Dokumentation zur Unitsammlung

Mp3FileUtils

Version 0.5b
August 2009

Inhaltsverzeichnis

---

www.gausi.de
mail@gausi.de

Einleitung

Die neue Version dieser Unit ist etwas mächtiger geworden, und nicht Alles wird direkt aus den Beispielprojekten klar. Ich habe mich daher entschlossen, eine etwas umfangreichere Dokumentation zu verfassen. Sie ist keineswegs vollständig, aber bietet einen guten Überblick über die Möglichkeiten und über die Verwendung der bereitgestellten Klassen und deren Methoden.

Das Dokument ist in mehrere Kapitel unterteilt.

- Im ersten Kapitel geht es kurz um die Mpeg-Informationen, die in *jeder* Mp3-Datei zu finden sind. Diese sind ausnahmslos read-only, können also nicht verändert werden.
- Im zweiten Teil betrachten wir den ID3-Tag in seiner einfachen Version, 1.0 bzw. 1.1.
- In den weiteren Kapiteln betrachten wir den ID3-Tag in den neuen, komplizierteren Versionen 2.2 bis 2.4. Das Lesen und Schreiben dieses Tags mit Mp3FileUtils lässt sich grob in drei "Level" einteilen, die unterschiedliche Vorkenntnisse über den Aufbau des ID3v2-Tags seitens des Programmierers erfordern.
- Im letzten Kapitel werden wir etwas auf Unicode eingehen.

Ich setze dabei voraus, dass grundlegende Kenntnisse über ID3-Tags vorhanden sind. D.h. Es ist bekannt, was ID3-Tags eigentlich sind, und worin der fundamentale Unterschied zwischen Version 1 und 2 besteht. Sollte das nicht der Fall sein, empfehle ich einen Blick auf www.id3.org.

Kapitel 1. Mpeg-Informationen

Jede Mp3-Datei besitzt Eigenschaften wie "Bitrate" oder "Spieldauer", die sich teils mehr, teils weniger tief in der Datei verstecken. Die Klasse `TMpegInfo` liefert diese Informationen. Zur Benutzung ist eigentlich nichts zu sagen. Die Informationen werden mit LoadFromStream oder LoadFromFile geladen, und über die einzelnen Properties können dann einzelne Daten abgefragt werden. Die Demo "ID3v1 und Mpeg" sollte alles weitere klären.

Folgende Properties werden bereitgestellt:

```
property Valid: boolean;
property FirstHeaderPosition: int64;
property Filesize: int64;

property Bitrate: word;
property Vbr: boolean;
property Samplerate: integer;
property Channelmode: byte;
property Frames: Integer;
property Dauer: Longint;
property Version: integer;
property Layer: integer;

property Emphasis: byte;
property Protection: boolean;
property Extension: byte;
property Copyright: boolean;
property Original: boolean;
```

Dabei bedeuten

- Valid: True, falls die MP3-Datei als solche erkannt wurde. Andernfalls ist die Datei mit sehr hoher Wahrscheinlichkeit korrupt.
- FirstHeaderPosition: Die Position des ersten Mpeg-Frames in der Datei.
- Bitrate, vbr: Die Bitrate der Datei. Vbr gibt dabei an, ob die Bitrate variabel ist - in dem Fall gibt Bitrate einen Durchschnittswert an.
- Samplerate: Dürfte fast immer bei 44.100 liegen.
- Channelmode: Stereo, Mono etc. Zum Anzeigen sollte man das Konstanten-Array benutzen:

```
Label1.Caption := channel_modes[MpegInfo.Channelmode];
```

- Frames, Dauer: Die Anzahl der Mpeg-Frames in der Datei und die Spieldauer der Datei in Sekunden.
- Version, Layer: Version und Layer der Mpeg-Daten. Mp3-Dateien sind typischerweise Version 1, Layer 3
- Die restlichen vier Werte sind diverse weitere Werte, die einen in der Regel nicht wirklich interessieren müssen.

Hervorheben möchte ich hier außerdem, dass es äußerst sinnvoll ist, vor dem Auslesen der Mpeg-Informationen auf einen ID3Tag in der Version 2 zu testen. Das beschleunigt den Vorgang erheblich.

Grund dafür ist die Tatsache, dass bei der Bestimmung der Mpeg-Daten nach einem Mpeg-Header gesucht werden muss. Wenn man vorher ermittelt, ob die Datei mit einem ID3v2-Tag beginnt und erst dahinter mit der Suche nach einem solchen Header anfängt, dann geht das schneller als den gesamten ID3v2-Tag erfolglos danach zu durchforsten.

Also zum Beispiel so:

```
if Opendialog1.Execute then
begin
  stream := TFileStream.Create(Opendialog1.Filename,
                                        fmOpenRead or fmShareDenyWrite);
  // ID3-Tag auslesen
  Id3v2Tag.ReadFromStream(stream);

  // an das Ende des Tags springen
  if Not Id3v2Tag.exists then
    stream.Seek(0, sobeginning)
  else
    stream.Seek(Id3v2Tag.size, soFromBeginning);

  // Mpeg-Daten auslesen
  MpegInfo.LoadFromStream(stream);
  stream.free;

  // Infos anzeigen
  LblBitrate.Caption := inttostr(MpegInfo.bitrate) + 'kbit/s';
  // ...
end;
```

Kapitel 2. ID3v1-Tags

Der ID3-Tag in der Version 1 ist sehr einfach aufgebaut, leicht zu finden und leicht auszuwerten. Dementsprechend ist auch hier die zuständige Klasse recht übersichtlich.

Das Grundprinzip bei der Anwendung ist dabei einfach:

- Tag-Objekt erstellen
- Daten aus einer Datei oder einem Stream lesen
- gewünschte Felder ändern
- Tag wieder in die Datei schreiben
- Tag-Objekt freigeben

Lässt man das Auslesen der Informationen weg, dann werden die Informationen, die nicht explizit neu gesetzt werden, aber bereits in der Datei vorhanden sind, gelöscht. Bei ID3v1-Tags ist das nicht so tragisch, weil in denen sowieso fast nichts drinsteht, aber bei den ID3v2-Tags kann das sehr ärgerlich werden.

Die Demo "ID3v1 und Mpeg" zeigt die Details. Die einzelnen Eigenschaften und Methoden der Klasse sollten selbsterklärend sein und bedürfen keiner weiteren Erklärungen.

Kapitel 3. ID3v2-Tags, Übersicht.

Mit Version 0.4 von Mp3FileUtils wurde die Klasse TID3v2Tag stark erweitert. Sie ist dabei vollständig abwärtskompatibel zu älteren Versionen, auch wenn einige Teile jetzt als veraltet markiert sind und in einer der nächsten Version verschwinden werden.

Die Klasse bietet viele Möglichkeiten, und ist daher für einen Einsteiger vielleicht recht unübersichtlich. Ich möchte hier einige Hinweise zur Benutzung dieser Klasse geben.

Man kann die Properties und Methoden grob in drei Level einteilen.

- **Level 1** bietet elementaren Zugriff auf die wichtigsten Eigenschaften eine ID3-Tags. Die Verwendung funktioniert hier genau wie beim ID3v1-Tag über die einzelnen Properties wie Artist, Titel oder Album. Als Programmierer kann man hierbei relativ wenig verkehrt machen.
- **Level 2** bietet Zugriff auf etwas detailliertere Eigenschaften. Es ist so, dass gewisse Informationen mehrfach im ID3v2-Tag vorkommen dürfen. Diese Informationsfelder beinhalten in der Regel eine Beschreibung dessen, welche Information in ihnen zu finden ist. Auf Level 2 lassen sich gezielt diese Informationen auslesen, wenn die Beschreibung bekannt ist. Auch die Verwendung dieser Methoden ist relativ ungefährlich. Es können aber unerwünschte Nebeneffekte auftreten - auch und vor allem in Verbindung mit anderen Playern oder Taggern.
- **Level 3** bietet Zugriff auf einer recht elementaren Ebene. Hier hat man mehr oder weniger direkten Zugriff auf die einzelnen Informationsfelder (=Frames) des ID3v2-Tags. Die Lese-Routinen können natürlich beliebig benutzt werden, bei Schreibzugriff sollte man genau wissen, was man tut, um den Tag nicht zu beschädigen.

ID3-Tags ändern

Änderungen am ID3-Tag einer Datei sollten **immer** nach folgendem Schema ablaufen:

1. Tag-Objekt erstellen
2. Daten aus einer Datei oder einem Stream lesen
3. gewünschte Felder ändern
4. Tag wieder in die Datei schreiben
5. Tag-Objekt freigeben

Wird vor den Änderungen (Schritt 3) der Tag nicht aus der Datei gelesen, werden beim Schreiben nur die explizit gesetzten Felder geschrieben. Alle anderen Daten gehen dann verloren!

Kapitel 4. ID3v2 - Einfacher Zugriff (Level 1)

Für die Standard-Informationen wie Artist oder Titel stehen wie beim ID3v1-Tag einfache Properties zur Verfügung, die nach Belieben ausgelesen und verändert werden können.

Die folgenden grundlegenden Methoden sollten in ihrer Funktion selbsterklärend sein. Der Rückgabewert gibt Auskunft darüber, ob und wenn ja was während der Ausführung schiefgelaufen ist.

```
function ReadFromStream(Stream: TStream): TMP3Error;
function WriteToStream(Stream: TStream): TMP3Error;
function RemoveFromStream(Stream: TStream): TMP3Error;
function ReadFromFile(Filename: WideString): TMP3Error;
function WriteToFile(Filename: WideString): TMP3Error;
function RemoveFromFile(Filename: WideString): TMP3Error;
procedure Clear;
```

Auch ein Teil der Properties bedarf glaube ich keiner besonderen Erläuterungen. Genre, Track und Jahr können hier im Gegensatz zum v1-Tag beliebige Strings sein.

```
property Title:   WideString;
property Artist:  WideString;
property Album:   WideString;
property Genre:   WideString;
property Track:   WideString;
property Year:    WideString;
```

Folgende Eigenschaften werden recht häufig "einfach so" verwendet. Hier kann es unter Umständen zu ungewünschten Effekten in Verbindung mit anderen Taggern oder Playern kommen. Das sollte aber nur dann eintreten, wenn man auch ausgiebig von Level 2 oder 3 Gebrauch macht.

```
property Comment: WideString;
property URL:     String;
property Lyrics : WideString;
property Rating:  Byte;
```

- **Comment**. Im ID3v2-Tag gibt es kein Feld für *den* Kommentar. Es gibt ein Feld für *einen* Kommentar, und dieses Feld kann mehrfach im Tag vorkommen. Die Property `Comment` wählt aus diesen möglicherweise mehrfach vorhandenen Feldern eines aus. Es kann passieren, dass andere Tagger oder Player ein anderes Feld auswählen. In der Regel kommt das nicht vor - meistens gibt es nur ein Kommentar-Feld in der Datei, aber iTunes z.B. macht von diesem Feld stärker Gebrauch um "private Informationen" abzulegen.
- **URL**. Hier gibt es dasselbe Problem wie beim Kommentar. Im ID3v2-Standard sind mehrere Felder für unterschiedliche URLs definiert, so dass sich für *die* URL ein weiteres Feld durchgesetzt hat (der WXXX-Frame, also eine benutzerdefinierte URL), was auch wieder mehrfach im Tag

vorkommen kann.

- **Lyrics**. Wieder das gleiche. Hier aber nicht ganz so tragisch, da erstens nur die wenigsten Mp3-Dateien überhaupt Lyrics im ID3-Tag enthalten, und wenn, dann auch nur einmal.
- **Rating**. Der Bewertungs-Frame kann auch mehrfach vorkommen. Eine Bewertung ist im Bereich 1..255 möglich, wobei 255 am Besten ist. Der Wert 0 in diesem Feld soll als "neutral" interpretiert werden. Gedacht ist es so, dass in diesem Frame eine E-Mail-Adresse von demjenigen zu finden ist, der die Bewertung vorgenommen hat. Meistens findet man auch hier nur eine - obwohl mehrere erlaubt wären. Sehr oft findet mal als "E-Mail" die Info "Windows Media Player 9 Series" - dann hat der WMP die Bewertung hinterlassen. Die Property Rating liest und schreibt das erste Rating-Feld, was im Tag zu finden ist, egal von wem es hinterlassen wurde.

Aus Version 0.3 von MP3FileUtils stehen noch folgende weitere Eigenschaften direkt zur Verfügung - einige davon machen mehr, andere etwas weniger Sinn. Viele davon sollten nachträglich nicht bearbeitet werden.

```
property Composer:            WideString;
property OriginalArtist:      WideString;
property Copyright:           WideString;
property EncodedBy:           WideString;
property Languages:           WideString;
property SoftwareSettings:    WideString;
property Mediatype:           WideString;
property id3Length:           WideString;
property Publisher:           WideString;
property OriginalFilename:    WideString;
property OriginalLyricist:    WideString;
property OriginalReleaseYear: WideString;
property OriginalAlbumTitel:  WideString;
```

Kapitel 5. ID3v2 - Darf's ein bisschen mehr sein? (Level 2)

Wie schon erwähnt, können einige Informationen mehrfach im ID3v2-Tag vorhanden sein. Dazu gehören Kommentare, Lyrics, benutzerdefinierte URLs, Bilder und Bewertungen. Es gibt noch einige andere solcher Informationsfelder, die aber (noch) nicht direkt unterstützt werden.

Diese mehrfach vorkommenden Frames sind etwas anders aufgebaut als die "normalen" Text-Frames. Im Wesentlichen unterscheiden sie sich dadurch, dass sie zusätzlich zur eigentlichen Information eine Beschreibung des Inhalts enthalten. Die Frames können mehrfach vorkommen, aber es sollten keine zwei Frames desselben Typs mit derselben Beschreibung vorkommen.

Dafür stehen dann folgende Methoden zur Verfügung, die Parameterliste ergibt sich aus dem Aufbau der Frames.

Kommentare und Lyrics

```
procedure SetExtendedComment
            (Language: string; Description: WideString; Value:WideString);
function GetExtendedComment
            (Language: string; Description: WideString): WideString;

procedure SetLyrics
            (Language: string; Description: WideString; Value: WideString);
function GetLyrics
            (Language: string; Description: WideString): WideString;
```

Lyrics und Kommentare enthalten neben einer Beschreibung auch ein Feld für die Sprache. Die Sprache wird dabei als ein drei Zeichen langer String erwartet. Die Ländercodes sind in der Stringlist `LanguageCodes` gegeben, die zugehörigen Bezeichnungen in `LanguageNames`. Wird als `Language` ein Sternchen (`'*'`) oder ein Leerstring (`''`) übergeben, wird der erste Frame gewählt, dessen Beschreibung mit `Description` übereinstimmt. Ist `Description = '*'`, so wird der erste Frame gewählt, dessen Sprache übereinstimmt. Die Setter/Getter der Properties `Lyrics` und `Comment` verwenden `Language='*'` und `Description = ''`.

Wird als Value ein Leerstring übergeben, so wird der entsprechende Frame gelöscht.

Benutzerdefinierte URLs

Bei den benutzerdefinierten URLs fehlt das Sprachfeld, und die URL selbst ist ein String, kein WideString.

```
function GetUserDefinedURL(Description: WideString): String;
procedure SetUserDefinedURL(Description: WideString; Value: String);
```

Sternchen in den Beschreibungen werden hier nicht besonders behandelt. `Value=''` bewirkt auch hier ein Löschen des Frames. Der Setter/Getter der

Property `URL` verwendet `Description=''`.

Bilder

```
procedure GetPicture(Stream: TStream; Description: WideString);
procedure SetPicture (MimeTyp: String; PicType: Byte;
                                Description: WideString; Stream: TStream);
```

Um ein Bild zu lesen, benötigt man nur eine Beschreibung. Ist `Description = ''`, so wird ein beliebiges Bild zurückgeliefert. Die Bildaten stehen anschließend in `Stream` und können mit entsprechenden Routinen ausgelesen werden. Das Format der Bilder ist fast immer JPEG oder PNG.

Zum Schreiben eines Bildes wird zusätzlich der Mime-Typ und der Typ des Bildes benötigt. MimeTyp sollte `'image/jpeg'` oder `'image/png'` sein, PicType ein Wert zwischen 0 und 20. Die Bedeutung der einzelnen Bild-Typen ist im Konstanten-Array `Picture_Types` definiert. Die Bilddaten selbst müssen in Form eines Streams übergeben werden.

Ein Bild kann aus dem Tag entfernt werden, indem SetPicture mit `Stream=Nil` aufgerufen wird.

Wenn bekannt ist, dass im ID3-Tag schon ein Bild vorhanden ist, dann kann durch Übergabe von `MimeTyp='*'`, `Description='*'` oder einen leeren Stream (d.h. `Stream.Size=0`) die jeweilige Information des vorhandenen Bildes übernommen werden.

Bewertungen

Bewertungsframes enthalten als Beschreibung einen String, der eine E-Mail Adresse sein soll - es aber oft nicht ist. Setter und Getter der Property Rating nutzen `aEMail='*'`, um eine beliebige Bewertung auszulesen oder zu ändern.

```
function GetRating(aEMail: String): Byte;
procedure SetRating(aEMail: String; Value: Byte);
```

Zum Auslesen der Bewertung, die der Windows-Media-Player vorgenommen hat, scheint der Wert `'Windows Media Player 9 Series'` der richtige zu sein - ohne Gewähr. Man könnte hiermit einen Player bauen, bei dem jeder User die Mp3-Dateien nach seinem Geschmack bewertet. Ob das besonders viel Sinn macht, sei mal dahin gestellt.

Weitere Text-Informationen

Neben den Text-Informationen, die über die Properties bereitgestellt werden, gibt es eine ganze Reihe weitere, die auf www.id3.org definiert sind. Diese können über die ID gelesen und gesetzt werden. Die Setter und Getter der Text-Properties rufen genau diese Methoden mit der entsprechenden ID auf.

```
function GetText(FrameID: TFrameIDs):WideString;
procedure SetText(FrameID:TFrameIDs; Value: WideString);
```

- **Vorsicht mit den IDs!** Es wird keine Überprüfung durchgeführt, ob die

übergebene Frame-ID eine gültige ID für einen Textframe ist. Textframes erkennt man daran, dass der ID-String mit einem 'T' beginnt. Es wird nur überprüft, ob ein Frame mit der ID in der Tag-Version definiert ist.

URLs

Neben den benutzerdefinierten URLs gibt es einige vorgefertigte, die in der Struktur etwas anders aussehen. Diese lassen sich über die jeweilige ID ansprechen.

```
function GetURL(FrameID: TFrameIDs):String;
procedure SetURL(FrameID:TFrameIDs; Value: String);
```

- **Auch hier: Vorsicht mit den IDs!** Gültige IDs für URL-Frames sind `IDv2_AUDIOFILEURL`, `IDv2_ARTISTURL`, `IDv2_AUDIOSOURCEURL`, `IDv2_COMMERCIALURL`, `IDv2_COPYRIGHTURL`, `IDv2_PUBLISHERSURL`, `IDv2_RADIOSTATIONURL`, `IDv2_PAYMENTURL`.

Kapitel 6. ID3v2 – Expertenmodus (Level 3)

- **Dringende Warnung am Anfang: Vorsicht!** Schon auf Level 2 kann man etwas Unsinn anstellen. Wenn der Programmierer auf Level 3 Mist baut, kann der gesamte Tag für andere Programme unlesbar werden.

Die bisherigen Methoden liefen auf Tag-Ebene ab. D.h. wir haben Mehoden der Klasse TID3v2Tag verwendet, um den ID3-Tag zu lesen oder zu verändern. Ein solcher Tag besteht wie schon angeklungen aus verschiedenen Frames, und jeder Frame speichert eine spezifische Information. Die bisher vorgestellten Methoden haben intern den passenden Frame gesucht und diesen dann bearbeitet.

Mit Version 0.4 bietet Mp3FileUtils auch einen direkteren Zugriff auf die Frames, um diese zu verändern. Dafür müssen in einem ersten Schritt die Frames nach außen geliefert werden. Dafür stehen mehrere Methoden zur Verfügung, die jeweils eine Teilmenge der vorhandenen Frames zurückliefern.

Die Frames in einem ID3-Tag lassen sich in verschiedene Gruppen einteilen, he nachdem, welche Art von Information sie enthalten. Mp3FileUtils unterstützt mit Version 0.4 Textframes (z.B. Artist, Titel), Kommentare, Lyrics, Benutzerdefnierte URLs, Bilder, Bewertungen und URLs. Es gibt noch eine ganze Reihe weiterer, die noch nciht unterstützt werden.

```
function GetAllFrames: TObjectlist;
function GetAllTextFrames: TObjectlist;
function GetAllCommentFrames: TObjectlist;
function GetAllLyricFrames: TObjectlist;
function GetAllUserDefinedURLFrames: TObjectlist;
function GetAllPictureFrames: TObjectlist;
function GetAllPopularimeterFrames: TObjectlist;
function GetAllURLFrames: TObjectlist;
```

In den Objectlists, die in diesen Methoden erzeugt werden, befinden sich dann mehr oder weniger viele Objekte vom Typ `TID3v2Frame`.

ID3v2-Frames

Bevor wir weitermachen, ist es sinnvoll, einen kurzen Überblick darüber zu geben, wie ein Frame in einem ID3-Tag aussieht. Das Problem dabei ist, dass der Aufbau der Frames in den unterschiedlichen ID3v2-Tag Versionen nicht gleich ist.

	Version 2.2	Version 2.3	Version 2.4
ID	3 Zeichen	4 Zeichen	4 Zeichen
Größenangabe	3 Byte	4 Byte	4 Byte (sync-safe)
Flags	-keine-	2 Bytes	2 Bytes
Daten	x Bytes	x Bytes	x Bytes

ID, Größenangabe und Flags bilden den *Header* des Frames, an den sich die eigentlichen Daten anschließen. Wie die Daten genau aussehen, hängt vom Typ des Frames ab, der über die ID definiert ist.

Die Klasse TID3v2Frame stellt zunächst einmal über einige Properties Informationen über den Frame an sich bereit:

```
property FrameID: String;
property FrameTypeDescription: String;
property FrameType: TID3v2FrameTypes;
```

Die FrameID ist die ID aus dem Header. Die FrameTypeDescription gibt eine kurze Beschreibung des Frames, wie sie auf id3.org zu finden ist. Der FrameType ordnet die Frames grob nach ihrem Inhalt, also TextFrame, Bild, Kommentar, Rating usw.

Diese Informationen können genutzt werden, um die über GetAll***Frames erhaltenen Frames in einer ListView oder einer vergleichbaren Komponente anzuzeigen.

Daten lesen und schreiben

Zum Auslesen und Schreiben der Informationen auf Frame-Ebene stehen für die einzelnen Frame-Typen passende Methoden bereit.

```
function GetText: Widestring;
procedure SetText(Value: Widestring);

function GetCommentsLyrics(out Language: String;
                  out Description: WideString): Widestring;
procedure SetCommentsLyrics(Language: String; Description, Value: Widestring);

function GetUserdefinedURL(out Description: WideString): string;
procedure SetUserdefinedURL(Description: WideString; URL: string);

function GetURL: String;
procedure SetURL(Value: String);

function GetPicture(out Mime: String; out PicType: Byte;
                  out Description: WideString; PictureData: TStream): Boolean;
procedure SetPicture(Mime: String; PicType: Byte;
                  Description: WideString; PictureData: TStream);

function GetRating(out UserEMail: String): Byte;
procedure SetRating(UserEMail: String; Value: Byte);

procedure GetData(Data: TStream);
procedure SetData(Data: TStream);
```

Die Bedeutung der einzelnen Methoden sollte mit dem Wissen aus dem vorigen Kapitel klar sein. Wichtig dabei ist:

- **Keine Sternchen!** Die Übergabe von * auf Tag-Ebene war dazu da, um einen bestimmten Frame zu finden. Hier haben wir ihn gefunden, und ein '*' wird dann auch so geschrieben!
- **Die Methoden nur auf Frames des passenden Typs anwenden!** Es macht keinen Sinn, auf einem Picture-Frame die Methode SetText

aufzurufen. Dadurch wird der Frame ungültig und somit unter Umständen der gesamte Tag. Es wird keine Überprüfung durchgeführt, ob die angewandte Methode Sinn macht oder nicht. Dafür ist der Programmierer zuständig.

- **Die Set-Methoden mit Vorsicht verwenden!** Bei der Bearbeitung von Frames muss darauf geachtet werden, dass der Tag als Gesamtes nicht ungültig wird. Das heißt unter anderem, dass die Beschreibungsfelder nicht nachträglich geändert werden sollten, damit keine doppelten Beschreibungen entstehen können.
- **SetData ist mit äußerster Vorsicht zu verwenden!** Diese Methode kann benutzt werden, um Frame-Typen, die hier nicht explizit unterstützt werden, zu beschreiben. Der Inhalt des Datenstreams muss dabei den Spezifikationen von www.id3.org entsprechen. Dabei müssen auch ggf. vorhandene Unterschiede zwischen den einzelnen Tag-Versionen berücksichtigt werden!

Neue Frames erstellen und vorhandene löschen

Die Frames werden von der Klasse TID3Tag verwaltet. Daher dürfen nur über diese Klasse Frames erstellt oder gelöscht werden!

Zum Löschen eines vorhandenen Frames steht

```
procedure DeleteFrame(aFrame: TID3v2Frame);
```

zur Verfügung, zum Einfügen eines neuen

```
function AddFrame(aID: TFrameIDs): TID3v2Frame;
```

Bei der ID muss beachtet werden, dass einige Frames nicht mehrfach vorkommen dürfen. Für eine Liste der erlaubten Frames stehen die Methoden

```
function GetAllowedTextFrames: TList;
function GetAllowedURLFrames: TList;
```

zur Verfügung. Andere, von Mp3FileUtils direkt unterstützte Frame-Typen dürfen mehrfach vorkommen.

- **Keine Frames doppelt erstellen, die nur einfach vorkommen dürfen!** Ansonsten kann es zu unschönen Effekten kommen, dass z.B. die Anzeige des Künstlers oder Titels in unterschiedlichen Playern abweicht.
- **Keine Frames direkt freigeben oder erstellen!** Konstruktor und Destruktor der Klasse TID3v2Frame sind außerhalb von TID3v2Tag absolut tabu!

Bei der Initialisierung eines neuen Frames ist zusätzlich zu beachten, dass ein Frame dieses Typs mit der entsprechenden Beschreibung noch nicht vorhanden ist. Dafür stellt die Klasse TID3v2Tag die folgenden Methoden zur Verfügung.

```
function ValidNewCommentFrame(Language: String;
                              Description: WideString): Boolean;
function ValidNewLyricFrame(Language: String; Description: WideString): Boolean;
function ValidNewPictureFrame(Description: WideString): Boolean;
function ValidNewUserDefUrlFrame(Description: WideString): Boolean;
function ValidNewPopularimeterFrame(EMail: String): Boolean;
```

Flags

Im Header des ID3-Tags und/oder der Header der einzelnen Frames gibt es einige Bytes mit Flags, also Bitwerten, die gewisse Eigenschaften anzeigen, die gesetzt oder nicht gesetzt sind.

Einige davon haben Einfluss darauf, wie der Tag bzw. Frame gelesen werden muss. Einige unterstützt MP3FileUtils, andere nicht.

Auf Tag-Ebene ist hier eigentlich nur ein Flag weiter interessant, nämlich `FlgUnsynch`. Dieser ist schätzungsweise in 99.99% aller Mp3-Dateien nicht gesetzt und wird wahrscheinlich nur von sehr wenigen (wenn überhaupt) Taggern schreibend unterstützt. Ist dieser Flag bei Version 2.3 oder 2.2 gesetzt, gibt es in aller Regel keine Probleme, bei 2.4 kommen einige Player ins Stolpern.

Auf Frame-Ebene (nur bei Version 2.3 und 2.4) gibt es mehrere Flags:

```
property FlagTagAlterPreservation : Boolean;
property FlagFileAlterPreservation: Boolean;
property FlagReadOnly             : Boolean;

property FlagCompression          : Boolean;
property FlagEncryption           : Boolean;
property FlagGroupingIndentity    : Boolean;
property FlagUnsynchronisation    : Boolean;
property FlagDataLengthIndicator  : Boolean;

property FlagUnknownStatus        : Boolean;
property FlagUnknownEncoding      : Boolean;
```

Die ersten beiden geben Auskunft darüber, wie die der Frame zu behandeln ist, wenn der ID3-Tag oder die Datei verändert wird. Ist einer dieser Flags nicht gesetzt, sollte der Frame bei einer entsprechenden Änderung gelöscht werden (es sei denn, man ändert die Information entsprechend). Der dritte gibt an, ob der Frame schreibgeschützt zu behandeln ist. Es liegt am Programmierer, sich daran zu halten oder nicht. Bei einer Veränderung des Frames wird FlagReadOnly automatisch entfernt.

`FlagCompression` und `FlagEncryption` geben an, dass der Frame komprimiert oder verschlüsselt ist. Ist einer gesetzt, führt das dazu, dass der Frame für Mp3FileUtils nicht lesbar ist. Das ist auch der Fall, wenn ein unbekannter Flag gesetzt ist. Schribzugriff auf den Frame ist aber möglich - in dem Fall werden diese beiden Flags entfernt.

`FlagGroupingIndentity`, `FlagUnsynchronisation` und `FlagDataLength-Indicator` werden unterstützt, letzterer allerdings nur lesend. Sie haben Einfluss darauf, wie der Datenteil des Frames zu behandeln ist. Für Details verweise ich auf www.id3.org.

- **GroupingIdentity und Unsynchronisation sollten nicht verwendet werden!** Auf Grund der sehr spärlichen Verbreitung kann es damit zu

Inkompatibilitäten zu anderen Playern oder Taggern kommen.

Padding

Der ID3v2-Tag steht am Anfang der Datei. Ändert man diesen Tag, so müsste prinzipiell die gesamte Datei neu geschrieben werden, wenn sich dabei die Länge ändert. Um das zu vermeiden, kann an den eigentlichen ID3v2-Tag eine Art Buffer angefügt werden. Dieses Anfügen von $00-Bytes an den Tag nennt man *Padding*.

Mp3fileUtils stellt dafür folgende Eigenschaften bereit:

```
property UsePadding;
property UseClusteredPadding;
property PaddingSize;
```

`UsePadding` gibt an, ob generell Padding erwünscht ist. `UseClusteredPadding` wählt die Paddinggröße dabei so, dass die fertige Datei die Cluster auf dem Datenträger komplett ausfüllt. Der Buffer wird also so groß gewählt, so dass nicht wirklich mehr Speicherplatz auf dem Datenträger benötigt wird. Standardwert ist bei beiden `True`.

Die read-only Eigenschaft `PaddingSize` gibt die tatsächliche Größe des Padding-Bereichs an, also die Größe, die noch für weitere Daten zu Verfügung steht, ohne die Datei komplett neu schreiben zu müssen.

- Das Vorhandensein oder Fehlen von Padding ist unproblematisch. Wer unbedingt ein paar Byte sparen möchte, kann es ausschalten, ansonsten ist es sinnvoll, Clustered Padding zu benutzen. Die Datei wird dann etwas größer, der benötigte Platz auf dem Datenträger allerdings nicht.

Kapitel 7. Unicode

In vielen Frames stehen Informationen in Form von kurzen Texten. Bei vielen kann man dabei die Textkodierung aus einigen Varianten wählen. Zur Verfügung stehen Iso8859-1, UTF-16 ("Unicode") und UTF-8. UTF-16 und UTF-8 machen keine Probleme - die Zeichensätze sind so definiert, dass praktisch alle Zeichen kodiert werden können. Bei Iso8859-1 sieht das anders aus - hier stehen nur die 256 Zeichen zur Verfügung, die man in den meisten westeuropäischen Sprachen benötigt.

Leider werden oft auch andere Zeichensätze verwendet, und die Kodierung dann auf Iso8859-1 gesetzt. Das hat zur Folge, dass die Textinformationen auf dem "Heimatsystem" korrekt dargestellt werden, aber nicht in anderen Sprachräumen.

Mp3FileUtils bietet die Möglichkeit, anhand des Dateinamens den Sprachraum zu ermitteln, aus dem die Datei stammt, um daraus einen möglichen Zeichensatz zu ermitteln, der für die Dekodierung der Texte benutzt wird.

In Version 0.4 und früher wurde dazu die Unit DIConverters benutzt, die zusätzlich heruntergeladen werden musste und nicht Teil von Mp3FileUtils ist.

Mit Version 0.5 wurde auf diese Unit verzichtet, und stattdessen die Windows-API-Funktion `MultiByteToWideChar` benutzt. Das macht die Verwendung deutlich einfacher, wenn auch einige Codepages nun nicht mehr zur Verfügung stehen.

Benutzung dieser Funktion wird über die Eigenschaft `AutoCorrectCodepage` geregelt. Initial ist sie auf `False` gesetzt, d.h. es wird keine Codepage-Konvertierung durchgeführt. Um eine wahrscheinliche Codepage anhand des Dateinamens zu ermitteln, gibt es in der Unit `U_Charcode` die Funktion `GetCodepage`, die entweder nur mit einem Dateinamen, oder zusätzlich mit einem Parameter vom Typ `TConvertOptions` aufgerufen werden kann, um dem User mehr Einstellungsmöglichkeiten zu gewähren, d.h. z.B. um zwischen verschiedenen griechischen Codepages wählen zu können.

Die Property `AlwaysWriteUnicode` bewirkt, dass grundsätzlich immer Unicode verwendet wird. Ansonsten wird darauf verzichtet, wenn der Text nur aus Zeichen besteht, die im Iso8859-1-Zeichensatz vorkommen.

Delphi 2009

Mit Version 0.5 ist MP3FileUtils auch kompatibel zu Delphi 2009. In diesem Fall wird die Verwendung der TNT Unicode Controls, die per Compilerschalter aktiviert werden können, um auch unter Delphi 2007 oder früher Dateien mit Unicodenamen öffnen zu können, automatisch deaktiviert. Selbstverständlich sind ältere Delphi-Versionen weiterhin unterstützt, und auch Unicode geht (dann mit der Verwendung der TNTs) weiter so wie gewohnt.