# Contents

1 Supported Metadata Fields 3

2 Compatibility 5

3 MediaFile Class 7

4 Exceptions 9

5 Internals 11

6 Changelog 13
   6.1 v0.4.0 13
   6.2 v0.3.0 13
   6.3 v0.2.0 13
   6.4 v0.1.0 13

Index 15
MediaFile is a simple interface to the metadata tags for many audio file formats. It wraps Mutagen, a high-quality library for low-level tag manipulation, with a high-level, format-independent interface for a common set of tags.

It currently supports MP3 files (ID3 tags), AAC files (as tagged by iTunes) as well as FLAC, Ogg, Monkey’s Audio, WavPack, and Musepack.

MediaFile attempts to always return a usable value (i.e., it never returns None or throws an exception when a tag is accessed). If a tag is not present, an empty and false value of the appropriate type – such as zero or the empty string – is returned.
CHAPTER 1

Supported Metadata Fields

The metadata schema is generally based on MusicBrainz’ schema with similar naming. MediaFile supports:

- basic fields like `title`, `album`, `artist` and `albumartist`,
- sorting variants like `albumartist_sort` and `composer_sort`,
- identifiers like `asin` or `mb_releasegroupid`,
- dates like the release `year`, `month` and `day` with convenience wrapper `date`,
- detailed metadata like `language` or `media`,
- `lyrics`,
- calculated metadata like `bpm` (beats per minute) and `r128_track_gain` (ReplayGain),
- embedded images (e.g. album art),
- file metadata like `bitrate` and `length`. 
Compatibility

The ID3 and MPEG-4 test cases were created with iTunes and the FLAC and Ogg test cases were created (mostly) with MediaRage. The Monkey’s Audio tags were mainly fabricated using the open-source Tag. Thus, MediaFile’s tag support most closely aligns with those three applications. Some open questions remain about how to most compatibly tag files. In particular, some fields MediaFile supports don’t seem standardized among FLAC/Ogg taggers:

- **grouping and lyrics**: couldn’t find anyone who supports these in a cursory search; MediaFile uses the keys grouping and lyrics

- **tracktotal and disctotal**: we use the keys tracktotal, totaltracks, and trackc all to mean the same thing

- **year**: this field appears both as a part of the date field and on its own using some taggers; both are supported

For fields that have multiple possible storage keys, MediaFile optimizes for interoperability: it accepts _any_ of the possible storage keys and writes _all_ of them. This may result in duplicated information in the tags, but it ensures that other players with slightly divergent opinions on tag names will all be able to interact with beets.

Images (album art) are stored in the standard ways for ID3 and MPEG-4. For all other formats, images are stored with the METADATA_BLOCK_PICTURE standard from Vorbis Comments. The older COVERART unofficial format is also read but is not written.
CHAPTER 3

MediaFile Class

class mediafile.MediaFile (path, id3v23=False)

Represents a multimedia file on disk and provides access to its metadata.

__init__ (path, id3v23=False)

Constructs a new MediaFile reflecting the file at path. May throw UnreadableFileError.

By default, MP3 files are saved with ID3v2.4 tags. You can use the older ID3v2.3 standard by specifying the id3v23 option.

classmethod fields()

Get the names of all writable properties that reflect metadata tags (i.e., those that are instances of MediaField).

classmethod readable_fields()

Get all metadata fields: the writable ones from fields() and also other audio properties.

save()

Write the object’s tags back to the file. May throw UnreadableFileError.

update (dict)

Set all field values from a dictionary.

For any key in dict that is also a field to store tags the method retrieves the corresponding value from dict and updates the MediaFile. If a key has the value None, the corresponding property is deleted from the MediaFile.
CHAPTER 4

Exceptions

class mediafile.UnreadableFileError (path, msg)
   Mutagen is not able to extract information from the file.

class mediafile.FileTypeError (path, mutagen_type=None)
   Reading this type of file is not supported.
   
   If passed the mutagen_type argument this indicates that the mutagen type is not supported by Mediafile.

class mediafile.MutagenError (path, mutagen_exc)
   Raised when Mutagen fails unexpectedly—probably due to a bug.
**MediaField**

A descriptor providing access to a particular (abstract) metadata field.

```python
__init__(*styles, **kwargs)
```

Creates a new MediaField.

**Parameters**

- `styles` – *StorageStyle* instances that describe the strategy for reading and writing the field in particular formats. There must be at least one style for each possible file format.
- `out_type` – the type of the value that should be returned when getting this property.

**StorageStyle**

A strategy for storing a value for a certain tag format (or set of tag formats). This basic StorageStyle describes simple 1:1 mapping from raw values to keys in a Mutagen file object; subclasses describe more sophisticated translations or format-specific access strategies.

MediaFile uses a StorageStyle via three methods: `get()`, `set()`, and `delete()`. It passes a Mutagen file object to each.

Internally, the StorageStyle implements `get()` and `set()` using two steps that may be overridden by subtypes. To get a value, the StorageStyle first calls `fetch()` to retrieve the value corresponding to a key and then `deserialize()` to convert the raw Mutagen value to a consumable Python value. Similarly, to set a field, we call `serialize()` to encode the value and then `store()` to assign the result into the Mutagen object.

Each StorageStyle type has a class-level `formats` attribute that is a list of strings indicating the formats that the style applies to. MediaFile only uses StorageStyles that apply to the correct type for a given audio file.

```python
def delete(mutagen_file)
    Remove the tag from the file.

def deserialize(mutagen_value)
    Given a raw value stored on a Mutagen object, decode and return the represented value.

def fetch(mutagen_file)
    Retrieve the raw value of for this tag from the Mutagen file object.
```
formats = ['FLAC', 'OggOpus', 'OggTheora', 'OggSpeex', 'OggVorbis', 'OggFlac', 'APEv2File']
List of mutagen classes the StorageStyle can handle.

get (mutagen_file)
Get the value for the field using this style.

serialize (value)
Convert the external Python value to a type that is suitable for storing in a Mutagen file object.

set (mutagen_file, value)
Assign the value for the field using this style.

store (mutagen_file, value)
Store a serialized value in the Mutagen file object.
6.1 v0.4.0

- Added a barcode field.
- Added new tag mappings for albumtype and albumstatus.

6.2 v0.3.0

- Fixed tests for compatibility with Mutagen 1.43.
- Fix the MPEG-4 tag mapping for the label field to use the right capitalization.

6.3 v0.2.0

- R128 gain tags are now stored in Q7.8 integer format, as per the relevant standard.
- Added an mb_workid field.
- The Python source distribution now includes an __init__.py file that makes it easier to run the tests.

6.4 v0.1.0

This is the first independent release of MediaFile. It is now synchronised with the embedded version released with beets v1.4.8.
Symbols

__init__() (mediafile.MediaField method), 11
__init__() (mediafile.MediaFile method), 7

D
delete() (mediafile.StorageStyle method), 11
deserialize() (mediafile.StorageStyle method), 11

F
fetch() (mediafile.StorageStyle method), 11
fields() (mediafile.MediaFile class method), 7
FileTypeError (class in mediafile), 9
formats (mediafile.StorageStyle attribute), 11

G
get() (mediafile.StorageStyle method), 12

M
MediaField (class in mediafile), 11
MediaFile (class in mediafile), 7
MutagenError (class in mediafile), 9

R
readable_fields() (mediafile.MediaFile class method), 7

S
save() (mediafile.MediaFile method), 7
serialize() (mediafile.StorageStyle method), 12
set() (mediafile.StorageStyle method), 12
StorageStyle (class in mediafile), 11
store() (mediafile.StorageStyle method), 12

U
UnreadableFileError (class in mediafile), 9
update() (mediafile.MediaFile method), 7