# EymAWM utilities – Version 1.x

Audio watermarking encoding and decoding command line utilities

www.metois.com

Eric Metois – 02/03/2007

The EyM Audio Watermarking tools are shareware utilities that are offered "as is". The encoding and the decoding command line utilities embed (resp. retrieve) a short data payload within (resp. from) 16-bit PCM audio wav files. This shareware is intended to target small to medium size applications that may require an audio watermark. Although proof of ownership, tracking and copy deterrence might be the most popular applications of digital watermarks this shareware provides complete control over the nature of the data you might decide to embed in an audio recording. From the utilities' point of view your payload is just a series of byte values.



# Installation

The Eym Audio Watermarking command line utilities are standalone Win32 applications with no external dependencies and they require no particular installation other than being copied on your drive. The entire system consists of the following files:

- **eymAWMenc.exe**: the command line watermark encoder
- **eymAWMdec.exe**: the command line watermark decoder
- **eymAWMlic.txt**: your license file
- **eymAWMhelp.pdf**: this present document.

Both the encoder and the decoder utilities will expect your license file to live in the current directory. If the file is either not found or the key it contains is not valid, these utilities will prompt you so and proceed to perform in their demo mode.

Even in their demo mode, these utilities should offer enough flexibility to perform a variety of performance tests so that you may assess whether or not they will work for your application before purchasing any license.

***License key purchase:*** You may purchase a license key online securely via a trusted third-party e-Seller following a link you will find on http://www.metois.com/Eymwatermark/main_eymawmkey.htm

***License key installation:*** If you've purchased a license key, simply open your **eymAWMlic.txt** file and copy and paste your key inside. This text file should contain nothing else but your license key.

# EymAWM encoder

## Usage

```
eymAWMenc.exe [options] <infile> <hexPayload> <outfile>
```

- <infile>: input wav file. This file must be a 16-bit PCM wav file. The number of channels can be arbitrary and the sampling rate must be one of the following: 48 kHz, 44.1 kHz, 32 kHz, 24 kHz, or 22.05 kHz.
- <hexPayload>: hexadecimal user payload. This payload should be provided as a string of {'0' – '9', 'A' – 'F'} (**upper case**) characters. The actual payload that will be embedded in the audio file will have an integer number of bytes (i.e. an even number of hexadecimal digits) and it will be padded with an additional x0 if you provide an odd number of hexadecimals. If the size of the payload you enter exceeds the limit of your license, the "extra" portion of your payload will be replaced with xFF bytes. When running in demo mode (i.e. without a license key) this entire payload will be replaced by a matching number xF values.
- <outfile>: output wav file. The utility will create this watermarked audio file using the same format as the input wav file you provided. Note that any optional metadata that might have been conveyed in the input file (such as via a broadcast extension chunk for instance) will effectively be ignored and striped from the output WAVE file.
- [options]: Eym Audio Watermark encoder configuration parameters. These consist in (tag, value) pairs where the tags are **case-sensitive** and maybe any of the following:
    - "-layer": "public" for a public mark or "private" for a mark only you can write. When running in demo mode (i.e. without a license key), only the public layer will be available. These values are **case-sensitive**.
    - "-gain": watermark encoding gain (float). This value controls the scaling ot the mark embedded into the audio stream. The larger the value and the more robust the mark might be, the samller the value and the less audible the mark might be.
    - "-addts": time stamp toggle (1: add time stamp mark; 0: don't). This bonus feature adds a 12-bit time code to the user payload. This time code is incremented in 15 seconds interval and it is embedded in a way that shouldn't affect the recovery rate of the user payload. On the decoding side, there is no guarantee that this time stamp will be successfully recovered for every instance of the user payload that is found.

  Default configuration: "-layer public -gain 1.0 -addts 0"

## Output

As it proceeds, the eymAWMenc.exe utility provides some progress feedback on "stderr". Once it's done is outputs a short summary on "stdout", consisting of the original wav file's name, the actual user payload that was embedded, the name of the watermarked file that was created, the number of times the embedding was repeated throughout the file and the effective speed at which this process took place.

Here is a sample summary you might get:

```
"original.wav" + (1122334455667788) => "watermarked.wav"
-> 284 packets encoded at an average of 29.3x
```

**Hint:** If you are running a batch of numerous watermarking processes it might be useful to capture that summary in a text file by piping stdout into it. For instance:

```
FOR %%i IN (Orig\*.wav) DO eymAWMenc %%i 1FA35D WM\%%~ni.wav >> encOut.txt
```

## Return codes

Upon termination the command line application returns one of the following error return codes:

| *ERC* | *Description* |
|---|---|
| 0 | SUCCESS - The watermarking process did not generate any error or warning. |
| 1 | BAD_ARGUMENTS - The command line syntax is incorrect. The proper usage is displayed on stderr. |
| 2 | BAD_LICENSE - No valid license key was found in a local "eymAWMlic.txt" file. This is not an error per se as the utility proceeded with the watermarking in its demo mode (i.e. replacing the payload with xFF byte values and using the public layer) |
| 3 | MEMALLOC_FAILURE - The utility failed to allocate a sufficient amount of memory in order to perform its task. |
| 4 | INTERNAL_ERR - An error was generated within the audio watermarking core. The cause for this could be a badly formed option string on the command line interface, a recognized but not supported audio file format, or a shortage of memory. |
| 10 | WAVEIN_FILE_ERR - The specified input audio file either does not exist or it is not a 16-bit WAVE file. |
| 11 | WAVEIN_NOT_SUPPORTED – The input was recognized as a 16-bit PCM WAVE file but its format (most likely the sampling rate) is not supported by the watermark encoder. |
| 20 | WAVEOUT_FILE_ERR - The system failed to create the specified output file on your system. |

## Examples

```
eymAWMenc.exe original.wav 1FA35D marked.wav
```

Uses default configuration (i.e. public layer, encoding gain 1.0 and no time stamp) to mark "original.wav" with 3 bytes.

```
eymAWMenc.exe –gain 0.8 –layer private original.wav 1FA35D marked.wav
```

Uses the user's private layer and a slightly decreased gain (0.8 instead of the default 1.0) to mark "original.wav" with 3 bytes.

```
eymAWMenc.exe –addts 1 –layer private original.wav 1FA35D marked.wav
```

Uses the user's private layer and the default gain to mark "original.wav" with 3 bytes along with a 12-bit time stamp.

# EymAWM decoder

## Usage

```
eymAWMdec.exe [options] <infile>
```

- <infile>: input wav file you'd like to retrieve watermarks from. This file must be a 16-bit PCM wav file. The number of channels can be arbitrary and the sampling rate must be one of the following: 48 kHz, 44.1 kHz, 32 kHz, 24 kHz, 22.05 kHz, 16 kHz or 11.025 kHz.
- [options]: Eym Audio Watermark decoder configuration parameters. These consist in (tag, value) pairs where the tags are **case-sensitive** and maybe any of the following:
    - "-layer": "public" for a public mark, "private" for your own private mark, or alternatively this could be another private layer code that was given to you by someone else. When running in demo mode (i.e. without a license key), only the public layer will be available. These values are **case-sensitive**.

  Default configuration: "-layer public"

## Sharing your private layer code

Depending on your application you may want to allow others to read the contents of a private watermark you created while remaining the only one who can actually write watermark on your private layer.

In order to do this you will need to share your private layer code. It provides enough information for a user to read your mark but not enough to allow anyone to write a watermark on your private layer.

Your private layer code is the center subset of your license key.

For instance if your full license key is **`EYMAWM4-234JOHNSMITH1-CICHGFGM`** then your private layer code would be **`234JOHNSMITH1`**

***Do not openly share your personal license key. That would also allow anyone else to forge any private watermark you may create.***

## Output

As it proceeds, the eymAWMdec.exe utility provides some progress feedback on "stderr". This feedback will echo every watermark it finds as it goes through the input wav file. If the marks included optional time stamps, these will also be shown when they are detected. Once it's done is outputs a short summary on "stdout", consisting of the wav name of the wav file that was analyzed, the number of data packets that were found and the effective speed at which this process took place.

Here is a sample summary you might get:

```
"watermarked.wav"
(x1122334455667788) - 132 instance(s) - (T 00:15) through (T 03:00)
-> 132 packet(s) decoded at an average of 48.8x
```

**Hint:** If you are running a batch of numerous decoding processes it might be useful to capture that summary in a text file by piping stdout into it. For instance:

```
FOR %%i IN (Orig\*.wav) DO eymAWMdec %%i >> decOut.txt
```

## Return codes

Upon termination the command line application returns one of the following error return codes:

| *ERC* | *Description* |
|---|---|
| 0 | SUCCESS - The watermark decoding process did not generate any error or warning. |
| 1 | BAD_ARGUMENTS - The command line syntax is incorrect. The proper usage is displayed on stderr. |
| 2 | BAD_LICENSE - No valid license key was found in a local "eymAWMlic.txt" file. This is not an error per se as the utility proceeded with the watermark decoding in its demo mode (i.e. restricted to using the public layer) |
| 3 | MEMALLOC_FAILURE - The utility failed to allocate a sufficient amount of memory in order to perform its task. |
| 4 | INTERNAL_ERR - An error was generated within the audio watermarking core. The cause for this could be a badly formed option string on the command line interface, a recognized but not supported audio file format, or a shortage of memory. |
| 10 | WAVEIN_FILE_ERR - The specified input audio file either does not exist or it is not a 16-bit WAVE file. |
| 11 | WAVEIN_NOT_SUPPORTED – The input was recognized as a 16-bit PCM WAVE file but its format (most likely the sampling rate) is not supported by the watermark decoder. |

## Examples

```
eymAWMdec.exe marked.wav
```

This command uses the default configuration (i.e. public layer) to retrieve public watermarks from the wav file "marked.wav".

```
eymAWMdec.exe –layer private marked.wav
```

This command retrieves your own private watermarks from "marked.wav".

```
eymAWMdec.exe –layer 234JOHNSMITH1 marked.wav
```

This command retrieves John Smith's private watermarks from "marked.wav". In this instance it is assumed that John gave you his private layer code, consisting of the string "234JOHNSMITH1". This will not give you enough credential to forge one of his private marks but it will be enough for you to read them.