
repositorytools Documentation

Release 0.0.0

Michel Samia

Jul 13, 2017

Contents

1	Python API and command-line interface for working with Sonatype Nexus	3
1.1	How to install	3
1.2	Some command line examples	3
1.3	Some library examples	4
1.4	Documentation	5
1.5	Support	5
2	Credits	7
2.1	Development Lead	7
2.2	Contributors	7
3	How to run commands locally	9
4	How to run tests locally	11
5	repositorytools	13
5.1	repositorytools package	13
6	Indices and tables	19
	Python Module Index	21

Contents:

Python API and command-line interface for working with Sonatype Nexus

How to install

```
pip install repositorytools
```

Some command line examples

Preparing env. variables

```
export REPOSITORY_URL=https://repo.example.com
export REPOSITORY_USER=admin
export REPOSITORY_PASSWORD=mysecretpassword
```

Uploading an artifact

```
artifact upload foo-1.2.3.ext releases com.fooware
```

Resolving artifact's URL

```
artifact resolve com.fooware:foo:latest
```

Deleting artifacts

```
# by url
artifact delete https://repo.example.com/content/repositories/releases/com/fooware/
↳foo/1.2.3/foo-1.2.3.ext

# by coordinates
artifact resolve com.fooware:foo:latest | xargs artifact delete
```

Working with staging repositories

Nexus Professional only

```
repo create -h
repo close -h
repo release -h
repo drop -h
repo list -h
```

Working with custom maven metadata

Nexus Professional only

```
artifact get-metadata -h
artifact set-metadata -h
```

Some library examples

For most of methods the same env. variables as above have to be exported or specified in call of repository_client_factory()

Uploading artifacts

```
import repositorytools

artifact = repositorytools.LocalArtifact(local_path='~/foo-1.2.3.jar', group='com.
↳fooware')
client = repositorytools.repository_client_factory(user='admin', password=
↳'myS3cr3tPasswOrd')
remote_artifacts = client.upload_artifacts(local_artifacts=[artifact], repo_id=
↳'releases')
print(remote_artifacts)
```

Resolving artifacts

Works even without authentication.

```
import repositorytools

artifact = repositorytools.RemoteArtifact.from_repo_id_and_coordinates('test', 'com.
↳fooware:foo:1.2.3')
```



```
client = repositorytools.repository_client_factory()
client.resolve_artifact(artifact)
print(artifact.url)
```

Deleting artifacts

```
import repositorytools

artifact = repositorytools.RemoteArtifact.from_repo_id_and_coordinates('test', 'com.
↳fooware:foo:1.2.3')
client = repositorytools.repository_client_factory(user='admin', password=
↳'myS3cr3tPasswOrd')
client.resolve_artifact(artifact)
client.delete_artifact(artifact.url)
```

Documentation

is on <http://repositorytools.readthedocs.org/en/latest/>

Support

You can support my effort many ways:

- create issues
- fix issues (by sending pull requests)
- donating on <https://gratipay.com/~stardust85/>, you can send even 1 cent per month ;) (but the more the better)

CHAPTER 2

Credits

Development Lead

- Michel SAMIA <stardust1985@gmail.com>

Contributors

- Laurent LAPORTE <tantale.solutions@gmail.com>

CHAPTER 3

How to run commands locally

Activate the virtualenv and run the script

```
{artifact,repo}
```


CHAPTER 4

How to run tests locally

Activate the virtualenv, go into the project directory and run:

```
nosetests -v
```

Alternatively:

1. Install pycharm >=3.4.1
2. Install virtualbox and vagrant
3. Tools -> Vagrant -> Up
4. Settings -> Interpreters -> add -> remote -> vagrant
5. Run -> Edit Configurations -> add variables `REPOSITORY_PASSWORD` (nexus admin password) and `REPOSITORY_URL`. The nexus instance has to be professional for some tests.
6. Shift F10

repositorytools package

Subpackages

repositorytools.cli package

Command line interface

Subpackages

repositorytools.cli.commands package

Submodules

repositorytools.cli.commands.artifact module

```
class repositorytools.cli.commands.artifact.ArtifactCLI
    Bases: repositorytools.cli.common.CLI
    delete (args)
    get_metadata (args)
    resolve (args)
    set_metadata (args)
    upload (args)
```

repositorytools.cli.commands.repo module

```
class repositorytools.cli.commands.repo.RepoCLI
    Bases: repositorytools.cli.common.CLI
    close (args)
    create (args)
        Parameters args –
        Returns staging repository id as string
    drop (args)
    list (args)
    release (args)
```

Submodules

repositorytools.cli.common module

```
class repositorytools.cli.common.CLI
    Bases: _abcoll.Callable
    Base class for cli
    run (args=None)
repositorytools.cli.common.configure_logging (debug)
```

repositorytools.lib package

Submodules

repositorytools.lib.artifact module

```
exception repositorytools.lib.artifact.NameVerDetectionError
    Bases: repositorytools.lib.artifact.ArtifactError
class repositorytools.lib.artifact.Artifact (group, artifact='', version='', classifier='', extension='')
    Bases: object
    Generic class describing an artifact
    get_coordinates_string ()
class repositorytools.lib.artifact.LocalArtifact (group, local_path, artifact='', version='', classifier='', extension='')
    Bases: repositorytools.lib.artifact.Artifact
    Artifact for upload to repository
    detect_name_ver_ext ()
```

```
class repositorytools.lib.artifact.LocalRpmArtifact (local_path, group=None)
    Bases: repositorytools.lib.artifact.LocalArtifact

    Special case of local artifact, which can detect it's coordinates from RPM metadata

    static get_artifact_group (url)

class repositorytools.lib.artifact.RemoteArtifact (group=None, artifact='', version='',
                                                    classifier='', extension='', url=None,
                                                    repo_id=None)
    Bases: repositorytools.lib.artifact.Artifact

    Artifact in repository

    classmethod from_repo_id_and_coordinates (repo_id, coordinates)

        Parameters
        • repo_id –
        • coordinates – e.g. 'com.foo:foo:1.0.0'

        Returns
```

repositorytools.lib.repository module

Contains classes for manipulating with a repository server

```
exception repositorytools.lib.repository.RepositoryClientError
    Bases: exceptions.Exception

    Base exception raised when working with NexusRepositoryClient and its descendants

exception repositorytools.lib.repository.WrongDataTypeError
    Bases: repositorytools.lib.repository.RepositoryClientError

exception repositorytools.lib.repository.ArtifactNotFoundError
    Bases: repositorytools.lib.repository.RepositoryClientError

class repositorytools.lib.repository.NexusRepositoryClient (repository_url=None,
                                                            user=None,          pass-
                                                            word=None,          ver-
                                                            ify_ssl=True)

    Bases: object

    Class for working with Sonatype Nexus OSS

    DEFAULT_REPOSITORY_URL = 'https://repository'

    delete_artifact (url)
        Deletes an artifact from repository.

        Parameters url – string

        Returns

    resolve_artifact (remote_artifact)

    upload_artifacts (local_artifacts,      repo_id,      print_created_artifacts=True,      _host-
                    name_for_download=None,      _path_prefix='content/repositories',
                    use_direct_put=False)

        Uploads artifacts to repository.

        Parameters
```

- **local_artifacts** – list[LocalArtifact]
- **repo_id** – id of target repository
- **print_created_artifacts** – if True prints to stdout what was uploaded and where

Returns list[RemoteArtifact]

class repositorytools.lib.repository.**NexusProRepositoryClient** (*repository_url=None*,
user=None, pass-word=None, verify_ssl=True, staging_repository_url=None)

Bases: *repositorytools.lib.repository.NexusRepositoryClient*

Class for working with Sonatype Nexus Professional

close_staging_repo (*repo_id, description=''*)

Closes a staging repository. After close, no files can be added.

Parameters

- **repo_id** – id of staging repository
- **description** – if specified, updates description of staged repository

Returns

close_staging_repos (*repo_ids, description=''*)

Closes multiple staging repositories.

Parameters

- **repo_ids** – list of repo IDs (strings) For description of other params see `close_staging_repo`.
- **description** – Description message.

Returns

create_staging_repo (*profile_name, description*)

Creates a staging repository :param profile_name: name of staging profile :param description: description of created staging repository :return: id of newly created staging repository

drop_staging_repo (*repo_id, description='No description'*)

Deletes a staging repository and all artifacts inside.

Parameters **repo_id** – id of staging repository

Returns

drop_staging_repos (*repo_ids, description='No description'*)

Deletes multiple staging repositories.

Parameters **repo_ids** – list of repo IDs (strings)

Returns

get_artifact_metadata (*remote_artifact*)

Gets artifact's maven metadata.

Metadata capability needs to be enabled to use this. Also indexing has to be enabled for that repo to make it work.

Parameters **remote_artifact** –

Returns

list_staging_repos (*filter_dict=None*)

Parameters **filter_dict** – dictionary with filters, for example {'description': 'foo'}

Returns list of dictionaries, each dict describes one staging repo

release_staging_repo (*repo_id, description='No description', auto_drop_after_release=True, keep_metadata=False*)

Releases all contents of a staging repository to a release repository which this staging repository targets.

Parameters

- **repo_id** – id of staging repository
- **description** –
- **auto_drop_after_release** – set this to True if you want to delete the staging repository after releasing
- **keep_metadata** – Keeps custom maven metadata of artifacts after release. Works only there is list of artifacts created by `upload_artifacts_to_new_staging` with `upload_filelist=False`. It is because current Nexus 2.x can't do keep the metadata after release, so we manually read the metadata, release and then set them again.

Returns

set_artifact_metadata (*remote_artifact, metadata*)

Sets artifact metadata.

The same requirements as for `get_artifact_metadata` have to be met.

Parameters

- **remote_artifact** –
- **metadata** – dict of keys and values you want to save there

Returns

upload_artifacts_to_new_staging (*local_artifacts, repo_id, print_created_artifacts=True, description='No description', upload_filelist=False*)

Creates a staging repository in staging profile with name `repo_id` and uploads `local_artifacts` there.

Parameters

- **local_artifacts** – list[LocalArtifact]
- **repo_id** – name of target repository
- **print_created_artifacts** – if True prints to stdout what was uploaded and where
- **description** – description of staging repo
- **upload_filelist** – see `upload_artifacts_to_staging`

Returns list[RemoteArtifact]

upload_artifacts_to_staging (*local_artifacts, repo_id, print_created_artifacts=True, upload_filelist=False*)

Parameters

- **local_artifacts** – list[LocalArtifact]
- **repo_id** – name of staging repository
- **print_created_artifacts** – if True prints to stdout what was uploaded and where
- **staging** – bool

- **upload_filelist** – if True, creates and uploads a list of uploaded files

Returns list[RemoteArtifact]

`repositorytools.lib.repository.repository_client_factory(*args, **kwargs)`

Detects which kind of repository user wants to use and returns appropriate instance of it.

Parameters

- **args** –
- **kwargs** –

Returns

CHAPTER 6

Indices and tables

- `genindex`
- `modindex`
- `search`

r

- `repositorytools`, [13](#)
- `repositorytools.cli`, [13](#)
- `repositorytools.cli.commands`, [13](#)
- `repositorytools.cli.commands.artifact`,
[13](#)
- `repositorytools.cli.commands.repo`, [14](#)
- `repositorytools.cli.common`, [14](#)
- `repositorytools.lib`, [14](#)
- `repositorytools.lib.artifact`, [14](#)
- `repositorytools.lib.repository`, [15](#)

A

Artifact (class in repositorytools.lib.artifact), 14
 ArtifactCLI (class in repositorytools.cli.commands.artifact), 13
 ArtifactNotFoundError, 15

C

CLI (class in repositorytools.cli.common), 14
 close() (repositorytools.cli.commands.repo.RepoCLI method), 14
 close_staging_repo() (repositorytools.lib.repository.NexusProRepositoryClient method), 16
 close_staging_repos() (repositorytools.lib.repository.NexusProRepositoryClient method), 16
 configure_logging() (in module repositorytools.cli.common), 14
 create() (repositorytools.cli.commands.repo.RepoCLI method), 14
 create_staging_repo() (repositorytools.lib.repository.NexusProRepositoryClient method), 16

D

DEFAULT_REPOSITORY_URL (repositorytools.lib.repository.NexusRepositoryClient attribute), 15
 delete() (repositorytools.cli.commands.artifact.ArtifactCLI method), 13
 delete_artifact() (repositorytools.lib.repository.NexusRepositoryClient method), 15
 detect_name_ver_ext() (repositorytools.lib.artifact.LocalArtifact method), 14
 drop() (repositorytools.cli.commands.repo.RepoCLI method), 14
 drop_staging_repo() (repositorytools.lib.repository.NexusProRepositoryClient method), 16

drop_staging_repos() (repositorytools.lib.repository.NexusProRepositoryClient method), 16

F

from_repo_id_and_coordinates() (repositorytools.lib.artifact.RemoteArtifact class method), 15

G

get_artifact_group() (repositorytools.lib.artifact.LocalRpmArtifact static method), 15
 get_artifact_metadata() (repositorytools.lib.repository.NexusProRepositoryClient method), 16
 get_coordinates_string() (repositorytools.lib.artifact.Artifact method), 14
 get_metadata() (repositorytools.cli.commands.artifact.ArtifactCLI method), 13

L

list() (repositorytools.cli.commands.repo.RepoCLI method), 14
 list_staging_repos() (repositorytools.lib.repository.NexusProRepositoryClient method), 16
 LocalArtifact (class in repositorytools.lib.artifact), 14
 LocalRpmArtifact (class in repositorytools.lib.artifact), 14

N

NameVerDetectionError, 14
 NexusProRepositoryClient (class in repositorytools.lib.repository), 16
 NexusRepositoryClient (class in repositorytools.lib.repository), 15

R

`release()` (`repositorytools.cli.commands.repo.RepoCLI` method), 14

`release_staging_repo()` (`repositorytools.lib.repository.NexusProRepositoryClient` method), 17

`RemoteArtifact` (class in `repositorytools.lib.artifact`), 15

`RepoCLI` (class in `repositorytools.cli.commands.repo`), 14

`repository_client_factory()` (in module `repositorytools.lib.repository`), 18

`RepositoryClientError`, 15

`repositorytools` (module), 13

`repositorytools.cli` (module), 13

`repositorytools.cli.commands` (module), 13

`repositorytools.cli.commands.artifact` (module), 13

`repositorytools.cli.commands.repo` (module), 14

`repositorytools.cli.common` (module), 14

`repositorytools.lib` (module), 14

`repositorytools.lib.artifact` (module), 14

`repositorytools.lib.repository` (module), 15

`resolve()` (`repositorytools.cli.commands.artifact.ArtifactCLI` method), 13

`resolve_artifact()` (`repositorytools.lib.repository.NexusRepositoryClient` method), 15

`run()` (`repositorytools.cli.common.CLI` method), 14

S

`set_artifact_metadata()` (`repositorytools.lib.repository.NexusProRepositoryClient` method), 17

`set_metadata()` (`repositorytools.cli.commands.artifact.ArtifactCLI` method), 13

U

`upload()` (`repositorytools.cli.commands.artifact.ArtifactCLI` method), 13

`upload_artifacts()` (`repositorytools.lib.repository.NexusRepositoryClient` method), 15

`upload_artifacts_to_new_staging()` (`repositorytools.lib.repository.NexusProRepositoryClient` method), 17

`upload_artifacts_to_staging()` (`repositorytools.lib.repository.NexusProRepositoryClient` method), 17

W

`WrongDataTypeError`, 15