PySubler Documentation

Release 0.4.2

Jon Nappi

July 20, 2015

Contents

Release v0.4.2. (*Installation*)

With the use of this module, it's now even easier to script the writing of iTunes style metadata to your media files using the SublerCLI.

CHAPTER 1

Atoms

To construct metadata you simply create a collection of Metadata Atoms like so,

```
>>> artist = Atom('Artist', 'Linkin Park')
>>> album = Atom('Album', 'Hybrid Theory')
>>> metadata = [artist, album]
```

CHAPTER 2

Tagging

Then, you simply pass that through to a Subler instance and use the Subler tag method, like so,

```
>>> subler = Subler(path_to_source_file, dest=path_to_dest_file,
                    metadata=metadata)
>>> subler.tag()
```

Contents:

2.1 Installation

This part of the documentation covers the installation of the subler module. The first step to using any software package is getting it properly installed.

2.1.1 Distribute & Pip

Installing PySubler is simple via pip.

$ pip install subler

2.1.2 Get the Code

subler is actively developed on GitHub, the code is always available.

You can either clone the public repository:

```
git clone https://github.com/moogar0880/PySubler.git
```

Download the tarball:

```
$ curl -OL https://github.com/moogar0880/PySubler/tarball/master
```

Or, download the zipball:

```
$ curl -OL https://github.com/moogar0880/PySubler/zipball/master
```

Once you have a copy of the source, you can embed it in your Python package, or install it into your site-packages by running:

```
$ python setup.py install
```

2.2 subler module

This module provides an easily scriptable interface to tagging x264 video format metadata via the SublerCLI.

2.2.1 subler

This module provides an easily scriptable interface to tagging an assortment of media types including x264 video, AAC audio, and many others with iTunes formatted metadata via the SublerCLI. By simply creating new metadata *Atom*'s and specifying any additionally desired variables in an instance of *Subler* you can quickly execute the tagging of metadata to a specified file.

class subler.subler.**Atom**
Bases: subler.subler._Atom

A class representing a single metadata atom. It's important to note that if you attempt to use an invalid tag for this Atom it will not be used when actually tagging the media file via a *Subler* instance.

data
The Subler argument formatted version of this *Atom*

is_valid()
Check that the data in this *Atom* is valid

class subler.subler.**Subler**(*source*, *dest=None*, *chapters=None*, *delay=None*, *chapters_preview=False*, *height=None*, *language='English'*, *remove=False*, *optimize=True*, *downmix=False*, *rating=None*, *media_kind='Movie'*, *explicit=None*, *metadata=None*)
Bases: object

A Python interface to the SublerCLI that can be used to easily read from and write to a specified source file.

__init__(*source*, *dest=None*, *chapters=None*, *delay=None*, *chapters_preview=False*, *height=None*, *language='English'*, *remove=False*, *optimize=True*, *downmix=False*, *rating=None*, *media_kind='Movie'*, *explicit=None*, *metadata=None*)
Create a Subler tagging instance

Parameters

- **source** – The source file to feed to Subler
- **dest** – The destination file to save any changes to
- **chapters** – A .txt file with chapter information
- **chapters_preview** – Boolean option to create an additional video track with the preview of the chapters. Used by iTunes and some other media clients.
- **delay** – The delay of the subtitle track in ms
- **height** – The pixel height of the subtitle track
- **language** – The language of the subtitle track (i.e. English)
- **remove** – Boolean flag for remove all existing subtitles tracks
- **optimize** – Boolean flag for optimizing the file by moving the moov atom at the beginning and interleaving the samples
- **downmix** – downmix audio (mono, stereo, dolby, pl2) from the source file
- **rating** – A valid US, UK, or German content rating

- **media_kind** – The type of media represented by the source file. Valid values are Music, Audiobook, Music Video, Movie, TV Show, Booklet, or Ringtone
- **explicit** – The explicit-ness warning of the content in the source file. Valid values are: None, "Clean", and "Explicit"
- **metadata** – A list of *Atom*'s to be applied to the source file as metadata

existing_metadata
Atom representations of the metadata currently contained in the source file

existing_metadata_collection
AtomCollection representation of the metadata currently contained in the source file

existing_metadata_raw
The metadata currently contained in the source file

explicitness
The explicit rating of the content in the source file. Valid explicit ratings are: None, "Clean", and "Explicit"

media_kind
The type of media encapsulated in the source file. Valid values are: Music, Audiobook, Music Video, Movie, TV Show, Booklet, or Ringtone

rating
The content rating of the source file. Valid US content ratings are: Not Rated, G, PG, PG-13, R, NC-17, TV-Y, TV-Y7, TV-G, TV-PG, TV-14, TV-MA, and Unrated. Valid UK content ratings are: Not Rated, U, Uc, PG, 12, 12A, 15, 18, R18, Exempt, Unrated, and Caution. Valid German content ratings are FSK 0, FSK 6, FSK 12, FSK 16, and FSK 18.

tag()
Apply the specified metadata to the source file and output it to the specified destination file

tracks
A list of tracks found the source file

version
The current version of your systems SublerCLI package

2.2.2 Subler Examples

There are many permutations of ways that a user may arrange their metadata, so all of those variations will not be covered here. However, several smaller examples and examples highlighting the additional functionality of this module will be shown below

Basic Information Gathering

The subler module can be used to access a variety of information about the specified source file:

```
>>> from subler import Subler
>>> sub = Subler('/Users/Jon/Movies/Iron Man.m4v')
>>> sub.tracks
['Track: 1, Video Track, 2:06:23:60, 208 kbit/s, H.264, 1280 x 720', 'Track: 2, Stereo, 2:06:23:60, 1
>>> sub.existing_metadata
[Atom(tag='Rating', value='PG-13'), Atom(tag='Name', value='Iron Man'),...]
>>> sub.version
'version 0.19'
```

Although it's documented above, I'd like to specify that there is a distinct difference between subler.version_info/__version__ and subler.Subler.version.

```
>>> import subler
>>> subler.version_info
(0, 3, 0)
>>> subler.__version__
'0.3.0'
```

As shown above, subler.version_info and __version__ depict the version of this Python package, while the subler instances version attribute (seen in the first example) is actually the version information about your systems SublerCLI executable

Tagging an Audio File

This example will show how to tag an Audio file with a variety of different types of metadata

```
>>> from subler import Atom, Subler
>>> atoms = []
>>> artist = Atom('Artist', 'Linkin Park')
>>> album = Atom('Album', 'Hybrid Theory')
>>> track = Atom('Name', 'Papercut')
>>> artwork = Atom('Artwork', '/path/to/artwork.jpg')
>>> metadata = [artist, album, track, artwork]
>>> tagger = Subler('/path/to/papercut.mp3', media_kind='Music',
                    explicit='Explicit', metadata=metadata)
>>> tagger.tag()
```

Tagging other Media Types

Tagging a Audiobook, Music Video, Movie, TV Show, Booklet, or Ringtone is fundamentally no different than an audio file, you just need to remember to explicity set the media_kind attribute for anything that isn't a Movie.

2.2.3 Valid Tags

Also probably worth noting is the list of valid Tag names as the official documentation appears to be out of date:

- Name
- Artist
- Album Artist
- Album
- Grouping
- Composer
- Comments
- Genre
- Release Date
- Track
- Disk
- Tempo

- TV Show
- TV Episode
- TV Network
- TV Episode ID
- TV Season
- Description
- Long Description
- Series Description
- Rating
- Studio
- Cast
- Director
- Codirector
- Producers
- Screenwriters
- Lyrics
- Copyright
- Encoding Tool
- Encoded By
- Keywords
- Category
- contentID
- artistID
- playlistID
- genreID
- composerID
- XID
- iTunes Account
- iTunes Account Type
- iTunes Country
- Track Sub- Title
- Song Description
- Art Director
- Arranger
- Lyricist
- Acknowledgement

- Conductor
- Linear Notes
- Record Company
- Original Artist
- Phonogram Rights
- Producer
- Performer
- Publisher
- Sound Engineer
- Soloist
- Credits
- Thanks
- Online Extras
- Executive Producer
- Sort Name
- Sort Artist
- Sort Album Artist
- Sort Album
- Sort Composer
- Sort TV Show
- Media Kind
- HD Video
- Gapless
- Artwork

2.3 subler.tools module

A collection of tools to help ease the pain of tagging metadata

class subler.tools.**AtomCollection**
Bases: dict

A dictionary collection of *Atom* instances. When a tag is added, the tag for it is used as the key to the dictionary, the value for which is an *Atom* instance. When you key back on an item already in the dictionary the value of that *Atom* is returned. Thus, if you know a key 'Artist' exists, you can get the value of that tag by doing `my_collection['Artist']`

atoms
The list of *Atom*'s contained in this collection

get(*k*[, *d*]) → D[k] if k in D, else d. d defaults to None.

items() → list of D's (key, value) pairs, as 2-tuples

subler.tools.**tag_dict**(*d*, *source*, ***kwargs*)

Normally, you are only allowed to tag using a list of *Atom* objects. This function will allow you to provide a dict of metadata to be tagged via a *Subler* instance. Note: the same *Atom* limitations of valid tags will still apply

Parameters

- **d** – A dict of Atom data
- **source** – The source file you wish to write the metadata to
- ****kwargs** – Any other keyword args you would like passed to *Subler*. Note: if you provide a 'metadata' field it will be merged with the data stored in *d*

2.3.1 Tools Examples

Below are some examples of how one might make use of the functions provided within the tools module.

AtomCollection Examples

AtomCollection's can be used to store Atom data in a dict representation.

```
>>> from subler import Subler
>>> from subler.tools import AtomCollection
>>> metadata = AtomCollection()
>>> metadata['TV Show'] = 'Firefly'
>>> metadata['TV Episode #'] = 1
>>> metadata['Genre'] = 'Drama'
>>> s = Subler('/Users/Me/Movies/Firefly/S1/S1E1.m4v', media_kind='TV Show',
...            metadata=metadata.atoms)
>>> s.tag()
```

tag_dict Examples

If you'd prefer to not deal with *Atom* instances or an *AtomCollection* for managing your metadata, you can optionally store your data in a stock Python *dict* instance and still easily tag your metadata

```
>>> from subler.tools import tag_dict
>>> metadata = {'TV Show': 'Firefly', 'TV Episode #': 1, 'Genre': 'Drama'}
>>> tag_dict(metadata, '/Users/Me/Movies/Firefly/S1/S1E1.m4v',
...          media_kind='TV Show')
```

2.4 pysubler CLI

As of PySubler 0.4.0, PySubler ships with the *pysubler* commandline utility, which provides an interactive manner through which you can easily edit the metadata found within your media file, using nothing more than your favorite text editor!

From the commandline, simply run:

```
$ pysubler /path/to/your/file.m4v
```

From there you will be provided a template file you can fill with whatever metadata you want written to your media file. This template will also automatically populate with any metadata currently in your media file. The template will look similar to the following

```
[       Artist      ]:
[    Album Artist   ]:
[       Album       ]:
[      Grouping     ]:
[      Composer     ]:
[      Comments     ]:
[       Genre       ]:
[    Release Date   ]:
[      Track #      ]:
[       Disk #      ]:
[       Tempo       ]:
[      TV Show      ]:
[    TV Episode #   ]:
[     TV Network    ]:
[   TV Episode ID   ]:
[     TV Season     ]:
[    Description    ]:
[  Long Description ]:
[ Series Description]:
[      HD Video     ]:
[ Rating Annotation ]:
[       Studio      ]:
[        Cast       ]:
[      Director     ]:
[      Gapless      ]:
[     Codirector    ]:
[     Producers     ]:
[   Screenwriters   ]:
[       Lyrics      ]:
[     Copyright     ]:
[   Encoding Tool   ]: HandBrake 0.9.9 2013051800
...
```

And by simply writing information similar to this

```
[       Artist      ]: Firefly
[    Album Artist   ]: Firefly
[       Album       ]: Firefly, Season 1
...
```

The metadata you provided will be passed through to Subler and written to your media file as soon as you save the file in your text editor.

2.5 subler.utils module

Although the functions contained in this util module will likely be less than helpful to someone trying to use this module, their behavior is documented below Utilities for assisting with the writing/tagging of metadata

`subler.utils.`**`subler_executable`**`()`
Find a localized subler executable and return it's path. If no executable can be found, None is returned

`subler.utils.`**`get_output`**`(`*input_args*`)`
Pass the provided *input_args* `list` to *subprocess.check_output*. The decoded and stipped output is then returned

Parameters **input_args** – Parameters to pass to the **popenargs* argument of the *subprocess.check_output* function call

Returns The decoded and stripped return value of the call to *subprocess.check_output*

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

S

Symbols

A

D

E

G

I

M

R

S

T

V