KANTAR

Snap Live Embedder SDK

User Guide

Product version: 8.1

Issue Date: 18/12/2020

Table of Contents

1 Introduction

The SNAP Live Embbeder SDK is dedicated to SNAP watermarking encoding into linear feed, typically for a live TV channel or a live Radio station. The SNAP watermarking is dedicated to the audience measurement of Radio and/or TV but can also be used in parallel for the synchronization of second screen application.

This SDK aims at being integrated into third party products, typically into Channel in a Box solution. The SDK can support 2 types of integration. The first one is an "**offline**" integration which consists to integrate the SDK on a standalone hosting platform where watermarking license are tied to the hardware of the hosting platform. A soon as we speak about Virtual or Cloud environment, the "offline" integration is not possible anymore because you cannot rely on a well-identified hardware. Thus, there is a second kind of integration so called "**online**" where license authorization and information are requested to an online Kantar server.

1.1 SNAP Technology overview

The SNAP technology is an audio watermarking technique based on phase modulation: audio signal phase is manipulated to embed inaudible and robust information. This quite innovative approach has proved to give very good results in being able to encode large amounts of information, while preserving robustness to signal transformation as well as imperceptibility to meet requirements of most demanding customers and applications.

The SNAP audio watermarking technology is designed to carry data which address two applications at the same time. The first one is the portable audience measurement for TV and/or Radio. The second one is the synchronization of 2nd screen application.

The audio watermarking technology can be described by 3 main properties which are the inaudibility of the watermark, the amount of data carried by the watermark and the robustness of the watermark.

The new SNAP technology maintains inaudibility properties designed for the Radio audience measurement application.

Regarding the amount of data, the SNAP technology proposes one payload for Channel and Content identification. The structure of the watermark is split into 2 parts: The first one is dedicated to the identifier of the TV channel or the Radio Channel and the second part is dedicated to a timestamp. When the watermarking is embedded in linear content, this timestamp represents the UTC airing time; when it is applied on nonlinear content, the timestamp represents the relative timecode within the content asset.

Kantar proposes a unique watermark payload both for linear audience measurement (TV or Radio) and for non-linear contents (Radio Podcast, Catch-up TV) and for the synchronization of 2nd second screen applications.

This structure is made of 2 parts; the first part carries the value of an identifier and the second part carries timestamp:

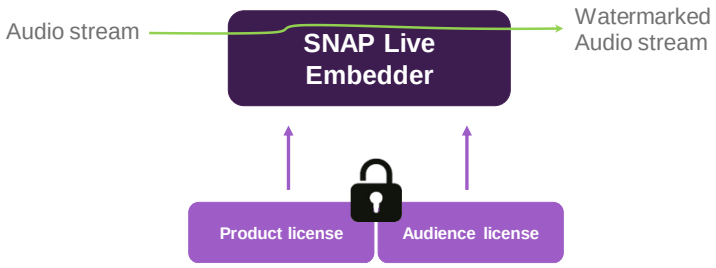
- Identifier part (16 bits in 4,096 seconds): this identifier allows 65536 values to be shared between TV channels, Catch-up TV platforms, Radio, Podcast platforms.
- Timestamp part (20 bits in 4,096 seconds): The timestamp is a number of 8,192 second intervals since the first January 2006. This counter is encoded on 20bits which means a depth of 99 days, 10 hours, 5 minutes, 34 seconds, 592 milliseconds.

1.2 Software architecture overview

1.2.1 General software architecture overview

The Audio Watermarking Embedding library is a software component dedicated to the embedding of the audio watermark for audience measurement, and to be integrated in third party products based on a PC platform. The library is a dynamic library, performing the embedding of watermark made of the channel ID and a timestamp. The value of the channel ID is selected among a list of authorized channel ID carried by the audience license file and the timestamp is the UTC time taken from the server system time.
The library implements an embedding process that needs to be fed by the raw audio samples at 48Khz.
This library does not handle the acquisition and the restitution of the audio from the PC capture card.

The following figure describes the main interfaces of the Audio Watermarking Embedding library:



1.2.2 Audience measurement licenses, channels names and channels identifiers

As said previously, audience measurement licenses can contain several channels name that manage themselves several channels IDs. The SDK manages automatically the usage of channels IDs to watermark content and the SDK functions handle directly Channels names. Consequently, the management of channels IDs is “hidden” to the SDK integrator who will handle Channel names contained in audience measurement licenses.

Example of audience measurement structure regarding channels names and channels Ids.

The license contains 3 Channels names (channel 1, channel2, channel3) and each channel name contains several channels identifiers.

Channels names
- channel 1
  - channel identifiers : 27116, 20207, 20209,20208, 20201, 20202
- channel 2
  - channel identifiers : 27216, 20217
- channel 3
  - channel identifiers : 20227, 27115, 20219, 20220, 20221, 20222, 20223 , 20224, 20225, 20226

1.2.3 Time management and synchronization

The audio watermarking is made of 2 parts. The first part is an ID identifying the TV channel and the second part is a timestamp identifying the airing time. This timestamp is based on the system clock of the hosting platform. It requires that the platform is synchronized with the time of the playout thanks to an external time clock like LTC or NTP.

The video server shall ensure the synchronization of the server with the master time clock of the playout center. The Kantar embedding library always uses the UTC time derived from the system time.

1.2.4 Log management

The SDK does not manage any log file in order to avoid any issue with the main application and the management of the space on the hard drive of the hosting platform.
Nevertheless, the SNAP Embedding SDK will generate some events using a dedicated callback function. All those events shall be caught by the main application and written in the applicative log file by the main

application. The objective is to be able to know if the output stream is watermarked or not at any time. If one day, the audience measurement operator reports that there is no watermark anymore on a TV channel, our technical support will ask the log files to the broadcaster to check if the video server was watermarking at a given date and time.

1.2.5 Audio formats

The watermarking SDK supports only uncompressed audio at 48Khz. If the main application wants to offer the ability to watermark other formats, it means that the audio shall be decoded first, watermarked and re-encoded by the server before being streamed.
This is a real use case with the Dolby-E format.

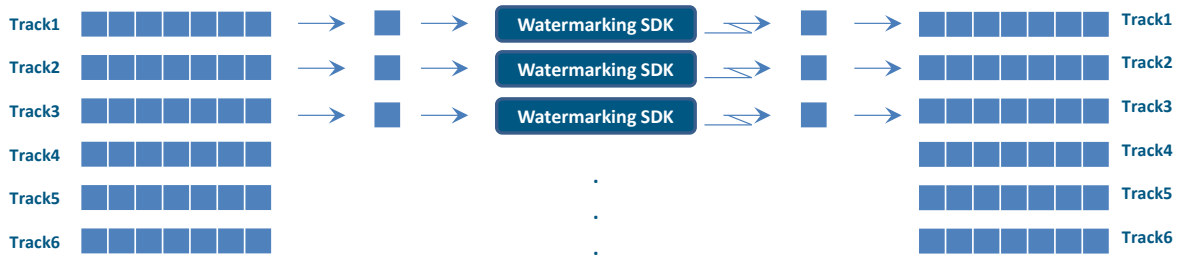
1.2.6 Audio configuration

The embedding SDK can be configured in 3 audio input configurations. The configuration can be Mono, Stereo or Multi-Channel. If you know by advance the configuration of the audio channel, you have to configure the SDK accordingly. If you don't know the configuration of the audio channel to be watermarked, then you have to select the Mono configuration.
For instance, 2 mono channels can be watermarked independently and next recombined as a stereo channel. The only requirement is that the 2 mono channels have to stay synchronized.

1.2.6.1 Mono

The Mono configuration is the simplest. One watermarking embedder shall be instantiated per mono audio track. The audio samples are provided sequentially to the watermarking embedder. As a guideline, Kantar, in its own products, provides the audio samples by packets of 1920 samples to the watermarking embedder.

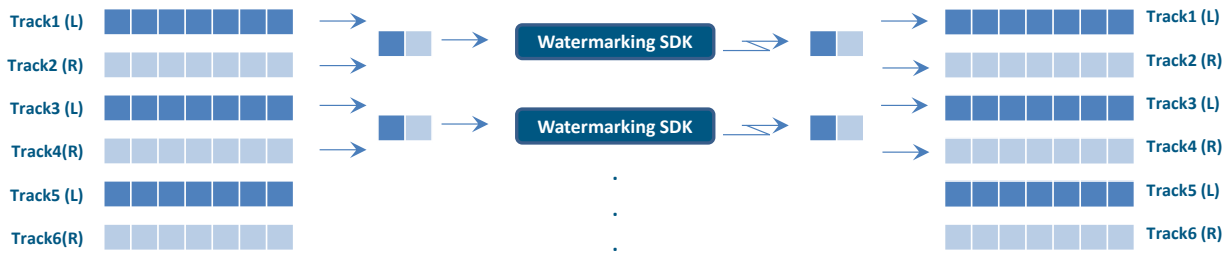
The following figure illustrates the audio sample management with the watermarking embedder for the audio Mono channels:



1.2.6.2 Stereo

When stereo audio channels have to be watermarked, one watermarking embedder shall be instantiated per stereo audio track. The audio samples are to be provided interlaced sample per sample with right and left channels to the watermarking embedder. As a guideline, Kantar, in its own product, provides with the audio samples by packets of 3840 interlaced samples to the watermarking embedder (L/R/L/R/L/R/...).

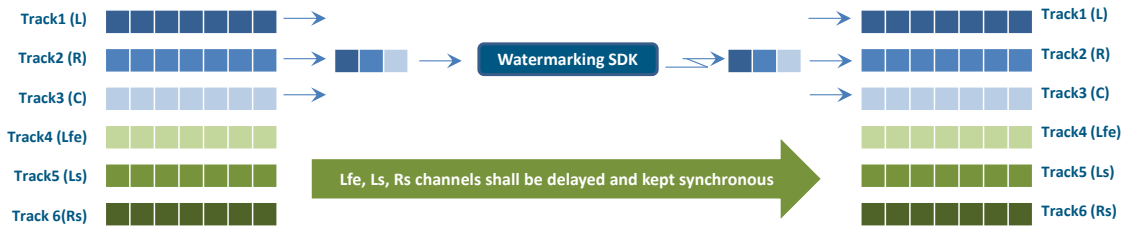
The following figure illustrates the audio sample management with the watermarking embedder for the audio Stereo channels:



1.2.6.3 Multi-Channel

When an audio Multi-channel has to be watermarked, one watermarking embedder shall be instantiated for a 5.1 or 7.1 audio track. The audio samples are to be provided interlaced sample per sample with Right, Left and Center channels to the watermarking embedder. The audio channels, Lfe, Ls and Rs have to be delayed and kept synchronous with the L, R C audio channels.
As a guideline, Kantar, in its own product, provides with the audio samples by packets of 5760 interlaced samples to the watermarking embedder (L/R/C/L/R/C/L/R/C/…)

The following figure illustrates the audio sample management with the watermarking embedder for the audio Stereo channels:



1.2.7 Audio/Video synchronization

The watermarking SDK does not handle the synchronization between the audio and the video. It does not handle the synchronization between the audio streams being watermark and the other delayed outside the watermarking library. This is the case when you have a 5.1 stream to watermark where only the Left, Right and Center channels have to be watermarked, the rest being delayed outside the library.

By the way, the library will offer the capability to link any metadata to each audio buffer structure using a “void *” pointer. This mechanism is one possible solution to keep any contextual information of the audio stream.

1.2.8 Latency on audio introduced by watermarking processing

Watermarking processing will introduce a constant audio delay of 2560 audio samples, which is **53ms at 48kHz**.

1.2.9 Watermarking value selection

The watermark identifier to be inserted in the audio channels shall to be unique for a given TV channel. A Video server will be able to play multiple SDI stream for multiple TV channels or can be used as a spare for several TV channels. It means that for each video server, a list of possible watermarks identifier will be defined in a specific audience measurement license. This license is provided to the broadcasters by the Audience Measurement Operator.
The SDK offers a specific function that will list the possible TV channels to be watermarked. The end user shall be able to select the Channel ID among in this list.

1.3 Certification procedure

The watermarking technology is the basis of the TV audience measurement and then crucial for the audience measurement operator in charge of delivering the audience measurement figures. Kantar imposes a certification of any product integrating the Kantar watermarking technology for audience measurement in order to ensure that the watermark is correctly applied on the audio streams to be aired.

A certification process shall be passed for **EACH new version** of the product integrating the Embedding SDK.

To limit the extra workload related to this certification process, Kantar offers a dedicated tool, the QC File Detector and a light paper procedure. The integrator of the watermarking SDK is then autonomous to perform the certification himself. This certification shall be registered to Kantar.

Kantar recommends doing the integration of the watermarking SDK in an independent software module with its own versioning. This allows the release a new product release without having to redo the watermarking certification if nothing changes regarding the watermarking module.

Ideally certification procedure is automated in the unit testing pipeline of the product. That way, only the registration form must be sent to Kantar.

To have a complete description of the Kantar certification procedure and on the QC File detector tool, please refer to the following documents:

- UG_Certification_Procedure_for_SDK.pdf
- UG_SNAP_File_QC_Detector.pdf

1.4 Offline and Online mode for license Information

The SDK behaves in the same way for the 2 types of integration. The main differences are during the initialization phase to get the authorization and some initial settings.

1.4.1 Offline mode for SDK running of physical hardware platform

For an offline integration, the SDK needs 2 kinds of license. The first one is the Kantar license and the second one is the Audience license. Kantar will deliver the 2 kinds of license for the evaluation or for the integration of the SDK on a development platform.

At the end of the integration, **a certification procedure** shall be passed and registered by Kantar.
Kantar will provide the Kantar license only for the final and **certified** product versions, and the Audience Measurement operator will deliver the audience license.

1.4.1.1 Kantar license

Kantar watermarking uses the concept of a “watermark key”, which is specific to each country. This key is unique so that each audience measurement operator is only able to embed/detect its own watermark. This key is included in the license delivered by Kantar. The license enables execution of the Embedder Library. This license is checked at run-time by the library. If you do not already have this Kantar license, please ask Kantar support team to generate one.

1.4.1.2 Audience license

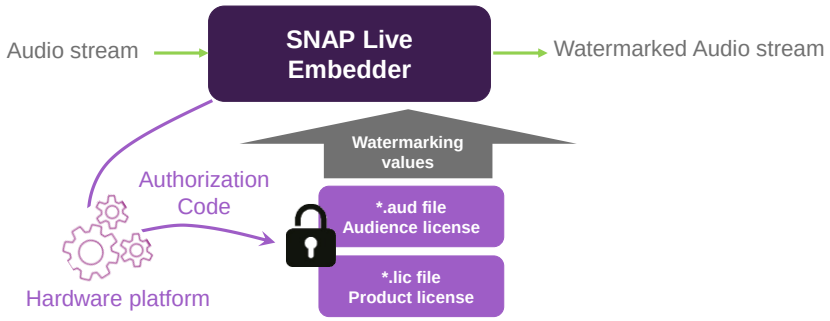
The value of the watermarking identifier shall be different for each server and linked to a specific TV or radio channel. The attribution and the association of the watermark identifier to a TV or radio channel are made by the audience measurement operator thanks to the audience license file. On the final and certified product versions, the Audience Measurement operator will deliver the audience license.

1.4.1.3 Locking mechanism with offline mode

Kantar offers 2 locking modes for license files. The locking mode depends on the platform where the SDK will be executed.

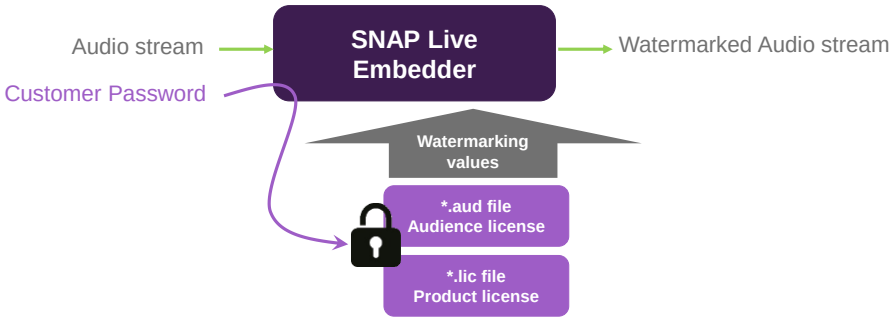
When SDK is integrated on a **physical platform** (opposed to virtual environment), the license files will be locked on the signatures of the hardware platform. This signature (called Authorization code) is generated thanks to a Kantar tool delivered with the SDK. At runtime, the SDK calculate the authorization code and compare it to the authorization code used to generate the license files. If it matches, the licenses are checked, otherwise an error code is generated and SDK stop working.

The following diagram illustrates how the licenses are tied to the hardware platform:



When the SDK is integrated in a **private virtual environment** without access to Internet, the license file will be locked on password delivered by Kantar to the integrator. At runtime, the SDK compare the integrator password to the password used to generate the license files. If it matches, the licenses are checked, otherwise an error code is generated and SDK stop working.

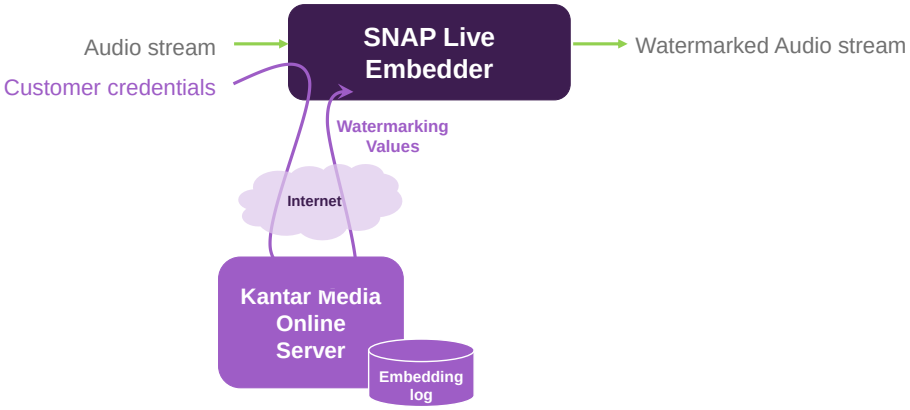
The following diagram illustrates how the licenses are locked thanks to a password:



1.4.2 Online mode for SDK running on Virtual or Cloud environment

For an online integration, some credentials are submitted by the SDK to the Kantar server. Once authentication phase is successfully passed, then the SDK can be initialized directly by usual known parameters like the license ID and the channel name. If those parameters are unknown, they can be listed by asking to the Kantar server thanks to specific SDK functions.

The following diagram illustrates the online mode for the Embedder SDK:



2 Quick start

2.1 Introduction

This chapter aims to provide quick “step by step” instructions for executing the Sample Application delivered in the Live SNAP Embedder SDK.
This Sample Application is located in the bin folder.

2.2 Prerequisite

Make sure that the machine on which the sample is running is compliant with the platform requirements described in the Package Description / Requirements chapter.

Extra steps for online embedding on CentOS:
Open a terminal as root and first check that ca-bundle.crt exists and is already a symlink:

```
ls -la /etc/ssl/certs/ca-bundle.crt
```

then create a copy of the symlink:

```
cp -av /etc/ssl/certs/ca-bundle.crt /etc/ssl/certs/ca-certificates.crt
```

2.3 Installation

For Windows

- Execute setupSnapLiveAudioEmbedderSDK.XXX.exe (with XXX replaced by the Embedder SDK version).
- Default installation path is:
    - C:\Program Files (x86)\Kantar Media\ NexTracker Snap Live Embedder SDK

For Linux

- Extract files of setupSnapLiveAudioEmbedderSDK.XXX.x86_64.tar.gz archive (with XXX replaced by the Embedder SDK version).

Get licenses files (see 2.4 for offline and 2.5 for online).

2.4 Retrieve a License for Offline SDK

- Go to NexTracker Snap Live Audio Embedder SDK folder.
- Retrieve your authorization code
    - On Linux, run on a terminal: “sh GenerateAuthorizationCode.sh”. This will create an AuthorizationCode.txt file.
    - On Windows, the AuthorizationCode.txt file is generated during installation.
- Send to Kantar the AuthorizationCode.txt file and our support will send back to you the license file: license.lic
- Send to the AuthorizationCode.txt file to the audience measurement operator and their support will send back to you the license file: license.aud
- Copy paste these two files (license.lic and license.aud) in the bin folder.

N.B: for Windows 7, Windows 10, you will have to change User Account Control to “Never notify” in order to have the software working as expected. To do this:

- Click on start in the search program enter “rights”.
- Click on “Change user control settings”.
- And put it in never notify.
- Restart your computer.

The software license (license.lic) can be set with a **time limitation**. In this case, the software check periodically the license validity:

- At startup, if the license is expired, the software can't start.
- During processing, if the license is expired, the software bypass the signal. No watermarking is performed.
- A dedicated GetLicenseRemainingDays method returns number of remaining days before license expiration

2.5 Retrieve a License for Online SDK

The online sample application needs a login and password for retrieving all licenses online thanks to specific methods described after.

2.6 Launching the Sample Application

The 2 samples applications are in the bin folder:

- SampleSnapLiveAudioEmbedderSDK
- SampleOnlineSnapLiveAudioEmbedderSDK

For launching it:

- On Linux open a terminal, on Windows open a command window with administrator privileges.
- Go to the bin folder
- Run the following command:
  - Offline:
    - On Windows:
      - SampleSnapLiveAudioEmbedderSDK.exe input.wav output.wav
    - On Linux:
      - ./SampleSnapLiveAudioEmbedderSDK input.wav output.wav
  - Online
    - On windows:
      - SampleOnlineSnapLiveAudioEmbedderSDK.exe input.wav output.wav –login="login" –password="password" --metadataPath=SampleOnlineSnapLiveMetadata.json
    - On Linux:
      - ./SampleOnlineSnapLiveAudioEmbedderSDK input.wav output.wav –login="login" –password="password" --metadataPath=SampleOnlineSnapLiveMetadata.json

The Sample Application should display the following information:

- SDK version.
- Input file frequency, number of audio channels and number of bits per sample.
- The channel names contained in the audio license. Note: The sample offline selects the first channel while the sample online selects channel name defined in SampleOnlineSnapLiveMetadata.json

Sample offline Snap Live SDK execution display:

```
Snap Live Embedding SDK version:XXX
Input file:input.wav frequency:48000 channels:2 bits per sample:16
License remaining days: -1
[INFO] - Using channel name: tv1
[INFO] - Embedder Initialization starting ...
[INFO] - Start watermarking using ID 41 (SNAP)
[INFO] - SIMD level used: AVX2
[INFO] - Applying CPU affinity mask - All CPU(s) selected
[INFO] - Embedder Initialization finished
```

```
[INFO] - Resynchronizing timecode
[INFO] - Flushing remaining audio data from embedder ...
[INFO] - Embedder uninitialization starting ...
[INFO] - Embedder uninitialization finished
[SUCCESS]
```

Sample online Snap Live SDK execution display:

```
Snap Live Embedding SDK version:XXX
Input file:input.wav frequency:48000 channels:2 bits per sample:16
[INFO] - Connection to license.kantarmedia.com:443
Get license 'name - id' list from Kantar server
    sdk_live_snap - 25
Channel 'name - id' found in audio license:
    tv1 - 1
    tv2 - 2
[INFO] - Embedder Initialization starting
[INFO] - Using channel name: tv2
[INFO] - Start watermarking using ID 2 (SNAP)
[INFO] - SIMD level used: AVX2
[INFO] - Applying CPU affinity mask - All CPU(s) selected
[INFO] - Embedder Initialization finished
[INFO] - Resynchronizing timecode
[INFO] - Flushing remaining audio data from embedder
[INFO] - Embedder uninitialization starting
[INFO] - Embedder uninitialization finished
[SUCCESS]
```

Running the Sample Application is a good way to check the compatibility of your environment.
In case of problem, Kantar can provide some support for troubleshooting the issue.

2.7 Using pipe commands

It is possible to use pipe commands with the embedder.
This feature is only available for linux.

1. Check Version

First, check that your SDK version supports "stdin" and "stdout" labels:

```
$ ./SampleSnapLiveAudioEmbedderSDK -h

usage: SampleSnapLiveAudioEmbedderSDK inputFile outputFile
with
```

```
  inputFile                              : .wav input filename. Set 'stdin' value to
read audio data from standard input (raw pcm).
  outputFile                             : .wav output filename. Set 'stdout' value to
write audio data to standard output (raw pcm).
  {--product-license-path=value}         : Path to Kantar product license (optional).
Default: in 'SampleSnapLiveAudioEmbedderSDK' executable directory.
  {--audience-license-path=value}        : Path to Kantar audience license (optional).
Default: in 'SampleSnapLiveAudioEmbedderSDK' sample executable directory.
  {--channelName=channel_name}           : Channel name used to select id in audio
license (optional).
  {--affinity=affinity_mask}             : CPU affinity mask (optional).
  {--prio=thread_policy,thread_priority} : Thread priority to use for the given policy
(optional).
  {--record=directory_path}              : Path where record wav files are stored
(optional).
  {--silent}                             : Silent mode. No message on stdout
(optional).
if inputFile value is 'stdin':
  --stdin-sampling-rate=value            : input sample rate (Hz). Ex: 48000.
  --stdin-nb-channels=value              : input channels count. Ex 2 for stereo, 1
for mono.
  --stdin-bits-per-sample=value          : input bit per sample. Ex: 16 or 24.
  --stdin-endianness=value               : input endianness. values: le (little
endian) or be (big endian)
```

2. Check license

Then, check that your Kantar licenses are valid ('license.lic' and 'license.aud'). To test it, start a simple watermarking job on uncompressed wav file:

```
$ ./SampleSnapLiveAudioEmbedderSDK song.wav song.wmk.wav

Snap Live Embedding SDK version:8.0
Input file:song.wav frequency:48000 channels:1 bits per sample:24
License remaining days: -1
Channel 'name - id' found in audio license:
        BBC_RADIO_1 - 2001
        BBC_RADIO_2 - 2002
        BBC_RADIO_3 - 2003
[INFO] - Using channel name: BBC_RADIO_1
[INFO] - Embedder Initialization starting ...
[INFO] - Start watermarking using ID 2001 (SNAP)
[INFO] - SIMD level used: AVX
[INFO] - Applying CPU affinity mask - All CPU(s) selected
[INFO] - Embedder Initialization finished
[INFO] - Resynchronizing timecode
[INFO] - Flushing remaining audio data from embedder ...
[INFO] - Embedder uninitialization starting ...
[INFO] - Embedder uninitialization finished
[SUCCESS]
```

3. Check input pipe

To test input pipe, pipe a raw pcm stream to watermarking process. Note: watermark sample uses "stdin" label as input file:

```
$ ffmpeg -loglevel quiet -re -i song.wav -ac 2 -ar 48000 -f s16le -acodec pcm_s16le -
    | ./SampleSnapLiveAudioEmbedderSDK stdin song.wmk.wav --stdin-sampling-rate=48000
--stdin-nb-channels=2 --stdin-bits-per-sample=16 --stdin-endianness=le

Snap Live Embedding SDK version:8.0
Input file:stdin frequency:48000 channels:2 bits per sample:16
License remaining days: -1
Channel 'name - id' found in audio license:
        BBC_RADIO_1 - 2001
        BBC_RADIO_2 - 2002
        BBC_RADIO_3 - 2003
[INFO] - Using channel name: BBC_RADIO_1
[INFO] - Embedder Initialization starting ...
[INFO] - Start watermarking using ID 2001 (SNAP)
[INFO] - SIMD level used: AVX
[INFO] - Applying CPU affinity mask - All CPU(s) selected
[INFO] - Embedder Initialization finished
[INFO] - Resynchronizing timecode
[INFO] - Flushing remaining audio data from embedder ...
[INFO] - Embedder uninitialization starting ...
[INFO] - Embedder uninitialization finished
[SUCCESS]
```

4. Check output pipe

To test output pipe, set "stdout" label as output file. Note: when using "stdout" piping, there is no more debug trace on console.

```
$ ./SampleSnapLiveAudioEmbedderSDK song.wav stdout
    | ffmpeg -loglevel warning -f s16le -channel_layout stereo -ar 48000 -i - -y
song.wmk.wav

(no output)
```

Check full piping

Finally, Watermarking sample may be used in full piping

```
$ ffmpeg -loglevel warning -re -channel_layout stereo -i song.wav -ac 2 -ar 48000 -f
s16le -acodec pcm_s16le -
    | ./SampleSnapLiveAudioEmbedderSDK stdin stdout --stdin-sampling-rate=48000 --
stdin-nb-channels=2 --stdin-bits-per-sample=16 --stdin-endianness=le
    | ffmpeg -loglevel warning -f s16le -channel_layout stereo -ar 48000 -i - -y
song.wmk.wav

(no output)
```

Make a test with an existing feed and stream on udp the result of the embedding:

NB: the URL feed can no longer work.

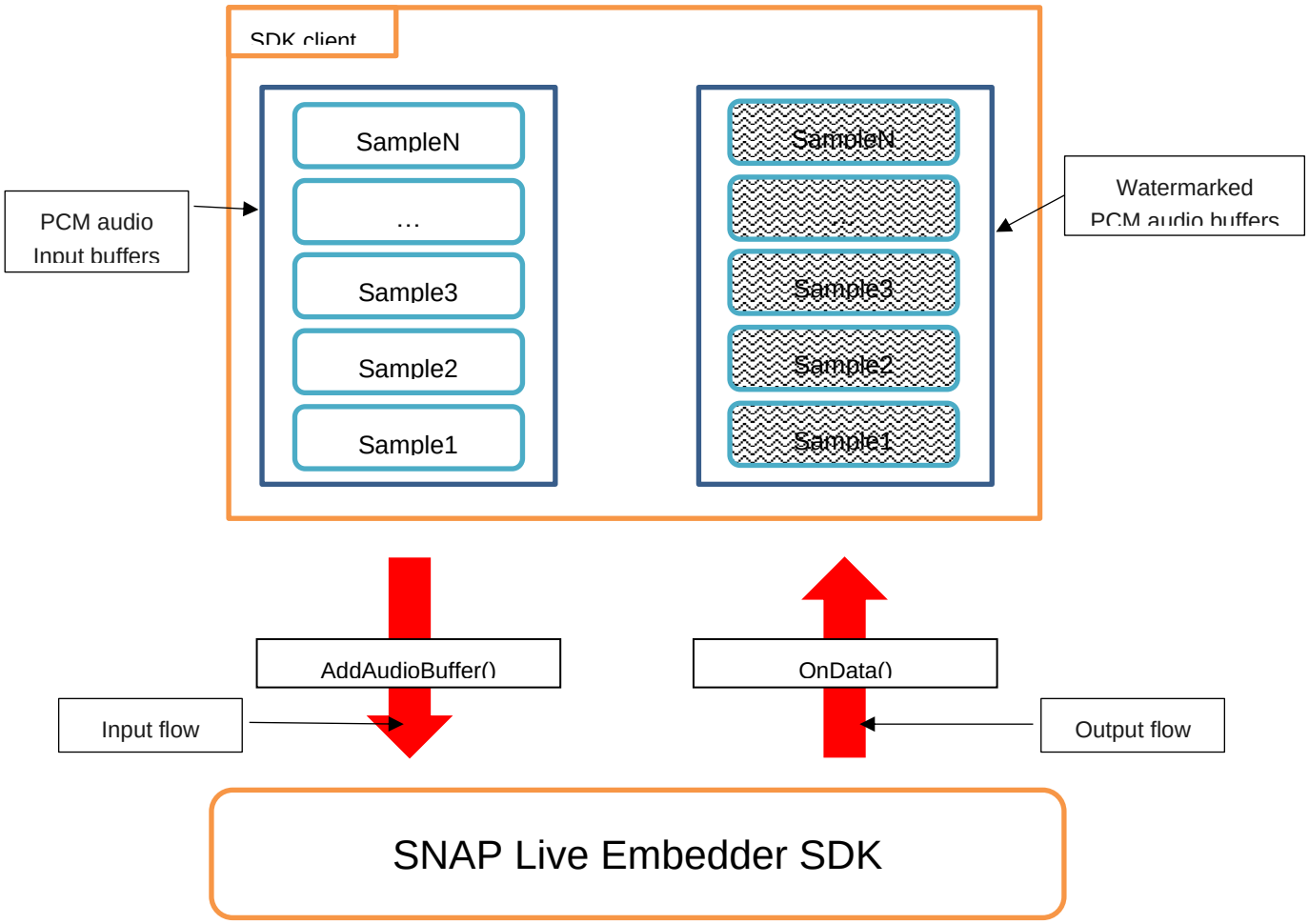
```
$ ffmpeg -re -i http://ais-live.cloud-services.paris:8000/virgin.mp3 -ac 2 -ar 48000 -
f s16le -c:a pcm_s16le -| ./SampleSnapLiveAudioEmbedderSDK stdin stdout --stdin-
sampling-rate=48000 --stdin-nb-channels=2 --stdin-bits-per-sample=16 --stdin-
endianness=le | ffmpeg -re -f s16le -channel_layout stereo -ar 48000 -i - -f mp3
udp://227.2.21.10:1234


(no output)
```

3 Package Description

3.1 Introduction

The SNAP Live Embedder SDK library is a software component that implements the SNAP audio watermarking technology. The SDK receives PCM audio as input buffers and applies the watermark operation on these buffers. Watermarked buffers are asynchronously returned to the library client.



The SDK library is implemented in C++. Library integration is also detailed in this document through a C++ sample code that illustrates a wav file watermarking (Cf. 0).

3.2 Minimum external requirements for the Embedding SDK

3.2.1 Hardware

CPU: 10% of 1 CPU Xeon 5606 @ 2,13 Ghz for one stereo stream
RAM: 1GB

3.2.2 Operating System and development environment

- **X86 platforms (64 bits):**
  - GNU/Linux operating system:
    - Package: setupSnapFileAudioEmbedderSDK.x.x-x.**debian9.x86_64.tar.gz**
    - kernel 4.9.0-3-amd64
    - glibc 2.24-11+deb9u1
    - Development Environment: GNU GCC 6.3.0, glibc 2.24
  - Windows operating System:
    - Package: setupSnapLiveAudioEmbedderSDK.x.x-x.exe
    - Microsoft Windows (64 bit): Windows 7 / Server 2008 / Windows 8.1 / Windows 10 / Server 2012.
    - Development Environment: Visual Studio 2015
    - If Microsoft Visual Studio is not present on the target platform, it is necessary to install Microsoft Visual C++ redistributable for Visual Studio 2015 package in order to run the sample application. It can be downloaded from Microsoft:
      - x64 : https://www.microsoft.com/fr-fr/download/details.aspx?id=48145
- **ARM A platforms (hard float) (32 bits):**
  - IMX 6 Cortex A9 (ARMv7-A processor)
    - Package: setupSnapLiveAudioEmbedderSDK.x.x-x.ubuntu1404-imx6hf.tar.gz
    - Operating System: GNU/Linux
      - kernel 3.19
      - glibc V2.20
    - Development environment is the GCC OpenEmbedded crosscompiler arm-oe-linux-gnueabi V4.9 (hard float):

3.3 Key Concepts

3.3.1 API exchange data format

The Snap Live embedder API offers a “memory to memory” API style which means input data and output data are all exchanged in RAM buffers.

3.3.2 Supported audio format

- Supported input audio format:
  - PCM little endian
- Supported audio sample rate:
  - 48 kHz
- Supported audio sample bits length:
  - 16 bits per sample
  - 32 bits per sample
- Note: for samples coded in 24 bits format, use a 32 bits format instead, where the least significant bytes will be set to 0
- Supported types of PCM audio buffers:
  - Single, stereo or multiple audio channels. Use *SetChannels* method to set the number of channel: 1 for mono, 2 for stereo…

3.4 Disclaimer

This embedding software source code sample cannot be integrated or used AS IS in any given application.

This embedding software source code sample is provided as a mere example of the possible integration of the embedding libraries.

KANTAR MAKES NO REPRESENTATION ABOUT THE SUITABILITY OF THIS SOURCE CODE FOR ANY PURPOSE. IT IS PROVIDED "AS IS" WITHOUT EXPRESS OR IMPLIED WARRANTY OF ANY KIND. KANTAR DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOURCE CODE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL KANTAR BE LIABLE FOR ANY SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOURCE CODE.

Subject to the above, Kantar hereby grants the integrator a non-exclusive, non-assignable, copyright license to use and modify the embedding software source code sample to develop and test audience measurement applications.

3.5 Content of the Package

3.5.1 Package content

Once deployed, SDK delivery contains:

- A bin folder, with x86_64 (64 bits) sub-folder. It contains SampleSnapLiveAudioEmbedderSDK.exe and SampleSnapLiveAudioEmbedderSDK.exe, the executables of the sample application, and the libraries that they require at run-time. There is no sub-folder on the ARM version.
- A docs folder that contains HTML documentation of SNAP Live Embedder API Library classes and methods. Open file /docs/html/index.html to browse API documentation.
- An include folder that contains all header files needed for SDK integration.
- A lib folder, with x86_64 sub-folder that contains the Snap Live Audio Embedder SDK library.
- A sample folder that contains the code of the Sample Application and the headers and libraries needed to integrate the Embedder Library.
- An AuthorizationCodeCL executable to generate AuthorizationCode.txt file.

3.5.2 Sample Folder Description

The sample folder contains:

- A lib folder with x86_64 (64 bits) sub-folder that contains the WavLibrary.lib for Windows and libWavLibrary.so for Linux.
- An include folder: contains the headers files needed by the sample applications
    - OnlineSnapLiveEmbedderListenerSample.h
    - RenderingClock.h
    - SnapLiveEmbedderListenerSample.h
    - SampleUtils.h
    - WAVPCMMuxer.h
    - WAVPCMParser.h
- A source folder: contains the source code of the Sample Application
    - OnlineSnapLiveEmbedderListenerSample.cpp
    - OnlineSnapLiveEmbedderSample.cpp
    - SnapLiveEmbedderListenerSample.cpp
    - SnapLiveEmbedderSample.cpp
- Resources to build sample at root of sample folder:
    - For Windows, the project to compile the sample application:
        - Sample-2015.sln for Visual Studio 2015
        - Sample-2015.vcxproj for offline project SnapLiveEmbedderSDK
        - Online-Sample-2015.vcxproj for online project OnlineSnapLiveAudioEmbedderSDKSample
    - For Linux:
        - Makefile make file to compile the Sample Application
- A shared library WavLibrary used for Wave files I/O in the sample code.

3.6 How to use the package

3.6.1 Sample Application

The sample application provided with the package shows how the SDK API can be used to watermark a wav file.

Audio buffers are read from the input file and passed to the SDK API. In return, the sample retrieves the watermarked buffers (Cf. *SnapLiveEmbedderListenerSample::OnData* ()) and builds a watermarked output file.

The sample application (*SnapLiveEmbedderSamples.cpp*) may process 48kHz mono or stereo PCM .WAV files with basic .WAV file header and is not designed to operate with extended .WAV files.

3.6.2 Build the Sample Application

3.6.2.1 On Windows

- Go to sample folder.
- Open the visual studio project: Sample-2015.sln
- Select **Release**. Choose x64 option to build the corresponding binary. Choose project SnapLiveAudioEmbedderSDK or OnlineSnapLiveAudioEmbedderSDK.
- Generate solution:
  - SnapLiveAudioEmbedderSDKSample_**Release**.exe is created in SDK/bin/x86_64 folder for 64 bits.
  - OnlineSnapLiveAudioEmbedderSDKSample_**Release**.exe is created in SDK/bin/x86_64 folder for 64 bits.

3.6.2.2 On Linux

- Open a terminal
- Go to sample folder.
- For 64 bits:
  - Run command: make x86_64
  - SnapLiveAudioEmbedderSDK_Sample and OnlineSnapLiveAudioEmbedderSDK_Sample are created in project/bin/x86_64 folder.
- For ARM:
  - Run command: make
  - SnapLiveAudioEmbedderSDK_Sample is created in project/bin folder.

3.6.3 Sample Application overview

Once the application has been recompiled, go to project/bin folder according to target and launch the newly compiled binary in a similar way of the pre-compiled Sample Application (as described in the Quick start chapter at the beginning).

4 Sample Code Explanation

This chapter aims to describe the sample code content in order to illustrate the use of the Snap Live embedder API.

4.1 Introduction

The sample is divided in different parts:

- *SnapLiveEmbedderSample.cpp* (project/sample/source):
  - Command line management.
  - Input WAV file management.
  - Creation and initialization of the embedding library with specific parameters.
  - Embedding process.
- *SnapLiveEmbedderListenerSample.cpp/h* (project/sample/source)
  - Output WAV files management.
  - Handling of data/alarm events through callbacks.
- *SamplesUtils.h* (project/sample/include)
  - Class to manage command line parameter.
  - Exception management.
  - Class regroup method SnapLive EmbedderSDK

4.2 Input and Output WAV File Management

RenderingClock, WAVPCMMuxer and WAVPCMParser are helper class provided along with the Embedder Library to process input and output WAV files to demonstrate how you can use the SDK.

IWavPcmParser class is used to retrieve different characteristics from the header of the input WAV file.

IWavPcmMuxer class is used to create the output WAV file formatted with the proper header.

IRenderingClock class is used to play as real time the audio sample.

4.3 Embedding Library Initialization for Offline SDK

4.3.1 Create embedder listener

The listener is used to get back watermark data throw OnData callback, and to get all SDK event (like alarm) throw OnEvent callback.

```
SnapLiveEmbedderListenerSample* pListener = new SnapLiveEmbedderListenerSample();
pListener->Init(…);
```

4.3.2 Embedder Creation

Create the embedder:

```
ISnapLiveAudioEmbedder *pEmbedder = NULL;
result = CreateSnapLiveAudioEmbedder("", "", pListener, &pEmbedder);
```

The first parameter is the license.lic path. The second parameter is the license.aud path.
In this example, they are not set, which means that executable searches licenses in the current application folder.

4.3.3 Embedder Parameters

Get default embedder parameters and update them if needed.

```
ISnapLiveAudioEmbedder::IEmbedderParameters* pParam;
result = pEmbedder->GetEmbedderParameters(pParam);
```

Update the number of bits per sample:

```
result = pParam->SetBitsPerSample(audio_description.bits_per_sample);
```

Update the number of channel (Mono, Stereo…):

```
result = pParam->SetChannels(audio_description.nb_channels);
```

4.3.4 Select the Channel to use

Get the channel list from audio license and select the channel to use for watermarking.

```
size_t list_elements = 0;
char** channelNameList = NULL;
result = pEmbedder->GetChannelNameList(channelNameList, list_elements);
```

Select the first channel name:

```
result = pEmbedder->SetChannelName(channelNameList[0]);
```

4.3.5 Finish the Initialization

Once all these initializations are done, the Initialize() method must be called to complete the embedder initialization.

```
result = pEmbedder->Initialize();
```

The library is now ready to embed watermark data.

Note that if one of the precedent steps is omitted, the Initialize() method will return an error.

4.4 Embedding Library Initialization for Online SDK

4.4.1 Create embedder listener

The listener is used to get back watermark data throw OnData callback, and to get all SDK event (like alarm) throw OnEvent callback.

```
SnapLiveEmbedderListenerSample* pListener = new SnapLiveEmbedderListenerSample();
pListener->Init(…);
```

4.4.2 Embedder Creation

Create the embedder:

```
ISnapLiveAudioEmbedder *pEmbedder = NULL;
result = CreateOnlineSnapLiveAudioEmbedder(pListener, &pEmbedder, login, password);
```

The third parameter is the login string value for to connect to license server. The four parameter is the password string value for to connect to license server. Function returns an error code which is verified by “ExceptionSdk” object.

4.4.3 Embedder Parameters

Get default embedder parameters and update them if needed.

```
ISnapLiveAudioEmbedder::IEmbedderParameters* pParam;
result = pEmbedder->GetEmbedderParameters(pParam);
```

Update the number of bits per sample:

```
result = pParam->SetBitPerSample(audio_description.bits_per_sample);
```

Update the number of channel (Mono, Stereo…):

```
result = pParam->SetChannels(audio_description.nb_channels);
```

4.4.4 Select licence to be used

Get licenses available on the server with credentials used

```
pEmbedder->GetLicenses(licenseNameList, licenseIdList, nbLicense)
for (size_t i = 0; i < nbLicense; i++)
    cout << "\t" << licenseNameList[i] << " - " << licenseIdList[i] << endl;
```

Select the license to use with Id:

```
pEmbedder->SelectLicense(licenseIdList[0])
```

4.4.5 Set Metadata

To set metadata, use the function *AddMetadata()* with JSON content in UTF-8.

For the sample, we use a JSON file and give path of metadata file containing appropriate information, formatted as described below.

For example, json metadata file.

```
{
  "channel_name": "tv1",
}
```

Note: Channel name must match one of channel names present in audio license.

Call to load a metadata file in embedder.

```
pEmbedder->AddMetadata(GetFileContent(args.metadata_filepath.c_str()).c_str())
```

Note “Set metadata” must be done before the end of initialization.

4.4.6 Finish the Initialization

Once all these initializations are done, the Initialize() method must be called to complete the embedder initialization.

```
result = pEmbedder->Initialize();
```

The library is now ready to embed watermark data.

Note that if one of the precedent steps is omitted, the Initialize() method will return an error.

4.4.7 Integrate SNAP Live Embedder SDK in Your Own Application

The project folder is where the SDK is installed on Windows and where it is unpacked on Linux.
For example, by default on Windows 64bits, the project folder is *C:\Program Files (x86)\Kantar Media \SnapLiveAudioEmbedderSDK*.

- Use the header files located in *project/include*.
- Use the libraries located:
  - For 64 bits application in *project/lib/x86_64*
- Read documentation project/doc/api/index.html to have a detailed view of the C++ API.

5 SDK API description

5.1 List of classes (API documentation)

Please refer to the HTML documentation provided within the archive to have more detailed information: doc/api/index.html

5.2 Asynchronous API

The SNAP Live embedder API runs in an asynchronous way.

Input buffers provided to AddAudioBuffer() are returned as watermarked output buffers **asynchronously** later on by the OnData() callback.

The Event() callback defined in IEmbedderListener is called synchronously. Then, you have to pay attention to treat them in another thread for time costly event management, such as file logging for instance. Otherwise, you block watermarking process that sends data to OnData() callback, thus it slows down watermarking process.

For a dynamic view approach of the API please refer to the Appendix.

5.3 Embedding Process

The audio input data read from the input file is passed to the embedder using AddAudioBuffer() method.

Input data can be 16, 24 and 32 bits PCM little endian with 48kHz as sample rate.

For 24 bits processing, the sample Application code illustrates the specific management to be done. In fact the SDK library uses 32 bits buffers with least significant byte to null. So in this case, you will need to transform your 24 bits buffers in 32 bits buffers with LSB to null.

To achieve better performances, you should provide buffers having a size of a multiple of 512 audio samples.

Once the embedding process is finished, the embedder instance must call the *Finalize* method and then the *DestroySnapLiveAudioEmbedder* method to properly close the Embedder Library.

```
pEmbedder->Finalize();
DestroySnapLiveAudioEmbedder(pEmbedder);
```

☞ Note: Pay attention that *Finalize* method may be a blocking method. It insures before returning that all input buffers received will be watermarked properly and returned by the OnData callback. The method Finalize have a json report on the project Online because if there is a connexion problem since embedding the method return json report. Don't change the report, just resend with method SendReport (see 5.9.1).

If you have finished all embedding, you have to delete listener also.

```
if(pListener != NULL)
  delete pListener;
```

5.4 Set CPU affinity and Thread priorities

The thread priorities and CPU affinity can be changed dynamically, and anytime (before running and while running), with SetThreadCPUAffinity and SetThreadPriority methods.

```
result = pEmbedder->SetThreadCPUAffinity(0);
```

```
result = pEmbedder->SetThreadPriority(THREAD_PRIORITY_TIME_CRITICAL);
```

Note: To change the thread priority, the application must be executed with root privileges.

For Windows platforms please refer to the Microsoft MSDN documentation to get all possible priority values allowed:
http://msdn.microsoft.com/en-us/library/windows/desktop/ms686277(v=vs.85).aspx

For Linux platforms, 3 different kinds of scheduling policies are available: The normal policy SCHED_OTHER and 2 real-time policies, SCHED_FIFO and SCHED_RR:

- For the normal SCHED_OTHER policy, the priority range is from 20 to 19, the lowest value standing for the highest priority level.
- For the 2 real-time policies, the priority range is from 1 to 99, the higher value standing for the highest priority level.

Please refer to linux sched man page for more details: http://man7.org/linux/man-pages/man7/sched.7.html

5.5 Enable record

Input and output of SDK can be recorded into WAV files. This functionality should be used only during integration, to check flows. To enable record, you have to set a directory path where files will be stored.

Call *EnableRecord()* method before calling the *Initialize()* method of the SDK. Path given should already exist and it should have write access for the user. In case of wrong parameters given, it will raise a SDK exception when you call the Initialize() method of the SDK.

```
result = pEmbedder->EnableRecord("./a_path");
```

5.6 Resynchronize timecode

If the system time of the computer changes, you will need to resynchronize the timecode by a call to *ResynchronizeTimecode* method:

```
pEmbedder->ResynchronizeTimecode();
```

Watermarking process inserts a timestamp in the watermark payload based on the reference time (UTC) of the equipment. As a consequence, the watermarking process clock must be regularly synchronized to the reference time with ResynchronizeTimecode() function.

The time drift between the timestamp and the master clock of the playout center must never exceed a few seconds.

Thus, the ResynchronizeTimecode function must be called **once** a day **at a minimum** and **four times** a day **at a maximum** or if the difference between the equipment clock reference (UTC time) and the application clock exceeds ten seconds.

5.7 Callbacks

When data are sent to the library, the library will report results through the callbacks defined in *SnapLiveEmbedderListenerSample*.cpp (or Online*SnapLiveEmbedderListenerSample.cpp)*

5.7.1 OnData

This callback provides the watermarked output buffers. This is the AddAudioBuffer counterpart.

In our sample, the watermarked output wav file is created thanks to this callback: it reports the buffer containing the watermarked audio data.

The specific case of 24 bits is also managed in this callback, for transforming the 32 bits internal buffer format to 24 bits buffer.

☞ Note (OnData vs AddAudioBuffer): OnData first call only appears after the watermarking pipe processing has enough data to process (a minimum of 5 x 512 samples per channel). Thus, depending of the buffer size provided, several calls to AddAudioBuffer may have been performed to start the watermarking operation before watermark data arrives in OnData. From this point, the SDK embedder will follow a “one to one” call pattern between AddAudioBuffer and OnData.

For example, consider a PAL audio / Video signal (audio sampled at 48 KHz with 25 images per second).

As stated above, the watermarking pipe requires to have a least 5 x 512 samples = 2560 audio samples per channel, so a duration of 53 ms which means to have at least 2 video frames (2 x 1920 samples so a duration of 2 x 40 ms = 80 ms).

This example is just given as information in order to illustrate the time constraints related to the control of the watermarking buffer for a PAL signal. The SDK integrator has to make its own computation taking into account the constraints of its implementation.

5.7.2 OnEvent

This callback reports error, warning or information.

IEmbedderListener::Event structure provides information about events that may occur. The type field indicates the kind of alarm:

- TYPE_INFO
- TYPE_WARNING
- TYPE_ERROR

Remaining code and message fields provide more detailed information for each event type.
If you need to do time costly action when receiving OnEvent callback, you have to do it in another new thread.

5.8 Specific functions for Offline SDK

5.8.1 GetLicenseRemainingDays

Get remaining days allow retrieve the number of day before expiration date of the license. A license without expiration date return -1.

```
Int remainingDay = 0;
result = pEmbedder->GetLicenseRemainingDays(&remainingDay);
```

5.9 Specific functions for Online SDK

5.9.1 SendReport

For the online SDK, in case of an error with Finalize method (for example a network connection lost), the SendReport must be called later (for example when network connection is back).

```
const char* jsonReport = NULL;
result = pEmbedder->Finalize(&jsonReport);      // Example: Error due to connection lost
```

```
// try to resend the report
result = pEmbedder->SendReport(jsonReport);
```

Note: the string *jsonReport* is still available as long as the *pEmbedder* is not destroyed.

6 Change log

6.1 Release 8.1: changes since release 8.0

- Added CentOS 7 support
- Linux: added an audio pipeline support in sample application

6.2 Release 8.0: changes since release 7.1

- Algorithm improvement

6.3 Release 7.1: changes since release 7.0.2

- Algorithm improvement

6.4 Release 7.0.2: changes since release 6.0

- Algorithm improvement
- Add online audio embedder SDK
    - Licenses retrieve from Kantar server and check periodically
    - Watermarking jobs are stored on Kantar server
- For offline SDK, it's possible to define license with expiration date

6.5 Release 6.0: changes since release 5.1

- Add VST plugin sample integration
- Algorithm improvement
- ARM FFT improvement using Neon10 lib

6.6 Release 5.1: changes since release 5.0

- Add GetChannelNameIDList method that return channel name with id value.
- Improve sample code delay

6.7 Release 5.0: changes since release 4.0

- New default path: *C:\Program Files (x86)\Kantar Media*
- Audio license path can be set.

6.8 Release 4.0: changes since release 3.2

- SNAP frequency improvements

6.9 Release 3.2: changes since release 3.1

- Add the ability to avoid overmarking if the input stream is already watermarked with the same key. This is done by license management.
- New timecode management in order to have all the timecode (of the different Ids) of one license incremented at the same rhythm.

6.10 Release 3.1: changes since release 3.0

Windows and Linux version, x86 and x86_64:

- Rebranding for Kantar
- SafeNet licensing replaced by Kantar licensing.
- SampleUtils shared library has been replaced by WavLibrary

6.11 Release 3.0: changes since release 2.0.2

No API change.

This version supports ARMv7-A architecture in hard float in 32 bits.

Rebranding for Kantar.

SafeNet licensing replaced by Kantar licensing.

SampleUtils library is now a shared library for Linux platform.

6.12 Release 2.0.2: changes since release 2.0.1

No API change.
Bug fix on the psycho acoustic model.

6.13 Release 2.0.1: changes since release 2.0.0

No API change.
Disable license check thread every 60 seconds.

A. Licenses and 3rd party software used

A.1 Third party

A.1.1 Intel IPP

Intel® Integrated Performance Primitives (IPP) license for distribution of Redistributable runtime library files, which prohibits disassembly and reverse engineering of the Redistributables, as per End User License Agreement for the Intel(R) Software Development Products, version as of May 2012.

A.1.2 Inno Setup (For Windows only)

http://www.jrsoftware.org/files/is/license.txt

A.2 Open Source

A.2.1 OpenMax (Only for Linux arm)

1. Subject to the provisions of this Agreement, ARM hereby grants to YOU (either an individual or single entity), under ARM's copyright in the Software, a perpetual, non-exclusive, non-transferable, royalty free, worldwide licence to ; (i) use, copy, modify, the Software for the purposes of developing or having developed software applications and; (ii) distribute and sublicense the right to use, copy and modify the software applications to third parties.

2. THE SOFTWARE IS LICENSED "AS IS". ARM EXPRESSLY DISCLAIMS ALL REPRESENTATIONS, AND WARRANTIES EXPRESS, IMPLIED OR STATUTORY, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF SATISFACTORY QUALITY, MERCHANTABILITY, NON-INFRINGMENT OR FITNESS FOR A PARTICULAR PURPOSE.

3. Your use of this Software and the right to redistribute any software applications developed by or for YOU and which are derived from the Software may require you to obtain patent licences from third parties ("Third Party Patents"). ARM therefore requires and YOU hereby agree that prior to exercise of any of the rights to distribute any software applications in accordance with the licences granted under this Agreement, YOU shall have obtained all necessary rights and licences to Third Party Patents, of which YOU are aware of or become aware during the term of this Agreement, to enable YOU to distribute the ARM Software in accordance with the licences granted hereunder without infringing the Third Party Patents whether as a primary, secondary, indirect or contributory infringer, or otherwise, and the copyright licences contained herein are conditional on you agreeing to obtain such licences. For the purpose of interpretation of this Clause 3, any allegation by a third party that any action by YOU infringes any Third Party Patents shall be presumed as valid until properly rebutted by YOU and ARM may suspend the licences granted in Clause 1 until any such allegation is resolved in favour of YOU or YOU reach a settlement with the party making the allegation. If any breach by YOU of the provisions of this Clause 3 results in ARM being subject to a claim for infringement of any Third Party Patents, YOU shall indemnify against and hold ARM harmless from any claims, demands, damages, costs and expenses made against or suffered by ARM as a result of any such claim or action.

4. No licence, express, implied or otherwise, is granted to YOU under the provisions of Clause 1, to use the ARM tradename in connection with the Software or any products based thereon. Nothing in Clause 1 shall be construed as authority for YOU to make any representations on behalf of ARM in respect of the Software.

5. If you are downloading the Software on behalf of a company, partnership or other legal entity, you represent and warrant that you have authority to bind that entity to these terms and Conditions. If you do not have this authority you should not proceed to download the Software.

6. Any breach by YOU of the terms of this Agreement shall entitle ARM to terminate this Agreement with immediate effect. Upon termination of this Agreement, all licences granted to YOU shall cease immediately and YOU shall at ARM's option either return to ARM or destroy all copies of the Software including any modifications or derivatives thereof.

7. This Agreement shall be governed by and construed in accordance with the laws of England and Wales.

A.2.2 **Jsoncpp**

http://jsoncpp.sourceforge.net/LICENSE

A.2.3 **LibTomCrypt**

https://github.com/libtom/libtomcrypt/tree/1.17

A.2.4 **Boost**

A.2.5 **Boost Software License - Version 1.0 - August 17th, 2003**

http://www.boost.org/LICENSE_1_0.txt

A.2.6 **Boost / Thread license**

Copyright (C) 2001-2003
William E. Kempf
Permission to use, copy, modify, distribute and sell this software and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation. William E. Kempf makes no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.

A.2.7 **Pthread**

http://www.gnu.org/licenses/old-licenses/lgpl-2.1.en.html

B. Technical Support by Kantar

To get technical assistance, check on the status of problems, or report new problems, contact Kantar Product Support via e-mail, phone, or fax. We welcome any suggestions, improvements and feedback concerning the present User Guide or software described herein.

B.1 Web technical support

Technical Support link: http://www.kantarmedia.com/watermarkinghelpdesk

B.2 Phone support

Kantar S.A.S.
12 square du Chêne Germain
35510 Cesson-Sévigné
France
Tel: +33 2 90 92 37 37
Fax: +33 2 99 22 61 63