
Twine Documentation

Release 5.0.1.dev78+g5d17a43

Donald Stufft and individual contributors

May 02, 2024

CONTENTS

1	Twine 5.0.0 (2024-02-10)	3
1.1	Bugfixes	3
1.2	Misc	3
2	Twine 4.0.2 (2022-11-30)	5
2.1	Bugfixes	5
3	Twine 4.0.1 (2022-06-01)	7
3.1	Bugfixes	7
4	Twine 4.0.0 (2022-03-31)	9
4.1	Features	9
4.2	Bugfixes	9
5	Twine 3.8.0 (2022-02-02)	11
5.1	Features	11
5.2	Bugfixes	11
6	Twine 3.7.1 (2021-12-07)	13
6.1	Improved Documentation	13
7	Twine 3.7.0 (2021-12-01)	15
7.1	Features	15
8	Twine 3.6.0 (2021-11-10)	17
8.1	Features	17
9	Twine 3.5.0 (2021-11-02)	19
9.1	Features	19
9.2	Bugfixes	19
10	Twine 3.4.2 (2021-07-20)	21
10.1	Bugfixes	21
11	Twine 3.4.1 (2021-03-16)	23
11.1	Bugfixes	23
12	Twine 3.4.0 (2021-03-15)	25
12.1	Features	25
13	Twine 3.3.0 (2020-12-23)	27
13.1	Features	27

13.2	Bugfixes	27
13.3	Improved Documentation	27
14	Twine 3.2.0 (2020-06-24)	29
14.1	Features	29
14.2	Bugfixes	29
15	Twine 3.1.1 (2019-11-27)	31
15.1	Bugfixes	31
16	Twine 3.1.0 (2019-11-23)	33
16.1	Features	33
17	Twine 3.0.0 (2019-11-18)	35
17.1	Features	35
17.2	Bugfixes	35
18	Twine 2.0.0 (2019-09-24)	37
18.1	Features	37
18.2	Bugfixes	37
19	Twine 1.15.0 (2019-09-17)	39
19.1	Features	39
20	Twine 1.14.0 (2019-09-06)	41
20.1	Features	41
20.2	Bugfixes	41
21	Twine 1.13.0 (2019-02-13)	43
21.1	Features	43
21.2	Bugfixes	43
21.3	Misc	44
22	Twine 1.12.1 (2018-09-24)	45
22.1	Bugfixes	45
23	Twine 1.12.0 (2018-09-24)	47
23.1	Features	47
23.2	Bugfixes	47
24	Twine 1.11.0 (2018-03-19)	49
24.1	Features	49
24.2	Bugfixes	49
24.3	Misc	49
25	Twine 1.10.0 (2018-03-07)	51
25.1	Features	51
25.2	Bugfixes	51
26	Twine 1.9.1 (2017-05-27)	53
26.1	Bugfixes	53
27	Twine 1.9.0 (2017-05-22)	55
27.1	Bugfixes	55
27.2	Misc	55

28 Twine 1.8.1 (2016-08-09)	57
28.1 Misc	57
29 Twine 1.8.0 (2016-08-08)	59
29.1 Features	59
29.2 Misc	59
30 Twine 1.7.4 (2016-07-09)	61
30.1 Bugfixes	61
31 Twine 1.7.3 (2016-07-08)	63
31.1 Bugfixes	63
31.2 Misc	63
32 Twine 1.7.2 (2016-07-05)	65
32.1 Bugfixes	65
33 Twine 1.7.1 (2016-07-05)	67
33.1 Bugfixes	67
34 Twine 1.7.0 (2016-07-04)	69
34.1 Features	69
34.2 Bugfixes	69
35 Twine 1.6.5 (2015-12-16)	71
35.1 Bugfixes	71
36 Twine 1.6.4 (2015-10-27)	73
36.1 Bugfixes	73
37 Twine 1.6.3 (2015-10-05)	75
37.1 Bugfixes	75
38 Twine 1.6.2 (2015-09-28)	77
38.1 Bugfixes	77
39 Twine 1.6.1 (2015-09-18)	79
39.1 Bugfixes	79
40 Twine 1.6.0 (2015-09-14)	81
40.1 Features	81
40.2 Bugfixes	81
41 Twine 1.5.0 (2015-03-10)	83
41.1 Features	83
41.2 Bugfixes	83
41.3 Misc	83
42 Twine 1.4.0 (2014-12-12)	85
42.1 Features	85
42.2 Bugfixes	85
43 Twine 1.3.0 (2014-03-31)	87
43.1 Features	87
44 Twine 1.2.2 (2013-10-03)	89
44.1 Features	89

45 Contributing	91
45.1 Getting started	91
45.1.1 Building the documentation	92
45.1.2 Code style	92
45.1.3 Testing	92
45.2 Submitting changes	93
45.2.1 Changelog entries	93
45.3 Architectural overview	93
45.3.1 twine package	94
45.3.2 Where Twine gets configuration and credentials	104
45.4 Adding a maintainer	104
45.5 Making a new release	104
45.6 Future development	105
46 Twine	107
46.1 Why Should I Use This?	107
46.2 Features	107
46.3 Installation	107
46.4 Using Twine	108
46.5 Commands	108
46.5.1 twine upload	108
46.5.2 twine check	109
46.5.3 twine register	110
46.6 Configuration	111
46.6.1 Environment Variables	111
46.6.2 Proxy Support	112
46.7 Keyring Support	112
46.7.1 Disabling Keyring	112
Python Module Index	113
Index	115

This project follows the [semantic versioning](#) and [pre-release versioning](#) schemes recommended by the Python Packaging Authority.

TWINE 5.0.0 (2024-02-10)

1.1 Bugfixes

- Use `email.message` instead of `cgi` as `cgi` has been deprecated ([#969](#))

1.2 Misc

- [#931](#), [#991](#), [#1028](#), [#1040](#)

TWINE 4.0.2 (2022-11-30)

2.1 Bugfixes

- Remove deprecated function to fix `twine` check with `pkginfo` 1.9.0. ([#941](#))

TWINE 4.0.1 (2022-06-01)

3.1 Bugfixes

- Improve logging when keyring fails. ([#890](#))
- Reconfigure root logger to show all log messages. ([#896](#))

TWINE 4.0.0 (2022-03-31)

4.1 Features

- Drop support for Python 3.6. (#869)
- Use Rich to add color to upload output. (#851)
- Use Rich to add color to check output. (#874)
- Use Rich instead of tqdm for upload progress bar. (#877)

4.2 Bugfixes

- Remove Twine's dependencies from the User-Agent header when uploading. (#871)
- Improve detection of disabled BLAKE2 hashing due to FIPS mode. (#879)
- Restore warning for missing long_description. (#887)

TWINE 3.8.0 (2022-02-02)

5.1 Features

- Add `--verbose` logging for querying keyring credentials. (#849)
- Log all upload responses with `--verbose`. (#859)
- Show more helpful error message for invalid metadata. (#861)

5.2 Bugfixes

- Require a recent version of `urllib3`. (#858)

TWINE 3.7.1 (2021-12-07)

6.1 Improved Documentation

- Fix broken link to packaging tutorial. ([#844](#))

TWINE 3.7.0 (2021-12-01)

7.1 Features

- Add support for core metadata version 2.2, defined in PEP 643. ([#833](#))

TWINE 3.6.0 (2021-11-10)

8.1 Features

- Add support for Python 3.10. ([#827](#))

TWINE 3.5.0 (2021-11-02)

9.1 Features

- Show more helpful messages for invalid passwords. (#815)
- Allow the `--skip-existing` option to work with GCP Artifact Registry. (#823)

9.2 Bugfixes

- Add a helpful error message when an upload fails due to missing a trailing slash in the URL. (#812)
- Generalize `--verbose` suggestion when an upload fails. (#817)

TWINE 3.4.2 (2021-07-20)

10.1 Bugfixes

- Improve error message for unsupported metadata. ([#755](#))
- Improve error message for a missing config file. ([#770](#))
- Do not include md5_digest or blake2_256_digest if FIPS mode is enabled on the host. This removes those fields from the metadata before sending the metadata to the repository. ([#776](#))

TWINE 3.4.1 (2021-03-16)

11.1 Bugfixes

- Fix a regression that was causing some namespace packages with dots in them fail to upload to PyPI. ([#745](#))

TWINE 3.4.0 (2021-03-15)

12.1 Features

- Prefer `importlib.metadata` for entry point handling. ([#728](#))
- Rely on `importlib_metadata` 3.6 for nicer entry point processing. ([#732](#))
- Eliminate dependency on `setuptools/pkg_resources` and replace with `packaging` and `importlib_metadata`. ([#736](#))

TWINE 3.3.0 (2020-12-23)

13.1 Features

- Print files to be uploaded using `upload --verbose` (#670)
- Print configuration file location when using `upload --verbose` (#675)
- Print source and values of credentials when using `upload --verbose` (#685)
- Add support for Python 3.9 (#708)
- Turn warnings into errors when using `check --strict` (#715)

13.2 Bugfixes

- Make password optional when using `upload --client-cert` (#678)
- Support more Nexus versions with `upload --skip-existing` (#693)
- Support Gitlab Enterprise with `upload --skip-existing` (#698)
- Show a better error message for malformed files (#714)

13.3 Improved Documentation

- Adopt PSF code of conduct (#680)
- Adopt towncrier for the changelog (#718)

TWINE 3.2.0 (2020-06-24)

14.1 Features

- Improve display of HTTP errors during upload (#666)
- Print packages and signatures to be uploaded when using `--verbose` option (#652)
- Use red text when printing errors on the command line (#649)
- Require repository URL scheme to be `http` or `https` (#602)
- Add type annotations, checked with mypy, with **PEP 561** support for users of Twine's API (#231)

14.2 Bugfixes

- Update URL to `.pypirc` specification (#655)
- Don't raise an exception when Python version can't be parsed from filename (#612)
- Fix inaccurate retry message during upload (#611)
- Clarify error messages for archive format (#601)

TWINE 3.1.1 (2019-11-27)

15.1 Bugfixes

- Restore `--non-interactive` as a flag not expecting an argument. ([#548](#))

TWINE 3.1.0 (2019-11-23)

16.1 Features

- Add support for specifying `--non-interactive` as an environment variable. ([#547](#))

TWINE 3.0.0 (2019-11-18)

17.1 Features

- When a client certificate is indicated, all password processing is disabled. (#336)
- Add `--non-interactive` flag to abort upload rather than interactively prompt if credentials are missing. (#489)
- Twine now unconditionally requires the keyring library and no longer supports uninstalling `keyring` as a means to disable that functionality. Instead, use `keyring --disable` keyring functionality if necessary. (#524)
- Add Python 3.8 to classifiers. (#518)

17.2 Bugfixes

- More robust handling of server response in `--skip-existing` (#332)

TWINE 2.0.0 (2019-09-24)

18.1 Features

- Twine now requires Python 3.6 or later. Use pip 9 or pin to “twine<2” to install twine on older Python versions. (#437)

18.2 Bugfixes

- Require requests 2.20 or later to avoid reported security vulnerabilities in earlier releases. (#491)

TWINE 1.15.0 (2019-09-17)

19.1 Features

- Improved output on `check` command: Prints a message when there are no distributions given to check. Improved handling of errors in a distribution's markup, avoiding messages flowing through to the next distribution's errors. (#488)

TWINE 1.14.0 (2019-09-06)

20.1 Features

- Show Warehouse URL after uploading a package (#459)
- Better error handling and gpg2 fallback if gpg not available. (#456)
- Now provide a more meaningful error on redirect during upload. (#310)

20.2 Bugfixes

- Fail more gracefully when encountering bad metadata (#341)

TWINE 1.13.0 (2019-02-13)

21.1 Features

- Add `disable_progress_bar` option to disable `tqdm`. (#427)
- Allow defining an empty username and password in `.pypirc`. (#426)
- Support `keyring.get_credential`. (#419)
- Support `keyring.get_username_and_password`. (#418)
- Add Python 3.7 to classifiers. (#416)

21.2 Bugfixes

- Restore prompts while retaining support for suppressing prompts. (#452)
- Avoid `requests-toolbelt` to 0.9.0 to prevent attempting to use `openssl` when it isn't available. (#447)
- Use `io.StringIO` instead of `StringIO`. (#444)
- Only install `pyblake2` if needed. (#441)
- Use modern Python language features. (#436)
- Specify `python_requires` in `setup.py` (#435)
- Use `https` URLs everywhere. (#432)
- Fix `-skip-existing` for Nexus Repos. (#428)
- Remove unnecessary usage of `readme_render.markdown`. (#421)
- Don't crash if there's no package description. (#412)
- Fix `keyring` support. (#408)

21.3 Misc

- Refactor tox env and travis config. ([#439](#))

TWINE 1.12.1 (2018-09-24)

22.1 Bugfixes

- Fix regression with upload exit code ([#404](#))

TWINE 1.12.0 (2018-09-24)

23.1 Features

- Add `twine check` command to check long description (#395)
- Drop support for Python 3.3 (#392)
- Empower `--skip-existing` for Artifactory repositories (#363)

23.2 Bugfixes

- Avoid MD5 when Python is compiled in FIPS mode (#367)

TWINE 1.11.0 (2018-03-19)

24.1 Features

- Remove PyPI as default `register` package index. (#320)
- Support Metadata 2.1 ([PEP 566](#)), including Markdown for description fields. (#319)

24.2 Bugfixes

- Raise exception if attempting upload to deprecated legacy PyPI URLs. (#322)
- Avoid uploading to PyPI when given alternate repository URL, and require `http://` or `https://` in `repository_url`. (#269)

24.3 Misc

- Update PyPI URLs. (#318)
- Add new maintainer, release checklists. (#314)
- Add instructions on how to use keyring. (#277)

TWINE 1.10.0 (2018-03-07)

25.1 Features

- Link to changelog from README (#46)
- Reorganize & improve user & developer documentation. (#304)
- Revise docs predicting future of `twine` (#303)
- Add architecture overview to docs (#296)
- Add doc building instructions (#295)
- Declare support for Python 3.6 (#257)
- Improve progressbar (#256)

25.2 Bugfixes

- Degrade gracefully when keyring is unavailable (#315)
- Fix changelog formatting (#299)
- Fix syntax highlighting in README (#298)
- Fix Read the Docs, tox, Travis configuration (#297)
- Fix Travis CI and test configuration (#286)
- Print progress to `stdout`, not `stderr` (#268)
- Fix `--repository[-url]` help text (#265)
- Remove obsolete registration guidance (#200)

TWINE 1.9.1 (2017-05-27)

26.1 Bugfixes

- Blacklist known bad versions of Requests. ([#253](#))

TWINE 1.9.0 (2017-05-22)

27.1 Bugfixes

- Twine sends less information about the user's system in the User-Agent string. (#229)
- Fix `--skip-existing` when used to upload a package for the first time. (#220)
- Fix precedence of `--repository-url` over `--repository`. (#206)

27.2 Misc

- Twine will now resolve passwords using the `keyring` if available. Module can be required with the `keyring` extra.
- Twine will use `hashlib.blake2b` on Python 3.6+ instead of `pyblake2`

TWINE 1.8.1 (2016-08-09)

28.1 Misc

- Check if a package exists if the URL is one of:
 - `https://pypi.python.org/pypi/`
 - `https://upload.pypi.org/`
 - `https://upload.pypi.io/`

This helps people with `https://upload.pypi.io` still in their `.pypirc` file.

TWINE 1.8.0 (2016-08-08)

29.1 Features

- Switch from `upload.pypi.io` to `upload.pypi.org`. (#201)
- Retrieve configuration from the environment as a default. (#144)
 - Repository URL will default to `TWINE_REPOSITORY`
 - Username will default to `TWINE_USERNAME`
 - Password will default to `TWINE_PASSWORD`
- Allow the Repository URL to be provided on the command-line (`--repository-url`) or via an environment variable (`TWINE_REPOSITORY_URL`). (#166)
- Generate Blake2b 256 digests for packages *if* `pyblake2` is installed. Users can use `python -m pip install twine[with-blake2]` to have `pyblake2` installed with Twine. (#171)

29.2 Misc

- Generate SHA256 digest for all packages by default.
- Stop testing on Python 2.6.
- Warn users if they receive a 500 error when uploading to `*pypi.python.org` (#199)

TWINE 1.7.4 (2016-07-09)

30.1 Bugfixes

- Correct a packaging error.

TWINE 1.7.3 (2016-07-08)

31.1 Bugfixes

- Fix uploads to instances of pypiserver using `--skip-existing`. We were not properly checking the return status code on the response after attempting an upload. ([#195](#))

31.2 Misc

- Avoid attempts to upload a package if we can find it on Legacy PyPI.

TWINE 1.7.2 (2016-07-05)

32.1 Bugfixes

- Fix issue where we were checking the existence of packages even if the user didn't specify `--skip-existing`.
(#189) (#191)

TWINE 1.7.1 (2016-07-05)

33.1 Bugfixes

- Clint was not specified in the wheel metadata as a dependency. ([#187](#))

TWINE 1.7.0 (2016-07-04)

34.1 Features

- Support `--cert` and `--client-cert` command-line flags and config file options for feature parity with pip. This allows users to verify connections to servers other than PyPI (e.g., local package repositories) with different certificates. (#142)
- Add progress bar to uploads. (#152)
- Allow `--skip-existing` to work for 409 status codes. (#162)
- Implement retries when the CDN in front of PyPI gives us a 5xx error. (#167)
- Switch Twine to upload to pypi.io instead of pypi.python.org. (#177)

34.2 Bugfixes

- Allow passwords to have %s in them. (#186)

TWINE 1.6.5 (2015-12-16)

35.1 Bugfixes

- Bump requests-toolbelt version to ensure we avoid ConnectionErrors ([#155](#))

TWINE 1.6.4 (2015-10-27)

36.1 Bugfixes

- Paths with hyphens in them break the Wheel regular expression. ([#145](#))
- Exception while accessing the `repository` key (sic) when raising a redirect exception. ([#146](#))

TWINE 1.6.3 (2015-10-05)

37.1 Bugfixes

- Fix uploading signatures causing a 500 error after large file support was added. ([#137](#), [#140](#))

TWINE 1.6.2 (2015-09-28)

38.1 Bugfixes

- Upload signatures with packages appropriately ([#132](#))

As part of the refactor for the 1.6.0 release, we were using the wrong name to find the signature file.

This also uncovered a bug where if you're using twine in a situation where `*` is not expanded by your shell, we might also miss uploading signatures to PyPI. Both were fixed as part of this.

TWINE 1.6.1 (2015-09-18)

39.1 Bugfixes

- Fix signing support for uploads ([#130](#))

TWINE 1.6.0 (2015-09-14)

40.1 Features

- Allow the user to specify the location of their `.pypirc` (#97)
- Support registering new packages with `twine register` (#8)
- Add the `--skip-existing` flag to `twine upload` to allow users to skip releases that already exist on PyPI. (#115)
- Upload wheels first to PyPI (#106)
- Large file support via the `requests-toolbelt` (#104)

40.2 Bugfixes

- Raise an exception on redirects (#92)
- Work around problems with Windows when using `getpass.getpass` (#116)
- Warnings triggered by `pkginfo` searching for `PKG-INFO` files should no longer be user visible. (#114)
- Provide more helpful messages if `.pypirc` is out of date. (#111)

TWINE 1.5.0 (2015-03-10)

41.1 Features

- Support commands not named “gpg” for signing ([#29](#))

41.2 Bugfixes

- Display information about the version of setuptools installed ([#85](#))
- Support deprecated pypirc file format ([#61](#))

41.3 Misc

- Add lower-limit to requests dependency

TWINE 1.4.0 (2014-12-12)

42.1 Features

- Switch to a git style dispatching for the commands to enable simpler commands and programmatic invocation. (#6)
- Parse ~/.pypirc ourselves and use `subprocess` instead of the `distutils.spawn` module. (#13)

42.2 Bugfixes

- Expand globs and check for existence of dists to upload (#65)
- Fix issue uploading packages with `_s` in the name (#47)
- List registered commands in help text (#34)
- Use `pkg_resources` to load registered commands (#32)
- Prevent `ResourceWarning` from being shown (#28)
- Add support for uploading Windows installers (#26)

TWINE 1.3.0 (2014-03-31)

43.1 Features

- Additional functionality.

TWINE 1.2.2 (2013-10-03)

44.1 Features

- Basic functionality.

CONTRIBUTING

We are happy you have decided to contribute to Twine.

Please see [the GitHub repository](#) for code and more documentation, and the [official Python Packaging User Guide](#) for user documentation. To ask questions or get involved, you can join the [Python Packaging Discourse forum](#), [#pypa](#) or [#pypa-dev](#) on IRC, or the [distutils-sig mailing list](#).

Everyone interacting in the Twine project's codebases, issue trackers, chat rooms, and mailing lists is expected to follow the [PSF Code of Conduct](#).

45.1 Getting started

We use `tox` to run tests, check code style, and build the documentation. To install `tox`, run:

```
python3 -m pip install tox
```

Clone the twine repository from GitHub, then run:

```
cd /path/to/your/local/twine
tox -e dev
```

This creates a [virtual environment](#), so that twine and its dependencies do not interfere with other packages installed on your machine. In the virtual environment, twine is pointing at your local copy, so when you make changes, you can easily see their effect.

The virtual environment also contains the tools for running tests and checking code style, so you can run them on single files directly or in your code editor. However, we still encourage using the `tox` commands below on the whole codebase.

To use the virtual environment, run:

```
source venv/bin/activate
```

45.1.1 Building the documentation

Additions and edits to twine’s documentation are welcome and appreciated.

To preview the docs while you’re making changes, run:

```
tox -e watch-docs
```

Then open a web browser to <http://127.0.0.1:8000>.

When you’re done making changes, lint and build the docs locally before making a pull request. In your active virtual environment, run:

```
tox -e docs
```

The HTML of the docs will be written to docs/_build/html.

45.1.2 Code style

To automatically reformat your changes with [isort](#) and [black](#), run:

```
tox -e format
```

To detect any remaining code smells with [flake8](#), run:

```
tox -e lint
```

To perform strict type-checking using [mypy](#), run:

```
tox -e types
```

Any errors from `lint` or `types` need to be fixed manually.

Additionally, we prefer that `import` statements be used for packages and modules only, rather than individual classes or functions.

45.1.3 Testing

We use [pytest](#) for writing and running tests.

To run the tests in your virtual environment, run:

```
tox -e py
```

To pass options to `pytest`, e.g. the name of a test, run:

```
tox -e py -- tests/test_upload.py::test_exception_for_http_status
```

Twine is continuously tested against supported versions of Python using [GitHub Actions](#). To run the tests against a specific version, e.g. Python 3.8, you will need it installed on your machine. Then, run:

```
tox -e py38
```

To run the “integration” tests of uploading to real package indexes, run:

```
tox -e integration
```

To run the tests against all supported Python versions, check code style, and build the documentation, run:

```
tox
```

45.2 Submitting changes

1. Fork [the GitHub repository](#).
2. Make a branch off of `main` and commit your changes to it.
3. Run the tests, check code style, and build the docs as described above.
4. Optionally, add your name to the end of the `AUTHORS` file using the format `Name <email@domain.com> (url)`, where the `(url)` portion is optional.
5. Submit a pull request to the `main` branch on GitHub, referencing an open issue.
6. Add a changelog entry.

45.2.1 Changelog entries

The `docs/changelog.rst` file is built by [towncrier](#) from files in the `changelog/` directory. To add an entry, create a file in that directory named `{number}.{type}.rst`, where `{number}` is the pull request number, and `{type}` is `feature`, `bugfix`, `doc`, `removal`, or `misc`.

For example, if your PR number is 1234 and it's fixing a bug, then you would create `changelog/1234.bugfix.rst`. PRs can span multiple categories by creating multiple files: if you added a feature and deprecated/removed an old feature in PR #5678, you would create `changelog/5678.feature.rst` and `changelog/5678.removal.rst`.

A changelog entry is meant for end users and should only contain details relevant to them. In order to maintain a consistent style, please keep the entry to the point, in sentence case, shorter than 80 characters, and in an imperative tone. An entry should complete the sentence "This change will ...". If one line is not enough, use a summary line in an imperative tone, followed by a description of the change in one or more paragraphs, each wrapped at 80 characters and separated by blank lines.

You don't need to reference the pull request or issue number in a changelog entry, since [towncrier](#) will add a link using the number in the file name, and the pull request should reference an issue number. Similarly, you don't need to add your name to the entry, since that will be associated with the pull request.

Changelog entries are rendered using [reStructuredText](#), but they should only have minimal formatting (such as ```monospaced text```).

45.3 Architectural overview

Twine is a command-line tool for interacting with PyPI securely over HTTPS. Its three purposes are to be:

1. A user-facing tool for publishing on `pypi.org`
2. A user-facing tool for publishing on other Python package indexes (e.g., `devpi` instances)
3. A useful API for other programs (e.g., `zest.releaser`) to call for publishing on any Python package index

Currently, twine has two principle functions: uploading new packages and registering new projects (register is no longer supported on PyPI, and is in Twine for use with other package indexes).

Its command line arguments are parsed in `twine/cli.py`. The code for registering new projects is in `twine/commands/register.py`, and the code for uploading is in `twine/commands/upload.py`. The file `twine/package.py` contains a single class, `PackageFile`, which hashes the project files and extracts their metadata. The file `twine/repository.py` contains the `Repository` class, whose methods control the URL the package is uploaded to (which the user can specify either as a default, in the `.pypirc` file, or pass on the command line), and the methods that upload the package securely to a URL.

For more details, refer to the source documentation (currently a [work in progress](#)):

45.3.1 twine package

Top-level module for Twine.

The contents of this package are not a public API. For more details, see <https://github.com/pypa/twine/issues/194> and <https://github.com/pypa/twine/issues/665>.

twine.commands package

Module containing the logic for the twine sub-commands.

The contents of this package are not a public API. For more details, see <https://github.com/pypa/twine/issues/194> and <https://github.com/pypa/twine/issues/665>.

twine.commands.check module

Module containing the logic for twine check.

class `twine.commands.check._WarningStream`

write(*text: str*) → int

Write string to file.

Returns the number of characters written, which is always equal to the length of the string.

`twine.commands.check._parse_content_type`(*value: str*) → `Tuple[str, Dict[str, str]]`

Implement logic of deprecated `cgi.parse_header()`.

From https://docs.python.org/3.11/library/cgi.html#cgi.parse_header.

`twine.commands.check._check_file`(*filename: str, render_warning_stream: _WarningStream*) → `Tuple[List[str], bool]`

Check given distribution.

`twine.commands.check.check`(*dists: List[str], strict: bool = False*) → bool

Check that a distribution will render correctly on PyPI and display the results.

This is currently only validates `long_description`, but more checks could be added; see <https://github.com/pypa/twine/projects/2>.

Parameters

- **dists** – The distribution files to check.
- **output_stream** – The destination of the resulting output.

- **strict** – If True, treat warnings as errors.

Returns

True if there are rendering errors, otherwise False.

`twine.commands.check.main(args: List[str]) → bool`

Execute the check command.

Parameters

args – The command-line arguments.

Returns

The exit status of the check command.

twine.commands.register module

Module containing the logic for `twine register`.

`twine.commands.register.register(register_settings: Settings, package: str) → None`

Pre-register a package name with a repository before uploading a distribution.

Pre-registration is not supported on PyPI, so the `register` command is only necessary if you are using a different repository that requires it.

Parameters

- **register_settings** – The configured options relating to repository registration.
- **package** – The path of the distribution to use for package metadata.

Raises

- **twine.exceptions.TwineException** – The registration failed due to a configuration error.
- **requests.HTTPError** – The repository responded with an error.

`twine.commands.register.main(args: List[str]) → None`

Execute the register command.

Parameters

args – The command-line arguments.

twine.commands.upload module

Module containing the logic for `twine upload`.

`twine.commands.upload.skip_upload(response: Response, skip_existing: bool, package: PackageFile) → bool`

Determine if a failed upload is an error or can be safely ignored.

Parameters

- **response** – The response from attempting to upload package to a repository.
- **skip_existing** – If True, use the status and content of `response` to determine if the package already exists on the repository. If so, then a failed upload is safe to ignore.
- **package** – The package that was being uploaded.

Returns

True if a failed upload can be safely ignored, otherwise False.

`twine.commands.upload._make_package(filename: str, signatures: Dict[str, str], attestations: List[str], upload_settings: Settings) → PackageFile`

Create and sign a package, based off of filename, signatures, and settings.

Additionally, any supplied attestations are attached to the package when the settings indicate to do so.

class `twine.commands.upload.Inputs`

Represents structured user inputs.

dists: `List[str]`

Alias for field number 0

signatures: `Dict[str, str]`

Alias for field number 1

attestations_by_dist: `Dict[str, List[str]]`

Alias for field number 2

static `__new__(_cls, dists: List[str], signatures: Dict[str, str], attestations_by_dist: Dict[str, List[str]])`

Create new instance of Inputs(dists, signatures, attestations_by_dist)

`_asdict()`

Return a new dict which maps field names to their values.

`_field_defaults = {}`

`_fields = ('dists', 'signatures', 'attestations_by_dist')`

classmethod `_make(iterable)`

Make a new Inputs object from a sequence or iterable

`_replace(**kws)`

Return a new Inputs object replacing specified fields with new values

`twine.commands.upload._split_inputs(inputs: List[str]) → Inputs`

Split the unstructured list of input files provided by the user into groups.

Three groups are returned: upload files (i.e. dists), signatures, and attestations.

Upload files are returned as a linear list, signatures are returned as a dict of `basename` → `path`, and attestations are returned as a dict of `dist-path` → `[attestation-path]`.

`twine.commands.upload.upload(upload_settings: Settings, dists: List[str]) → None`

Upload one or more distributions to a repository, and display the progress.

If a package already exists on the repository, most repositories will return an error response. However, if `upload_settings.skip_existing` is `True`, a message will be displayed and any remaining distributions will be uploaded.

For known repositories (like PyPI), the web URLs of successfully uploaded packages will be displayed.

Parameters

- **upload_settings** – The configured options related to uploading to a repository.
- **dists** – The distribution files to upload to the repository. This can also include `.asc` and `.attestation` files, which will be added to their respective file uploads.

Raises

- `twine.exceptions.TwineException` – The upload failed due to a configuration error.

- `requests.HTTPError` – The repository responded with an error.

`twine.commands.upload.main(args: List[str]) → None`

Execute the upload command.

Parameters

args – The command-line arguments.

twine.auth module

`class twine.auth.CredentialInput`

`__init__(username: str | None = None, password: str | None = None) → None`

`class twine.auth.Resolver`

`__init__(config: Dict[str, str | None], input: CredentialInput) → None`

`classmethod choose(interactive: bool) → Type[Resolver]`

`property username: str | None`

`property password: str | None`

`property system: str | None`

`get_username_from_keyring() → str | None`

`get_password_from_keyring() → str | None`

`username_from_keyring_or_prompt() → str`

`password_from_keyring_or_prompt() → str`

`prompt(what: str, how: Callable[[...], str]) → str`

`class twine.auth.Private`

`prompt(what: str, how: Callable[[...], str] | None = None) → str`

twine.cli module

`twine.cli.configure_output() → None`

`twine.cli.list_dependencies_and_versions() → List[Tuple[str, str]]`

`twine.cli.dep_versions() → str`

`twine.cli.dispatch(argv: List[str]) → Any`

twine.exceptions module

Module containing exceptions raised by twine.

exception `twine.exceptions.TwineException`

Base class for all exceptions raised by twine.

exception `twine.exceptions.RedirectDetected`

A redirect was detected that the user needs to resolve.

In some cases, requests refuses to issue a new POST request after a redirect. In order to prevent a confusing user experience, we raise this exception to allow users to know the index they're uploading to is redirecting them.

classmethod `from_args(repository_url: str, redirect_url: str) → RedirectDetected`

exception `twine.exceptions.PackageNotFound`

A package file was provided that could not be found on the file system.

This is only used when attempting to register a package_file.

exception `twine.exceptions.UploadToDeprecatedPyPIDetected`

An upload attempt was detected to deprecated PyPI domains.

The sites pypi.python.org and testpypi.python.org are deprecated.

classmethod `from_args(target_url: str, default_url: str, test_url: str) → UploadToDeprecatedPyPIDetected`

Return an UploadToDeprecatedPyPIDetected instance.

exception `twine.exceptions.UnreachableRepositoryURLDetected`

An upload attempt was detected to a URL without a protocol prefix.

All repository URLs must have a protocol (e.g., https://).

exception `twine.exceptions.InvalidSigningConfiguration`

Both the sign and identity parameters must be present.

exception `twine.exceptions.InvalidSigningExecutable`

Signing executable must be installed on system.

exception `twine.exceptions.InvalidConfiguration`

Raised when configuration is invalid.

exception `twine.exceptions.InvalidDistribution`

Raised when a distribution is invalid.

exception `twine.exceptions.NonInteractive`

Raised in non-interactive mode when credentials could not be found.

exception `twine.exceptions.InvalidPyPIUploadURL`

Repository configuration tries to use PyPI with an incorrect URL.

For example, <https://pypi.org> instead of <https://upload.pypi.org/legacy>.

twine.package module

`twine.package._safe_name(name: str) → str`

Convert an arbitrary string to a standard distribution name.

Any runs of non-alphanumeric/. characters are replaced with a single '-'.

Copied from `pkg_resources.safe_name` for compatibility with warehouse. See <https://github.com/pypa/twine/issues/743>.

class `twine.package.PackageFile`

`__init__(filename: str, comment: str | None, metadata: Distribution, python_version: str | None, filetype: str | None) → None`

classmethod `from_filename(filename: str, comment: str | None) → PackageFile`

metadata_dictionary() → Dict[str, str | None | Sequence[str] | Tuple[str, bytes]]

Merge multiple sources of metadata into a single dictionary.

Includes values from filename, PKG-INFO, hashers, and signature.

add_attestations(attestations: List[str]) → None

add_gpg_signature(signature_filepath: str, signature_filename: str) → None

sign(sign_with: str, identity: str | None) → None

classmethod `run_gpg(gpg_args: Tuple[str, ...]) → None`

class `twine.package.Hexdigest`

Hexdigest(md5, sha2, blake2)

md5: str | None

Alias for field number 0

sha2: str | None

Alias for field number 1

blake2: str | None

Alias for field number 2

static `__new__(cls, md5: str | None, sha2: str | None, blake2: str | None)`

Create new instance of Hexdigest(md5, sha2, blake2)

_asdict()

Return a new dict which maps field names to their values.

_field_defaults = {}

_fields = ('md5', 'sha2', 'blake2')

classmethod `_make(iterable)`

Make a new Hexdigest object from a sequence or iterable

_replace(**kws)

Return a new Hexdigest object replacing specified fields with new values

class `twine.package.HashManager`

Manage our hashing objects for simplicity.

This will also allow us to better test this logic.

__init__(*filename: str*) → *None*

Initialize our manager and hasher objects.

_md5_update(*content: bytes*) → *None*

_md5_hexdigest() → *str* | *None*

_sha2_update(*content: bytes*) → *None*

_sha2_hexdigest() → *str* | *None*

_blake_update(*content: bytes*) → *None*

_blake_hexdigest() → *str* | *None*

hash() → *None*

Hash the file contents.

hexdigest() → *Hexdigest*

Return the hexdigest for the file.

twine.repository module**class** `twine.repository.Repository`

__init__(*repository_url: str*, *username: str* | *None*, *password: str* | *None*, *disable_progress_bar: bool* = *False*) → *None*

static **_make_adapter_with_retries**() → *HTTPAdapter*

static **_make_user_agent_string**() → *str*

close() → *None*

static **_convert_data_to_list_of_tuples**(*data: Dict[str, Any]*) → *List[Tuple[str, Any]]*

set_certificate_authority(*cacert: str* | *None*) → *None*

set_client_certificate(*clientcert: str* | *None*) → *None*

register(*package: PackageFile*) → *Response*

_upload(*package: PackageFile*) → *Response*

upload(*package: PackageFile*, *max_redirects: int* = 5) → *Response*

package_is_uploaded(*package: PackageFile*, *bypass_cache: bool* = *False*) → *bool*

release_urls(*packages: List[PackageFile]*) → *Set[str]*

verify_package_integrity(*package: PackageFile*) → *None*

twine.settings module

Module containing logic for handling settings.

class twine.settings.Settings

Object that manages the configuration for Twine.

This object can only be instantiated with keyword arguments.

For example,

```
Settings(True, username='fakeusername')
```

Will raise a `TypeError`. Instead, you would want

```
Settings(sign=True, username='fakeusername')
```

```
__init__(*, attestations: bool = False, sign: bool = False, sign_with: str = 'gpg', identity: str | None =
None, username: str | None = None, password: str | None = None, non_interactive: bool = False,
comment: str | None = None, config_file: str = utils.DEFAULT_CONFIG_FILE, skip_existing:
bool = False, cacert: str | None = None, client_cert: str | None = None, repository_name: str =
'pypi', repository_url: str | None = None, verbose: bool = False, disable_progress_bar: bool =
False, **ignored_kwargs: Any) → None
```

Initialize our settings instance.

Parameters

- **attestations** – Whether the package file should be uploaded with attestations.
- **sign** – Configure whether the package file should be signed.
- **sign_with** – The name of the executable used to sign the package with.
- **identity** – The GPG identity that should be used to sign the package file.
- **username** – The username used to authenticate to the repository (package index).
- **password** – The password used to authenticate to the repository (package index).
- **non_interactive** – Do not interactively prompt for username/password if the required credentials are missing.
- **comment** – The comment to include with each distribution file.
- **config_file** – The path to the configuration file to use.
- **skip_existing** – Specify whether twine should continue uploading files if one of them already exists. This primarily supports PyPI. Other package indexes may not be supported.
- **cacert** – The path to the bundle of certificates used to verify the TLS connection to the package index.
- **client_cert** – The path to the client certificate used to perform authentication to the index. This must be a single file that contains both the private key and the PEM-encoded certificate.
- **repository_name** – The name of the repository (package index) to interact with. This should correspond to a section in the config file.
- **repository_url** – The URL of the repository (package index) to interact with. This will override the settings inferred from `repository_name`.
- **verbose** – Show verbose output.

- **disable_progress_bar** – Disable the progress bar.

property username: `str` | `None`

property password: `str` | `None`

_allow_noninteractive() → `AbstractContextManager[None]`

Bypass NonInteractive error when client cert is present.

property verbose: `bool`

static register_argparse_arguments(parser: `ArgumentParser`) → `None`

Register the arguments for argparse.

classmethod from_argparse(args: `Namespace`) → `Settings`

Generate the Settings from parsed arguments.

_handle_package_signing(sign: `bool`, sign_with: `str`, identity: `str` | `None`) → `None`

_handle_repository_options(repository_name: `str`, repository_url: `str` | `None`) → `None`

_handle_certificates(cacert: `str` | `None`, client_cert: `str` | `None`) → `None`

check_repository_url() → `None`

Verify we are not using legacy PyPI.

Raises

`twine.exceptions.UploadToDeprecatedPyPIDetected` – The configured repository URL is for legacy PyPI.

create_repository() → `Repository`

Create a new repository for uploading.

twine.utils module

twine.utils.get_config(path: `str`) → `Dict[str, Dict[str, str | None]]`

Read repository configuration from a file (i.e. `~/.pypirc`).

Format: <https://packaging.python.org/specifications/pypirc/>

If the default config file doesn't exist, return a default configuration for pypyi and testpypi.

twine.utils._validate_repository_url(repository_url: `str`) → `None`

Validate the given url for allowed schemes and components.

twine.utils.get_repository_from_config(config_file: `str`, repository: `str`, repository_url: `str` | `None` = `None`) → `Dict[str, str | None]`

Get repository config command-line values or the `.pypirc` file.

twine.utils.normalize_repository_url(url: `str`) → `str`

twine.utils.get_file_size(filename: `str`) → `str`

Return the size of a file in KB, or MB if `>= 1024 KB`.

twine.utils.check_status_code(response: `Response`, verbose: `bool`) → `None`

Generate a helpful message based on the response from the repository.

Raise a custom exception for recognized errors. Otherwise, print the response content (based on the verbose option) before re-raising the `HTTPError`.

```
twine.utils.get_userpass_value(cli_value: str | None, config: Dict[str, str | None], key: str, prompt_strategy:
    Callable[[str], str | None] = None) → str | None
```

Get a credential (e.g. a username or password) from the configuration.

Uses the following rules:

1. If `cli_value` is specified, use that.
2. If `config[key]` is specified, use that.
3. If `prompt_strategy` is specified, use its return value.
4. Otherwise return `None`

Parameters

- **cli_value** – The value supplied from the command line.
- **config** – A dictionary of repository configuration values.
- **key** – The credential to look up in `config`, e.g. "username" or "password".
- **prompt_strategy** – An argumentless function to get the value, e.g. from keyring or by prompting the user.

Returns

The credential value, i.e. the username or password.

```
twine.utils.get_cacert(cli_value: str | None, config: Dict[str, str | None], *, key: str = 'ca_cert',
    prompt_strategy: Callable[[str], str | None] = None) → str | None
```

Get the CA bundle via `get_userpass_value()`.

```
twine.utils.get_clientcert(cli_value: str | None, config: Dict[str, str | None], *, key: str = 'client_cert',
    prompt_strategy: Callable[[str], str | None] = None) → str | None
```

Get the client certificate via `get_userpass_value()`.

```
class twine.utils.EnvironmentDefault
```

Get values from environment variable.

```
__init__(env: str, required: bool = True, default: str | None = None, **kwargs: Any) → None
```

```
class twine.utils.EnvironmentFlag
```

Set boolean flag from environment variable.

```
__init__(env: str, **kwargs: Any) → None
```

```
static bool_from_env(val: str | None) → bool
```

Allow '0' and 'false' and 'no' to be False.

twine.wheel module

```
class twine.wheel.Wheel
```

```
__init__(filename: str, metadata_version: str | None = None) → None
```

```
property py_version: str
```

```
static find_candidate_metadata_files(names: List[str]) → List[List[str]]
```

Filter files that may be METADATA files.

`read()` → bytes

`parse(data: bytes)` → None

twine.wininst module

`class twine.wininst.WinInst`

`__init__(filename: str, metadata_version: str | None = None)` → None

`property py_version: str`

`read()` → bytes

45.3.2 Where Twine gets configuration and credentials

A user can set the repository URL, username, and/or password via command line, `.pypirc` files, environment variables, and `keyring`.

45.4 Adding a maintainer

A checklist for adding a new maintainer to the project.

1. Add them as a Member in the GitHub repo settings.
2. Get them Test PyPI and canon PyPI usernames and add them as a Maintainer on [our Test PyPI project](#) and [canon PyPI](#).

45.5 Making a new release

A checklist for creating, testing, and distributing a new version.

1. Choose a version number, and create a new branch

```
VERSION=3.4.2
```

```
git switch -c release-$VERSION
```

2. Update docs/changelog.rst

```
tox -e changelog -- --version $VERSION
```

```
git commit -am "Update changelog for $VERSION"
```

3. Open a pull request for review
4. Merge the pull request, and ensure the [GitHub Actions](#) build passes
5. Create a new git tag for the version

```
git switch main  
  
git pull --ff-only upstream main  
  
git tag -m "Release v$VERSION" $VERSION
```

6. Push to start the release, and watch it in [GitHub Actions](#)

```
git push upstream $VERSION
```

7. View the new release on [PyPI](#)

45.6 Future development

See our [open issues](#).

In the future, pip and twine may merge into a single tool; see [ongoing discussion](#).

TWINE

Twine is a utility for publishing Python packages to PyPI and other repositories. It provides build system independent uploads of source and binary distribution artifacts for both new and existing projects.

46.1 Why Should I Use This?

The goal of Twine is to improve PyPI interaction by improving security and testability.

The biggest reason to use Twine is that it securely authenticates you to PyPI over HTTPS using a verified connection, regardless of the underlying Python version. Meanwhile, `python setup.py upload` will only work correctly and securely if your build system, Python version, and underlying operating system are configured properly.

Secondly, Twine encourages you to build your distribution files. `python setup.py upload` only allows you to upload a package as a final step after building with `distutils` or `setuptools`, within the same command invocation. This means that you cannot test the exact file you're going to upload to PyPI to ensure that it works before uploading it.

Finally, Twine allows you to pre-sign your files and pass the `.asc` files into the command line invocation (`twine upload myproject-1.0.1.tar.gz myproject-1.0.1.tar.gz.asc`). This enables you to be assured that you're typing your `gpg` passphrase into `gpg` itself and not anything else, since *you* will be the one directly executing `gpg --detach-sign -a <filename>`.

46.2 Features

- Verified HTTPS connections
- Uploading doesn't require executing `setup.py`
- Uploading files that have already been created, allowing testing of distributions before release
- Supports uploading any packaging format (including `wheels`)

46.3 Installation

```
pip install twine
```

46.4 Using Twine

1. Create some distributions in the normal way:

```
python -m build
```

2. Upload to [Test PyPI](#) and verify things look right:

```
twine upload -r testpypi dist/*
```

Twine will prompt for your username and password.

3. Upload to [PyPI](#):

```
twine upload dist/*
```

4. Done!

Note: Like many other command line tools, Twine does not show any characters when you enter your password.

If you're using Windows and trying to paste your username, password, or token in the Command Prompt or PowerShell, Ctrl-V and Shift+Insert won't work. Instead, you can use "Edit > Paste" from the window menu, or enable "Use Ctrl+Shift+C/V as Copy/Paste" in "Properties". This is a [known issue](#) with Python's `getpass` module.

More documentation on using Twine to upload packages to PyPI is in the [Python Packaging User Guide](#).

46.5 Commands

46.5.1 `twine upload`

Uploads one or more distributions to a repository.

```
usage: twine upload [-h] [-r REPOSITORY] [--repository-url REPOSITORY_URL]
                  [--attestations] [-s] [--sign-with SIGN_WITH]
                  [-i IDENTITY] [-u USERNAME] [-p PASSWORD]
                  [--non-interactive] [-c COMMENT]
                  [--config-file CONFIG_FILE] [--skip-existing]
                  [--cert path] [--client-cert path] [--verbose]
                  [--disable-progress-bar]
                  dist [dist ...]

positional arguments:
  dist                  The distribution files to upload to the repository
                        (package index). Usually dist/* . May additionally
                        contain a .asc file to include an existing signature
                        with the file upload.

options:
  -h, --help            show this help message and exit
  -r REPOSITORY, --repository REPOSITORY
                        The repository (package index) to upload the package
```

(continues on next page)

(continued from previous page)

```

to. Should be a section in the config file (default:
pypi). (Can also be set via TWINE_REPOSITORY
environment variable.)
--repository-url REPOSITORY_URL
    The repository (package index) URL to upload the
    package to. This overrides --repository. (Can also be
    set via TWINE_REPOSITORY_URL environment variable.)
--attestations
    Upload each file's associated attestations.
-s, --sign
    Sign files to upload using GPG.
--sign-with SIGN_WITH
    GPG program used to sign uploads (default: gpg).
-i IDENTITY, --identity IDENTITY
    GPG identity used to sign files.
-u USERNAME, --username USERNAME
    The username to authenticate to the repository
    (package index) as. (Can also be set via
    TWINE_USERNAME environment variable.)
-p PASSWORD, --password PASSWORD
    The password to authenticate to the repository
    (package index) with. (Can also be set via
    TWINE_PASSWORD environment variable.)
--non-interactive
    Do not interactively prompt for username/password if
    the required credentials are missing. (Can also be set
    via TWINE_NON_INTERACTIVE environment variable.)
-c COMMENT, --comment COMMENT
    The comment to include with the distribution file.
--config-file CONFIG_FILE
    The .pypirc config file to use.
--skip-existing
    Continue uploading files if one already exists. (Only
    valid when uploading to PyPI. Other implementations
    may not support this.)
--cert path
    Path to alternate CA bundle (can also be set via
    TWINE_CERT environment variable).
--client-cert path
    Path to SSL client certificate, a single file
    containing the private key and the certificate in PEM
    format.
--verbose
    Show verbose output.
--disable-progress-bar
    Disable the progress bar.

```

46.5.2 twine check

Checks whether your distribution's long description will render correctly on PyPI.

```

usage: twine check [-h] [--strict] dist [dist ...]

positional arguments:
  dist                The distribution files to check, usually dist/*

options:
  -h, --help          show this help message and exit

```

(continues on next page)

(continued from previous page)

`--strict` Fail on warnings

46.5.3 twine register

Pre-register a name with a repository before uploading a distribution.

Warning: Pre-registration is [not supported on PyPI](#), so the `register` command is only necessary if you are using a different repository that requires it. See [issue #1627 on Warehouse](#) (the software running on PyPI) for more details.

```
usage: twine register [-h] [-r REPOSITORY] [--repository-url REPOSITORY_URL]
                    [--attestations] [-s] [--sign-with SIGN_WITH]
                    [-i IDENTITY] [-u USERNAME] [-p PASSWORD]
                    [--non-interactive] [-c COMMENT]
                    [--config-file CONFIG_FILE] [--skip-existing]
                    [--cert path] [--client-cert path] [--verbose]
                    [--disable-progress-bar]
                    package
```

register operation is not required with PyPI.org

positional arguments:

 package File from which we read the package metadata.

options:

 -h, --help show this help message and exit

 -r REPOSITORY, --repository REPOSITORY
 The repository (package index) to upload the package to. Should be a section in the config file (default: pypi). (Can also be set via TWINE_REPOSITORY environment variable.)

 --repository-url REPOSITORY_URL
 The repository (package index) URL to upload the package to. This overrides --repository. (Can also be set via TWINE_REPOSITORY_URL environment variable.)

 --attestations Upload each file's associated attestations.

 -s, --sign Sign files to upload using GPG.

 --sign-with SIGN_WITH
 GPG program used to sign uploads (default: gpg).

 -i IDENTITY, --identity IDENTITY
 GPG identity used to sign files.

 -u USERNAME, --username USERNAME
 The username to authenticate to the repository (package index) as. (Can also be set via TWINE_USERNAME environment variable.)

 -p PASSWORD, --password PASSWORD
 The password to authenticate to the repository (package index) with. (Can also be set via TWINE_PASSWORD environment variable.)

(continues on next page)

(continued from previous page)

```

--non-interactive    Do not interactively prompt for username/password if
                    the required credentials are missing. (Can also be set
                    via TWINE_NON_INTERACTIVE environment variable.)
-c COMMENT, --comment COMMENT
                    The comment to include with the distribution file.
--config-file CONFIG_FILE
                    The .pypirc config file to use.
--skip-existing      Continue uploading files if one already exists. (Only
                    valid when uploading to PyPI. Other implementations
                    may not support this.)
--cert path          Path to alternate CA bundle (can also be set via
                    TWINE_CERT environment variable).
--client-cert path   Path to SSL client certificate, a single file
                    containing the private key and the certificate in PEM
                    format.
--verbose            Show verbose output.
--disable-progress-bar
                    Disable the progress bar.

```

46.6 Configuration

Twine can read repository configuration from a `.pypirc` file, either in your home directory, or provided with the `--config-file` option. For details on writing and using `.pypirc`, see the [specification](#) in the Python Packaging User Guide.

46.6.1 Environment Variables

Twine also supports configuration via environment variables. Options passed on the command line will take precedence over options set via environment variables. Definition via environment variable is helpful in environments where it is not convenient to create a `.pypirc` file (for example, on a CI/build server).

- `TWINE_USERNAME` - the username to use for authentication to the repository.
- `TWINE_PASSWORD` - the password to use for authentication to the repository.
- `TWINE_REPOSITORY` - the repository configuration, either defined as a section in `.pypirc` or provided as a full URL.
- `TWINE_REPOSITORY_URL` - the repository URL to use.
- `TWINE_CERT` - custom CA certificate to use for repositories with self-signed or untrusted certificates.
- `TWINE_NON_INTERACTIVE` - Do not interactively prompt for username/password if the required credentials are missing.

46.6.2 Proxy Support

Twine can be configured to use a proxy by setting environment variables. For example, to use a proxy for just the `twine` command, without `export`-ing it for other tools:

```
HTTPS_PROXY=socks5://user:pass@host:port twine upload dist/*
```

For more information, see the Requests documentation on [Proxies](#) and [SOCKS](#), and [an in-depth article about proxy environment variables](#).

46.7 Keyring Support

Instead of typing in your password every time you upload a distribution, Twine allows storing a username and password securely using [keyring](#). Keyring is installed with Twine but for some systems (Linux mainly) may require [additional installation steps](#).

Once Twine is installed, use the `keyring` program to set a username and password to use for each repository to which you may upload.

For example, to set a username and password for PyPI:

```
keyring set https://upload.pypi.org/legacy/ your-username
```

and enter the password when prompted.

For a different repository, replace the URL with the relevant repository URL. For example, for Test PyPI, use `https://test.pypi.org/legacy/`.

The next time you run `twine`, it will prompt you for a username, and then get the appropriate password from Keyring.

Note: If you are using Linux in a headless environment (such as on a server) you'll need to do some additional steps to ensure that Keyring can store secrets securely. See [Using Keyring on headless systems](#).

46.7.1 Disabling Keyring

In most cases, simply not setting a password with `keyring` will allow Twine to fall back to prompting for a password. In some cases, the presence of Keyring will cause unexpected or undesirable prompts from the backing system. In these cases, it may be desirable to disable Keyring altogether. To disable Keyring, run:

```
keyring --disable
```

See [Twine issue #338](#) for discussion and background.

PYTHON MODULE INDEX

t

- `twine`, 94
- `twine.auth`, 97
- `twine.cli`, 97
- `twine.commands`, 94
- `twine.commands.check`, 94
- `twine.commands.register`, 95
- `twine.commands.upload`, 95
- `twine.exceptions`, 98
- `twine.package`, 99
- `twine.repository`, 100
- `twine.settings`, 101
- `twine.utils`, 102
- `twine.wheel`, 103
- `twine.wininst`, 104

Symbols

- `_WarningStream` (class in `twine.commands.check`), 94
 - `__init__()` (`twine.auth.CredentialInput` method), 97
 - `__init__()` (`twine.auth.Resolver` method), 97
 - `__init__()` (`twine.package.HashManager` method), 100
 - `__init__()` (`twine.package.PackageFile` method), 99
 - `__init__()` (`twine.repository.Repository` method), 100
 - `__init__()` (`twine.settings.Settings` method), 101
 - `__init__()` (`twine.utils.EnvironmentDefault` method), 103
 - `__init__()` (`twine.utils.EnvironmentFlag` method), 103
 - `__init__()` (`twine.wheel.Wheel` method), 103
 - `__init__()` (`twine.wininst.WinInst` method), 104
 - `__new__()` (`twine.commands.upload.Inputs` static method), 96
 - `__new__()` (`twine.package.Hexdigest` static method), 99
 - `_allow_noninteractive()` (`twine.settings.Settings` method), 102
 - `_asdict()` (`twine.commands.upload.Inputs` method), 96
 - `_asdict()` (`twine.package.Hexdigest` method), 99
 - `_blake_hexdigest()` (`twine.package.HashManager` method), 100
 - `_blake_update()` (`twine.package.HashManager` method), 100
 - `_check_file()` (in module `twine.commands.check`), 94
 - `_convert_data_to_list_of_tuples()` (`twine.repository.Repository` static method), 100
 - `_field_defaults` (`twine.commands.upload.Inputs` attribute), 96
 - `_field_defaults` (`twine.package.Hexdigest` attribute), 99
 - `_fields` (`twine.commands.upload.Inputs` attribute), 96
 - `_fields` (`twine.package.Hexdigest` attribute), 99
 - `_handle_certificates()` (`twine.settings.Settings` method), 102
 - `_handle_package_signing()` (`twine.settings.Settings` method), 102
 - `_handle_repository_options()` (`twine.settings.Settings` method), 102
 - `_make()` (`twine.commands.upload.Inputs` class method), 96
 - `_make()` (`twine.package.Hexdigest` class method), 99
 - `_make_adapter_with_retries()` (`twine.repository.Repository` static method), 100
 - `_make_package()` (in module `twine.commands.upload`), 95
 - `_make_user_agent_string()` (`twine.repository.Repository` static method), 100
 - `_md5_hexdigest()` (`twine.package.HashManager` method), 100
 - `_md5_update()` (`twine.package.HashManager` method), 100
 - `_parse_content_type()` (in module `twine.commands.check`), 94
 - `_replace()` (`twine.commands.upload.Inputs` method), 96
 - `_replace()` (`twine.package.Hexdigest` method), 99
 - `_safe_name()` (in module `twine.package`), 99
 - `_sha2_hexdigest()` (`twine.package.HashManager` method), 100
 - `_sha2_update()` (`twine.package.HashManager` method), 100
 - `_split_inputs()` (in module `twine.commands.upload`), 96
 - `_upload()` (`twine.repository.Repository` method), 100
 - `_validate_repository_url()` (in module `twine.utils`), 102
- ## A
- `add attestations()` (`twine.package.PackageFile` method), 99
 - `add_gpg_signature()` (`twine.package.PackageFile` method), 99
 - `attestations_by_dist` (`twine.commands.upload.Inputs` attribute), 96
- ## B
- `blake2` (`twine.package.Hexdigest` attribute), 99
 - `bool_from_env()` (`twine.utils.EnvironmentFlag` static method), 103

C

`check()` (in module `twine.commands.check`), 94
`check_repository_url()` (`twine.settings.Settings` method), 102
`check_status_code()` (in module `twine.utils`), 102
`choose()` (`twine.auth.Resolver` class method), 97
`close()` (`twine.repository.Repository` method), 100
`configure_output()` (in module `twine.cli`), 97
`create_repository()` (`twine.settings.Settings` method), 102
`CredentialInput` (class in `twine.auth`), 97

D

`dep_versions()` (in module `twine.cli`), 97
`dispatch()` (in module `twine.cli`), 97
`dist` (`twine.commands.upload.Inputs` attribute), 96

E

`EnvironmentDefault` (class in `twine.utils`), 103
`EnvironmentFlag` (class in `twine.utils`), 103

F

`find_candidate_metadata_files()` (`twine.wheel.Wheel` static method), 103
`from_argparse()` (`twine.settings.Settings` class method), 102
`from_args()` (`twine.exceptions.RedirectDetected` class method), 98
`from_args()` (`twine.exceptions.UploadToDeprecatedPyPIDetected` class method), 98
`from_filename()` (`twine.package.PackageFile` class method), 99

G

`get_cacert()` (in module `twine.utils`), 103
`get_clientcert()` (in module `twine.utils`), 103
`get_config()` (in module `twine.utils`), 102
`get_file_size()` (in module `twine.utils`), 102
`get_password_from_keyring()` (`twine.auth.Resolver` method), 97
`get_repository_from_config()` (in module `twine.utils`), 102
`get_username_from_keyring()` (`twine.auth.Resolver` method), 97
`get_userpass_value()` (in module `twine.utils`), 102

H

`hash()` (`twine.package.HashManager` method), 100
`HashManager` (class in `twine.package`), 99
`Hexdigest` (class in `twine.package`), 99
`hexdigest()` (`twine.package.HashManager` method), 100

I

`Inputs` (class in `twine.commands.upload`), 96
`InvalidConfiguration`, 98
`InvalidDistribution`, 98
`InvalidPyPIUploadURL`, 98
`InvalidSigningConfiguration`, 98
`InvalidSigningExecutable`, 98

L

`list_dependencies_and_versions()` (in module `twine.cli`), 97

M

`main()` (in module `twine.commands.check`), 95
`main()` (in module `twine.commands.register`), 95
`main()` (in module `twine.commands.upload`), 97
`md5` (`twine.package.Hexdigest` attribute), 99
`metadata_dictionary()` (`twine.package.PackageFile` method), 99
`module`
 `twine`, 94
 `twine.auth`, 97
 `twine.cli`, 97
 `twine.commands`, 94
 `twine.commands.check`, 94
 `twine.commands.register`, 95
 `twine.commands.upload`, 95
 `twine.exceptions`, 98
 `twine.package`, 99
 `twine.repository`, 100
 `twine.settings`, 101
 `twine.utils`, 102
 `twine.wheel`, 103
 `twine.wininst`, 104

N

`NonInteractive`, 98
`normalize_repository_url()` (in module `twine.utils`), 102

P

`package_is_uploaded()` (`twine.repository.Repository` method), 100
`PackageFile` (class in `twine.package`), 99
`PackageNotFound`, 98
`parse()` (`twine.wheel.Wheel` method), 104
`password` (`twine.auth.Resolver` property), 97
`password` (`twine.settings.Settings` property), 102
`password_from_keyring_or_prompt()` (`twine.auth.Resolver` method), 97
`Private` (class in `twine.auth`), 97
`prompt()` (`twine.auth.Private` method), 97
`prompt()` (`twine.auth.Resolver` method), 97

`py_version` (*twine.wheel.Wheel* property), 103
`py_version` (*twine.wininst.WinInst* property), 104
 Python Enhancement Proposals
 PEP 561, 29
 PEP 566, 49

R

`read()` (*twine.wheel.Wheel* method), 103
`read()` (*twine.wininst.WinInst* method), 104
 RedirectDetected, 98
`register()` (in module *twine.commands.register*), 95
`register()` (*twine.repository.Repository* method), 100
`register_argparse_arguments()`
 (*twine.settings.Settings* static method), 102
`release_urls()` (*twine.repository.Repository* method),
 100
 Repository (class in *twine.repository*), 100
 Resolver (class in *twine.auth*), 97
`run_gpg()` (*twine.package.PackageFile* class method),
 99

S

`set_certificate_authority()`
 (*twine.repository.Repository* method), 100
`set_client_certificate()`
 (*twine.repository.Repository* method), 100
 Settings (class in *twine.settings*), 101
`sha2` (*twine.package.Hexdigest* attribute), 99
`sign()` (*twine.package.PackageFile* method), 99
 signatures (*twine.commands.upload.Inputs* attribute),
 96
`skip_upload()` (in module *twine.commands.upload*), 95
 system (*twine.auth.Resolver* property), 97

T

twine
 module, 94
twine.auth
 module, 97
twine.cli
 module, 97
twine.commands
 module, 94
twine.commands.check
 module, 94
twine.commands.register
 module, 95
twine.commands.upload
 module, 95
twine.exceptions
 module, 98
twine.package
 module, 99
twine.repository

 module, 100
twine.settings
 module, 101
twine.utils
 module, 102
twine.wheel
 module, 103
twine.wininst
 module, 104
 TwineException, 98

U

UnreachableRepositoryURLDetected, 98
`upload()` (in module *twine.commands.upload*), 96
`upload()` (*twine.repository.Repository* method), 100
 UploadToDeprecatedPyPIDetected, 98
 username (*twine.auth.Resolver* property), 97
 username (*twine.settings.Settings* property), 102
`username_from_keyring_or_prompt()`
 (*twine.auth.Resolver* method), 97

V

verbose (*twine.settings.Settings* property), 102
`verify_package_integrity()`
 (*twine.repository.Repository* method), 100

W

Wheel (class in *twine.wheel*), 103
 WinInst (class in *twine.wininst*), 104
`write()` (*twine.commands.check._WarningStream*
 method), 94