# tld Documentation

***Release 0.13***

**Artur Barseghyan <artur.barseghyan@gmail.com>**

**Feb 28, 2023**

# Contents


**1 Prerequisites** **3**

**2 Documentation** **5**

**3 Installation** **7**

**4 Usage examples** **9**
- 4.1 Get the TLD name **as string** from the URL given . . . 9
- 4.2 Get the TLD as **an object** . . . 9
- 4.3 Get TLD name, **ignoring the missing protocol** . . . 10
- 4.4 Return TLD parts as tuple . . . 10
- 4.5 Get the first level domain name **as string** from the URL given . . . 10
- 4.6 Check if some tld is a valid tld . . . 10

**5 Update the list of TLD names** **13**

**6 Custom TLD parsers** **15**

**7 Custom list of TLD names** **17**

**8 Free up resources** **19**

**9 Troubleshooting** **21**

**10 Testing** **23**

**11 Writing documentation** **25**

**12 License** **27**

**13 Support** **29**

**14 Author** **31**

**15 Project documentation** **33**
- 15.1 Security Policy . . . 34
  - 15.1.1 Reporting a Vulnerability . . . 34
  - 15.1.2 Supported Versions . . . 34
- 15.2 Contributor Covenant Code of Conduct . . . 34

15.2.1 Our Pledge . . . 34
15.2.2 Our Standards . . . 35
15.2.3 Enforcement Responsibilities . . . 35
15.2.4 Scope . . . 35
15.2.5 Enforcement . . . 35
15.2.6 Enforcement Guidelines . . . 35
15.2.6.1 1. Correction . . . 36
15.2.6.2 2. Warning . . . 36
15.2.6.3 3. Temporary Ban . . . 36
15.2.6.4 4. Permanent Ban . . . 36
15.2.7 Attribution . . . 36
15.3 Contributor guidelines . . . 36
15.3.1 Developer prerequisites . . . 36
15.3.1.1 pre-commit . . . 36
15.3.2 Code standards . . . 37
15.3.3 Requirements . . . 37
15.3.4 Virtual environment . . . 37
15.3.5 Documentation . . . 37
15.3.6 Testing . . . 37
15.3.7 Pull requests . . . 37
15.3.8 Questions . . . 38
15.3.9 Issues . . . 38
15.4 Release history and notes . . . 38
15.4.1 0.13 . . . 38
15.4.2 0.12.7 . . . 39
15.4.3 0.12.6 . . . 39
15.4.4 0.12.5 . . . 39
15.4.5 0.12.4 . . . 39
15.4.6 0.12.3 . . . 39
15.4.7 0.12.2 . . . 40
15.4.8 0.12.1 . . . 40
15.4.9 0.12 . . . 40
15.4.10 0.11.11 . . . 40
15.4.11 0.11.10 . . . 40
15.4.12 0.11.9 . . . 41
15.4.13 0.11.8 . . . 41
15.4.14 0.11.7 . . . 41
15.4.15 0.11.6 . . . 41
15.4.16 0.11.5 . . . 41
15.4.17 0.11.4 . . . 41
15.4.18 0.11.3 . . . 41
15.4.19 0.11.2 . . . 42
15.4.20 0.11.1 . . . 42
15.4.21 0.11 . . . 42
15.4.22 0.10 . . . 42
15.4.23 0.9.8 . . . 42
15.4.24 0.9.7 . . . 43
15.4.25 0.9.6 . . . 43
15.4.26 0.9.5 . . . 43
15.4.27 0.9.4 . . . 43
15.4.28 0.9.3 . . . 43
15.4.29 0.9.2 . . . 43
15.4.30 0.9.1 . . . 44
15.4.31 0.9 . . . 44

15.4.32 0.8 . . . 44
15.4.33 0.7.10 . . . 45
15.4.34 0.7.9 . . . 46
15.4.35 0.7.8 . . . 46
15.4.36 0.7.7 . . . 46
15.4.37 0.7.6 . . . 46
15.4.38 0.7.5 . . . 46
15.4.39 0.7.4 . . . 46
15.4.40 0.7.3 . . . 46
15.4.41 0.7.2 . . . 47
15.4.42 0.7.1 . . . 47
15.4.43 0.7 . . . 47
15.4.44 0.6.4 . . . 47
15.4.45 0.6.3 . . . 47
15.4.46 0.6.2 . . . 47
15.4.47 0.6.1 . . . 47
15.4.48 0.6 . . . 48
15.4.49 0.5 . . . 48
15.4.50 0.4 . . . 48
15.5 tld package . . . 48
15.5.1 Submodules . . . 48
15.5.2 tld.base module . . . 48
15.5.3 tld.conf module . . . 49
15.5.4 tld.defaults module . . . 49
15.5.5 tld.exceptions module . . . 49
15.5.6 tld.helpers module . . . 49
15.5.7 tld.registry module . . . 49
15.5.8 tld.result module . . . 50
15.5.9 tld.trie module . . . 50
15.5.10 tld.utils module . . . 51
15.5.11 Module contents . . . 55

**16 Indices and tables** **59**

**Python Module Index** **61**

**Index** **63**

Extract the top level domain (TLD) from the URL given. List of TLD names is taken from Public Suffix.

Optionally raises exceptions on non-existing TLDs or silently fails (if `fail_silently` argument is set to True).

CHAPTER 1

# Prerequisites

- Python 3.7, 3.8, 3.9, 3.10 or 3.11.

CHAPTER 2

# Documentation

Documentation is available on Read the Docs.

CHAPTER 3

# Installation

Latest stable version on PyPI:

```
pip install tld
```

Or latest stable version from GitHub:

```
pip install https://github.com/barseghyanartur/tld/archive/stable.tar.gz
```

CHAPTER 4

# Usage examples

In addition to examples below, see the jupyter notebook workbook file.

## 4.1 Get the TLD name as string from the URL given

```
from tld import get_tld

get_tld("http://www.google.co.uk")
# 'co.uk'

get_tld("http://www.google.idontexist", fail_silently=True)
# None
```

## 4.2 Get the TLD as an object

```
from tld import get_tld

res = get_tld("http://some.subdomain.google.co.uk", as_object=True)

res
# 'co.uk'

res.subdomain
# 'some.subdomain'

res.domain
# 'google'

res.tld
# 'co.uk'
```

(continues on next page)

(continued from previous page)

```
res.fld
# 'google.co.uk'

res.parsed_url
# SplitResult(
#     scheme='http',
#     netloc='some.subdomain.google.co.uk',
#     path='',
#     query='',
#     fragment=''
# )
```

## 4.3 Get TLD name, ignoring the missing protocol

```
from tld import get_tld, get_fld

get_tld("www.google.co.uk", fix_protocol=True)
# 'co.uk'

get_fld("www.google.co.uk", fix_protocol=True)
# 'google.co.uk'
```

## 4.4 Return TLD parts as tuple

```
from tld import parse_tld

parse_tld('http://www.google.com')
# 'com', 'google', 'www'
```

## 4.5 Get the first level domain name as string from the URL given

```
from tld import get_fld

get_fld("http://www.google.co.uk")
# 'google.co.uk'

get_fld("http://www.google.idontexist", fail_silently=True)
# None
```

## 4.6 Check if some tld is a valid tld

```
from tld import is_tld

is_tld('co.uk)
```

(continues on next page)

(continued from previous page)

```
# True

is_tld('uk')
# True

is_tld('tld.doesnotexist')
# False

is_tld('www.google.com')
# False
```

CHAPTER 5

# Update the list of TLD names

To update/sync the tld names with the most recent versions run the following from your terminal:

```
update-tld-names
```

Or simply do:

```
from tld.utils import update_tld_names

update_tld_names()
```

Note, that this will update all registered TLD source parsers (not only the list of TLD names taken from Mozilla). In order to run the update for a single parser, append `uid` of that parser as argument.

```
update-tld-names mozilla
```

CHAPTER 6

# Custom TLD parsers

By default list of TLD names is taken from Mozilla. Parsing implemented in the `tld.utils.MozillaTLDSourceParser` class. If you want to use another parser, subclass the `tld.base.BaseTLDSourceParser`, provide `uid`, `source_url`, `local_path` and implement the `get_tld_names` method. Take the `tld.utils.MozillaTLDSourceParser` as a good example of such implementation. You could then use `get_tld` (as well as other `tld` module functions) as shown below:

```
from tld import get_tld
from some.module import CustomTLDSourceParser

get_tld(
    "http://www.google.co.uk",
    parser_class=CustomTLDSourceParser
)
```

# CHAPTER 7

## Custom list of TLD names

You could maintain your own custom version of the TLD names list (even multiple ones) and use them simultaneously with built in TLD names list.

You would then store them locally and provide a path to it as shown below:

```
from tld import get_tld
from tld.utils import BaseMozillaTLDSourceParser

class CustomBaseMozillaTLDSourceParser(BaseMozillaTLDSourceParser):

    uid: str = 'custom_mozilla'
    local_path: str = 'tests/res/effective_tld_names_custom.dat.txt'

get_tld(
    "http://www.foreverchild",
    parser_class=CustomBaseMozillaTLDSourceParser
)
# 'foreverchild'
```

Same goes for first level domain names:

```
from tld import get_fld

get_fld(
    "http://www.foreverchild",
    parser_class=CustomBaseMozillaTLDSourceParser
)
# 'www.foreverchild'
```

Note, that in both examples shown above, there the original TLD names file has been modified in the following way:

```
...
// ===BEGIN ICANN DOMAINS===

```

(continues on next page)

(continued from previous page)

```
// This one actually does not exist, added for testing purposes
foreverchild
...
```

CHAPTER 8

# Free up resources

To free up memory occupied by loading of custom TLD names, use `reset_tld_names` function with `tld_names_local_path` parameter.

```
from tld import get_tld, reset_tld_names

# Get TLD from a custom TLD names parser
get_tld(
    "http://www.foreverchild",
    parser_class=CustomBaseMozillaTLDSourceParser
)

# Free resources occupied by the custom TLD names list
reset_tld_names("tests/res/effective_tld_names_custom.dat.txt")
```

CHAPTER 9

# Troubleshooting

If somehow domain names listed here are not recognised, make sure you have the most recent version of TLD names in your virtual environment:

```
update-tld-names
```

To update TLD names list for a single parser, specify it as an argument:

```
update-tld-names mozilla
```

CHAPTER 10

# Testing

Simply type:

```
pytest
```

Or use tox:

```
tox
```

Or use tox to check specific env:

```
tox -e py39
```

CHAPTER 11

# Writing documentation

Keep the following hierarchy.

```
=====
title
=====

header
======

sub-header
----------

sub-sub-header
~~~~~~~~~~~~~~

sub-sub-sub-header
^^^^^^^^^^^^^^^^^^

sub-sub-sub-sub-header
++++++++++++++++++++++

sub-sub-sub-sub-sub-header
**************************
```

CHAPTER 12

# License

MPL-1.1 OR GPL-2.0-only OR LGPL-2.1-or-later

CHAPTER 13

# Support

For security issues contact me at the e-mail given in the *Author* section.

For overall issues, go to GitHub.

CHAPTER 14

# Author

Artur Barseghyan <artur.barseghyan@gmail.com>

CHAPTER 15

# Project documentation

Contents:

**Table of Contents**


- *tld*
  - *Prerequisites*
  - *Documentation*
  - *Installation*
  - *Usage examples*
    - *Get the TLD name **as string** from the URL given*
    - *Get the TLD as **an object***
    - *Get TLD name, **ignoring the missing protocol***
    - *Return TLD parts as tuple*
    - *Get the first level domain name **as string** from the URL given*
    - *Check if some tld is a valid tld*
  - *Update the list of TLD names*
  - *Custom TLD parsers*
  - *Custom list of TLD names*
  - *Free up resources*
  - *Troubleshooting*
  - *Testing*
  - *Writing documentation*

- *License*
- *Support*
- *Author*
- *Project documentation*
- *Indices and tables*


## 15.1 Security Policy

### 15.1.1 Reporting a Vulnerability

**Do not report security issues on GitHub!**

Please report security issues by emailing Artur Barseghyan <artur.barseghyan@gmail.com>.

### 15.1.2 Supported Versions

**Make sure to use the latest version.**

The tree most recent `tld` release series receive security support.

For example, during the development cycle leading to the release of `tld` 0.12.x, support will be provided for `tld` 0.11.x, 0.10.x and 0.9.x.

Upon the release of `tld` 0.13.x, security support for `tld` 0.9.x will end.

| Version | Supported |
|---|---|
| 0.12.x | Yes |
| 0.11.x | Yes |
| 0.10.x | Yes |
| 0.9.x | Yes |
| < 0.9 | No |

## 15.2 Contributor Covenant Code of Conduct

### 15.2.1 Our Pledge

We as members, contributors, and leaders pledge to make participation in our community a harassment-free experience for everyone, regardless of age, body size, visible or invisible disability, ethnicity, sex characteristics, gender identity and expression, level of experience, education, socio-economic status, nationality, personal appearance, race, religion, or sexual identity and orientation.

We pledge to act and interact in ways that contribute to an open, welcoming, diverse, inclusive, and healthy community.

### 15.2.2 Our Standards

Examples of behavior that contributes to a positive environment for our community include:

- Demonstrating empathy and kindness toward other people
- Being respectful of differing opinions, viewpoints, and experiences
- Giving and gracefully accepting constructive feedback
- Accepting responsibility and apologizing to those affected by our mistakes, and learning from the experience
- Focusing on what is best not just for us as individuals, but for the overall community

Examples of unacceptable behavior include:

- The use of sexualized language or imagery, and sexual attention or advances of any kind
- Trolling, insulting or derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or email address, without their explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

### 15.2.3 Enforcement Responsibilities

Community leaders are responsible for clarifying and enforcing our standards of acceptable behavior and will take appropriate and fair corrective action in response to any behavior that they deem inappropriate, threatening, offensive, or harmful.

Community leaders have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, and will communicate reasons for moderation decisions when appropriate.

### 15.2.4 Scope

This Code of Conduct applies within all community spaces, and also applies when an individual is officially representing the community in public spaces. Examples of representing our community include using an official e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event.

### 15.2.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported to the community leaders responsible for enforcement at artur.barseghyan@gmail.com. All complaints will be reviewed and investigated promptly and fairly.

All community leaders are obligated to respect the privacy and security of the reporter of any incident.

### 15.2.6 Enforcement Guidelines

Community leaders will follow these Community Impact Guidelines in determining the consequences for any action they deem in violation of this Code of Conduct:

#### 15.2.6.1 1. Correction

**Community Impact**: Use of inappropriate language or other behavior deemed unprofessional or unwelcome in the community.

**Consequence**: A private, written warning from community leaders, providing clarity around the nature of the violation and an explanation of why the behavior was inappropriate. A public apology may be requested.

#### 15.2.6.2 2. Warning

**Community Impact**: A violation through a single incident or series of actions.

**Consequence**: A warning with consequences for continued behavior. No interaction with the people involved, including unsolicited interaction with those enforcing the Code of Conduct, for a specified period of time. This includes avoiding interactions in community spaces as well as external channels like social media. Violating these terms may lead to a temporary or permanent ban.

#### 15.2.6.3 3. Temporary Ban

**Community Impact**: A serious violation of community standards, including sustained inappropriate behavior.

**Consequence**: A temporary ban from any sort of interaction or public communication with the community for a specified period of time. No public or private interaction with the people involved, including unsolicited interaction with those enforcing the Code of Conduct, is allowed during this period. Violating these terms may lead to a permanent ban.

#### 15.2.6.4 4. Permanent Ban

**Community Impact**: Demonstrating a pattern of violation of community standards, including sustained inappropriate behavior, harassment of an individual, or aggression toward or disparagement of classes of individuals.

**Consequence**: A permanent ban from any sort of public interaction within the community.

### 15.2.7 Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 2.0, available at https://www.contributor-covenant.org/version/2/0/code_of_conduct.html.

Community Impact Guidelines were inspired by Mozilla's code of conduct enforcement ladder.

For answers to common questions about this code of conduct, see the FAQ at https://www.contributor-covenant.org/faq. Translations are available at https://www.contributor-covenant.org/translations.

## 15.3 Contributor guidelines

### 15.3.1 Developer prerequisites

#### 15.3.1.1 pre-commit

Refer to pre-commit for installation instructions.

TL;DR:

```
pip install pipx --user  # Install pipx
pipx install pre-commit  # Install pre-commit
pre-commit install  # Install pre-commit hooks
```

Installing pre-commit will ensure you adhere to the project code quality standards.

### 15.3.2 Code standards

black, isort, ruff and doc8 will be automatically triggered by pre-commit. Still, if you want to run checks manually:

```
./scripts/black.sh
./scripts/doc8.sh
./scripts/isort.sh
./scripts/ruff.sh
```

### 15.3.3 Requirements

Requirements are compiled using pip-tools.

```
./scripts/compile_requirements.sh
```

### 15.3.4 Virtual environment

You are advised to work in virtual environment.

TL;DR:

```
python -m venv env
pip install -e .
pip install -r requirements/test.txt
```

### 15.3.5 Documentation

Check documentation.

### 15.3.6 Testing

Check testing.

If you introduce changes or fixes, make sure to test them locally using all supported environments. For that use tox.

```
tox
```

In any case, GitHub Actions will catch potential errors, but using tox speeds things up.

### 15.3.7 Pull requests

You can contribute to the project by making a pull request.

For example:

- To fix documentation typos.
- To improve documentation (for instance, to add new recipe or fix an existing recipe that doesn't seem to work).
- To improve performance.
- To introduce a new feature.

**General list to go through:**

- Does your change require documentation update?
- Does your change require update to tests?
- Does your change rely on third-party cloud based service? If so, please make sure it's added to tests that should be retried a couple of times. Example: `@pytest.mark.flaky(reruns=5)`.

**When fixing bugs (in addition to the general list):**

- Make sure to add regression tests.

**When adding a new feature (in addition to the general list):**

- Check the licenses of added dependencies carefully and make sure to list them in prerequisites.
- Make sure to update the documentation (check whether the installation, usage examples and prerequisites require changes).

### 15.3.8 Questions

Questions can be asked on GitHub discussions.

### 15.3.9 Issues

For reporting a bug or filing a feature request use GitHub issues.

**Do not report security issues on GitHub**. Check the support section.

## 15.4 Release history and notes

Sequence based identifiers are used for versioning (schema follows below):

```
major.minor[.revision]
```

- It's always safe to upgrade within the same minor version (for example, from 0.3 to 0.3.4).
- Minor version changes might be backwards incompatible. Read the release notes carefully before upgrading (for example, when upgrading from 0.3.4 to 0.4).
- All backwards incompatible changes are mentioned in this document.

### 15.4.1 0.13

2023-02-28

- Drop Python 2.7, 3.5 and 3.6 support. Minimum required version now is Python 3.7.

### 15.4.2 0.12.7

2023-02-01

- Make sure to fail silently on bad URL patterns.
- Tested against Python 3.11.
- Tested against Python 3.10.
- Updated bundled tld names.

### 15.4.3 0.12.6

2021-06-05

- Move `Registry` class from `tld.registry` to `tld.base`.
- Reformat code using `black`.
- Log information on updated resources of the `update_tld_names`.

### 15.4.4 0.12.5

2021-01-11

---

**Note:** Release dedicated to defenders of Armenia and Artsakh (Nagorno Karabakh) and all the victims of Turkish and Azerbaijani aggression.

---

- Fixed lower-cased *parsed_url* attributes (*SplitResult*) when getting tld as object (*as_object=True*).

### 15.4.5 0.12.4

2021-01-02

- Tested against Python 3.9.

### 15.4.6 0.12.3

2020-11-26

- Separate parsers for (a) public and private and (b) public only domains. This fixes a bug. If you want an old behaviour:

  The following code would raise exception in past.

```
from tld import get_tld

get_tld(
    'http://silly.cc.ua',
    search_private=False
)
```

  Now it would return *ua*.

```
get_tld(
    'http://silly.cc.ua',
    search_private=False
)
```

If you want old behavior, do as follows:

```
from tld.utils import MozillaTLDSourceParser

get_tld(
    'http://silly.cc.ua',
    search_private=False,
    parser_class=MozillaTLDSourceParser
)
```

Same goes for `get_fld`, `process_url`, `parse_tld` and `is_tld` functions.

### 15.4.7 0.12.2

2020-05-20

- Add mozilla license to dist.
- Fix MyPy issues.

### 15.4.8 0.12.1

2020-04-25

**Note:** In commemoration of Armenian Genocide.

- Correctly handling domain names ending with dot(s).

### 15.4.9 0.12

2020-04-19

- Use Public Suffix list instead of deprecated Mozilla's MXR.

### 15.4.10 0.11.11

2020-03-10

- Minor speed-ups, reduce memory usage.

### 15.4.11 0.11.10

2020-02-05

- Python 2.7 and 3.5 fixes.

### 15.4.12 0.11.9

2019-12-16

- Adding test TLDs list to the package.

### 15.4.13 0.11.8

2019-12-13

- Minor fixes in setup.py.

### 15.4.14 0.11.7

2019-12-13

---

**Note:** There have been no code changes since 0.11.2. The only change is that support for Python 2.7 and 3.5 has been added.

---

- Added support for Python 2.7.

### 15.4.15 0.11.6

2019-12-12

- Targeted releases for all supported Python versions.

### 15.4.16 0.11.5

2019-12-12

- Targeted releases for all supported Python versions.

### 15.4.17 0.11.4

2019-12-12

- Changed order of the releases (Python 3.6 and up come first, then Python 3.5).
- Make all distributions except Python 3.5 universal.

### 15.4.18 0.11.3

2019-12-12

- Added missing resources to the Python 3.5 release.

### 15.4.19 0.11.2

2019-12-12

- Bring back Python 3.5 support.

### 15.4.20 0.11.1

2019-12-11

- Minor speed ups.
- More on adding typing.

### 15.4.21 0.11

2019-12-09

---

**Note:** Since introduction of parser classes, usage of `NAMES_SOURCE_URL` and `NAMES_LOCAL_PATH` of the `tld.conf` module is deprecated. Also, `tld_names_local_path` and `tld_names_source_url` arguments are deprecated as well. If you want to customise things, implement your own parser (inherit from `BaseTLDSourceParser`).

---

- Drop support for Python versions prior to 3.6.
- Clean-up dependencies.
- Introduce parsers.
- Drop `tld_names_source_url` and `tld_names_local_path` introduced in the previous release.
- Minor speed-ups (including tests).

### 15.4.22 0.10

2019-11-27

---

**Note:** This is the last release to support Python 2.

---

- Make it possible to provide a custom path to the TLD names file.
- Make it possible to free up some resources occupied due to loading custom tld names by calling the `reset_tld_names` function with `tld_names_local_path` parameter.

### 15.4.23 0.9.8

2019-11-15

- Fix for occasional issue when some domains are not correctly recognised.

### 15.4.24 0.9.7

2019-10-30

**Note:** This release is dedicated to my newborn daughter. Happy birthday, my dear Ani.

- Handling urls that are only a TLD.
- Accepts already splitted URLs.
- Tested against Python 3.8.

### 15.4.25 0.9.6

2019-09-12

- Fix for update-tld-names returns a non-zero exit code on success (introduced with optimisations in 0.9.4).
- Minor tests improvements.

### 15.4.26 0.9.5

2019-09-11

- Tests improvements.

### 15.4.27 0.9.4

2019-09-11

- Optimisations in setup.py, tests and console scripts.
- Skip testing the update-tld-names functionality if no internet is available.

### 15.4.28 0.9.3

2019-04-05

- Added *is_tld* function.
- Docs updated.
- Upgrade test suite.

### 15.4.29 0.9.2

2019-01-10

- Fix an issue causing certain punycode TLDs to be deemed invalid.
- Tested against Python 3.7.
- Added tests for commands.
- Dropped Python 2.6 support.

- TLD source updated to the latest version.

### 15.4.30 0.9.1

2018-07-09

- Correctly handling nested TLDs.

### 15.4.31 0.9

2018-06-14

---

**Note:** This release contains backward incompatible changes. You should update your code.

The `active_only` option has been removed from `get_tld`, `get_fld` and `parse_url` functions. Update your code accordingly.

---

- Removed `active_only` option from `get_tld`, `get_fld` and `parse_url` functions.
- Correctly handling exceptions (!) in the original TLD list.
- Fixes in documentation.
- Added `parse_tld` function.
- Fixes the `python setup.py test` command.

### 15.4.32 0.8

2018-06-13

---

**Note:** This release contains backward incompatible changes. You should update your code.

Old `get_tld` functionality is moved to `get_fld` (first-level domain definition). The `as_object` argument (False by default) has been deprecated for `get_fld`.

```
res = get_tld("http://www.google.co.uk", as_object=True)
```

**Old behaviour**

```
In: res.domain
Out: 'google'

In: res.extension
Out: 'co.uk'

In: res.subdomain
Out: 'www'

In: res.suffix
Out: 'co.uk'

In: res.tld
Out: 'google.co.uk'
```

**New behaviour**

```
In: res.fld
Out: 'google.co.uk'

In: res.tld
Out: 'co.uk'

In: res.domain
Out: 'google'

In: res.subdomain
Out: 'www'
```

When used without `as_object` it returns `co.uk`.

**Recap**

If you have been happily using old version of `get_tld` function without `as_object` argument set to `True`, you might want to replace `get_tld` import with `get_fld` import:

```
# Old
from tld import get_tld
get_tld('http://google.co.uk')

# New
from tld import get_fld
get_fld('http://google.co.uk')
```

---

- Move to a Trie to match TLDs. This brings a speed up of 15-20%.
- It's now possible to search in public, private or all suffixes (old behaviour). Use `search_public` and `search_private` arguments accordingly. By default (to support old behavior), both are set to `True`.
- Correct TLD definitions.
- Domains like ******.xn–fiqs8s* are now recognized as well.
- Due to usage of `urlsplit` instead of `urlparse`, the initial list of TLDs is assembled quicker (a speed-up of 15-20%).
- Docs/ directory is included in source distribution tarball.
- More tests.

### 15.4.33 0.7.10

2018-04-07

- The `fix_protocol` argument respects protocol relative URLs.
- Change year in the license.
- Improved docstrings.
- TLD source updated to the latest version.

### 15.4.34 0.7.9

2017-05-02

- Added base path override for local .dat file.
- *python setup.py test* can used to execute the tests

### 15.4.35 0.7.8

2017-02-19

- Fix relative import in non-package for update-tls-names script. #15
- `get_tld` got a new argument `fix_protocol`, which fixes the missing protocol, having prepended "https" if missing or incorrect.

### 15.4.36 0.7.7

2017-02-09

- Tested against Python 3.5, 3.6 and PyPy.
- pep8 fixes.
- removed deprecated *tld.update* module. Use `update-tld-names` command instead.

### 15.4.37 0.7.6

2016-01-23

- Minor fixes.

### 15.4.38 0.7.5

2015-11-22

- Minor fixes.
- Updated tld names file to the latest version.

### 15.4.39 0.7.4

2015-09-24

- Exposed TLD initialization as `get_tld_names`.

### 15.4.40 0.7.3

2015-07-18

- Support for wheel packages.
- Fixed failure on some unicode domains.
- TLD source updated to the latest version.

- Documentation updated.

### 15.4.41 0.7.2

2014-09-28

- Minor fixes.

### 15.4.42 0.7.1

2014-09-23

- Force lower case of the URL for correct search.

### 15.4.43 0.7

2014-08-14

- Making it possible to obtain object instead of just extracting the TLD by setting the `as_object` argument of `get_tld` function to True.

### 15.4.44 0.6.4

2014-05-21

- Softened dependencies and lowered the `six` package version requirement to 1.4.0.
- Documentation improvements.

### 15.4.45 0.6.3

2013-12-05

- Speed up search

### 15.4.46 0.6.2

2013-12-03

- Fix for URLs with a port not handled correctly.
- Adding licenses.

### 15.4.47 0.6.1

2013-09-15

- Minor fixes.
- Credits added.

#### 15.4.48 0.6

2013-09-12

- Fixes for Python 3 (Windows encoding).

#### 15.4.49 0.5

2013-09-13

- Python 3 support added.

#### 15.4.50 0.4

2013-08-03

- Tiny code improvements.
- Tests added.

### 15.5 tld package

#### 15.5.1 Submodules

#### 15.5.2 tld.base module

**class** tld.base.**BaseTLDSourceParser**
Bases: object

Base TLD source parser.

**classmethod get_tld_names**(*fail_silently: bool = False*, *retry_count: int = 0*)
Get tld names.

**Parameters**

- **fail_silently** –
- **retry_count** –

**Returns**

**include_private = True**

**uid = None**

**classmethod update_tld_names**(*fail_silently: bool = False*) → bool
Update the local copy of the TLD file.

**Parameters** **fail_silently** –

**Returns**

**classmethod validate**()
Constructor.

**class** tld.base.**Registry**
Bases: type

**REGISTRY = {'mozilla': <class 'tld.utils.MozillaTLDSourceParser'>, 'mozilla_public_on**

**classmethod get**(*key: str*, *default: Optional[tld.base.BaseTLDSourceParser] = None*) → Optional[tld.base.BaseTLDSourceParser]

**classmethod items**() → ItemsView[str, tld.base.BaseTLDSourceParser]

**classmethod reset**() → None

### 15.5.3 tld.conf module

### 15.5.4 tld.defaults module

### 15.5.5 tld.exceptions module

**exception** tld.exceptions.**TldBadUrl**(*url*)
Bases: ValueError

TldBadUrl.

Supposed to be thrown when bad URL is given.

**exception** tld.exceptions.**TldDomainNotFound**(*domain_name*)
Bases: ValueError

TldDomainNotFound.

Supposed to be thrown when domain name is not found (didn't match) the local TLD policy.

**exception** tld.exceptions.**TldImproperlyConfigured**
Bases: Exception

TldImproperlyConfigured.

Supposed to be thrown when code is improperly configured. Typical use-case is when user tries to use *get_tld* function with both *search_public* and *search_private* set to False.

**exception** tld.exceptions.**TldIOError**
Bases: OSError

TldIOError.

Supposed to be thrown when problems with reading/writing occur.

### 15.5.6 tld.helpers module

tld.helpers.**project_dir**(*base: str*) → str
Project dir.

tld.helpers.**PROJECT_DIR**(*base: str*) → str
Project dir.

### 15.5.7 tld.registry module

**class** tld.registry.**Registry**
Bases: type

**REGISTRY = {'mozilla': <class 'tld.utils.MozillaTLDSourceParser'>, 'mozilla_public_on**

**classmethod get**(*key: str*, *default: Optional[tld.base.BaseTLDSourceParser] = None*) → Optional[tld.base.BaseTLDSourceParser]

**classmethod items**() → ItemsView[str, tld.base.BaseTLDSourceParser]

**classmethod reset**() → None

### 15.5.8 tld.result module

**class** tld.result.**Result**(*tld: str*, *domain: str*, *subdomain: str*, *parsed_url: urllib.parse.SplitResult*)

Bases: object

Container.

**domain**

**extension**

Alias of tld.

**Return str**

**fld**

First level domain.

**Returns**

**Return type** str

**parsed_url**

**subdomain**

**suffix**

Alias of tld.

**Return str**

**tld**

### 15.5.9 tld.trie module

**class** tld.trie.**Trie**

Bases: object

An adhoc Trie data structure to store tlds in reverse notation order.

**add**(*tld: str*, *private: bool = False*) → None

**class** tld.trie.**TrieNode**

Bases: object

Class representing a single Trie node.

**children**

**exception**

**leaf**

**private**

### 15.5.10 tld.utils module

**class** tld.utils.**BaseMozillaTLDSourceParser**
Bases: *tld.base.BaseTLDSourceParser*

**classmethod get_tld_names**(*fail_silently: bool = False*, *retry_count: int = 0*) → Optional[Dict[str, tld.trie.Trie]]
Parse.

**Parameters**

- **fail_silently** –
- **retry_count** –

**Returns**

tld.utils.**get_fld**(*url: Union[str, urllib.parse.SplitResult]*, *fail_silently: bool = False*, *fix_protocol: bool = False*, *search_public: bool = True*, *search_private: bool = True*, *parser_class: Type[tld.base.BaseTLDSourceParser] = None*, ***kwargs*) → Optional[str]
Extract the first level domain.

Extract the top level domain based on the mozilla's effective TLD names dat file. Returns a string. May throw `TldBadUrl` or `TldDomainNotFound` exceptions if there's bad URL provided or no TLD match found respectively.

**Parameters**

- **url** (*str | SplitResult*) – URL to get top level domain from.
- **fail_silently** (*bool*) – If set to True, no exceptions are raised and None is returned on failure.
- **fix_protocol** (*bool*) – If set to True, missing or wrong protocol is ignored (https is appended instead).
- **search_public** (*bool*) – If set to True, search in public domains.
- **search_private** (*bool*) – If set to True, search in private domains.
- **parser_class** –

**Returns** String with top level domain (if `as_object` argument is set to False) or a `tld.utils.Result` object (if `as_object` argument is set to True); returns None on failure.

**Return type** str

tld.utils.**get_tld**(*url: Union[str, urllib.parse.SplitResult]*, *fail_silently: bool = False*, *as_object: bool = False*, *fix_protocol: bool = False*, *search_public: bool = True*, *search_private: bool = True*, *parser_class: Type[tld.base.BaseTLDSourceParser] = None*) → Union[str, tld.result.Result, None]
Extract the top level domain.

Extract the top level domain based on the mozilla's effective TLD names dat file. Returns a string. May throw `TldBadUrl` or `TldDomainNotFound` exceptions if there's bad URL provided or no TLD match found respectively.

**Parameters**

- **url** (*str | SplitResult*) – URL to get top level domain from.
- **fail_silently** (*bool*) – If set to True, no exceptions are raised and None is returned on failure.

- **as_object** (*bool*) – If set to True, `tld.utils.Result` object is returned, `domain`, `suffix` and `tld` properties.
- **fix_protocol** (*bool*) – If set to True, missing or wrong protocol is ignored (https is appended instead).
- **search_public** (*bool*) – If set to True, search in public domains.
- **search_private** (*bool*) – If set to True, search in private domains.
- **parser_class** –

**Returns** String with top level domain (if `as_object` argument is set to False) or a `tld.utils.Result` object (if `as_object` argument is set to True); returns None on failure.

**Return type** str

`tld.utils.`**get_tld_names**(*fail_silently: bool = False*, *retry_count: int = 0*, *parser_class: Type[tld.base.BaseTLDSourceParser] = None*) → Dict[str, tld.trie.Trie]

Build the `tlds` list if empty. Recursive.

**Parameters**

- **fail_silently** (*bool*) – If set to True, no exceptions are raised and None is returned on failure.
- **retry_count** (*int*) – If greater than 1, we raise an exception in order to avoid infinite loops.
- **parser_class** (`BaseTLDSourceParser`) –

**Returns** List of TLD names

**Return type** obj:*tld.utils.Trie*

`tld.utils.`**get_tld_names_container**() → Dict[str, tld.trie.Trie]

Get container of all tld names.

**Returns**

**Rtype dict**

`tld.utils.`**is_tld**(*value: Union[str, urllib.parse.SplitResult]*, *search_public: bool = True*, *search_private: bool = True*, *parser_class: Type[tld.base.BaseTLDSourceParser] = None*) → bool

Check if given URL is tld.

**Parameters**

- **value** (*str*) – URL to get top level domain from.
- **search_public** (*bool*) – If set to True, search in public domains.
- **search_private** (*bool*) – If set to True, search in private domains.
- **parser_class** –

**Returns**

**Return type** bool

**class** `tld.utils.`**MozillaTLDSourceParser**

Bases: *tld.utils.BaseMozillaTLDSourceParser*

Mozilla TLD source.

**local_path = 'res/effective_tld_names.dat.txt'**

**source_url = 'https://publicsuffix.org/list/public_suffix_list.dat'**

**uid = 'mozilla'**

**class** tld.utils.**MozillaPublicOnlyTLDSourceParser**

Bases: *tld.utils.BaseMozillaTLDSourceParser*

Mozilla TLD source.

**include_private = False**

**local_path = 'res/effective_tld_names_public_only.dat.txt'**

**source_url = 'https://publicsuffix.org/list/public_suffix_list.dat?publiconly'**

**uid = 'mozilla_public_only'**

tld.utils.**parse_tld**(*url: Union[str, urllib.parse.SplitResult], fail_silently: bool = False, fix_protocol: bool = False, search_public: bool = True, search_private: bool = True, parser_class: Type[tld.base.BaseTLDSourceParser] = None*) → Union[Tuple[None, None, None], Tuple[str, str, str]]

Parse TLD into parts.

**Parameters**

- **url** –
- **fail_silently** –
- **fix_protocol** –
- **search_public** –
- **search_private** –
- **parser_class** –

**Returns** Tuple (tld, domain, subdomain)

**Return type** tuple

tld.utils.**pop_tld_names_container**(*tld_names_local_path: str*) → None

Remove TLD names container item.

**Parameters** **tld_names_local_path** –

**Returns**

tld.utils.**process_url**(*url: Union[str, urllib.parse.SplitResult], fail_silently: bool = False, fix_protocol: bool = False, search_public: bool = True, search_private: bool = True, parser_class: Type[tld.base.BaseTLDSourceParser] = <class 'tld.utils.MozillaTLDSourceParser'>*) → Union[Tuple[List[str], int, urllib.parse.SplitResult], Tuple[None, None, urllib.parse.SplitResult]]

Process URL.

**Parameters**

- **parser_class** –
- **url** –
- **fail_silently** –
- **fix_protocol** –
- **search_public** –
- **search_private** –

**Returns**

tld.utils.**reset_tld_names**(*tld_names_local_path: str = None*) → None
Reset the tld_names to empty value.

If tld_names_local_path is given, removes specified entry from tld_names instead.

**Parameters** **tld_names_local_path** (*str*) –

**Returns**

**class** tld.utils.**Result**(*tld: str*, *domain: str*, *subdomain: str*, *parsed_url: urllib.parse.SplitResult*)
Bases: object

Container.

**domain**

**extension**
Alias of tld.

**Return str**

**fld**
First level domain.

**Returns**

**Return type** str

**parsed_url**

**subdomain**

**suffix**
Alias of tld.

**Return str**

**tld**

tld.utils.**update_tld_names**
Update TLD names.

**Parameters**

- **fail_silently** –
- **parser_uid** –

**Returns**

tld.utils.**update_tld_names_cli**() → int
CLI wrapper for update_tld_names.

Since update_tld_names returns True on success, we need to negate the result to match CLI semantics.

tld.utils.**update_tld_names_container**(*tld_names_local_path: str*, *trie_obj: tld.trie.Trie*) → None
Update TLD Names container item.

**Parameters**

- **tld_names_local_path** –
- **trie_obj** –

**Returns**

### 15.5.11 Module contents

tld.**get_fld**(*url: Union[str, urllib.parse.SplitResult], fail_silently: bool = False, fix_protocol: bool = False, search_public: bool = True, search_private: bool = True, parser_class: Type[tld.base.BaseTLDSourceParser] = None, **kwargs*) → Optional[str]

Extract the first level domain.

Extract the top level domain based on the mozilla's effective TLD names dat file. Returns a string. May throw `TldBadUrl` or `TldDomainNotFound` exceptions if there's bad URL provided or no TLD match found respectively.

**Parameters**

- **url** (*str | SplitResult*) – URL to get top level domain from.
- **fail_silently** (*bool*) – If set to True, no exceptions are raised and None is returned on failure.
- **fix_protocol** (*bool*) – If set to True, missing or wrong protocol is ignored (https is appended instead).
- **search_public** (*bool*) – If set to True, search in public domains.
- **search_private** (*bool*) – If set to True, search in private domains.
- **parser_class** –

**Returns** String with top level domain (if `as_object` argument is set to False) or a `tld.utils.Result` object (if `as_object` argument is set to True); returns None on failure.

**Return type** str

tld.**get_tld**(*url: Union[str, urllib.parse.SplitResult], fail_silently: bool = False, as_object: bool = False, fix_protocol: bool = False, search_public: bool = True, search_private: bool = True, parser_class: Type[tld.base.BaseTLDSourceParser] = None*) → Union[str, tld.result.Result, None]

Extract the top level domain.

Extract the top level domain based on the mozilla's effective TLD names dat file. Returns a string. May throw `TldBadUrl` or `TldDomainNotFound` exceptions if there's bad URL provided or no TLD match found respectively.

**Parameters**

- **url** (*str | SplitResult*) – URL to get top level domain from.
- **fail_silently** (*bool*) – If set to True, no exceptions are raised and None is returned on failure.
- **as_object** (*bool*) – If set to True, `tld.utils.Result` object is returned, `domain`, `suffix` and `tld` properties.
- **fix_protocol** (*bool*) – If set to True, missing or wrong protocol is ignored (https is appended instead).
- **search_public** (*bool*) – If set to True, search in public domains.
- **search_private** (*bool*) – If set to True, search in private domains.
- **parser_class** –

**Returns** String with top level domain (if `as_object` argument is set to False) or a `tld.utils.Result` object (if `as_object` argument is set to True); returns None on failure.

**Return type** str

tld.**get_tld_names**(*fail_silently: bool = False*, *retry_count: int = 0*, *parser_class: Type[tld.base.BaseTLDSourceParser] = None*) → Dict[str, tld.trie.Trie]

Build the `tlds` list if empty. Recursive.

**Parameters**

- **fail_silently** (*bool*) – If set to True, no exceptions are raised and None is returned on failure.
- **retry_count** (*int*) – If greater than 1, we raise an exception in order to avoid infinite loops.
- **parser_class** (BaseTLDSourceParser) –

**Returns** List of TLD names

**Return type** obj:*tld.utils.Trie*

tld.**is_tld**(*value: Union[str, urllib.parse.SplitResult], search_public: bool = True, search_private: bool = True, parser_class: Type[tld.base.BaseTLDSourceParser] = None*) → bool

Check if given URL is tld.

**Parameters**

- **value** (*str*) – URL to get top level domain from.
- **search_public** (*bool*) – If set to True, search in public domains.
- **search_private** (*bool*) – If set to True, search in private domains.
- **parser_class** –

**Returns**

**Return type** bool

tld.**parse_tld**(*url: Union[str, urllib.parse.SplitResult], fail_silently: bool = False, fix_protocol: bool = False, search_public: bool = True, search_private: bool = True, parser_class: Type[tld.base.BaseTLDSourceParser] = None*) → Union[Tuple[None, None, None], Tuple[str, str, str]]

Parse TLD into parts.

**Parameters**

- **url** –
- **fail_silently** –
- **fix_protocol** –
- **search_public** –
- **search_private** –
- **parser_class** –

**Returns** Tuple (tld, domain, subdomain)

**Return type** tuple

**class** tld.**Result**(*tld: str*, *domain: str*, *subdomain: str*, *parsed_url: urllib.parse.SplitResult*)

Bases: `object`

Container.

**domain**

**extension**
Alias of `tld`.

**Return str**

**fld**
First level domain.

**Returns**

**Return type** str

**parsed_url**

**subdomain**

**suffix**
Alias of `tld`.

**Return str**

**tld**

`tld.`**update_tld_names**
Update TLD names.

**Parameters**

- **fail_silently** –
- **parser_uid** –

**Returns**

CHAPTER 16

# Indices and tables

- genindex
- modindex
- search

# Python Module Index

## t

tld, 55
tld.base, 48
tld.conf, 49
tld.defaults, 49
tld.exceptions, 49
tld.helpers, 49
tld.registry, 49
tld.result, 50
tld.trie, 50
tld.utils, 51

# Index


## A

add() (*tld.trie.Trie method*), 50

## B

BaseMozillaTLDSourceParser (*class in tld.utils*), 51
BaseTLDSourceParser (*class in tld.base*), 48

## C

children (*tld.trie.TrieNode attribute*), 50

## D

domain (*tld.Result attribute*), 56
domain (*tld.result.Result attribute*), 50
domain (*tld.utils.Result attribute*), 54

## E

exception (*tld.trie.TrieNode attribute*), 50
extension (*tld.Result attribute*), 56
extension (*tld.result.Result attribute*), 50
extension (*tld.utils.Result attribute*), 54

## F

fld (*tld.Result attribute*), 57
fld (*tld.result.Result attribute*), 50
fld (*tld.utils.Result attribute*), 54

## G

get() (*tld.base.Registry class method*), 49
get() (*tld.registry.Registry class method*), 49
get_fld() (*in module tld*), 55
get_fld() (*in module tld.utils*), 51
get_tld() (*in module tld*), 55
get_tld() (*in module tld.utils*), 51
get_tld_names() (*in module tld*), 55
get_tld_names() (*in module tld.utils*), 52
get_tld_names() (*tld.base.BaseTLDSourceParser class method*), 48
get_tld_names() (*tld.utils.BaseMozillaTLDSourceParser class method*), 51
get_tld_names_container() (*in module tld.utils*), 52

## I

include_private (*tld.base.BaseTLDSourceParser attribute*), 48
include_private (*tld.utils.MozillaPublicOnlyTLDSourceParser attribute*), 53
is_tld() (*in module tld*), 56
is_tld() (*in module tld.utils*), 52
items() (*tld.base.Registry class method*), 49
items() (*tld.registry.Registry class method*), 50

## L

leaf (*tld.trie.TrieNode attribute*), 50
local_path (*tld.utils.MozillaPublicOnlyTLDSourceParser attribute*), 53
local_path (*tld.utils.MozillaTLDSourceParser attribute*), 52

## M

MozillaPublicOnlyTLDSourceParser (*class in tld.utils*), 53
MozillaTLDSourceParser (*class in tld.utils*), 52

## P

parse_tld() (*in module tld*), 56
parse_tld() (*in module tld.utils*), 53
parsed_url (*tld.Result attribute*), 57
parsed_url (*tld.result.Result attribute*), 50
parsed_url (*tld.utils.Result attribute*), 54
pop_tld_names_container() (*in module tld.utils*), 53
private (*tld.trie.TrieNode attribute*), 50
process_url() (*in module tld.utils*), 53
PROJECT_DIR() (*in module tld.helpers*), 49
project_dir() (*in module tld.helpers*), 49

## R


Registry (*class in tld.base*), 48
Registry (*class in tld.registry*), 49
REGISTRY (*tld.base.Registry attribute*), 48
REGISTRY (*tld.registry.Registry attribute*), 49
reset() (*tld.base.Registry class method*), 49
reset() (*tld.registry.Registry class method*), 50
reset_tld_names() (*in module tld.utils*), 54
Result (*class in tld*), 56
Result (*class in tld.result*), 50
Result (*class in tld.utils*), 54


## S


source_url (*tld.utils.MozillaPublicOnlyTLDSourceParser attribute*), 53
source_url (*tld.utils.MozillaTLDSourceParser attribute*), 52
subdomain (*tld.Result attribute*), 57
subdomain (*tld.result.Result attribute*), 50
subdomain (*tld.utils.Result attribute*), 54
suffix (*tld.Result attribute*), 57
suffix (*tld.result.Result attribute*), 50
suffix (*tld.utils.Result attribute*), 54


## T


tld (*module*), 55
tld (*tld.Result attribute*), 57
tld (*tld.result.Result attribute*), 50
tld (*tld.utils.Result attribute*), 54
tld.base (*module*), 48
tld.conf (*module*), 49
tld.defaults (*module*), 49
tld.exceptions (*module*), 49
tld.helpers (*module*), 49
tld.registry (*module*), 49
tld.result (*module*), 50
tld.trie (*module*), 50
tld.utils (*module*), 51
TldBadUrl, 49
TldDomainNotFound, 49
TldImproperlyConfigured, 49
TldIOError, 49
Trie (*class in tld.trie*), 50
TrieNode (*class in tld.trie*), 50


## U


uid (*tld.base.BaseTLDSourceParser attribute*), 48
uid (*tld.utils.MozillaPublicOnlyTLDSourceParser attribute*), 53
uid (*tld.utils.MozillaTLDSourceParser attribute*), 53
update_tld_names (*in module tld*), 57
update_tld_names (*in module tld.utils*), 54
update_tld_names() (*tld.base.BaseTLDSourceParser class method*), 48
update_tld_names_cli() (*in module tld.utils*), 54
update_tld_names_container() (*in module tld.utils*), 54


## V


validate() (*tld.base.BaseTLDSourceParser class method*), 48