subliminal Documentation

Release 1.0.1

Antoine Bertin

July 22, 2015

Contents

Subliminal is a python 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 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:

```
>>> from __future__ import unicode_literals
>>> 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 informations were guessed based on the name of the video, you can access some video attributes:

```
>>> video.video_codec
'h264'
>>> 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 persistant 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', 'format', 'hearing_impaired', 'release_group', 'season', 'series', 'video_codec', 'year']
['episode', 'format', 'hearing_impaired', 'season', 'series', 'year']
```

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

```
>>> for s in subtitles[video]:
...     {s: compute_score(s.get_matches(video), video)}
{<PodnapisiSubtitle 'ZtAW' [hu]>: 132}
{<PodnapisiSubtitle 'ONAW' [hu]>: 117}
```

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
- OpenSubtitles
- Podnapisi
- 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 make working with multiple providers seamlessly, the *ProviderPool* exposes the same API but distributes it to its providers.

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 informations 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

```
$ subliminal --help
Usage: subliminal [OPTIONS] COMMAND [ARGS]...

  Subtitles, faster than your thoughts.

Options:
```

```
  --addic7ed USERNAME PASSWORD  Addic7ed configuration.
  --cache-dir DIRECTORY         Path to the cache directory.  [default:
                                ~/.config/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

```
$ subliminal download --help
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|opensubtitles|podnapisi|thesubdb|tvsubtitles]
                                  Provider 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.
  -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).
  -v, --verbose                   Increase verbosity.
  --help                          Show this message and exit.
```

1.3.3 subliminal cache

```
$ subliminal cache --help
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. If guessit is used to extract data from the *Subtitle* subclass, you can use *guess_matches()* as a helper to compute matches between the *Video* and the `Guess`.

Refer to the *scores* attribute of *Episode* and *Movie* for a list of possible matches.

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 API

subliminal.api.**provider_manager**
ExtensionManager based on the entry point subliminal.providers

class subliminal.api.**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

manager = None
Dedicated *provider_manager* as EnabledExtensionManager

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*, *scores=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.
- **scores** (*dict*) – scores to use, if *None*, the *scores* from the video are used.

Returns downloaded subtitles.

Return type list of *Subtitle*

terminate()
Terminate all the *initialized_providers*.

subliminal.api.**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.api.**list_subtitles**(*videos*, *languages*, ***kwargs*)
List subtitles.

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

All other parameters are passed onwards to the `ProviderPool` constructor.

Parameters

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

Returns found subtitles per video.

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

subliminal.api.**download_subtitles**(*subtitles*, ***kwargs*)
Download `content` of *subtitles*.

All other parameters are passed onwards to the `ProviderPool` constructor.

Parameters **subtitles** (list of `Subtitle`) – subtitles to download.

subliminal.api.**download_best_subtitles**(*videos*, *languages*, *min_score=0*, *hearing_impaired=False*, *only_one=False*, *scores=None*, ***kwargs*)
List and download the best matching subtitles.

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

All other parameters are passed onwards to the `ProviderPool` constructor.

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.
- **scores** (*dict*) – scores to use, if *None*, the `scores` from the video are used.

Returns downloaded subtitles per video.

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

subliminal.api.**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

subliminal.video.**SUBTITLE_EXTENSIONS**
Subtitle extensions

class subliminal.video.**Video**(*name*, *format=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. Attributes have an associated score based on equations defined in `score`.

Parameters

- **name** (*str*) – name or path of the video.
- **format** (*str*) – format of the video (HDTV, WEB-DL, BluRay, ...).
- **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** (*int*) – 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

scores = {}
Score by match property

name = None
Name or path of the video

format = None
Format of the video (HDTV, WEB-DL, BluRay, ...)

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, like a Guess instance.

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*, *format=None*, *release_group=None*, *resolution=None*, *video_codec=None*, *audio_codec=None*, *imdb_id=None*, *hashes=None*, *size=None*, *subtitle_languages=None*, *title=None*, *year=None*, *tvdb_id=None*)
Episode *Video*.

Scores are defined by a set of equations, see solve_episode_equations()

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 series.
- **tvdb_id** (*int*) – TVDB id of the episode

scores = {u'hash': 137, u'format': 6, u'series': 44, u'tvdb_id': 88, u'imdb_id': 110, u'audio_codec': 2, u'year': 44, u'he
Score by match property

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

tvdb_id = None
TVDB id of the episode

class subliminal.video.**Movie**(*name*, *title*, *format=None*, *release_group=None*, *resolution=None*, *video_codec=None*, *audio_codec=None*, *imdb_id=None*, *hashes=None*, *size=None*, *subtitle_languages=None*, *year=None*)
Movie *Video*.

Scores are defined by a set of equations, see solve_movie_equations()

Parameters

- **title** (*str*) – title of the movie.
- **year** (*int*) – year of the movie

scores = {u'hash': 62, u'hearing_impaired': 1, u'audio_codec': 2, u'title': 23, u'video_codec': 4, u'format': 6, u'year':
Score by match property

title = None
Title of the movie

year = None
Year of the movie

subliminal.video.**search_external_subtitles**(*path*)
Search for external subtitles from a video *path* and their associated language.

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

Returns found subtitles with their languages.

Return type dict

subliminal.video.**scan_video**(*path*, *subtitles=True*, *embedded_subtitles=True*)
Scan a video and its subtitle languages from a video *path*.

Parameters

- **path** (*str*) – existing path to the video.
- **subtitles** (*bool*) – scan for subtitles with the same name.
- **embedded_subtitles** (*bool*) – scan for embedded subtitles.

Returns the scanned video.

Return type *Video*

subliminal.video.**scan_videos**(*path*, *subtitles=True*, *embedded_subtitles=True*)
Scan *path* for videos and their subtitles.

Parameters

- **path** (*str*) – existing directory path to scan.
- **subtitles** (*bool*) – scan for subtitles with the same name.
- **embedded_subtitles** (*bool*) – scan for embedded subtitles.

Returns the scanned videos.

Return type list of *Video*

subliminal.video.**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.video.**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

2.3 Subtitle

class subliminal.subtitle.**Subtitle**(*language*, *hearing_impaired=False*, *page_link=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.

provider_name = u'‘
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*, *hearing_impaired=False*)
Get the matches against the *video*.

Parameters

- **video** (*Video*) – the video to get the matches with.
- **hearing_impaired** (*bool*) – hearing impaired preference.

Returns matches of the subtitle.

Return type set

subliminal.subtitle.**compute_score**(*matches*, *video*, *scores=None*)
Compute the score of the *matches* against the *video*.

Some matches count as much as a combination of others in order to level the final score:

•*hash* removes everything else

•For *Episode*

–*imdb_id* removes *series*, *tvdb_id*, *season*, *episode*, *title* and *year*

–*tvdb_id* removes *series* and *year*

–*title* removes *season* and *episode*

Parameters

- **video** (*Video*) – the video to get the score with.
- **scores** (*dict*) – scores to use, if *None*, the *scores* from the video are used.

Returns score of the subtitle.

Return type int

subliminal.subtitle.**get_subtitle_path**(*video_path*, *language=None*, *extension=u'.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.**guess_properties**(*string*)
Extract properties from *string* using guessit's *guess_properties* transformer.

Parameters **string** (*str*) – the string potentially containing properties.

Returns the guessed properties.

Return type dict

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

subliminal.providers.**get_version**(*version*)
Put the *version* in the major.minor form.

Parameters **version** (*str*) – the full version.

Returns the major.minor form of the *version*.

Return type str

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

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*

2.5 Cache

subliminal.cache.**CACHE_VERSION**
Subliminal's cache version

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

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

subliminal.cache.**region**
The CacheRegion

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

2.6 Score

2.7 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.

class subliminal.cli.**StringPath**(*exists=False*, *file_okay=True*, *dir_okay=True*, *writable=False*, *readable=True*, *resolve_path=False*)
A Path as str.

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

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

2.8 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.**DownloadLimitExceeded**
Exception raised by providers when download limit is exceeded.

CHAPTER 3

License

MIT

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

S

N

P

Q

R

S

T

V

Y