network_name': 'vm network', 'ipam': True, 'requested_ip_address': '172.16.100.51', },]

- **wait** (*bool, optional*) – Wait for the task to complete. (defaults=True)
- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns True or False to indicate whether the nic(s) were successfully added.

Return type bool

attach_vg(*index: int*, *uuid: str*, *vm_uuid: str*, *wait: bool = True*, *clusteruuid: str = None*)

Attach a volume group to an existing virtual machine.

Parameters

- **index** (*int*) – The index where the volume groups will be attached.
- **uuid** (*str*) – The uuid of the volume group.
- **vm_uuid** (*str*) – The uuid for the VM to have the volume group attached.
- **wait** (*bool, optional*) – Wait for the task to complete. (defaults=True)
- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns True or False to indicate whether the volume group was successfully attached.

Return type bool

clone_name(*source_name: str*, *name: str*, *cores: int = None*, *sockets: int = None*, *memory_gb: int = None*, *nics: list = []*, *sysprep: str = None*, *cloudinit: str = None*, *wait: bool = True*, *clusteruuid: str = None*)

Clones an existing virtual machine based on the provided virtual machine name.

Parameters

- **source_name** (*str*) – The name for the virtual machine to be cloned.
- **name** (*str*) – The name for the new virtual machine.
- **cores** (*int*) – The number of virtual CPU cores per virtual CPU socket
- **sockets** (*int, optional*) – The number of virtual CPU sockets to distribute the defined vCPUs over (default=1)
- **memory_gb** (*int*) – The amount of memory in GB to be assigne to this VM
- **nics** (*list, optional*) – A list of NIC dicts to be added to this VM (default='null').

  The dictionary format per-NIC is as follows::

  – network_name (str, optional). The name of the network or port group to attach the NIC onto. Mutually exclusive with "network_uuid".

  – network_uuid (str, optional). The uuid of the network or port group to attach the NIC onto. Mutually exclusive with "network_name".

  – adaptor_type (str, optional, default='e1000'). The network adaptor type to use for the NIC. Choice of 'e1000', 'e1000e', 'pcnet32', 'vmxnet', 'vmxnet2', 'vmxnet3',

  – connect (bool, optional, default=True). Whether to connect the NIC to the network.

  – mac_address (str, optional, default=None). A user-defined MAC address to use for this NIC.

  – ipam (bool, optional, default=False). Whether to use AHV IPAM to automatically provide an IP address.

  – requested_ip_address (str, optional). A user-defined IP address to use in conjunction with AHV IPAM. Requires 'ipam' to also be set to True.

  Examples;

  1. Create a simple NIC - [{'network_name': 'vm network',},]
  2. Create a NIC with AHV IPAM - [{'network_name': 'vm network', 'ipam': True, },]
  3. Create multiple NICs with mixed configuration - [{'network_name': 'vm network 1', }, {'network_name': 'vm network 2', 'ipam': True, },]
  4. Create a NIC with AHV IPAM and a defined IP address - [{'network_name': 'vm network', 'ipam': True, 'requested_ip_address': '172.16.100.51', },]
- **sysprep** (*str, optional*) – The sysprep XML string to use to customize this VM upon first power on. Only applicable for AHV and a Windows OS. (default='null')
- **cloudinit** (*str, optional*) – The cloudinit text string to use to customize this VM upon first power on. Only applicable for AHV and a Linux OS. (default='null')
- **wait** (*bool, optional*) – Wait for the task to complete. (defaults=True)
- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns True or False to indicate whether the VM was successfully cloned.

Return type bool

> **Warning:** As VM names are not necessarily unique, the first result returned will be used.

clone_uuid(*source_uuid: str*, *name: str*, *cores: int = None*, *sockets: int = None*, *memory_gb: int = None*, *nics: list = []*, *sysprep: str = None*, *cloudinit: str = None*, *wait: bool = True*, *clusteruuid: str = None*)

Clones an existing virtual machine based on the provided virtual machine uuid.

Parameters

- **source_uuid** (*str*) – The uuid for the virtual machine to be cloned.
- **name** (*str*) – The name for the new virtual machine.
- **cores** (*int*) – The number of virtual CPU cores per virtual CPU socket
- **sockets** (*int, optional*) – The number of virtual CPU sockets to distribute the defined vCPUs over (default=1)
- **memory_gb** (*int*) – The amount of memory in GB to be assigne to this VM
- **nics** (*list, optional*) – A list of NIC dicts to be added to this VM (default='null').

  The dictionary format per-NIC is as follows::

  – network_name (str, optional). The name of the network or port group to attach the NIC onto. Mutually exclusive with "network_uuid".

  – network_uuid (str, optional). The uuid of the network or port group to attach the NIC onto. Mutually exclusive with "network_name".

  – adaptor_type (str, optional, default='e1000'). The network adaptor type to use for the NIC. Choice of 'e1000', 'e1000e', 'pcnet32', 'vmxnet', 'vmxnet2', 'vmxnet3',

  – connect (bool, optional, default=True). Whether to connect the NIC to the network.

  – mac_address (str, optional, default=None). A user-defined MAC address to use for this NIC.

  – ipam (bool, optional, default=False). Whether to use AHV IPAM to automatically provide an IP address.

  – requested_ip_address (str, optional). A user-defined IP address to use in conjunction with AHV IPAM. Requires 'ipam' to also be set to True.

  Examples;

  1. Create a simple NIC - [{'network_name': 'vm network',},]
  2. Create a NIC with AHV IPAM - [{'network_name': 'vm network', 'ipam': True, },]
  3. Create multiple NICs with mixed configuration - [{'network_name': 'vm network 1', }, {'network_name': 'vm network 2', 'ipam': True, },]
  4. Create a NIC with AHV IPAM and a defined IP address - [{'network_name': 'vm network', 'ipam': True, 'requested_ip_address': '172.16.100.51', },]

- **sysprep** (*str, optional*) – The sysprep XML string to use to customize this VM upon first power on. Only applicable for AHV and a Windows OS. (default='null')
- **cloudinit** (*str, optional*) – The cloudinit text string to use to customize this VM upon first power on. Only applicable for AHV and a Linux OS. (default='null')
- **wait** (*bool, optional*) – Wait for the task to complete. (defaults=True)
- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns True or False to indicate whether the VM was successfully cloned.

Return type bool

create(*name: str*, *cores: int*, *memory_gb: int*, *sockets: int = 1*, *vcpu_reservation_hz: int = 0*, *memory_reservation_gb: int = 0*, *description: str = ''*, *power_state: str = 'on'*, *disks: list = []*, *storage_container_uuid: str = None*, *nics: list = []*, *gpus: list = []*, *serial_ports: list = []*, *timezone: str = 'UTC'*, *sysprep: str = None*, *cloudinit: str = None*, *add_cdrom: bool = True*, *ha_priority: int = 0*, *machine_type: str = 'pc'*, *wait: bool = True*, *clusteruuid: str = None*)

Create a new virtual machine.

Parameters

- **name** (*str*) – The name for the VM to be created. VM names do not have to be unique.
- **description** (*str, optional*) – A description for the VM (default='')
- **cores** (*int*) – The number of virtual CPU cores per virtual CPU socket
- **sockets** (*int, optional*) – The number of virtual CPU sockets to distribute the defined vCPUs over (default=1)
- **memory_gb** (*int*) – The amount of memory in GB to be assigne to this VM
- **vcpu_reservation_hz** (*int, optional*) – A CPU reservation in hz for this VM. Only applicable on the ESXi hypervisor. (default=0)
- **memory_reservation_gb** (*int, optional*) – An amount of memory to lock to this VM. Only applicable on the ESXi hypervisor. (default=0)
- **power_state** (*str('on', 'off'), optional*) – The desired power state for this VM after creation. (default='on')
- **storage_container_uuid** (*str, optional*) – The UUID of the storage contain on which to create this VM. Only applicable on the ESXi hypervisor. (default='null')
- **add_cdrom** (*bool, optional*) – Whether to add a cdrom drive to this VM (default=True)
- **disks** (*list, optional*) – A list of vdisks dicts to be added to this VM (default='null').

  The dictionary format per-vDisk is as follows::

  - bus (str, optional, default='scsi'). The bus to use for the vDisk. Choice of 'scsi', 'ide', 'pci', 'sata', 'spapr', 'nvme'
  - size_gb (int, optional). Size of vDisk in GB. Use this when creating a new disk or cloning from an image. Can be used to increase the size of vDisk created from an image

- storage_container_name (str, optional). Name of Storage Container. Only used when creating a new vDisk. Mutually exclusive with "storage_container_uuid"
- storage_container_uuid (str, optional). UUID of Storage Container. Only used when creating a new vDisk. Mutually exclusive with "storage_container_name"
- image_name (str, optional). Name of Image to clone from. Only used when creating a new vDisk from an existing image. Mutually exclusive with "image_uuid"
- image_uuid (str, optional). UUID of Image to clone from. Only used when creating a new vDisk from an existing image. Mutually exclusive with "image_name"
- volume_group_name (str, optional). Name of Volume Group to attach. Only used when attaching an existing volume group. Mutually exclusive with "volume_group_uuid"
- volume_group_uuid (str, optional). UUID of Volume Group to attach. Only used when attaching an existing volume group. Mutually exclusive with "volume_group_name"
- flash_mode (bool, optional, default=False). True or False
- label (str, optional). Unknown

Examples;

1. Add a single virtual disk - [{'bus': 'scsi', 'storage_container_name': 'default', 'size_gb': 50, },]
2. Add multiple virtual disk - [{'bus': 'scsi', 'storage_container_name': 'default', 'size_gb': 50, }, {'bus': 'scsi', 'storage_container_name': 'default', 'size_gb': 200, },]
3. Add a single virtual disk with flash mode enabled - [{'bus': 'scsi', 'storage_container_name': 'default', 'size_gb': 15, 'flash_mode': True},]
4. Add a single virtual disk from an image - [{'bus': 'scsi', 'image_name': 'centos8', },]
5. Add a single virtual disk from an image with a new minimum size - [{'bus': 'scsi', 'image_name': 'centos8', 'size_gb': 500, },]
6. Add a volume group - [{'bus': 'scsi', 'volume_group_name': 'volume_group_database1', },]

- **nics** (*list, optional*) – A list of NIC dicts to be added to this VM (default='null').

  The dictionary format per-NIC is as follows::

  - network_name (str, optional). The name of the network or port group to attach the NIC onto. Mutually exclusive with "network_uuid".
  - network_uuid (str, optional). The uuid of the network or port group to attach the NIC onto. Mutually exclusive with "network_name".
  - adaptor_type (str, optional, default='e1000'). The network adaptor type to use for the NIC. Choice of 'e1000', 'e1000e', 'pcnet32', 'vmxnet', 'vmxnet2', 'vmxnet3',
  - connect (bool, optional, default=True). Whether to connect the NIC to the network.

- mac_address (str, optional, default=None). A user-defined MAC address to use for this NIC.
- ipam (bool, optional, default=False). Whether to use AHV IPAM to automatically provide an IP address.
- requested_ip_address (str, optional). A user-defined IP address to use in conjunction with AHV IPAM. Requires 'ipam' to also be set to True.

Examples;

1. Create a simple NIC - [{'network_name': 'vm network',},]
2. Create a NIC with AHV IPAM - [{'network_name': 'vm network', 'ipam': True, },]
3. Create multiple NICs with mixed configuration - [{'network_name': 'vm network 1', }, {'network_name': 'vm network 2', 'ipam': True, },]
4. Create a NIC with AHV IPAM and a defined IP address - [{'network_name': 'vm network', 'ipam': True, 'requested_ip_address': '172.16.100.51', },]

- **gpus** (*list, optional*) – A list of GPU dicts to be added to the VM (default='null')

  The dictionary format per-GPU is as follows::

  - device_id (int).
  - gpu_type (str, optional, default='null'). The type of GPU to add. Choice of 'pass_through_graphics'
  - gpu_vendor (str, optional, default='null'). The GPU vendor to add. Choice of 'nvidia',

- **serial_ports** (*list, optional*) – A list of serial port dicts to be added to the VM (default='null')

  The dictionary format per-serial port is as follows::

  - port_index (int).
  - port_type (str, optional, default='null'). The type of serial port to add. Choice of 'null', 'server'

- **timezone** (*str, optional*) – The timezone for the virtual machine (default='UTC').
- **sysprep** (*str, optional*) – The sysprep XML string to use to customize this VM upon first power on. Only applicable for AHV and a Windows OS. (default='null')
- **cloudinit** (*str, optional*) – The cloudinit text string to use to customize this VM upon first power on. Only applicable for AHV and a Linux OS. (default='null')
- **ha_priority** (*int, optional*) – VM HA priority. Only applicable to ESXi hypervisor (default=0)
- **machine_type** (*str('pc', 'pseries', 'q35'), optional*) – The type of VM being deployed. This donates the target cpu architecture of the cluster. (default='pc')
- **wait** (*bool, optional*) – Wait for the task to complete. (defaults=True)

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns True or False to indicate whether the VM was successfully created.

Return type bool

delete_name(*name: str*, *snapshots: bool = False*, *vg_detach: bool = True*, *wait: bool = True*, *clusteruuid: str = None*)

Delete an existing virtual machine by name.

Parameters

- **name** (*str*) – The name for the VM to be deleted.
- **snapshots** (*bool, optional*) – Whether to also delete VM snapshots when deleting the VM. (Default=False)
- **vg_detach** (*bool, optional*) – Whether to also detatch any connected volume groups when deleting the VM. (Default=True)
- **wait** (*bool, optional*) – Wait for the task to complete. (defaults=True)
- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns True or False to indicate whether the VM was sucessfully created.

Return type bool

> **Warning:** As VM names are not necessarily unique, the first result returned will be used.

delete_uuid(*uuid: str*, *snapshots: bool = False*, *vg_detach: bool = True*, *wait: bool = True*, *clusteruuid: str = None*)

Delete an existing virtual machine by virtual machine uuid.

Parameters

- **uuid** (*str*) – The uuid for the VM to be deleted.
- **snapshots** (*bool, optional*) – Whether to also delete VM snapshots when deleting the VM. (Default=False)
- **vg_detach** (*bool, optional*) – Whether to also detatch any connected volume groups when deleting the VM. (Default=True)
- **wait** (*bool, optional*) – Wait for the task to complete. (defaults=True)
- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns True or False to indicate whether the VM was sucessfully created.

Return type bool

detach_vg(*index: int*, *uuid: str*, *vm_uuid: str*, *wait: bool = True*, *clusteruuid: str = None*)

Detach a volume group to an existing virtual machine.

Parameters

- **index** (*int*) – The index where the volume groups will be attached.

- **uuid** (*str*) – The uuid of the volume group.
- **vm_uuid** (*str*) – The uuid for the VM to have the volume group attached.
- **wait** (*bool, optional*) – Wait for the task to complete. (defaults=True)
- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns True or False to indicate whether the volume group was successfully detached.

Return type bool

get (*clusteruuid=None*, *include_disks=True*, *include_nics=True*)
Retrieve host data for each virtual machine in a specific cluster

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.
- **include_disks** (*bool, optional*) – Include VM disk info in returned data (Default=False)
- **include_nics** (*bool, optional*) – Include VM nic info in returned data (Default=False)

Returns A list of dictionaries describing each vm from the specified cluster.

Return type ResponseList

get_categories (*uuid*, *refresh=False*)
Retrieve the categories assigned to the specified VM if connected to a prism central

Parameters

- **uuid** (*str*) – The UUID of a VM.
- **refresh** (*bool, optional*) – Whether to refresh the class VM Metadata dataset (default=False).

Returns A dictionary with all .

Return type ResponseDict

get_metadata (*refresh=False*)
Retrieve metadata for each virtual machine from the connected PC instance

Returns A list of dictionaries describing each vm from the specified cluster.

Return type ResponseList

get_project (*uuid*, *refresh=False*)
Retrieve the project assigned to the specified VM if connected to a prism central

Parameters

- **uuid** (*str*) – The UUID of a VM.
- **refresh** (*bool, optional*) – Whether to refresh the class VM Metadata dataset (default=False).

Returns A string containing the project name.

Return type str

get_protection_rules(*uuid*, *refresh=False*)
Retrieve the protection rules assigned to the specified VM if connected to a prism central

Parameters

- **uuid** (*str*) – The UUID of a VM.
- **refresh** (*bool, optional*) – Whether to refresh the class VM Metadata dataset (default=False).

Returns A dictionary with all .

Return type ResponseDict

power_state(*uuid: str*, *desired_state: str = 'on'*, *wait: bool = True*, *clusteruuid: str = None*)
Change the power state of a specific virtual machine.

Parameters

- **uuid** (*str*) – The uuid for the virtual machine to have the nic attached.
- **desired_state** (*str('on', 'off', 'powercycle', 'reset', 'pause', 'suspend', 'resume', 'save', 'acpi_shutdown', 'acpi_reboot')*) – The desired power state for the virtual machine. .
- **wait** (*bool, optional*) – Wait for the task to complete. (defaults=True)
- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns True or False to indicate whether the virtual machines power state was successfully changed.

Return type bool

remove_disk(*vm_uuid: str*, *bus: str = None*, *index: int = None*, *wait: bool = True*, *clusteruuid: str = None*)
Remove a disk from a VM

Parameters

- **vm_uuid** (*str*) – The uuid for the VM to have a disk removed.
- **bus** (*str*) – The bus where the disk to be removed is located.
- **index** (*int*) – The index for the disk to be removed.
- **wait** (*bool, optional*) – Wait for the task to complete. (defaults=True)
- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns True or False to indicate whether the disk was successfully removed.

Return type bool

remove_nic(*vm_uuid: str*, *mac_address: str*, *wait: bool = True*, *clusteruuid: str = None*)
Remove a single nic from an existing virtual machine.

Parameters

- **vm_uuid** (*str*) – The uuid for the VM to have the nic attached.
- **mac_address** (*str*) – The mac address for the nic to be removed.
- **wait** (*bool, optional*) – Wait for the task to complete. (defaults=True)

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns True or False to indicate whether the nic was successfully removed.

Return type bool

search_name (*name*, *clusteruuid=None*, *refresh=False*)
Retrieve data for a specific vm, in a specific cluster by vm name

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.
- **name** (*str, optional*) – A vm name to search for.
- **refresh** (*bool, optional*) – Whether to refresh the class VM dataset (default=False).

Returns A dictionary describing the found vm.

Return type ResponseDict

search_uuid (*uuid*, *clusteruuid=None*, *refresh=False*)
Retrieve data for a specific vm, in a specific cluster by vm uuid

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.
- **uuid** (*str, optional*) – A vm uuid to search for.
- **refresh** (*bool, optional*) – Whether to refresh the class VM dataset (default=False).

Returns A dictionary describing the found vm.

Return type ResponseDict

set_categories (*category*)

set_cateories (*categories*, *create_missing=True*)

update_name (*name: str*, *new_name: str = None*, *cores: int = None*, *sockets: int = None*, *memory_gb: int = None*, *vcpu_reservation_hz: int = None*, *memory_reservation_gb: int = None*, *description: str = None*, *disks: list = []*, *nics: list = []*, *gpus: list = []*, *serial_ports: list = []*, *timezone: str = None*, *add_cdrom: bool = None*, *ha_priority: int = None*, *force: bool = False*, *wait: bool = True*, *clusteruuid: str = None*)
Updates a specific virtual machine by the vm name provided.

Parameters

- **name** (*str*) – The name for the virtual machine to be updated.
- **new_name** (*str, optional*) – A new name for the virtual machine.
- **cores** (*int*) – The number of virtual CPU cores per virtual CPU socket
- **sockets** (*int, optional*) – The number of virtual CPU sockets to distribute the defined vCPUs over (default=1)
- **memory_gb** (*int*) – The amount of memory in GB to be assigne to this VM

- **vcpu_reservation_hz** (*int, optional*) – A CPU reservation in hz for this VM. Only applicable on the ESXi hypervisor. (default=0)
- **memory_reservation_gb** (*int, optional*) – An amount of memory to lock to this VM. Only applicable on the ESXi hypervisor. (default=0)
- **description** (*str, optional*) – A description for the VM (default='')
- **add_cdrom** (*bool, optional*) – Whether to add a cdrom drive to this VM (default=True)
- **disks** (*list, optional*) – A list of vdisks dicts to be added to this VM (default='null').

  The dictionary format per-vDisk is as follows::

  – bus (str, optional, default='scsi'). The bus to use for the vDisk. Choice of 'scsi', 'ide', 'pci', 'sata', 'spapr', 'nvme'

  – size_gb (int, optional). Size of vDisk in GB. Use this when creating a new disk or cloning from an image. Can be used to increase the size of vDisk created from an image

  – storage_container_name (str, optional). Name of Storage Container. Only used when creating a new vDisk. Mutually exclusive with "storage_container_uuid"

  – storage_container_uuid (str, optional). UUID of Storage Container. Only used when creating a new vDisk. Mutually exclusive with "storage_container_name"

  – image_name (str, optional). Name of Image to clone from. Only used when creating a new vDisk from an existing image. Mutually exclusive with "image_uuid"

  – image_uuid (str, optional). UUID of Image to clone from. Only used when creating a new vDisk from an existing image. Mutually exclusive with "image_name"

  – volume_group_name (str, optional). Name of Volume Group to attach. Only used when attaching an existing volume group. Mutually exclusive with "volume_group_uuid"

  – volume_group_uuid (str, optional). UUID of Volume Group to attach. Only used when attaching an existing volume group. Mutually exclusive with "volume_group_name"

  – flash_mode (bool, optional, default=False). True or False

  – label (str, optional). Unknown

  Examples;

  1. Add a single virtual disk - [{'bus': 'scsi', 'storage_container_name': 'default', 'size_gb': 50, },]
  2. Add multiple virtual disk - [{'bus': 'scsi', 'storage_container_name': 'default', 'size_gb': 50, }, {'bus': 'scsi', 'storage_container_name': 'default', 'size_gb': 200, },]
  3. Add a single virtual disk with flash mode enabled - [{'bus': 'scsi', 'storage_container_name': 'default', 'size_gb': 15, 'flash_mode': True},]
  4. Add a single virtual disk from an image - [{'bus': 'scsi', 'image_name': 'centos8', },]

5. Add a single virtual disk from an image with a new minimum size - [{'bus': 'scsi', 'image_name': 'centos8', 'size_gb': 500, },]
6. Add a volume group - [{'bus': 'scsi', 'volume_group_name': 'volume_group_database1', },]

- **nics** (*list, optional*) – A list of NIC dicts to be added to this VM (default='null').

  The dictionary format per-NIC is as follows::

  - network_name (str, optional). The name of the network or port group to attach the NIC onto. Mutually exclusive with "network_uuid".
  - network_uuid (str, optional). The uuid of the network or port group to attach the NIC onto. Mutually exclusive with "network_name".
  - adaptor_type (str, optional, default='e1000'). The network adaptor type to use for the NIC. Choice of 'e1000', 'e1000e', 'pcnet32', 'vmxnet', 'vmxnet2', 'vmxnet3',
  - connect (bool, optional, default=True). Whether to connect the NIC to the network.
  - mac_address (str, optional, default=None). A user-defined MAC address to use for this NIC.
  - ipam (bool, optional, default=False). Whether to use AHV IPAM to automatically provide an IP address.
  - requested_ip_address (str, optional). A user-defined IP address to use in conjunction with AHV IPAM. Requires 'ipam' to also be set to True.

  Examples;

  1. Create a simple NIC - [{'network_name': 'vm network',},]
  2. Create a NIC with AHV IPAM - [{'network_name': 'vm network', 'ipam': True, },]
  3. Create multiple NICs with mixed configuration - [{'network_name': 'vm network 1', }, {'network_name': 'vm network 2', 'ipam': True, },]
  4. Create a NIC with AHV IPAM and a defined IP address - [{'network_name': 'vm network', 'ipam': True, 'requested_ip_address': '172.16.100.51', },]

- **gpus** (*list, optional*) – A list of GPU dicts to be added to the VM (default='null')

  The dictionary format per-GPU is as follows::

  - device_id (int).
  - gpu_type (str, optional, default='null'). The type of GPU to add. Choice of 'pass_through_graphics'
  - gpu_vendor (str, optional, default='null'). The GPU vendor to add. Choice of 'nvidia',

- **serial_ports** (*list, optional*) – A list of serial port dicts to be added to the VM (default='null')

  The dictionary format per-serial port is as follows::

  - port_index (int).

- port_type (str, optional, default='null'). The type of serial port to add. Choice of 'null', 'server'

- **timezone** (*str, optional*) – The timezone for the virtual machine (default='UTC').
- **ha_priority** (*int, optional*) – VM HA priority. Only applicable to ESXi hypervisor (default=0)
- **force** (*bool, optional*) – If the VM is not in a power state that will allow the change to be made, force will change the VM power state to apply the update, then return the VM to its original power state once the update has been completed. (defaults=False)
- **wait** (*bool, optional*) – Wait for the task to complete. (defaults=True)
- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns True or False to indicate whether the virtual machine was successfully updated.

Return type bool

> **Warning:** As VM names are not necessarily unique, the first result returned will be used.

update_uuid(*uuid: str*, *new_name: str = None*, *cores: int = None*, *sockets: int = None*, *memory_gb: int = None*, *vcpu_reservation_hz: int = None*, *memory_reservation_gb: int = None*, *description: str = None*, *disks: list = []*, *nics: list = []*, *gpus: list = []*, *serial_ports: list = []*, *timezone: str = None*, *add_cdrom: bool = None*, *ha_priority: int = None*, *force: bool = False*, *wait: bool = True*, *clusteruuid: str = None*)

Updates a specific virtual machine by the uuid provided

Parameters

- **uuid** (*str*) – The uuid for the virtual machine to be updated.
- **new_name** (*str, optional*) – A new name for the virtual machine.
- **memory_gb** (*int*) – The amount of memory in GB to be assigne to this VM
- **cores** (*int*) – The number of virtual CPU cores per virtual CPU socket
- **sockets** (*int, optional*) – The number of virtual CPU sockets to distribute the defined vCPUs over (default=1)
- **vcpu_reservation_hz** (*int, optional*) – A CPU reservation in hz for this VM. Only applicable on the ESXi hypervisor. (default=0)
- **memory_reservation_gb** (*int, optional*) – An amount of memory to lock to this VM. Only applicable on the ESXi hypervisor. (default=0)
- **description** (*str, optional*) – A description for the VM (default='')
- **add_cdrom** (*bool, optional*) – Whether to add a cdrom drive to this VM (default=True)
- **disks** (*list, optional*) – A list of vdisks dicts to be added to this VM (default='null').

  The dictionary format per-vDisk is as follows::

- bus (str, optional, default='scsi'). The bus to use for the vDisk. Choice of 'scsi', 'ide', 'pci', 'sata', 'spapr', 'nvme'
- size_gb (int, optional). Size of vDisk in GB. Use this when creating a new disk or cloning from an image. Can be used to increase the size of vDisk created from an image
- storage_container_name (str, optional). Name of Storage Container. Only used when creating a new vDisk. Mutually exclusive with "storage_container_uuid"
- storage_container_uuid (str, optional). UUID of Storage Container. Only used when creating a new vDisk. Mutually exclusive with "storage_container_name"
- image_name (str, optional). Name of Image to clone from. Only used when creating a new vDisk from an existing image. Mutually exclusive with "image_uuid"
- image_uuid (str, optional). UUID of Image to clone from. Only used when creating a new vDisk from an existing image. Mutually exclusive with "image_name"
- volume_group_name (str, optional). Name of Volume Group to attach. Only used when attaching an existing volume group. Mutually exclusive with "volume_group_uuid"
- volume_group_uuid (str, optional). UUID of Volume Group to attach. Only used when attaching an existing volume group. Mutually exclusive with "volume_group_name"
- flash_mode (bool, optional, default=False). True or False
- label (str, optional). Unknown

Examples;

1. Add a single virtual disk - [{'bus': 'scsi', 'storage_container_name': 'default', 'size_gb': 50, },]
2. Add multiple virtual disk - [{'bus': 'scsi', 'storage_container_name': 'default', 'size_gb': 50, }, {'bus': 'scsi', 'storage_container_name': 'default', 'size_gb': 200, },]
3. Add a single virtual disk with flash mode enabled - [{'bus': 'scsi', 'storage_container_name': 'default', 'size_gb': 15, 'flash_mode': True},]
4. Add a single virtual disk from an image - [{'bus': 'scsi', 'image_name': 'centos8', },]
5. Add a single virtual disk from an image with a new minimum size - [{'bus': 'scsi', 'image_name': 'centos8', 'size_gb': 500, },]
6. Add a volume group - [{'bus': 'scsi', 'volume_group_name': 'volume_group_database1', },]

- **nics** (*list, optional*) – A list of NIC dicts to be added to this VM (default='null').

  The dictionary format per-NIC is as follows::

  - network_name (str, optional). The name of the network or port group to attach the NIC onto. Mutually exclusive with "network_uuid".
  - network_uuid (str, optional). The uuid of the network or port group to attach the NIC onto. Mutually exclusive with "network_name".

- adaptor_type (str, optional, default='e1000'). The network adaptor type to use for the NIC. Choice of 'e1000', 'e1000e', 'pcnet32', 'vmxnet', 'vmxnet2', 'vmxnet3',
- connect (bool, optional, default=True). Whether to connect the NIC to the network.
- mac_address (str, optional, default=None). A user-defined MAC address to use for this NIC.
- ipam (bool, optional, default=False). Whether to use AHV IPAM to automatically provide an IP address.
- requested_ip_address (str, optional). A user-defined IP address to use in conjunction with AHV IPAM. Requires 'ipam' to also be set to True.

Examples;

1. Create a simple NIC - [{'network_name': 'vm network',},]
2. Create a NIC with AHV IPAM - [{'network_name': 'vm network', 'ipam': True, },]
3. Create multiple NICs with mixed configuration - [{'network_name': 'vm network 1', }, {'network_name': 'vm network 2', 'ipam': True, },]
4. Create a NIC with AHV IPAM and a defined IP address - [{'network_name': 'vm network', 'ipam': True, 'requested_ip_address': '172.16.100.51', },]

- **gpus** (*list, optional*) – A list of GPU dicts to be added to the VM (default='null')

  The dictionary format per-GPU is as follows::

  - device_id (int).
  - gpu_type (str, optional, default='null'). The type of GPU to add. Choice of 'pass_through_graphics'
  - gpu_vendor (str, optional, default='null'). The GPU vendor to add. Choice of 'nvidia',

- **serial_ports** (*list, optional*) – A list of serial port dicts to be added to the VM (default='null')

  The dictionary format per-serial port is as follows::

  - port_index (int).
  - port_type (str, optional, default='null'). The type of serial port to add. Choice of 'null', 'server'

- **timezone** (*str, optional*) – The timezone for the virtual machine (default='UTC').
- **ha_priority** (*int, optional*) – VM HA priority. Only applicable to ESXi hypervisor (default=0)
- **force** (*bool, optional*) – If the VM is not in a power state that will allow the change to be made, force will change the VM power state to apply the update, then return the VM to its original power state once the update has been completed. (defaults=False)
- **wait** (*bool, optional*) – Wait for the task to complete. (defaults=True)

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns True or False to indicate whether the virtual machine was successfully updated.

Return type bool

4.2.5 Images

class ntnx_api.prism.**Images**(*api_client*)

A class to represent a Nutanix Clusters Images

Parameters **api_client** (`ntnx.client.ApiClient`) – Initialized API client class

delete_name(*name*, *clusteruuid=None*, *wait=False*)

Delete an existing image based on the image name provided

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.
- **name** (*str*) – A image name to be deleted.
- **wait** (*bool, optional*) – Wait for the image task to complete. Defaults to False.

delete_uuid(*uuid*, *clusteruuid=None*, *wait=False*)

Delete an existing image based on the image uuid provided

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.
- **uuid** (*str*) – A image uuid to be deleted.
- **wait** (*bool, optional*) – Wait for the image task to complete. Defaults to False.

get(*clusteruuid=None*)

Retrieve data for each image in a specific cluster

Parameters **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns A list of dictionaries describing each image from the specified cluster.

Return type ResponseList

Note: Images are only present for cluster running the AHV hypervisor.

---

search_name(*name*, *clusteruuid=None*, *refresh=False*)

Retrieve data for a specific image, in a specific cluster by image name

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

- **name** (*str*) – A image name to search for.
- **refresh** (*bool, optional*) – Whether to refresh the data stored in the class prior to performing the search. Defaults to False.

Returns A dictionary describing the found image.

Return type ResponseDict

search_uuid(*uuid*, *clusteruuid=None*, *refresh=False*)
Retrieve data for a specific image, in a specific cluster by image uuid

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.
- **uuid** (*str*) – A image uuid to search for.
- **refresh** (*bool, optional*) – Whether to refresh the data stored in the class prior to performing the search. Defaults to False.

Returns A dictionary describing the found image.

Return type ResponseDict

upload_from_file(*name*, *file_path*, *storage_container_uuid*, *image_type='disk'*, *annotation=''*, *clusteruuid=None*, *wait=False*)
Upload an image from a file path. The target file path needs to be accessible on the device running this script.

Parameters

- **name** (*str*) – A name for the image to be creted.
- **file_path** (*str*) – A file path that resolves to the file of the image to be created.
- **storage_container_uuid** (*str*) – The UUID of the storage container on which to place the image.
- **image_type** (*str('disk', 'iso'), optional*) – The type of image to be created. (default=disk).
- **annotation** (*str, optional*) – The annotation to set on the image. (default='').
- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.
- **wait** (*bool, optional*) – Wait for the task to complete. (default=false).

Returns Result of image upload. If Tur the image was created successfully. If False the image creation was unsuccessful

Return type Bool

upload_from_url(*name*, *url*, *storage_container_uuid*, *image_type='disk'*, *annotation=''*, *clusteruuid=None*, *wait=False*)
Upload an image from a URL. The target URL needs to be accessible from the CVM network on the target Nutanix cluster.

Parameters

- **name** (*str*) – A name for the image to be creted.

- **url** (*str*) – A URL that resolves to the file of the image to be created.
- **storage_container_uuid** (*str*) – The UUID of the storage container on which to place the image.
- **image_type** (*str('disk', 'iso'), optional*) – The type of image to be created. (default=disk).
- **annotation** (*str, optional*) – The annotation to set on the image. (default='').
- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.
- **wait** (*bool, optional*) – Wait for the task to complete. (default=false).

Returns Result of image upload. If Tur the image was created successfully. If False the image creation was unsuccessful

Return type Bool

4.2.6 Network

class ntnx_api.prism.**Network**(*api_client*)

A class to represent a Nutanix Cluster AHV Network object.

Parameters **api_client** (`ntnx.client.ApiClient`) – Initialized API client class

create(*name*, *vlan=0*, *vswitch='br0'*, *network_address=None*, *network_cidr=None*, *default_gw=None*, *dhcp_boot_filename=None*, *dhcp_domain_name=None*, *dhcp_domain_nameservers=None*, *dhcp_domain_search=None*, *dhcp_tftp_server_name=None*, *dhcp_server_override=None*, *dhcp_pools=None*, *clusteruuid=None*)

Create a new network on a specified vSwitch

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.
- **name** (*str*) – The name of the network to create.
- **vlan** (*int, optional*) – The vlan id of the network to be created. To select the native VLAN set this value to 0. (default=0)
- **vswitch** (*str, optional*) – The name of the vswitch on which to create the network. (default='br0')
- **network_address** (*str, optional*) – The network address for the network being created. This is required to enable IP Address Management (IPAM) within AHV.
- **network_cidr** (*int, optional*) – The CIDR for the network being created. This is required to enable IP Address Management (IPAM) within AHV.
- **default_gw** (*str, optional*) – The default gateway for the network being created. This is required to enable IP Address Management (IPAM) within AHV.
- **dhcp_boot_filename** (*str, optional*) – The default gateway for the network being created.

- **dhcp_domain_name** (*str, optional*) –
- **dhcp_domain_nameservers** (*str, optional*) –
- **dhcp_domain_search** (*str, optional*) –
- **dhcp_tftp_server_name** (*str, optional*) –
- **dhcp_server_override** (*str, optional*) –
- **dhcp_pools** (*list, optional*) – A list of dicts describing the ip pool ranges to create. Each dict should be in the format {"start": "w.x.y.z", "end": ""w.x.y.z"}.

delete_name (*name*, *clusteruuid=None*)
Delete an existing network by name.

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.
- **name** (*str*) – The name of the network to delete.

delete_uuid (*uuid*, *clusteruuid=None*)
Delete an existing network by uuid.

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.
- **uuid** (*str*) – The uuid of the network to delete.

get (*clusteruuid=None*)
Retrieve data for each network in a specific cluster

Parameters **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns A list of dictionaries describing each network from the specified cluster.

Return type ResponseList

search_name (*name*, *clusteruuid=None*, *refresh=False*)
Retrieve data for a specific network, in a specific cluster by uuid

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.
- **name** (*str, optional*) – A network name to search for.

Returns A dictionary describing the found network.

Return type ResponseDict

search_uuid (*uuid*, *clusteruuid=None*, *refresh=False*)
Retrieve data for a specific network, in a specific cluster by uuid

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.
- **uuid** (*str, optional*) – A network uuid to search for.

Returns A dictionary describing the found network.

Return type ResponseDict

search_vlan(*vlan*, *clusteruuid=None*, *refresh=False*)
Retrieve data for a specific vlan, in a specific cluster by uuid

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.
- **vlan** (*str, optional*) – A vlan to search for.

Returns A dictionary describing the found network.

Return type ResponseDict

update(*name*, *vlan=None*, *network_address=None*, *network_cidr=None*, *default_gw=None*, *dhcp_boot_filename=None*, *dhcp_domain_name=None*, *dhcp_domain_nameservers=None*, *dhcp_domain_search=None*, *dhcp_tftp_server_name=None*, *dhcp_server_override=None*, *dhcp_pools=None*, *clusteruuid=None*)
Update the configuration of an existing network.

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.
- **name** (*str*) – The name of the network to update.
- **vlan** (*int, optional*) – The vlan id of the network to be created. To select the native VLAN set this value to 0. (default=0)
- **network_address** (*str, optional*) – The network address for the network being created. This is required to enable IP Address Management (IPAM) within AHV.
- **network_cidr** (*int, optional*) – The CIDR for the network being created. This is required to enable IP Address Management (IPAM) within AHV.
- **default_gw** (*str, optional*) – The default gateway for the network being created. This is required to enable IP Address Management (IPAM) within AHV.
- **dhcp_boot_filename** (*str, optional*) – The default gateway for the network being created.
- **dhcp_domain_name** (*str, optional*) –
- **dhcp_domain_nameservers** (*str, optional*) –
- **dhcp_domain_search** (*str, optional*) –
- **dhcp_tftp_server_name** (*str, optional*) –
- **dhcp_server_override** (*str, optional*) –

- **dhcp_pools** (*list, optional*) – A list of dicts describing the ip pool ranges to create. Each dict should be in the format {"start": "w.x.y.z", "end": ""w.x.y.z"}.

4.2.7 NetworkSwitch

class ntnx_api.prism.**NetworkSwitch**(*api_client*)
A class to represent a Nutanix Cluster AHV vSwitch object.

Parameters **api_client** (ntnx.client.ApiClient) – Initialized API client class

get(*clusteruuid=None*)
Retrieve data for each vSwitch in a specific AHV cluster

Parameters **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the ntnx.client.ApiClient *connection_type* is set to *pc*.

Returns A list of dictionaries describing each network from the specified cluster.

Return type ResponseList

search_name(*name*, *clusteruuid=None*, *refresh=False*)
Retrieve data for a specific network vSwitch, in a specific cluster by name

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the ntnx.client.ApiClient *connection_type* is set to *pc*.
- **name** (*str, optional*) – A storage pool name to search for.

Returns A dictionary describing the found storage pool.

Return type ResponseDict

4.2.8 StoragePool

class ntnx_api.prism.**StoragePool**(*api_client*)
A class to represent a Nutanix Clusters Storage Pool object.

Parameters **api_client** (ntnx.client.ApiClient) – Initialized API client class

get(*clusteruuid=None*)
Retrieve data for each storage pool in a specific cluster

Parameters **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the ntnx.client.ApiClient *connection_type* is set to *pc*.

Returns A list of dictionaries describing each storage pool from the specified cluster.

Return type ResponseList

search_name(*name*, *clusteruuid=None*, *refresh=False*)
Retrieve data for a specific storage pool, in a specific cluster by uuid

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the ntnx.client.ApiClient *connection_type* is set to *pc*.

- **name** (*str, optional*) – A storage pool name to search for.

Returns A dictionary describing the found storage pool.

Return type ResponseDict

search_uuid(*uuid*, *clusteruuid=None*, *refresh=False*)
Retrieve data for a specific storage pool, in a specific cluster by uuid

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.
- **uuid** (*str, optional*) – A container uuid to search for.

Returns A dictionary describing the found storage pool.

Return type ResponseDict

4.2.9 StorageContainer

class ntnx_api.prism.**StorageContainer**(*api_client*)
A class to represent a Nutanix Clusters Storage Container object.

Parameters **api_client** (`ntnx.client.ApiClient`) – Initialized API client class

create(*name*, *rf=2*, *oplog_rf=2*, *reserved=None*, *advertised=None*, *compression=True*, *compression_delay=0*, *dedupe_cache=False*, *dedupe_capacity=False*, *ecx=False*, *ecx_delay=None*, *whitelist=None*, *storage_pool_uuid=None*, *clusteruuid=None*)
Create a new container in a specific cluster

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.
- **name** (*str*) – The name for the new container. Only unique container names are allowed.
- **rf** (*int, optional*) – The RF level of the container.
- **oplog_rf** (*int, optional*) – The RF level of the container.
- **reserved** (*int, optional*) – The reservation size of the container in bytes.
- **advertised** (*int, optional*) – The advertised size of the container in bytes.
- **compression** (*bool, optional*) – Whether to enable compression.
- **compression_delay** (*int, optional*) – The amount of time in secs before data is compressed Set to 0 for inline compression.
- **dedupe_cache** (*bool, optional*) – Whether to apply deduplication to data in the cache tier.
- **dedupe_capacity** (*bool, optional*) – Whether to apply deduplication to data in the capacity tier.
- **ecx** (*bool, optional*) – Whether to enable erasure coding.
- **ecx_delay** (*int, optional*) – The age of the data in the capacity tier in seconds before erasure coding is applied.

- **whitelist** (*list, optional*) – A list of IPs/subnets to whiteist for access to this container. Used for data migration purposes onle.

Returns *True* if the container was sucessfully created, *False* if creation failed.

Return type bool

delete_name(*name*, *clusteruuid=None*)
Delete a specific container by its name in a specific cluster

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.
- **name** (*str*) – The name for the container.

Returns *True* if the container was successfully updated, *False* if the update failed.

Return type bool

delete_uuid(*uuid*, *clusteruuid=None*)
Delete a specific container by its UUID in a specific cluster

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.
- **uuid** (*str*) – The uuid for the container.

Returns *True* if the container was successfully updated, *False* if the update failed.

Return type bool

get(*clusteruuid=None*)
Retrieve data for each container in a specific cluster

Parameters **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns A list of dictionaries describing each container from the specified cluster.

Return type ResponseList

search_name(*name*, *clusteruuid=None*, *refresh=False*)
Retrieve data for a specific container, in a specific cluster by container uuid

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.
- **name** (*str, optional*) – A container name to search for.

Returns A dictionary describing the found container.

Return type ResponseDict

search_uuid(*uuid*, *clusteruuid=None*, *refresh=False*)
Retrieve data for a specific container, in a specific cluster by container uuid

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.
- **uuid** (*str, optional*) – A container uuid to search for.

Returns A dictionary describing the found container.

Return type ResponseDict

update(*name*, *reserved=None*, *advertised=None*, *compression=True*, *compression_delay=0*, *dedupe_cache=False*, *dedupe_capacity=False*, *ecx=False*, *ecx_delay=None*, *whitelist=None*, *clusteruuid=None*)

Update a specific container in a specific cluster

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.
- **name** (*str*) – The name for the existing container.
- **reserved** (*int, optional*) – The reservation size of the container in bytes.
- **advertised** (*int, optional*) – The advertised size of the container in bytes.
- **compression** (*bool, optional*) – Whether to enable compression.
- **compression_delay** (*int, optional*) – The amount of time in secs before data is compressed Set to 0 for inline compression.
- **dedupe_cache** (*bool, optional*) – Whether to apply deduplication to data in the cache tier.
- **dedupe_capacity** (*bool, optional*) – Whether to apply deduplication to data in the capacity tier.
- **ecx** (*bool, optional*) – Whether to enable erasure coding.
- **ecx_delay** (*int, optional*) – The age of the data in the capacity tier in seconds before erasure coding is applied.
- **whitelist** (*list, optional*) – A list of IPs/subnets to whiteist for access to this container. Used for data migration purposes onle.

Returns *True* if the container was successfully updated, *False* if the update failed.

Return type bool

4.2.10 StorageVolume

class ntnx_api.prism.**StorageVolume**(*api_client*)

A class to represent a Nutanix Clusters Storage Volumes object.

Parameters **api_client** (`ntnx.client.ApiClient`) – Initialized API client class

add_disk_by_volume_group_name(*volume_group_name: str*, *size_gb: int*, *index: int = 0*, *storage_container_uuid: str = None*, *storage_container_name: str = None*, *wait: bool = True*, *clusteruuid=None*)

Add a new disk to an existing volume group using the volume group name.

Parameters

- **volume_group_name** (*str*) – The name of the volume group to which the new disk is to be added.
- **size_gb** (*int*) – The size of the volume group disk to add in GB.
- **index** (*int, optional*) – The SCSI index for the volume group on this VM to be detached.
- **storage_container_uuid** (*str, optional*) – The uuid of the container that this volume group disk will be created on. Either this parameter or 'storage_container_name' must be provided.
- **storage_container_name** (*str, optional*) – The name of the container that this volume group disk will be created on. Either this parameter or 'storage_container_uuid' must be provided.
- **wait** (*bool, optional*) – Wait for the task to complete. (defaults=True)
- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns True or False to indicate whether the disk was successfully added to the volume group.

Return type bool

add_disk_by_volume_group_uuid(*volume_group_uuid: str*, *size_gb: int*, *index: int = 0*, *storage_container_uuid: str = None*, *storage_container_name: str = None*, *wait: bool = True*, *clusteruuid=None*)

Add a new disk to an existing volume group using the volume group uuid.

Parameters

- **volume_group_uuid** (*str*) – The uuid of the volume group to which the new disk is to be added.
- **size_gb** (*int*) – The size of the volume group disk to add in GB.
- **storage_container_uuid** (*str, optional*) – The uuid of the container that this volume group disk will be created on. Either this parameter or 'storage_container_name' must be provided.
- **storage_container_name** (*str, optional*) – The name of the container that this volume group disk will be created on. Either this parameter or 'storage_container_uuid' must be provided.
- **wait** (*bool, optional*) – Wait for the task to complete. (defaults=True)
- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns True or False to indicate whether the disk was successfully added to the volume group.

Return type bool

attach_volume_group(*vm_uuid: str*, *vg_uuid: str*, *index: int = None*, *wait: bool = True*, *clusteruuid: str = None*)

Attach a VM to a volume group.

Parameters

- **vg_uuid** (*str*) – The uuid for the volume group.

- **vm_uuid** (*str*) – The uuid for the vm to be attached to the volume group.
- **index** (*int, optional*) – The SCSI index to attached the volume group onto on the VM.
- **wait** (*bool, optional*) – Wait for the task to complete. (defaults=True)
- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns True or False to indicate whether the volume group was successfully attached.

Return type bool

clone_volume_group_name(*source_name: str*, *dest_name: str*, *load_balancing: bool = True*, *iscsi_chap_password: str = None*, *iscsi_initiators: list = []*, *iscsi_target: str = None*, *wait: bool = True*, *clusteruuid=None*)

Clone a volume group by name.

Parameters

- **source_name** (*str*) – The name for the volume group that is to be cloned.
- **dest_name** (*str*) – The name for the new volume group.
- **load_balancing** (*bool, optional*) – Whether to enable volume group load balancing on this volume group. (defaults=True)
- **iscsi_chap_password** (*str, optional*) – The iscsi CHAP password if wanted for this volume group.
- **iscsi_initiators** (*list, optional*) – A list of the iscsi initiators to be present on this volume group.
- **iscsi_target** (*str, optional*) – The iscsi target for this volume group.
- **wait** (*bool, optional*) – Wait for the task to complete. (defaults=True)
- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns True or False to indicate whether the volume group was successfully cloned.

Return type bool

clone_volume_group_uuid(*source_uuid: str*, *dest_name: str*, *load_balancing: bool = True*, *iscsi_chap_password: str = None*, *iscsi_initiators: list = []*, *iscsi_target: str = None*, *wait: bool = True*, *clusteruuid=None*)

Clone a volume group by name.

Parameters

- **source_uuid** (*str*) – The uuid for the volume group that is to be cloned.
- **dest_name** (*str*) – The name for the new volume group.
- **load_balancing** (*bool, optional*) – Whether to enable volume group load balancing on this volume group. (defaults=True)
- **iscsi_chap_password** (*str, optional*) – The iscsi CHAP password if wanted for this volume group.
- **iscsi_initiators** (*list, optional*) – A list of the iscsi initiators to be present on this volume group.

- **iscsi_target** (*str, optional*) – The iscsi target for this volume group.
- **wait** (*bool, optional*) – Wait for the task to complete. (defaults=True)
- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns True or False to indicate whether the volume group was successfully cloned.

Return type bool

create_volume_group(*name: str*, *description: str = ''*, *flash_mode: bool = False*, *load_balancing: bool = True*, *disks: list = []*, *vms: list = []*, *iscsi_initators: list = []*, *iscsi_target: str = None*, *iscsi_chap_password: str = None*, *wait: bool = True*, *clusteruuid=None*)

Create a new volume group.

Parameters

- **name** (*str*) – A name for the new volume group.
- **description** (*str, optional*) – A description for this volume group.
- **flash_mode** (*bool, optional*) – Whether to enable flash mode on this volume group. (defaults=False)
- **load_balancing** (*bool, optional*) – Whether to enable volume group load balancing on this volume group. (defaults=True)
- **disks** (*list, optional*) – A list of dicts describing the disks to be added to this volume group.

  The dictionary format per-disk is as follows::

  - size_gb (str). The size of the disk in GB.
  - storage_container_name (str). The name of the storage container on which to place the disk. Mutually exclusive with "storage_container_uuid".
  - storage_container_uuid (str). The uuid of the storage container on which to place the disk. Mutually exclusive with "storage_container_name".
  - index (int, optional). The index of the drive within the volume group.

  Examples;

  1. A single disk - [{'size_gb': 50, 'storage_container_name': 'default',},]
  2. Multiple disks - [{'size_gb': 50, 'storage_container_name': 'default',}, {'size_gb': 25, 'storage_container_name': 'default',},]
  3. Multiple disks with specific indexes - [{'size_gb': 50, 'storage_container_name': 'default', 'index': 1}, {'size_gb': 25, 'storage_container_name': 'default', , 'index': 0},]

- **vms** (*list, optional*) – A list of dicts describing the VMs to be attached to this volume group.

  The dictionary format per-vm is as follows::

  - vm_uuid (str). The uuid of the virtual machine.
  - index (str). The scsi index to attach the volume group with one the VM.

  Examples;

1. Attach a single VM - [{'vm_uuid': '95764410-db35-48f8-8cf9-217a8fabb547', 'index': 10, },]
2. Attach a multiple VMs - [{'vm_uuid': '95764410-db35-48f8-8cf9-217a8fabb547', 'index': 10, }, {'vm_uuid': 'e3c543e9-c68c-468b-abd1-61a6d039a84d', 'index': 10, },]

- **iscsi_initators** (*list, optional*) – A list of the iscsi initiators to be present on this volume group.
- **iscsi_target** (*str, optional*) – The iscsi target for this volume group.
- **iscsi_chap_password** (*str, optional*) – The iscsi CHAP password if wanted for this volume group.
- **wait** (*bool, optional*) – Wait for the task to complete. (defaults=True)
- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns True or False to indicate whether the volume group was successfully created.

Return type bool

delete_volume_group_name(*name: str*, *wait: bool = True*, *clusteruuid=None*)
Delete a volume group specified by name.

Parameters

- **name** (*str*) – The name for the volume group to be deleted.
- **wait** (*bool, optional*) – Wait for the task to complete. (defaults=True)
- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns True or False to indicate whether the volume group was successfully deleted.

Return type bool

delete_volume_group_uuid(*uuid: str*, *wait: bool = True*, *clusteruuid=None*)
Delete a volume group specified by uuid.

Parameters

- **uuid** (*str*) – The uuid for the volume group to be deleted.
- **wait** (*bool, optional*) – Wait for the task to complete. (defaults=True)
- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns True or False to indicate whether the volume group was successfully deleted.

Return type bool

detach_volume_group(*vm_uuid: str*, *vg_uuid: str*, *index: int*, *wait: bool = True*, *clusteruuid: str = None*)
Detach a VM to a volume group.

Parameters

- **vg_uuid** (*str*) – The uuid for the volume group.

- **vm_uuid** (*str*) – The uuid for the vm to be detached to the volume group.
- **index** (*int, optional*) – The SCSI index for the volume group on this VM to be detached.
- **wait** (*bool, optional*) – Wait for the task to complete. (defaults=True)
- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns True or False to indicate whether the volume group was successfully detached.

Return type bool

get (*clusteruuid=None*)
Retrieve data for each volume group & all volumes in a specific cluster

Parameters **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

get_volume_groups (*clusteruuid=None*)
Retrieve data for each volume group in a specific cluster

Parameters **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns A list of dictionaries describing each volume group from the specified cluster.

Return type ResponseList

get_volumes (*clusteruuid=None*, *refresh=False*)
Retrieve data for each volume in a specific cluster

Parameters **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns A list of dictionaries describing each volume group from the specified cluster.

Return type ResponseList

remove_disk_by_volume_group_name (*volume_group_name: str*, *index: int*, *wait: bool = True*, *clusteruuid=None*)
Remove a disk from a volume group by volume group name.

Parameters

- **volume_group_name** (*str*) – The name of the volume group to which the new disk is to be added.
- **index** (*int, optional*) – The SCSI index for the volume group on this VM to be detached.
- **wait** (*bool, optional*) – Wait for the task to complete. (defaults=True)
- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns True or False to indicate whether the disk was successfully removed to the volume group.

Return type bool

remove_disk_by_volume_group_uuid(*volume_group_uuid: str*, *index: int*, *wait: bool = True*, *clusteruuid=None*)

Remove a disk from a volume group by volume group uuid.

Parameters

- **volume_group_uuid** (*str*) – The uuid of the volume group to which the new disk is to be added.
- **index** (*int, optional*) – The SCSI index for the volume group on this VM to be detached.
- **wait** (*bool, optional*) – Wait for the task to complete. (defaults=True)
- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns True or False to indicate whether the disk was successfully removed to the volume group.

Return type bool

search_volume_groups_name(*name*, *clusteruuid=None*, *refresh=False*)

Retrieve data for a specific volume group, in a specific cluster by volume group uuid

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.
- **name** (*str, optional*) – A volume group name to search for.

Returns A dictionary describing the found volume group.

Return type ResponseDict

search_volume_groups_uuid(*uuid*, *clusteruuid=None*, *refresh=False*)

Retrieve data for a specific volume group, in a specific cluster by volume group uuid

Parameters

- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.
- **uuid** (*str, optional*) – A volume group uuid to search for.

Returns A dictionary describing the found volume group.

Return type ResponseDict

update_disk_by_volume_group_name(*volume_group_name: str*, *index: int*, *size_gb: int = None*, *preserve_data: bool = True*, *flash_mode: bool = False*, *clusteruuid=None*)

Add a new disk to an existing volume group using the volume group name.

Parameters

- **volume_group_name** (*str*) – The name of the volume group to which the new disk is to be added.
- **size_gb** (*int*) – The size of the volume group disk to add in GB.

- **index** (*int, optional*) – The SCSI index for the volume group on this VM to be detached.
- **preserve_data** (*bool, optional*) – Whether data on this volume should be preserved during the update. (default=True)
- **flash_mode** (*bool, optional*) – Whether to enable flash mode on this volume group. (defaults=False)
- **wait** (*bool, optional*) – Wait for the task to complete. (defaults=True)
- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns True or False to indicate whether the disk was successfully updated on the volume group.

Return type bool

update_disk_by_volume_group_uuid(*volume_group_uuid: str*, *index: int*, *size_gb: int = None*, *preserve_data: bool = True*, *flash_mode: bool = False*, *clusteruuid=None*)

Add a new disk to an existing volume group using the volume group uuid.

Parameters

- **volume_group_uuid** (*str*) – The uuid of the volume group to which the new disk is to be added.
- **size_gb** (*int*) – The size of the volume group disk to add in GB.
- **index** (*int, optional*) – The SCSI index for the volume group on this VM to be detached.
- **preserve_data** (*bool, optional*) – Whether data on this volume should be preserved during the update. (default=True)
- **flash_mode** (*bool, optional*) – Whether to enable flash mode on this volume group. (defaults=False)
- **wait** (*bool, optional*) – Wait for the task to complete. (defaults=True)
- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns True or False to indicate whether the disk was successfully updated on the volume group.

Return type bool

update_volume_group_name(*name: str*, *description: str = None*, *flash_mode: bool = None*, *load_balancing: bool = None*, *iscsi_initiators: list = None*, *iscsi_target: str = None*, *wait: bool = True*, *clusteruuid=None*)

Update the configuration of a volume group specified by name.

Parameters

- **name** (*str*) – The name for the volume group to be updated.
- **description** (*str, optional*) – A description for this volume group.
- **flash_mode** (*bool, optional*) – Whether to enable flash mode on this volume group. (defaults=False)

- **load_balancing** (*bool, optional*) – Whether to enable volume group load balancing on this volume group. (defaults=True)
- **iscsi_initiators** – A list of the iscsi initiators to be present on this volume group.
- **iscsi_target** (*str, optional*) – The iscsi target for this volume group.
- **wait** (*bool, optional*) – Wait for the task to complete. (defaults=True)
- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns True or False to indicate whether the volume group was successfully updated.

Return type bool

update_volume_group_uuid(*uuid: str*, *name: str = None*, *description: str = None*, *flash_mode: bool = None*, *load_balancing: bool = None*, *iscsi_initiators: list = None*, *iscsi_target: str = None*, *wait: bool = True*, *clusteruuid=None*)

Update the configuration of a volume group specified by uuid.

Parameters

- **uuid** (*str*) – The uuid for the volume group to be updated.
- **name** (*str, optional*) – A new name for the volume group to be updated.
- **description** (*str, optional*) – A description for this volume group.
- **flash_mode** (*bool, optional*) – Whether to enable flash mode on this volume group. (defaults=False)
- **load_balancing** (*bool, optional*) – Whether to enable volume group load balancing on this volume group. (defaults=True)
- **iscsi_initiators** (*list, optional*) – A list of the iscsi initiators to be present on this volume group.
- **iscsi_target** (*str, optional*) – The iscsi target for this volume group.
- **wait** (*bool, optional*) – Wait for the task to complete. (defaults=True)
- **clusteruuid** (*str, optional*) – A cluster UUID to define the specific cluster to query. Only required to be used when the `ntnx.client.ApiClient` *connection_type* is set to *pc*.

Returns True or False to indicate whether the volume group was successfully updated.

Return type bool

4.2.11 Prism Central Categories

class ntnx_api.prism.**Categories**(**args*, ***kwargs*)

A class to represent Nutanix Prism Central Categories.

param api_client Initialized API client class

type api_client `ntnx.client.ApiClient`

New in version 1.5.0: This class combines all functions to manage categories into a single class.

assign_category_value(*categories: list*, *uuid: str*, *kind: str = 'vm'*, *wait: bool = True*)
Assign a list of categories & values to a VM, host or cluster.

Parameters

- **categories** (*list*) – A list of of category/value key-pairs.

  The dictionary format for each list item is as follows::

  - category (str). The category to be assigned.
  - value (str). The category value to be assigned.

  Examples;

  1. Add a single category/value - [{'category': 'AppFamily', 'value': 'Databases', },]
  2. Add multiple categories/values - [{'category': 'AppFamily', 'value': 'Databases', }, {'category': 'Environment', 'value': 'Production', },]

- **uuid** (*str*) – UUID of the object to be modified.
- **kind** (*str('vm', 'host', 'cluster'), optional*) – The kind of object to be modified. (default='vm')
- **wait** (*bool, optional*) – Wait for the task to complete. (defaults=True)

Returns *True* or *False* to indicate whether the category was successfully assigned.

Return type bool

Note: Will only return data when *connection_type=='pc'*

---

get_categories(*refresh=False*)
Retrieve data for all categories.

Parameters **refresh** (*str, optional*) – Whether to refresh an existing dataset if it exists.

Returns A list of dictionaries describing all categories.

Return type ResponseList

Note: Will only return data when *connection_type=='pc'*

---

get_category_value_usage(*category*, *value*, *refresh=False*)
Retrieve data for all vms or hosts belonging to a specific category & value.

Parameters

- **category** (*str*) – Category name
- **value** (*str*) – Key name
- **refresh** (*bool, optional*) – Whether to refresh an existing dataset if it exists. (default=False)

Returns

A list of dictionaries containing where the category & value key-pair is in use.

The per-item dictionary format is;

- name (str). The name of the VM found.
- uuid (str). The UUID of the VM found.
- kind (str). The kind of record found. Choice of 'vm', 'host', 'cluster'

Return type ResponseList

Note: Will only return data when *connection_type=='pc'*

---

get_category_values(*category*, *refresh=False*)
Retrieve data for all keys belonging to a specific category.

Parameters

- **category** (*str*) – Category name
- **refresh** (*str, optional*) – Whether to refresh an existing dataset if it exists.

Returns A list of dictionaries describing all category values.

Return type ResponseList

Note: Will only return data when *connection_type=='pc'*

---

remove_category(*name*, *force=False*)
Remove a category

Parameters

- **name** (*str*) – Category name
- **force** (*bool, optional*) – If *True* remove category values and also unassign those values from any VM. (default=False)

Returns *True* or *False* to indicate whether the category removal was successful.

Return type bool

Note: Will only return data when *connection_type=='pc'*

---

remove_category_value(*category: str*, *value: str*, *force=False*)
Remove a category value.

Parameters

- **category** (*str*) – Category name
- **value** (*str*) – Name for new category value
- **force** (*bool, optional*) – If *True* remove category values and also unassign those values from any VM. (default=False)

Returns *True* or *False* to indicate whether the category value removal was successful.

Return type bool

Note: Will only return data when *connection_type=='pc'*

---

search_category(*name*, *refresh=False*)
Search for a specific category.

Parameters

- **name** (*str*) – Category name
- **refresh** (*str, optional*) – Whether to refresh an existing dataset if it exists.

Returns A dictionary describing a single category.

Return type ResponseDict

Note: Will only return data when *connection_type=='pc'*

---

search_category_value(*category*, *value*, *refresh=False*)
Search for a specific key within a category.

Parameters

- **category** (*str*) – Category name
- **value** (*str*) – Category value name
- **refresh** (*str, optional*) – Whether to refresh an existing dataset if it exists.

Returns A dictionary describing a single category value.

Return type ResponseDict

Note: Will only return data when *connection_type=='pc'*

---

set_category(*name*, *description=''*)
Create or update category

Parameters

- **name** (*str*) – Name of new category
- **description** (*str, optional*) – Description for new category

Returns *True* or *False* to indicate whether the category creation or update was successful.

Return type bool

Note: Will only return data when *connection_type=='pc'*

---

set_category_value(*category: str*, *value: str*, *description: str = ''*)
Create or update Category value. If the value does not exist it is created, if it does exist it is updated.

Parameters

- **category** (*str*) – Category name
- **value** (*str*) – Name for new category value
- **description** (*str, optional*) – Description for the new category value

Returns *True* or *False* to indicate whether the category creation or update was successfully.

Return type bool

Note: Will only return data when *connection_type=='pc'*

unassign_category_value(*categories: list*, *uuid: str*, *kind: str = 'vm'*, *wait: bool = True*)
Unassign a list of categories & values from a VM, host or cluster.

Parameters

- **categories** (*list*) – A list of of category/value key-pairs.

  The dictionary format for each list item is as follows::

  - category (str). The category to be assigned.
  - value (str). The category value to be assigned.

  Examples;

  1. Add a single category/value - [{'category': 'AppFamily', 'value': 'Databases', },]
  2. Add multiple categories/values - [{'category': 'AppFamily', 'value': 'Databases', }, {'category': 'Environment', 'value': 'Production', },]

- **uuid** (*str*) – UUID of the object to be modified.
- **kind** (*str('vm', 'host', 'cluster'), optional*) – The kind of object to be modified. (default='vm')
- **wait** (*bool, optional*) – Wait for the task to complete. (defaults=True)

Returns *True* or *False* to indicate whether the category was successfully unassigned.

Return type bool

Note: Will only return data when *connection_type=='pc'*

4.2.12 Prism Central Projects

class ntnx_api.prism.**Projects**(**args*, ***kwargs*)
A class to represent Nutanix Prism Projects.

param api_client Initialized API client class

type api_client ntnx.client.ApiClient

New in version 1.5.0: Added Projects class.

get(*refresh=False*)
Retrieve data for all projects.

Parameters **refresh** (*str, optional*) – Whether to refresh an existing dataset if it exists.

Returns A list of dictionaries describing all projects.

Return type ResponseList

Note: Will only return data when *connection_type=='pc'*

get_usage(*name*, *refresh=False*)
Retrieve a list of the vms that belong to a specific project.

Parameters

- **name** (*str*) – Project name
- **refresh** (*str, optional*) – Whether to refresh an existing dataset if it exists.

Returns

A list of dictionaries containing where the project is in use.

The per-item dictionary format is;

- name (str). The name of the VM found.
- uuid (str). The UUID of the VM found.
- kind (str). The kind of record found. As VMs are only returned by this function this will be 'vm'.

Return type ResponseList

Note: Will only return data when *connection_type=='pc'*

search(*name*, *refresh=False*)
Search for a specific project.

Parameters

- **name** (*str*) – Project name
- **refresh** (*str, optional*) – Whether to refresh an existing dataset if it exists.

Returns A dictionary describing the found project.

Return type ResponseDict

Note: Will only return data when *connection_type=='pc'*

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

CHAPTER 6

Change Log

A log of changes by version and date.

Version	Date	Notes
1.5.0	6/10/2021	Added prism.Categories and prism.Projects to enhance Prism Central category & project management. Updated prism.Tasks.watch_task to also return task status via the v3 API for Prism Central jobs. Redirected original category and project functions from prism.Config to the new Classes. Added global function to search nested dictionaries for key/value presence
1.4.3	6/1/2021	Changed Vm create/update terminology for CPU allocation to aid in transparency. Changed terminology to core/socket rather than vcpu/socket.
1.4.2	6/1/2021	Updated vCPU/socket calculation for Vms.update to match that in Vms.create.
1.4.1	6/1/2021	Updated Images.search_uuid to also check for a match with the uuid of the image disk.
1.4.0	5/20/2021	Added functions to the VMs class for CRUD operations to related to the management of virtual machines. Added function to the VolumeGroups class for CRUD operations to related to the management of volumes and volume groups.
1.3.1	4/14/2021	Added NetworkSwitch.get & NetworkSwitch.search_name to check network bridge configuration. Added Network.get, Network.search_uuid, Network.search_name, Network.search_vlan, Network.create, Network.update, Network.delete_name and Network.delete_uuid to manage AHV networks. Updated StorageContainer CRUD function to only pass updated values.
1.2.4	4/14/2021	Resolved issue with call to StorageContainer.delete_uuid from within StorageContainer.delete_name. Added tests for PC using StorageContainer.create, StorageContainer.update, StorageContainer.delete_name and StorageContainer.delete_uuid.
1.2.3	4/14/2021	Removed comments and extra troubleshooting logging from newly added functions.
1.2.2	4/14/2021	Added StoragePool.get, StoragePool.search_uuid, StoragePool.search_name, StoragePool.get, StorageContainer.create, StorageContainer.update, StorageContainer.delete_name and StorageContainer.delete_uuid.
1.1.30	4/1/2021	Added prism.Vms.get_protection_rules.
1.1.29	3/31/2021	Added prism.Config.get_protection_rules.

Continued on next page

Table 1 – continued from previous page

Version	Date	Notes
1.1.28	3/31/2021	Resolved issue where ProtectionRules were being returned as categories on each VM/Host assigned to the protection rule.
1.1.27	3/31/2021	Resolved issue with if statements within prism.Config.get_projects and prism.Config.get_project_usage.
1.1.26	3/31/2021	Resolved issue with identifying a PC instance within prism.Cluster.get_all_uuids.
1.1.25	3/31/2021	Changed logging level on prism.Cluster.get_all_uuids to info.
1.1.24	3/31/2021	Added debug logging to prism.Cluster.get_all_uuids.
1.1.23	3/30/2021	Fixed issue with if clause in prism.Hosts.get_project and prism.Vms.get_project.
1.1.22	3/30/2021	Fixed variable typo in prism.Hosts.get_project and prism.Hosts.get_categories.
1.1.21	3/30/2021	Fixed flake8 warnings for logging changes.
1.1.21	3/30/2021	Updated logging to use an environment variable NTNX_API_LOG_LEVEL to dictate stdout logging. Default log level is WARNING
1.1.20	3/30/2021	Resolved issues with metadata functions. Renamed variables in vms tests.
1.1.19	3/30/2021	Updated prism.Hosts/Vms.get_project and prism.Hosts/Vmss.get_categories. Added prism.Vm.get_metadata and prism.Host.get_metadata.
1.1.18	3/26/2021	Resolved issue payload dict config with prism.Config.*_categories and prism.Config.*_projects
1.1.17	3/26/2021	Resolved issue with the returned value from prism.Cluster.get
1.1.16	3/26/2021	Changed version import in __init.py__ to be absolute. Modified Vm.get to allow for conditional return of both VM disks and VM nics.
1.1.15	3/25/2021	Fixed inconsistency between master& develop branches
1.1.14	3/25/2021	Added the ability to search for and return host/vm project & categories.
1.1.13	3/23/2021	Resolved code quality issues in client.py
1.1.12	3/23/2021	Updated docsctings within for client.py
1.1.11	3/22/2021	Added docstring for prism.Config.change_ui_admin_password
1.1.10	3/22/2021	Resolved issue with logging text for prism.Config.accept_eula. Added function to change prism admin password.
1.1.9	3/22/2021	Resolved issue with default http code return in ntnx_api.client.PrismApi
1.1.8	3/22/2021	Ignored flake8 check C901
1.1.7	3/22/2021	Separated prism tests into more a more logical structure. Resolved issues with NTP & DNS functions.
1.1.6	3/22/2021	Fixed code quality issues identified by flake8. Also, resolved issue with tox.ini
1.1.5	3/22/2021	Removed logging from requirements.txt
1.1.3	3/19/2021	Re-ordered changelog to improve readability. Added Config.accept_elua
1.1.2	3/19/2021	Fix to ensure that class variables holding data are cleaned up prior to be refreshed.
1.1.1	3/19/2021	Added new PrismApi class to client.py to replace existing ApiClient class. Added console logged for enhanced troubleshooting. Added Image upload from URL & file. Added task monitoring to support image upload completion tracking.
1.0.1	10/20/2020	For all set_* functions in ntnx_api.prism updated the return value to indicate whether a record has been added or updated.
1.0.0	10/19/2020	Release 1.0.0.
0.0.17	10/15/2020	Added SAST to gitlab-ci.yml. Added tests for all added functions.
1.0.0	10/19/2020	Release 1.0.0.
1.0.1	10/20/2020	For all set_* functions in ntnx_api.prism updated the return value to indicate whether a record has been added or updated.
0.0.16	10/15/2020	Added alert configuration to ntnx_api.prism
0.0.15	10/15/2020	Commented out windows gitlab build step as its note required currently.
0.0.14	10/15/2020	Set correct method for ntnx_api.prism.add_local_user and ntnx_api.prism.update_local_user

Continued on next page

Table 1 – continued from previous page

Version	Date	Notes
0.0.13	10/14/2020	Added directory authentication, directory role & local user add/update/delete/set to ntnx_api.prism. Added OS tags to gitlab ci to ensure tasks use correct runners. Resolved issue calling incorrect API for ntnx_api.prism.*_smtp
0.0.12	10/13/2020	Added initial unit tests & gitlab runner in homelab for testing.
0.0.11	10/13/2020	Resolved issue in README.rst causing publication to pypi to fail.
0.0.10	10/13/2020	Included changelog.rst in README.rst. Updated tox.ini to improve test troubleshooting. Improved error messaging to client.NutanixRestHTTPError. Resolved issue for Cluster.get_all_uuids() for connections to Prism Central where the UUIDs were being returned as None. Updated Prism.set_smtp docstring. Renamed Prism.*_auth_directory to Prism.*_auth_dir.
0.0.9	10/12/2020	Updated docstring for auth_directory to include default values
0.0.8	10/12/2020	Updated author email address & added README.rst
0.0.7	10/12/2020	Updated documentation for API glossary to provide headings for each class to improve navigation
0.0.6	10/12/2020	Added auth_type get. Added auth_directory get/add/update/set to ntnx_api.prism. Moved multiple occurances of a dict lookup to a static function for improved code usability.
0.0.5	10/12/2020	Added smtp get/set/update/delete to ntnx_api.prism
0.0.4	10/09/2020	Added pulse get/set/update to ntnx_api.prism
0.0.3	10/09/2020	Added proxy get/add/delete/set to ntnx_api.cluster. Added ui_color, ui_text, ui_banner, ui_2048_game, ui_animation get/set to ntnx_api.prism
0.0.2	10/08/2020	Added ntp & dns get/add/delete/set to ntnx_api.cluster
0.0.1	10/03/2020	Initial Version

CHAPTER 7

Overview

The **Nutanix REST Client** is a python module that simplifies integration with the Nutanix Prism REST interface.

It wraps REST calls with simple APIs and abstracts the HTTP request and response handling. For specifics on API arguments, consult the REST API guide for the Purity release currently running on the target array.

This documentation should be supplemental and attempts to explain installation, basic usage, and provides a glossary of the exposed APIs.

Installation Guide How to get the source code, and how to build or install the python package.

Quick Start Guide A quick start guide for the REST client.

Contributing Guide Process and guidelines for community contributions.

API Glossary A glossary of all exposed REST client APIs.

CHAPTER 8

Community

To learn what other Nutanix customers are doing with the client, to share an implementation or contribute code, or to interact with Nutanix, visit the Nutanix NEXT Community.

CHAPTER 9

Changes

See the *Change Log* for a list of changes to the REST Client.

CHAPTER 10

License

This code is licensed under the simple BSD 2-Clause License.

See the LICENSE.txt file in the top level of the source tree.

CHAPTER 11

Indices and tables

- genindex
- modindex
- search

Python Module Index

n

Index

H

I

N

P

R

S

T

U

V