Scientific Working Group on Digital Evidence

SWGDE Technical Notes on FFmpeg for Forensic Audio Examination

23-A-001-1.0

Disclaimer Regarding Use of SWGDE Documents

SWGDE documents are developed by a consensus process that involves the best efforts of relevant subject matter experts, organizations, and input from other stakeholders to publish suggested best practices, practical guidance, technical positions, and educational information in the discipline of digital and multimedia forensics and related fields. No warranty or other representation as to SWGDE work product is made or intended.

SWGDE requests notification by e-mail before or contemporaneous to the introduction of this document, or any portion thereof, as a marked exhibit offered for or moved into evidence in such proceeding. The notification should include: 1) The formal name of the proceeding, including docket number or similar identifier; 2) the name and location of the body conducting the hearing or proceeding; and 3) the name, mailing address (if available) and contact information of the party offering or moving the document into evidence. Subsequent to the use of this document in the proceeding please notify SWGDE as to the outcome of the matter. Notifications should be submitted via the SWGDE Notice of Use/Redistribution Form or sent to secretary@swgde.org.

From time to time, SWGDE documents may be revised, updated, or sunsetted. Readers are advised to verify on the SWGDE website (https://www.swgde.org) they are utilizing the current version of this document. Prior versions of SWGDE documents are archived and available on the SWGDE website.

Redistribution Policy

SWGDE grants permission for redistribution and use of all publicly posted documents created by SWGDE, provided that the following conditions are met:

1. Redistribution of documents or parts of documents must retain this SWGDE cover page containing the Disclaimer and Conditions of Use.
2. Neither the name of SWGDE nor the names of contributors may be used to endorse or promote products derived from its documents.
3. Any reference or quote from a SWGDE document must include the version number (or creation date) of the document and also indicate if the document is in a draft status.

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

ARCHIVED

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

Unauthorized use of the SWGDE logo or documents 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.

ARCHIVED

SWGDE Technical Notes on FFmpeg for Forensic Audio Examination

Table of Contents

1. Purpose **5**
2. Scope **5**
3. Limitations **5**
4. FFmpeg Tools **5**
4.1 ffmpeg 6
4.2 ffprobe 6
4.3 ffplay 6
5. FFmpeg Installation **6**
6. FFmpeg informational commands **6**
6.1 Help (-h) 6
6.2 Show license (-L) 6
6.3 Installed codecs (-codecs) 7
6.4 Available formats (-formats) 7
7. Basic command entry format **7**
7.1 ffprobe (basic usage) 7
7.1.1 Explanation of results 8
7.2 ffprobe (stream and format details) 8
7.2.1 Explanation of results (partial): 10
7.3 ffplay (basic usage) 10
8. ffmpeg **10**
9. Commonly used ffmpeg commands in forensic audio examinations **11**
9.1 Conversion of a compressed audio file to a signed 16-bit PCM WAV file 12
9.2 Conversion of a compressed audio file to a 32-bit floating point PCM WAV file 13
9.3 Conversion of an audio file into a losslessly compressed format 13
9.4 Read raw stereo, signed 16-bit PCM/little endian, 44.1 kHz audio data and output to a RIFF WAV file 14
9.5 Split stereo PCM WAV file into mono left and right files 15
9.6 Split stereo compressed audio file into mono left and right files (with conversion to signed 16-bit PCM) 16
9.7 Combine two mono PCM WAV files into a stereo file (with input files having the same length and encoding characteristics) 17
9.8 Concatenate two or more PCM WAV files into a single file (with input files having the

ARCHIVED

same encoding characteristics).....18
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”.....19
10. Validation commands.....20
10.1 Streamhash.....20
10.1.1 Streamhash of a single stereo stream in a compressed audio file.....20
10.1.2 Streamhash of a single stereo stream in a compressed audio file with the output hash algorithm set to Message Digest 5 (MD5).....21
10.1.3 Streamhash of each channel of a single stereo stream in a compressed audio file.22
10.1.4 Streamhash of a PCM WAV file containing multiple audio streams.....23
10.1.5 Streamhash of a mono, 24-bit PCM WAV file.....24
11. References.....25
12. History.....25

ARCHIVED

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 Technical Notes on FFmpeg for Forensic Video Examination* [1] 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 (www.ffmpeg.org) for full documentation. The website documentation refers only to the most current stable version. Therefore, it is recommended that notes be kept on the version implemented within your laboratory.

This is also not an endorsement of FFmpeg to the exclusion of other multimedia processing tools.

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.

ARCHIVED

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 Technical Notes on FFmpeg for Forensic Video Examination* [1] for installation instructions. The latest stable version of FFmpeg should be used in forensic examinations and 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`)

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

6.2 Show license (`-L`)

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

ARCHIVED

6.3 Installed codecs (`-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`)

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).

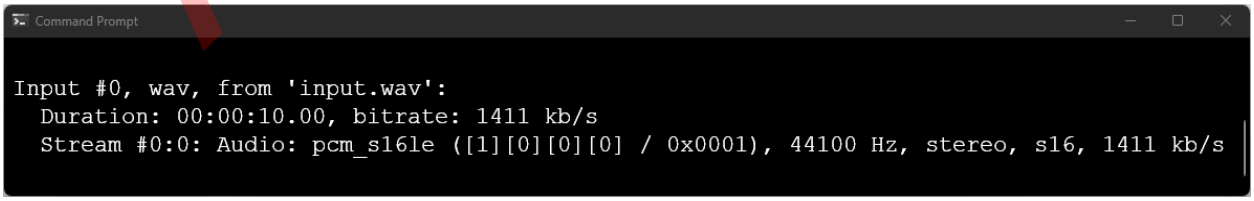
7.1 ffprobe (basic usage)

`ffprobe`

Calls the ffprobe executable.

`input.wav`

The name of the file in the local directory.



ARCHIVED

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.ss)

`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`

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.

ARCHIVED

Command Prompt
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: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
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]

ARCHIVED

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`

Calls the ffplay executable.

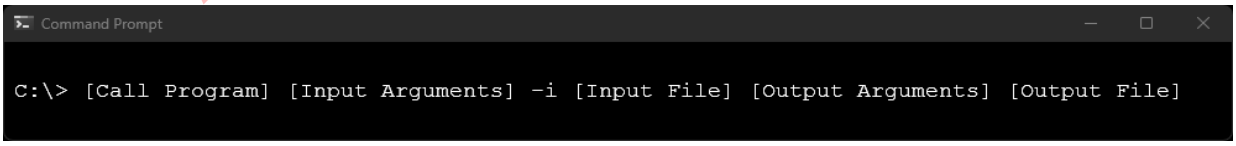
`input.wav`

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:



ARCHIVED

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

`-map_channel 0.0.0`
`-map_channel 0.0.1`
etc.
	Access individual channels of an input audio file, according to "input_file.stream.channel" mapping (channel "0" is the left channel of a stereo file, channel "1" is the right channel).

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

ARCHIVED

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.

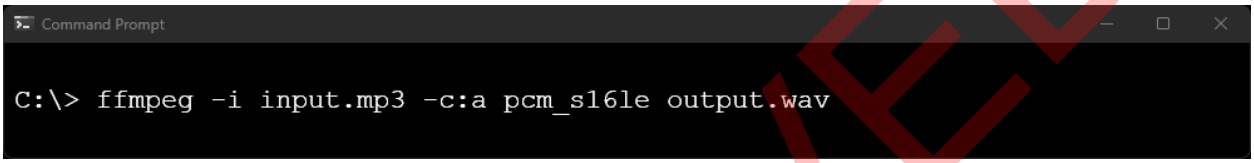
Typically, the channel configuration and sampling rate of an input audio file or stream will be retained for an output audio file or stream, unless explicitly modified by certain arguments within the ffmpeg command. Many of the example commands given below are provided without

input or output arguments (such as channel configuration and sampling rate) and therefore must be verified for use in forensic examinations.

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`

Calls the ffmpeg executable.

`-i input.mp3`

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

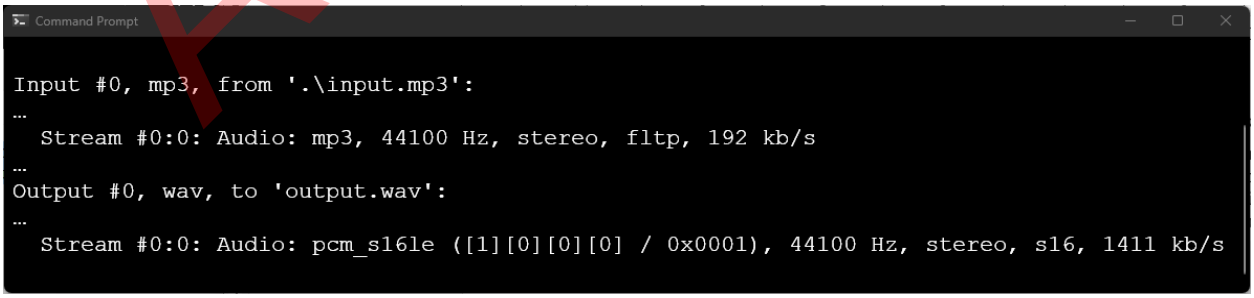
`-c:a pcm_s16le`

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:



ARCHIVED

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`

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 losslessly compressed format

`ffmpeg`

Calls the ffmpeg executable.

`-i input.mp3`

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

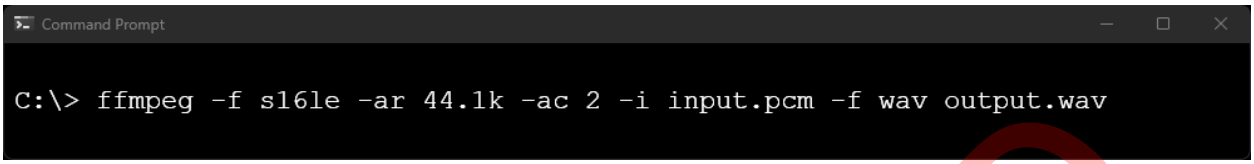
`output.flac`

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.

ARCHIVED

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`

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*, 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.

ARCHIVED

9.5 Split stereo PCM WAV file into mono left and right files

`ffmpeg`

Calls the ffmpeg executable.

`-i input.wav`

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

`-map_channel 0.0.0`

Identifies the channel selected to be output by input file (first "`0`"), the first stream of input file (second "`0`"), and the left channel of the stream (third "`0`").

`-c:a copy`

Copy the audio stream data of the left channel as identified by "`map_channel`". Using the special "`copy`" argument ensures the stream is duplicated from the input to the output without any conversion or other processing. It should be noted when outputting to a WAV file container, if "`-c:a copy`" is not included as an argument then FFmpeg defaults to encoding the audio data as 16-bit little endian PCM, regardless of the encoding characteristics of the input file(s). For example, if your input file contains 32-bit PCM audio data and "`-c:a copy`" is not included as an argument, then the output WAV files will each contain 16-bit PCM audio data.

`output_left.wav`

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

`-map_channel 0.0.1`

Identifies the channel selected to be output by input file (first "`0`"), the first stream of input file (second "`0`"), and the right channel of the stream ("`1`").

`-c:a copy`

Copy the audio stream data of the right channel as identified by "`map_channel`".

`output_right.wav`

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

ARCHIVED

9.6 Split stereo compressed audio file into mono left and right files (with conversion to signed 16-bit PCM)

Rather than copying input stream to the output, as above, in this example the output file is set to a specific bit depth.

`ffmpeg`

Calls the ffmpeg executable.

`-i input.mp3`

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

`-map_channel 0.0.0`

Identifies the channel selected to be output by input file (first "`0`"), the first stream of input file (second "`0`"), and the left channel of the stream (third "`0`").

`-c:a pcm_s16le`

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

`output_left.wav`

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

`-map_channel 0.0.1`

Identifies the channel selected to be output by input file (first "`0`"), the first stream of input file (second "`0`"), and the right channel of the stream ("`1`").

`-c:a pcm_s16le`

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

`output_right.wav`

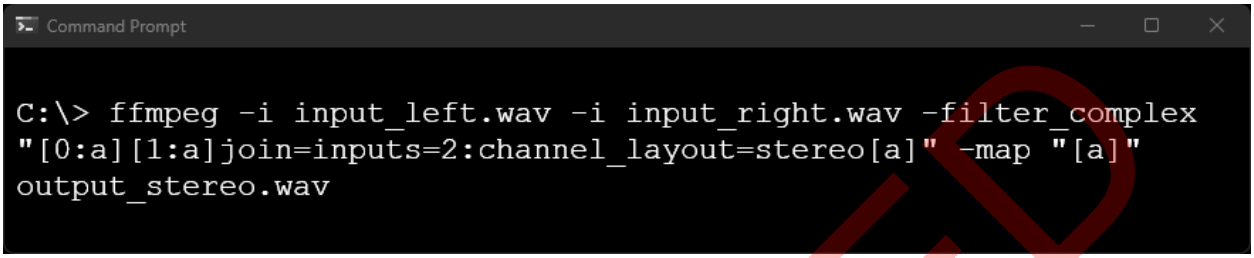
The right channel is output to a file named "`output_right.wav`", which is saved into the local directory.

ARCHIVED

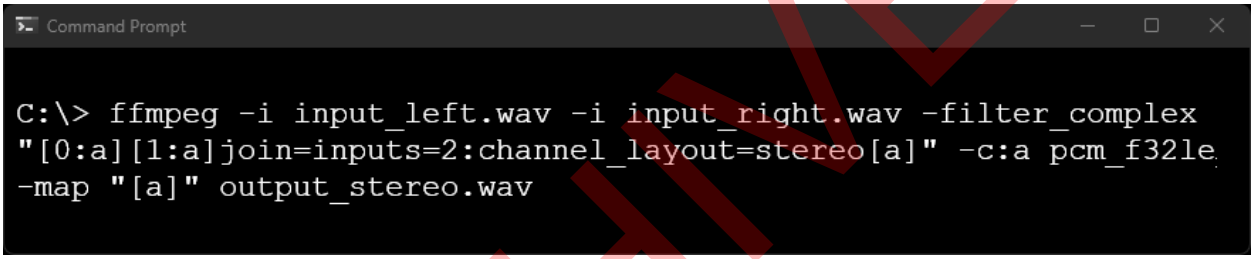
9.7 Combine two mono PCM WAV files into a stereo file (with input files having the same length and encoding characteristics)

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

16-bit example:



32-bit example:



`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 to 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 muti-channel stream is "`[a]`".

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

ARCHIVED

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`

Calls the ffmpeg executable.

`-f concat`

ARCHIVED

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`

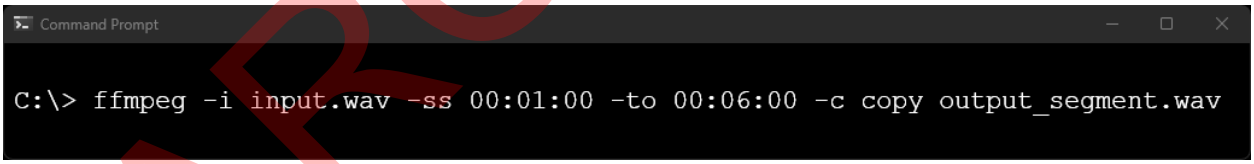
The name and container file format of the output file to be saved into the local directory.

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:

Example 2:



`ffmpeg`

Calls the ffmpeg executable.

`-i input.wav`

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

`-ss 00:01:00`

ARCHIVED

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

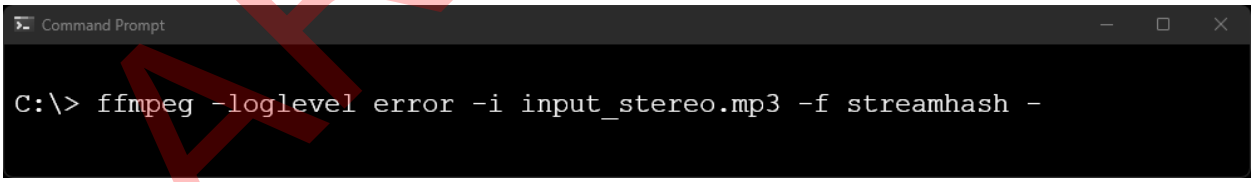
10.1 Streamhash

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) [2].

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.

10.1.1 Streamhash of a single stereo stream in a compressed audio file



`ffmpeg`

Calls the ffmpeg executable.

`-loglevel error`

ARCHIVED

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_stereo.mp3`

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

`-f streamhash -`

Tells ffmpeg to compute the SHA-256 hash of the input 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 would appear as a single SHA-256 value, as follows:

10.1.2 Streamhash of a single stereo stream in a compressed audio file with the output hash algorithm set to Message Digest 5 (MD5)

`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 md5 -`

Tells ffmpeg to utilize the MD5 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 MD5 value, as follows:

ARCHIVED

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).

Channel "0" (left channel) example:

Channel "1" (right channel) example:

`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.

`-map_channel 0.0.0` or `-map_channel 0.0.1`

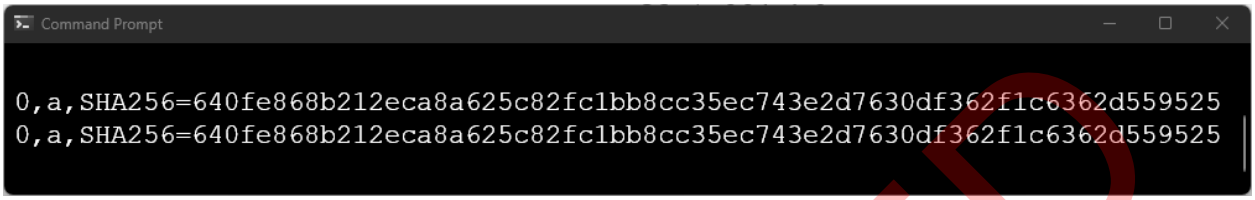
Tells ffmpeg to use file "`0`", stream "`0`", and channel "`0`" (left channel) for the resulting streamhash computation. For the right channel, ffmpeg uses file "`0`", stream "`0`", and channel "`1`".

`-f streamhash -`

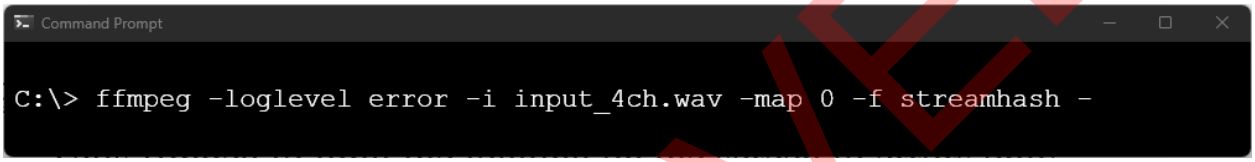
ARCHIVED

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 identical information in both channels:



10.1.4 Streamhash of a PCM WAV file containing multiple audio streams



`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`

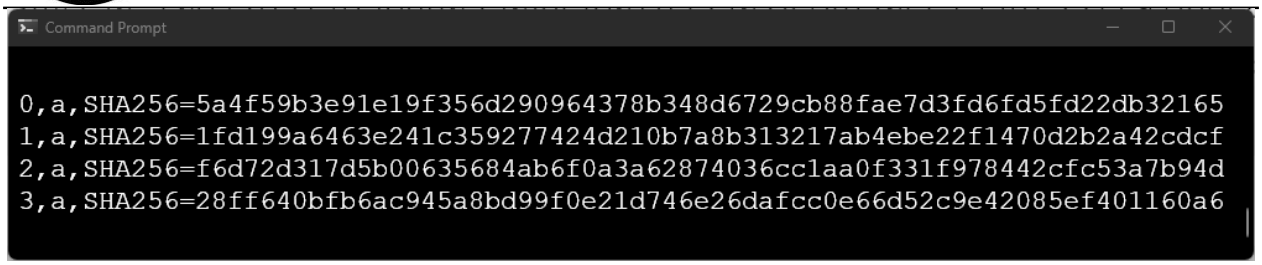
Tells ffmpeg to use all stream(s) of the input file. As seen before, "`0`" indicates the first file counting from zero.

`-f streamhash -`

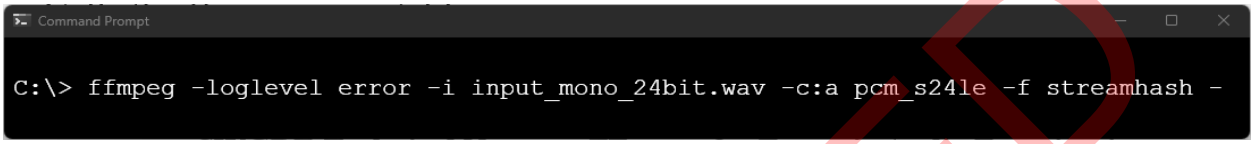
Tells ffmpeg to compute the SHA-256 hash value of the input audio streams on a stream-by-stream basis, and output the result to the command window, signified by the dash at the end of the argument.

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

ARCHIVED



10.1.5 Streamhash of a mono, 24-bit PCM WAV file



`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:

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

ARCHIVED

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. Scientific Working Group on Digital Evidence, *SWGDE Technical Notes on FFmpeg for Forensic Video Examination*, SWGDE, 2023.
2. Wales, GS, Smith, JM, Lacey, DS, Grigoras, C. Multimedia stream hashing: A forensic method for content verification. *J Forensic Sci*. 2023; 68: 289– 300. https://doi.org/10.1111/1556-4029.15148

12. History

Revision	Issue Date	History
1.0 DRAFT	1/13/2023	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.
1.0	9/21/2023	SWGDE voted to approve as a Final Approved Document. Formatted for release as a Final Approved Document.
1.0	10/17/2025	Added ARCHIVED watermark.

ARCHIVED