Specification of the ENF Audio File Format

A format for ENF audio files

Version 2.0

HSN Kongsberg

Author Note

Qinghui.Liu@student.hbv.no, HSN Kongsberg.

Abstract

The ENF Audio File Format is a tailored file format for storing and classifying ENF (Electric Network Frequency) audio data. The raw ENF audio data in “.ENF” files is uncompressed pulse-code modulation (PCM).

Keywords: ENF Audio File Format, .ENF, PCM

Table of Contents

Specification of the ENF Audio File Format

1. Introduction

Electric Network Frequency (ENF) signal analysis has rapidly emerged in the forensic fields where the features of power-line-related signals can be used to identify the time and place in which audio recordings might have been made.

The ENF Audio File Format is specially designed for the efficient storage and classification of the ENF audio data. Compared with other audio file formats, e.g. WAV, AIFF, AU etc. "ENF" format has a tailored header that holds two unique fields: Location and Datetime which can be used to efficiently index ENF sample database.

The header of ENF files (filename extension ".enf") consists of 36 bytes which includes a <Classification> chunk and a <Data> chunk. The <Classification> chunk contains two sub-chunk: Location and Datetime. The raw sample data in ".ENF" files is uncompressed mono pulse-code modulation (PCM) which has specific sample rate, bit-width and size indicated in <Data> chunk. This illustrated in Figure 1, and Table 1.

2. Header structure of an ENF file

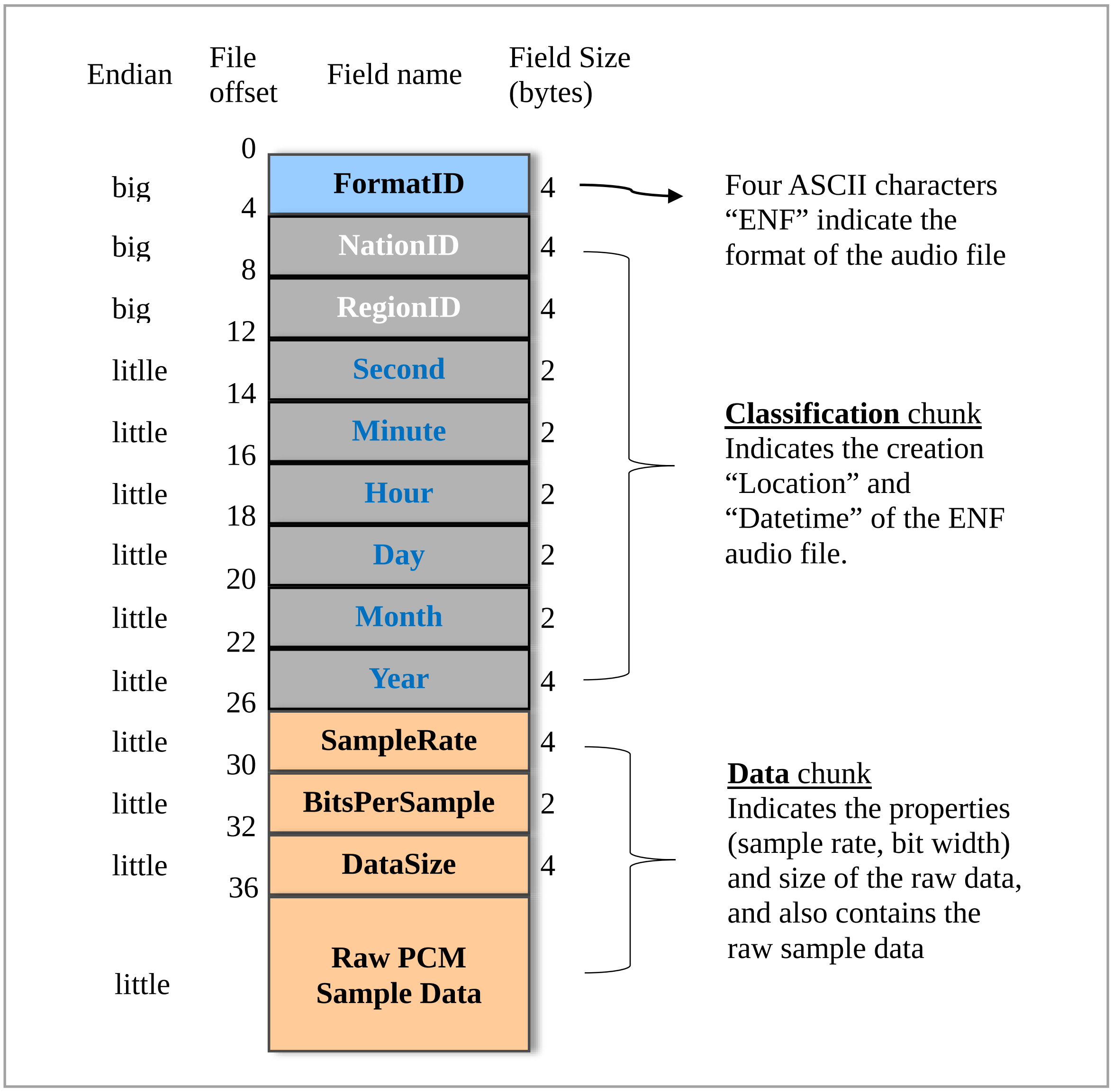


Figure 1. The "ENF" Audio File Format

2.1 The header fields description of ENF

Table 1. Fields Description

Endian	File Offset	Field Name		Field Size /bytes	Description
big	0	FormatID		4	Four ASCII "ENF" indicates the file format, the filename extension is ".enf"
big	4	*Location*	NationID	4	Up to four ASCII char, e.g. "NORW"
			RegionID	4	Up to four ASCII char, e.g. "OSLO"
little	12	*DateTime*	Second	2	Seconds of minutes from 0 ~ 59
			Minute	2	Minutes of hour from 0 ~ 59
			Hour	2	Hours of day from 0 ~ 24
			Day	2	Days of month from 1 ~ 31
			Month	2	Months of year from 1 ~ 12
			Year	4	Year since 1900
little	26	SampleRate		4	The number of samples per second, e.g. 1000, which is the number of times per second that samples are taken
little	30	BitsPerSample		2	Bit depth (Sample width), e.g. 8 or 16 means 8-bit or 16-bit depth, which determines the number of possible digital values representing each sample.
little	32	DataSize		4	Indicates total sample data size in bytes. If unknown, the value of 0xffffffff should be used. Duration time can be calculated by the formula: Duration time = DataSize * 8 / (SampleRate * BitsPerSample)
	36	Raw PCM sample data			Pulse-code modulation (PCM) is a method used to digitally represent sampled analog signals. 8-bit samples are stored as unsigned bytes, ranging from 0 to 255. 16-bit samples are stored as 2's-complement signed integers, ranging from -32768 to 32767

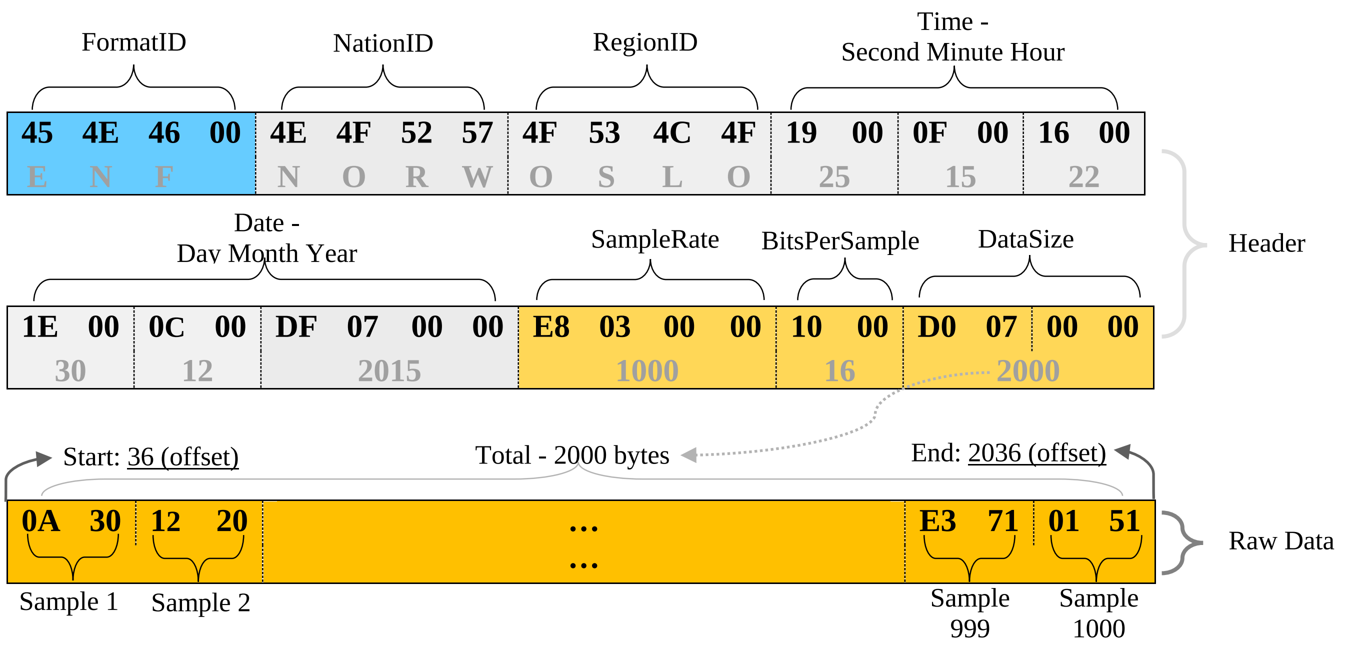
2.2 An example of ENF file

Here is an example of ENF file with bytes shown as hexadecimal numbers;

```
45 4E 46 00 4E 4F 52 57 4F 53 4C 4F 19 00 0F 00 16 00
1E 00 0C 00 DF 07 00 00 E8 03 00 00 10 00 D0 07 00 00
0A 30 12 20                ... ...                E3 71 01 51
```

An ".ENF" file

Below is the interpretation of these bytes as an ENF file



By reading the head of the ENF file, we can know that it was recorded on 2015/12/30 at 22:15:25, in Oslo of Norway with 1000Hz sample rate and 16-bit per sample. And its raw data size is total 2000 bytes, which means that its duration is only 2 seconds (1000 samples collected).

3. References

[1] EBU-TECH 3285: Specification of the Broadcast Wave Format (BWF)

[2] WAVE PCM sound file format: http://soundfile.sapp.org/doc/WaveFormat/

[3] Pulse-code modulation: https://en.wikipedia.org/wiki/Pulse-code_modulation

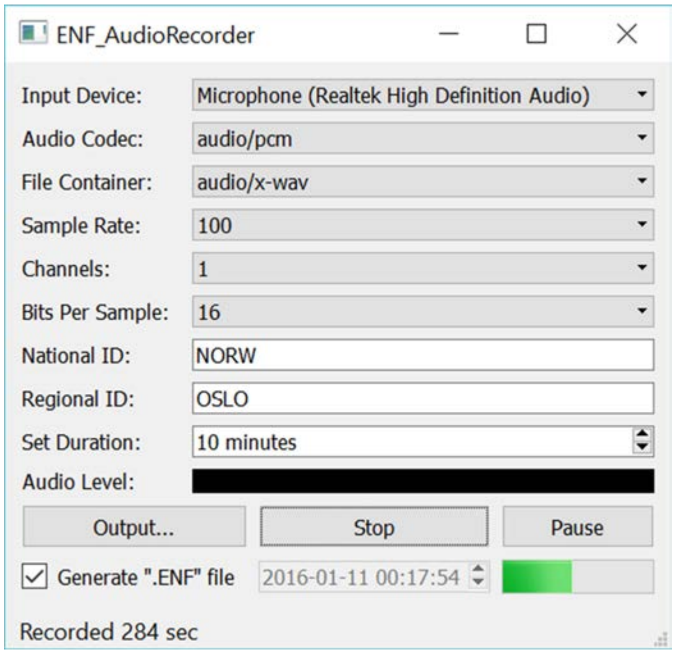
[4] Audio file format: https://en.wikipedia.org/wiki/Audio_file_format

Appendix

Appendix A: ENF Audio Recorder Application

ENF Audio Recorder is a handy tool for collecting the variation in frequency in the electric network via the audio card with an AC adapter with the correct voltage. It can assist in forensic research for determining the time of the recording.

ENF Audio Recorder is developed in Qt C++ programming language and can be run on Windows systems. Its GUI looks like as below Fig 4.1.



Input Device:	Microphone (Realtek High Definition Audio)
Audio Codec:	audio/pcm
File Container:	audio/x-wav
Sample Rate:	100
Channels:	1
Bits Per Sample:	16
National ID:	NORW
Regional ID:	OSLO
Set Duration:	10 minutes
Audio Level:	

☑ Generate ".ENF" file 2016-01-11 00:17:54

Recorded 284 sec

Fig 4.1 ENF Audio Recorder GUI

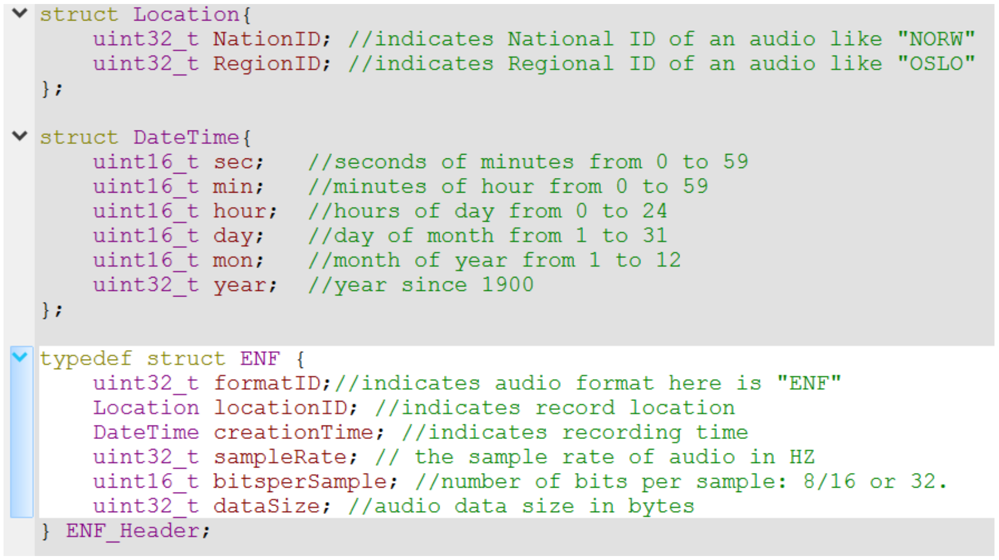
Users can select Input Device, File Container (-wav, or -raw), Sample Rate (100, 500, 1000, 8000, 11025, 16000…up to 192000), Channels (1/ 2), BitsPerSample (16), and specify the recording place (National ID, and Regional ID). And also this application support manual stop record or automatically stop by specify duration – changing the value of “Set Duration”.

The record start time shall be automatically generated once users start record (by press Record button after finish settings.). And then the green bar will appear to indicate the progress of record if user specified a duration. After finish record (by Stop button or reach Duration setting), if user clicked the option “Generate .ENF file”, the application will generate a proper ENF file with a suffix of “.enf”

The latest version download link as below:
https://www.dropbox.com/s/ugxucvc9ouel65j/ENF_Recorder_v3.rar?dl=0

Appendix B: ENF Audio File Format definition with C++

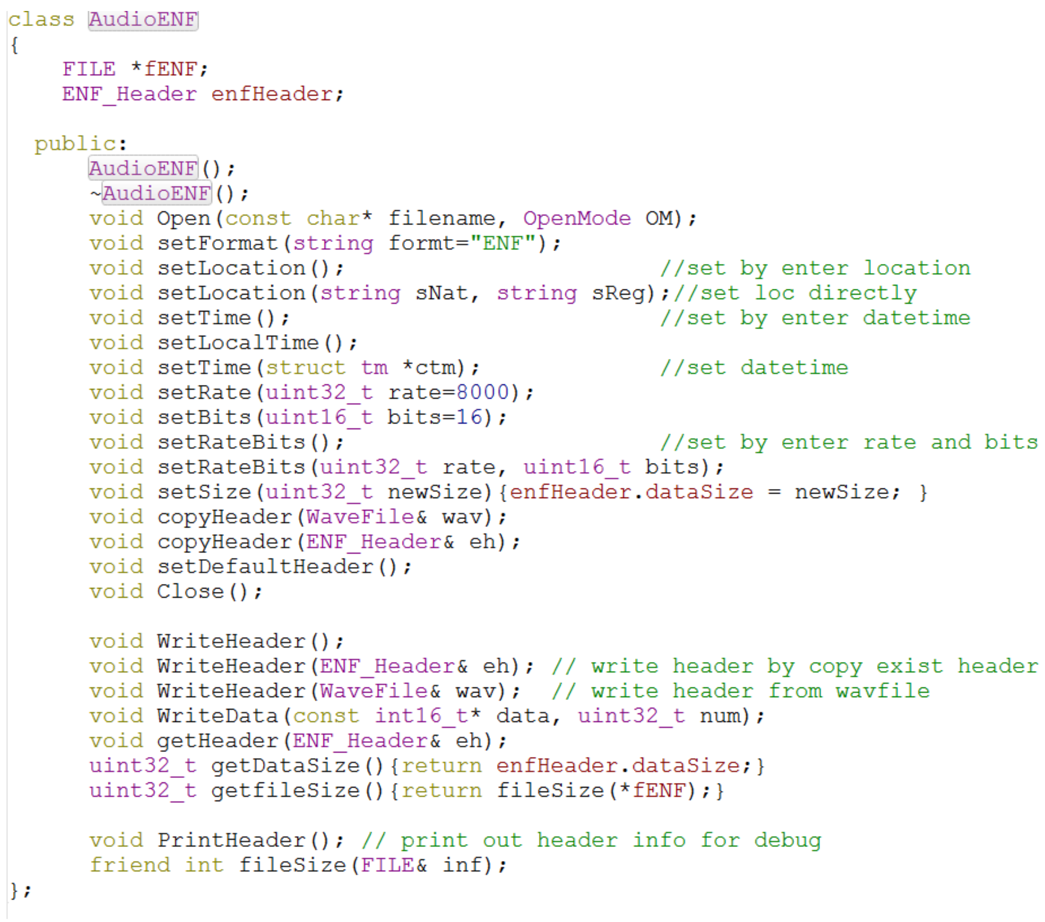
The below 2 figs show ENF Audio Format definitions with C++.



```
struct Location{
    uint32_t NationID; //indicates National ID of an audio like "NORW"
    uint32_t RegionID; //indicates Regional ID of an audio like "OSLO"
};

struct DateTime{
    uint16_t sec;   //seconds of minutes from 0 to 59
    uint16_t min;   //minutes of hour from 0 to 59
    uint16_t hour;  //hours of day from 0 to 24
    uint16_t day;   //day of month from 1 to 31
    uint16_t mon;   //month of year from 1 to 12
    uint32_t year;  //year since 1900
};

typedef struct ENF {
    uint32_t formatID;//indicates audio format here is "ENF"
    Location locationID; //indicates record location
    DateTime creationTime; //indicates recording time
    uint32_t sampleRate; // the sample rate of audio in HZ
    uint16_t bitsperSample; //number of bits per sample: 8/16 or 32.
    uint32_t dataSize; //audio data size in bytes
} ENF_Header;
```



```
class AudioENF
{
    FILE *fENF;
    ENF_Header enfHeader;

  public:
      AudioENF();
      ~AudioENF();
      void Open(const char* filename, OpenMode OM);
      void setFormat(string formt="ENF");
      void setLocation();                       //set by enter location
      void setLocation(string sNat, string sReg);//set loc directly
      void setTime();                           //set by enter datetime
      void setLocalTime();
      void setTime(struct tm *ctm);             //set datetime
      void setRate(uint32_t rate=8000);
      void setBits(uint16_t bits=16);
      void setRateBits();                       //set by enter rate and bits
      void setRateBits(uint32_t rate, uint16_t bits);
      void setSize(uint32_t newSize){enfHeader.dataSize = newSize; }
      void copyHeader(WaveFile& wav);
      void copyHeader(ENF_Header& eh);
      void setDefaultHeader();
      void Close();

      void WriteHeader();
      void WriteHeader(ENF_Header& eh); // write header by copy exist header
      void WriteHeader(WaveFile& wav);  // write header from wavfile
      void WriteData(const int16_t* data, uint32_t num);
      void getHeader(ENF_Header& eh);
      uint32_t getDataSize(){return enfHeader.dataSize;}
      uint32_t getfileSize(){return fileSize(*fENF);}

      void PrintHeader(); // print out header info for debug
      friend int fileSize(FILE& inf);
};
```