

audiotags Manual

Tianyi Shi

2020-10-27

Contents

Preface	5
1 Start Simple	7
2 Conversion	9
3 AnyTag	11
4 Downcast	13

Preface

Thank you for considering **audiotags**!

Examples in this manual

If you want to run the examples in this book:

1. clone the repo and navigate into it
2. create `src/main.rs`
3. all examples, unless otherwise specified, can be copied verbatim from this book to `src/main.rs` and run with `cargo run` (if you're reading it online, the copy button wil show if you hover over a code block)

Chapter 1

Start Simple

The following example shows how you can read an audio file, parse, set, and save its metadata:

```
use audiotags::{MimeType, Picture, Tag, TagType};

const MP3_FILE: &'static str = "assets/a.mp3";

fn main() {
    // using `default()` so that the metadata format is guessed
    // (from the file extension) (in this case, Id3v2 tag is read)
    let mut tag = Tag::default().read_from_path(MP3_FILE).unwrap();
    // You can also specify the metadata format (tag type):
    let _tag = Tag::with_tag_type(TagType::Id3v2)
        .read_from_path(MP3_FILE)
        .expect("Fail to read!");

    tag.set_title("foo title");
    assert_eq!(tag.title(), Some("foo title"));
    tag.remove_title();
    assert!(tag.title().is_none());
    tag.remove_title();
    // trying to remove a field that's already empty won't hurt

    let cover = Picture {
        mime_type: MimeType::Jpeg,
        data: &vec![0u8; 10],
    };

    tag.set_album_cover(cover.clone());
}
```

```
    assert_eq!(tag.album_cover(), Some(cover));
    tag.remove_album_cover();
    assert!(tag.album_cover().is_none());
    tag.remove_album_cover();

    tag.save_to_path(MP3_FILE).expect("Fail to save");
    // TASK: reload the file and prove the data have been saved
}
```


Chapter 2

Conversion

The following example shows how you can read the tag in an mp3 file, convert it into an mp4 tag, and write it to an m4a file.

```
use audiotags::{Config, Tag, TagType};

fn main() {
    // we have an mp3 and an m4a file
    const MP3_FILE: &'static str = "assets/a.mp3";
    const M4A_FILE: &'static str = "assets/a.m4a";
    // read tag from the mp3 file. Using `default()` so that the
    // type of tag is guessed from the file extension
    let mut mp3tag = Tag::default().read_from_path(MP3_FILE).unwrap();
    // set the title
    mp3tag.set_title("title from mp3 file");
    // we can convert it to an mp4 tag and save it to an m4a file.
    let mut mp4tag = mp3tag.into_tag(TagType::Mp4);
    mp4tag.write_to_path(M4A_FILE).unwrap();

    // reload the tag from the m4a file; this time specifying the
    // tag type (you can also use `default()`)
    let mut mp4tag = Tag::with_tag_type(TagType::Mp4)
        .read_from_path(M4A_FILE)
        .unwrap();
    // the tag originated from an mp3 file is successfully written
    // to an m4a file!
    assert_eq!(mp4tag.title(), Some("title from mp3 file"));
    // multiple artists
    mp4tag.add_artist("artist1 of mp4");
    mp4tag.add_artist("artist2 of mp4");
}
```

```
assert_eq!(
    mp4tag.artists(),
    Some(vec!["artist1 of mp4", "artist2 of mp4"])
);
// convert to id3 tag, which does not support multiple artists
mp4tag.set_config(Config::default().sep_artist("/"));
// separator is by default `;` but we can customise it
let mp3tag = mp4tag.into_tag(TagType::Id3v2);
assert_eq!(mp3tag.artist(), Some("artist1 of mp4/artist2 of mp4"));
}
```

Chapter 3

AnyTag

The following example shows how you can create a “generic” `AnyTag` and convert it into a specific tag type.

```
use audiotags::{AnyTag, AudioTagEdit, Id3v2Tag};

fn main() {
    let mut tag = AnyTag::default();
    tag.set_title("foo");
    tag.set_year(2001);
    let tag: Id3v2Tag = tag.into();
    assert_eq!(tag.year(), Some(2001));
    tag.write_to_path("assets/a.mp3").unwrap();
}
```


Chapter 4

Downcast

The following example shows how you can downcast a `Box<dyn AudioTag>` into its “backend” tag type. This allows you to set the uncommon metadata supported by the corresponding backend but not by **audiotags**.

```
use audiotags::*;

fn main() {
    let mut innertag = metaflac::Tag::default();
    innertag
        .vorbis_comments_mut()
        .set_title(vec!["title from metaflac::Tag"]);
    let tag: FlacTag = innertag.into();
    let mut id3tag = tag.into_tag(TagType::Id3v2);
    id3tag
        .write_to_path("assets/a.mp3")
        .expect("Fail to write!");

    let id3tag_reload = Tag::default()
        .read_from_path("assets/a.mp3")
        .expect("Fail to read!");
    assert_eq!(id3tag_reload.title(), Some("title from metaflac::Tag"));

    let mut id3tag_inner: id3::Tag = id3tag_reload.try_into().unwrap();
    // this would fail if `id3tag_reload` isn't really a id3 tag.

    let timestamp = id3::Timestamp {
        year: 2013,
        month: Some(2u8),
        day: Some(5u8),
    }
```

```
        hour: Some(6u8),
        minute: None,
        second: None,
    };
    id3tag_inner.set_date_recorded(timestamp.clone());
    id3tag_inner
        .write_to_path("assets/a.mp3", id3::Version::Id3v24)
        .expect("Fail to write!");

    let id3tag_reload = id3::Tag::read_from_path("assets/a.mp3")
        .expect("Fail to read!");
    assert_eq!(id3tag_reload.date_recorded(), Some(timestamp));
}
```