IdEst – ID3 Edit and Scripting Tool

version 2.1, 31 December 2016

Sergey Poznyakoff

Copyright © 2009-2011, 2015, 2016 Sergey Poznyakoff
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, and without restrictions as to the Front and Back-Cover texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Short Contents

1 Introduction 1
2 ID3 Tag Versions 3
3 ID3 Frames 5
4 Viewing Existing Tags 7
5 Modifying Existing Tags 9
6 Copying Tags Between Files 11
7 Deleting Tags and Frames 13
8 Storing Tags in Different ID3 Versions 15
9 Examining File Structure 17
10 Scripting 19
11 Keeping Backup Copies 35
12 Invocation Summary 37
13 How to Report a Bug 41
A ID3 Genre Codes 43
B GNU Free Documentation License 47
Concept Index 57

Table of Contents

1 Introduction 1

2 ID3 Tag Versions 3

3 ID3 Frames 5

4 Viewing Existing Tags 7

5 Modifying Existing Tags 9

6 Copying Tags Between Files 11

7 Deleting Tags and Frames 13

8 Storing Tags in Different ID3 Versions 15

9 Examining File Structure 17

10 Scripting 19

10.1 Using Scripts to List ID3 Frames 22
10.2 Using Scripts to Modify ID3 Frames 24
10.3 Format 25
10.3.1 How to Write Format Modules 26
10.3.2 Existing Formats 27
10.3.2.1 help: List and Describe Available Formats 27
10.3.2.2 framelist: Display List of Frames Present in Each File 27
10.3.2.3 lyrics: Display Lyrics 28
10.3.2.4 pic: Display Attached Pictures 29
10.3.2.5 shortlist: Display Short Information 30
10.4 Batch 30
10.4.1 How to Write New Batch Modules 30
10.4.2 Existing Batch Modules 31
10.4.2.1 help: List and Describe Available Batches 31
10.4.2.2 setlyrics 32
10.4.2.3 setpic: Attach a Picture 32
10.5 Testing Scripts 33

11 Keeping Backup Copies **35**

12 Invocation Summary **37**

13 How to Report a Bug **41**

Appendix A ID3 Genre Codes **43**

Appendix B GNU Free Documentation License **47**

Concept Index **57**

1 Introduction

Many ‘mp3’ files carry additional blocks of information, called ID3 tags. These tags supply additional information about the file, such as artist’s name, the title of the song, etc. There are currently two major versions of these tags. The version 1 is able to keep a predefined number of textual fields of limited length, and is written at the end of the file. The version 2 is much more flexible. It is able to keep arbitrary number of fields, called *frames*, which may be textual or binary data. The frame length is not limited. ID3 tags of this version are written at the beginning of the file, which makes them suitable for streaming.

Idest is a command line tool for manipulating ID3 tags. It allows you to create new tags, and to view, modify or delete the existing ones. When compiled with Guile[1], `idest` allows you to write programs of arbitrary complexity for manipulating ID3 tags and to apply them to any number of files.

The program name is an abbreviation for ‘`ID3 Edit and Scripting Tool`’. When speaking about the whole package, we spell its name as ‘`IdEst`’. When speaking about the program file name, we spell it `idest`. This latter spelling may be capitalized, if it occurs at the beginning of a sentence.

[1] Guile is the *GNU’s Ubiquitous Intelligent Language for Extensions*, `http://www.gnu.org/software/guile`.

2 ID3 Tag Versions

The version 1 of ID3 tags offers a very limited set of possibilities, compared to its successor, version 2. You should know these limitations in order to understand their implications when creating version 1 tags or converting version 2 to version 1.

Properly speaking, the pure version 1 tag is seldom used. It is its modified version, called 1.1 which is used most often.

The version 1.1 tag contains the following frames:

Field	**Width**	**Description**
title	30	The title
artist	30	The artist name
album	30	The album name
year	4	A four-digit year
comment	28	The comment.
track	1	The number of the track on the album.
genre	1	Index in a list of genres, see below.

Table 2.1: ID3v1.1 tag

The last field, '`genre`', merits special notice. It is an ordinal number of genre in a predefined table of genres (see Appendix A [Genre Codes], page 43). When modifying or setting this tag, you should supply one of the values listed in that table (case-insensitive). If the value you supply is not found in that table, the value '`Other`' will be used.

The version 2 is much more advanced and flexible. It was initially described in `http://www.id3.org/id3v2-00`. The current version is 2.4.0 and it is discussed in detail in `http://www.id3.org/id3v2.4.0-frames`. In this manual, unless expressly noted otherwise, when speaking about version 2 we actually mean 2.4.0.

The ID3v2 frames are named using four-character abbreviations. The 7 most used frames, corresponding to the ID3v1 fields, are:

Frame	**V1 field**
TIT2	title
TPE1	artist
TALB	album
TDRC	year
COMM	comment
TRCK	track
TCON	genre

Table 2.2: Some ID3v2 frames

`Idest` 2.1 internally operates on ID3v2 tags of version 2.4.0. It is able to handle any prior versions as well: such tags are implicitly converted to the latest version. When creating or modifying tags, `idest` always stores updated tags in version 2.4.0.

3 ID3 Frames

Each ID3 tag consists of frames. As described above, IDv1 tags contain a fixed set of frames, whereas IDv2 tags can contain any number of this. The frame *ID* is a four-character name which identifies a frame.

There are frames that can appear only once in a tag, and there are ones that can appear multiple times. These latter have some additional fields which serve to discern between them. In `idest` parlance we call these fields frame *qualifiers*. The number and semantics of qualifiers are frame-dependent. For example, the '`COMM`' (comment) frame contains two qualifiers: *language*, which holds a three-letter code of the language the comment is written in, and *content descriptor*, which holds arbitrary string describing the comment.

There are two ways to address a frame: by its ID, and by its fully-qualified name. Addressing the frame by its ID retrieves all instances of that frame. A *fully-qualified name*, on the other hand, provides a way to retrieve a particular instance of the frame. A fully-qualified name consists of frame ID, followed by a colon and a list of qualifier values, delimited with colons. For example, the name '`COMM:eng:my-comment`' will select the '`COMM`' frame which has '`eng`' in its language field and '`my-comment`' in its content descriptor field. Any of qualifiers in a fully-qualified name can be omitted. Such an empty qualifier works as a *wildcard*, matching any value in the actual field. Thus, '`COMM::my-comment`' selects the '`COMM`' frames with content descriptor '`my-comment`', no matter what their language.

To see all the frames along with their qualifiers and a short description, use the `--list-frames` (`-L`) option:

```
$ idest --list-frames
COMM:lang:condesc           Comments
TALB                        Album/movie/show title
TBPM                        BPM (beats per minute)
TCOM                        Composer
...
```

The output it produces consists of two columns: the first one shows the frame ID and its qualifiers (if any). The second one contains a short description of this frame purpose.

To select one or several frames of interest, give their names as argument (a comma-separated list) to the `--filter` (`-F`) argument, e.g.:

```
$ idest --list --filter=COMM,TXXX
COMM:lang:condesc           Comments
TXXX:descr                  User defined text information
```

The `--filter` option is a standard way to abridge `idest` operation to a subset of frames.

4 Viewing Existing Tags

Viewing existing tags is simple. Just give `idest` a list of files to extract information from, with no additional options:

```
$ idest file.mp3
title: Diamonds & Rust
album: Diamonds & Rust
track: 1
comment:
artist: Joan Baez
year: 1975
genre: Folk
```

This operation mode is called *query mode*. By default, `idest` shows all these fields in this order. If there are several comment fields, they will be shown in the fully-qualified form, e.g.:

```
$ idest track01.mp3
title: Plou i fa sol
album: Camins de Tarda
track: 3
comment:eng:my: Comment text
comment:eng:encoder: lame
artist: Josep Tero
year: 1995
genre: Folk
```

If you wish to display another frames, use the `--filter` (`-F`) option:

```
$ idest --filter=artist,title,year file.mp3
artist: Joan Baez
title: Diamonds & Rust
year: 1975
```

The names given in the filter list can be either IDv1 or IDv2 names, `idest` will convert the to IDv2 automatically.

Frames can also be given in a fully-qualified form, for example:

```
$ idest --filter=title,comment::encoder track01.mp3
title: Plou i fa sol
comment:eng:encoder: lame
```

You can also define a string which will be printed instead of the frame name in the output. This string is given as a prefix to the frame name. The two parts are delimited by a percent sign, e.g.:

```
$ idest --filter=Title%title,'Encoded by'%comment::encoder \
        track01.mp3
Title: Plou i fa sol
Encoded by: lame
```

To describe frames in a verbose manner, use the `--describe` (`-D`) option:

```
$ idest --describe --filter=artist,title,year file.mp3
```

```
Lead performer(s)/soloist(s): Joan Baez
Title/songname/content description: Diamonds & Rust
Recording time: 1975
```

For compatibility with previous versions, the `--query` option (or `-q`, for short) is supported. When used without argument it forces the query mode. If argument is supplied, it must be in the same format as for the `--filter` option and has the same effect (e.g. `idest -qartist,title,year file.mp3`).

If the long option form (`--query`) is used, then the frame list must be separated from the option by an equal sign, with no surrounding white space. If the short option form (`-q`) is used, the list must follow the option letter, with no white space in between.

There is a special option which instructs `idest` to output all frames: the `--all` (`-a`) option:

```
$ idest --all track01.mp3
title: Cor i arbre
album: Fronteres
track: 1
comment:eng:Bit_Rate: 320
comment:eng:Sample_Rate: 44100
TENC: Myencoder 1.0
artist: Josep Tero
year: 2009
genre:
```

All textual fields are displayed using the current locale settings. Sometimes it may be necessary to force displaying them in another locale. The `--charset` option allows you to do so. Its argument is a valid character set name. For example

```
$ idest --charset=iso-8859-2 track06.mp3
```

This will cause all textual tags to be converted to iso-8859-2 on output. Notice that such conversion is not always possible, for example if the tag is stored internally in UTF-8 and is using characters not present in the iso-8859-2 plane.

You may occasionally encounter files with textual frames stored as iso-8859-1 strings, but actually using another 8-bit encoding. Such frames are displayed as sequences of unintelligible characters. You can display them properly if you know or can guess the actual character set they were written in. To do so, use the `--broken-8bit-charset` option. For example, the following command will assume all textual options use the iso-8859-2 character set and will convert them to the output character set:

```
$ idest --broken-8bit-charset=iso-8859-2 dm.mp3
```

5 Modifying Existing Tags

To modify a particular frame, use the `--set` (`-s`) option. For example:

```
$ idest --set artist='Jacques Brel' track01.mp3
```

Several frames can be set at once. To do so you can either supply a separate `--set` option for each frame, or to give a single `--set` option followed by as many frame assignments as you need, for example:

```
$ idest --set artist='Jacques Brel' \
        --set title='Ne me quitte pas' track01.mp3
```

or

```
$ idest --set artist='Jacques Brel' \
        title='Ne me quitte pas' track01.mp3
```

You can use fully qualified form (see [fully-qualified name], page 5) for frames that require it:

```
$ idest --set comment:eng:My_comment='Noise reduction on' \
        track01.mp3
```

In the example above, if a comment with this content descriptor exists, its content will be replaced with the new one. If not, a new comment frame will be created.

If a frame which can appear multiple times (such as e.g. comment) is being set without qualifiers, all existing frames of this type will be removed and replaced with the new instance. Its qualifiers will be set to default values.

Textual strings are assumed to be written in the current locale. If that's not the case, use the `--charset` option, e.g.

```
$ idest --charset=latin1 --set artist='Lluís Llach' *.mp3
```

Textual strings are stored in UTF-8 by default. If you prefer another encoding, specify it with the `--encoding` option. The ID3 specification offers the following encodings: '`iso-8859-1`' (or '`latin1`'), '`utf-8`', '`utf-16`', and '`utf-16be`' (the suffix stands for "big-endian"). For example, to store texts in '`utf-16`':

```
$ idest --encoding=utf-16 --set album='Itaca'  *.mp3
```

Not all devices support full ID3 specification. Most of them support only a subset of it. The `--fixup` command line option is provided to convert ID3 tags to a form understood by most devices. The usage is simple:

```
$ idest --fixup *.mp3
```

If the input tags also contain malformed 8-bit encodings (see [broken 8bit encoding], page 8), you can fix them as shown below:

```
$ idest --broken-8bit-encoding=iso-8859-1 --fixup *.mp3
```

6 Copying Tags Between Files

To copy tags from one file to another, use the `--copy` (`-c`) option. Its argument supplies the *source file*. Non-optional arguments supply destination files:

```
$ idest --copy sample.mp3 track1.mp3 track2.mp3
```

As a result of this operation all tags from `sample.mp3` will be copied to `track1.mp3` and `track2.mp3`.

As in other operations, you can abridge the scope of copying to a certain subset of frames by using the `--filter` option, e.g.:

```
$ idest --copy sample.mp3 --filter TPE1,TCOM \
        track1.mp3 track2.mp3
```

You can also use `--copy` together with `--set` in a single invocation. In this case, the frames will first be copied from the source file and then the resulting tags will be modified according to the `--set` options. For example:

```
$ idest --copy sample.mp3 --filter TPE1,TCOM \
        --set year=2003 track1.mp3 track2.mp3
```

7 Deleting Tags and Frames

The `--delete` (`-d`) option instructs `idest` to remove ID3 tags from the file (or files). If no argument is specified, all tags are deleted:

```
$ idest --delete *.mp3
```

After this operation, all ID3 data are irrevocably lost, so use it with caution.

A list of frame names can be given either with the `--filter` option, or (for compatibility with `idest` 1.x) as an argument to `--delete` (similarly to `--query`). For example, to delete only comment and genre tags:

```
$ idest --delete --filter=comment,genre *.mp3
```

or

```
$ idest --delete=comment,genre *.mp3
```

Specifying '`comment`' (a non-qualified form) results in removing all comment frames. To remove a particular one, use its qualified form:

```
$ idest --delete --filter=comment::Bit_Rate track01.mp3
```

The same applies to other frames that can appear multiple times (see Chapter 3 [Frames], page 5).

8 Storing Tags in Different ID3 Versions

There are currently two major versions of ID3 format (see Chapter 1 [Intro], page 1). A file may contain tags in any format, or even in both formats at once. By default, the `--set` option will store data in the same format as found initially in the file. If the file did not contain any tags before running `idest --set`, new tags will be created in both versions 1 and 2. This choice can be overridden by setting the desired tag version with the `--default-id-version` (`-U`) option. This option expects a comma-separated list of version numbers (1 and 2) as its argument. For example, `idest --default-id-version=1` will write new tags in version 1. The default setting corresponds to `--default-id-version=1,2`.

The `--convert` (`-C`) command line option takes a a comma-separated list of ID3 major version number and converts the existing tags to the given formats. If there is no existing data, the new tags will be created in these formats. Thus, for example:

```
$ idest --convert=1 *.mp3
```

changes the ID3 format in all '`*.mp3`' files to version 1. The command:

```
$ idest --convert=1 --set artist='Jacques Brel' *.mp3
```

sets the '`artist`' field on each file. Those files that already had ID3 data will be converted to version 1. Those that did not, will have it created in version 1 format.

The `--id-version` (`-V`) instructs the program to write new and changed tags in the specified ID3 format. In the contrast to `--convert`, this option does not affect files which underwent no changes. On the other hand, it differs from `--default-id-version` in that it sets new tag format unconditionally, whereas the latter does so only if the input file contained no ID3 tags.

9 Examining File Structure

The `--info` (`-i`) option instructs `idest` to show the structure of input files. The output is formatted as a sequence of keyword / value pairs, as shown in the example below:

```
$ idest --info jt_lluny.mp3
file: jt_lluny.mp3
ntags: 2
version: 2.4.0
offset: 0
length: 2131
version: 1.1
offset: 2725554
length: 128
```

The first two lines show the name of the input file and the number of ID3 tags in it. Following are tag descriptions formatted as three values for each tag. The '`version`' line shows the tag version (major and minor numbers, separated by dots). The '`offset`' line shows the offset of this tag in the file, and the '`length`' line contains size of this tag in bytes.

10 Scripting

`Idest` offers a scripting facility, which makes it possible to extend its functionality beyond the basic operations, described in previous chapters. Scripts must be written in Scheme, using ‘`Guile`’, the *GNU’s Ubiquitous Intelligent Language for Extensions*. For information about the language, refer to *Revised(5) Report on the Algorithmic Language Scheme*. For a detailed description of Guile and its features, see Section “Overview” in *The Guile Reference Manual*.

The scripting mode is enabled when the option `--script` (`-S`) is given in the command line. This option stops further option processing, so any other `idest` command line options must be given before it. The argument to this option specifies the name of the script file:

```
$ idest --script list.scm *.mp3
```

You can omit the ‘`.scm`’ suffix, as `idest` will try it automatically (see below).

When this option is given, the following operations are performed:

1. The program looks for files `.idest.scm`, `$HOME/.idest.scm` and *`guile-site-dir`*`/idest/idest.scm` in that order. Here, ‘`$HOME`’ stands for the user home directory and *guile-site-dir* stands for the Guile site-wide directory, as described in [guile-site-dir], page 20. If any of these files is found, it is loaded as a Scheme source code and further search is discontinued. This allows you to configure Guile settings on per-directory, per-user and site-wide basis.

   This step is omitted if the program is given the `--no-init-files` (`-N`) option.

   When a startup file is loaded, the list of files which were to be tried after it is passed to it as arguments. This allows for chain-loading all files in the list using the following code:

   ```
   (let load-loop ((name-list (cdr (command-line))))
     (if (not (null? name-list))
         (let ((name (car name-list)))
   (load-loop (cdr name-list))
   (if (file-exists? name)
       (primitive-load name)))))
   ```

2. Unless the supplied script name contains directory separators (‘`/`’), it is searched in the Guile’s `%load-path`. The default load path is formed as follows:

   ```
   version-site-dir
   .
   package-site-dir
   guile-site-dir
   %load-path
   ```

where the components are as follows:

%load-path
: The standard Guile load path (see Section "Build Config" in *The Guile Reference Manual*).

guile-site-dir
: This directory is selected at compile time using the rules below. Its value is returned by the `(%idest-guile-site-dir)` primitive:

  1. Determine actual value of the default Guile site directory, by inspecting the value returned by the '`%site-dir`' primitive.
  2. If that value lies under the current installation prefix, use it.
  3. Otherwise, if the `--with-guile-site-dir` option is supplied:
     a. If it is used without arguments, use the '`%site-dir`' value.
     b. Otherwise, the value of this option is taken as new site directory.
  4. Otherwise, a warning is issued and `$(datadir)/guile/site` is used as the site directory.

  The reason for using this directory is described in `http://www.gnu.org.ua/software/gint/#guile-site-dir`.

  If `guile-site-dir` coincides with the standard `%site-dir`, this part is omitted, because the latter is always present in the `%load-path`.

package-site-dir
: This is the directory for installing version-independent `idest` files. It is formed as follows:

  ```
  guile-site-dir/idest
  ```

  This value is returned by the `(%idest-package-site-dir)` primitive.

version-site-dir
: This is the directory for installing version-dependent `idest` files. It is formed as follows:

  ```
  package-site-dir/2.1
  ```

  This value is returned by the `(%idest-version-site-dir)` primitive.

The load path can be modified using the `--load-path` (`-P`) and `--prepend-load-path` (`-p`) command line options. Both options take as argument a list of directory names, separated by colons. The `--load-path` option adds these directories to the tail of the load path list. The `--prepend-load-path` option adds them to the head of the load path list.

The script is loaded via `primitive-load-path` (see Section "primitive-load-path" in *The Guile Reference Manual*), so `idest` will consult the `%load-extensions` list and try suffixes from that list as described in Section "%load-extensions" in *The Guile Reference Manual*).

3. The script is read and evaluated.

   The script can access command line arguments via the usual `command-line` function (see Section "command-line" in *The Guile Reference Manual*). It can also modify the argument list (e.g. by removing its command line options). It must not, however, modify '`argv[0]`'. Any changes it does to the argument list become visible to `idest`. The only requirement is that the modified argument list consist of the script name (as argv[0]) and input file names.

4. The script's *main function* is applied to each input file in turn.

The main function must be declared as:

`idest-main` *file frames* [Function]

It takes two arguments. The *file* argument supplies the name of the file being processed. The *frames* argument is a list of ID3 frames read from that file. Each element of *frames* is a pair, with the frame name in its `car` and an association list of *frame properties* in its `cdr`.

The properties are identified by property names, which are Scheme symbols. The following property names are defined:

text
: Value of this frame, as a string.

descr
: Frame description. It is a string, verbosely describing the frame. For example, the description of '`TRCK`' frames is '`Track number/position in set`'.

  These are the same descriptions that are output with the `--describe` option (see [describe], page 7).

rawdata
: Unsupported or partially-supported frames contain only this property. Its value is a list of frame fields. Each field is represented by a triplet '`(`*`ord type value`*`)`', where *ord* is the ordinal number of that field in frame, *type* is its type (integer) and *value* is its value. If *type* is one of numeric types, *value* is the numeric value converted to string (as per `number->string`). If *type* is a string type, *value* contains the string in the appropriate encoding. Otherwise, *value* holds the field value as a binary string. Each byte in such a string is represented by two hexadecimal digits. For example, '`AB\n`' is represented as '`41420A`'.

More properties are defined at a per-frame basis to represent frame qualifiers. They are named after corresponding qualifiers as listed in `--list-frames` output (see [describe], page 7). For example, for '`comment`' ('`COMM`') frames:

lang	A three-letter code of the language in which the text is written.
condesc	Content descriptor.

The mode in which input files are open is controlled by the `idest-readonly` variable:

`idest-readonly` [Variable]

This is a boolean variable that indicates whether `idest-main` can modify tag frames. If its value is `#t`, the function return value will be ignored, and input files will be opened in read-only mode. This is the default.

If `idest-readonly` is '`#f`' the `idest-main` function should return the new list of frames. If it returns an empty list, all existing frames will be deleted. If the function chooses not to modify any frames, it must return `#f`.

The two following sections show how to write script files. The sample scripts they discuss can be found in subdirectory `examples` of the `idest` distribution.

10.1 Using Scripts to List ID3 Frames

This section illustrates how to use the scripting facility for listing the contents of ID3 tags.

The simplest way to list all frames using a Guile script is:

```
;; list1.scm -- lists all frames.
(define (idest-main name frames)
  (display name)
  (newline)
  (for-each
   (lambda (frame)
     (display frame)
     (newline))
   frames))
```

Here is a sample output:

```
$ idest --script list1.scm track01.scm
track01.mp3
(TIT2 (descr . Title/songname/content description)
      (text . Cor i arbre))
(TRCK (descr . Track number/position in set)
      (text . 1))
(COMM (descr . Comments) (condesc . Bit_Rate)
      (lang . eng) (text . 320))
(TENC (descr . Encoded by) (text . Myencoder 1.0))
```

```
(COMM (descr . Comments) (condesc . Sample_Rate)
      (lang . eng) (text . 44100))
```

As mentioned above, a script can access the command-line arguments. To illustrate this, let's modify the `list1.scm` to display only a subset of frames, given as a comma-separated list in the first argument. To do so, we will need a list of requested frames:

```
(define frame-list '())
```

The main function consults this list to see whether to display a frame:

```
(define (idest-main name frames)
  (display name)
  (newline)
  (for-each
   (lambda (frame)
     (if (member (car frame) frame-list)
 (begin
   (display frame)
   (newline))))
   frames))
```

Finally, the following code initializes `frame-list` from the first argument and removes that argument from the list seen by `idest`. Note that the 0th argument is the name of the script itself, and it should not be modified.

```
(let ((cmd (command-line)))
  (cond
   ((< (length cmd) 3)
    (error "usage: idest -S list2 FRAME-LIST FILE...")
    (exit 1))
   (else
    (set! frame-list (string-split (list-ref cmd 1) #\,))
    (set-program-arguments (cons (car cmd)
                                 (list-tail cmd 2))))))
```

The full script text is then:

```
;; list2.scm -- lists only requested frames.
(define frame-list '())

(define (idest-main name frames)
  (display name)
  (newline)
  (for-each
   (lambda (frame)
     (if (member (car frame) frame-list)
 (begin
   (display frame)
   (newline))))
   frames))
```

```
(let ((cmd (command-line)))
  (cond
   ((< (length cmd) 3)
    (error "usage: idest -S list2 FRAME-LIST FILE...")
    (exit 1))
   (else
    (set! frame-list (string-split (list-ref cmd 1) #\,))
    (set-program-arguments (cons (car cmd)
                                 (list-tail cmd 2))))))
```

Sample usage:

```
$ idest --script list2 TIT2,TENC track01.scm
(TIT2 (descr . Title/songname/content description)
      (text . Cor i arbre))
(TENC (descr . Encoded by) (text . Myencoder 1.0))
```

A more elaborate example will print, for each input file, its name, followed by the title, artist name and year, as shown in this sample output:

```
$ idest -S shortlist *.mp3
dnr.mp3: Diamonds & Rust by Joan Baez, 1975
ams.mp3: Amsterdam, by Jacques Brel, 1968
```

To implement this, we would need a function that returns the value of a given frame from the frame list. Remember, that the latter is a list of pairs, so the task is achieved easily by using the `assoc-ref` function:

```
(define (get-frame code frames)
  (or (assoc-ref
       (or (assoc-ref frames code) '())
       'text)
      "unknown"))
```

The inner `assoc-ref` selects a requested frame. An empty list is returned if such a frame is not found. The outer `assoc-ref` selects the '`text`' property.

Now, we define the main function:

```
(define (idest-main name frames)
  (format #t "~A: ~A by ~A, ~A~%"
  name
  (get-frame "TIT2" frames)   ; Title
  (get-frame "TPE1" frames)   ; Artist
  (get-frame "TDRC" frames))) ; Year
```

10.2 Using Scripts to Modify ID3 Frames

This section illustrates how to write scripts that modify ID3 tags. We will write a script which creates a new value for the '`title`' (`TIT2`) frame from the name of the input file. The title is created using the following algorithm:

1. Strip off leading directories and the '`.mp3`' suffix.

2. Replace underscores with spaces.

Here is the implementation:

```
;; settitle.scm - set title (TIT2) frame based on
;; the file name.

(use-modules (ice-9 regex)
     (srfi srfi-13))

(define (idest-main file frames)
  (cond
   ((string-match "(.*)\\.mp3" file) =>
    (lambda (match)
      (cons
       (cons "TIT2"
     (list
      (cons
       'text
       (string-map
(lambda (c)
  (if (char=? c #\_) #\space c))
(match:substring match 1)))))
       ;;
       (filter
(lambda (elt)
  (not (string=? (car elt) "TIT2")))
frames))))
   (else
    #f)))

(set! idest-readonly #f)
```

An example of using this script on all files in the current directory:

```
$ idest --script settitle *.mp3
```

10.3 Format

Formats are advanced scripting feature which allows for extending `idest` output by writing an appropriate script in Scheme. A format is invoked using the `--format` (`-H`) command line option. The format name is given as argument to that option. Similarly to the `--source` option, the `--format` option stops further argument processing and passes the rest of arguments to the format module, which is supposed to remove its option arguments and leave only input file names. For example:

```
$ idest --format=framelist -Q -l *.mp3
```

This example invokes `idest` with the '`framelist`' format (see Section 10.3.2.2 [framelist], page 27). The `-Q` and `-l` flags are format options.

10.3.1 How to Write Format Modules

The source for format module *name* must be saved in the file named `name.scm` located in the subdirectory `idest/format` somewhere in the Guile load path. It must begin with the following clause:

```
(define-module (idest format name))
```

The module must define and export the '`idest-main`' function, whose calling convention and return type is the same as that in the usual `idest` scripts (see [idest-main], page 21). For example, the following is a simplified version of the '`framelist`' module (see Section 10.3.2.2 [framelist], page 27):

```
(define-module (idest format framelist))

(define frame-list '())

(define-public (idest-main name frames)
  (for-each
    (lambda (elt)
      (cond
       ((member (car elt) frame-list)
        (display (car elt))
        (newline))))
    frames))
```

If the module needs to process command line arguments, it may not do so in the main code, as the usual `idest` modules do. Instead, it should export a special function, '`idest-init`', defined as:

```
(define-public (idest-init)
  ...)
```

This function analyzes the command line, removes the consumed modules options and returns. For example:

```
(define-public (idest-init)
  (let ((cmd (command-line)))
    (cond
     ((< (length cmd) 3)
      (error "usage: idest --format=framelist
             FRAME-LIST FILE...")
      (exit 1))
     (else
      (set! frame-list (string-split (list-ref cmd 1) #\,))
      (set-program-arguments
        (cons (car cmd) (list-tail cmd 2)))))))
```

The module should also export the symbol '`description`', which should contain a string with a concise description of the module. This description will be shown in the `--format=help` output (see Section 10.3.2.1 [help format], page 27). For example:

```
(define-public description
  "display a list of frames defined in each file")
```

10.3.2 Existing Formats

`Idest` is shipped with a set of predefined formats. These formats are found in the `scheme/idest/format` subdirectory of the source tree. They are installed into the `'version-site-dir'/format` directory (see [version-site-dir], page 20).

10.3.2.1 help: List and Describe Available Formats

The '`help`' format searches the load path for available format modules and lists them. For each module its name and short description are shown on a separate line. The output is sorted alphabetically by the format name:

```
$ idest --format=help
framelist: display a list of frames defined in each file
lyrics: display lyrics (the USLT content), if present
pic: show attached picture (APIC frame) or save it on disk
shortlist: display title, artist name and year
```

If '`help`' is used with the `--which` (`-w`) option, the format includes the directory where the module is found:

```
$ idest --format=help --which
framelist (/usr/share/idest/format): display a list of frames
defined in each file
...
```

10.3.2.2 framelist: Display List of Frames Present in Each File

The '`framelist`' format displays a list of ID3 frames present in each input file, e.g.:

```
$ idest --format=framelist file.mp3
TIT2
TRCK
COMM
TENC
COMM
```

The following command line options are understood:

`-F`
`--full` Display all qualifiers. For example:

```
$ idest --format=framelist --full file.mp3
```

```
TIT2 descr="Title/songname/content description"
TRCK descr="Track number/position in set"
COMM descr="Comments" lang="eng" condesc=""
TENC descr="Encoded by"
COMM descr="Comments" lang="cat" condesc=""
```

`-f` *`flist`*
`--frames` *`flist`*
Display only frames from *flist*, which is a list of frame names, separated by commas.

`-Q`
`--qualified`
Display frames in qualified form:

```
$ idest --format=framelist --qualified file.mp3
TIT2
TRCK
COMM:eng:
TENC
COMM:cat:
```

`-l`
`--single-line`
Fit output on single-line, e.g.:

```
$ idest --format=framelist --single-line file.mp3
TIT2,TRCK,COMM,TENC,COMM
```

`-h`
`--help` Show a short help summary

10.3.2.3 lyrics: Display Lyrics

The '`lyrics`' format displays the lyrics (as found in the '`USLT`' frame). The text is preceded by the song title from the '`TIT2`' frame, e.g.:

```
$ idest --format lyrics file.mp3
How doth the little

How doth the little crocodile
Improve his shining tail,
And pour the waters of the Nile
On every golden scale!

How cheerfully he seems to grin,
How neatly spreads his claws,
And welcomse little fishes in
With gently smiling jaws!
```

If the environment variable `PAGER` is set, its value is used to paginate the output.

This module supports the following command line options:

`-l` *name*
`--lang` *name*
: Select '`USLT`' frames with *name* as the value of their '`lang`' qualifier.

`-c` *text*
`--content` *text*
: Select '`USLT`' frames with *text* as the value of their '`condesc`' qualifier.

`-h`
`--help` Show a short help summary

10.3.2.4 pic: Display Attached Pictures

The '`pic`' format displays or stores on disk attached pictures. It supports the following options:

`-v` *prog*
`--viewer` *prog*
: Use *prog* to view images (default: `xv`).

`-d` *text*
`--description` *text*
: Look for pictures with this descriptive text.

`-m` *type*
`--mime-type` *type*
: Look for pictures with this MIME type.

`-s`
`--store` Store pictures on disk, instead of displaying them. The picture names are created by expanding the file name template, given with the following option:

`-f` *template*
`--file` *template*
: Set the template for output file names (implies –store). The *template* can contain the following meta-characters:

Char	**Expands to**
~D	Input file directory part
~N	Input file base name
~C	Content description
~T	Mime type without the '`image/`' prefix
~P	Picture type
~I	PID of the `idest` process

The default template is '`/tmp/~I-~N.~T`'.

`-h`
`--help` Show a short help summary

10.3.2.5 shortlist: Display Short Information

The '`shortlist`' format module is similar to the '`shortlist.scm`' example program, discussed in [shortlist example], page 24. It does not take any command line options – everything after the format name is treated as file names:

```
$ idest --format=shortlist *.mp3
dnr.mp3: Diamonds & Rust by Joan Baez, 1975
ams.mp3: Amsterdam, by Jacques Brel, 1968
```

10.4 Batch

Batch modules or *batches* are `idest` module files located in a set of predefined directories which apply a set of modifications to the argument files. In other words, batches are file-modifying counterpart of formats. A batch is invoked using the `--batch` (`-B`) command line option. The batch name is given as argument to that option. Similarly to the `--source` and `--format` options, the `--batch` option stops further argument processing and passes the rest of arguments to the batch module, which is supposed to remove its option arguments and leave only the input file names. For example:

```
$ idest --batch=setpic -f cover.png file.mp3
```

In this example, '`setpic`' is the batch module name, '`-f cover.png`' are its arguments (see Section 10.4.2.3 [setpic], page 32), and '`file.mp3`' is the argument file.

10.4.1 How to Write New Batch Modules

The rules for writing batch modules are similar to those for formats (see Section 10.3.1 [format modules], page 26) with only few differences.

The source for format module *name* must be saved in the file named ***name*****`.scm`** located in the subdirectory `idest/batch` somewhere in the Guile load path. It must begin with the following clause:

```
(define-module (idest batch name))
```

The module must define and export the '`idest-main`' function, whose calling convention is the same as that in the usual `idest` scripts (see [idest-main], page 21). This function must return the new list of frames. If it returns an empty list, all existing frames will be deleted. If the function chooses not to modify any frames, it must return `#f`.

If the module needs to process command line arguments, it should do so in the function '`idest-init`', defined as:

```
(define-public (idest-init)
  ...)
```

Finally, the module should export the symbol '`description`' with a concise description of the module. This description will be shown in the `--batch=help` output (see Section 10.4.2.1 [help batch], page 31).

To illustrate this, here is the code for module '`delfrm`', which removes the requested frames from all argument files:

```
(define-module (idest batch delfrm))

(define-public description
 "remove requested frames from the input files")

(define frame-list '())

(define-public (idest-main)
  (filter
   (lambda (frame)
     (not (member (car frame) frame-list)))
   frames))

(define-public (idest-init)
  (let ((cmd (command-line)))
    (cond
     ((< (length cmd) 3)
      (error
       "usage: idest --batch=delfrm FRAME-LIST FILE...")
      (exit 1))
     (else
      (set! frame-list (string-split (list-ref cmd 1) #\,))
      (set-program-arguments
        (cons (car cmd) (list-tail cmd 2)))))))
```

10.4.2 Existing Batch Modules

`Idest` is shipped with a set of predefined batch modules. These modules are found in the `scheme/idest/batch` subdirectory of the source tree. They are installed into the `'version-site-dir'/batch` directory (see [version-site-dir], page 20).

10.4.2.1 help: List and Describe Available Batches

The '`help`' batch searches the load path for available batch modules and lists them. For each module its name and short description are shown on a separate line. The output is sorted alphabetically by the format name:

```
$ idest --format=help
setlyrics: set song lyrics (USLT frame) from a file
setpic: set attached picture from a file
```

If '`help`' is used with the `--which` (`-w`) option, the format includes the directory where the module is found:

```
    $ idest --format=help --which
    setlyrics (/usr/share/idest/format): set song lyrics
    (USLT frame) from a file
    ...
```

10.4.2.2 setlyrics

The '`setlyrics`' batch reads the text from the specified file (or standard input, if no file is given) and stores it in the '`USLT`' frame. It supports the following command line options:

`-f` *`file`*
`--file` *`file`*
: Read text from *file* (default: stdin).

`-l` *`name`*
`--lang` *`name`*
: Set language in which the lyrics is written, i.e. the value of the '`lang`' qualifier (default: '`eng`').

`-c` *`text`*

`--content` *`text`*
: Set content description.

`-h`
`--help` Show a short help summary

10.4.2.3 setpic: Attach a Picture

The '`setpic`' module reads a picture from a supplied file and attaches it to the argument files. It supports the following options:

`-f` *`file`*
`--file` *`file`*
: Read picture from *file*. This option is required.

`-d` *`text`*
`--description` *`text`*
: Set the value of '`condesc`' qualifier.

`-m` *`type`*
`--mime-type` *`type`*
: Set MIME type. By default it is deduced from the picture file suffix.

`-p` *`num`*

`--pic-type` *`num`*
: Set picture type (a decimal number). Default is '`0`'.

`-h`
`--help` Show a short help summary

For example:

```
$ idest --batch setpic --file cover.png \
        --description='Album Cover' file.mp3
```

10.5 Testing Scripts

When writing a script which modifies tags, it is good idea to test it before applying it to your data. `Idest` provides a special option for that: `--dry-run` (`-n`, e.g.:

```
$ idest --dry-run --script settitle *.mp3
```

This will run your script as usual, but instead of applying the changes to the input files, `idest` will verbosely print results of each invocation of '`idest-main`'. When `--dry-run` is used, input files are opened in read-only mode.

This option works with batch files as well, e.g.:

```
$ idest --dry-run --batch delfrm *.mp3
```

Here is an example of the dry-run output, obtained from the command above:

```
dry-run: loading ../examples/settitle.scm ...
dry-run: loading /usr/share/guile/1.8/ice-9/regex.scm ...
dry-run: loading /usr/share/guile/1.8/srfi/srfi-13.scm ...
File Tinc_un_clavell_per_a_tu.mp3
(TIT2 (text . Tinc un clavell per a tu))
(TALB (descr . Album/movie/show title) (text . Maremar))
...
```

The first frame shown ('`TIT2`') was produced by `settitle.scm` (see the previous chapter). Rest of frames come from the input file itself.

Notice the diagnostics lines which start with '`dry-run`'. In dry-run mode `idest` verbosely reports the full file names of all files it loads. In this particular case, the line

```
dry-run: loading ../examples/settitle.scm ...
```

shows the full path of the script file itself, whereas the two lines

```
dry-run: loading /usr/share/guile/1.8/ice-9/regex.scm ...
dry-run: loading /usr/share/guile/1.8/srfi/srfi-13.scm ...
```

reflect the `use-modules` clause at the beginning of `settitle.scm` (see Section 10.2 [settitle.scm], page 24).

Implementation note

The '`dry-run`' mode is actually implemented as a usual `idest` Guile script, named `dry-run.scm`. The script is installed to the package script directory. Its source can be found in the subdirectory `scheme` of the `idest` distribution.

11 Keeping Backup Copies

`Idest` offers options for making backups of files before modifying them. Two ways of creating backup copies are supported. First, backups may be made by copying the file to another file before modifying it. This backup method is enabled using the `--backup` command line option. This option takes a single optional argument, which specifies *backup method*, i.e. the naming scheme for backup copies. If used without the argument, the value of the `VERSION_CONTROL` environment variable is used. And if `VERSION_CONTROL` is not set, the '`existing`' method is assumed.

Available backup methods are:

'`never`'
'`simple`'
: Make simple backups. The backup file name is created by appending the *backup suffix* ('`~`' character by default) to the original file name. If a file with such name already exists, this algorithm is applied again, until a unique name is found.

  For example, first call to:

```
$ idest --backup=simple track01.mp3
```

  will create backup copy in file `track01.mp~`. Second invocation of the same command will create a backup file named `track01.mp~~`, and so on.

  The default backup suffix is '`~`', but it can be changed using the `--backup-suffix` command line option or `SIMPLE_BACKUP_SUFFIX` environment variable.

'`t`'
'`numbered`'
: Always make numbered backups. The backup file name is created by appending a unique numeric suffix to the original file name. For example, when using:

```
$ idest --backup=t track01.mp3
```

  the first backup will be called `track01.mp3.~0~`, the second one will be called `track01.mp3.~1~`, etc.

'`nil`'
'`existing`'
: Make numbered backups of files that already have them, simple backups of the others.

Yet another way to create backup copies is to copy the file to be modified to a separate directory. It can be requested with the `backup-directory` option, e.g.:

```
$ idest --backup-directory=/var/backups track01.mp3
```

If the backup directory already contains a copy of the file, the new backup name will be chosen using the method set with the `--backup` command line option.

12 Invocation Summary

This chapter summarizes all available command line options. Options are listed in alphabetical order. Optional arguments are enclosed in square brackets.

`-a`
`--all`
: Query all frames. See [all-frames query], page 8.

`--backup[=`*`control`*`]`
: backup before modifying, choose version *control*. See Chapter 11 [Backups], page 35.

`--backup-directory=`*`dir`*
: Backup to given directory. See Chapter 11 [Backups], page 35.

`--backup-suffix=`*`suf`*
: Set backup suffix, instead of the default '~'. See Chapter 11 [Backups], page 35.

`--broken-8bit-encoding=`*`charset`*
: Textual frames are stored as '`ISO-8859-1`' strings, but are actually using the specified 8bit *charset*. Use this option to properly convert such frames (see [broken 8bit encoding], page 8), or to fix them (see [fixup], page 9).

`-C `*`version`*
`--convert=`*`version`*
: Create tags in given *version*, and convert existing ones to *version*. Argument is a comma-separated list of major version numbers. See Chapter 8 [ID Versioning], page 15.

`-c `*`file`*

`--copy=`*`file`*
: Copy tags from *file* to destination files. See Chapter 6 [Copy], page 11.

`--charset=`*`name`*
: In query mode – convert textual strings to character set *name* on output.

  In modify mode – input strings are written using character set *name*.

  By default, character set is deduced from the locale settings in both cases.

`-d[`*`flist`*`]`
`--delete[=`*`flist`*`]`
: Delete ID3 tags. The *flist* is a comma-separated list of the names of frames to delete. If *flist* is not given, all frames are deleted. See Chapter 7 [Delete], page 13.

`-D`
`--describe`
Print verbose frame descriptions instead of short names. See [describe], page 7.

`--encoding=`*`name`*
Specifies encoding for storing textual fields in ID3 tags. Valid only in modify mode. Valid encoding names are:

iso-8859-1 latin1	
utf-8	This is the default.
utf-16	UTF-16, little-endian
utf-16be	UTF-16, big-endian

`-F` *`flist`*
`--filter=`*`flist`*
Operate only on frames from *flist*. This option affects the following options: `--copy` (see [filter–copy], page 11), `--query` (see [filter–query], page 7), `--delete` (see [filter–delete], page 13) and `--list-frames` (see [filter–list-frames], page 5).

`--fixup` Attempt to fix the ID tags so that they are understood by most devices.

`-h`
`--help` Print a short help list.

`-i`
`--info` Show tag structure information. See Chapter 9 [Structure], page 17.

`--latin1` Same as `--encoding=latin1`.

`-L`

`--list-frames`
List the supported ID3v2 frames. See [list-frames], page 5.

`-P` *`path`*

`--load-path=`*`path`*
Append *path* to the Guile load path (see [load-path], page 19). The argument is a list of directory names separated by colons.

`-p` *`path`*
`--prepend-load-path=`*`path`*
Add *path* to the beginning of the Guile load path (see [load-path], page 19). The argument is a list of directory names separated by colons.

`-N`
`--no-init-files`
: Do not load Scheme init files (see [startup files], page 19).

`-n`
`--dry-run`
: Initiate the *dry-run mode*. See Section 10.5 [dry-run], page 33.

`-q[`*`flist`*`]`
`--query[=`*`flist`*`]`
: Query mode. The *flist* is a comma-separated list of the names of frames to query. If not given, it defaults to '`title,album,track,comment,artist,year,genre`'. See Chapter 4 [View], page 7.

`-S `*`file`*
`--script=`*`file`*
: Guile script name. See Chapter 10 [Scripting], page 19.

`-s `*`field=value`*
`--set=`*`field=value`*
: Set *field* ID3 field to the given *value*. See Chapter 5 [Modify], page 9.

`--trace[=`*`level`*`]`
: Start with debugging evaluator and backtraces. See Chapter 10 [Scripting], page 19.

`-V `*`version`*
`--id-version=`*`version`*
: Write new and changed tags in the given ID3 version. Argument is a comma-separated list of major version numbers. See Chapter 8 [ID Versioning], page 15.

`-U `*`version`*
`--default-id-version=`*`version`*
: Create new tags in the given *version*. Argument is a comma-separated list of major version numbers. See Chapter 8 [ID Versioning], page 15.

`--version`
: Print program version and copyright information.

`--usage` Print a short usage message.

13 How to Report a Bug

Email bug reports to `bug-idest@gnu.org.ua` (or `gray+idest@gnu.org.ua`). Please include a detailed description of the bug and information about the conditions under which it occurs, so we can reproduce it.

Appendix A ID3 Genre Codes

The following genres are defined in ID3v1:

0	Blues
1	Classic Rock
2	Country
3	Dance
4	Disco
5	Funk
6	Grunge
7	Hip-Hop
8	Jazz
9	Metal
10	New Age
11	Oldies
12	Other
13	Pop
14	R&B
15	Rap
16	Reggae
17	Rock
18	Techno
19	Industrial
20	Alternative
21	Ska
22	Death Metal
23	Pranks
24	Soundtrack
25	Euro-Techno
26	Ambient
27	Trip-Hop
28	Vocal
29	Jazz+Funk
30	Fusion
31	Trance
32	Classical
33	Instrumental
34	Acid
35	House
36	Game
37	Sound Clip
38	Gospel
39	Noise
40	AlternRock
41	Bass
42	Soul

43	Punk
44	Space
45	Meditative
46	Instrumental Pop
47	Instrumental Rock
48	Ethnic
49	Gothic
50	Darkwave
51	Techno-Industrial
52	Electronic
53	Pop-Folk
54	Eurodance
55	Dream
56	Southern Rock
57	Comedy
58	Cult
59	Gangsta
60	Top 40
61	Christian Rap
62	Pop/Funk
63	Jungle
64	Native American
65	Cabaret
66	New Wave
67	Psychedelic
68	Rave
69	Showtunes
70	Trailer
71	Lo-Fi
72	Tribal
73	Acid Punk
74	Acid Jazz
75	Polka
76	Retro
77	Musical
78	Rock & Roll
79	Hard Rock
80	Folk
81	Folk-Rock
82	National Folk
83	Swing
84	Fast Fusion
85	Bebob
86	Latin
87	Revival
88	Celtic

89	Bluegrass
90	Avantgarde
91	Gothic Rock
92	Progressive Rock
93	Psychedelic Rock
94	Symphonic Rock
95	Slow Rock
96	Big Band
97	Chorus
98	Easy Listening
99	Acoustic
100	Humour
101	Speech
102	Chanson
103	Opera
104	Chamber Music
105	Sonata
106	Symphony
107	Booty Bass
108	Primus
109	Porn Groove
110	Satire
111	Slow Jam
112	Club
113	Tango
114	Samba
115	Folklore
116	Ballad
117	Power Ballad
118	Rhythmic Soul
119	Freestyle
120	Duet
121	Punk Rock
122	Drum Solo
123	A capella
124	Euro-House
125	Dance Hall

Appendix B GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.
`http://fsf.org/`

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

0. PREAMBLE

   The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

   This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

   We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

   This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

   A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

   A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

   You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

   You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

   If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

   If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled “History”, Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled “History” in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the “History” section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled “Acknowledgements” or “Dedications”, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled “Endorsements”. Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled “Endorsements” or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

   However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

   Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

   Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

   The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See `http://www.gnu.org/copyleft/`.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
  Copyright (C)  year  your name.
  Permission is granted to copy, distribute and/or modify this document
  under the terms of the GNU Free Documentation License, Version 1.3
  or any later version published by the Free Software Foundation;
  with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
  Texts.  A copy of the license is included in the section entitled ``GNU
  Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with. . . Texts." line with this:

```
    with the Invariant Sections being list their titles, with
    the Front-Cover Texts being list, and with the Back-Cover Texts
    being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Concept Index

This is a general index of all issues discussed in this manual.

%

%idest-guile-site-dir . . . 20
%idest-package-site-dir . . . 20
%idest-version-site-dir . . . 20

–

--filter, ----filter option, used with --copy . . . 11

.

.idest.scm . . . 19

8

8-bit encodings . . . 8

A

all, --all option, introduced . . . 8
all, --all option, summary . . . 37

B

backup, --backup option, described . . . 35
backup, --backup option, summary . . . 37
backup-directory, --backup-directory option, described . . . 35
backup-directory, --backup-directory option, summary . . . 37
backup-suffix, --backup-suffix option, described . . . 35
backup-suffix, --backup-suffix option, summary . . . 37
broken 8-bit encodings . . . 8
broken-8bit-charset, --broken-8bit-charset option, introduced . . . 8
broken-8bit-encoding, --broken-8bit-encoding option, summary . . . 37

C

character set . . . 8
charset . . . 8
charset, --charset option, introduced . . . 8
charset, --charset option, summary . . . 37
charset, --charset option, using with --set . . . 9
comment, fully qualified form . . . 8
convert, --convert option, described . . . 15
convert, --convert option, summary . . . 37
copy, --copy option, described . . . 11
copy, --copy option, summary . . . 37

D

default-id-version, --default-id-version option, described . . . 15
default-id-version, --default-id-version option, summary . . . 39
delete, --delete option, described . . . 13
delete, --delete option, summary . . . 37
describe, --describe option, summary . . . 37
description . . . 26, 30
dry-run . . . 33
dry-run, --dry-run option, introduced . . . 33
dry-run, --dry-run option, summary . . . 39
dry-run.scm . . . 33

E

encoding, --encoding option, summary . . . 38
existing backup method . . . 35

F

`filter, --filter` option, in query mode . . . 7
`filter, --filter` option, introduced . . . 5
`filter, --filter` option, summary . . . 38
`filter, --filter` option, used with `--delete` . . . 13
fixup . . . 9
`fixup, --fixup` option, introduced . . . 9
`fixup, --fixup` option, summary . . . 38
`format, --format` option, described . . . 25
frame . . . 1
frame fixup . . . 9
frames . . . 5
fully-qualified frame name . . . 5

G

genre, code . . . 43
Guile . . . 1, 19
Guile startup files . . . 19

H

`help, --help` option, summary . . . 38
help, a format . . . 27, 31

I

`id-version, --id-version` option, described . . . 15
`id-version, --id-version` option, summary . . . 39
ID3 . . . 1
ID3 version 1 . . . 3
ID3 version 1.1 . . . 3
ID3 version 2 . . . 3
ID3v1 . . . 3
ID3v1.1 . . . 3
ID3v2 . . . 3
`idest-init` . . . 26, 30
`idest-main` . . . 21, 30
`idest-readonly` . . . 22
`idest.scm` . . . 19
`info, --info` option, summary . . . 38

L

`latin1, --latin1` option, summary . . . 38
`list-frames, --list-frames` option, introduced . . . 5
`list-frames, --list-frames` option, summary . . . 38
load path, Guile . . . 19
`load-path, --load-path` option, described . . . 20
`load-path, --load-path` option, summary . . . 38

M

mp3 . . . 1

N

`no-init-files, --no-init-files` option, described . . . 19
`no-init-files, --no-init-files` option, summary . . . 38
numbered backup method . . . 35

P

`PAGER` . . . 28
`prepend-load-path, --prepend-load-path` option, summary . . . 38

Q

qualifiers . . . 5
query mode . . . 7
`query, --query` option, introduced . . . 7, 8
`query, --query` option, summary . . . 39

S

Scheme . . . 19
`script, --script` option, described . . . 19
`script, --script` option, summary . . . 39
scripting . . . 19
`set, --set` option, introduced . . . 9
`set, --set` option, summary . . . 39
simple backup method . . . 35
`SIMPLE_BACKUP_SUFFIX` . . . 35

T

tag 1
test mode 33
`trace, --trace` option, summary 39

U

`usage, --usage` option, summary 39

V

`version, --version` option, summary 39
`VERSION_CONTROL` 35

W

`which, --which` option, help batch option 31
`which, --which` option, help format option 27