Enigma Transportable File Specification

The following PRELIMINARY documentation of the Enigma File Format is being provided by Coda engineering in response to requests from some of our customers.

We solicit your input as to the usefulness of what is provided here, and ask that you specify in detail the additional documentation of the file format you would require. Coda will use your input to determine the best course of action concerning the publication of the Enigma file format.

Please send your responses to the following address:
tech_support@codamusic.com
Please title the subject of your message "FILE FORMAT PUBLICATION."

Disclaimer

By releasing this information, Coda can take no responsibility for how it is used. If you choose to use this information to make changes to your files, Coda cannot be responsible for mistakes or corruption that may result. **Always keep a back up of the _unaltered_ version of your file.**

The information in this document is correct to the best of our knowledge. We will make any reasonable effort to correct any errors that may be found.

This document pertains to Finale 97 for Mac and Windows. It is subject to change without notice in future versions.

Finally, this is a technical document for technical readers. It is not intended as a tutorial. A working knowledge of C/C++ syntax is required. We will be working with ETF (Enigma Transportable File) files as a convenient means of describing the format.

We'll start with the entry pool. This is a good starting point because it is relatively self-contained, and is immediately useful.

The Entry Pool

In Enigma terminology, an "entry" is either a note, a chord, or a rest. Entries are streamed together in a doubly linked list that roughly corresponds to a voice on a staff. Certain situations like mirrors and voice 2 create complications that will not be dealt with at this time.

Each entry can have up to twelve notes.

Here is a typical entry pool from an ETF, with annotations:

entries

```
^eE(1) 0 2 1024 0 $C0000800 128 3        <<== entry
   48 $80030000                          <<== note
   80 $80020000                          <<== note
   112 $80010000
^eE(2) 1 3 1024 0 $C0000800 128 1
   128 $80010000
^eE(3) 2 4 1024 0 $C0000800 128 3
   48 $80010000
   80 $80020000
   112 $80030000
^eE(4) 3 0 1024 0 $C0000800 128 1
   48 $80010000
```

Each entry is signified by the tag ^eE with the entry number in parentheses. The line containing the ^eE contains information that pertains to the entry as a whole. Information about individual notes within the entry follows on zero or more subsequent lines. (NOTE: All numerical values in an ETF are in decimal unless preceeded by a '$', in which case they are in hexadecimal.)

Entry fields

tag	field 1	field 2	field 3	field 4	field 5	field 6	field 7
^E(1)	0	2	1024	0	$C0008000	128	3

field 1: link to previous entry (0 if none) (32 bits)
field 2: link to next entry (0 if none) (32 bits)
field 3: the entry duration in EDUs (Enigma Duration Units, 1024 == quarter note) (unsigned 16 bits)
field 4: manual positioning in EVPUs (Enigma Virtual Page Units, 288 per inch) (signed 16 bits)
field 5: entry flag (see below)
field 6: extended entry flag (see below)
field 7: number of note records following

The Entry Flag

The entry flag is a 32 bit hexadecimal value consisting of the bit flags and fields described below.

All bits not described below are either for internal use only or reserved for future expansion. By "internal use", we mean flags that are used at run-time but whose state in the file is not meaningful.

There is a direct one-to-one correspondence between these flags and the checkboxes in the Frame Dialog in Finale.

Most rests are considered "floating rests: that is, they "float" to the midline of the staff. Since they have no pitch content, floating rests have no note records.

When rests are manually positioned away from the midline, the FLOATREST bit is turned off and a note record is added to indicate vertical position.

Enigma makes extensive use of bit flags that indicate the existence of supplementary records. Those records are not documented here. Please let us know specifically what you need.

```
#define SETBIT          0x80000000L     // always set (indicates a legal entry)
#define NOTEBIT         0x40000000L     // set for note, off for rest
#define CNTLRBIT        0x20000000L     // set if entry is a controller ("V2 launch")
#define CNTLBIT         0x10000000L     // set for voice 2 notes
```

```
#define CHORDBIT        0x04000000L     // indicates existence of chord record
#define FLOATREST       0x01000000L     // indicates "floating" rest

#define GRACENOTE       0x00800000L     // set for grace notes */
#define NOTEDTAIL       0x00400000L     /* indicates existence of note detail record
                                           (controls note reduction, notehead displace-
                                           ment, accidental displacement, alternate note
                                           head) */
#define IMRKDTAIL       0x00200000L     /* indicates existence of articulation assign-
                                           ment record */
#define TEXTDTAIL       0x00100000L     /* indicates existence of lyric assignment
                                           record */
#define TUPLSTART       0x00080000L     /* tuplet start (indicates existence of tuplet
                                           record) */
#define CTUPPARA        0x00040000L     /* "controlled tuplet parasite bit" -- currently
                                           only passively supported; may be dropped in
                                           future revs */
#define PRFMDATA        0x00020000L     /* indicates existence of performance record
                                           (MIDI velocity, note duration alterations) */
#define IGNOREBIT       0x00008000L     // set to hide the entry
#define BMEXTDTAIL      0x00004000L     // mask for beam extension records
#define FLIPTIE         0x00002000L     // set to freeze ties in stem direction
#define SPECIALALTS     0x00001000L     // mask for tie and dot alteration records
#define BEATBIT         0x00000800L     /* mask for beatbit, used to determine beam-
                                           ing */
#define SECBEAMBIT      0x00000400L     /* mask for secondary beam breaks record */
#define CNTLCONTBIT     0x00000200L     /* mask to continue controlled beaming. This
                                           bit is set on the first controlled entry after a
                                           controller to indicate it should be grouped
                                           with the previouscontrolled group. (This is
                                           currently only passively supported by
                                           Enigma/Finale and will most likely be
                                           dropped in future versions). */
#define FREEZSTEM       0x00000100L     /* set to freeze stem direction (see
                                           UPSTEMBIT) */
#define STEMDTAIL       0x00000080L     /* mask for detail stemming (stem length,
                                           displacement, custom stems) */
#define CROSUPBIT       0x00000040L     // mask for cross-stave upward placement
#define CROSDWBIT       0x00000020L     // mask for cross-stave downward placement
#define REVUPSTEMBIT    0x00000010L     // mask to reverse all upstems in the entry
#define REVDWSTEMBIT    0x00000008L     // mask to reverse all downstems in the entry
#define DBLSTEMBIT      0x00000004L     // mask for double stemming
#define SPLITBIT        0x00000002L     // mask for split stemming
#define UPSTEMBIT       0x00000001L     /* mask for up or down stem (1 == up); this
                                           is recomputed on the fly unless freeze stem is
                                           set */
```

The Extended Entry Flag

The extended entry flag is a 16 bit value. Although it is a bit flag like the entry flag, it is represented in the ETF as a decimal rather than hex value. There is no particular reason for this, and it does make the decoding somewhat trickier, but it has been retained to avoid conversion issues.

#define STAFFEXPRDETAIL0x0001	/* indicates existence of staff expression assignment record */
#define CHECKACCIS 0x0004	/* this bit forces Finale to recompute the ACCIBITs when the frame is loaded. */
#define SMARTSHAPEDETAIL0x0020	// for smart shape entry attachment
#define NOLEGER 0x0040	/* if set, don't draw leger lines for this entry (used for Penderecki-like "highest note possible" stuff) */
#define ENTRY_IS_SORTED0x0080	/* if set (which it should be most of the time) the notes in the entry are sorted in the normal way, for standard notation. If not set, the notes are in some other ordering, which for now would mean percussion notation.*/

The Note Record

The note record is comprised of two fields, the TCD and the note flag.

TCD	note flag
48	$80010000

TCD (Tone Center Displacement)

This is a 16 bit value divided into two fields that describes the pitch and alteration of the note. The alteration is stored in the low order nybble (four bits) as a signed quantity. Example: 0 == unaltered ("natural"), 1 == "sharp", 2 == "double sharp", -1 == "flat", etc.

Note that alteration refers to the note's relationship to the current key, not to whether an accidental appears on the note, or what the accidental would be. Example: in the key of G major, an F natural would have an alteration of -1, being a half step lower than the diatonic F#.

Four bits allows alterations from -8 to +7. Traditional western music rarely if ever exceeds double sharp (+2) and double flat (-2). Additional alterations could be useful in a microtonal or non-western context.

Historical note: In earlier versions of Finale, the number of bits alloted to alterations and pitch was user-definable. This was the infamous "TCD Watershed Bit" that confused many users of 2.6.3 and before. So now you know.

The remaining twelve bits of the TCD contain the harmonic value (pitch) for the note. Twelve bits gives us a range of -2048...2047. Pitch is always defined relative to the current key signature, with middle C as the reference point. Each value corresponds to a diatonic pitch in the current key. Thus, middle C in the key of C major has a harmonic value of 0. D above middle C has a harmonic value of 1, B below middle C has a harmonic value of -1, C an octave above middle C has a value of 7, and so on.

The harmonic value 0 will always be the tonic of the current key in the octave from middle C to the C above. For example, in G major, 0 is the G above middle C. This arrangement makes it easy to change the key in a region of music without having to alter any notes within the region.

The Note Flag

```
#define SETBIT          0x80000000L     // legality (must be set)
#define TSBIT           0x40000000L     // tie start
#define TEBIT           0x20000000L     // tie end

#define CROSSBIT        0x10000000L     // cross staff

#define UPSECBIT        0x08000000L     // upstem second
#define DWSECBIT        0x04000000L     // downstem second

#define UPSPBIT         0x02000000L     /* for split stems, set if the note goes on the
                                           upper stem
#define ACCIBIT         0x01000000L     /* indicates whether to show an accidental
                                           (will be recomputed during editing; to freeze
                                           this bit in place, see FREEZEACCI below)*/

#define PARENACCI       0x00800000L     // parenthesize accidental

/* Note IDs range from 1 to TGFNN (currently 12). Zero means no id. Note IDs were introduced
in Finale 2.x, so older files would have no id (0). We now enforce the protocol that all entries
should have note ids. */

#define TGFNID          0x001F0000L     /* since entries can now have only 12 notes
                                           (at one time they were designed to have 24),
                                           we don't really need five bits here. The high
                                           bit is reserved for future expansion. */

#define FREEZEACCI      0x00000002L     // if set, freeze the state of ACCIBIT
```

Others, Details, Entry Details

These are file structures that are identified by a tag (letters after ^ in each line) and by one comparator (others) or two (details), Comparators are the numbers enclosed in parentheses after the tag. The data for the detail or other follows the tag and comparator. Each other contains enough space for 6 twobyte values (or 3 fourbytes) and each detail holds 5 twobytes. If a structure cannot fit into one other or detail there are multiple incidences of the tag and comparator(s) to satisfy the structure size. An entry detail is a detail whose comparators are an entry number (msw and lsw of entry number)

^others
^01(65534) 0 0 0 0 0 0
^02(65534) -144 144 144 -144 0 0
^03(65534) 0 0 18 18 0 0
^04(65534) -72 -22 0 0 0 -32751
^05(65534) 0 0 0 144 0 0
^06(65534) 0 207 250 119 87 110
^07(65534) 98 35 220 186 46 106
^08(65534) 74 221 227 183 238 206
....

^details

^CL(0,0) 144 0 0 0
^#v1(0,0) -144 0 0 0
^#v2(0,0) -184 0 0 0
^#v3(0,0) -224 0 0 0
^#v4(0,0) -264 0 0 0
^#v5(0,0) -304 0 0 0
^#v6(0,0) -344 0 0 0
....

SPECIFIC EXAMPLES

For each record the data type (other,detail,text) is specified, followed by a tag, and a description of what the id (cmper and/or cmper2) means. An incident is an occurance of data stored under a specific tag and cmper, for others, or a single tag, cmper and cmper2 for details. Multiple incidents (multiple others/detail with same cmper, cmper2,tag) are used for structures that are too big to fit in a single other or detail (see Staff Spec, for example).

If a record is an entry detail, then it is a detail with cmper1=msw of entry number, cmper2 = lsw of entry number (the cmper1,cmper2 specify the entry number of the entry to which the detail is attached.)

Measure Spec

other, tag=MS, cmper=measure number

Most of this information is directly related to fields in the measure attributes dialog (measspace, auxflag, meflag). key correspond to the value set using the key signature tool and beats, divbeat to values set in the time signature tool, and repeat barline to values in the repeat selection dialog.

There is exactly one MS record for every measure in the piece.

See EDTMeasureSpec in EDATA.H for field definitions/documentation.

key:

A key signature is represented by a two byte number. A key is either considered "Linear" (e.g. Western tonality, mode), or "Non-linear" (arbitrary number of different accidentals)

Linear key sigs are < 16384. In other words, their top two bits are 0, the next six bits are a bank number (0..63), and the bottom eight bits are the accidentals (-128...127). Bank 0 is reserved for western tonality, major mode; bank 1 is reserved for western tonality, minor mode.

bank	accis
0 0 B B B B B B	A A A A A A A A

Nonlinear keys have no special bit layout. They are just any key >= MAXLINKEYS. They have no "bank" or "accis": that is what makes them non-linear.

```
#define MAXLINKEYS      16384       /* Maximum number for linear keys. A key
                                       signature greater than or equal to this is a
                                       non-linear key signature */
#define MAXLINBANKS     64          /* Maximum number of linear key formats.
                                       Each format can have from 127 flats to 127
                                       sharps. This corresponds to the six bits in the
                                       high byte of the keysig below MAXLIN-
                                       KEYS (0x3F00). The alterations are in the
                                       low byte. */

#define KEYBANK_MAJOR 0
#define KEYBANK_MINOR 1
```

time

Non-complex time signatures are represented by a beats field and a divbeat field. (Like the time signature dialog). Beats is the number of divisions in the measure, and divbeat is the EDU value of each division.

Non-linear keys and complex time signatures will be explained later...

example:

^others
...
^MS(1) 600 0 4 1024 1 16

Translation:

Measure 1 has:
measpace: 600 (measure width is 600 EVPUs)
key: 0 (C Major),
beats: 4
divbeat: 1024, (so time sig == 4/4)
auxflag: 0x0001, (use time signature spacing)
meflag: 0x0010 (normal barline)

Floats (Independent Key and Time)

detail, tag=FL, cmper1=instrument,cmper2=measure number

If this record exists for a measure, it overrides the timesig and/or key sig in the MS record for the instrument. Staff Spec must have FLOATKEYS or FLOATTIME

See EDTMeasureFloat in EDATA.H for field definitions/documentation.

Staff Spec

other, tag=IS, cmper=instrument number

Most of this information is directly related to fields in the Staff attributes dialog. (also, position full, abbrv staff name, and Staff Setup)

There is exactly one IS record for every staff (i.e. instrument) in the piece.

See EDTStaffSpec in EDATA.H for field definitions/documentation. Staff specs are stored over 3 incidences.

example:

^others
...
^IS(1) 0 0 0 0 0 0
^IS(1) 0 0 5 0 0 0
^IS(1) -772 -772 -4 0 0 0

Translation: (these 3 incidences comprise one Staff Spec)

Staff (instrument) 1 has:
botBarlineOffset: 0 (measure width is 600 EVPUs)
baseyoff: 0 (only used for tab staves),
mfont: 0 (mfont, sizeefx aren't used because staff doesn't use an independent notehead font)
sizeefx: 0
flag: 0 (nothing special, see flag defines below)
clefs: 0 (treble clef)
topLines: 0 (normal, non-custom staff)
botLines: 5 (five staff lines)
topBarlineOffset: 0
transposition: 0 (no transposition)
instflag: 0
dw_wRest: -772 (0xfcfc) (all rests are placed -4 steps from top line)
h_otherRest: -772 (0xfcfc)
stemReversal: -4 (stem reversal line is -4 steps from top line)
fullName: 0 (no text blocks for staff name)
abbrvName: 0

Group Spec

detail, tag=NG, cmper1=iuList (instrument list), cmper2=groupID

Most of this information is directly related to fields in the Group attributes dialog.

There is exactly one NG record for every group in the piece.

See EDTGroupSpec in EDATA.H for field definitions/documentation.

Group specs are stored over 3 incidences.

example:

^details
...
^NG(0,1) 1 2 0 -48 0
^NG(0,1) 3 -24 0 0 0
^NG(0,1) 1088 0 0 0 0

Translation: (these 3 incidences comprise one Group Spec)

starting instrument: 1
ending instrument: 2
full name text block id: 0 (no name)
full name x adj: -48 evpus
full name y adj: 0
bracket type: 3 (piano brace)
bracket pos: -24 evpus
bracket top: 0 from top staff line
bracket bottom: 0 from bottom staff line
bracket flag: 0 (no bracket on single staves)
flag: 1088 = 0x0440 (barline through all staves, connecting; normal barline style)
abrvNameID: 0 (no abbrv. name)
abrvNameXadj:0
abrvNameYadj:0
auxflag: 0

Performance data

entry detail, tag=ac

One incident for each note w/perf data, match noteID in struct w/note ID in entry:

This is the performance, midi information is usually modified/read through the Midi Tool.

Entry with the performance data must have its ef flagged with PRFMDATA

See EDTPerformanceData in EEDDATA.H for field definitions/documentation.

example:

^details
.
.
.
^ac(0,1) 1 4 8 0 63

means:

performance data for entry 1, note 1:
add 4 edus to start time, 8 edus to end time,
0 to midi number, 63 to velocity.

Instrument Used

other, tag=IU, cmper=IUList id

All the incidents under a single cmper define an instrument list (which instruments are in the list, and the distance between staves. Each incident represents a slot in the iulist. (slot=incident + 1, since slots are 1-based)

See EDTInstrumentUsed in EDATA.H for field definitions/documentation.

example:

^others
.
.
.
^IU(0) 1 0 0 0 -80
^IU(0) 2 0 0 0 -388

means:

staff set 0 has two staves (instruments) in it.
top line of instrument 2 is -388 evpus from top line of
instrument 1. instrument 1 is -80 evpus from top of page

Tempo

other, tag=AC, cmper =measure

There is an incident for each time dialation in a particular measure, incidents should be in chronological order.

This is the time dialation info that you normally modify with the Tempo Tool. The correlation between the dialog and the record is:

"Measure": (records are created under cmpers for each specified measure)
"Unit": incident
"Start Time in Measure": eldur translated to a beat
"Set To": flag=TDIL_ABSOLUTE, ratio=beats per minute as EDUS per RTU, unit == RTU.
"Change By": flag=TDIL_RELATIVE, ratio=%

See EDTTempo in EDATA.H for field definitions/documentation.

Score Expression

other, tag=DY, cmper = measure

There is an incident for each score expression in a particular measure

This is the base record of score expressions. Info is normally set in the score expression assignment dialog:

Measure Spec for the measure should have its meflag flagged with DYNAMBIT;

See EDTScoreExpression in EDATA.H for field definitions/documentation.

Separate Placement

other, tag=DI, cmper=measure number of score expression in staff/score

there is an incident for each separate score expression placement information in a particular measure

This record specifies separate placement info for a score expression (gives extra offset for a particular instrument. Each instrument that has separate placement will have one of these records.

See EDTSeparatePlacement in EDATA.H for field definitions/documentation.

Staff Expression

entry detail, tag=ED

There is an incident for each staff expression attached to a particular entry.

This is the base record of staff expressions. Info is normally set in the staff expression assignment dialog:

Entry with the staff expression must have its xef flagged with STAFFEXPRDETAIL;

See EDTStaffExpression in EEDDATA.H for field definitions/documentation.

Play Dump

other, tag=PD, cmper=value in text/shape expression, incident=0

This record holds arbitrary midi data for a staff/score expression with a playback type of Midi Dump (see EDTTextExpression)

See EDTPlayDump in EDATA.H for field definitions/documentation.

Text Expression

other, tag=DT, cmper=dynumber in staff/score expression

The first incident gives placement and font information for the text expression, subsequent incidences specify the character array for the text in the text expression (text is appended automatically to struct in Extension API)

This record specifies text expression related info and hangs off of a staff or score expressions.

Info is normally set in the text expression designer dialog:

See EDTTextExpression in EDATA.H for field definitions/documentation.

Shape Expression

other, tag=DO, cmper=dynumber in staff/score expression

This record specifies shape expression-related info and hangs off staff and score expressions. Info is normally set in the shape expression designer dialog.

See EDTShapeExpression in EDATA.H for field definitions/documentation.

Shape

other, tag=SD, cmper=shapedef in shape expression.

This record specifies a shape definition (used in shape expression, executable shapes, etc.) related info see shapetag.h for instruction types, required data. Info is normally constructed in the shape designer dialog.

Storage for the shape is separated into SL and SB others:
Shape Instructions (other, tag=SL, cmper=instlist in EDTShape)
Shape Data (other, tag=SB, cmper=datalist in EDTShape)

(NOTE: The Extension interface assembles all the related info into one, long variable length struct, so extensions don't use this record definition:)

```
typedef struct
{
        twobyte instlist;       /* instructions stored in multi incidences of SL others w/ this cmper */
        twobyte datalist;       /* data stored in multi incidences of SB others w/ this cmper */
        twobyte AAAA;           /* currently unused */
        twobyte BBBB;           /* currently unused */
        twobyte CCCC;           /* currently unused */
        twobyte DDDD;           /* currently unused */
} EDTRawShape;
```

Format for SL (shape instruction other)

each ot_SL holds 3 instructions, each of which is fourbytes:

shape tags are defined in shapetag.h (along with number of data items used by each tag)

```
#define revMASK         0xFF000000L     /* version info */
#define nDATAMASK       0x00FF0000L     /* number of data items for this instruction */
#define tagMASK         0x0000FFFFL     /* tag (instruction type) is here. */
```

Format for ot_SB (shape data) each ot_SB holds 3 of these.

All of the data for all of the instructions in the shape are strung together in ot_SB records:

```
typedef union                   /* data for shape, specify either f or l depending on isFloat, (some
                                things are always fourbyte, like font ids) */
{
                float f;
                fourbyte l;
} EDTPathData;
```

Text Block

other, tag=TX, cmper=text block id

text block id's are stored in Staff, Group Records; and Page and Measure-Attached Text Records

This record specifies layout info for a text block, it hangs off a staff or group record (for staff group names) or a Page or Measure Text Block.

See EDTTextBlock in EDATA.H for field definitions/documentation.

Text Blocks are stored over four incidences (the last 2 incidences are currently unused)

example

```
^TX(1) 1 0 0 0 100 0
^TX(1) 0 6664 0 0 0 0
^TX(1) 0 0 0 0 0 0
^TX(1) 0 0 0 0 0 0
^TX(2) 2 0 0 0 100 0
^TX(2) 0 6664 0 0 0 0
^TX(2) 0 0 0 0 0 0
^TX(2) 0 0 0 0 0 0
^TX(3) 3 0 0 0 100 0
^TX(3) 0 6664 0 0 0 0
^TX(3) 0 0 0 0 0 0
^TX(3) 0 0 0 0 0 0
```

Means:
Here are 3 text blocks (they must have a referencing EDTPageText, EDTMeasureText, or Staff-Spec, or GroupSpec to show up on the page)

First text block:
raw text is ^block(1), width and height are unbounded,
there is no layout shape, line height is 100% of font size
xadd, yadd are both 0 (unused since this is not a custom frame)
flags are 6664 = 0x1a08
justification is TEXT_JUSTIFY_LEFT, new postioning is true,
use word wrap, show shape(but there is no shape, so none is shown)
linedeline is a percent,
inset of text from shape is 0 (no shape),
line width of outline is 0 (no outline)

Text block 2 and 3 are the same, but reference raw text ^block(2) and ^block(3)

Page Text Block

other, tag=pT, cmper=page

There is one incident for each text block on a page.

A Page Text Block attaches a Text Block to a page or range of pages

See EDTPageText in EDATA.H for field definitions/documentation.

Page text blocks are stored over 2 incidences

example:
^others
.
.
.
^pT(1) 3 592 -904 1 1 0
^pT(1) 0 0 0 0 0 0

Means this is a text block assigned to page 1.
Text block is TX other w/cmper 3.
Offset is (592,-904) from page reference point. (top left margin)
First and last page is 1
Flags are 0, meaning text block is assigned to all pages in range,
align modes are TEXT_HALIGN_LEFT, TEXT_VALIGN_TOP, positioned from margins
No independent right page position.

Measure Text Block

detail, tag=mt, cmper=instrument, cmper2=measure,

There is one incident for each text block attached to a measure.

A Measure Text Block attaches a Text Block to a measure

See EDTMeasureText in EDATA.H for field definitions/documentation.

example:
^details
.
.
.
^mt(1,1) 1 224 -236 0 0
^mt(1,1) 2 -32 -516 0 0

Means there are two text blocks attached to measure 1, instrument 1
Measure text block 1: Text block is TX other w/cmper 1, offset is (224,-236) from (left,top) of measure
Measure text block 2: Text block is TX other w/cmper 2, offset is (-32,-516) from (left,top) of measure

Lyrics

(entry detail, tag=ve,ch,or se (for verse, chorus, or section, respectively)
(multiple incidences of each lyric type (verse/chorus/section, for each verse/section/chorus with lyric on this entry)

Entry's ef flag must have TEXTDTAIL set if it has any lyrics.

Lyrics are stored as a entry details which give a syllable offset into a raw text record that represents a whole verse/chorus or section. (See raw text documentation)

Raw text for the lyric is found in the ^lyric section of the etf, under ^verse(rawTextNum), ^chorus(rawTextNum), or ^section(rawTextNum) depending on the tag of the detail for the lyric.

See EDTLyric in EEDDATA.H for field definitions/documentation.

example:

^details
.
.
.
^ve(0,1) 1 1 0 0 0
^ve(0,1) 2 1 0 0 0
^ve(0,2) 1 2 0 0 0
^ve(0,2) 2 2 0 0 0
^ve(0,3) 1 3 0 0 0
^ve(0,3) 2 3 0 0 0
^ve(0,4) 1 4 0 0 0
^ve(0,4) 2 4 0 0 0
^ve(0,5) 1 5 0 0 0
^ve(0,6) 1 6 0 0 0

means:

for entries 1 through 4
verse 1, syllable n, and verse 2, syllable n are attached to entry n.

for entry 5 and 6:
verse 1, syllable n are attached to entry n. (no verse 2 lyrics on these entries)

There are no extra offsets or word extensions for these lyrics (last 3 values are all 0)

Page Spec

other, tag=PS, cmper=page number, inci = 0

There is exactly one Page Spec record for every page in the piece.

See EDTPageSpec in EDATA.H for field definitions/documentation.

Page specs are stored over 2 incidences.

Most of this information is directly related to fields in the page layout dialog.

example:

^others
...
^PS(1) 3168 2448 1 2
^PS(1) -144 144 144 -144 80 0

Translation: (these 2 incidences comprise one Page Spec)

Page 1 has:
height: 3168 EVPUs
width: 2448 EVPUs
stavestr: staff system 1 is first system on page
pageflag: page was resized with "hold margins" (resizing affects page contents, not physical page size)(PS_MARGSCOPING flag is set); no ossias (PS_OSSIA not set)
margTop: top margin -144 EVPUs
margLeft: left margin 144 EVPUs
margBottom: bottom margin 144 EVPUs
margRight: right margin -144 EVPUs
percent: 80% page reduction

Staff System Spec

other, tag=SS, cmper=system number

There is exactly one SSPEC record for every staff system in the piece.

See EDTStaffSystemSpec in EDATA.H for field definitions/documentation.

Staff system specs are stored over two incidences.

Most of this information is directly related to fields in the page layout dialog.

The record structure in an ETF file is slightly different from EDTStaffSystemSpec.

The ETF structure looks like this:

```
struct
{
        twobyte top;
        twobyte left;
        twobyte right;
        twobyte bottom;
        twobyte mestart;        // first measure on this system
        FLAG_16 staveflag;

        twobyte mend;           /* first measure on next system; if 0, recompute the
                                   number of measures on this system */
        twobyte horzPercentHi;  /* high word of horzPercent (by breaking fourbyte
                                   into two twobytes we don't have to worry about spe-
                                   cial storage considerations in ETF)(in hundredths of
                                   a percent) */
```

```
        twobyte horzPercentLo;      // low word of horzPercent
        twobyte ssysPercent;        // percent reduction for this system
        twobyte AAAA;               // unused (pad to 2 incidences)
        twobyte BBBB;               // unused (pad to 2 incidences)
};
```

example:

^others
...
^SS(1) -463 144 0 -199 1 1
^SS(1) 5 0 12100 75 0 0

Translation: (these 2 incidences comprise one Staff System Spec)

top: -463 EVPUs
left: 144 EVPUs
right: 0 EVPUs
bottom: -199 EVPUs
mestart: starts with measure 1
staveflag: SSPEC_HOLD_MARGINS is set
mend: next system starts with measure 5
horzPercentHi, horzPercentLo: in ETF, horzPercent is split over two words. Combining these words we get 121% (12100 hundredths). This is the amount we will stretch the measures in order to fit properly within the system.
ssysPercent: there is a 75% reduction on this system

Staff Enduction

detail, tag=LP, cmper1=staff system ID,cmper2=instrument

Staff Enduction contains information about an individual staff enlargement or reduction ("enduction") within a specified staff system.

See EDTStaffEnduction in EDATA.H for field definitions/documentation.

This is related to information in the Resize Staff dialog (Resize tool).

The SS_LINEPERC flag in EDTStaffSystemSpec must be set before this record can be processed.

example:

^details
...
^LP(1,2) 90 0 0 0 0

Translation:

This record pertains to staff system 1, instrument 2. There is a 90% reduction on this staff within this system.

Cross Staffing

(entry detail, tag=CD)

Entry's ef flag must have CROSSBIT set if it has cross-staffed notes.

See EDTCrossStaff in EEDDATA.H for field definitions/documentation.

example:
^details
.
.
.
^CD(0,1) 1 2 0 0 0

means:
note id 1 of entry one is cross-staffed to instrument 2.

Articulation

(entry detail, tag=IM)

This gives individual positioning info for the articulation on a single entry and points to a more detailed ArticulationDefinition other.

Entry needs IMRKDTAIL set in ef if it has any articulations.

See EDTArticulation in EEDDATA.H for field definitions/documentation.

example:
^details
.
.
.
^IM(0,1) 1 0 0 10 0

means:
entry 1 has an articulation centered horizontally on, and 10 evpus above the entry. The definition for the articulation is stored in incidences of an 'IX' other with comparator 1

MeasNumberRegion

other data, tag=MN, cmper=which measure number region.

specifies info for a measure number region.

Normally edited, created with Measure Number dialog.

see EDTMeasNumberRegion in edata.h for field definitions/documentation.

MeasNumberRegion are stored over 9 incidences of others.

/* cmper=which measure number region (1-based), inci = 0 */

MeasNumberSeparate

detail data, tag=MI, cmper1= inst, cmper2= meas

specifies staff-specific placement for measure numbers.

(there are 3 incidents per MeasNumberSeparatestruct)

see EDTMeasNumberSeparate in edata.h for field definitions/documentation.

MidiExpression

detail data, tag=ME, cmper1=inst, 1-based; cmper2=meas, 1-based;

There is one incident for each midi expression in the measure, incidents should be in chronological order.

specifies measure specific midi expression (continous data captured in Midi transcription or hyperscribe, or non-entry specific data created/edited in Midi Tool)

See EDTCrossStaff in EDATA.H for field definitions/documentation.

example:
^details
...
^ME(1,2) 0 192 100 0 0

Translation:
This record pertains to instrument 1, measure 2. The expression is at the beginning of the measure on this staff within this system.

Chord

entry detail data, tag=CH

This info is normally edited in the "Chord Definition" dialog box.

Learned chords use the same format, but are stored under the hC detail tag (more info below)

Entry needs CHORDBIT set in ef if it has any chords.

Multiple incidences of this record indicate multiple chords on an entry.

See EDTChord in EEDDATA.H for field definitions/documentation.

example:
^details
.
.
.
^CH(0,1) 97 99 1 0 10

means:
entry 1 has a chord whose root is on scale degree 1, it is shown and played. The alternate bass is on scale degree 3, and it is also shown and played. The suffix is stored as incidents of 'IV' others with cmper 1. The chord is centered horizontally and is 10 EPVUs above the entry.

This could be a C Major chord with an E bass in C Major.

Learned chords are detail records stored under the tag 'hC' with cmper1=root and cmper2=alternate bass and with a hashvalue stored in the posadd field.

This hashvalue encodes the distance of each note in the chord from the root. Each bit represents the number of half steps from the root. For example, a major triad has a note 4 half steps above the root (the third) and another 7 half steps above the root (the fifth). It would be encoded as 01001000 in binary, and stored in posadd

example:

^details
.
.
.
^hC(1,3) 97 99 1 72 0

means:

This is the same Chord stored as a learned chord. It is stored under it's root and alternate bass, the first 3 fields are the same as in the CH entry detail and 72 is the hash value for a major triad.

NoteheadMods

entry detail,tag=CN

Holds various note-specific custom attributes (normally edited in Special Tools.

If an entry has EDTNoteheadMods, it's ef field should be flagged w/NOTEDTAIL

See EDTNoteheadMods in EEDDATA.H for field definitions/documentation.

example:

^details
.
.
.
^CN(0,1) 1 50 0 0 0

means:

note id 1 of entry one is reduced to 50%

ChordSuffix

other,tag=IV,cmper=which chord suffix

specifies the suffix letters in a chord symbol (normally edited in the chord suffix editor. These 'others' are attached to CH (Chord) entry details and hC (Learned Chord) details.

A complete suffix is defined by a sequence of IV others (ordered by incident) under the same cmper.

See EDTChordSuffix in EDATA.H for field definitions/documentation.

example:

^others
.
.
.
^IV(1) 77 0 0 2564 0 0
^IV(1) 55 72 0 2564 0 2048

means:

'M7' suffix in 10 point, font id 4
first char is 'M' no x or y disp, font is 10 points, font id 4
first char is '7' x disp of 72 Evpus, font is 10 points, font id 4 is number is set.

ChordPlayback

other,tag=IK,cmper matches cmper of IV chord suffix others

specifies the playback for a chord suffix.

stored as an null-terminated array of twobytes, one twobyte per midi note in chord.

See EDTChordPlayback in EDATA.H for field definitions/documentation.

example:

^others
.
.
.
^IK(1) 60 64 67 0 0 0

means:

Playback for a C Major triad.

Raw Text for text blocks and lyrics.

The raw text data consists of text and formatting escapes called caret commands. The caret commands specify style info (font,size,effects), tracking, and special macros like composer, date, page number (inserts menu command in text tool). Note: All the embedded codes end with a right paren.

Caret commands and their meanings:

Text identifier caret commands:

^text: start of text block raw text (tag serves no purpose, don't rely on it being present, but do preserve it)
^block(n): raw text for a text block with id n follows (id stored in the TX other)
^lyrics: start of lyrics raw text (tag serves no purpose, don't rely on it being present, but do preserve it)
^verse(n): raw text for verse n follows. (id stored in the ve entry detail)
^chorus(n): raw text for chorus n follows. (id stored in the ch entry detail)
^section(n): raw text for section n follows. (id stored in the se entry detail)
^end: end of text block or lyric (added for Finale97)

Font style-related commands:

^font(fontname): following text should be drawn in font, fontname
^size(n): n is size of following text
^efx(plain | bold | italic | underline): style of text, styles are cumulative, plain cancels all styles.
^baseline(shift): baseline change without line ht affect, shift is in EVPUs.
^superscript(shift): baseline change affecting line ht, shift is in EVPUs.
^tracking(space): inter-letter spacing, space is in 'EMs'

File Info Dialog related insert commands:

(text for these comes from File Info Dialog)

^composer:	composer
^copyright:	copyright
^title:	title
^description:	description

Text Inserts Dialog related insert commands:

(font/size for these comes from Text Inserts Dialog)

^sharp:	insert a sharp
^flat:	insert a flat.
^natural:	insert a natural.
^dbsharp:	insert a double sharp.
^dbflat:	insert a double flat.

Misellaneous inserts:

^staffname:	embedded object (not supported yet)
^abrvstaffname:	embedded object (not supported yet)
^date(form):	insert current date, form: 0 = short form, 1= long form, 2 = abbreviated
^time(showseconds):	insert current time, form: 1 = show seconds, 0= don't show seconds
^page(n):	current page number + n

Example:

Here is the text for 3 text blocks, 2 verses of lyrics, and 2 choruses:
(NOTE: the ETF will not necessarily have new lines delimiting the various blocks/verses/etc.)

^text^block(1)^font(Times)^size(12)^efx(plain)This is the first text block. Here is the date:^date(0)^end
^block(2)^font(Times)^size(12)^efx(plain)This is the second text block. Here is the time with seconds:^time(1)^end
^block(3)^font(Times)^size(12)^efx(plain)^efx(bold)^efx(italic)This is the third text block in Times-Bold Italic^end

^lyrics^verse(1)	this is verse 1
^verse(2)	this is verse 2^end
^chorus(1)	this is chor-us 1^end
^chorus(2)	this is chor-us 2^end