MetaData

https://www.npmjs.com/package/ffmetadata

node-ffmetadata

Read and write media metadata using ffmpeg's metadata framework.

Usage

```
var ffmetadata = require("ffmetadata");

// Set path to ffmpeg - optional if in $PATH or $FFMPEG_PATH
ffmetadata.setFfmpegPath("/path/to/ffmpeg");

// Read song.mp3 metadata
ffmetadata.read("song.mp3", function(err, data) {
        if (err) console.error("Error reading metadata", err);
        else console.log(data);
});

// Set the artist for song.mp3
var data = {
  artist: "Me",
};
ffmetadata.write("song.mp3", data, function(err) {
        if (err) console.error("Error writing metadata", err);
        else console.log("Data written");
});
```

Artwork

You can optionally include an array of files to be added to the source file. This is a destructive action, it will overwrite any previous streams on the file. For audio data, this is typically just one image. For video, this is where you would write additional audio streams or subtitle tracks.

```
var options = {
  attachments: ["cover.jpg"],
};
```

```
ffmetadata.write("song.mp3", {}, options, function(err) {
        if (err) console.error("Error writing cover art");
        else console.log("Cover art added");
});
```

Metadata

Metadata might include the following fields:

- `"artist"` : artist name
- `"album"` : album name
- `"title"` : song title
- `"track"` : place in the album (e.g. `"5/8"`)
- `"disc"` : for multidisc albums
- `"label"` : record label
- `"date"` : arbitrary, but usually year (e.g. `"2002"`)

See FFmpeg Metadata for details.

API

`ffmetadata.read(file, [options], callback)`

The `callback` function is called (`callback(err, data)`) with an error or an object containing metadata from `file` .

If `options.dryRun` is truthy, the `ffmpeg` process is not actually invoked and instead an array of the arguments that *would* have been used to invoke ffmpeg is returned synchronously. The `callback` argument is not used in this case.

`options.coverPath` : Option to provide a path, where `ffmpeg` will save cover image. Unfortunately, `ffmpeg` will always convert resulted image, based on extension in provided `coverPath` .

`ffmetadata.write(file, data, [options], callback)`

Write metadata to `file` and optionally append additional attachments (e.g., artwork image).

The `data` object should contain metadata fields supported by FFmpeg. See the metadata section above for more information.

The `options` object may be provided with any of the following fields:

- `options.attachments` : An array of files that should be appended as additional streams to the output. This can be used to e.g., attach artwork images; see the artwork section above for more information.
- `options["id3v1"]` : If this property is truthy, id3 v1 will also be included. This is useful if compatibility with *Windows Explorer* is desired.
- `options["id3v2.3"]` : If this property is truthy, id3 v2.3 will be used (instead of the default of v2.4). This is useful if compatibility with *Windows Explorer* is desired (see #8).
- `options.dryRun` : If this property is truthy, the `ffmpeg` process is not actually invoked and instead an array of the arguments that *would* have been used to invoke ffmpeg is returned synchronously. The `callback` argument is not used in this case.

`callback(err)` is called when finished, with an error if one occurred.

Path to FFmpeg

By default, the module looks for the ffmpeg executable either in the `FFMPEG_PATH` and `PATH` environmental variables. You may also set this path in the code by calling the `ffmetadata.setFfmpegPath("path/to/ffmpeg")` function.

Installation

Dependency: FFmpeg or libav must be installed on the system. This module uses the `ffmpeg` command-line tool. Version: 0.10.

```
1 npm install ffmetadata
```