The Kid3 Handbook

Software development: Urs Fleisch

Contents

List of Tables

Abstract

Kid3 is an application to edit the ID3v1 and ID3v2 tags in MP3 files in an efficient way. Also tags in Ogg/Vorbis, Opus, DSF, FLAC, MPC, APE, MP4/AAC, MP2, Speex, TrueAudio, WavPack, WMA, WAV, AIFF files and tracker modules (MOD, S3M, IT, XM) are supported. It is easy to set tags of multiple files to the same values (e.g. album, artist, year and genre in all files of the same album) and generate the tags from the file name or vice versa.

Synopsis

`kid3` [`--help` | `--author` | `--version` | `--license` | `--desktopfile FILE`] [*`FILE`*...]

`kid3-qt` [`--portable`] [`Qt-options`] [*`FILE`*...]

`kid3-cli` [`--portable`] [`--dbus`] [`-h` | `--help`] [`-c COMMAND1`] [`-c COMMAND2`...] [*`FILE`*...]

Options

--portable
: Store configuration in file `kid3.ini` inside application folder.

FILE
: If `FILE` is the path to a folder, it will be opened. If one or more file paths are given, their common folder is opened and the files are selected.

kid3

--help
: Show help about options.

--author
: Show author information.

--version
: Show version information.

--license
: Show license information.

--desktopfile FILE
: The base file name of the desktop entry for this application.

kid3-qt

Qt-options
: See qt5options(7).

kid3-cli

--dbus
: Activate the D-Bus interface.

-c
: Execute a command. Multiple `-c` options are possible, they are executed in sequence. See the section about kid3-cli for a description of the available commands.

-h|--help
: Show help about options and commands.

Chapter 1

Introduction

Kid3 is an application to edit the ID3v1 and ID3v2 tags in MP3 files in an efficient way. These tags can be edited by most MP3 players, but not in a very comfortable and efficient way. Moreover the tags in Ogg/Vorbis, Opus, DSF, FLAC, MPC, APE, MP4/AAC, MP2, Speex, TrueAudio, WavPack, WMA, WAV, AIFF files and tracker modules (MOD, S3M, IT, XM) are supported too.

Kid3 does not grab nor encode MP3 files, but it is targeted to edit the ID3 tags of all files of an album in an efficient way, i.e. with as few mouse clicks and key strokes as possible. Where most other programs can edit either ID3v1 or ID3v2 tags, Kid3 has full control over both versions, can convert tags between the two formats and has access to all ID3v2 tags. Tags of multiple files can be set to the same value, e.g. the artist, album, year and genre of all files of an album typically have the same values and can be set together. If the information for the tags is contained in the file name, the tags can be automatically set from the file name. It is also possible to set the file name according to the tags found in the file in arbitrary formats.

The editing task is further supported by automatic replacement of characters or substrings, for instance to remove illegal characters from filenames. Automatic control of upper and lower case characters makes it easy to use a consistent naming scheme in all tags.

The tag information for full albums can be taken from gnudb.org, MusicBrainz, Discogs, Amazon or other sources of track lists. The import format is freely configurable by regular expressions.

Please report any problems or feature requests to the author.

Chapter 2

Using Kid3

2.1 Kid3 features

- Edit ID3v1.1 tags
- Edit all ID3v2.3 and ID3v2.4 frames
- Edit tags of multiple files
- Convert between ID3v1 and ID3v2 tags
- Edit MP3, Ogg/Vorbis, Opus, DSF, FLAC, MPC, APE, MP4/AAC, MP2, Speex, TrueAudio, WavPack, WMA, WAV and AIFF tags
- Generate tags from filename
- Generate tags from the contents of tag fields
- Generate filename from tags
- Generate and change folder names from tags
- Generate playlist file
- Automatic case conversion and string translation
- Import from gnudb.org, MusicBrainz, Discogs, Amazon and other data sources
- Export as CSV, HTML, playlist, Kover XML and other formats. Exported CSV files can be imported again.

2.2 Example Usage

This section describes a typical session with Kid3. Let's assume we have a folder containing MP3 files with the tracks from the album "Let's Tag" from the band "One Hit Wonder". The folder is named in the "artist - album" format, in our case `One Hit Wonder - Let's Tag`. The folder contains the tracks in the "track title.mp3" format, which I think is useful because the filenames are short (important when using mobile MP3 players with small displays) and in the correct order when sorted alphabetically (important when using hardware MP3 players which play the tracks in alphabetical order or in the order in which they are burnt on CD and that order is alphabetical when using **mkisofs**). Besides this, the artist and album information is already in the folder name and does not have to be repeated in the filename. But back to our example, the folder listing looks like this:

```
01 Intro.mp3
02 We Only Got This One.mp3
03 Outro.mp3
```

These files have no tags yet and we want to generate them using Kid3. We use **File** → **Open** menu item (or toolbar button) and select one of the files in this folder. All files will be displayed in the file listbox. Lazy as we are, we want to use the information in the folder and file names to generate tags. Therefore we select all files, then click the **To: Tag 1** button in the **File** section. This will set the title, artist, album and track values in all files. To set the year and genre values of all files, we keep all files selected and type in ″2002″ for the **Date** and select ″Pop″ from the **Genre** combobox. To set only these two values, their check boxes are automatically checked and all other check boxes are left unchecked. Now we change the selection by only selecting the first file and we see that all tags contain the correct values. The tags of the other files can be verified too by selecting them one by one. When we are satisfied with the tags, we use **File** → **Save** menu item (or toolbar button). Selecting **File** → **Create Playlist** menu item (or toolbar button) will generate a file `One Hit Wonder - Let's Tag.m3u` in the folder.

Chapter 3

Command Reference

3.1 The GUI Elements

The Kid3 GUI is separated in six sections: At the left are the file and folder listboxes, the right side contains the **File**, **Tag 1**, **Tag 2** and **Tag 3** sections.

To navigate between the different sections using the keyboard, several keyboard shortcuts are supported. In the tag sections, the shortcuts are active while not editing text or when being in the first column.

- **Alt-Left**: Go to previous section (**Command-[** on macOS®)
- **Alt-Right**: Go to next section (**Command-]** on macOS®)
- **Ctrl-Shift-V**: From other tag
- **Ctrl-C**: Copy
- **Ctrl-V**: Paste
- **Shift-Delete**: Remove
- **F2**: Edit
- **Insert**: Add
- **Delete**: Delete

3.1.1 File List

The file list contains the names of all the files in the opened folder which match the selected file name filter (typically `*.mp3 *.ogg *.opus *.dsf *.flac *.mpc *.aac *.m4a *.m4b *.m4p *.mp4 *.mp2 *.spx *.tta *.wv *.wma *.wav *.aiff *.ape`). A single or multiple files can be selected. To select no file, click into the empty area after the listbox entries. The selection determines the files which are affected by the operations which are available by using the buttons described below.

Besides **Name**, also other columns **Size**, **Type**, **Date Modified** with file details can be displayed. Columns can be hidden by unchecking their name in the context menu of the list header. The order of the columns can be changed by drag and drop. The sort order can be toggled by clicking on the column header.

The values of the standard tags can also be displayed and edited in columns of the file list.

At the left of the names an icon can be displayed: a disc to show that the file has been modified or information about which tags are present (V1, V2, V1V2 or NO TAG, no icon is displayed if the file has not been read in yet).

Folders are displayed with a folder icon. If a folder is opened, its files are displayed in a hierarchical tree. By selecting files from subfolders, operations can be executed on files in different folders, which is useful if the music collection is organized with a folder for each artist containing folders for albums of this artist.

Clicking the right mouse button inside the file list opens a context menu with the following commands:

- **Expand all**: Expands all folder trees (only the current tree if the **Shift** key is pressed)
- **Collapse all**: Collapses all folder trees
- **Rename**: Changes the name of a file
- **Move to Trash**: Moves a file to the trash
- **Play**: Plays a file, see Play. If the selected file is a playlist, the files of the playlist will be played.
- **Edit**: Edit a playlist, see Edit Playlist.
- The subsequent entries are user commands, which can be defined in the **User Actions** tab of Configure Kid3. The playback on double click can also be activated there.

  When **Select file on play** is activated, the currently played track is automatically selected in the file list.

3.1.2 Edit Playlist

A playlist can be created empty or containing the tracks of a folder, see Create Playlist. The playlist file created in such a way can be edited by double click or using **Edit** from the file list context menu. A dialog with the entries of the playlist is shown. It is possible to open multiple playlists simultaneously.

New entries can be added by drag and drop from the file list, a file manager or another playlist. If an entry is dragged from another playlist, it will be moved or copied depending on the system. To invoke the other operation, respectively, the **Shift**, **Ctrl** or **Alt** (to copy instead of move on macOS®) key has to be pressed. Reordering entries within the playlist is also possible via drag and drop. Alternatively, entries can be moved using the keyboard shortcuts **Ctrl**-**Shift**-**Up** and **Ctrl**-**Shift**-**Down** (on macOS® **Command** has to be pressed instead of **Ctrl**). An entry can be removed using the **Delete** key.

Please note the following: To drag entries from the file list, they have to be held at the left side (near the icons), the same gesture at the right side will perform a multi selection, such an action is hereby still easily possible.

When a playlist has been modified, the changes can be stored using **Save** or discarded using **Cancel**. When the window is closed, a confirmation prompt is shown if there are unsaved changes.

Tracks selected in a playlist will be automatically selected in the file list, thereby making it possible to edit their tags.

To execute actions on a playlist, its file must be selected in the file list. **Edit** from the context menu will lead to the dialog described in this section, and **Play** will start the media player with the tracks from the playlist. User actions can act on playlists, for example **Export Playlist Folder**, which copies the files from a playlist into a folder.

3.1.3 Folder List

The folder list contains the names of the folders in the opened folder, as well as the current (`.`) and the parent (`..`) folder. It allows one to quickly change the folder without using the **Open** command or drag and drop.

Column visibility, order and sorting can be configured as described in the section about the file list.

3.1.4 File

Shows information about the encoding (MP3, Ogg, Opus, DSF, FLAC, MPC, APE, MP2, MP4, AAC, Speex, TrueAudio, WavPack, WMA, WAV, AIFF), bit rate, sample rate, channels and the length of the file.

The **Name** line edit contains the name of the file (if only a single file is selected). If this name is changed, the file will be renamed when the Save command is used.

The **Format** combo box and line edit contains the format to be used when the filename is generated from the first or the second tag. The filename can contain arbitrary characters, even a folder part separated by a slash from the file name, but that folder must already exist for the renaming to succeed. The following special codes are used to insert tag values into the filename:

- %s %{title} Title (Song)
- %a %{artist} Artist
- %l %{album} Album
- %c %{comment} Comment
- %y %{year} Year
- %t %{track} Track (e.g. 01)
- %t %{track.n} Track with field width n (e.g. 001 for %{track.3})
- %T %{tracknumber} Track (without leading zeros, e.g. 1)
- %g %{genre} Genre
- %{ignore} Ignored when generating tags from the file name

The format codes are not restricted to the examples given above. Any frame name can be used, for instance unified frame names like **`%{albumartist}`**, **`%{discnumber.1}`**, **`%{bpm}`** or format specific names like **`%{popm}`**.

It is possible to prepend and append strings to the replacement for a format code by adding them in double quotes inside the curly braces of a format code. These strings will only be put into the resulting string if the format code yields a nonempty value. For example, if the file name shall both contain the title and the subtitle, one could use **`%{title} [%{subtitle}]`** in the format string. But this would result in a string ending with `[]` if no subtitle frame exists for a file. In order to omit the brackets if no subtitle is present, **`%{title}%{" ["subtitle"]"}`** shall be used instead. This will omit the brackets, the leading space and the subtitle if not subtitle exists.

The list of available formats can be edited in the dialog which appears when clicking the **Filename from tag** button in the **File** tab of the settings.

A second **Format** combo box (with arrow down) is used to generate the tags from the filename. If the format of the filename does not match this pattern, a few other commonly used formats are tried.

Some commonly used filename formats are already available in the combo box, but it is also possible to type in some special format into the line edit.

The list of available formats can be edited in the dialog which appears when clicking the **Tag from filename** button in the **File** tab of the settings.

Internally, a regular expression is built from the format codes. If advanced regular expressions are required, the format to generate the tags from the filenames can be given as a complete regular expression with captures which are preceded by the format codes, e.g. to extract the track numbers without removal of leading zeros, a format like "`/%{track}(\d+) %{title}(.*)`" could be used.

From: Tag 1, **Tag 2**: Sets the filename using the selected format and the first tag or the second tag, respectively.

To: Tag 1, **Tag 2**: The tags are set from the filename. First, the format specified in **Format** is used. If the existing filename does not match this format, the following formats are tried:

- `Artist - Album/Track Song`
- `Album/Track - Artist - Song`
- `/Artist - Album - Track - Song`
- `Album/Artist - Track - Song`
- `Album/Artist - Song`
- `Artist/Album/Track Song`

If a single file is selected, the GUI controls are filled with the values extracted from the filename. If multiple files are selected, the tags of the files are directly set according to the filenames.

3.1.5 Tag 1

The line edit widgets for **Title**, **Artist**, **Album**, **Comment**, **Date**, **Track Number** and **Genre** are used to edit the corresponding value in the first tag of the selected files. The value will be changed when the file selection is altered or before operations like **Save** and **Quit** and when the corresponding check box at the left of the field name is checked. This is useful to change only some values and leave the other values unchanged.

If a single file is selected, all check boxes are checked and the line edit widgets contain the values found in the tags of this file. If a tag is not found in the file, the corresponding empty value is displayed, which is an empty string for the **Title**, **Artist**, **Album** and **Comment** line edits, 0 for the numerical **Date** and **Track Number** edits and an empty selected value for the **Genre** combo box. The values can be changed and if the corresponding check box is checked, they will be set for the selected file after the selection is changed. The file is then marked as modified by a disk symbol in the file listbox but remains unchanged until the **Save** command is used.

If multiple files are selected, only the values which are identical in all selected files are displayed. In all other controls, the empty values as described above are displayed. All check boxes are unchecked to avoid unwanted changes. If a value has to be set for all selected files, it can be edited and the check box has to be set. The values will be set for all selected files when the selection is changed and can be saved using the **Save** command.

The check boxes also control the operation of most commands affecting the tags, such as copy, paste and transfer between tags 1 and 2. To make it easier to use with multiple files where all check boxes are unchecked, these commands behave in the same way when all check boxes are checked and when all check boxes are unchecked.

From Tag 2: The tag 1 fields are set from the corresponding values in tag 2. If a single file is selected, the GUI controls are filled with the values from tag 2. If multiple files are selected, the tags of the files are directly set.

Copy: The copy buffer is filled with the Tag 1 values. Only values with checked check box will be used in subsequent Paste commands.

Paste: Pastes the values from the copy buffer into the GUI controls.

Remove: This will set all GUI controls to their empty values which results in removing all values. The saved file will then contain no tag 1.

3.1.6 Tag 2

The GUI controls function in the same way as described for the **Tag 1** section, but the size of the strings is not limited.

For the tag 2 **Genre** you can also use your own names besides the genres listed in the combo box, just type the name into the line edit.

The tag 2 cannot only contain the same values as the tag 1, the format is built in a flexible way from several frames which are themselves composed of several fields. The tag 2 table shows all the frames which are available in the selected file.

Edit: This will open a window which allows one to edit all fields of the selected frame. If multiple files are selected, the edited fields are applied to all selected files which contain such a frame.

Add: A requester to select the frame type will appear and a frame of the selected type can be edited and added to the file. This works also to add a frame to multiple selected files.

Delete: Deletes the selected frame in the selected files.

Drag album artwork here is shown if the file does not contain embedded cover art. A picture can be added using drag and drop from a browser or file manager and will be displayed here. Picture frames can be edited or added by double clicking on this control.

3.1.7 Tag 3

Some files can have more than two tags, and a third tag section is visible. The following file types can have such a **Tag 3** section:

- MP3 files can have an ID3v1.1 tag, an ID3v2 (2.3.0 or 2.4.0) tag and in the third section an APE tag. Such APE tags are used for replay gain information. In the **Tag 3** section, this information is visible, and the APE tag can be removed with the **Remove** button.
- The RIFF INFO chunk of WAV files is available in the **Tag 3** section because the **Tag 1** section is dedicated to ID3v1.1 tags and handles their restrictions. The **Tag 2** is still used for ID3v2.4.0 tags, which are also supported for WAV files, but RIFF INFO chunks seem to be supported better.
- FLAC files normally use a Vorbis comment for their meta data. However, there are FLAC files which have ID3v1 and ID3v2 tags, which can be found in the **Tag 1** and **Tag 3** sections. ID3 tags in FLAC files are only supported by TagLib, therefore the OggFlacMetadata plugin has to be disabled in the **Plugins** tab of the settings.

The GUI controls work in the same way as in the **Tag 2** section.

3.1.8 Frame List

Kid3 can edit most of the frames for all of the supported file types. Some frames are accessed using unified names, so that they can be exchanged between files with different formats. Frames which are not unified can be accessed as format specific frames.

Unified	ID3v2.3	ID3v2.4	MP4	ASF	Vorbis	RIFF
Title	TIT2	TIT2	©nam	Title	TITLE	INAM
Artist	TPE1	TPE1	©ART	Author	ARTIST	IART
Album	TALB	TALB	©alb	WM/AlbumTitle	ALBUM	IPRD
Comment	COMM	COMM	©cmt	Description	COMMENT	ICMT

Date	TYER	TDRC	©day	WM/Year	DATE	ICRD
Track Number	TRCK	TRCK	trkn	WM/TrackNumber	TRACKNUMBER	IPRT or ITRK
Genre	TCON	TCON	©gen	WM/Genre	GENRE	IGNR
Album Artist	TPE2	TPE2	aART	WM/AlbumArtist	ALBUMARTIST	
Arranger	IPLS	TIPL	ARRANGER	WM/Producer	ARRANGER	IENG
Author	TOLY	TOLY	AUTHOR		AUTHOR	
BPM	TBPM	TBPM	tmpo	WM/BeatsPerMinute	BPM	IBPM
Catalog Number	TXXX:CATALOGNUMBER	TXXX:CATALOGNUMBER			CATALOGNUMBER	
Compilation	TCMP	TCMP	cpil		COMPILATION	
Composer	TCOM	TCOM	©wrt	WM/Composer	COMPOSER	IMUS
Conductor	TPE3	TPE3	CONDUCTOR	WM/Conductor	CONDUCTOR	
Copyright	TCOP	TCOP	cprt	Copyright	COPYRIGHT	ICOP
Description	TIT3	TIT3	desc	WM/SubTitleDescription	DESCRIPTION	
Disc Number	TPOS	TPOS	disk	WM/PartOfSet	DISCNUMBER	
Encoded-by	TENC	TENC	©enc	WM/EncodedBy	ENCODED-BY	ITCH
Encoder Settings	TSSE	TSSE	©too	WM/EncodingSettings	ENCODERSETTINGS	ISFT
Encoding Time		TDEN		WM/EncodingTime	ENCODINGTIME	IDIT
Grouping	GRP1	GRP1	©grp		GROUPING	
Initial Key	TKEY	TKEY		WM/InitialKey	INITIALKEY	
ISRC	TSRC	TSRC	ISRC	WM/ISRC	ISRC	ISRC
Language	TLAN	TLAN	LANGUAGE	WM/Language	LANGUAGE	ILNG
Lyricist	TEXT	TEXT	LYRICIST	WM/Writer	LYRICIST	IWRI
Lyrics	USLT	USLT	©lyr	WM/Lyrics	LYRICS	
Media	TMED	TMED	SOURCEMEDIA		SOURCEMEDIA	IMED
Mood		TMOO		WM/Mood	MOOD	
Original Album	TOAL	TOAL	ORIGINALALBUM	WM/OriginalAlbumTitle	ORIGINALALBUM	
Original Artist	TOPE	TOPE	ORIGINALARTIST	WM/OriginalArtist	ORIGINALARTIST	
Original Date	TORY	TDOR	ORIGINALDATE	WM/OriginalReleaseYear	ORIGINALDATE	
Performer	IPLS	TMCL	PERFORMER		PERFORMER	ISTR

Picture	APIC	APIC	covr	WM/Picture	METADATA_BLOCK_PICTURE	
Publisher	TPUB	TPUB	PUBLISHER	WM/Publisher	PUBLISHER	IPUB
Rating	POPM	POPM	rate	WM/SharedUserRating	RATING	IRTD
Release Country	TXXX:RELEASECOUNTRY	TXXX:RELEASECOUNTRY			RELEASECOUNTRY	ICNT
Release Date		TDRL	RELEASEDATE		RELEASEDATE	
Remixer	TPE4	TPE4	REMIXER	WM/ModifiedBy	REMIXER	IEDT
Sort Album	TSOA	TSOA	soal	WM/AlbumSortOrder	ALBUMSORT	
Sort Album Artist	TSO2	TSO2	soaa		ALBUMARTISTSORT	
Sort Artist	TSOP	TSOP	soar	WM/ArtistSortOrder	ARTISTSORT	
Sort Composer	TSOC	TSOC	soco		COMPOSERSORT	
Sort Name	TSOT	TSOT	sonm	WM/TitleSortOrder	TITLESORT	
Subtitle		TSST	SUBTITLE	WM/SubTitle	SUBTITLE	PRT1
Website	WOAR	WOAR	WEBSITE	WM/AuthorURL	WEBSITE	IBSU
Work	TIT1	TIT1	©wrk	WM/ContentGroupDescription	WORK	
WWW Audio File	WOAF	WOAF		WM/AudioFileURL	WWWAUDIOFILE	
WWW Audio Source	WOAS	WOAS		WM/AudioSourceURL	WWWAUDIOSOURCE	

Table 3.1: Mapping of Unified Frame Types to Various Formats

Remarks concerning the mappings to unified frame names:

- The number of unified frame names is limited by the fact that a sensible mapping shall be possible for all supported file formats. Most tags support frames with arbitrary names; these will be used if no specific frame is available (e.g. the names in uppercase in the column MP4). If no such possibility exists, some frame types may not be supported for the format, e.g. Author and Performer for ASF (WMA).
- The mappings are not chosen arbitrarily, they are geared to the usage of the frames in other applications and devices. Thus the ID3v2 frame ″TPE2 - Band/orchestra/accompaniment″ does not suggest its usage as Album Artist, but this is commonly used. The actual meaning for ID3v2 on the other hand is the reason why this frame is used for the orchestra when importing (e.g. from Discogs), although this may seem a bit strange for other tag formats.

- The mappings are not always bijective. So ID3v2.3 uses an IPLS frame for both Arranger and Performer. When reading back, both frames are displayed as "Arranger".
- The frames Arranger and Performer use a particular format for their contents: "involvement 1|involvee 1|involvement 2|involvee 2|...", for instance "Chorus Master|Ernst Dunshirn|Soprano Vocals|Anna Netrebko". This will create IPLS (ID3v2.3) or TIPL/TMCL (ID3v2.4) frames with a string list in the specified format (the "|" is used as a separator between the strings). Values in this format are also set when importing data from servers which offer this information.
- To explicitly use a specific frame name which conflicts with a unified frame name, prepend an exclamation mark. For example, adding a frame of type "*Media*" to a Vorbis comment will create a frame with name "SOURCEMEDIA" because of the unified type mapping. In order to add a frame with name "MEDIA" and not "SOURCEMEDIA", use "!MEDIA" to force the explicit name.
- If you need a frame which is not in this list, you still have the possibility to enter arbitrary names using the **Add** button. Often used frame names can be added to the **Custom Frames** in the **Tags** configuration, and will then be available in the **Quick Access Frames**.

3.1.9 Synchronized Lyrics and Event Timing Codes

For information synchronized with the audio data, a specific editor is available. These frames are supported for ID3v2.3.0 and ID3v2.4.0 tags. To add such a frame, the specific frame name has to be selected in the list which appears when the **Add** button is clicked - **Synchronized Lyrics** or **Event Timing Codes**, respectively. The editor is the same for both types, for the event timing codes, only a predefined set of events is available whereas for the synchronized lyrics, text has to be entered. In the following, editing synchronized lyrics is explained.

A file having an ID3v2 tag is selected, the lyrics editor is entered using **Add** and selecting **Synchronized Lyrics**. For an existing Synchronized Lyrics frame, it is selected and **Edit** is clicked. The player is automatically opened with the current file so that the file can be played and paused to synchronize lyrics.

The settings at the top of the SYLT editor normally do not have to be changed. If the lyrics contains characters which are not present in the Latin 1 character set, changing the text encoding to UTF16 (or UTF8 for ID3v2.4.0) is advisable. For English lyrics and maximum compatibility, ISO-8859-1 should be used.

The **Lyrics** section has five buttons at the top. **Add** will add a new time event in the table. The time is taken from the position of the player, thus adding an entry while playing the track will add a line for the currently played position. The events in the table have to be chronologically ordered, therefore the row will be inserted accordingly. Entries with an invalid time are treated specially: If the currently selected row has an invalid time, its time stamp will be replaced by the current time instead of adding a new row. If the current time is not invalid, the first row with an invalid time will be used if present. This behavior should facilitate adding time stamps if the lyrics text is already in the table but the time stamps are missing (which is the case when importing unsynchronized lyrics). Note that the invalid time is represented as 00:00.00, i.e. the same as the time at the absolute beginning of the track, which is not invalid. To make a time invalid, press the **Delete** key, or use **Clear** from the context menu. New rows inserted using **Insert row** from the context menu or created when importing unsynchronized lyrics with **From Clipboard** or **Import** also contain invalid time stamps. Rows in the table can be deleted by clicking the **Delete** button or using **Delete rows** from the context menu.

Synchronized lyrics can be imported from a file using **Import**. The expected format is simple or enhanced LRC. If the selected file does not contain a square bracket in the first line, it is supposed to be a simple text file with unsynchronized lyrics. The lines from such a file are then imported having invalid time stamps. The time information can be added using the **Add** button or by manual entry. It is also possible to import lyrics via copy-paste using **From Clipboard**. Synchronized lyrics can be written to LRC files using **Export**. Note that only entries with valid time stamps will be exported and that the entries will be sorted by time. Entries with invalid time won't be

stored in the SYLT frame either, so make sure to include all timing information before leaving the dialog.

The ID3 specification suggests a time stamp for each syllable. However most players only support the granularity of a line or sentence. To support both use cases, Kid3 follows the same conventions as SYLT Editor. Text which is entered into the table is assumed to start a new line unless it starts with a space or a hyphen. Exceptions to this rule are possible by starting a line with an underscore ('_') to force continuation or a hash mark ('#') to force a new line. These escape characters are not stored inside the SYLT frame. Inside the SYLT frame, new lines start with a line feed character (hex 0A) whereas continuations do not. When reading SYLT frames, Kid3 checks if the first entry starts with a line feed. If this is not the case, it is assumed that all entries are new lines and that no syllable continuations are used.

While the track is played, the row associated with the current playing position is highlighted, so that the correctness of the synchronization information can be verified. If an offset has to be added to one or more time stamps, this can be accomplished with the **Add offset** context menu. Negative values can be used to reduce the time. Using **Seek to position** in the context menu, it is possible to set the playing position to the time of the selected row.

Recommended procedure to add new synchronized lyrics

- Get the unsynchronized lyrics, e.g. using **Lyrics** → **Embed Lyrics** from the file list context menu.
- Copy the unsynchronized lyrics to the clipboard, just go to the **Lyrics** row in the frame table and press **Ctrl-C**.
- Add a synchronized lyrics frame (**Add...**, **Synchronized Lyrics**, **OK**), click **From Clipboard**.
- Now all lines from the unsynchronized lyrics are in the table, all time stamps are invalid (0:0:0.00). You can delete empty entries beforehand.
- Start playing the song by clicking the play button ▶ in the play toolbar at the bottom of the main window.
- When the next lyrics line with invalid timestamp comes, click **Add** or press **Alt-A**, the timestamp will be updated.
- Continue like this until all timestamps are set. If you missed something, stop playback and clear the timestamps using the **Delete** key or by selecting them and using **Clear** from the context menu. To restart playback from a given timestamp, use **Seek to position** from the context menu.

3.1.10 Chapters in MP4 Files

MP4 audiobooks typically have a `.m4b` extension and are rather large because they contain all chapters in a single file. To navigate in such files, they can contain chapter marks, which can be edited in Kid3 in a pseudo "Chapters" frame using the same editor which is used for synchronized lyrics. Note, however, that this feature is only available with the **Mp4v2Metadata** plugin, so make sure that it is activated and above the **TaglibMetadata** plugin in the **Plugins** tab of the settings if you have to edit MP4 chapters.

3.2 The File Menu

File → Open... (Ctrl-O)

Opens a folder. All files matching the selected file name filter will be displayed in the file listbox and the chosen file is selected.

File → Open Recent

Opens a recently opened folder.

File → Open Folder... (Ctrl-D)

Opens a folder. All files matching the selected file name filter will be displayed in the file listbox.

File → Reload (F5)

Reload folder. Modified files have to be saved before. Expanded subfolders will be collapsed.

File → Save (Ctrl-S)

Saves all changed files in the folder. The changed files are marked with a disk symbol in the file listbox. If any file names have been changed, those files will be renamed.

File → Revert

Reverts the changes of one or multiple files. If no files are selected in the file listbox, the changes of all files will be reverted, else only the changes of the selected files are reverted.

File → Import...

The Import dialog can be used to import data directly from a freedb.org server, from a MusicBrainz server, from Discogs, Amazon or other sources of album track lists in textual format.

Import from a freedb.org server is possible using a dialog which appears when **From Server: gnudb.org** is selected. The artist and album name to search for can be entered in the two topmost fields, the albums which match the query will be displayed when **Find** is clicked and the results from www.gnudb.org are received. Importing the track data for an album is done by double-clicking the album in the list. The freedb.org server to import from can be selected as well as the CGI path. The imported data is displayed in the preview table of the import dialog. When satisfied with the displayed tracks, they can be imported by terminating the import dialog with **OK**.

If you already have a search result open in the web browser, you can enter the URL into the first search field. The result will then appear in the album list and can be directly imported into Kid3.

A search on the Discogs server can be performed using **Discogs**. As in the **gnudb.org** dialog, you can enter artist and album and then choose from a list of releases. A **Token** can be entered to use the RESTful Discogs API instead of their web interface, which is often changed, thereby breaking the import parser. You have to register for an account on Discogs and then generate a token on their web site (Settings/Developers, Generate new token). Don't forget to **Save Settings** after entering the token in order to use it in subsequent requests too. If **Standard Tags** is marked, the standard information is imported, e.g. artist, album, and title. If **Additional Tags** is marked, more information is imported if available, e.g. performers, arrangers, or the publisher. If **Cover Art** is marked, cover art will be downloaded if available.

A search on Amazon can be performed using **Amazon**. As in the **gnudb.org** dialog, you can enter artist and album and then choose from a list of releases. If **Additional Tags** is marked, more information is imported if available, e.g. performers, arrangers, or the publisher. If **Cover Art** is marked, cover art will be downloaded if available.

You can search in the same way in the release database of MusicBrainz using **From MusicBrainz Release**. The workflow is the same as described for **From gnudb.org**.

Import from a MusicBrainz server is possible using the dialog which appears when **From MusicBrainz Fingerprint** is selected. The Server can be selected as in the freedb import dialog. Below is a table displaying the imported track data. The right column shows the state of the MusicBrainz query, which starts with ″Pending″ when the dialog is opened. Then the fingerprint is looked up and if it does not yield a result, another lookup using the tags in the file is tried. Thus it can be helpful for a successful MusicBrainz query to store known information (e.g. artist and album) in the tags before the import. If a result was

found, the search ends in the state ″Recognized″, otherwise nothing was found or multiple ambiguous results and one of them has to be selected by the user. **OK** and **Apply** use the imported data, **Cancel** closes the dialog. The closing can take a while since the whole MusicBrainz machinery has to be shut down.

For the import of textual data, **From File/Clipboard** opens a subdialog, where several preconfigured import formats are available. The first two, ″CSV unquoted″ and ″CSV quoted″ can be used to import data which was exported by the Export dialog. The CSV data can be edited with a spreadsheet, and shall be written using tabs as delimiters. Import should then be possible using ″CSV quoted″, which is more flexible than ″CSV unquoted″. However, its fields cannot contain any double quotes. If you only export from Kid3 and import later, ″CSV unquoted″ can be used as a simple format for this purpose. Note that there are also ″Export CSV″ and ″Import CSV″ commands in the context menu of the file list, which use scripts to export and import CSV data in a more complete, powerful and flexible way.

The next format, ″freedb HTML text″, can be used to copy information from an HTML page of freedb.org. Search an album in freedb and if the desired information is displayed in the web browser, copy the contents to the clipboard. Then click the **From Clipboard** button and the imported tracks will be displayed in the preview table at the top of the dialog. If you are satisfied with the imported data, terminate the dialog with **OK**, which will insert the data into the tags of the current folder. The destination (**Tag 1**, **Tag 2** or **Tag 1 and Tag 2**) can be selected with a combo box. The files in the current folder should be in the correct track order to get their tags assigned. This is the case if they are numbered.

The next preconfigured import format, ″freedb HTML source″, can be used, if the data is available as an HTML document. Import is possible using the **From File** button, which opens a file selector, or copying its contents from an editor and then importing from clipboard. This format can be useful for offline import, although the HTML document could also be opened in a browser and then be imported in the first format via the clipboard.

More preconfigured formats, e.g. ″Track Title Time″, are available. An empty custom format can be created with **Add** to be set by the user. Two lines below the format name can be set with a regular expression to capture the fields from the import text. The first regular expression will be parsed once per document to gather per-album data such as artist, album, year and genre. The second line is tried to match from the start of the document to the end to get track data, usually number and title. The regular expressions include all the features offered by Qt™, which is most of the what Perl offers. Bracketing constructs ″(..)″ create capture buffers for the fields to import and are preceded by Kid3 specific codes to specify which field to capture. The codes are the same as used for the filename format, besides the codes listed below, any frame name is possible:

- %s %{title} Title (Song)
- %a %{artist} Artist
- %l %{album} Album
- %c %{comment} Comment
- %y %{year} Year
- %t %{track} Track
- %g %{genre} Genre
- %d %{duration} Duration

For example, a track regular expression (second line) to import from an `.m3u` playlist could be ″`%{track}(\d+)\s+%{title}(\S[^\r\n]*)\.mp3[\r\n]`″. All formats can be changed by editing the regular expressions and the name and then clicking **Save Settings**. They will be stored in the `kid3rc` file in the configuration folder. This file can be directly edited to have more import formats or it can be deleted to revert to the default formats. Formats can be deleted using **Remove**.

Accuracy shows an estimation of how good the imported information matches the given tracks. It uses track durations or file names to calculate the level of similarity in percent. **Cover Art** shows the URL of the album cover image which will be downloaded.

To check whether the imported tracks match the current set of files, the duration of the imported tracks can be compared with the duration of the files. This option can be enabled with the check box **Check maximum allowable time difference (sec):** and the maximum tolerated difference in time can be set in seconds. If a mismatch in a length is detected, the length is displayed with a red background in the preview table.

If the files are ordered differently than the imported tracks, their assigned tracks have to be changed. This task can be facilitated using the **Match with** option with the buttons **Length**, **Track**, and **Title**, which will reorder the tracks according to the corresponding field. To correct the assignments manually, a track can be dragged with the left mouse button and the **Ctrl** key hold down, and then dropped at the new location.

When the import dialog is opened, it contains the actual contents of the tags. The tag type (Tag 1, Tag 2, Tag 1 and Tag 2) can be selected using the **Destination** combo box. The button on the right of this combo box can be used to revert the table to the current contents of the tags. The check boxes in the first table column can be used to select the tracks which are imported. This can be useful if a folder contains the tracks of both CDs of a double CD and only the tracks of the second CD have to be imported.

To identify the tracks which are imported, it is possible to display the file names or the full paths to the files using the context menu of the table header. The values in the import table can be edited. The revert-button to the right of the **Destination** combo box can be used to restore the contents of the tags, which can also be useful after changing the **Destination**.

Almost all dialogs feature a **Save Settings** button, which can be used to store the dialog specific settings and the window size persistently.

From Tags leads to a subdialog to set tag frames from the contents of other tag frames. This can be used to simply copy information between tags or extract a part from one frame and insert it in another.

As in the import from file/clipboard dialog, there are freely configurable formats to perform different operations. Already preconfigured are formats to copy the Album value to Album Artist, Composer or Conductor, and to extract the Track Number from Title fields which contain a number. There is also a format to extract a Subtitle from a Title field.

The following example explains how to add a custom format, which sets the information from the Subtitle field also in the Comment field. Create a new format using **Add** button and set a new name, e.g. "Subtitle to Comment". Then enter "**`%{subtitle}`**" in **Source** and "**`%{comment}(.*)`**" for **Extraction** and click **Save Settings**.

The expression in **Source** can contain format codes for arbitrary tag frames, multiple codes can be used to combine the contents from different frames. For each track, a text is generated from its tags using the **Source** format, and the regular expression from **Extraction** is applied to this text to set new values for the tags. Format codes are used before the capturing parentheses to specify the tag frame where the captured text shall be stored. It works in the same way as for the import from file/clipboard.

Import from Tags... is also directly available from the **File** menu. The difference between these two functions is that the import dialog subdialog operates on all files of the current folder whereas the menu function operates on the selected files (which can be in different folders). The menu function supports an additional code "`%{__return}`" to return the extracted value, which can be useful with the CLI and QML interfaces.

File → Import from gnudb.org...

Import from a freedb.org server using gnudb.org album search. This menu item opens the same import dialog as **Import...**, but opens directly the **gnudb.org** dialog.

File → Import from Discogs...

Import from the Discogs server. This menu item opens the same import dialog as **Import...**, but opens directly the **From Discogs** dialog.

File → Import from Amazon...

Import from Amazon. This menu item opens the same import dialog as **Import...**, but opens directly the **From Amazon** dialog.

File → Import from MusicBrainz Release...

Import from the MusicBrainz release database. This menu item opens the same import dialog as **Import...**, but opens directly the **From MusicBrainz Release** dialog.

File → Import from MusicBrainz Fingerprint...

Import from a MusicBrainz server. This menu item opens the same import dialog as **Import...**, but opens directly the **From MusicBrainz Fingerprint** dialog.

File → Import from Tags...

Like From Tags, but the import is applied to the selected files.

File → Automatic Import...

Automatic Import allows one to import information for multiple albums from various web services. If folders are selected in the file list, track data for the selected folders will be imported. If no folder is selected, all folders in the file list will be imported.

The tag type (Tag 1, Tag 2, Tag 1 and Tag 2) can be selected using the **Destination** combo box.
Profiles determine which servers will be contacted to fetch album information. Some profiles are predefined (All, MusicBrainz, Discogs, Cover Art), custom profiles can be added using the **Add** button at the right of the **Profile** combo box.

The table below shows the servers which will be used when importing album information using the selected profile. The import process for an album is finished if all required information has been found, so the order of the rows in the table is important. It can be changed using the **Move Up** and **Move Down** buttons. **Edit** can be used to change an existing entry. The **Server** selection offers the same servers as can be used in the import functions. **Standard Tags**, **Additional Tags**, **Cover Art** determine the information which shall be fetched from the server. Finally, **Accuracy** is the minimum accuracy which must be achieved to accept the imported data. If the accuracy is insufficient, the next server in the list will be tried. The same dialog containing the server properties appears when **Add** is clicked to add a new server entry. Existing entries can be deleted using **Remove**.

To launch an automatic batch import with the selected profile, click **Start**. Details about the running import are displayed at the top of the dialog. The process can be aborted with the **Abort** button.

File → Browse Cover Art...

The Browse Cover Art dialog helps to find album cover art. **Artist/Album** is filled from the tags if possible. **Source** offers a variety of websites with album cover art. The URL with artist and album as parameters can be found beneath the name. URL-encoded values for artist and album can be inserted using "`%u{artist}`" and "`%u{album}`", other values from the tags are possible too, as described in Configure Kid3, **User Actions**. More sources can be entered after the entry "Custom Source" by replacing "Custom Source" with the source's name, pressing **Enter**, then inserting the URL and finally pressing **Save Settings**. The resulting browser command is displayed at the top of the dialog and can be started by clicking **Browse**. The browser, which can be configured in the settings, is started with the selected source. A cover image can then be dragged from the browser into the Kid3 window and will be set in the picture frame of the selected files.

Because not all browsers support drag and drop of images and the pictures on websites often have a URL, in such cases Kid3 will receive the URL and not the picture. If the URL points to a picture, it will be downloaded. However, if the URL refers to some other web resource, it has to be translated to the corresponding picture. Such mappings are defined in the table **URL extraction**. The left column **Match** contains a regular expression which is compared with the URL. If it matches, the captured expressions in parentheses are inserted into the pattern of the right **Picture URL** column (at the positions marked with \1 etc.). The replaced regular expression contains the URL of the picture. By this means cover art can be imported from Amazon, Google Images, etc. using drag and drop. It is also possible to define your own mappings.

File → Export...

The Export Dialog is used to store data from the tags in a file or the clipboard. The editor at the top shows a preview of the data to export. If the export data contain tabulator characters, the export is displayed in a table. The data will be generated from the tags in the current folder according to the configured format.

The format settings are similar as in the Import dialog: The topmost field contains the title (e.g. ″CSV unquoted″), followed by the header, which will be generated at the begin of the file. The track data follows; it is used for every track. Finally, the trailer can be used to generate some finishing text.

The format fields do not contain regular expressions as in the Import dialog, but only output format expressions with special %-expressions, which will be replaced by values from the tags. The whole thing works like the file name format, and the same codes are used plus some additional codes. Not only the codes listed below but all tag frame names can be used.

- %s %{title} Title (Song)
- %a %{artist} Artist
- %l %{album} Album
- %c %{comment} Comment
- %y %{year} Year
- %t %{track} Track (e.g. 01)
- %t %{track.n} Track with field width n (e.g. 001 for %{track.3})
- %T %{tracknumber} Track (without leading zeros, e.g. 1)
- %g %{genre} Genre
- %f %{file} File name
- %p %{filepath} Path
- %{modificationdate} Modification date
- %{creationdate} Creation date
- %u %{url} URL
- %{dirname} Folder name
- %d %{duration} Duration in minutes:seconds
- %D %{seconds} Duration in seconds
- %n %{tracks} Number of tracks of the album
- %e %{extension} File extension
- %O %{tag1} The format of tag 1 (ID3v1.1 or empty if not existing)
- %o %{tag2} The format of tag 2 (ID3v2.3.0, ID3v2.4.0, ID3v2.2.0, ID3v2.2.1, Vorbis, APE, MP4, ASF, or empty if not existing)
- %b %{bitrate} Bit rate in kbit/s
- %v %{vbr} VBR or empty (only for ID3v2.3 with id3lib)
- %r %{samplerate} Sample rate in Hz
- %m %{mode} Channel mode (Stereo or Joint Stereo)
- %h %{channels} Number of channels (1 or 2)
- %k %{codec} Codec (e.g. MPEG 1 Layer 3, MP4, Ogg Vorbis, FLAC, MPC, APE, ASF, AIFF, WAV)

A few formats are predefined. ″CSV unquoted″ separates the fields by tabs. Data in this format can be imported again into Kid3 using the import format with the same name. ″CSV quoted″ additionally encloses the fields by double quotes, which eases the import into spreadsheet applications. However, the fields shall not contain any double quotes when this format is used. ″Extended M3U″ and ″Extended PLS″ generate playlists with extended attributes and absolute path names. ″HTML″ can be used to generate an HTML

page with hyperlinks to the tracks. "Kover XML" creates a file which can be imported by the cover printing program Kover. "Technical Details" provides information about bit rate, sample rate, channels, etc. Finally, "Custom Format" is left empty for definition of a custom format. You can define more formats of your own by adding lines in the file `kid3rc` in the configuration folder. The other formats can be adapted to your needs.

The **Source** of the tags to generate the export data (**Tag 1** or **Tag 2**) can be selected with a combo box. Pushing **To File** or **To Clipboard** stores the data in a file or on the clipboard. **OK** and **Cancel** close the dialog, whereas **OK** accepts the current dialog settings.

File → Create Playlist...

Creates a playlist. The format and contents of the playlist can be set by various options.

The name of the playlist can be the **Same as folder name** or use a **Format** with values from the tags, e.g. "**`%{artist} - %{album}`**" to have the artist and album name in the playlist file name. The format codes are the same as for Export. The list of available formats can be edited in the **Format** section of the **Files** tab in the settings. **Create new empty playlist** will make an empty playlist with the given name. The extension depends on the playlist format.
The location of the generated playlist is determined by the selection of the **Create in** combo box.

Current folder
: The playlist is created in the current folder and contains only files of the current folder. The current folder is the folder where the current file is located. If multiple files are selected, the current file is probably the last selected file.

Every folder
: A playlist is created in every folder which contains listed files, and each playlist contains the files of that folder.

Top-level folder
: Only one playlist is created in the top-level folder (i.e. the folder of the file list) and it contains the listed files of the top-level folder and all of its sub-folders.

The **Format** of the playlist can be **M3U**, **PLS** or **XSPF**.

If **Include only the selected files** is checked, only the selected files will be included in the playlist. If a folder is selected, all of its files are selected. If this check box is not activated, all audio files are included in the playlist.

Sort by file name selects the usual case where the files are ordered by file name. With **Sort by tag field**, it is possible to sort by a format string with values from tag fields. For instance, "**`%{track.3}`**" can be used to sort by track number (the "**`.3`**" is used to get three digits with leading zeros because strings are used for sorting). It is also possible to use multiple fields, e.g. "**`%{genre}%{year}`**" to sort using a string composed of genre and year.

The playlist entries will have relative or absolute file paths depending on whether **Use relative path for files in playlist** or **Use full path for files in playlist** is set.

When **Write only list of files** is set, the playlist will only contain the paths to the files. To generate an extended playlist with additional information, a format string can be set using the **Write info using** control.

File → Quit (Ctrl-Q)

Quits the application.

3.3 The Edit Menu

Edit → Select All (Alt-A)

Selects all files.

Edit → Deselect (Ctrl-Shift-A)

Deselects all files.

Edit → Select All in Folder

Selects all files of the current folder.

Edit → Previous File (Alt-Up)

Selects the previous file.

Edit → Next File (Alt-Down)

Selects the next file.

Edit → Find... (Ctrl-F)

Find strings in the file names and the tags. The **Find** dialog is a subset of the **Replace** dialog, which is described below.

Edit → Replace... (Ctrl-R)

This function opens a dialog to find and replace strings in the file names and the tags. The set of frames where the search is performed can be restricted by deactivating the **Select all** check box and selecting the frames which shall be searched. There are also search options available to search backwards, case sensitively, and to use regular expressions.

Depending on the number of files, the search might take some time, therefore it can be aborted by closing the dialog.

3.4 The Tools Menu

Tools → Apply Filename Format

When **Automatically apply format** is switched off for the filename format in the configuration dialog, this menu item can be used to apply the configured format to the names of the selected files. This can also be used to check whether the file names conform with the configured format by applying the format to all saved files and then checking if any files were changed (and therefore marked with a disk symbol in the file listbox).

Tools → Apply Tag Format

When **Automatically apply format** is switched off for the tag format in the configuration dialog, this menu item can be used to apply the configured format to the tags of the selected files. This can also be used to check whether the tags conform with the configured format by applying the format to all saved files and then checking if any files were changed (and therefore marked with a disk symbol in the file listbox).

Tools → Apply Text Encoding

Sets the **Text encoding** selected in **Settings → Configure Kid3... → Tags section → Tag 2 tab** for all selected files. If UTF8 is selected, UTF16 will be used for ID3v2.3.0 tags because UTF8 is not supported for this format.

Tools → Rename Folder...

This dialog offers the possibility to automatically rename the currently open folder according to the tags in the files. Several formats are preconfigured to include information about artist, album and year in the folder name. It is also possible to set a custom format and **Edit** the list of available formats. The following special codes are used to insert tag values into the folder name:

- %s %{title} Title (Song)
- %a %{artist} Artist
- %l %{album} Album

- %c %{comment} Comment
- %y %{year} Year
- %t %{track} Track (e.g. 01)
- %t %{track.n} Track with field width n (e.g. 001 for %{track.3})
- %T %{tracknumber} Track (without leading zeros, e.g. 1)
- %g %{genre} Genre
- %{dirname} Folder name (e.g. %{year" "}%{dirname} will prepend the year to the current folder name)
- %{max-year} The maximum year value found for this folder, can also be used with other codes than "year"
- %{min-year} The minimum year value found for this folder
- %{unq-year} The unique year value found for this folder or empty if not unique

If a folder separator "/" is found in the format, multiple folders are created. If you want to create a new folder instead of renaming the current folder, in the **Action** combo box select **Create Folder** instead of **Rename Folder**. The **Source** of the tag information can be chosen between **Tag 1 and Tag 2**, **Tag 1** and **Tag 2**. A preview for the rename operation performed on the first file can be seen in the **From** and **To** sections of the dialog.

Multiple folders can be renamed by selecting them.

Tools → Number Tracks...

If the track numbers in the tags are not set or have the wrong values, this function can number the tracks automatically in ascending order. The start number can be set in the dialog. If only part of the tracks have to be numbered, they must be selected.

When **Total number of tracks** is checked, the number of tracks will also be set in the tags.

It is possible to number the tracks over multiple folders. The folders have to be expanded and selected.
If **Reset counter for each folder** is checked, track numbering is restarted with the given number for each folder when multiple folders are selected.

The number tracks dialog can also be used to format existing track numbers without changing the values when the check box left to **Start number** is deactivated. The total number of tracks will be added if the corresponding check box is active, which can be used to set the total for all selected tracks. If only formatting of the existing numbers is desired, this check box has to be deactivated too.

Tools → Filter...

The filter can be used to display only those files which match certain criteria. This is helpful if you want to organize a large collection and only edit those files which are not in the desired scheme. The expression defining which files to display uses the same format codes which are used in the file name format, import and export.

- %s %{title} Title (Song)
- %a %{artist} Artist
- %l %{album} Album
- %c %{comment} Comment
- %y %{year} Year
- %t %{track} Track (e.g. 01)
- %t %{track.n} Track with field width n (e.g. 001 for %{track.3})
- %T %{tracknumber} Track (without leading zeros, e.g. 1)
- %g %{genre} Genre
- %f %{file} File name
- %p %{filepath} Absolute path to file

- %e %{extension} File extension
- %O %{tag1} The format of tag 1 (ID3v1.1 or empty if not existing)
- %o %{tag2} The format of tag 2 (ID3v2.3.0, ID3v2.4.0, ID3v2.2.0, ID3v2.2.1, Vorbis, APE, MP4, ASF, or empty if not existing)
- %b %{bitrate} Bit rate in kbit/s
- %v %{vbr} VBR or empty (only for ID3v2.3 with id3lib)
- %r %{samplerate} Sample rate in Hz
- %m %{mode} Channel mode (Stereo or Joint Stereo)
- %h %{channels} Number of channels (1 or 2)
- %k %{codec} Codec (e.g. MPEG 1 Layer 3, MP4, Ogg Vorbis, FLAC, MPC, APE, ASF, AIFF, WAV)
- %w %{marked} Marked, is 1 if the file is marked (e.g. because of truncation or standard violation), empty otherwise
- %1a %1{artist}, ... Use the prefix 1 to get values of tag 1
- %2a %2{artist}, ... Use the prefix 2 to get values of tag 2

These codes are replaced with the values for the file, and the resulting strings can be compared with the following operations:

- s1 equals s2: true if s1 and s2 are equal.
- s1 contains s2: true if s1 contains s2, i.e. s2 is a substring of s1.
- s matches re: true if s matches the regular expression re.

True expressions are replaced by 1, false by 0. True values are represented by 1, true, on and yes, false values by 0, false, off and no. Boolean operations are not, and, or (in this order of precedence) and can be grouped by parentheses.

Some filter rules are predefined and can serve as examples for your own expressions:

All

When the file list is filtered - this is shown by ″[filtered]″ in the window title - and all files shall be displayed again, the filtering can be reverted using this filter. It uses an empty expression, but a true value would have the same effect.

Filename Tag Mismatch

```
not (%{filepath} contains ″%{artist} – %{album}/%{track} %{title}″)
```

Tests if the file path conforms with the file name format. This rule is automatically adapted if the file name format changes.

No Tag 1

```
%{tag1} equals ″″
```

Displays only files which do not have a tag 1.

No Tag 2

```
%{tag2} equals ″″
```

Displays only files which do not have a tag 2.

ID3v2.3.0 Tag

```
%{tag2} equals ″ID3v2.3.0″
```

Displays only files which have an ID3v2.3.0 tag.

ID3v2.4.0 Tag

```
%{tag2} equals ″ID3v2.4.0″
```

Displays only files which have an ID3v2.4.0 tag.

Tag 1 != Tag 2

```
not (%1{title} equals %2{title} and %1{album} equals %2{alb
um} and %1{artist} equals %2{artist} and %1{comment} equals
%2{comment} and %1{year} equals %2{year} and %1{track} equals
%2{track} and %1{genre} equals %2{genre})
```

Displays files with differences between tag 1 and tag2.

Tag 1 == Tag 2

```
%1{title} equals %2{title} and %1{album} equals %2{album} and
%1{artist} equals %2{artist} and %1{comment} equals %2{comment
} and %1{year} equals %2{year} and %1{track} equals %2{track}
and %1{genre} equals %2{genre}
```

Displays files with identical tag 1 and tag 2.

Incomplete

```
%{title} equals "" or %{artist} equals "" or %{album} equals
"" or %{year} equals "" or %{tracknumber} equals "" or %{genre}
equals ""
```

Displays files with empty values in the standard tags (title, artist, album, date, track number, genre).

No Picture

```
%{picture} equals ""
```

Displays only files which do not have a picture.

Marked

```
not (%{marked} equals "")
```

Displays only files which are marked because they violate the ID3 standard, are truncated or the picture is too large.

Custom Filter

To add your own filter, select this entry. For instance, if you want to have a filter for artists starting with "The", replace "Custom Filter" with the name "The Bands" and press **Enter**. Then insert the following expression into the line edit:

```
%{artist} matches "The.*"
```

Then click **Save Settings**. Click **Apply** to filter the files. All files processed are displayed in the text view, with a "+" for those who match the filter and a "-" for the others. When finished, only the files with an artist starting with "The" are displayed, and the window title is marked with "[filtered]".

Tools → Convert ID3v2.3 to ID3v2.4

If there are any ID3v2.3 tags in the selected files, they will be converted to ID3v2.4 tags. Frames which are not supported by TagLib will be discarded. Only files without unsaved changes will be converted.

Tools → Convert ID3v2.4 to ID3v2.3

If there are any ID3v2.4 tags in the selected files, they will be converted to ID3v2.3 tags. Only files without unsaved changes will be converted.

Tools → Play

This opens a simple toolbar to play audio files. It contains buttons for the basic operations (**Play/Pause**, **Stop playback**, **Previous Track**, **Next Track**, **Close**), sliders for position and volume and a display of the current position. If multiple files are selected, the selected tracks are played, else all files will be played.

The time displayed can be toggled between elapsed and remaining time by clicking on the display.

3.5 The Settings Menu

Settings → Show Toolbar

Toggles displaying of the toolbar.

Settings → Show Statusbar

Toggles displaying of the statusbar, which displays longer actions such as opening or saving a folder.

Settings → Show Picture

Toggles displaying of the album cover art preview picture.

Settings → Auto Hide Tags

Empty tags are automatically hidden if this option is active. The **File**, **Tag 1** and **Tag 2** sections can be manually collapsed and expanded by clicking on the corresponding **-/+** buttons.

Settings → Configure Shortcut keys...

Opens a dialog to assign keyboard shortcuts for most of the program functions. There are even functions without corresponding menu or button available, e.g. next file, previous file, select all.

Settings → Configure Kid3...

Opens the configuration dialog, which consists of pages for tags, files, user actions, and network settings.

Tag specific options can be found on the **Tags** page, which is itself separated into four tabs for **Tag 1**, **Tag 2**, **Tag 3**, and **All Tags**.

If **Mark truncated fields** is checked, truncated ID3v1.1 fields will be marked red. The text fields of ID3v1.1 tags can only have 30 characters, the comment only 28 characters. Also the genre and track numbers are restricted, so that fields can be truncated when imported or transferred from ID3v2. Truncated fields and the file will be marked red, and the mark will be removed after the field has been edited.

With **Text encoding** for **ID3v1** it is possible to set the character set used in ID3v1 tags. This encoding is supposed to be ISO-8859-1, so it is recommended to keep this default value. However, there are tags around with different encoding, so it can be set here and the ID3v1 tags can then be copied to ID3v2 which supports Unicode.

The check box **Use track/total number of tracks format** controls whether the track number field of ID3v2 tags contains simply the track number or additionally the total number of tracks in the folder.

When **Genre as text instead of numeric string** is checked, all ID3v2 genres will be stored as a text string even if there is a corresponding code for ID3v1 genres. If this option is not set, genres for which an ID3v1 code exists are stored as the number of the genre code (in parentheses for ID3v2.3). Thus the genre Metal is stored as "Metal" or "(9)" depending on this option. Genres which are not in the list of ID3v1 genres are always stored as a text string. The purpose of this option is improved compatibility with devices which do not correctly interpret genre codes.

When **WAV files with lowercase id3 chunk** is checked, the RIFF chunk used to store ID3v2 tags in WAV files will be named "id3 " instead of "ID3 ". By default, Kid3 and other applications using TagLib accept both the lowercase and the uppercase variant when reading WAV files, but they use "ID3 " when writing ID3v2 tags to WAV files. As there exist other applications which only accept "id3 " (e.g. JRiver Media Center and foobar2000), this option can be used to create tags which can be read by such applications.

When **Mark standard violations** is checked, ID3v2 fields which violate the standard will be marked red. Details about the violation are shown in a tooltip:

- Must be unique
- New line is forbidden
- Carriage return is forbidden

- Owner must be non-empty
- Must be numeric
- Must be numeric or number/total
- Format is DDMM
- Format is HHMM
- Format is YYYY
- Must begin with a year and a space character
- Must be ISO 8601 date/time
- Must be musical key, 3 characters, A-G, b, #, m, o
- Must have ISO 639-2 language code, 3 lowercase characters
- Must be ISRC code, 12 characters
- Must be list of strings separated by '|'
- Has excess white space

The ID3 standard documents are available online:

- ID3 tag version 2.3.0
- ID3 tag version 2.4.0 - Main Structure
- ID3 tag version 2.4.0 - Native Frames

Text encoding defines the default encoding used for ID3v2 frames and can be set to **ISO-8859-1**, **UTF16**, or **UTF8**. **UTF8** is not valid for ID3v2.3.0 frames; if it is set, **UTF16** will be used instead. For ID3v2.4.0 frames, all three encodings are possible.

Version used for new tags determines whether new ID3v2 tags are created as version 2.3.0 or 2.4.0.

Track number digits is the number of digits in Track Number fields. Leading zeros are used to pad. For instance, with a value of 2 the track number 5 is set as "05".

The combo box **Comment field name** is only relevant for Ogg/Vorbis and FLAC files and sets the name of the field used for comments. Different applications seem to use different names, "COMMENT" for instance is used by XMMS, whereas Amarok uses "DESCRIPTION".

The format of pictures in Ogg/Vorbis files is determined by **Picture field name**, which can be "METADATA_BLOCK_PICTURE" or "COVERART". The first is the official standard and uses the same format as pictures in FLAC tags. "COVERART" is an earlier unofficial way to include pictures in Vorbis comments. It can be used for compatibility with legacy players.

If the **Mark if larger than (bytes)** check box is activated, files containing embedded album cover art exceeding the given size in bytes are marked red. This can be used to find files containing oversized pictures which are not accepted by some applications and players. The default value is 131072 bytes (128 KB).

Custom Genres can be used to define genres which are not available in the standard genre list, e.g. "Gothic Metal". Such custom genres will appear in the **Genre** combo box of **Tag 2**. For ID3v1.1 tags, only the predefined genres can be used.

The list of custom genres can also be used to reduce the number of genres available in the **Genre** combo box to those typically used. If your collection mostly contains music in the genres Metal, Gothic Metal, Ancient and Hard Rock, you can enter those genres and mark **Show only custom genres**. The **Tag 2 Genre** combo box will then only contain those four genres and you will not have to search through the complete genres list for them. In this example, only Metal and Hard Rock will be listed in the tag 1 genres list, because those two custom genres entries are standard genres. If **Show only custom genres** is not active, the custom genres can be found at the end of the genres list.

In **Custom Frames**, up to eight custom frame names can be defined, which can then be used like the unified frames, for example as quick access frames.

Quick Access Frames defines which frame types are always shown in the Tag 2 section. Such frames can then be added without first using the **Add** button. The order of these quick access frames can be changed by dragging and dropping items.

The combo box **Track number field name** is only relevant for RIFF INFO and sets the name of the field used for track numbers. Track numbers are not specified in the original RIFF standard, there are applications which use "ITRK", others use "IPRT".

Tag Format contains options for the format of the tags. When **Automatically apply format** is checked, the format configuration is automatically used while editing text in the line edits. **Validation** enables validators in the controls with track/total and date/time values. The **Case conversion** can be set to **No changes**, **All lowercase**, **All uppercase**, **First letter uppercase** or **All first letters uppercase**. To use locale-aware conversion between lowercase and uppercase characters, a locale can be selected in the combobox below. The string replacement list can be set to arbitrary string mappings. To add a new mapping, select the **From** cell of a row and insert the text to replace, then go to the **To** column and enter the replacement text. When the text to replace starts and ends with a slash ("/"), a regular expression is used. For regular expressions containing capturing groups, occurrences of \1, \2, ... in **To** are replaced with the string captured by the corresponding capturing group. To remove a mapping set the **From** cell to an empty value (e.g. by first typing space and then backspace). Inserting and deleting rows is also possible using a context menu which appears when the right mouse button is clicked. Replacement is only active, if the **String replacement** check box is checked.

The table in **Rating** contains the mapping of star ratings to the effective values stored in the tag. The frames with rating information are listed in the Rating row of the frame list. For these frames, the rating can be set by giving a number of stars out of five stars. Different tag formats and different applications use different values to map the star rating to the value stored in the tag. In order to display the correct number of stars, Kid3 will look up a map in this table. The key to look up the mapping is the frame name, for example "RATING" as used for Vorbis comments or "IRTD" for RIFF INFO. For ID3v2 tags, a combined key is used consisting of the frame ID "POPM" of the Popularimeter frame and its "Email" field, separated by a dot. Therefore, different keys for ID3v2 exist, e.g. "POPM.Windows Media Player 9 Series" for the mapping used by Windows Media Player and Explorer, and simply "POPM" for POPM frames with an empty "Email" field. As multiple entries for "POPM" can exist, their order is important. When Kid3 adds a new Popularimeter frame, it will use the first "POPM" entry to determine the value to be written into the "Email" field. This value will then specify the mapping to be used for star ratings. The first entry is also used if no key was found, it is therefore the default entry.

Besides the **Name** column containing the keys, the table has columns **1** to **5** for the values to be stored when the corresponding number of stars is given. The other way round, the values determine the number of stars which are displayed for the value stored in the frame. For instance, the row in the table below contains the values 1, 64, 128, 196, 255. The thresholds for the number of stars to be displayed lay between these values and are compatible with what the Windows® Explorer uses.

Name	**1**	**2**	**3**	**4**	**5**
POPM	1	64	128	196	255
Range	1-31	32-95	96-159	160-223	224-255

Table 3.2: Entry in Rating Table

On the page **Files** the check box **Load last-opened files** can be marked so that Kid3 will open and select the last selected file when it is started the next time. **Preserve file timestamp** can be checked to preserve the file modification time stamp. **Filename for cover** sets the name which is suggested when an embedded image is exported to a file. With **Text encoding (Export, Playlist)** the encoding used when writing files can be set. The default **System** can be changed for example if playlists have to be used on a different device.

If **Mark changes** is active, changed fields are marked with a light gray label background.

The section **File List** determines which files are displayed in the file list. A **Filter** can be used to restrict the items in this list to files with supported extensions. To explicitly specify which folders to display in the file list or exclude certain folders, the options **Include folders** and **Exclude folders** can be used. They can contain wildcard expressions, for instance `*/Music/*` to include only the `Music` folder, or `*/iTunes/*` to exclude the iTunes folder from the file list. If multiple such expressions have to be used, they can be separated by spaces or semicolons.
The buttons **Filename from tag** and **Tag from filename** in section **Format** open dialogs to edit the formats which are available in the **Format** combo boxes (with arrows up and down), which can be found in the File section of the main window.
The **Playlist** button can be used to edit the file name formats available in the Create Playlist dialog.

Filename Format contains options for the format of the filenames. The same options as in **Tag Format** are available.

Additionally, the **Maximum length** allowed for file names can be set. Most modern file systems have a limit of 255 characters, but if you want to burn the files to CD, you should set the limit to 64. If **Use for playlist and folder names** is checked, the file name format is also used when creating playlists and renaming folders.

The **User Actions** page contains a table with the commands which are available in the context menu of the file list. For critical operations such as deleting files, it is advisable to mark **Confirm** to pop up a confirmation dialog before executing the command. **Output** can be marked to see the output written by console commands (standard output and standard error). **Name** is the name displayed in the context menu. **Command** is the command line to be executed. Arguments can be passed using the following codes:

- %F %{files} File paths (a list if multiple files selected)
- %f %{file} File path to single file
- %uF %{urls} URLs (a list if multiple files selected)
- %uf %{url} URL to single file
- %d %{directory} Folder
- %s %{title} Title (Song)
- %a %{artist} Artist
- %l %{album} Album
- %c %{comment} Comment
- %y %{year} Year
- %t %{track} Track (e.g. 01)
- %t %{track.n} Track with field width n (e.g. 001 for %{track.3})
- %T %{tracknumber} Track (without leading zeros, e.g. 1)
- %g %{genre} Genre
- %b %{browser} Command to start the web browser
- %q %{qmlpath} Base folder of provided QML files

The special code **@separator** can be set as a command to insert a separator into the user actions context menu. Menu items can be put into a submenu by enclosing them with **@beginmenu** and **@endmenu** commands. The name of the submenu is determined by the **Name** column of the **@beginmenu** command.

To execute QML scripts, **@qml** is used as a command name. The path to the QML script is passed as a parameter. The provided scripts can be found in the folder `%{qmlpath}/script/` (on Linux® typically `/usr/share/kid3/qml/script/` , on Windows `qml/script/` inside the installation folder, and on macOS® in the app folder `kid3.app/Contents/Resources/qml/script/`). Custom scripts can be stored in any folder. If the QML code uses GUI components, **@qmlview** shall be used instead of **@qml**. Additional parameters are

passed to the QML script where they will be available via the `getArguments()` function. An overview of some functions and properties which are available in QML can be found in the appendix QML Interface.

The command which will be inserted with %{browser} can be defined in the **Web browser** line edit above. Commands starting with %{browser} can be used to fetch information about the audio files from the web, for instance

```
%{browser} http://lyricwiki.org/%u{artist}:%u{title}
```

will query the lyrics for the current song in LyricWiki. The "u" in %u{artist} and %u{title} is used to URL-encode the artist %{artist} and song %{title} information. It is easy to define your own queries in the same way, e.g. an image search with Google:

```
%{browser} http://images.google.com/images?q=%u{artist}%20%u{album}
```

To add album cover art to tag 2, you can search for images with Google or Amazon using the commands described above. The picture can be added to the tag with drag and drop. You can also add an image with **Add**, then select the Picture frame and import an image file or paste from the clipboard. Picture frames are supported for ID3v2, MP4, FLAC, Ogg and ASF tags.

To add and delete entries in the table, a context menu can be used.

The **Network** page contains only a field to insert the proxy address and optionally the port, separated by a colon. The proxy will be used when importing from an Internet server when the check box is checked.

In the **Plugins** page, available plugins can be enabled or disabled. The plugins are separated into two sections. The **Metadata Plugins & Priority** list contains plugins which support audio file formats. The order of the plugins is important because they are tried from top to bottom. Some formats are supported by multiple plugins, so files will be opened with the first plugin supporting them. The **TaglibMetadata** supports most formats, if it is at the top of the list, it will open most of the files. If you want to use a different plugin for a file format, make sure that it is listed before the **TaglibMetadata** plugin. Details about the metadata plugin and why you may want to use them instead of TagLib are listed below.

- **Id3libMetadata**: Uses id3lib for ID3v1.1 and ID3v2.3 tags in MP3, MP2, AAC files. Supports a few more frame types than TagLib.
- **OggFlacMetadata**: Uses libogg, libvorbis, libvorbisfile for Ogg files, and additionally libFLAC++ and libFLAC for FLAC files. These are the official libraries for these formats.
- **TaglibMetadata**: Uses TagLib which supports a lot of audio file formats. It can be used for all audio files supported by Kid3.
- **Mp4v2Metadata**: mp4v2 was originally used by Kid3 to support M4A files. Can be used in case of problems with the M4A support of TagLib.

The **Available Plugins** section lists the remaining plugins. Their order is not important, but they can be enabled or disabled using the check boxes.

- **AmazonImport**: Used for the **Import from Amazon...** function.
- **DiscogsImport**: Used for the **Import from Discogs...** function.
- **FreedbImport**: Used for the **Import from gnudb.org...** function.
- **MusicBrainzImport**: Used for the **Import from MusicBrainz Release...** function.
- **AcoustidImport**: Used for the **Import from MusicBrainz Fingerprint...** function, which depends on the Chromaprint and libav libraries.

Plugins which are disabled will not be loaded. This can be used to optimize resource usage and startup time. The settings on this page take only effect after a restart of Kid3.

3.6 The Help Menu

Help → **Kid3 Handbook**
Opens this handbook.

Help → **About Kid3**
Displays a short information about Kid3.

Chapter 4

kid3-cli

4.1 Commands

kid3-cli offers a command-line-interface for Kid3. If a folder path is used, the folder is opened. If one or more file paths are given, the common folder is opened and the files are selected. Subsequent commands will then work on these files. Commands are specified using `-c` options. If multiple commands are passed, they are executed in the given order. If files are modified by the commands, they will be saved at the end. If no command options are passed, **kid3-cli** starts in interactive mode. Commands can be entered and will operate on the current selection. The following sections list all available commands.

4.1.1 Help

`help` [*COMMAND-NAME*]

Displays help about the parameters of *COMMAND-NAME* or about all commands if no command name is given.

4.1.2 Timeout

`timeout` [default | off | *TIME*]

Overwrite the default command timeout. The CLI commands abort after a command specific timeout is expired. This timeout is 10 seconds for **ls** and **albumart**, 60 seconds for **autoimport** and **filter**, and 3 seconds for all other commands. If a huge number of files has to be processed, these timeouts may be too restrictive, thus the timeout for all commands can be set to *TIME* ms, switched off altogether or be left at the default values.

4.1.3 Quit application

`exit` [force]

Exit application. If there are modified unsaved files, the *force* parameter is required.

4.1.4 Change folder

`cd` [*FOLDER*]

If no *FOLDER* is given, change to the home folder. If a folder is given, change into the folder. If one or more file paths are given, change to their common folder and select the files.

4.1.5 Print the filename of the current folder

`pwd`
Print the filename of the current working folder.

4.1.6 Folder list

`ls`
List the contents of the current folder. This corresponds to the file list in the Kid3 GUI. Five characters before the file names show the state of the file.

- `>` File is selected.
- `*` File is modified.
- `1` File has a tag 1, otherwise '–' is displayed.
- `2` File has a tag 2, otherwise '–' is displayed.
- `3` File has a tag 3, otherwise '–' is displayed.

```
kid3-cli> ls
  1-- 01 Intro.mp3
> 12- 02 We Only Got This One.mp3
 *1-- 03 Outro.mp3
```

In this example, all files have a tag 1, the second file also has a tag 2 and it is selected. The third file is modified.

4.1.7 Save the changed files

`save`

4.1.8 Select file

`select` [all | none | first | previous | next | *FILE*]

To select all files, enter **`select all`**, to deselect all files, enter **`select none`**. To traverse the files in the current folder start with **`select first`**, then go forward using **`select next`** or backward using **`select previous`**. Specific files can be added to the current selection by giving their file names. Wildcards are possible, so **`select *.mp3`** will select all MP3 files in the current folder.

```
kid3-cli> select first
kid3-cli> ls
> 1-- 01 Intro.mp3
  12- 02 We Only Got This One.mp3
 *1-- 03 Outro.mp3
kid3-cli> select next
kid3-cli> ls
  1-- 01 Intro.mp3
> 12- 02 We Only Got This One.mp3
 *1-- 03 Outro.mp3
kid3-cli> select *.mp3
kid3-cli> ls
> 1-- 01 Intro.mp3
> 12- 02 We Only Got This One.mp3
>*1-- 03 Outro.mp3
```

4.1.9 Select tag

`tag` [*TAG-NUMBERS*]

Many commands have an optional *TAG-NUMBERS* parameter, which specifies whether the command operates on tag 1, 2, or 3. If this parameter is omitted, the default tag numbers are used, which can be set by this command. At startup, it is set to `12` which means that information is read from tag 2 if available, else from tag 1; modifications are done on tag 2. The `TAG-NUMBERS` can be set to **`1`**, **`2`**, or **`3`** to operate only on the corresponding tag. If the parameter is omitted, the current setting is displayed.

4.1.10 Get tag frame

`get` [all | *FRAME-NAME*] [*TAG-NUMBERS*]

This command can be used to read the value of a specific tag frame or get information about all tag frames (if the argument is omitted or `all` is used). Modified frames are marked with a '*'.

```
kid3-cli> get
File: MPEG 1 Layer 3 192 kbps 44100 Hz Joint Stereo
  Name: 01 Intro.mp3
Tag 1: ID3v1.1
  Title          Intro
  Artist         One Hit Wonder
  Album          Let's Tag
  Date           2013
  Track Number   1
  Genre          Pop
kid3-cli> get title
Intro
```

To save the contents of a picture frame to a file, use

```
get picture:'/path/to/folder.jpg'
```

To save synchronized lyrics to an LRC file, use

```
get SYLT:'/path/to/lyrics.lrc'
```

It is possible to get only a specific field from a frame, for example **`get POPM.Email`** for the Email field of a Popularimeter frame. If a file has multiple frames of the same kind, the different frames can be indexed with brackets, for example the first performer from a Vorbis comment can be retrieved using **`get performer[0]`**, the second using **`get performer[1]`**.

The pseudo field name "selected" can be used to check if a frame is selected, for example **`get artist.selected`** will return `1` if the artist frame is selected, else `0`.

The pseudo frame name "ratingstars" can be used to get the value of the "rating" frame as the format specific value corresponding to the number of stars (0 to 5). When using "rating", the internal value is returned.

4.1.11 Set tag frame

`set` *FRAME-NAME FRAME-VALUE* [*TAG-NUMBERS*]

This command sets the value of a specific tag frame. If *FRAME-VALUE* is empty, the frame is deleted.

```
kid3-cli> set remixer 'O.H. Wonder'
```

To set the contents of a picture frame from a file, use

```
set picture:'/path/to/folder.jpg' 'Picture Description'
```

To set synchronized lyrics from an LRC file, use

```
set SYLT:'/path/to/lyrics.lrc' 'Lyrics Description'
```

To set a specific field of a frame, the field name can be given after a dot, e.g. to set the Counter field of a Popularimeter frame, use

```
set POPM.Counter 5
```

An application for field specifications is the case where you want a custom TXXX frame with ″rating″ description instead of a standard Popularimeter frame (this seems to be used by some plugins). You can create such a TXXX rating frame with **kid3-cli**, however, you have to first create a TXXX frame with description ″rating″ and then set the value of this frame to the rating value.

```
kid3-cli> set rating ""
kid3-cli> set TXXX.Description rating
kid3-cli> set rating 5
```

The first command will delete an existing POPM frame, because if such a frame exists, **`set rating 5`** would set the POPM frame and not the TXXX frame. Another possibility would be to use **`set TXXX.Text 5`**, but this would only work if there is no other TXXX frame present.

To set multiple frames of the same kind, an index can be given in brackets, e.g. to set multiple performers in a Vorbis comment, use

```
kid3-cli> set performer[0] 'Liza don Getti (soprano)'
kid3-cli> set performer[1] 'Joe Barr (piano)'
```

To select certain frames before a copy, paste or remove action, the pseudo field name ″selected″ can be used. Normally, all frames are selected, to deselect all, use **`set '*.selected' 0`**, then for example **`set artist.selected 1`** to select the artist frame.

The pseudo frame name ″ratingstars″ can be used to set the value of the ″rating″ frame to the format specific value corresponding to the number of stars (0 to 5). The frame name ″rating″ can be used to set the internal value.

Setting ″ratingstars″ on multiple files having different tag formats will not work because the frame with the value mapped from the star count is created for the first file and then used for all files. So instead of **`kid3-cli -c ″set ratingstars 2″ *`** you should rather use **`for f in *; do kid3-cli -c ″set ratingstars 2″ ″$f″; done`**.

4.1.12 Revert

`revert`

Revert all modifications in the selected files (or all files if no files are selected).

4.1.13 Import from file

`import` *FILE FORMAT-NAME* [*TAG-NUMBERS*]

Tags are imported from the file *FILE* in the format with the name *FORMAT-NAME* (e.g. **`″CSV unquoted″`**, see Import).

If **`tags`** is given for *FILE*, tags are imported from other tags. Instead of *FORMAT-NAME* parameters *SOURCE* and *EXTRACTION* are required, see Import from Tags. To apply the import from tags on the selected files, use **`tagsel`** instead of **`tags`**. This function also supports output of the extracted value by using an *EXTRACTION* with the value **`%{__return}(.+)`**.

4.1.14 Automatic import

`autoimport` [*PROFILE-NAME*] [*TAG-NUMBERS*]

Batch import using profile *PROFILE-NAME* (see Automatic Import, **"All"** is used if omitted).

4.1.15 Download album cover artwork

`albumart` *URL* [all]

Set the album artwork by downloading a picture from *URL*. The rules defined in the Browse Cover Art dialog are used to transform general URLs (e.g. from Amazon) to a picture URL. To set the album cover from a local picture file, use the set command.

```
kid3-cli> albumart
http://www.amazon.com/Versus-World-Amon-Amarth/dp/B000078DOC
```

4.1.16 Export to file

`export` *FILE FORMAT-NAME* [*TAG-NUMBERS*]

Tags are exported to file *FILE* in the format with the name *FORMAT-NAME* (e.g. **"CSV unquoted"**, see Export).

4.1.17 Create playlist

`playlist`

Create playlist in the format set in the configuration, see Create Playlist.

4.1.18 Apply filename format

`filenameformat`

Apply file name format set in the configuration, see Apply Filename Format.

4.1.19 Apply tag format

`tagformat`

Apply tag name format set in the configuration, see Apply Tag Format.

4.1.20 Apply text encoding

`textencoding`

Apply text encoding set in the configuration, see Apply Text Encoding.

4.1.21 Rename folder

`renamedir` [*FORMAT*] [create | rename | dryrun] [*TAG-NUMBERS*]

Rename or create folders from the values in the tags according to a given *FORMAT* (e.g. **%{artist} - %{album}**, see Rename Folder), if no format is given, the format defined in the **Rename folder** dialog is used. The default mode is `rename`; to create folders, `create` must be given explicitly. The rename actions will be performed immediately, to just see what would be done, use the `dryrun` option.

4.1.22 Number tracks

`numbertracks` [*TRACK-NUMBER*] [*TAG-NUMBERS*]

Number the selected tracks starting with *TRACK-NUMBER* (`1` if omitted).

4.1.23 Filter

`filter` [*FILTER-NAME* | *FILTER-FORMAT*]

Filter the files so that only the files matching the *FILTER-FORMAT* are visible. The name of a predefined filter expression (e.g. ˝**Filename Tag Mismatch**˝) can be used instead of a filter expression, see Filter.

```
kid3-cli> filter '%{title} contains "tro"'
Started
  /home/urs/One Hit Wonder - Let's Tag
+ 01 Intro.mp3
- 02 We Only Got This One.mp3
+ 03 Outro.mp3
Finished
kid3-cli> ls
  1-- 01 Intro.mp3
  1-- 03 Outro.mp3
kid3-cli> filter All
Started
  /home/urs/One Hit Wonder - Let's Tag
+ 01 Intro.mp3
+ 02 We Only Got This One.mp3
+ 03 Outro.mp3
Finished
kid3-cli> ls
  1-- 01 Intro.mp3
  12- 02 We Only Got This One.mp3
  1-- 03 Outro.mp3
```

4.1.24 Convert ID3v2.3 to ID3v2.4

`to24`

4.1.25 Convert ID3v2.4 to ID3v2.3

`to23`

4.1.26 Filename from tag

`fromtag` [*FORMAT*] [*TAG-NUMBERS*]

Set the file names of the selected files from values in the tags, for example **`fromtag '%{track} - %{title}' 1`**. If no format is specified, the format set in the GUI is used.

4.1.27 Tag from filename

`totag` [*FORMAT*] [*TAG-NUMBERS*]

Set the tag frames from the file names, for example **`totag '%{albumartist} - %{album}/%{track} %{title}' 2`**. If no format is specified, the format set in the GUI is used. If the format of the filename does not match this pattern, a few other commonly used formats are tried.

4.1.28 Tag to other tag

`syncto` *TAG-NUMBER*

Copy the tag frames from one tag to the other tag, e.g. to set the ID3v2 tag from the ID3v1 tag, use **`syncto 2`**.

4.1.29 Copy

`copy` [*TAG-NUMBER*]

Copy the tag frames of the selected file to the internal copy buffer. They can then be set on another file using the **paste** command.

To copy only a subset of the frames, use the "selected" pseudo field with the **set** command. For example, to copy only the disc number and copyright frames, use

```
set '*.selected' 0
set discnumber.selected 1
set copyright.selected 1
copy
```

4.1.30 Paste

`paste` [*TAG-NUMBER*]

Set tag frames from the contents of the **copy** buffer in the selected files.

4.1.31 Remove

`remove` [*TAG-NUMBER*]

Remove a tag.

It is possible to remove only a subset of the frames by selecting them as described in the **copy** command.

4.1.32 Configure Kid3

`config` [*OPTION*] [*VALUE*]

Query or set a configuration option.

The *OPTION* consists of a group name and a property name separated by a dot. When no *OPTION* is given, all available groups are displayed. If only a group name is given, all available properties of the group are displayed. For a given group and property, the currently configured value is displayed. To change the setting, the new value can be passed as a second argument.

If the value of a setting is a list, all list elements have to be given as arguments. This means that to append an element to an existing list of elements, all existing elements have to be passed followed by the new element. In such a situation, it is easier to use the JSON mode, where the current list can be copied with the new element appended.

4.1.33 Execute program or QML script

`execute` [@qml] *`FILE`* [*`ARGS`*]

Execute a QML script or an executable.

Without `@qml` a program is executed with arguments. When `@qml` is given as the first argument, the following arguments are the QML script and its arguments. For example, the tags of a folder can be exported to the file `export.csv` with the following command.

```
kid3-cli -c "execute @qml
/usr/share/kid3/qml/script/ExportCsv.qml export.csv"
/path/to/folder/
```

Here `export.csv` is the argument for the `ExportCsv.qml` script, whereas `/path/to/folder/` is the `FILE` argument for **kid3-cli**.

4.2 Examples

Set title containing an apostrophe. Commands passed to **kid3-cli** with *`-c`* have to be in quotes if they do not only consist of a single word. If such a command itself has an argument containing spaces, that argument has to be quoted too. In UNIX® shells single or double quotes can be used, but on the Windows Command Prompt, it is important that the outer quoting is done using double quotes and inside these quotes, single quotes are used. If the text inside the single quotes contains a single quote, it has to be escaped using a backslash character, as shown in the following example:

```
kid3-cli -c "set title 'I\'ll be there for you'" /path/to/folder
```

Set album cover in all files of a folder using the batch import function:

```
kid3-cli -c "autoimport 'Cover Art'" /path/to/folder
```

Remove comment frames and apply the tag format in both tags of all MP3 files of a folder:

```
kid3-cli -c "set comment '' 1" -c "set comment '' 2" \
-c "tagformat 1" -c "tagformat 2" /path/to/folder/*.mp3
```

Automatically import tag 2, synchronize to tag 1, set file names from tag 2 and finally create a playlist:

```
kid3-cli -c autoimport -c "syncto 1" -c fromtag -c playlist \
  /path/to/folder/*.mp3
```

For all files with an ID3v2.4.0 tag, convert to ID3v2.3.0 and remove the arranger frame:

```
kid3-cli -c "filter 'ID3v2.4.0 Tag'" -c "select all" -c to23 \
  -c "set arranger ''" /path/to/folder
```

This Python script uses **kid3-cli** to generate iTunes Sound Check iTunNORM frames from replay gain information.

```
#!/usr/bin/env python3
# Generate iTunes Sound Check from ReplayGain.
import os, sys, subprocess
```

```
def rg2sc(dirpath):
  for root, dirs, files in os.walk(dirpath):
    for name in files:
      if name.endswith(('.mp3', '.m4a', '.aiff', '.aif')):
        fn = os.path.join(root, name)
        rg = subprocess.check_output([
          'kid3-cli', '-c', 'get "replaygain_track_gain"',
           fn]).strip()
        if rg.endswith(b' dB'):
          rg = rg[:-3]
        try:
          rg = float(rg)
        except ValueError:
          print('Value %s of %s in not a float' % (rg, fn))
          continue
        sc = (' ' + ('%08X' % int((10 ** (-rg / 10)) * 1000) )) * 10
        subprocess.call([
          'kid3-cli', '-c', 'set iTunNORM "%s"' % sc, fn])

if __name__ == '__main__':
  rg2sc(sys.argv[1])
```

4.3 JSON Format

In order to make it easier to parse results from **kid3-cli**, it is possible to get the output in JSON format. When the request is in JSON format, the response will also be JSON. A compact format of the request will also give a compact representation of the response. If the request contains an ″id″ field, it is assumed to be a JSON-RPC request and the response will contain a ″jsonrpc″ field and the ″id″ of the request. The request format uses the same commands as the standard CLI, the ″method″ field contains the command and the parameters (if any) are given in the ″params″ list. The response contains a ″result″ object, which can also be null if the corresponding **kid3-cli** command does not return a result. In case of an error, an ″error″ object is returned with ″code″ and ″message″ fields as used in JSON-RPC.

```
kid3-cli> {"method":"set","params":["artist","An Artist"]}
{"result":null}
kid3-cli> {"method":"get","params":["artist",2]}
{"result":"An Artist"}
kid3-cli> {"method": "get", "params": ["artist"]}
{
    "result": "An Artist"
}

kid3-cli> {"jsonrpc":"2.0","id":"123","method":"get","params":["artist"]}
{"id":"123","jsonrpc":"2.0","result":"An Artist"}
```

Chapter 5

Credits and License

Kid3

Program written by Urs Fleisch ufleisch@users.sourceforge.net

Appendix A

Installation

A.1 How to obtain Kid3

Kid3 can be found at https://kid3.kde.org.

A.2 Requirements

Kid3 needs Qt. KDE is recommended but not necessary, as Kid3 can also be compiled as a Qt™ application. Kid3 can be compiled for systems where these libraries are available, e.g. for GNU/Linux®, Windows® and macOS®. To tag Ogg/Vorbis files, libogg, libvorbis and libvorbisfile are required, for FLAC files libFLAC++ and libFLAC. id3lib is used for MP3 files. These four formats are also supported by TagLib, which can also handle Opus, MPC, APE, MP2, Speex, TrueAudio, WavPack, WMA, WAV, AIFF files and tracker modules. To import from acoustic fingerprints, Chromaprint and libav are used.

Kid3 is available for most Linux® distributions, Windows® and macOS®. Links can be found on https://kid3.kde.org.

A.3 Compilation and Installation

You can compile Kid3 with or without KDE. Without KDE, Kid3 is a simple Qt™ application and lacks some configuration and session features.

For a KDE version, go into the top folder and type

```
% cmake .
% make
% make install
```

To compile for different versions of Qt™ or KDE, set the corresponding **cmake** options.

If not all libraries are present, Kid3 is built with reduced functionality. So you should take care to have all desired development packages installed. On the other side, **cmake**-options control which libraries are compiled in. The default is **-DWITH_TAGLIB:BOOL=ON -DWITH_MP4V2:BOOL=OFF -DWITH_ID3LIB:BOOL=ON -DWITH_CHROMAPRINT:BOOL=ON -DWITH_VORBIS:BOOL=ON -DWITH_FLAC:BOOL=ON** . These options can be disabled using **OFF**.

To build Kid3 as a Qt™ application without KDE, use the **cmake** option **-DWITH_APPS=Qt**. To build both a KDE and a Qt™ application, set **-DWITH_APPS="Qt;KDE"**.

To use a specific Qt™ installation, set **-DQT_QMAKE_EXECUTABLE=/path/to/qmake**.

Generation of RPM-Packages is supported by the file `kid3.spec`, for Debian® Packages, run **build.sh deb**.

The Qt™ application can also be compiled for Windows® and macOS®. The script `build.sh` can be used to download and build all required libraries and create a Kid3 package.

A.4 Configuration

With KDE, the settings are stored in `.config/kid3rc`, the application state in `.local/share/kid3/kid3staterc`. As a Qt™ application, this file is in `.config/Kid3/Kid3.conf`. On Windows®, the configuration is stored in the registry. on macOS® in a plist file.

The environment variable `KID3_CONFIG_FILE` can be used to set the path of the configuration file.

Appendix B

D-Bus Interface

B.1 D-Bus Examples

On Linux® a D-Bus-interface can be used to control Kid3 by scripts. Scripts can be written in any language with D-Bus-bindings (e.g. in Python) and can be added to the **User Actions** to extend the functionality of Kid3.

The artist in tag 2 of the current file can be set to the value ″One Hit Wonder″ with the following code:

Shell

```
dbus-send --dest=org.kde.kid3 --print-reply=literal \
/Kid3 org.kde.Kid3.setFrame int32:2 string:'Artist' \
string:'One Hit Wonder'
```

or easier with Qt™'s **qdbus** (**qdbusviewer** can be used to explore the interface in a GUI):

```
qdbus org.kde.kid3 /Kid3 setFrame 2 Artist \
'One Hit Wonder'
```

Python

```
import dbus
kid3 = dbus.SessionBus().get_object(
  'org.kde.kid3', '/Kid3')
kid3.setFrame(2, 'Artist', 'One Hit Wonder')
```

Perl

```
use Net::DBus;
$kid3 = Net::DBus->session->get_service(
  "org.kde.kid3")->get_object(
  "/Kid3", "org.kde.Kid3");
$kid3->setFrame(2, "Artist", "One Hit Wonder");
```

B.2 D-Bus API

The D-Bus API is specified in `org.kde.Kid3.xml`. The Kid3 interface has the following methods:

B.2.1 Open file or folder

boolean **openDirectory**(string path);

path
: path to file or folder

Returns true if OK.

B.2.2 Unload the tags of all files which are not modified or selected

unloadAllTags(void);

B.2.3 Save all modified files

boolean **save**(void);
Returns true if OK.

B.2.4 Get a detailed error message provided by some methods

string **getErrorMessage**(void);
Returns detailed error message.

B.2.5 Revert changes in the selected files

revert(void);

B.2.6 Start an automatic batch import

boolean **batchImport**(int32 tagMask, string profileName);

tagMask
: tag mask (bit 0 for tag 1, bit 1 for tag 2)

profileName
: name of batch import profile to use

B.2.7 Import tags from a file

boolean **importFromFile**(int32 tagMask, string path, int32 fmtIdx);

tagMask
: tag bit (1 for tag 1, 2 for tag 2)

path
: path of file

fmtIdx
: index of format

Returns true if OK.

B.2.8 Import tags from other tags

importFromTags(int32 tagMask, string source, string extraction);

tagMask
: tag bit (1 for tag 1, 2 for tag 2)

source
: format to get source text from tags

extraction
: regular expression with frame names and captures to extract from source text

B.2.9 Import tags from other tags on selected files

array **importFromTagsToSelection**(int32 tagMask, string source, string extraction);

tagMask
: tag bit (1 for tag 1, 2 for tag 2)

source
: format to get source text from tags

extraction
: regular expression with frame names and captures to extract from source text

returnValues
: extracted value for "%{__return}(.+)"

B.2.10 Download album cover art

downloadAlbumArt(string url, boolean allFilesInDir);

url
: URL of picture file or album art resource

allFilesInDir
: true to add the image to all files in the folder

B.2.11 Export tags to a file

boolean **exportToFile**(int32 tagMask, string path, int32 fmtIdx);

tagMask
: tag bit (1 for tag 1, 2 for tag 2)

path
: path of file

fmtIdx
: index of format

Returns true if OK.

B.2.12 Create a playlist

boolean **createPlaylist**(void);
Returns true if OK.

B.2.13 Get items of a playlist

array **getPlaylistItems**(string path);

path
: path to playlist file

Returns list of absolute paths to playlist items.

B.2.14 Set items of a playlist

boolean **setPlaylistItems**(string path, array items);

path
: path to playlist file

items
: list of absolute paths to playlist items

Returns true if OK, false if not all items were found and added or saving failed.

B.2.15 Quit the application

quit(void);

B.2.16 Select all files

selectAll(void);

B.2.17 Deselect all files

deselectAll(void);

B.2.18 Set the first file as the current file

boolean **firstFile**(void);
Returns true if there is a first file.

B.2.19 Set the previous file as the current file

boolean **previousFile**(void);
Returns true if there is a previous file.

B.2.20 Set the next file as the current file

boolean **nextFile**(void);
Returns true if there is a next file.

B.2.21 Select the first file

boolean **selectFirstFile**(void);
Returns true if there is a first file.

B.2.22 Select the previous file

boolean **selectPreviousFile**(void);
Returns true if there is a previous file.

B.2.23 Select the next file

boolean **selectNextFile**(void);
Returns true if there is a next file.

B.2.24 Select the current file

boolean **selectCurrentFile**(void);
Returns true if there is a current file.

B.2.25 Expand or collapse the current file item if it is a folder

boolean **expandDirectory**(void);
A file list item is a folder if getFileName() returns a name with '/' as the last character.
Returns true if current file item is a folder.

B.2.26 Apply the file name format

applyFilenameFormat(void);

B.2.27 Apply the tag format

applyTagFormat(void);

B.2.28 Apply text encoding

applyTextEncoding(void);

B.2.29 Set the folder name from the tags

boolean **setDirNameFromTag**(int32 tagMask, string format, boolean create);

tagMask
: tag mask (bit 0 for tag 1, bit 1 for tag 2)

format
: folder name format

create
: true to create, false to rename

Returns true if OK, else the error message is available using getErrorMessage().

B.2.30 Set subsequent track numbers in the selected files

numberTracks(int32 tagMask, int32 firstTrackNr);

tagMask
: tag mask (bit 0 for tag 1, bit 1 for tag 2)

firstTrackNr
: number to use for first file

B.2.31 Filter the files

filter(string expression);

expression
: filter expression

B.2.32 Convert ID3v2.3 tags to ID3v2.4

convertToId3v24(void);

B.2.33 Convert ID3v2.4 tags to ID3v2.3

convertToId3v23(void);

Returns true if OK.

B.2.34 Get path of folder

string **getDirectoryName**(void);

Returns absolute path of folder.

B.2.35 Get name of current file

string **getFileName**(void);

Returns true absolute file name, ends with "/" if it is a folder.

B.2.36 Set name of selected file

setFileName(string name);

name
: file name

The file will be renamed when the folder is saved.

B.2.37 Set format to use when setting the filename from the tags

setFileNameFormat(string format);

format
: file name format

B.2.38 Set the file names of the selected files from the tags

setFileNameFromTag(int32 tagMask);

tagMask
: tag bit (1 for tag 1, 2 for tag 2)

B.2.39 Get value of frame

string **getFrame**(int32 tagMask, string name);

tagMask
: tag bit (1 for tag 1, 2 for tag 2)

name
: name of frame (e.g. "artist")

To get binary data like a picture, the name of a file to write can be added after the *name*, e.g. "Picture:/path/to/file". In the same way, synchronized lyrics can be exported, e.g. "SYLT:/path/to/file".

Returns value of frame.

B.2.40 Set value of frame

boolean **setFrame**(int32 tagMask, string name, string value);

tagMask
tag bit (1 for tag 1, 2 for tag 2)

name
name of frame (e.g. ″artist″)

value
value of frame

For tag 2 (*tagMask* 2), if no frame with *name* exists, a new frame is added, if *value* is empty, the frame is deleted. To add binary data like a picture, a file can be added after the *name*, e.g. ″Picture:/path/to/file″. ″SYLT:/path/to/file″ can be used to import synchronized lyrics.

Returns true if OK.

B.2.41 Get all frames of a tag

array of string **getTag**(int32 tagMask);

tagMask
tag bit (1 for tag 1, 2 for tag 2)

Returns list with alternating frame names and values.

B.2.42 Get technical information about file

array of string **getInformation**(void);

Properties are Format, Bitrate, Samplerate, Channels, Duration, Channel Mode, VBR, Tag 1, Tag 2. Properties which are not available are omitted.

Returns list with alternating property names and values.

B.2.43 Set tag from file name

setTagFromFileName(int32 tagMask);

tagMask
tag bit (1 for tag 1, 2 for tag 2)

B.2.44 Set tag from other tag

setTagFromOtherTag(int32 tagMask);

tagMask
tag bit (1 for tag 1, 2 for tag 2)

B.2.45 Copy tag

copyTag(int32 tagMask);

tagMask
tag bit (1 for tag 1, 2 for tag 2)

B.2.46 Paste tag

pasteTag(int32 tagMask);

tagMask
tag bit (1 for tag 1, 2 for tag 2)

B.2.47 Remove tag

removeTag(int32 tagMask);

tagMask
tag bit (1 for tag 1, 2 for tag 2)

B.2.48 Reparse the configuration

reparseConfiguration(void);

Automated configuration changes are possible by modifying the configuration file and then reparsing the configuration.

B.2.49 Plays the selected files

playAudio(void);

Appendix C

QML Interface

C.1 QML Examples

QML scripts can be invoked via the context menu of the file list and can be set in the tab User Actions of the settings dialog. The scripts which are set there can be used as examples to program custom scripts. QML uses JavaScript, here is the obligatory "Hello World":

```
import Kid3 1.0

Kid3Script {
  onRun: {
    console.log("Hello world, folder is", app.dirName)
    Qt.quit()
  }
}
```

If this script is saved as `/path/to/Example.qml`, the user command can be defined as **@qml /path/to/Example.qml** with name **QML Test** and **Output** checked. It can then be started using the **QML Test** item in the file list context menu, and the output will be visible in the window.

Unfortunately, starting the QML scripts using the **qml** (e.g. **qml -apptype widget -I /usr/lib/kid3/plugins/imports /path/to/Example.qml**) is broken in recent versions of Qt. But **kid3-cli** offers an alternative way to run a QML script from the command line using its **execute** command.

```
kid3-cli -c "execute @qml /path/to/Example.qml"
```

To list the titles in the tags 2 of all files in the current folder, the following script could be used:

```
import Kid3 1.0

Kid3Script {
  onRun: {
    app.firstFile()
    do {
      if (app.selectionInfo.tag(Frame.Tag_2).tagFormat) {
        console.log(app.getFrame(tagv2, "title"))
      }
    } while (app.nextFile())
  }
}
```

If the folder contains many files, such a script might block the user interface for some time. For longer operations, it should therefore have a break from time to time. The alternative implementation below has the work for a single file moved out into a function. This function invokes itself with a timeout of 1 ms at the end, given that more files have to be processed. This will ensure that the GUI remains responsive while the script is running.

```
import Kid3 1.0

Kid3Script {
  onRun: {
    function doWork() {
      if (app.selectionInfo.tag(Frame.Tag_2).tagFormat) {
        console.log(app.getFrame(tagv2, "title"))
      }
      if (!app.nextFile()) {
        Qt.quit()
      } else {
        setTimeout(doWork, 1)
      }
    }

    app.firstFile()
    doWork()
  }
}
```

When using **app.firstFile()** with **app.nextFile()**, all files of the current folder will be processed. If only the selected files shall be affected, use **firstFile()** and **nextFile()** instead, these are convenience functions of the `Kid3Script` component. The following example is a script which copies only the disc number and copyright frames of the selected file.

```
import Kid3 1.1

Kid3Script {
  onRun: {
    function doWork() {
      if (app.selectionInfo.tag(Frame.Tag_2).tagFormat) {
        app.setFrame(tagv2, "*.selected", false)
        app.setFrame(tagv2, "discnumber.selected", true)
        app.setFrame(tagv2, "copyright.selected", true)
        app.copyTags(tagv2)
      }
      if (!nextFile()) {
        Qt.quit()
      } else {
        setTimeout(doWork, 1)
      }
    }

    firstFile()
    doWork()
  }
}
```

More example scripts come with Kid3 and are already registered as user commands.

- **ReplayGain to SoundCheck** (`ReplayGain2SoundCheck.qml`): Create iTunNORM SoundCheck information from replay gain frames.

- **Resize Album Art** (`ResizeAlbumArt.qml`): Resize embedded cover art images which are larger than 500x500 pixels.
- **Extract Album Art** (`ExtractAlbumArt.qml`): Extract all embedded cover art pictures avoiding duplicates.
- **Embed Album Art** (`EmbedAlbumArt.qml`): Embed cover art found in image files into audio files in the same folder.
- **Embed Lyrics** (`EmbedLyrics.qml`): Fetch unsynchronized lyrics from web service.
- **Text Encoding ID3v1** (`ShowTextEncodingV1.qml`): Helps to find the encoding of ID3v1 tags by showing the tags of the current file in all available character encodings.
- **ID3v1 to ASCII** (`Tag1ToAscii.qml`): Transliterate extended latin characters in the ID3v1 tag to ASCII.
- **English Title Case** (`TitleCase.qml`): Formats text in the tags to English title case.
- **Rewrite Tags** (`RewriteTags.qml`): Rewrite all tags in the selected files.
- **Export CSV** (`ExportCsv.qml`): Export recursively all tags of all files to a CSV file.
- **Import CSV** (`ImportCsv.qml`): Import recursively all tags of all files from a CSV file.
- **Export JSON** (`ExportJson.qml`): Export recursively all tags of all files to a JSON file.
- **Import JSON** (`ImportJson.qml`): Import recursively all tags of all files from a JSON file.
- **Export Playlist Folder** (`ExportPlaylist.qml`): Copy all files from a playlist into a folder and rename them according to their position.
- **QML Console** (`QmlConsole.qml`): Simple console to play with Kid3's QML API.

C.2 QML API

The API can be easily explored using the QML Console, which is available as an example script with a user interface.

C.2.1 Kid3Script

`Kid3Script` is a regular QML component located inside the plugin folder. You could use another QML component just as well. Using `Kid3Script` makes it easy to start the script function using the `onRun` signal handler. Moreover it offers some functions:

```
onRun: Signal handler which is invoked when the script is started
tagv1, tagv2, tagv2v1: Constants for tag parameters
script: Access to scripting functions
configs: Access to configuration objects
getArguments(): List of script arguments
isStandalone(): true if the script was not started from within Kid3
setTimeout(callback, delay): Starts callback after delay ms
firstFile(): To first selected file
nextFile(): To next selected file
```

C.2.2 Scripting Functions

As JavaScript and therefore QML too has only a limited set of functions for scripting, the `script` object has some additional methods, for instance:

```
script.properties(obj): String with Qt properties
script.writeFile(filePath, data): Write data to file, true if OK
script.readFile(filePath): Read data from file
script.removeFile(filePath): Delete file, true if OK
script.fileExists(filePath): true if file exists
script.fileIsWritable(filePath): true if file is writable
script.getFilePermissions(filePath): Get file permission mode bits
script.setFilePermissions(filePath, modeBits): Set file permission mode bits
script.classifyFile(filePath): Get class of file (folder "/", symlink "@", exe "*",
  file " ")
script.renameFile(oldName, newName): Rename file, true if OK
script.copyFile(source, dest): Copy file, true if OK
script.makeDir(path): Create folder, true if OK
script.removeDir(path): Remove folder, true if OK
script.tempPath(): Path to temporary folder
script.musicPath(): Path to music folder
script.listDir(path, [nameFilters], [classify]): List folder entries
script.system(program, [args], [msecs]): Synchronously start a system command,
  [exit code, standard output, standard error] if not timeout
script.systemAsync(program, [args], [callback]): Asynchronously start a system
command, callback will be called with [exit code, standard output, standard
error]
script.getEnv(varName): Get value of environment variable
script.setEnv(varName, value): Set value of environment variable
script.getQtVersion(): Qt version string, e.g. "5.4.1"
script.getDataMd5(data): Get hex string of the MD5 hash of data
script.getDataSize(data): Get size of byte array
script.dataToImage(data, [format]): Create an image from data bytes
script.dataFromImage(img, [format]): Get data bytes from image
script.loadImage(filePath): Load an image from a file
script.saveImage(img, filePath, [format]): Save an image to a file, true if OK
script.imageProperties(img): Get properties of an image, map containing
  "width", "height", "depth" and "colorCount", empty if invalid image
script.scaleImage(img, width, [height]): Scale an image, returns scaled image
```

C.2.3 Application Context

Using QML, a large part of the Kid3 functions are accessible. The API is similar to the one used for D-Bus. For details, refer to the respective notes.

```
app.openDirectory(path): Open folder
app.unloadAllTags(): Unload all tags
app.saveDirectory(): Save folder
app.revertFileModifications(): Revert
app.importTags(tag, path, fmtIdx): Import file
app.importFromTags(tag, source, extraction): Import from tags
```

```
app.importFromTagsToSelection(tag, source, extraction): Import from tags of ↩
    selected files
app.downloadImage(url, allFilesInDir): Download image
app.exportTags(tag, path, fmtIdx): Export file
app.writePlaylist(): Write playlist
app.getPlaylistItems(path): Get items of a playlist
app.setPlaylistItems(path, items): Set items of a playlist
app.selectAllFiles(): Select all
app.deselectAllFiles(): Deselect
app.firstFile([select], [onlyTaggedFiles]): To first file
app.nextFile([select], [onlyTaggedFiles]): To next file
app.previousFile([select], [onlyTaggedFiles]): To previous file
app.selectCurrentFile([select]): Select current file
app.selectFile(path, [select]): Select a specific file
app.getSelectedFilePaths([onlyTaggedFiles]): Get paths of selected files
app.requestExpandFileList(): Expand all
app.applyFilenameFormat(): Apply filename format
app.applyTagFormat(): Apply tag format
app.applyTextEncoding(): Apply text encoding
app.numberTracks(nr, total, tag, [options]): Number tracks
app.applyFilter(expr): Filter
app.convertToId3v23(): Convert ID3v2.4.0 to ID3v2.3.0
app.convertToId3v24(): Convert ID3v2.3.0 to ID3v2.4.0
app.getFilenameFromTags(tag): Filename from tags
app.getTagsFromFilename(tag): Filename to tags
app.getAllFrames(tag): Get object with all frames
app.getFrame(tag, name): Get frame
app.setFrame(tag, name, value): Set frame
app.getPictureData(): Get data from picture frame
app.setPictureData(data): Set data in picture frame
app.copyToOtherTag(tag): Tags to other tags
app.copyTags(tag): Copy
app.pasteTags(tag): Paste
app.removeTags(tag): Remove
app.playAudio(): Play
app.readConfig(): Read configuration
app.applyChangedConfiguration(): Apply configuration
app.dirName: Folder name
app.selectionInfo.fileName: File name
app.selectionInfo.filePath: Absolute file path
app.selectionInfo.detailInfo: Format details
app.selectionInfo.tag(Frame.Tag_1).tagFormat: Tag 1 format
app.selectionInfo.tag(Frame.Tag_2).tagFormat: Tag 2 format
app.selectionInfo.formatString(tag, format): Substitute codes in format  ↩
    string
app.selectFileName(caption, dir, filter, saveFile): Open file dialog to
select a file
app.selectDirName(caption, dir): Open file dialog to select a folder
```

For asynchronous operations, callbacks can be connected to signals.

```
function automaticImport(profile) {
  function onAutomaticImportFinished() {
    app.batchImporter.finished.disconnect(onAutomaticImportFinished)
  }
  app.batchImporter.finished.connect(onAutomaticImportFinished)
  app.batchImport(profile, tagv2)
}
```

```
function renameDirectory(format) {
  function onRenameActionsScheduled() {
    app.renameActionsScheduled.disconnect(onRenameActionsScheduled)
    app.performRenameActions()
  }
  app.renameActionsScheduled.connect(onRenameActionsScheduled)
  app.renameDirectory(tagv2v1, format, false)
}
```

C.2.4 Configuration Objects

The different configuration sections are accessible via methods of `configs`. Their properties can be listed in the QML console.

```
script.properties(configs.networkConfig())
```

Properties can be set:

```
configs.networkConfig().useProxy = false
```

```
configs.batchImportConfig()
configs.exportConfig()
configs.fileConfig()
configs.filenameFormatConfig()
configs.filterConfig()
configs.findReplaceConfig()
configs.guiConfig()
configs.importConfig()
configs.mainWindowConfig()
configs.networkConfig()
configs.numberTracksConfig()
configs.playlistConfig()
configs.renDirConfig()
configs.tagConfig()
configs.tagFormatConfig()
configs.userActionsConfig()
```