Scientific Working Group on Digital Evidence

Technical Notes on FFmpeg for Forensic Audio Examination

23-A-001-1.1

Disclaimer Regarding Use of SWGDE Documents

Redistribution Policy

Requests for Modification

SWGDE encourages stakeholder participation in the preparation of documents. Suggestions for modifications are welcome and must be submitted via the SWGDE Request for Modification Form or forwarded to the Secretary in writing at secretary@swgde.org. The following information is required as a part of any suggested modification:

a) Submitter's name
b) Affiliation (agency/organization)
c) Address

d) Telephone number and email address
e) SWGDE Document title and version number
f) Change from (note document section number)
g) Change to (provide suggested text where appropriate; comments not including suggested text will not be considered)
h) Basis for suggested modification

Intellectual Property

All images, tables, and figures in SWGDE documents are developed and owned by SWGDE, unless otherwise credited.

Unauthorized use of the SWGDE logo or document content, including images, tables, and figures, without written permission from SWGDE is a violation of our intellectual property rights.

Individuals may not misstate and/or over represent duties and responsibilities of SWGDE work. This includes claiming oneself as a contributing member without actively participating in SWGDE meetings; claiming oneself as an officer of SWGDE without serving as such; claiming sole authorship of a document; use the SWGDE logo on any material and/or curriculum vitae.

Any mention of specific products within SWGDE documents is for informational purposes only; it does not imply a recommendation or endorsement by SWGDE.

Technical Notes on FFmpeg for Forensic Audio Examination

Table of Contents

1. Purpose

This document provides a general awareness of FFmpeg (Fast Forward mpeg), its functions, basic use, and common uses as it pertains to forensic audio examinations. FFmpeg is an open source, cross-platform tool that uses a command line interface to play, convert, and stream multimedia data.

2. Scope

This document is intended for use by forensic analysts/examiners seeking familiarization with FFmpeg's open-source suite in forensic audio examinations. It can be used to complement training, experience, and tool validation. Refer to *SWGDE 16-V-002-3.0 Technical Notes on FFmpeg for Forensic Video Examination* for information regarding commands commonly used in forensic video examinations, including handling audio streams in video files.

3. Limitations

This document was prepared with the resources available at the time of publication. As with all technology, FFmpeg is constantly evolving with frequent implementation of new functions, as well as some deprecations. Because installation configurations can vary based on chosen options or versions, functionality may not conform to the tasks cited here.

This document is not intended for use as a step-by-step guide for conducting complete forensic audio examinations. While FFmpeg will process many codecs, it may not accurately decode some proprietary codecs or file containers. Interpretation of some lossy audio files, for example, may require customization of the arguments used in the commands. FFmpeg will revert to defaults in conversion processes when certain parameters are not specified (e.g., bit depth).

This document also does not address all commands available in FFmpeg, which can be found in FFmpeg's documentation. Refer to the FFmpeg website for full documentation. The website documentation refers only to the most current stable version.

This is also not an endorsement of FFmpeg to the exclusion of other multimedia processing tools. As with all forensic tools utilized in casework, testing and validating FFmpeg's functionality using known test data is necessary before use on examination materials. Refer to *SWGDE 18-Q-001-2.1 Minimum Requirements for Testing Tools Used in Digital and Multimedia Forensics*.

4. FFmpeg Tools

FFmpeg is an open-source framework that can be implemented across most operating systems. The base function of the platform is to leverage multiple libraries of codecs to gain insight into multimedia files as well as allow playback and conversion of multimedia files. FFmpeg contains three command line applications with unique functions: **ffmpeg**, **ffprobe**, and **ffplay**. In this document, "FFmpeg" refers to the application framework. The individual command line tools are all lowercase.

4.1 ffmpeg

A command line tool capable of many useful processing and analysis functions. Some uses in audio forensics include file format and codec conversion, filter processing, and streamhashing.

4.2 ffprobe

Application for displaying digital multimedia file metadata and file properties, including, but not limited to, stream information (video, audio, and data), channel configuration, sampling rate, duration, codec, etc.

4.3 ffplay

A media player that utilizes the installed FFmpeg libraries to play multimedia files.

5. FFmpeg Installation

Refer to *SWGDE 16-V-002-3.0 Technical Notes on FFmpeg for Forensic Video Examination* for installation instructions. The latest stable version of FFmpeg should be used in forensic examinations and shall be documented in examination notes.

6. FFmpeg Informational Commands

The following command line options are usable with any of the executables (ffmpeg, ffprobe, ffplay). In the examples below, only ffmpeg will be shown. Note that commands are case sensitive.

6.1 Help (`-h`)

```
ffmpeg -h
```

The `-h` option displays general help or can be used with arguments to get function-specific help.

6.2 Show license (`-L`)

```
ffmpeg -L
```

The `-L` option displays the version number, the compiler, the enabled libraries, and the version numbers of built-in libraries.

6.3 Installed codecs (`-codecs`)

```
ffmpeg -codecs
```

The `-codecs` option lists all the media bitstream formats supported by the libavcodec library, a key library installed with FFmpeg. The list indicates whether FFmpeg supports encoding and/or decoding of data in the specified codec.

6.4 Available formats (`-formats`)

```
ffmpeg -formats
```

The `-formats` option lists all the file formats available to FFmpeg. The list indicates whether each format supports stream multiplexing or demultiplexing.

7. Basic Command Entry Format

For purposes of this section, the example file *input.wav* is a two-channel audio file, with the audio data encoded with signed 16-bit pulse code modulation (PCM) (little endian byte order) at a sampling rate of 44,100 samples per second ("Hz" as reported by FFmpeg).

7.1 ffprobe (Basic Usage)

```
ffprobe input.wav
```

`ffprobe`

Calls the ffprobe executable.

`input.wav`

The name of the file in the local directory.

```
Input #0, wav, from 'input.wav':
 Duration: 00:00:10.00, bitrate: 1411 kb/s
 Stream #0:0: Audio: pcm_s16le ([1][0][0][0] / 0x0001), 44100 Hz, stereo, s16, 1411 kb/s
```

7.1.1 Explanation of Results:

`Input`

Indicates the file number (`#0`), starting from 0, the file container format (`wav` [Resource Interchange File Format (RIFF) WAV]), and the input file name (`'input.wav'`) contained in the current directory from which ffmpeg was run (not listed).

`Duration`

Displays the duration of the file (`00:00:10.00`, hh:mm:ss.hundredths)

`bitrate`

Displays the bitrate of the file (`1411 kb/s`).

`Stream`

Displays the streams in the file and their corresponding stream numbers. This file has one stream (#0:0), which is identified as containing audio data.

`Audio`

Displays the codec used to encode the audio stream and its two-character code (TwoCC) (`pcm_s16le`, `0x0001`), audio sampling rate (`44100 Hz`), channel configuration (`stereo`), audio sample format (`s16`, signed 16-bit PCM), and audio bitrate (`1411 kb/s`). It is noted that the audio bitrate value for PCM data is a calculated value and is not an explicit field within the RIFF WAV header.

7.2 ffprobe (stream and format Details)

The following page shows the ffprobe output from this command:

```
ffprobe -show_streams -show_format input.wav
```

`ffprobe`

Calls the ffprobe executable.

`-show_streams`

Displays detailed properties and metadata of each stream in the file.

`-show_format`

Displays detailed properties and metadata of the container format.

`input.wav`

The name of the file in the local directory.

```
Input #0, wav, from 'input.wav':
 Duration: 00:00:10.00, bitrate: 1411 kb/s
 Stream #0:0: Audio: pcm_s16le ([1][0][0][0] / 0x0001), 44100 Hz, 2 channels, s16, 1411 kb/s
[STREAM]
index=0
codec_name=pcm_s16le
codec_long_name=PCM signed 16-bit little-endian
profile=unknown
codec_type=audio
codec_tag_string=[1][0][0][0]
codec_tag=0x0001
sample_fmt=s16
sample_rate=44100
channels=2
channel_layout=unknown
bits_per_sample=16
initial_padding=0
id=N/A
r_frame_rate=0/0
avg_frame_rate=0/0
time_base=1/44100
start_pts=N/A
start_time=N/A
duration_ts=441000
duration=10.000000
bit_rate=1411200
max_bit_rate=N/A
bits_per_raw_sample=N/A
nb_frames=N/A
nb_read_frames=N/A
nb_read_packets=N/A
DISPOSITION:default=0
DISPOSITION:dub=0
DISPOSITION:original=0
DISPOSITION:comment=0
DISPOSITION:lyrics=0
DISPOSITION:karaoke=0
DISPOSITION:forced=0
DISPOSITION:hearing_impaired=0
DISPOSITION:visual_impaired=0
DISPOSITION:clean_effects=0
DISPOSITION:attached_pic=0
DISPOSITION:timed_thumbnails=0
DISPOSITION:non_diegetic=0
DISPOSITION:captions=0
DISPOSITION:descriptions=0
DISPOSITION:metadata=0
DISPOSITION:dependent=0
DISPOSITION:still_image=0
[/STREAM]
[FORMAT]
filename=input.wav
nb_streams=1
nb_programs=0
nb_stream_groups=0
format_name=wav
format_long_name=WAV / WAVE (Waveform Audio)
start_time=N/A
duration=10.000000
size=1764044
bit_rate=1411235
probe_score=99
[/FORMAT]
```

7.2.1 Explanation of Results (Partial):

`[STREAM]…[/STREAM]`

Displays various characteristics related to the audio stream within the file Note that some values are calculated values and are not explicit fields within the metadata (e.g., “duration_ts”, “duration”, and “bit_rate”). Furthermore, some fields are not applicable but are still reported. More detailed information can be found in the ffmpeg documentation.

`[FORMAT]…[/FORMAT]`

Displays various characteristics related to the container file including basic information related to the characteristics of the media stream within it Note that some values are calculated values and are not explicit fields within the metadata (e.g., “duration” and “bit_rate”) In this example, the “bit_rate” calculation is erroneously based on the size of the entire file including the 44-byte RIFF WAV header (1,764,044 bytes), and not just the audio data contained in the “data” chunk (1,764,000 bytes). More detailed information can be found in the ffmpeg documentation.

7.3 ffplay (Basic Usage)

```
ffplay input.wav
```

`ffplay`

Calls the ffplay executable.

`input.wav`

The name of the file in the local directory.

By default, when an audio file is opened and played back using ffplay, a real-time spectrographic display will appear for the audio data. The spectrographic display will show a combination of all channels within the given audio stream as a single spectrogram without axis labels or numerical values.

8. ffmpeg

This is the basic command structure for ffmpeg; all other commands will follow this structure:

```
[Call Program] [Input Arguments] -i [Input File] [Output Arguments] [Output File]
```

Below is a table setting forth common arguments when working with audio files. These are demonstrated in Sections 9 and 10.

Argument	Description
`-i input.wav`	Specify the input audio file

`-ar 8000`
`-ar 11.025k`
`-ar 44100`
etc.
	Set audio sampling rate for raw audio data (e.g., PCM) or resampling rate for output file. "`k`" is a keyword for FFmpeg to interpret the value in thousands.

`-ac 2`
`-ac 1`
etc.
	Set number of audio channels

`-c:a pcm_s16le` `-c:a pcm_s24le`	Set audio encoding to signed 16- or 24-bit PCM, little endian, as is standard for RIFF WAV files
`-c:a pcm_f32le`	Set audio encoding to 32-bit floating point PCM, little endian
`-c:a copy`	Copy the audio stream(s) from input to output without conversion or other processing

`-filter_complex "[0:a]channelsplit=channel_layout=stereo[left][right]"`
`-map "[left]"`
`-map "[right]"`
	Identify, label, and map the left and right channels of a stereo file for separate extraction, conversion, or streamhashing.

`-f streamhash -`	Computes a hash value (SHA-256 by default) of an input file's decoded media streams and displays the result

Table 1. Common arguments when working with audio files (Table Credit: SWGDE, 2023).

9. Commonly Used ffmpeg Commands in Forensic Audio Examinations

While it is possible to utilize ffmpeg without including input or output arguments, doing so will prompt ffmpeg to apply settings based on the application's defaults. As such, ffmpeg's default settings may not be optimal for forensic purposes.

Typically, the channel configuration and sampling rate of an input audio file or stream will be retained for an output audio file or stream, unless explicitly modified by certain arguments within the ffmpeg command. Many of the example commands given below are provided without input or output arguments (such as channel configuration and sampling rate) and therefore must be verified for use in forensic examinations. Actual commands used in a given case shall be documented in the case notes.

A given task may be accomplished using ffmpeg with different commands, arguments, etc. Each example command provided below may be one of several possible solutions for the respective task.

For purposes of the example commands below, MP3 audio files are used when compressed audio files are referenced. Other compressed audio files/streams, such as M4A files containing Advanced Audio Coding (AAC) encoded data and Windows Media Audio (WMA) files, can be processed using the same commands with appropriate modifications.

9.1 Conversion of a Compressed Audio File to a Signed 16-bit PCM WAV File

```
ffmpeg -i input.mp3 -c:a pcm_s16le output.wav
```

`ffmpeg`

Calls the ffmpeg executable.

`-i input.mp3`

The name of the input MP3 audio file in the local directory.

`-c:a pcm_s16le`

Tells ffmpeg to convert the input audio data to signed 16-bit little endian PCM audio data.

`output.wav`

The name and container file format of the output file to be saved into the local directory.

To verify that the channel configuration and sampling rate of the input MP3 file is maintained in the output WAV file, the text that is displayed by default when the ffmpeg command is run should be reviewed. As an example, the following excerpted text shows the input file and output file characteristics resulting from the command above, verifying the intended output:

```
Input #0, mp3, from 'input.mp3':
…
 Stream #0:0: Audio: mp3, 44100 Hz, stereo, fltp, 192 kb/s
…
Output #0, wav, to 'delete.wav':
…
 Stream #0:0: Audio: pcm_s16le ([1][0][0][0] / 0x0001), 44100 Hz, stereo, s16, 1411 kb/s
```

9.2 Conversion of a Compressed Audio File to a 32-bit Floating Point PCM WAV File

An input audio file can be transcoded to a higher resolution which may be more appropriate for further processing.

```
ffmpeg -i input.mp3 -c:a pcm_f32le output.wav
```

`ffmpeg`

Calls the ffmpeg executable.

`-i input.mp3`

The name of the input MP3 audio file in the local directory.

`-c:a pcm_f32le`

Tells ffmpeg to convert the input audio data to 32-bit floating point PCM audio data.

`output.wav`

The name and container file format of the output file to be saved into the local directory.

9.3 Conversion of an Audio File into a Lossless Compressed Format

```
ffmpeg -i input.mp3 output.flac
```

`ffmpeg`

Calls the ffmpeg executable.

`-i input.mp3`

The name of the input MP3 audio file in the local directory.

`output.flac`

The name and container file format of the output file to be saved into the local directory. In this case, Free Lossless Audio Codec (FLAC) is used, and ffmpeg will retain the original channel configuration and sampling rate.

9.4 Read Raw Stereo, Signed 16-bit PCM/Little Endian, 44.1 kHz Audio Data, and Output to a RIFF WAV File

If working with raw PCM audio for which the channel configuration, bit depth, and sampling rate are known, ffmpeg can be used to output the data in a playable RIFF WAV file.

```
ffmpeg -f s16le -ar 44.1k -ac 2 -i input.pcm -f wav output.wav
```

`ffmpeg`

Calls the ffmpeg executable.

`-f s16le`

Sets the format of the input data to signed 16-bit/little endian PCM audio data. Arguments preceding "`-i input.pcm`", are global arguments which determine how the input file is interpreted.

`-ar 44.1k`

Sets the sampling rate of the input data as 44,100 samples per second or 44.1 kHz.

`-ac 2`

Sets the channel configuration of the input data as 2-channel or stereo.

`-i input.pcm`

The name of the input file in the local directory.

`-f wav`

Tells ffmpeg to output the audio data into a RIFF WAV audio file with the necessary header information.

`output.wav`

The name of the output file to be saved into the local directory.

9.5 Split Stereo PCM WAV File Into Mono Left And Right Files

The following demonstrates how ffmpeg will default to 16-bit PCM WAV encoding unless specified otherwise.

16-bit example:

```
ffmpeg -i input.wav -filter_complex "[0:a]channelsplit=channel_layout=stereo[left][right]"
-map "[left]" output_left.wav -map "[right]" output_right.wav
```

32-bit example:

```
ffmpeg -i input.wav -filter_complex "[0:a]channelsplit=channel_layout=stereo[left][right]"
-c:a pcm_f32le -map "[left]" output_left.wav -map "[right]" output_right.wav
```

`ffmpeg`

Calls the ffmpeg executable.

`-i input.wav`

The name of the input WAV audio file in the local directory.

`-filter_complex "[0:a]channelsplit=channel_layout=stereo[left][right]"`

Identifies that the stereo audio track of input file "`0`" will be split into two channels labeled "`left`" and "`right`".

`-c:a pcm_f32le` [32-bit example]

Tells ffmpeg to encode the output audio data to floating point 32-bit little endian PCM audio data, to prevent ffmpeg from encoding to 16-bit PCM by default, as would occur with the 16-bit example.

`-map "[left]" output_left.wav`

Identifies that the channel labeled "`left`" is to be output to a file named "`output_left.wav`", which is saved into the local directory.

`-map "[right]" output_right.wav`

Identifies that the channel labeled "`right`" is to be output to a file named "`output_right.wav`", which is saved into the local directory.

9.6 Split Stereo Compressed Audio File Into Mono Left and Right Files (With Conversion to Signed 16-bit PCM)

```
ffmpeg -i input.mp3 -filter_complex "[0:a]channelsplit=channel_layout=stereo[left][right]"
-c:a pcm_s16le -map "[left]" output_left.wav -map "[right]" output_right.wav
```

`ffmpeg`

Calls the ffmpeg executable.

`-i input.mp3`

The name of the input MP3 audio file in the local directory.

`-filter_complex "[0:a]channelsplit=channel_layout=stereo[left][right]"`

Identifies that the stereo audio track of input file "`0`" will be split into two channels labeled "`left`" and "`right`".

`-c:a pcm_s16le`

Tells ffmpeg to convert the input audio data to signed 16-bit little endian PCM audio data.

`-map "[left]" output_left.wav`

Identifies that the channel labeled "`left`" is to be output to a file named "`output_left.wav`", which is saved into the local directory.

`-map "[right]" output_right.wav`

Identifies that the channel labeled "`right`" is to be output to a file named "`output_right.wav`", which is saved into the local directory.

9.7 Combine Two Mono PCM WAV Files Into a Stereo File (With input Files Having the Same Length and Encoding Characteristics)

The following demonstrates how ffmpeg will default to 16-bit PCM WAV encoding unless specified otherwise.

16-bit example:

```
ffmpeg -i input_left.wav -i input_right.wav -filter_complex
"[0:a][1:a]join=inputs=2:channel_layout=stereo[a]" -map "[a]"
output_stereo.wav
```

32-bit example:

```
ffmpeg -i input_left.wav -i input_right.wav -filter_complex
"[0:a][1:a]join=inputs=2:channel_layout=stereo[a]" -c:a pcm_f32le -
map "[a]" output_stereo.wav
```

`ffmpeg`

Calls the ffmpeg executable.

`-i input_left.wav`

The name of the first input WAV audio file in the local directory.

`-i input_right.wav`

The name of the second input WAV audio file in the local directory.

`-filter_complex "[0:a][1:a]join=inputs=2:channel_layout=stereo[a]"`

Tells ffmpeg to set up one or more connected filters to be applied to the audio streams of the two input files. In this case, there are two input files (files "`0`" and "`1`"), and the filter is applied to the audio streams ("`a`") of each file (i.e., "`[0:a][1:a]`"). One filter named "`join`" is being applied, which joins multiple input streams into one multi-channel stream. The number of input streams is 2 (i.e., "`inputs=2`"), the "`channel_layout`" is defined as "`stereo`", and the name of the multi-channel stream is "`[a]`".

`-c:a pcm_f32le` [32-bit example]

Tells ffmpeg to encode the output audio data to floating point 32-bit little endian PCM audio data, to prevent ffmpeg from encoding to 16-bit PCM by default, as would occur with the 16-bit example.

`-map "[a]" output_stereo.wav`

The multi-channel stream "`[a]`" is mapped to an output file named "`output_stereo.wav`", which is saved into the local directory.

9.8 Concatenate Two or More PCM WAV Files into a Single File (With Input Files Having the Same Encoding Characteristics)

Concatenating the audio streams from two or more PCM WAV files first requires that a text file containing the names of the input audio files on separate lines be created. A way to automate this is to move the input audio files into a single folder, ensure that they are in chronological order per their file names, and then run a Windows command batch file (".bat") to produce the text file, as follows:

```
for %%f in (*.wav) do (
     echo file '%%f' >> list.txt
)
```

The following would be the contents of the output "`list.txt`" file referencing a folder containing three audio files ("input1.wav", "input2.wav", and "input3.wav"):

```
file 'input1.wav'

file 'input2.wav'

file 'input3.wav'
```

ffmpeg can then use the "`list.txt`" file to create the concatenated PCM WAV file with the following command:

```
ffmpeg -f concat -safe 0 -i list.txt -c:a copy concatenated_output.wav
```

`ffmpeg`

Calls the ffmpeg executable.

`-f concat`

Tells ffmpeg to read a list of files and sequentially combine (concatenate) their designated streams into a composite output file.

`-safe 0`

Allow all file names and paths for the input files.

`-i list.txt`

The name of the text file in the local directory containing the input file names of audio to be concatenated.

`-c:a copy`

Copy the input audio streams to the output, concatenated file.

`concatenated_output.wav`

The name and container file format of the output file to be saved into the local directory.

9.9 Extract a Segment of a PCM WAV File Having Specific Starting Point and Length (Example 1) or Endpoint (Example 2) in "hours:minutes:seconds"

This example demonstrates two commands for extracting a segment of audio by either setting the start point and duration of segment (`-t`) or both start and end points of segment (`-to`).

Example 1:

```
ffmpeg -i input.wav -ss 00:01:00 -t 00:05:00 -c copy output_segment.wav
```

Example 2:

```
ffmpeg -i input.wav -ss 00:01:00 -to 00:06:00 -c copy output_segment.wav
```

`ffmpeg`

Calls the ffmpeg executable.

`-i input.wav`

The name of the input WAV audio file in the local directory.

`-ss 00:01:00`

Tells ffmpeg to start at minute one of the file with time referenced as "`hours:minutes:seconds`". Time may instead be expressed in seconds (e.g., "`60`"). If a "`-ss`" argument is not included in the command, ffmpeg will start the segment from the beginning of the file.

`-t 00:05:00`

Tells ffmpeg to limit the output segment to a length of 5 minutes. Alternatively, "`-to`" can be used to set the endpoint of the segment, as expressed in Example 2.

`-c:a copy`

Copy the audio encoding of the input files to the output file.

`output_segment.wav`

The name and container file format of the output file to be saved into the local directory.

10. Validation Commands

10.1 Streamhash

The "`-f streamhash`" argument produces hash values, Secure Hash Algorithm (SHA) 256 by default, for decoded media streams from an input file. It can be useful for verifying lossless transcoding from one audio codec to another (e.g., MP3 to PCM), or re-wrapping an audio stream from one container file format to another (e.g., AVI to WAV) [1].

This argument first converts the input audio data to raw signed 16-bit little endian PCM data, by default, which is then hashed with the SHA-256 algorithm. By computing the audio streamhash for both the source and destination files, a user can verify that the conversion or re-wrapping process of the audio data was transparent.

Unless explicitly specified, the streamhash is computed of the multiplexed audio stream of a multi-channel audio file and is not performed on a channel-by-channel basis.

An alternative method for streamhashing uses the "`-f hash`" switch, which currently defaults to the SHA-256 hash algorithm but can be overridden using the "`-hash XXX`" switch (where "`XXX`" is the identifier for the selected hash algorithm; e.g., "sha512"). Equivalent commands for "`-f streamhash`" and "`-f hash`" are given in the first example below (section 10.1.1); the remaining examples use only the "`-f streamhash`" switch.

10.1.1 Streamhash of a Mono or Single Stereo Stream in a Compressed Audio File

```
ffmpeg -loglevel error -i input.mp3 -f streamhash -
ffmpeg -loglevel error -i input.mp3 -f hash -
```

`ffmpeg`

Calls the ffmpeg executable.

`-loglevel error`

Tells ffmpeg to limit the logging on the screen to errors only. This makes it easier to view the displayed output hash value.

`-i input.mp3`

The name of the input MP3 audio file in the local directory.

`-f streamhash –` or `-f hash -`

Tells ffmpeg to compute the SHA-256 hash of the input mono or stereo MP3 audio file and output the result to the command window, signified by the dash at the end of the argument.

The resulting output for the "`-f streamhash -`" switch would appear as a single SHA-256 value, as follows:

```
0,a,SHA256=ef45a66266b75444e5ecd3c7a0d86ce2d87f66810763f704ea63f252a923a3f0
```

The resulting output for the "`-f hash -`" switch would appear as a single SHA-256 value, as follows:

```
SHA256=ef45a66266b75444e5ecd3c7a0d86ce2d87f66810763f704ea63f252a923a3f0
```

10.1.2 Streamhash of a Single Stereo Stream in a Compressed Audio File With the Output Hash Algorithm Set to SHA-512

```
ffmpeg -loglevel error -i input_stereo.mp3 -f streamhash -hash sha512 -
```

`ffmpeg`

Calls the ffmpeg executable.

`-loglevel error`

Tells ffmpeg to limit the logging on the screen to errors only.

`-i input_stereo.mp3`

The name of the input MP3 audio file in the local directory.

`-f streamhash`

Tells ffmpeg to compute the hash of the input stereo MP3 audio file.

`-hash sha512 -`

Tells ffmpeg to utilize the SHA512 hash algorithm, overriding the default SHA-256 hash algorithm, and output the result to the command window, signified by the dash at the end of the argument.

The resulting output would appear as a single SHA512 value, as follows:

```
0,a,SHA512=d2b6290e0483f4d41191844f3cabfb1928a575569a0c8fabb2cf963cc05
22635dca64e14ba06e2220cd51bf49c5b84844f1137546d7001176b9b3f80a943657f
```

Multiple hash algorithms can be utilized in a single command by including additional "`-f streamhash`" switches. The following command calculates both the default SHA256 streamhash value of an input stereo audio file followed by its SHA512 streamhash value:

```
ffmpeg -loglevel error -i input_stereo.mp3 -f streamhash - -f streamhash -hash sha512 -
```

The results would appear as follows:

```
0,a,SHA256=ef45a66266b75444e5ecd3c7a0d86ce2d87f66810763f704ea63f252a923a3f0
0,a,SHA512=d2b6290e0483f4d41191844f3cabfb1928a575569a0c8fabb2cf963cc0522635dc
a64e14ba06e2220cd51bf49c5b84844f1137546d7001176b9b3f80a943657f
```

10.1.3 Streamhash of Each Channel of a Single Stereo Stream in a Compressed Audio File

Streamhashing the left and right channels separately would be applicable when an examiner is attempting to determine if a two-channel file is true stereo (i.e., independent information in the left and right channels) or is dual mono (i.e., the same information in both channels).

```
ffmpeg -loglevel error -i input_stereo.mp3 -filter_complex "[0:a]channelsplit=
channel_layout=stereo[left][right]" -map "[left]" -map "[right]" -f streamhash -
```

`ffmpeg`

Calls the ffmpeg executable.

`-loglevel error`

Tells ffmpeg to limit the logging on the screen to errors only.

`-i input_stereo.mp3`

The name of the input MP3 audio file in the local directory.

`-filter_complex "[0:a]channelsplit=channel_layout=stereo[left][right]"`

Identifies that the stereo audio track of input file "`0`" will be split into two channels labeled "`left`" and "`right`".

`-map "[left]"`

Identifies that the channel labeled "`left`" is to be designated as a separate file for streamhashing.

`-map "[right]"`

Identifies that the channel labeled "`right`" is to be designated as a separate file for streamhashing.

`-f streamhash -`

Tells ffmpeg to compute the SHA-256 hash value of the input channel as defined above, and output the result to the command window, signified by the dash at the end of the argument.

The resulting outputs would appear as separate SHA-256 values, as follows, indicating the source two-channel audio file contains different information in each channel (file "`0`" is the left channel, file "`1`" is the left channel):

```
0,a,SHA256=65e65adbe950583cfa26fc4093ce4599b17a96ce1dbeaebd161a80330afc8154
1,a,SHA256=9de7f0459bd5a9b9d9006212dc5387ee3e656525d8da9f820effea60e651b0f2
```

For a source two-channel audio file containing identical information in both channels, the results would appear as follows (file "`0`" is the left channel, file "`1`" is the left channel):

```
0,a,SHA256=da32d48104f290c64ff0af8d4d5b9d1fd51845db84f37d09fcc98a702a9e7222
1,a,SHA256=da32d48104f290c64ff0af8d4d5b9d1fd51845db84f37d09fcc98a702a9e7222
```

10.1.4 Streamhash of a PCM WAV File Containing Multiple Audio Streams

```
ffmpeg -loglevel error -i input_4ch.wav -map 0 -f streamhash -
```

`ffmpeg`

Calls the ffmpeg executable.

`-loglevel error`

Tells ffmpeg to limit the logging on the screen to errors only.

`-i input_4ch.wav`

The name of the input 4-channel WAV audio file in the local directory.

`-map 0`

Tells ffmpeg to use all stream(s) of the input file. As seen before, "`0`" indicates the first file counting from zero.

`-f streamhash -`

Tells ffmpeg to compute the SHA-256 hash value of the input audio streams on a stream-by-stream basis, and output the result to the command window, signified by the dash at the end of the argument.

The resulting outputs would appear as separate SHA-256 values, one for each audio stream ("0" through "3"), as follows:

```
0,a,SHA256=5a4f59b3e91e19f356d290964378b348d6729cb88fae7d3fd6fd5fd22db32165
1,a,SHA256=1fd199a6463e241c359277424d210b7a8b313217ab4ebe22f1470d2b2a42cdcf
2,a,SHA256=f6d72d317d5b00635684ab6f0a3a62874036cc1aa0f331f978442cfc53a7b94d
3,a,SHA256=28ff640bfb6ac945a8bd99f0e21d746e26dafcc0e66d52c9e42085ef401160a6
```

10.1.5 Streamhash of a Mono, 24-bit PCM WAV File

```
ffmpeg -loglevel error -i input_mono_24bit.wav -c:a pcm_s24le -f streamhash -
```

`ffmpeg`

Calls the ffmpeg executable.

`-loglevel error`

Tells ffmpeg to limit the logging on the screen to errors only.

`-i input_mono_24bit.wav`

The name of the input 24-bit WAV audio file in the local directory.

`-c:a pcm_s24le`

Tells ffmpeg to treat the encoded audio data as signed 24-bit/little endian format. This overrides the default streamhash format of signed 16-bit/little endian for audio streams.

`-f streamhash -`

Tells ffmpeg to compute the streamhash of the input PCM audio stream and output the result to the command window, signified by the dash at the end of the argument.

The resulting output would appear as a single SHA-256 value, as follows:

```
0,a,SHA256=ddce44e9e5d85132d2c3e706c7c2ea5f2ce09126f42050a23754e167f5fbc507
```

If the "`-c:a pcm_s24le`" argument was not used in the command, the resulting output would appear as follows:

```
0,a,SHA256=96b8df582b9e7efe3d326df3e194fda4f619858ddc51624657b4cb8a0e74b2ce
```

Because down-converting non-identical files at a higher quantization than 16-bit to 16-bit may result in matching streamhash values, the appropriate argument for the source quantization must be used to compute accurate streamhash values.

11. References

[1] Wales, Gregory S., et al. "Multimedia Stream Hashing: A Forensic Method for Content Verification." *Journal of Forensic Science*, vol. 68, no. 1, 2023, pp. 289– 300. https://doi.org/10.1111/1556-4029.15148.

12. Additional Resources

- *Ffmpeg*, https://www.ffmpeg.org/. Accessed 13 Jan. 2023.
- Scientific Working Group on Digital Evidence. *Minimum Requirements for Testing Tools Used in Digital and Multimedia Forensics*. SWGDE 18-Q-001-2.1. *SWGDE*, 7 Mar. 2024, https://www.swgde.org/18-q-001-2/.
- Scientific Working Group on Digital Evidence. *Technical Notes on FFmpeg for Forensic Video Examinations*. SWGDE 16-V-002-3.0. *SWGDE*, 22 Mar. 2024, https://www.swgde.org/16-v-002/.

History

Revision	Issue Date	History
1.0 DRAFT	1/13/2023	Add existing rows from the current version of the doc. Check Initial draft created and voted on by SWGDE for release as a draft for public comment.
1.0 DRAFT	6/15/2023	Revisions made to all command window output graphics based on comments received. Resubmitted for SWGDE vote to be released as a draft for public comment.
1.0	9/21/2023	SWGDE voted to approve as a Final Approved Document. Formatted for release as a Final Approved Document.
1.1 DRAFT	9/19/2024	Updated Sections 8 (table), 9.5, 9.6, and 10.1.3 due to the deprecated "-map_channel" switch in FFmpeg version 7. Changed and expanded section 10.1.2 (MD5 example replaced with SHA512 example, multiple streamhash values in a single command). SWGDE voted to approve as a draft for public comment.
1.1 DRAFT	1/16/2025	Revised Sections 3, 5, and 9 in response to public comments. Added and updated references.
1.1	2/21/2025	SWGDE voted to approve as a Final Approved Document.
1.1	2/26/2025	Formatted for release as a Final Approved Document.