NAME

MP3::Info − Manipulate / fetch info from MP3 audio files

SYNOPSIS

```
#!perl −w
use MP3::Info;
my $file = 'Pearls_Before_Swine.mp3';
set_mp3tag($file, 'Pearls Before Swine', q"77's",
        'Sticks and Stones', '1990',
        q"(c) 1990 77's LTD.", 'rock & roll');

my $tag = get_mp3tag($file) or die "No TAG info";
$tag−>{GENRE} = 'rock';
set_mp3tag($file, $tag);

my $info = get_mp3info($file);
printf "$file length is %d:%d\n", $info−>{MM}, $info−>{SS};

my $mp3 = new MP3::Info $file;
$mp3−>title('Perls Before Swine');
printf "$file length is %s, title is %s\n",
        $mp3−>time, $mp3−>title;
```

DESCRIPTION

`$mp3` = MP3::Info−>new(FILE)

OOP interface to the rest of the module. The same keys available via get_mp3info and get_mp3tag are available via the returned object (using upper case or lower case; but note that all-caps ‘‘VERSION’’ will return the module version, not the MP3 version).

Passing a value to one of the methods will set the value for that tag in the MP3 file, if applicable.

use_mp3_utf8([STATUS])

Tells MP3::Info to (or not) return TAG info in UTF−8. TRUE is 1, FALSE is 0. Default is TRUE, if available.

Will only be able to turn it on if Encode is available. ID3v2 tags will be converted to UTF−8 according to the encoding specified in each tag; ID3v1 tags will be assumed Latin−1 and converted to UTF−8.

Function returns status (TRUE/FALSE). If no argument is supplied, or an unaccepted argument is supplied, function merely returns status.

This function is not exported by default, but may be exported with the `:utf8` or `:all` export tag.

use_winamp_genres()

Puts WinAmp genres into `@mp3_genres` and `%mp3_genres` (adds 68 additional genres to the default list of 80). This is a separate function because these are non-standard genres, but they are included because they are widely used.

You can import the data structures with one of:

```
use MP3::Info qw(:genres);
use MP3::Info qw(:DEFAULT :genres);
use MP3::Info qw(:all);
```

remove_mp3tag (FILE [, VERSION, BUFFER])

Can remove ID3v1 or ID3v2 tags. VERSION should be `1` for ID3v1 (the default), `2` for ID3v2, and `ALL` for both.

For ID3v1, removes last 128 bytes from file if those last 128 bytes begin with the text ’TAG’. File will be 128 bytes shorter.

For ID3v2, removes ID3v2 tag. Because an ID3v2 tag is at the beginning of the file, we rewrite the file after removing the tag data. The buffer for rewriting the file is 4MB. BUFFER (in bytes) ca change the buffer size.

Returns the number of bytes removed, or −1 if no tag removed, or undef if there is an error.

set_mp3tag (FILE, TITLE, ARTIST, ALBUM, YEAR, COMMENT, GENRE [, TRACKNUM])
set_mp3tag (FILE, `$HASHREF`)

Adds/changes tag information in an MP3 audio file. Will clobber any existing information in file.

Fields are TITLE, ARTIST, ALBUM, YEAR, COMMENT, GENRE. All fields have a 30−byte limit, except for YEAR, which has a four-byte limit, and GENRE, which is one byte in the file. The GENRE passed in the function is a case-insensitive text string representing a genre found in `@mp3_genres`.

Will accept either a list of values, or a hashref of the type returned by `get_mp3tag`.

If TRACKNUM is present (for ID3v1.1), then the COMMENT field can only be 28 bytes.

ID3v2 support may come eventually. Note that if you set a tag on a file with ID3v2, the set tag will be for ID3v1[.1] only, and if you call `get_mp3tag` on the file, it will show you the (unchanged) ID3v2 tags, unless you specify ID3v1.

get_mp3tag (FILE [, VERSION, RAW_V2, APE2])

Returns hash reference containing tag information in MP3 file. The keys returned are the same as those supplied for `set_mp3tag`, except in the case of RAW_V2 being set.

If VERSION is `1`, the information is taken from the ID3v1 tag (if present). If VERSION is `2`, the information is taken from the ID3v2 tag (if present). If VERSION is not supplied, or is false, the ID3v1 tag is read if present, and then, if present, the ID3v2 tag information will override any existing ID3v1 tag info.

If RAW_V2 is `1`, the raw ID3v2 tag data is returned, without any manipulation of text encoding. The key name is the same as the frame ID (ID to name mappings are in the global `%v2_tag_names`).

If RAW_V2 is `2`, the ID3v2 tag data is returned, manipulating for Unicode if necessary, etc. It also takes multiple values for a given key (such as comments) and puts them in an arrayref.

If APE is `1`, an APE tag will be located before all other tags.

If the ID3v2 version is older than ID3v2.2.0 or newer than ID3v2.4.0, it will not be read.

Strings returned will be in Latin−1, unless UTF−8 is specified (use_mp3_utf8), (unless RAW_V2 is `1`).

Also returns a TAGVERSION key, containing the ID3 version used for the returned data (if TAGVERSION argument is `0`, may contain two versions).

get_mp3info (FILE)

Returns hash reference containing file information for MP3 file. This data cannot be changed. Returned data:

```
        VERSION         MPEG audio version (1, 2, 2.5)
        LAYER           MPEG layer description (1, 2, 3)
        STEREO          boolean for audio is in stereo

        VBR             boolean for variable bitrate
        BITRATE         bitrate in kbps (average for VBR files)
        FREQUENCY       frequency in kHz
        SIZE            bytes in audio stream
        OFFSET          bytes offset that stream begins

        SECS            total seconds
        MM              minutes
        SS              leftover seconds
        MS              leftover milliseconds
        TIME            time in MM:SS

        COPYRIGHT       boolean for audio is copyrighted
        PADDING         boolean for MP3 frames are padded
        MODE            channel mode (0 = stereo, 1 = joint stereo,
                        2 = dual channel, 3 = single channel)
        FRAMES          approximate number of frames
        FRAME_LENGTH    approximate length of a frame
        VBR_SCALE       VBR scale from VBR header
```

On error, returns nothing and sets `$@`.

TROUBLESHOOTING

If you find a bug, please send me a patch (see the project page in "SEE ALSO"). If you cannot figure out why it does not work for you, please put the MP3 file in a place where I can get it (preferably via FTP, or HTTP, or .Mac iDisk) and send me mail regarding where I can get the file, with a detailed description of the problem.

If I download the file, after debugging the problem I will not keep the MP3 file if it is not legal for me to have it. Just let me know if it is legal for me to keep it or not.

TODO

ID3v2 Support

Still need to do more for reading tags, such as using Compress::Zlib to decompress compressed tags. But until I see this in use more, I won't bother. If something does not work properly with reading, follow the instructions above for troubleshooting.

ID3v2 *writing* is coming soon.

Get data from scalar

Instead of passing a file spec or filehandle, pass the data itself. Would take some work, converting the seeks, etc.

Padding bit ?

Do something with padding bit.

Test suite

Test suite could use a bit of an overhaul and update. Patches very welcome.

- Revamp getset.t. Test all the various get_mp3tag args.
- Test Unicode.
- Test OOP API.
- Test error handling, check more for missing files, bad MP3s, etc.

Other VBR

Right now, only Xing VBR is supported.

THANKS

Edward Allen, Vittorio Bertola, Michael Blakeley, Per Bolmstedt, Tony Bowden, Tom Brown, Sergio Camarena, Chris Dawson, Kevin Deane-Freeman, Anthony DiSante, Luke Drumm, Kyle Farrell, Jeffrey Friedl, brian d foy, Ben Gertzfield, Brian Goodwin, Andy Grundman, Todd Hanneken, Todd Harris, Woodrow Hill, Kee Hinckley, Roman Hodek, Ilya Konstantinov, Peter Kovacs, Johann Lindvall, Alex Marandon, Peter Marschall, michael, Trond Michelsen, Dave O'Neill, Christoph Oberauer, Jake Palmer, Andrew Phillips, David Reuteler, John Ruttenberg, Matthew Sachs, scfc_de, Hermann Schwaerzler, Chris Sidi, Roland Steinbach, Brian S. Stephan, Stuart, Dan Sully, Jeffery Sumler, Predrag Supurovic, Bogdan Surdu, Pierre-Yves Thoulon, tim, Pass F. B. Travis, Tobias Wagener, Ronan Waide, Andy Waite, Ken Williams, Ben Winslow, Meng Weng Wong, Justin Fletcher.

CURRENT AUTHOR

Dan Sully <daniel | at | cpan.org> & Logitech.

AUTHOR EMERITUS

Chris Nandor <pudge AT pobox DOT com>, http://pudge.net/

COPYRIGHT AND LICENSE

Copyright (c) 2006−2008 Dan Sully & Logitech. All rights reserved.

Copyright (c) 1998−2005 Chris Nandor. All rights reserved.

This program is free software; you can redistribute it and/or modify it under the same terms as Perl itself.

SEE ALSO

Logitech/Slim Devices

`http://www.slimdevices.com/`

mp3tools

`http://www.zevils.com/linux/mp3tools/`

mpgtools

`http://www.dv.co.yu/mpgscript/mpgtools.htm`
`http://www.dv.co.yu/mpgscript/mpeghdr.htm`

mp3tool

`http://www.dtek.chalmers.se/˜d2linjo/mp3/mp3tool.html`

ID3v2

`http://www.id3.org/`

Xing Variable Bitrate

`http://www.xingtech.com/support/partner_developer/mp3/vbr_sdk/`

MP3Ext

`http://rupert.informatik.uni−stuttgart.de/˜mutschml/MP3ext/`

Xmms

`http://www.xmms.org/`