nexus3-cli Documentation

Thiago Figueiró

Aug 08, 2019

Contents:

CHAPTER 1

Command-line Interface

Logging level can be configured by setting an environment variable named `LOG_LEVEL`. Valid values are: `DEBUG`, `INFO`, `WARNING` (default), `ERROR`, `CRITICAL`.

CHAPTER 2

API

Relevant Nexus 3 API documentation:

- REST API
- Script API

2.1 Examples

Here are some basic operations to get you started. The CLI implementation in `src/nexuscli/cli.py` is another good source of examples.

In all examples below you will need to instantiate a client:

```
>>> import nexuscli
>>> nexus_client = nexuscli.nexus_client.NexusClient()
>>> # update the local list of repositories on the client
>>> nexus_client.repositories.refresh()
>>> # retrieve the list of repositories
>>> repositories = nexus_client.repositories.raw_list()
>>> repositories[0]
{'name': 'maven-snapshots',
 'format': 'maven2',
 'type': 'hosted',
 'url': 'http://localhost:8081/repository/maven-snapshots'}
```

Whenever you see `nexus_client` being used, remember to copy the first two lines of code above as well.

2.1.1 Create a repository

```
>>> r = nexuscli.repository.Repository(
>>>     'hosted',
```

(continues on next page)

(continued from previous page)

```
>>>     name='my-repository',
>>>     format='raw',
>>>     blob_store_name='default',
>>>     strict_content_type_validation=False,
>>>     write_policy='allow',
>>> )
>>> nexus_client.repositories.create(r)
>>> nexus_client.repositories.get_raw_by_name('my-repository')
{'name': 'my-repository',
 'format': 'raw',
 'type': 'hosted',
 'url': 'http://localhost:8081/repository/my-repository'}
```

2.1.2 Delete a repository

```
>>> nexus_client.repositories.delete('my-repository')
```

2.1.3 Upload a file

```
>>> upload_count = nexus_client.upload(
>>>     '/etc/passwd', 'my-repository/etc/passwd')
>>> print(upload_count)
1
```

2.2 Modules

2.2.1 NexusClient

class nexuscli.nexus_client.**NexusClient**(*url=None*, *user=None*, *password=None*, *verify=None*, *config_path=None*)

Bases: object

A class to interact with Nexus 3's API.

Unless all keyword arguments `url`, `user` and `password` are supplied, the class will attempt to read the configuration file and, if unsuccessful, use defaults.

Parameters

- **url** (*str*) – URL to Nexus 3 OSS service. Default: *DEFAULT_URL*.
- **user** (*str*) – login for Nexus service at given url. Default: *DEFAULT_USER*.
- **password** (*str*) – password for given login. Default: *DEFAULT_PASS*.
- **verify** (*bool*) – toggle certificate validation. Default: *DEFAULT_VERIFY*.
- **config_path** (*str*) – local file containing configuration above in JSON format with these keys: `nexus_url`, `nexus_user` and `nexus_pass`. Default: *CONFIG_PATH*.

base_url

as per `url` argument of *NexusClient*.

Type str

config_path
as per `config_path` argument of *NexusClient*.

Type str

CONFIG_PATH = '/home/docs/.nexus-cli'

DEFAULT_PASS = 'admin123'

DEFAULT_URL = 'http://localhost:8081'

DEFAULT_USER = 'admin'

DEFAULT_VERIFY = True

delete(*repository_path*, ***kwargs*)
Delete artefacts, recursively if repository_path is a directory.

Parameters

- **repository_path** – location on the repository service.
- **kwargs** – implementation-specific arguments.

Returns number of deleted files. Negative number for errors.

Return type int

download(*source*, *destination*, ***kwargs*)
Process a download. The source must be a valid Nexus 3 repository path, including the repository name as the first component of the path.

The destination must be a local file name or directory.

If a file name is given as destination, the asset may be renamed. The final destination will depend on self.flatten: when True, the remote path isn't reproduced locally.

Parameters

- **source** – location of artefact or directory on the repository service.
- **destination** – path to the local file or directory.
- **flatten** – when True, the remote path isn't reproduced locally.
- **nocache** – Force download of a directory or artefact even if local copy is available and is up-to-date with the version available on Nexus.

Returns number of downloaded files.

download_file(*download_url*, *destination*)
Download an asset from Nexus artefact repository to local file system.

Parameters

- **download_url** – fully-qualified URL to asset being downloaded.
- **destination** – file or directory location to save downloaded asset. Must be an existing directory; any exiting file in this location will be overwritten.

Returns

list(*repository_path*)
List all the artefacts, recursively, in a given repository_path.

Parameters

- **repository_path** – location on the repository service.

- **kwargs** – implementation-specific arguments.

Returns list of artefacts

Return type list

list_raw(*repository_path*)
As per list but returns a generator of raw Nexus artefact objects

read_config()
Read the configuration settings from the file specified by *config_path* and activates them via *set_config()*.

The configuration file is in JSON format and expects these keys: `nexus_user`, `nexus_pass`, `nexus_url`, `nexus_verify`.

If the configuration file is not found, the default settings will be used instead.

repositories
Instance of *nexuscli.repository.model.RepositoryCollection*. This will automatically use the existing instance of *NexusClient* to communicate with the Nexus service.

rest_url
Full URL to the Nexus REST API, based on *base_url*.

Returns the URL.

scripts
Instance of *nexuscli.script.model.ScriptCollection*. This will automatically use the existing instance of *NexusClient* to communicate with the Nexus service.

set_config(*user*, *password*, *base_url*, *verify*)
Configures the Nexus service credentials and base URL. The credentials are stored in a private class attribute and the base URL in *base_url*.

Every subsequent operation that requires a request to the Nexus service will use this configuration.

Parameters

- **user** – as per `user` argument of *NexusClient*.
- **password** – as per `password` argument of *NexusClient*.
- **base_url** – as per `url` argument of *NexusClient*.
- **verify** – as per `verify` argument of *NexusClient*.

split_component_path(*component_path*)
Splits a given component path into repository, directory, filename.

A Nexus component path for a raw directory must have this format:

`repository_name/directory[(/subdir1)...][/|filename]`

A path ending in `/` means it represents a directory; otherwise it represents a filename.

```
>>> dst0 = 'myrepo0/dir/'
>>> dst1 = 'myrepo1/dir/subdir/'
>>> dst2 = 'myrepo2/dir/subdir/file'
>>> dst3 = 'myrepo3/dir/subdir/etc/file.ext'
>>> split_component_path(dst0)
>>> ('myrepo0', 'dir', None)
>>> split_component_path(dst1)
>>> ('myrepo1', 'dir/subdir', None)
>>> split_component_path(dst2)
```

(continues on next page)

(continued from previous page)

```
>>> ('myrepo2', 'dir/subdir', 'file')
>>> split_component_path(dst3)
>>> ('myrepo3', 'dir/subdir/etc', 'file.ext')
```

Parameters **component_path** (*str*) – the Nexus component path, as described above.

Returns tuple of (repository_name, directory, filename). If the given component_path doesn't represent a file, filename is set to None.

Return type tuple

upload(*source*, *destination*, ***kwargs*)

Process an upload. The source must be either a local file name or directory. The flatten and recurse options are honoured for directory uploads.

The destination must be a valid Nexus 3 repository path, including the repository name as the first component of the path.

Parameters

- **source** – location of file or directory to be uploaded.
- **destination** – destination path in Nexus, including repository name and, if required, directory name (e.g. raw repos require a directory).
- **recurse** – do not process sub directories for uploads to remote
- **flatten** – Flatten directory structure by not reproducing local directory structure remotely

Returns number of files uploaded.

upload_directory(*src_dir*, *dst_repo*, *dst_dir*, ***kwargs*)

Uploads all files in a directory, honouring options flatten and recurse.

Parameters

- **src_dir** – path to local directory to be uploaded
- **dst_repo** – destination repository
- **dst_dir** – destination directory in dst_repo

Returns number of files uploaded

Return type int

upload_file(*src_file*, *dst_repo*, *dst_dir*, *dst_file=None*)

Uploads a singe file to a Nexus repository under the directory and file name specified. If the destination file name isn't given, the source file name is used.

Parameters

- **src_file** – path to the local file to be uploaded.
- **dst_repo** – name of the Nexus repository.
- **dst_dir** – directory under dst_repo to place file in.
- **dst_file** – destination file name.

write_config()

Writes the latest configuration set using *set_config()* to disk under *config_path*.

If a file already exists, it will be overwritten. The permission will be set to read/write to the owner only.

2.2.2 Exception

exception nexuscli.exception.**DownloadError**
Bases: Exception

Error retrieving artefact from Nexus service.

exception nexuscli.exception.**NexusClientAPIError**
Bases: Exception

Unexpected response from Nexus service.

exception nexuscli.exception.**NexusClientCreateRepositoryError**
Bases: Exception

Used when a repository creation operation in Nexus fails.

exception nexuscli.exception.**NexusClientInvalidCredentials**
Bases: Exception

Login credentials not accepted by Nexus service. Usually the result of a HTTP 401 response.

exception nexuscli.exception.**NexusClientInvalidRepository**
Bases: Exception

The given repository does not exist in Nexus.

exception nexuscli.exception.**NexusClientInvalidRepositoryPath**
Bases: Exception

Used when an operation against the Nexus service uses an invalid or non-existent path.

2.2.3 Repository

class nexuscli.repository.model.**Repository**(*repo_type*, ***kwargs*)
Bases: object

Creates an object representing a Nexus repository with the given format, type and attributes.

Parameters

- **name** (*str*) – name of the new repository.
- **format** (*str*) – format (recipe) of the new repository. Must be one of `nexuscli.repository.validations.KNOWN_FORMATS`.
- **blob_store_name** (*str*) – an existing blob store; 'default' should work on most installations.
- **depth** (*int*) – only accepted when `repo_format='yum'`. The Yum repodata depth. Usually 1.
- **remote_url** (*str*) – only accepted when `repo_type='proxy'`. The URL of the repository being proxied, including the protocol scheme.
- **strict_content_type_validation** (*bool*) – only accepted when `repo_type='hosted'`. Whether to validate file extension against its content type.

- **version_policy** (*str*) – only accepted when `repo_type='hosted'`. Must be one of `nexuscli.repository.validations.VERSION_POLICIES`.
- **write_policy** (*str*) – only accepted when `repo_type='hosted'`. Must be one of `nexuscli.repository.validations.WRITE_POLICIES`.
- **layout_policy** (*str*) – only accepted when `format='maven'`. Must be one of `nexuscli.repository.validations.LAYOUT_POLICIES`.
- **ignore_extra_kwargs** (*bool*) – if True, do not raise an exception for unnecessary/extra/invalid kwargs.
- **repo_type** – type for the new repository. Must be one of `nexuscli.repository.validations.KNOWN_TYPES`.
- **kwargs** – attributes for the new repository.

Returns a Repository instance with the given settings

Return type *Repository*

configuration

Validate the configuration for the Repository and build its representation as a python dict. The dict returned by this property can be converted to JSON for use with the `nexus3-cli-repository-create` groovy script returned by *nexuscli.repository.groovy.script_create_repo()*.

Example structure and attributes common to all repositories:

```
>>> common_configuration = {
>>>     'name': 'my-repository',
>>>     'online': True,
>>>     'recipeName': 'raw',
>>>     '_state': 'present',
>>>     'attributes': {
>>>         'storage': {
>>>             'blobStoreName': 'default',
>>>         }
>>>     }
>>> }
```

Depending on the repository type and format (recipe), other attributes will be present.

Returns repository configuration

Return type dict

class nexuscli.repository.model.**RepositoryCollection**(*client=None*)

Bases: object

A class to manage Nexus 3 repositories.

Parameters **client** (nexuscli.nexus_client.NexusClient) – the client instance that will be used to perform operations against the Nexus 3 service. You must provide this at instantiation or set it before calling any methods that require connectivity to Nexus.

client

as per `client` argument of *RepositoryCollection*.

Type *nexuscli.nexus_client.NexusClient*

create(*repository*)

Creates a Nexus repository with the given format and type.

Parameters **repository** (Repository) –

Returns None

delete(*name*)
Delete a repository.

Parameters **name** – name of the repository to be deleted.

get_raw_by_name(*name*)
Return the raw dict for the repository called `name`. Remember to *refresh()* to get the latest from the server.

Parameters **name** (*str*) – name of wanted repository

Returns the repository, if found.

Return type dict

Raises **IndexError** – if no repository named `name` is found.

raw_list()
A raw representation of the Nexus repositories.

Returns for the format, see List Repositories.

Return type dict

refresh()
Refresh local list of repositories with latest from service. A raw representation of repositories can be fetched using *raw_list()*.

nexuscli.repository.groovy.**script_create_repo**()
A groovy script that can be used to create repositories in Nexus 3.

Returns the groovy script

Return type str

nexuscli.repository.validations.**is_target_supported**(*target*, *value*, *known*, *supported*)
Validate whether the a target argument is known and supported. The `target` is only used to provide a friendlier message to the user. The given `value` is checked against `known` and `supported`.

Parameters

- **target** (*str*) – name of the target, as known to the end-user.
- **value** (*str*) – value of the target key.
- **known** (*list*) – known possible values for the target.
- **supported** (*list*) – values for the target supported by us.

Raises

- **ValueError** – if given value is not in `known`.
- **NotImplementedError** – if given value is not in `supported`.

nexuscli.repository.validations.**repository_args**(*repo_type*, ***kwargs*)
Validate that the combination of arguments for a `nexuscli.repository.Repository` is valid.

Raises

- **ValueError** – If the value of a given argument is invalid or unsupported, or if unrecognised keyword arguments are given.
- **TypeError** – If the type of a given argument has the wrong object type.
- **NotImplementedError** – If the combination of arguments isn't yet supported.

Parameters

- **repo_type** – as given to `nexuscli.nexus_repository.Repository.create()`
- **kwargs** – as given to `nexuscli.nexus_repository.Repository.create()`

`nexuscli.repository.validations.`**upcase_policy_args**(*args*)

Forces upper-case on the value of the `layout_policy`, `write_policy` and `version_policy` keys of a kwargs dict.

Parameters **args** (*dict*) – kwargs given to caller

Returns a copy of the original dict with the updated values.

Return type dict

2.2.4 Script

class `nexuscli.script.model.`**Script**

Bases: `object`

A Class representing a Nexus 3 script.

class `nexuscli.script.model.`**ScriptCollection**(*client=None*)

Bases: `object`

A class to manage Nexus 3 scripts.

Parameters **client** (`nexuscli.nexus_client.NexusClient`) – the client instance that will be used to perform operations against the Nexus 3 service. You must provide this at instantiation or set it before calling any methods that require connectivity to Nexus.

client

as per `client` argument of *ScriptCollection*.

Type *nexuscli.nexus_client.NexusClient*

create(*script_dict*)

Create the given script in the Nexus 3 service.

Parameters **script_dict** (*dict*) – instance of script to be created.

Raises ***exception.NexusClientAPIError*** – if the script creation isn't successful; i.e.: any HTTP code other than 204.

create_if_missing(*script_dict*)

Creates a script in the Nexus 3 service IFF a script with the same name doesn't exist. Equivalent to checking if the script exists with *get()* and, if not, creating it with *create()*.

Parameters **script_dict** (*dict*) – instance of script to be created.

delete(*script_name*)

Deletes a script from the Nexus 3 repository.

Parameters **script_name** – name of script to be deleted.

Raises ***exception.NexusClientAPIError*** – if the Nexus service fails to delete the script; i.e.: any HTTP code other than 204.

get(*name*)

Get a Nexus 3 script by name.

Parameters **name** – of script to be retrieved.

Returns the script or None, if not found

Return type dict, None

Raises ***exception.NexusClientAPIError*** – if the response from the Nexus service isn't recognised; i.e.: any HTTP code other than 200, 404.

list()
List of all script names on the Nexus 3 service.

Returns a list of names

Return type list

Raises ***exception.NexusClientAPIError*** – if the script names cannot be retrieved; i.e.: any HTTP code other than 200.

run(*script_name*, *data=''*)
Runs an existing script on the Nexus 3 service.

Parameters

- **script_name** – name of script to be run.
- **data** – parameters to be passed to the script, via HTTP POST. If the script being run requires a certain format or encoding, you need to prepare it yourself.

Returns the content returned by the script, if any.

Return type str, dict

Raises ***exception.NexusClientAPIError*** – if the Nexus service fails to run the script; i.e.: any HTTP code other than 200.

2.2.5 Util

nexuscli.nexus_util.**calculate_hash**(*hash_name*, *file_path_or_handle*)
Calculate a hash for the given file.

Parameters

- **hash_name** (*str*) – name of the hash algorithm in hashlib
- **file_path_or_handle** (*Union[str, file object]*) – source file name (str) or file handle (from open()) for the hash algorithm.

Returns the calculated hash

Return type str

nexuscli.nexus_util.**filtered_list_gen**(*raw_response*, *term=None*, *partial_match=True*)

Iterates over items yielded by raw_response_gen, validating that:

1. the *path* dict key is a str
2. the *path* value starts with starts_with (if provided)

```
>>> r = [{
>>>     'checksum': {
>>>         'md5': 'd94b865aa7620c46ef8faef7059a311c',
>>>         'sha1': '2186934d880cf24dd9ecc578335e290026695522',
>>>         'sha256': 'b7bb3424a6a6(...)4113bc38fd7807528481a8ffe3cf',
>>>         'sha512': 'e7806f3caa3e(...)3caeb9bbc54bbde286c07f837fdc'
```

(continues on next page)

(continued from previous page)

```
>>>     },
>>>     'downloadUrl': 'http://nexus/repository/repo_name/a/file.ext',
>>>     'format': 'yum',
>>>     'id': 'Y2xvdWRlcmEtbWFuYWdlcj(...)mRiNWU0YjllZWQzMg',
>>>     'path': 'a/fake.rpm',
>>>     'repository': 'cloudera-manager'}]
>>>
>>> for i in filtered_list_gen(r, starts_with='a/fake.rpm')
>>>     print(i['path'])
a/fake.rpm
>>> for i in filtered_list_gen(r, starts_with='b')
>>>     print(i['path'])
# (nothing printed)
```

Parameters

- **raw_response** (*iterable*) – an iterable that yields one element of a nexus search response at a time, such as the one returned by _paginate_get().
- **term** (*str*) – if defined, only items with an attribute *path* that starts with the given parameter are returned.
- **partial_match** (*bool*) – if True, include items whose artefact path starts with the given term.

Yields *dict* – items that matched the filter.

nexuscli.nexus_util.**validate_strings**(**args*)

Checks that all given arguments have a string type (e.g. str, basestring, unicode etc)

Parameters ***args** – values to be validated.

Returns True if all arguments are of a string type. False otherwise.

Return type bool

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

n

Index

R

S

U

V

W