subliminal Documentation

Release 2.1.0.dev

Antoine Bertin

Nov 28, 2018

Contents

Subliminal is a python 2.7+ library to search and download subtitles. It comes with an easy to use yet powerful CLI (command-line interface) suitable for direct use or cron jobs.

CHAPTER 1

Documentation

1.1 Usage

1.1.1 CLI

Download English subtitles:

```
$ subliminal download -l en The.Big.Bang.Theory.S05E18.HDTV.x264-LOL.mp4
Collecting videos  [####################################]  100%
1 video collected / 0 video ignored
Downloading subtitles  [####################################]  100%
Downloaded 1 subtitle
```

Warning: For cron usage, make sure to specify a maximum age (with `--age`) so subtitles are searched for recent videos only. Otherwise you will get banned from the providers for abuse due to too many requests. If subliminal didn't find subtitles for an old video, it's unlikely it will find subtitles for that video ever anyway.

See *CLI* for more details on the available commands and options.

1.1.2 Nautilus/Nemo integration

See the dedicated project page for more information.

1.1.3 High level API

You can call subliminal in many different ways depending on how much control you want over the process. For most use cases, you can stick to the standard API.

Common

Let's start by importing subliminal:

```
>>> import os
>>> from babelfish import *
>>> from subliminal import *
```

Before going further, there are a few things to know about subliminal.

Video

The *Movie* and *Episode* classes represent a video, existing or not. You can create a video by name (or path) with *Video.fromname*, use *scan_video()* on an existing file path to get even more information about the video or use *scan_videos()* on an existing directory path to scan a whole directory for videos.

```
>>> video = Video.fromname('The.Big.Bang.Theory.S05E18.HDTV.x264-LOL.mp4')
>>> video
<Episode ['The Big Bang Theory', 5x18]>
```

Here video information was guessed based on the name of the video, you can access some video attributes:

```
>>> video.video_codec
'H.264'
>>> video.release_group
'LOL'
```

Configuration

Before proceeding to listing and downloading subtitles, you need to configure the cache. Subliminal uses a cache to reduce repeated queries to providers and improve overall performance with no impact on search quality. For the sake of this example, we're going to use a memory backend.

```
>>> my_region = region.configure('dogpile.cache.memory')
```

Warning: Choose a cache that fits your application and prefer persistent over volatile backends. The `file` backend is usually a good choice. See dogpile.cache's documentation for more details on backends.

Now that we're done with the basics, let's have some *real* fun.

Listing

To list subtitles, subliminal provides a *list_subtitles()* function that will return all found subtitles:

```
>>> subtitles = list_subtitles([video], {Language('hun')}, providers=['podnapisi'])
>>> subtitles[video]
[<PodnapisiSubtitle 'ZtAW' [hu]>, <PodnapisiSubtitle 'ONAW' [hu]>]
```

Note: As you noticed, all parameters are iterables but only contain one item which means you can deal with a lot of videos, languages and providers at the same time. For the sake of this example, we filter providers to use only one, pass `providers=None` (default) to search on all providers.

Scoring

It's usual you have multiple candidates for subtitles. To help you chose which one to download, subliminal can compare them to the video and tell you exactly what matches with *get_matches()*:

```
>>> for s in subtitles[video]:
...     sorted(s.get_matches(video))
['episode', 'release_group', 'season', 'series', 'source', 'video_codec', 'year']
['episode', 'season', 'series', 'source', 'year']
```

And then compute a score with those matches with *compute_score()*:

```
>>> for s in subtitles[video]:
...     {s: compute_score(s, video)}
{<PodnapisiSubtitle 'ZtAW' [hu]>: 354}
{<PodnapisiSubtitle 'ONAW' [hu]>: 337}
```

Now you should have a better idea about which one you should choose.

Downloading

We can settle on the first subtitle and download its content using *download_subtitles()*:

```
>>> subtitle = subtitles[video][0]
>>> subtitle.content is None
True
>>> download_subtitles([subtitle])
>>> subtitle.content.split(b'\n')[2]
b'Elszaladok a boltba'
```

If you want a string instead of bytes, you can access decoded content with the *text* property:

```
>>> subtitle.text.split('\n')[3]
'néhány apróságért.'
```

Downloading best subtitles

Downloading best subtitles is what you want to do in almost all cases, as a shortcut for listing, scoring and downloading you can use *download_best_subtitles()*:

```
>>> best_subtitles = download_best_subtitles([video], {Language('hun')}, providers=[
↪'podnapisi'])
>>> best_subtitles[video]
[<PodnapisiSubtitle 'ZtAW' [hu]>]
>>> best_subtitle = best_subtitles[video][0]
>>> best_subtitle.content.split(b'\n')[2]
b'Elszaladok a boltba'
```

We end up with the same subtitle but with one line of code. Neat.

Save

We got ourselves a nice subtitle, now we can save it on the file system using *save_subtitles()*:

```
>>> save_subtitles(video, [best_subtitle])
[<PodnapisiSubtitle 'ZtAW' [hu]>]
>>> os.listdir()
['The.Big.Bang.Theory.S05E18.HDTV.x264-LOL.hu.srt']
```

1.2 How it works

1.2.1 Providers

Subliminal uses multiple providers to give users a vast choice and have a better chance to find the best matching subtitles. Current supported providers are:

- Addic7ed
- LegendasTV
- NapiProjekt
- OpenSubtitles
- Podnapisi
- Shooter
- TheSubDB
- TvSubtitles

Providers all inherit the same *Provider* base class and thus share the same API. They are registered on the `subliminal.providers` entry point and are exposed through the *provider_manager* for easy access.

To work with multiple providers seamlessly, the *ProviderPool* exposes the same API but distributes it to its providers and *AsyncProviderPool* does it asynchronously.

1.2.2 Scoring

Rating subtitles and comparing them is probably the most difficult part and this is where subliminal excels with its powerful scoring algorithm.

Using guessit and enzyme, subliminal extracts properties of the video and match them with the properties of the subtitles found with the providers.

Equations in *subliminal.score* give a score to each property (called a match). The more matches the video and the subtitle have, the higher the score computed with *compute_score()* gets.

1.2.3 Libraries

Various libraries are used by subliminal and are key to its success:

- guessit to guess information from filenames
- enzyme to detect embedded subtitles in videos and read other video metadata
- babelfish to work with languages

- requests to make human readable HTTP requests
- BeautifulSoup to parse HTML and XML
- dogpile.cache to cache intermediate search results
- stevedore to manage the provider entry point
- chardet to detect subtitles' encoding
- pysrt to validate downloaded subtitles

1.3 CLI

1.3.1 subliminal

```
Usage: subliminal [OPTIONS] COMMAND [ARGS]...

  Subtitles, faster than your thoughts.

Options:
  --addic7ed USERNAME PASSWORD    Addic7ed configuration.
  --legendastv USERNAME PASSWORD  LegendasTV configuration.
  --opensubtitles USERNAME PASSWORD
                                  OpenSubtitles configuration.
  --cache-dir DIRECTORY           Path to the cache directory.  [default:
                                  /home/docs/.cache/subliminal]
  --debug                         Print useful information for debugging
                                  subliminal and for reporting bugs.
  --version                       Show the version and exit.
  --help                          Show this message and exit.

Commands:
  cache     Cache management.
  download  Download best subtitles.

  Suggestions and bug reports are greatly appreciated:
  https://github.com/Diaoul/subliminal/
```

1.3.2 subliminal download

```
Usage: subliminal download [OPTIONS] PATH...

  Download best subtitles.

  PATH can be an directory containing videos, a video file path or a video
  file name. It can be used multiple times.

  If an existing subtitle is detected (external or embedded) in the correct
  language, the download is skipped for the associated video.

Options:
  -l, --language LANGUAGE         Language as IETF code, e.g. en, pt-BR (can
                                  be used multiple times).  [required]
  -p, --provider␣
↪[addic7ed|legendastv|opensubtitles|podnapisi|shooter|thesubdb|tvsubtitles]
```

(continues on next page)

(continued from previous page)

```
                                Provider to use (can be used multiple
                                times).
  -r, --refiner [metadata|omdb|tvdb]
                                Refiner to use (can be used multiple times).
  -a, --age AGE                 Filter videos newer than AGE, e.g. 12h,
                                1w2d.
  -d, --directory DIR           Directory where to save subtitles, default
                                is next to the video file.
  -e, --encoding ENC            Subtitle file encoding, default is to
                                preserve original encoding.
  -s, --single                  Save subtitle without language code in the
                                file name, i.e. use .srt extension. Do not
                                use this unless your media player requires
                                it.
  -f, --force                   Force download even if a subtitle already
                                exist.
  -hi, --hearing-impaired       Prefer hearing impaired subtitles.
  -m, --min-score INTEGER RANGE Minimum score for a subtitle to be
                                downloaded (0 to 100).
  -w, --max-workers INTEGER RANGE
                                Maximum number of threads to use.
  -z, --archives / -Z, --no-archives
                                Scan archives for videos (supported
                                extensions: .rar).  [default: True]
  -v, --verbose                 Increase verbosity.
  --help                        Show this message and exit.
```

1.3.3 subliminal cache

```
Usage: subliminal cache [OPTIONS]

  Cache management.

Options:
  --clear-subliminal  Clear subliminal's cache. Use this ONLY if your cache is
                      corrupted or if you experience issues.
  --help              Show this message and exit.
```

1.4 Provider Guide

This guide is going to explain how to add a *Provider* to subliminal. You are encouraged to take a look at the existing providers, it can be a nice base to start your own provider.

1.4.1 Requirements

When starting a provider you should be able to answer to the following questions:

- What languages does my provider support?
- What are the language codes for the supported languages?
- Does my provider deliver subtitles for episodes? for movies?

- Does my provider require a video hash?

Each response of these questions will help you set the correct attributes for your *Provider*.

1.4.2 Video Validation

Not all providers deliver subtitles for *Episode*. Some may require a hash. The *check()* method does validation against a *Video* object and will return *False* if the given *Video* isn't suitable. If you're not happy with the default implementation, you can override it.

1.4.3 Configuration

API keys must not be configurable by the user and must remain linked to subliminal. Hence they must be written in the provider module.

Per-user authentication is allowed and must be configured at instantiation as keyword arguments. Configuration will be done by the user through the *provider_configs* argument of the *list_subtitles()* and *download_best_subtitles()* functions. No network operation must be done during instantiation, only configuration. Any error in the configuration must raise a *ConfigurationError*.

Beyond this point, if an error occurs, a generic *ProviderError* exception must be raised. You can also use more explicit exception classes *AuthenticationError* and *DownloadLimitExceeded*.

1.4.4 Initialization / Termination

Actual authentication operations must take place in the *initialize()* method. If you need anything to be executed when the provider isn't used anymore like logout, use *terminate()*.

1.4.5 Caching policy

To save bandwidth and improve querying time, intermediate data should be cached when possible. Typical use case is when a query to retrieve show ids is required prior to the query to actually search for subtitles. In that case the function that gets the show id from the show name must be cached. Expiration time should be *SHOW_EXPIRATION_TIME* for shows and *EPISODE_EXPIRATION_TIME* for episodes.

1.4.6 Language

To be able to handle various language codes, subliminal makes use of babelfish Language and converters. You must set the attribute *languages* with a set of supported *Language*.

If you cannot find a suitable converter for your provider, you can make one of your own.

1.4.7 Querying

The *query()* method parameters must include all aspects of provider's querying with primary types.

1.4.8 Subtitle

A custom *Subtitle* subclass must be created to represent a subtitle from the provider. It must have relevant attributes that can be used to compute the matches of the subtitle against a *Video* object.

1.4.9 Score computation

To be able to compare subtitles coming from different providers between them, the *get_matches()* method must be implemented.

1.4.10 Unittesting

All possible uses of *query()*, *list_subtitles()* and *download_subtitle()* methods must have integration tests. Use vcrpy for recording and playback of network activity. Other functions must be unittested. If necessary, you can use `unittest.mock` to mock some functions.

CHAPTER 2

API Documentation

If you are looking for information on a specific function, class or method, this part of the documentation is for you.

2.1 Core

subliminal.core.**ARCHIVE_EXTENSIONS**
Supported archive extensions

class subliminal.core.**ProviderPool**(*providers=None*, *provider_configs=None*)
A pool of providers with the same API as a single *Provider*.

It has a few extra features:

- Lazy loads providers when needed and supports the *with* statement to *terminate()* the providers on exit.
- Automatically discard providers on failure.

Parameters

- **providers** (*list*) – name of providers to use, if not all.
- **provider_configs** (*dict*) – provider configuration as keyword arguments per provider name to pass when instanciating the *Provider*.

providers = None
Name of providers to use

provider_configs = None
Provider configuration

initialized_providers = None
Initialized providers

discarded_providers = None
Discarded providers

list_subtitles_provider(*provider*, *video*, *languages*)
List subtitles with a single provider.

The video and languages are checked against the provider.

Parameters

- **provider** (*str*) – name of the provider.
- **video** (*Video*) – video to list subtitles for.
- **languages** (set of Language) – languages to search for.

Returns found subtitles.

Return type list of *Subtitle* or None

list_subtitles(*video*, *languages*)
List subtitles.

Parameters

- **video** (*Video*) – video to list subtitles for.
- **languages** (set of Language) – languages to search for.

Returns found subtitles.

Return type list of *Subtitle*

download_subtitle(*subtitle*)
Download *subtitle*'s *content*.

Parameters **subtitle** (*Subtitle*) – subtitle to download.

Returns *True* if the subtitle has been successfully downloaded, *False* otherwise.

Return type bool

download_best_subtitles(*subtitles*, *video*, *languages*, *min_score=0*, *hearing_impaired=False*, *only_one=False*, *compute_score=None*)
Download the best matching subtitles.

Parameters

- **subtitles** (list of *Subtitle*) – the subtitles to use.
- **video** (*Video*) – video to download subtitles for.
- **languages** (set of Language) – languages to download.
- **min_score** (*int*) – minimum score for a subtitle to be downloaded.
- **hearing_impaired** (*bool*) – hearing impaired preference.
- **only_one** (*bool*) – download only one subtitle, not one per language.
- **compute_score** – function that takes *subtitle* and *video* as positional arguments, *hearing_impaired* as keyword argument and returns the score.

Returns downloaded subtitles.

Return type list of *Subtitle*

terminate()
Terminate all the *initialized_providers*.

class subliminal.core.**AsyncProviderPool**(*max_workers=None*, **args*, ***kwargs*)
Subclass of *ProviderPool* with asynchronous support for *list_subtitles()*.

Parameters **max_workers** (*int*) – maximum number of threads to use. If *None*, *max_workers* will be set to the number of *providers*.

max_workers = None
Maximum number of threads to use

list_subtitles_provider(*provider*, *video*, *languages*)
List subtitles with a single provider.

The video and languages are checked against the provider.

Parameters

- **provider** (*str*) – name of the provider.
- **video** (*Video*) – video to list subtitles for.
- **languages** (set of Language) – languages to search for.

Returns found subtitles.

Return type list of *Subtitle* or None

list_subtitles(*video*, *languages*)
List subtitles.

Parameters

- **video** (*Video*) – video to list subtitles for.
- **languages** (set of Language) – languages to search for.

Returns found subtitles.

Return type list of *Subtitle*

subliminal.core.**check_video**(*video*, *languages=None*, *age=None*, *undefined=False*)
Perform some checks on the *video*.

All the checks are optional. Return *False* if any of this check fails:

- *languages* already exist in *video*'s *subtitle_languages*.
- *video* is older than *age*.
- *video* has an *undefined* language in *subtitle_languages*.

Parameters

- **video** (*Video*) – video to check.
- **languages** (set of Language) – desired languages.
- **age** (*datetime.timedelta*) – maximum age of the video.
- **undefined** (*bool*) – fail on existing undefined language.

Returns *True* if the video passes the checks, *False* otherwise.

Return type bool

subliminal.core.**search_external_subtitles**(*path*, *directory=None*)
Search for external subtitles from a video *path* and their associated language.

Unless *directory* is provided, search will be made in the same directory as the video file.

Parameters

- **path** (*str*) – path to the video.
- **directory** (*str*) – directory to search for subtitles.

Returns found subtitles with their languages.

Return type dict

subliminal.core.**scan_video**(*path*)
Scan a video from a *path*.

Parameters **path** (*str*) – existing path to the video.

Returns the scanned video.

Return type *Video*

subliminal.core.**scan_archive**(*path*)
Scan an archive from a *path*.

Parameters **path** (*str*) – existing path to the archive.

Returns the scanned video.

Return type *Video*

subliminal.core.**scan_videos**(*path*, *age=None*, *archives=True*)
Scan *path* for videos and their subtitles.

See *refine()* to find additional information for the video.

Parameters

- **path** (*str*) – existing directory path to scan.
- **age** (*datetime.timedelta*) – maximum age of the video or archive.
- **archives** (*bool*) – scan videos in archives.

Returns the scanned videos.

Return type list of *Video*

subliminal.core.**refine**(*video*, *episode_refiners=None*, *movie_refiners=None*, ***kwargs*)
Refine a video using *Refiners*.

Note: Exceptions raised in refiners are silently passed and logged.

Parameters

- **video** (*Video*) – the video to refine.
- **episode_refiners** (*tuple*) – refiners to use for episodes.
- **movie_refiners** (*tuple*) – refiners to use for movies.
- ****kwargs** – additional parameters for the *refine()* functions.

subliminal.core.**list_subtitles**(*videos*, *languages*, *pool_class=<class 'subliminal.core.ProviderPool'>*, ***kwargs*)
List subtitles.

The *videos* must pass the *languages* check of *check_video()*.

Parameters

- **videos** (set of *Video*) – videos to list subtitles for.
- **languages** (set of Language) – languages to search for.
- **pool_class** (*ProviderPool*, *AsyncProviderPool* or similar) – class to use as provider pool.
- ****kwargs** – additional parameters for the provided *pool_class* constructor.

Returns found subtitles per video.

Return type dict of *Video* to list of *Subtitle*

subliminal.core.**download_subtitles**(*subtitles*, *pool_class=<class 'subliminal.core.ProviderPool'>*, ***kwargs*)

Download *content* of *subtitles*.

Parameters

- **subtitles** (list of *Subtitle*) – subtitles to download.
- **pool_class** (*ProviderPool*, *AsyncProviderPool* or similar) – class to use as provider pool.
- ****kwargs** – additional parameters for the provided *pool_class* constructor.

subliminal.core.**download_best_subtitles**(*videos*, *languages*, *min_score=0*, *hearing_impaired=False*, *only_one=False*, *compute_score=None*, *pool_class=<class 'subliminal.core.ProviderPool'>*, ***kwargs*)

List and download the best matching subtitles.

The *videos* must pass the *languages* and *undefined* (*only_one*) checks of *check_video()*.

Parameters

- **videos** (set of *Video*) – videos to download subtitles for.
- **languages** (set of Language) – languages to download.
- **min_score** (*int*) – minimum score for a subtitle to be downloaded.
- **hearing_impaired** (*bool*) – hearing impaired preference.
- **only_one** (*bool*) – download only one subtitle, not one per language.
- **compute_score** – function that takes *subtitle* and *video* as positional arguments, *hearing_impaired* as keyword argument and returns the score.
- **pool_class** (*ProviderPool*, *AsyncProviderPool* or similar) – class to use as provider pool.
- ****kwargs** – additional parameters for the provided *pool_class* constructor.

Returns downloaded subtitles per video.

Return type dict of *Video* to list of *Subtitle*

subliminal.core.**save_subtitles**(*video*, *subtitles*, *single=False*, *directory=None*, *encoding=None*)

Save subtitles on filesystem.

Subtitles are saved in the order of the list. If a subtitle with a language has already been saved, other subtitles with the same language are silently ignored.

The extension used is *.lang.srt* by default or *.srt* is *single* is *True*, with *lang* being the IETF code for the *language* of the subtitle.

Parameters

- **video** (*Video*) – video of the subtitles.
- **subtitles** (list of *Subtitle*) – subtitles to save.
- **single** (*bool*) – save a single subtitle, default is to save one subtitle per language.
- **directory** (*str*) – path to directory where to save the subtitles, default is next to the video.
- **encoding** (*str*) – encoding in which to save the subtitles, default is to keep original encoding.

Returns the saved subtitles

Return type list of *Subtitle*

2.2 Video

subliminal.video.**VIDEO_EXTENSIONS**
Video extensions

class subliminal.video.**Video**(*name*, *source=None*, *release_group=None*, *resolution=None*, *video_codec=None*, *audio_codec=None*, *imdb_id=None*, *hashes=None*, *size=None*, *subtitle_languages=None*)
Base class for videos.

Represent a video, existing or not.

Parameters

- **name** (*str*) – name or path of the video.
- **source** (*str*) – source of the video (HDTV, Web, Blu-ray, ...).
- **release_group** (*str*) – release group of the video.
- **resolution** (*str*) – resolution of the video stream (480p, 720p, 1080p or 1080i).
- **video_codec** (*str*) – codec of the video stream.
- **audio_codec** (*str*) – codec of the main audio stream.
- **imdb_id** (*str*) – IMDb id of the video.
- **hashes** (*dict*) – hashes of the video file by provider names.
- **size** (*int*) – size of the video file in bytes.
- **subtitle_languages** (*set*) – existing subtitle languages.

name = None
Name or path of the video

source = None
Source of the video (HDTV, Web, Blu-ray, ...)

release_group = None
Release group of the video

resolution = None
Resolution of the video stream (480p, 720p, 1080p or 1080i)

video_codec = None
Codec of the video stream

audio_codec = None
Codec of the main audio stream

imdb_id = None
IMDb id of the video

hashes = None
Hashes of the video file by provider names

size = None
Size of the video file in bytes

subtitle_languages = None
Existing subtitle languages

exists
Test whether the video exists

age
Age of the video

classmethod fromguess(*name*, *guess*)
Create an *Episode* or a *Movie* with the given *name* based on the *guess*.

Parameters

- **name** (*str*) – name of the video.
- **guess** (*dict*) – guessed data.

Raise ValueError if the *type* of the *guess* is invalid

classmethod fromname(*name*)
Shortcut for *fromguess()* with a *guess* guessed from the *name*.

Parameters **name** (*str*) – name of the video.

class subliminal.video.**Episode**(*name*, *series*, *season*, *episode*, *title=None*, *year=None*, *original_series=True*, *tvdb_id=None*, *series_tvdb_id=None*, *series_imdb_id=None*, *alternative_series=None*, ***kwargs*)
Episode *Video*.

Parameters

- **series** (*str*) – series of the episode.
- **season** (*int*) – season number of the episode.
- **episode** (*int*) – episode number of the episode.
- **title** (*str*) – title of the episode.
- **year** (*int*) – year of the series.
- **original_series** (*bool*) – whether the series is the first with this name.
- **tvdb_id** (*int*) – TVDB id of the episode.
- **alternative_series** (*list*) – alternative names of the series
- ****kwargs** – additional parameters for the *Video* constructor.

series = None
Series of the episode

season = None
Season number of the episode

episode = None
Episode number of the episode

title = None
Title of the episode

year = None
Year of series

original_series = None
The series is the first with this name

tvdb_id = None
TVDB id of the episode

series_tvdb_id = None
TVDB id of the series

series_imdb_id = None
IMDb id of the series

alternative_series = None
Alternative names of the series

classmethod fromguess(*name*, *guess*)
Create an *Episode* or a *Movie* with the given *name* based on the *guess*.

Parameters

- **name** (*str*) – name of the video.
- **guess** (*dict*) – guessed data.

Raise ValueError if the *type* of the *guess* is invalid

classmethod fromname(*name*)
Shortcut for *fromguess()* with a *guess* guessed from the *name*.

Parameters **name** (*str*) – name of the video.

class subliminal.video.**Movie**(*name*, *title*, *year=None*, *alternative_titles=None*, ***kwargs*)
Movie *Video*.

Parameters

- **title** (*str*) – title of the movie.
- **year** (*int*) – year of the movie.
- **alternative_titles** (*list*) – alternative titles of the movie
- ****kwargs** – additional parameters for the *Video* constructor.

title = None
Title of the movie

year = None
Year of the movie

alternative_titles = None
Alternative titles of the movie

classmethod fromguess(*name*, *guess*)
Create an *Episode* or a *Movie* with the given *name* based on the *guess*.

Parameters

- **name** (*str*) – name of the video.
- **guess** (*dict*) – guessed data.

Raise ValueError if the *type* of the *guess* is invalid

classmethod fromname(*name*)
Shortcut for *fromguess()* with a *guess* guessed from the *name*.

Parameters **name** (*str*) – name of the video.

2.3 Subtitle

subliminal.subtitle.**SUBTITLE_EXTENSIONS**
Subtitle extensions

class subliminal.subtitle.**Subtitle**(*language*, *hearing_impaired=False*, *page_link=None*, *encoding=None*)
Base class for subtitle.

Parameters

- **language** (Language) – language of the subtitle.
- **hearing_impaired** (*bool*) – whether or not the subtitle is hearing impaired.
- **page_link** (*str*) – URL of the web page from which the subtitle can be downloaded.
- **encoding** (*str*) – Text encoding of the subtitle.

provider_name = ''
Name of the provider that returns that class of subtitle

language = None
Language of the subtitle

hearing_impaired = None
Whether or not the subtitle is hearing impaired

page_link = None
URL of the web page from which the subtitle can be downloaded

content = None
Content as bytes

encoding = None
Encoding to decode with when accessing *text*

id
Unique identifier of the subtitle

text
Content as string

If *encoding* is None, the encoding is guessed with *guess_encoding()*

is_valid()
Check if a *text* is a valid SubRip format.

Returns whether or not the subtitle is valid.

Return type bool

guess_encoding()
Guess encoding using the language, falling back on chardet.

Returns the guessed encoding.

Return type str

get_matches(*video*)
Get the matches against the *video*.

Parameters **video** (*Video*) – the video to get the matches with.

Returns matches of the subtitle.

Return type set

subliminal.subtitle.**get_subtitle_path**(*video_path*, *language=None*, *extension='.srt'*)
Get the subtitle path using the *video_path* and *language*.

Parameters

- **video_path** (*str*) – path to the video.
- **language** (Language) – language of the subtitle to put in the path.
- **extension** (*str*) – extension of the subtitle.

Returns path of the subtitle.

Return type str

subliminal.subtitle.**guess_matches**(*video*, *guess*, *partial=False*)
Get matches between a *video* and a *guess*.

If a guess is *partial*, the absence information won't be counted as a match.

Parameters

- **video** (*Video*) – the video.
- **guess** (*dict*) – the guess.
- **partial** (*bool*) – whether or not the guess is partial.

Returns matches between the *video* and the *guess*.

Return type set

subliminal.subtitle.**fix_line_ending**(*content*)
Fix line ending of *content* by changing it to .

param bytes content content of the subtitle.

return the content with fixed line endings.

rtype bytes

2.4 Providers

class subliminal.providers.**TimeoutSafeTransport**(*timeout*, **args*, ***kwargs*)
Timeout support for xmlrpc.client.SafeTransport.

class subliminal.providers.**ParserBeautifulSoup**(*markup*, *parsers*, ***kwargs*)
A `bs4.BeautifulSoup` that picks the first parser available in *parsers*.

Parameters

- **markup** – markup for the `bs4.BeautifulSoup`.
- **parsers** (*list*) – parser names, in order of preference.

class subliminal.providers.**Provider**
Base class for providers.

If any configuration is possible for the provider, like credentials, it must take place during instantiation.

Raise *ConfigurationError* if there is a configuration error

languages = set([])
Supported set of `Language`

video_types = (<class 'subliminal.video.Episode'>, <class 'subliminal.video.Movie'>)
Supported video types

required_hash = None
Required hash, if any

subtitle_class = None
Subtitle class to use

initialize()
Initialize the provider.

Must be called when starting to work with the provider. This is the place for network initialization or login operations.

Note: This is called automatically when entering the *with* statement

---

terminate()
Terminate the provider.

Must be called when done with the provider. This is the place for network shutdown or logout operations.

Note: This is called automatically when exiting the *with* statement

---

classmethod check(*video*)
Check if the *video* can be processed.

The *video* is considered invalid if not an instance of *video_types* or if the *required_hash* is not present in *hashes* attribute of the *video*.

Parameters **video** (*Video*) – the video to check.

Returns *True* if the *video* is valid, *False* otherwise.

Return type bool

query(**args*, ***kwargs*)
Query the provider for subtitles.

Arguments should match as much as possible the actual parameters for querying the provider

Returns found subtitles.

Return type list of *Subtitle*

Raise *ProviderError*

list_subtitles(*video*, *languages*)
List subtitles for the *video* with the given *languages*.

This will call the *query()* method internally. The parameters passed to the *query()* method may vary depending on the amount of information available in the *video*.

Parameters

- **video** (*Video*) – video to list subtitles for.
- **languages** (set of Language) – languages to search for.

Returns found subtitles.

Return type list of *Subtitle*

Raise *ProviderError*

download_subtitle(*subtitle*)
Download *subtitle*'s *content*.

Parameters **subtitle** (*Subtitle*) – subtitle to download.

Raise *ProviderError*

2.4.1 Addic7ed

subliminal.providers.addic7ed.**series_year_re = <_sre.SRE_Pattern object>**
Series header parsing regex

class subliminal.providers.addic7ed.**Addic7edSubtitle**(*language*, *hearing_impaired*, *page_link*, *series*, *season*, *episode*, *title*, *year*, *version*, *download_link*)
Addic7ed Subtitle.

get_matches(*video*)
Get the matches against the *video*.

Parameters **video** (*Video*) – the video to get the matches with.

Returns matches of the subtitle.

Return type set

class subliminal.providers.addic7ed.**Addic7edProvider**(*username=None*, *password=None*)
Addic7ed Provider.

subtitle_class
alias of *Addic7edSubtitle*

initialize()
Initialize the provider.

Must be called when starting to work with the provider. This is the place for network initialization or login operations.

Note: This is called automatically when entering the *with* statement

terminate()
Terminate the provider.

Must be called when done with the provider. This is the place for network shutdown or logout operations.

Note: This is called automatically when exiting the *with* statement

_get_show_ids(**kw)
Get the `dict` of show ids per series by querying the *shows.php* page.

Returns show id per series, lower case and without quotes.

Return type dict

_search_show_id(**kw)
Search the show id from the *series* and *year*.

Parameters

- **series** (*str*) – series of the episode.
- **year** (*int*) – year of the series, if any.

Returns the show id, if found.

Return type int

get_show_id(*series*, *year=None*, *country_code=None*)
Get the best matching show id for *series*, *year* and *country_code*.

First search in the result of *_get_show_ids()* and fallback on a search with *_search_show_id()*.

Parameters

- **series** (*str*) – series of the episode.
- **year** (*int*) – year of the series, if any.
- **country_code** (*str*) – country code of the series, if any.

Returns the show id, if found.

Return type int

query(*show_id*, *series*, *season*, *year=None*, *country=None*)
Query the provider for subtitles.

Arguments should match as much as possible the actual parameters for querying the provider

Returns found subtitles.

Return type list of *Subtitle*

Raise *ProviderError*

list_subtitles(*video*, *languages*)
List subtitles for the *video* with the given *languages*.

This will call the *query()* method internally. The parameters passed to the *query()* method may vary depending on the amount of information available in the *video*.

Parameters

- **video** (*Video*) – video to list subtitles for.
- **languages** (set of Language) – languages to search for.

Returns found subtitles.

Return type list of *Subtitle*

Raise *ProviderError*

download_subtitle(*subtitle*)
Download *subtitle*'s *content*.

Parameters **subtitle** (*Subtitle*) – subtitle to download.

Raise *ProviderError*

2.4.2 LegendasTv

subliminal.providers.legendastv.**type_map = {'C': 'episode', 'M': 'movie', 'S': 'episode'}**
Conversion map for types

subliminal.providers.legendastv.**season_re = <_sre.SRE_Pattern object at 0x264ea20>**
BR title season parsing regex

subliminal.providers.legendastv.**downloads_re = <_sre.SRE_Pattern object>**
Downloads parsing regex

subliminal.providers.legendastv.**rating_re = <_sre.SRE_Pattern object>**
Rating parsing regex

subliminal.providers.legendastv.**timestamp_re = <_sre.SRE_Pattern object>**
Timestamp parsing regex

subliminal.providers.legendastv.**title_re = <_sre.SRE_Pattern object>**
Title with year/country regex

subliminal.providers.legendastv.**releases_key = 'subliminal.providers.legendastv:releases|{a**
Cache key for releases

class subliminal.providers.legendastv.**LegendasTVArchive**(*id*, *name*, *pack*, *featured*, *link*, *downloads=0*, *rating=0*, *timestamp=None*)
LegendasTV Archive.

Parameters

- **id** (*str*) – identifier.
- **name** (*str*) – name.
- **pack** (*bool*) – contains subtitles for multiple episodes.
- **pack** – featured.
- **link** (*str*) – link.
- **downloads** (*int*) – download count.
- **rating** (*int*) – rating (0-10).
- **timestamp** (*datetime.datetime*) – timestamp.

id = None
Identifier

name = None
Name

pack = None
Pack

featured = None
Featured

link = None
Link

downloads = None
Download count

rating = None
Rating (0-10)

timestamp = None
Timestamp

content = None
Compressed content as `rarfile.RarFile` or `zipfile.ZipFile`

class subliminal.providers.legendastv.**LegendasTVSubtitle**(*language*, *type*, *title*, *year*, *imdb_id*, *season*, *archive*, *name*)

LegendasTV Subtitle.

get_matches(*video*, *hearing_impaired=False*)
Get the matches against the *video*.

Parameters **video** (*Video*) – the video to get the matches with.

Returns matches of the subtitle.

Return type set

class subliminal.providers.legendastv.**LegendasTVProvider**(*username=None*, *password=None*)

LegendasTV Provider.

Parameters

- **username** (*str*) – username.
- **password** (*str*) – password.

subtitle_class
alias of *LegendasTVSubtitle*

initialize()
Initialize the provider.

Must be called when starting to work with the provider. This is the place for network initialization or login operations.

Note: This is called automatically when entering the *with* statement

terminate()
Terminate the provider.

Must be called when done with the provider. This is the place for network shutdown or logout operations.

Note: This is called automatically when exiting the *with* statement

static is_valid_title(*title*, *title_id*, *sanitized_title*, *season*, *year*)
Check if is a valid title.

search_titles(***kw*)
Search for titles matching the *title*.

For episodes, each season has it own title :param str title: the title to search for. :param int season: season of the title :param int title_year: year of the title :return: found titles. :rtype: dict

get_archives(***kw*)
Get the archive list from a given *title_id*, *language_code*, *title_type*, *season* and *episode*.

Parameters

- **title_id** (*int*) – title id.
- **language_code** (*int*) – language code.
- **title_type** (*str*) – episode or movie
- **season** (*int*) – season
- **episode** (*int*) – episode

Returns the archives.

Return type list of *LegendasTVArchive*

download_archive(*archive*)
Download an archive's *content*.

Parameters **archive** (*LegendasTVArchive*) – the archive to download *content* of.

query(*language*, *title*, *season=None*, *episode=None*, *year=None*)
Query the provider for subtitles.

Arguments should match as much as possible the actual parameters for querying the provider

Returns found subtitles.

Return type list of *Subtitle*

Raise *ProviderError*

list_subtitles(*video*, *languages*)
List subtitles for the *video* with the given *languages*.

This will call the *query()* method internally. The parameters passed to the *query()* method may vary depending on the amount of information available in the *video*.

Parameters

- **video** (*Video*) – video to list subtitles for.
- **languages** (set of Language) – languages to search for.

Returns found subtitles.

Return type list of *Subtitle*

Raise *ProviderError*

download_subtitle(*subtitle*)
Download *subtitle*'s *content*.

Parameters **subtitle** (*Subtitle*) – subtitle to download.

Raise *ProviderError*

2.4.3 NapiProjekt

subliminal.providers.napiprojekt.**get_subhash**(*hash*)
Get a second hash based on napiprojekt's hash.

Parameters **hash** (*str*) – napiprojekt's hash.

Returns the subhash.

Return type str

class subliminal.providers.napiprojekt.**NapiProjektSubtitle**(*language*, *hash*)
NapiProjekt Subtitle.

get_matches(*video*)
Get the matches against the *video*.

Parameters **video** (*Video*) – the video to get the matches with.

Returns matches of the subtitle.

Return type set

class subliminal.providers.napiprojekt.**NapiProjektProvider**
NapiProjekt Provider.

subtitle_class
alias of *NapiProjektSubtitle*

initialize()
Initialize the provider.

Must be called when starting to work with the provider. This is the place for network initialization or login operations.

Note: This is called automatically when entering the *with* statement

---

terminate()
Terminate the provider.

Must be called when done with the provider. This is the place for network shutdown or logout operations.

Note: This is called automatically when exiting the *with* statement

---

query(*language*, *hash*)
Query the provider for subtitles.

Arguments should match as much as possible the actual parameters for querying the provider

Returns found subtitles.

Return type list of *Subtitle*

Raise *ProviderError*

list_subtitles(*video*, *languages*)
List subtitles for the *video* with the given *languages*.

This will call the *query()* method internally. The parameters passed to the *query()* method may vary depending on the amount of information available in the *video*.

Parameters

- **video** (*Video*) – video to list subtitles for.
- **languages** (set of Language) – languages to search for.

Returns found subtitles.

Return type list of *Subtitle*

Raise *ProviderError*

download_subtitle(*subtitle*)
Download *subtitle*'s *content*.

Parameters **subtitle** (*Subtitle*) – subtitle to download.

Raise *ProviderError*

2.4.4 OpenSubtitles

class subliminal.providers.opensubtitles.**OpenSubtitlesSubtitle**(*language*, *hearing_impaired*, *page_link*, *subtitle_id*, *matched_by*, *movie_kind*, *hash*, *movie_name*, *movie_release_name*, *movie_year*, *movie_imdb_id*, *series_season*, *series_episode*, *filename*, *encoding*)
OpenSubtitles Subtitle.

get_matches(*video*)
Get the matches against the *video*.

Parameters **video** (*Video*) – the video to get the matches with.

Returns matches of the subtitle.

Return type set

class subliminal.providers.opensubtitles.**OpenSubtitlesProvider**(*username=None*, *password=None*)
OpenSubtitles Provider.

Parameters

- **username** (*str*) – username.

- **password** (*str*) – password.

subtitle_class
alias of *OpenSubtitlesSubtitle*

initialize()
Initialize the provider.

Must be called when starting to work with the provider. This is the place for network initialization or login operations.

Note: This is called automatically when entering the *with* statement

---

terminate()
Terminate the provider.

Must be called when done with the provider. This is the place for network shutdown or logout operations.

Note: This is called automatically when exiting the *with* statement

---

query(*languages*, *hash=None*, *size=None*, *imdb_id=None*, *query=None*, *season=None*, *episode=None*, *tag=None*)
Query the provider for subtitles.

Arguments should match as much as possible the actual parameters for querying the provider

Returns found subtitles.

Return type list of *Subtitle*

Raise *ProviderError*

list_subtitles(*video*, *languages*)
List subtitles for the *video* with the given *languages*.

This will call the *query()* method internally. The parameters passed to the *query()* method may vary depending on the amount of information available in the *video*.

Parameters

- **video** (*Video*) – video to list subtitles for.
- **languages** (set of Language) – languages to search for.

Returns found subtitles.

Return type list of *Subtitle*

Raise *ProviderError*

download_subtitle(*subtitle*)
Download *subtitle*'s *content*.

Parameters **subtitle** (*Subtitle*) – subtitle to download.

Raise *ProviderError*

exception subliminal.providers.opensubtitles.**OpenSubtitlesError**
Base class for non-generic *OpenSubtitlesProvider* exceptions.

exception subliminal.providers.opensubtitles.**Unauthorized**
Exception raised when status is '401 Unauthorized'.

exception subliminal.providers.opensubtitles.**NoSession**
Exception raised when status is '406 No session'.

exception subliminal.providers.opensubtitles.**DownloadLimitReached**
Exception raised when status is '407 Download limit reached'.

exception subliminal.providers.opensubtitles.**InvalidImdbid**
Exception raised when status is '413 Invalid ImdbID'.

exception subliminal.providers.opensubtitles.**UnknownUserAgent**
Exception raised when status is '414 Unknown User Agent'.

exception subliminal.providers.opensubtitles.**DisabledUserAgent**
Exception raised when status is '415 Disabled user agent'.

subliminal.providers.opensubtitles.**checked**(*response*)
Check a response status before returning it.

Parameters **response** – a response from a XMLRPC call to OpenSubtitles.

Returns the response.

Raise *OpenSubtitlesError*

2.4.5 Podnapisi

class subliminal.providers.podnapisi.**PodnapisiSubtitle**(*language*, *hearing_impaired*, *page_link*, *pid*, *releases*, *title*, *season=None*, *episode=None*, *year=None*)
Podnapisi Subtitle.

get_matches(*video*)
Get the matches against the *video*.

Parameters **video** (*Video*) – the video to get the matches with.

Returns matches of the subtitle.

Return type set

class subliminal.providers.podnapisi.**PodnapisiProvider**
Podnapisi Provider.

subtitle_class
alias of *PodnapisiSubtitle*

initialize()
Initialize the provider.

Must be called when starting to work with the provider. This is the place for network initialization or login operations.

Note: This is called automatically when entering the *with* statement

---

terminate()
Terminate the provider.

Must be called when done with the provider. This is the place for network shutdown or logout operations.

Note: This is called automatically when exiting the *with* statement

query(*language*, *keyword*, *season=None*, *episode=None*, *year=None*)
Query the provider for subtitles.

Arguments should match as much as possible the actual parameters for querying the provider

Returns found subtitles.

Return type list of *Subtitle*

Raise *ProviderError*

list_subtitles(*video*, *languages*)
List subtitles for the *video* with the given *languages*.

This will call the *query()* method internally. The parameters passed to the *query()* method may vary depending on the amount of information available in the *video*.

Parameters

- **video** (*Video*) – video to list subtitles for.
- **languages** (set of Language) – languages to search for.

Returns found subtitles.

Return type list of *Subtitle*

Raise *ProviderError*

download_subtitle(*subtitle*)
Download *subtitle*'s *content*.

Parameters **subtitle** (*Subtitle*) – subtitle to download.

Raise *ProviderError*

2.4.6 Shooter

class subliminal.providers.shooter.**ShooterSubtitle**(*language*, *hash*, *download_link*)
Shooter Subtitle.

get_matches(*video*)
Get the matches against the *video*.

Parameters **video** (*Video*) – the video to get the matches with.

Returns matches of the subtitle.

Return type set

class subliminal.providers.shooter.**ShooterProvider**
Shooter Provider.

subtitle_class
alias of *ShooterSubtitle*

initialize()
Initialize the provider.

Must be called when starting to work with the provider. This is the place for network initialization or login operations.

Note: This is called automatically when entering the *with* statement

terminate()
Terminate the provider.

Must be called when done with the provider. This is the place for network shutdown or logout operations.

Note: This is called automatically when exiting the *with* statement

query(*language*, *filename*, *hash=None*)
Query the provider for subtitles.

Arguments should match as much as possible the actual parameters for querying the provider

Returns found subtitles.

Return type list of *Subtitle*

Raise *ProviderError*

list_subtitles(*video*, *languages*)
List subtitles for the *video* with the given *languages*.

This will call the *query()* method internally. The parameters passed to the *query()* method may vary depending on the amount of information available in the *video*.

Parameters

- **video** (*Video*) – video to list subtitles for.
- **languages** (set of Language) – languages to search for.

Returns found subtitles.

Return type list of *Subtitle*

Raise *ProviderError*

download_subtitle(*subtitle*)
Download *subtitle*'s *content*.

Parameters **subtitle** (*Subtitle*) – subtitle to download.

Raise *ProviderError*

2.4.7 TheSubDB

class subliminal.providers.thesubdb.**TheSubDBSubtitle**(*language*, *hash*)
TheSubDB Subtitle.

get_matches(*video*)
Get the matches against the *video*.

Parameters **video** (*Video*) – the video to get the matches with.

Returns matches of the subtitle.

Return type set

class subliminal.providers.thesubdb.**TheSubDBProvider**
TheSubDB Provider.

subtitle_class
alias of *TheSubDBSubtitle*

initialize()
Initialize the provider.

Must be called when starting to work with the provider. This is the place for network initialization or login operations.

Note: This is called automatically when entering the *with* statement

---

terminate()
Terminate the provider.

Must be called when done with the provider. This is the place for network shutdown or logout operations.

Note: This is called automatically when exiting the *with* statement

---

query(*hash*)
Query the provider for subtitles.

Arguments should match as much as possible the actual parameters for querying the provider

Returns found subtitles.

Return type list of *Subtitle*

Raise *ProviderError*

list_subtitles(*video*, *languages*)
List subtitles for the *video* with the given *languages*.

This will call the *query()* method internally. The parameters passed to the *query()* method may vary depending on the amount of information available in the *video*.

Parameters

- **video** (*Video*) – video to list subtitles for.
- **languages** (set of Language) – languages to search for.

Returns found subtitles.

Return type list of *Subtitle*

Raise *ProviderError*

download_subtitle(*subtitle*)
Download *subtitle*'s *content*.

Parameters **subtitle** (*Subtitle*) – subtitle to download.

Raise *ProviderError*

2.4.8 TVsubtitles

class subliminal.providers.tvsubtitles.**TVsubtitlesSubtitle**(*language*, *page_link*, *subtitle_id*, *series*, *season*, *episode*, *year*, *rip*, *release*)

TVsubtitles Subtitle.

get_matches(*video*)

Get the matches against the *video*.

Parameters **video** (*Video*) – the video to get the matches with.

Returns matches of the subtitle.

Return type set

class subliminal.providers.tvsubtitles.**TVsubtitlesProvider**

TVsubtitles Provider.

subtitle_class

alias of *TVsubtitlesSubtitle*

initialize()

Initialize the provider.

Must be called when starting to work with the provider. This is the place for network initialization or login operations.

Note: This is called automatically when entering the *with* statement

terminate()

Terminate the provider.

Must be called when done with the provider. This is the place for network shutdown or logout operations.

Note: This is called automatically when exiting the *with* statement

search_show_id(***kw*)

Search the show id from the *series* and *year*.

Parameters

- **series** (*str*) – series of the episode.
- **year** (*int*) – year of the series, if any.

Returns the show id, if any.

Return type int

get_episode_ids(***kw*)

Get episode ids from the show id and the season.

Parameters

- **show_id** (*int*) – show id.
- **season** (*int*) – season of the episode.

Returns episode ids per episode number.

Return type dict

query(*show_id*, *series*, *season*, *episode*, *year=None*)
Query the provider for subtitles.

Arguments should match as much as possible the actual parameters for querying the provider

Returns found subtitles.

Return type list of *Subtitle*

Raise *ProviderError*

list_subtitles(*video*, *languages*)
List subtitles for the *video* with the given *languages*.

This will call the *query()* method internally. The parameters passed to the *query()* method may vary depending on the amount of information available in the *video*.

Parameters

- **video** (*Video*) – video to list subtitles for.
- **languages** (set of Language) – languages to search for.

Returns found subtitles.

Return type list of *Subtitle*

Raise *ProviderError*

download_subtitle(*subtitle*)
Download *subtitle*'s *content*.

Parameters **subtitle** (*Subtitle*) – subtitle to download.

Raise *ProviderError*

2.5 Refiners

Refiners enrich a *Video* object by adding information to it.

A refiner is a simple function:

subliminal.refiners.**refine**(*video*, ***kwargs*)

Parameters

- **video** (*Video*) – the video to refine.
- ****kwargs** – additional parameters for refiners.

2.5.1 Metadata

subliminal.refiners.metadata.**refine**(*video*, *embedded_subtitles=True*, ***kwargs*)
Refine a video by searching its metadata.

Several *Video* attributes can be found:

- *resolution*
- *video_codec*
- *audio_codec*

- *subtitle_languages*

Parameters **embedded_subtitles** (*bool*) – search for embedded subtitles.

2.5.2 TVDB

subliminal.refiners.tvdb.**refine**(*video*, ***kwargs*)
Refine a video by searching TheTVDB.

Note: This refiner only work for instances of *Episode*.

Several attributes can be found:

- *series*
- *year*
- *series_imdb_id*
- *series_tvdb_id*
- *title*
- *imdb_id*
- *tvdb_id*

2.5.3 OMDb

subliminal.refiners.omdb.**refine**(*video*, ***kwargs*)
Refine a video by searching OMDb API.

Several *Episode* attributes can be found:

- *series*
- *year*
- *series_imdb_id*

Similarly, for a *Movie*:

- *title*
- *year*
- *imdb_id*

2.6 Extensions

class subliminal.extensions.**RegistrableExtensionManager**(*namespace*, *internal_extensions*, ***kwargs*)
:class:~stevedore.extensions.ExtensionManager‘ with support for registration.

It allows loading of internal extensions without setup and registering/unregistering additional extensions.

Loading is done in this order:

- Entry point extensions

- Internal extensions
- Registered extensions

Parameters

- **namespace** (*str*) – namespace argument for :class:~stevedore.extensions.ExtensionManager‘.
- **internal_extensions** (*list*) – internal extensions to use with entry point syntax.
- ****kwargs** – additional parameters for the :class:~stevedore.extensions.ExtensionManager‘ constructor.

registered_extensions = None
Registered extensions with entry point syntax

internal_extensions = None
Internal extensions with entry point syntax

list_entry_points()
Return the list of entry points for this namespace.

The entry points are not actually loaded, their list is just read and returned.

register(*entry_point*)
Register an extension

Parameters **entry_point** (*str*) – extension to register (entry point syntax).

Raise ValueError if already registered.

unregister(*entry_point*)
Unregister a provider

Parameters **entry_point** (*str*) – provider to unregister (entry point syntax).

subliminal.extensions.**provider_manager = <subliminal.extensions.RegistrableExtensionManager**
Provider manager

subliminal.extensions.**refiner_manager = <subliminal.extensions.RegistrableExtensionManager**
Refiner manager

2.7 Score

This module provides the default implementation of the *compute_score* parameter in *download_best_subtitles()* and *download_best_subtitles()*.

Note: To avoid unnecessary dependency on sympy and boost subliminal's import time, the resulting scores are hardcoded here and manually updated when the set of equations change.

Available matches:

- hash
- title
- year
- series
- season

- episode
- release_group
- source
- audio_codec
- resolution
- hearing_impaired
- video_codec
- series_imdb_id
- imdb_id
- tvdb_id

subliminal.score.**episode_scores = {'audio_codec': 3, 'episode': 30, 'hash': 359, 'heari**
Scores for episodes

subliminal.score.**movie_scores = {'audio_codec': 3, 'hash': 119, 'hearing_impaired': 1,**
Scores for movies

subliminal.score.**equivalent_release_groups = (set(['DIMENSION', 'LOL']), set(['FLEET', 'IM**
Equivalent release groups

subliminal.score.**get_equivalent_release_groups**(*release_group*)
Get all the equivalents of the given release group.

Parameters **release_group** (*str*) – the release group to get the equivalents of.

Returns the equivalent release groups.

Return type set

subliminal.score.**get_scores**(*video*)
Get the scores dict for the given *video*.

This will return either *episode_scores* or *movie_scores* based on the type of the *video*.

Parameters **video** (*Video*) – the video to compute the score against.

Returns the scores dict.

Return type dict

subliminal.score.**compute_score**(*subtitle*, *video*, *hearing_impaired=None*)
Compute the score of the *subtitle* against the *video* with *hearing_impaired* preference.

compute_score() uses the *Subtitle.get_matches* method and applies the scores (either from *episode_scores* or *movie_scores*) after some processing.

Parameters

- **subtitle** (*Subtitle*) – the subtitle to compute the score of.
- **video** (*Video*) – the video to compute the score against.
- **hearing_impaired** (*bool*) – hearing impaired preference.

Returns score of the subtitle.

Return type int

2.8 Utils

subliminal.utils.**hash_opensubtitles**(*video_path*)
Compute a hash using OpenSubtitles' algorithm.

Parameters **video_path** (*str*) – path of the video.

Returns the hash.

Return type str

subliminal.utils.**hash_thesubdb**(*video_path*)
Compute a hash using TheSubDB's algorithm.

Parameters **video_path** (*str*) – path of the video.

Returns the hash.

Return type str

subliminal.utils.**hash_napiprojekt**(*video_path*)
Compute a hash using NapiProjekt's algorithm.

Parameters **video_path** (*str*) – path of the video.

Returns the hash.

Return type str

subliminal.utils.**hash_shooter**(*video_path*)
Compute a hash using Shooter's algorithm

Parameters **video_path** (*string*) – path of the video

Returns the hash

Return type string

subliminal.utils.**sanitize**(*string*, *ignore_characters=None*)
Sanitize a string to strip special characters.

Parameters

- **string** (*str*) – the string to sanitize.
- **ignore_characters** (*set*) – characters to ignore.

Returns the sanitized string.

Return type str

subliminal.utils.**sanitize_release_group**(*string*)
Sanitize a *release_group* string to remove content in square brackets.

Parameters **string** (*str*) – the release group to sanitize.

Returns the sanitized release group.

Return type str

subliminal.utils.**timestamp**(*date*)
Get the timestamp of the *date*, python2/3 compatible

Parameters **date** (*datetime.datetime*) – the utc date.

Returns the timestamp of the date.

Return type float

2.9 Cache

subliminal.cache.**SHOW_EXPIRATION_TIME**
Expiration time for show caching

subliminal.cache.**EPISODE_EXPIRATION_TIME**
Expiration time for episode caching

subliminal.cache.**REFINER_EXPIRATION_TIME**
Expiration time for scraper searches

subliminal.cache.**region**
The CacheRegion

Refer to dogpile.cache's region configuration documentation to see how to configure the region

2.10 CLI

Subliminal uses click to provide a powerful CLI.

class subliminal.cli.**MutexLock**(*filename*)
MutexLock is a thread-based rw lock based on dogpile.core.ReadWriteMutex.

acquire_read_lock(*wait*)
Acquire a 'reader' lock.

Raises NotImplementedError by default, must be implemented by subclasses.

acquire_write_lock(*wait*)
Acquire a 'write' lock.

Raises NotImplementedError by default, must be implemented by subclasses.

release_read_lock()
Release a 'reader' lock.

Raises NotImplementedError by default, must be implemented by subclasses.

release_write_lock()
Release a 'writer' lock.

Raises NotImplementedError by default, must be implemented by subclasses.

class subliminal.cli.**Config**(*path*)
A ConfigParser wrapper to store configuration.

Interaction with the configuration is done with the properties.

Parameters **path** (*str*) – path to the configuration file.

path = None
Path to the configuration file

config = None
The underlying configuration object

read()
Read the configuration from *path*

write()
Write the configuration to *path*

class subliminal.cli.**LanguageParamType**
ParamType for languages that returns a Language

convert(*value*, *param*, *ctx*)
Converts the value. This is not invoked for values that are *None* (the missing value).

class subliminal.cli.**AgeParamType**
ParamType for age strings that returns a timedelta

An age string is in the form *number* + *identifier* with possible identifiers:

- w for weeks
- d for days
- h for hours

The form can be specified multiple times but only with that idenfier ordering. For example:

- 1w2d4h for 1 week, 2 days and 4 hours
- 2w for 2 weeks
- 3w6h for 3 weeks and 6 hours

convert(*value*, *param*, *ctx*)
Converts the value. This is not invoked for values that are *None* (the missing value).

2.11 Exceptions

exception subliminal.exceptions.**Error**
Base class for exceptions in subliminal.

exception subliminal.exceptions.**ProviderError**
Exception raised by providers.

exception subliminal.exceptions.**ConfigurationError**
Exception raised by providers when badly configured.

exception subliminal.exceptions.**AuthenticationError**
Exception raised by providers when authentication failed.

exception subliminal.exceptions.**ServiceUnavailable**
Exception raised when status is '503 Service Unavailable'.

exception subliminal.exceptions.**DownloadLimitExceeded**
Exception raised by providers when download limit is exceeded.

CHAPTER 3

License

MIT

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

Index

Symbols

A

C

D

L

M

N

O

P

Q

R

S

T

U

V

W

Y