ffprobe Documentation

Table of Contents

1. Synopsis

ffprobe [*options*] [‘`input_file`’]

2. Description

ffprobe gathers information from multimedia streams and prints it in human- and machine-readable fashion.

For example it can be used to check the format of the container used by a multimedia stream and the format and type of each media stream contained in it.

If a filename is specified in input, ffprobe will try to open and probe the file content. If the file cannot be opened or recognized as a multimedia file, a positive exit code is returned.

ffprobe may be employed both as a standalone application or in combination with a textual filter, which may perform more sophisticated processing, e.g. statistical processing or plotting.

Options are used to list some of the formats supported by ffprobe or for specifying which information to display, and for setting how ffprobe will show it.

ffprobe output is designed to be easily parsable by a textual filter, and consists of one or more sections of a form defined by the selected writer, which is specified by the ‘`print_format`’ option.

Sections may contain other nested sections, and are identified by a name (which may be shared by other sections), and an unique name. See the output of ‘`sections`’.

Metadata tags stored in the container or in the streams are recognized and printed in the corresponding "FORMAT" or "STREAM" section.

3. Options

All the numerical options, if not specified otherwise, accept a string representing a number as input, which may be followed by one of the SI unit prefixes, for example: ’K’, ’M’, or ’G’.

If ’i’ is appended to the SI unit prefix, the complete prefix will be interpreted as a unit prefix for binary multiplies, which are based on powers of 1024 instead of powers of 1000. Appending ’B’ to the SI unit prefix multiplies the value by 8. This allows using, for example: ’KB’, ’MiB’, ’G’ and ’B’ as number suffixes.

Options which do not take arguments are boolean options, and set the corresponding value to true. They can be set to false by prefixing the option name with "no". For example using "-nofoo" will set the boolean option with name "foo" to false.

3.1 Stream specifiers

Some options are applied per-stream, e.g. bitrate or codec. Stream specifiers are used to precisely specify which stream(s) a given option belongs to.

A stream specifier is a string generally appended to the option name and separated from it by a colon. E.g. `-codec:a:1 ac3` contains the `a:1` stream specifier, which matches the second audio stream. Therefore, it would select the ac3 codec for the second audio stream.

A stream specifier can match several streams, so that the option is applied to all of them. E.g. the stream specifier in `-b:a 128k` matches all audio streams.

An empty stream specifier matches all streams. For example, `-codec copy` or `-codec: copy` would copy all the streams without reencoding.

Possible forms of stream specifiers are:

‘*`stream_index`*’

Matches the stream with this index. E.g. `-threads:1 4` would set the thread count for the second stream to 4.

‘`*stream_type*[:*stream_index*]`’

stream_type is one of following: ’v’ for video, ’a’ for audio, ’s’ for subtitle, ’d’ for data, and ’t’ for attachments. If *stream_index* is given, then it matches stream number *stream_index* of this type. Otherwise, it matches all streams of this type.

‘`p:*program_id*[:*stream_index*]`’

If *stream_index* is given, then it matches the stream with number *stream_index* in the program with the id *program_id*. Otherwise, it matches all streams in the program.

‘`#*stream_id*`’

Matches the stream by a format-specific ID.

3.2 Generic options

These options are shared amongst the ff* tools.

‘`-L`’

Show license.

‘`-h, -?, -help, --help [*arg*]`’

Show help. An optional parameter may be specified to print help about a specific item.

Possible values of *arg* are:

‘`decoder=*decoder_name*`’

Print detailed information about the decoder named *decoder_name*. Use the ‘`-decoders`’ option to get a list of all decoders.

‘`encoder=*encoder_name*`’

Print detailed information about the encoder named *encoder_name*. Use the ‘`-encoders`’ option to get a list of all encoders.

‘`demuxer=*demuxer_name*`’

Print detailed information about the demuxer named *demuxer_name*. Use the ‘`-formats`’ option to get a list of all demuxers and muxers.

‘`muxer=*muxer_name*`’

Print detailed information about the muxer named *muxer_name*. Use the ‘`-formats`’ option to get a list of all muxers and demuxers.

‘`filter=`*`filter_name`*’

Print detailed information about the filter name *filter_name*. Use the ‘`-filters`’ option to get a list of all filters.

‘`-version`’

Show version.

‘`-formats`’

Show available formats.

‘`-codecs`’

Show all codecs known to libavcodec.

Note that the term ’codec’ is used throughout this documentation as a shortcut for what is more correctly called a media bitstream format.

‘`-decoders`’

Show available decoders.

‘`-encoders`’

Show all available encoders.

‘`-bsfs`’

Show available bitstream filters.

‘`-protocols`’

Show available protocols.

‘`-filters`’

Show available libavfilter filters.

‘`-pix_fmts`’

Show available pixel formats.

‘`-sample_fmts`’

Show available sample formats.

‘`-layouts`’

Show channel names and standard channel layouts.

‘`-loglevel [repeat+]`*`loglevel`*` | -v [repeat+]`*`loglevel`*’

Set the logging level used by the library. Adding "repeat+" indicates that repeated log output should not be compressed to the first line and the "Last message repeated n times" line will be omitted. "repeat" can also be used alone. If "repeat" is used alone, and with no prior loglevel set, the default loglevel will be used. If multiple loglevel parameters are given, using ’repeat’ will not change the loglevel. *loglevel* is a number or a string containing one of the following values:

‘`quiet`’

Show nothing at all; be silent.

‘`panic`’

Only show fatal errors which could lead the process to crash, such as and assert failure. This is not currently used for anything.

‘`fatal`’

Only show fatal errors. These are errors after which the process absolutely cannot continue after.

‘`error`’

Show all errors, including ones which can be recovered from.

‘`warning`’

Show all warnings and errors. Any message related to possibly incorrect or unexpected events will be shown.

‘`info`’

Show informative messages during processing. This is in addition to warnings and errors. This is the default value.

‘`verbose`’

Same as `info`, except more verbose.

‘`debug`’

Show everything, including debugging information.

By default the program logs to stderr, if coloring is supported by the terminal, colors are used to mark errors and warnings. Log coloring can be disabled setting the environment variable `AV_LOG_FORCE_NOCOLOR` or `NO_COLOR`, or can be forced setting the environment variable

`AV_LOG_FORCE_COLOR`. The use of the environment variable `NO_COLOR` is deprecated and will be dropped in a following FFmpeg version.

‘`-report`’

Dump full command line and console output to a file named `*program*-*YYYYMMDD*-*HHMMSS*.log` in the current directory. This file can be useful for bug reports. It also implies `-loglevel verbose`.

Setting the environment variable `FFREPORT` to any value has the same effect. If the value is a ’:’-separated key=value sequence, these options will affect the report; options values must be escaped if they contain special characters or the options delimiter ’:’ (see the “Quoting and escaping” section in the ffmpeg-utils manual). The following option is recognized:

‘`file`’

set the file name to use for the report; `%p` is expanded to the name of the program, `%t` is expanded to a timestamp, `%%` is expanded to a plain `%`

Errors in parsing the environment variable are not fatal, and will not appear in the report.

‘`-cpuflags flags (*global*)`’

Allows setting and clearing cpu flags. This option is intended for testing. Do not use it unless you know what you’re doing.

```
ffmpeg -cpuflags -sse+mmx ...
ffmpeg -cpuflags mmx ...
ffmpeg -cpuflags 0 ...
```

Possible flags for this option are:

‘`x86`’
- ‘`mmx`’
- ‘`mmxext`’
- ‘`sse`’
- ‘`sse2`’
- ‘`sse2slow`’
- ‘`sse3`’
- ‘`sse3slow`’
- ‘`ssse3`’
- ‘`atom`’
- ‘`sse4.1`’
- ‘`sse4.2`’
- ‘`avx`’
- ‘`xop`’

        ‘`fma4`’
        ‘`3dnow`’
        ‘`3dnowext`’
        ‘`cmov`’
    ‘`ARM`’
        ‘`armv5te`’
        ‘`armv6`’
        ‘`armv6t2`’
        ‘`vfp`’
        ‘`vfpv3`’
        ‘`neon`’
    ‘`PowerPC`’
        ‘`altivec`’
    ‘`Specific Processors`’
        ‘`pentium2`’
        ‘`pentium3`’
        ‘`pentium4`’
        ‘`k6`’
        ‘`k62`’
        ‘`athlon`’
        ‘`athlonxp`’
        ‘`k8`’

‘`-opencl_options options (`*`global`*`)`’

Set OpenCL environment options. This option is only available when FFmpeg has been compiled with `--enable-opencl`.

options must be a list of *key*=*value* option pairs separated by ’:’. See the “OpenCL Options” section in the ffmpeg-utils manual for the list of supported options.

3.3 AVOptions

These options are provided directly by the libavformat, libavdevice and libavcodec libraries. To see the list of available AVOptions, use the ‘`-help`’ option. They are separated into two categories:

‘`generic`’

These options can be set for any container, codec or device. Generic options are listed under AVFormatContext options for containers/devices and under AVCodecContext options for codecs.

‘`private`’

These options are specific to the given container, device or codec. Private options are listed under their corresponding containers/devices/codecs.

For example to write an ID3v2.3 header instead of a default ID3v2.4 to an MP3 file, use the ‘`id3v2_version`’ private option of the MP3 muxer:

```
ffmpeg -i input.flac -id3v2_version 3 out.mp3
```

All codec AVOptions are obviously per-stream, so the chapter on stream specifiers applies to them

Note ‘`-nooption`’ syntax cannot be used for boolean AVOptions, use ‘`-option 0`’/‘`-option 1`’.

Note2 old undocumented way of specifying per-stream AVOptions by prepending v/a/s to the options name is now obsolete and will be removed soon.

3.4 Main options

‘`-f` *`format`*’

Force format to use.

‘`-unit`’

Show the unit of the displayed values.

‘`-prefix`’

Use SI prefixes for the displayed values. Unless the "-byte_binary_prefix" option is used all the prefixes are decimal.

‘`-byte_binary_prefix`’

Force the use of binary prefixes for byte values.

‘`-sexagesimal`’

Use sexagesimal format HH:MM:SS.MICROSECONDS for time values.

‘`-pretty`’

Prettify the format of the displayed values, it corresponds to the options "-unit -prefix -byte_binary_prefix -sexagesimal".

‘`-of, -print_format` *`writer_name`*`[=`*`writer_options`*`]`’

Set the output printing format.

writer_name specifies the name of the writer, and *writer_options* specifies the options to be passed to the writer.

For example for printing the output in JSON format, specify:

```
-print_format json
```

For more details on the available output printing formats, see the Writers section below.

‘`-sections`’

Print sections structure and section information, and exit. The output is not meant to be parsed by a machine.

‘`-select_streams` *`stream_specifier`*’

Select only the streams specified by *stream_specifier*. This option affects only the options related to streams (e.g. `show_streams`, `show_packets`, etc.).

For example to show only audio streams, you can use the command:

```
ffprobe -show_streams -select_streams a INPUT
```

To show only video packets belonging to the video stream with index 1:

```
ffprobe -show_packets -select_streams v:1 INPUT
```

‘`-show_data`’

Show payload data, as an hexadecimal and ASCII dump. Coupled with ‘`-show_packets`’, it will dump the packets’ data. Coupled with ‘`-show_streams`’, it will dump the codec extradata.

The dump is printed as the "data" field. It may contain newlines.

‘`-show_error`’

Show information about the error found when trying to probe the input.

The error information is printed within a section with name "ERROR".

‘`-show_format`’

Show information about the container format of the input multimedia stream.

All the container format information is printed within a section with name "FORMAT".

‘`-show_format_entry` *`name`*’

Like '`-show_format`', but only prints the specified entry of the container format information, rather than all. This option may be given more than once, then all specified entries will be shown.

This option is deprecated, use `show_entries` instead.

'`-show_entries` *`section_entries`*'

Set list of entries to show.

Entries are specified according to the following syntax. *section_entries* contains a list of section entries separated by `:`. Each section entry is composed by a section name (or unique name), optionally followed by a list of entries local to that section, separated by `,`.

If section name is specified but is followed by no `=`, all entries are printed to output, together with all the contained sections. Otherwise only the entries specified in the local section entries list are printed. In particular, if `=` is specified but the list of local entries is empty, then no entries will be shown for that section.

Note that the order of specification of the local section entries is not honored in the output, and the usual display order will be retained.

The formal syntax is given by:

```
LOCAL_SECTION_ENTRIES ::= SECTION_ENTRY_NAME[,LOCAL_SECTION_ENTRIES]
SECTION_ENTRY         ::= SECTION_NAME[=[LOCAL_SECTION_ENTRIES]]
SECTION_ENTRIES       ::= SECTION_ENTRY[:SECTION_ENTRIES]
```

For example, to show only the index and type of each stream, and the PTS time, duration time, and stream index of the packets, you can specify the argument:

```
packet=pts_time,duration_time,stream_index : stream=index,codec_type
```

To show all the entries in the section "format", but only the codec type in the section "stream", specify the argument:

```
format : stream=codec_type
```

To show all the tags in the stream and format sections:

```
format_tags : format_tags
```

To show only the `title` tag (if available) in the stream sections:

```
stream_tags=title
```

‘`-show_packets`’

Show information about each packet contained in the input multimedia stream.

The information for each single packet is printed within a dedicated section with name "PACKET".

‘`-show_frames`’

Show information about each frame contained in the input multimedia stream.

The information for each single frame is printed within a dedicated section with name "FRAME".

‘`-show_streams`’

Show information about each media stream contained in the input multimedia stream.

Each media stream information is printed within a dedicated section with name "STREAM".

‘`-show_chapters`’

Show information about chapters stored in the format.

Each chapter is printed within a dedicated section with name "CHAPTER".

‘`-count_frames`’

Count the number of frames per stream and report it in the corresponding stream section.

‘`-count_packets`’

Count the number of packets per stream and report it in the corresponding stream section.

‘`-show_private_data, -private`’

Show private data, that is data depending on the format of the particular shown element. This option is enabled by default, but you may need to disable it for specific uses, for example when creating XSD-compliant XML output.

‘`-show_program_version`’

Show information related to program version.

Version information is printed within a section with name "PROGRAM_VERSION".

‘`-show_library_versions`’

Show information related to library versions.

Version information for each library is printed within a section with name "LIBRARY_VERSION".

‘`-show_versions`’

Show information related to program and library versions. This is the equivalent of setting both ‘`-show_program_version`’ and ‘`-show_library_versions`’ options.

‘`-bitexact`’

Force bitexact output, useful to produce output which is not dependent on the specific build.

‘`-i` *`input_file`*’

Read *input_file*.

4. Writers

A writer defines the output format adopted by `ffprobe`, and will be used for printing all the parts of the output.

A writer may accept one or more arguments, which specify the options to adopt. The options are specified as a list of *key*=*value* pairs, separated by ":".

A description of the currently available writers follows.

4.1 default

Default format.

Print each section in the form:

```
[SECTION]
key1=val1
...
keyN=valN
[/SECTION]
```

Metadata tags are printed as a line in the corresponding FORMAT or STREAM section, and are prefixed by the string "TAG:".

A description of the accepted options follows.

‘`nokey, nk`’

If set to 1 specify not to print the key of each field. Default value is 0.

‘`noprint_wrappers, nw`’

If set to 1 specify not to print the section header and footer. Default value is 0.

4.2 compact, csv

Compact and CSV format.

The `csv` writer is equivalent to `compact`, but supports different defaults.

Each section is printed on a single line. If no option is specifid, the output has the form:

```
section|key1=val1| ... |keyN=valN
```

Metadata tags are printed in the corresponding "format" or "stream" section. A metadata tag key, if printed, is prefixed by the string "tag:".

The description of the accepted options follows.

‘`item_sep, s`’

Specify the character to use for separating fields in the output line. It must be a single printable character, it is "|" by default ("," for the `csv` writer).

‘`nokey, nk`’

If set to 1 specify not to print the key of each field. Its default value is 0 (1 for the `csv` writer).

‘`escape, e`’

Set the escape mode to use, default to "c" ("csv" for the `csv` writer).

It can assume one of the following values:

‘`c`’

Perform C-like escaping. Strings containing a newline (’\n’), carriage return (’\r’), a tab (’\t’), a form feed (’\f’), the escaping character (’\’) or the item separator character *SEP* are escaped using C-like fashioned escaping, so that a newline is converted to the sequence "\n", a carriage return to "\r", ’\’ to "\\" and the separator *SEP* is converted to "*SEP*".

‘`csv`’

Perform CSV-like escaping, as described in RFC4180. Strings containing a newline (’\n’), a carriage return (’\r’), a double quote (’"’), or *SEP* are enclosed in double-quotes.

‘`none`’

Perform no escaping.

‘`print_section, p`’

Print the section name at the begin of each line if the value is `1`, disable it with value set to `0`. Default value is `1`.

4.3 flat

Flat format.

A free-form output where each line contains an explicit key=value, such as "streams.stream.3.tags.foo=bar". The output is shell escaped, so it can be directly embedded in sh scripts as long as the separator character is an alphanumeric character or an underscore (see *sep_char* option).

The description of the accepted options follows.

‘`sep_char, s`’

Separator character used to separate the chapter, the section name, IDs and potential tags in the printed field key.

Default value is ’.’.

‘`hierarchical, h`’

Specify if the section name specification should be hierarchical. If set to 1, and if there is more than one section in the current chapter, the section name will be prefixed by the name of the chapter. A value of 0 will disable this behavior.

Default value is 1.

4.4 ini

INI format output.

Print output in an INI based format.

The following conventions are adopted:

- all key and values are UTF-8
- ’.’ is the subgroup separator
- newline, ’\t’, ’\f’, ’\b’ and the following characters are escaped
- ’\’ is the escape character
- ’#’ is the comment indicator
- ’=’ is the key/value separator

- ’:’ is not used but usually parsed as key/value separator

This writer accepts options as a list of *key*=*value* pairs, separated by ":".

The description of the accepted options follows.

‘`hierarchical, h`’

> Specify if the section name specification should be hierarchical. If set to 1, and if there is more than one section in the current chapter, the section name will be prefixed by the name of the chapter. A value of 0 will disable this behavior.
>
> Default value is 1.

4.5 json

JSON based format.

Each section is printed using JSON notation.

The description of the accepted options follows.

‘`compact, c`’

> If set to 1 enable compact output, that is each section will be printed on a single line. Default value is 0.

For more information about JSON, see http://www.json.org/.

4.6 xml

XML based format.

The XML output is described in the XML schema description file ‘`ffprobe.xsd`’ installed in the FFmpeg datadir.

An updated version of the schema can be retrieved at the url http://www.ffmpeg.org/schema/ffprobe.xsd, which redirects to the latest schema committed into the FFmpeg development source code tree.

Note that the output issued will be compliant to the ‘`ffprobe.xsd`’ schema only when no special global output options (‘`unit`’, ‘`prefix`’, ‘`byte_binary_prefix`’, ‘`sexagesimal`’ etc.) are specified.

The description of the accepted options follows.

‘`fully_qualified, q`’

If set to 1 specify if the output should be fully qualified. Default value is 0. This is required for generating an XML file which can be validated through an XSD file.

‘`xsd_compliant, x`’

If set to 1 perform more checks for ensuring that the output is XSD compliant. Default value is 0. This option automatically sets ‘`fully_qualified`’ to 1.

For more information about the XML format, see http://www.w3.org/XML/.

5. Timecode

`ffprobe` supports Timecode extraction:

- MPEG1/2 timecode is extracted from the GOP, and is available in the video stream details (‘`-show_streams`’, see *timecode*).
- MOV timecode is extracted from tmcd track, so is available in the tmcd stream metadata (‘`-show_streams`’, see *TAG:timecode*).
- DV, GXF and AVI timecodes are available in format metadata (‘`-show_format`’, see *TAG:timecode*).

6. See Also

ffprobe-all, ffmpeg, ffplay, ffserver, ffmpeg-utils, ffmpeg-scaler, ffmpeg-resampler, ffmpeg-codecs, ffmpeg-bitstream-filters, ffmpeg-formats, ffmpeg-devices, ffmpeg-protocols, ffmpeg-filters

7. Authors

The FFmpeg developers.

For details about the authorship, see the Git history of the project (git://source.ffmpeg.org/ffmpeg), e.g. by typing the command `git log` in the FFmpeg source directory, or browsing the online repository at http://source.ffmpeg.org.

Maintainers for the specific components are listed in the file ‘`MAINTAINERS`’ in the source code tree.

This document was generated by *john* on *September 24, 2013* using *texi2html 1.82*.

ffprobe Documentation

Table of Contents

1. Synopsis

ffprobe [*options*] [‘`input_file`’]

2. Description

ffprobe gathers information from multimedia streams and prints it in human- and machine-readable fashion.

For example it can be used to check the format of the container used by a multimedia stream and the format and type of each media stream contained in it.

If a filename is specified in input, ffprobe will try to open and probe the file content. If the file cannot be opened or recognized as a multimedia file, a positive exit code is returned.

ffprobe may be employed both as a standalone application or in combination with a textual filter, which may perform more sophisticated processing, e.g. statistical processing or plotting.

Options are used to list some of the formats supported by ffprobe or for specifying which information to display, and for setting how ffprobe will show it.

ffprobe output is designed to be easily parsable by a textual filter, and consists of one or more sections of a form defined by the selected writer, which is specified by the ‘`print_format`’ option.

Sections may contain other nested sections, and are identified by a name (which may be shared by other sections), and an unique name. See the output of ‘`sections`’.

Metadata tags stored in the container or in the streams are recognized and printed in the corresponding "FORMAT" or "STREAM" section.

3. Options

All the numerical options, if not specified otherwise, accept a string representing a number as input, which may be followed by one of the SI unit prefixes, for example: ’K’, ’M’, or ’G’.

If ’i’ is appended to the SI unit prefix, the complete prefix will be interpreted as a unit prefix for binary multiplies, which are based on powers of 1024 instead of powers of 1000. Appending ’B’ to the SI unit prefix multiplies the value by 8. This allows using, for example: ’KB’, ’MiB’, ’G’ and ’B’ as number suffixes.

Options which do not take arguments are boolean options, and set the corresponding value to true. They can be set to false by prefixing the option name with "no". For example using "-nofoo" will set the boolean option with name "foo" to false.

3.1 Stream specifiers

Some options are applied per-stream, e.g. bitrate or codec. Stream specifiers are used to precisely specify which stream(s) a given option belongs to.

A stream specifier is a string generally appended to the option name and separated from it by a colon. E.g. `-codec:a:1 ac3` contains the `a:1` stream specifier, which matches the second audio stream. Therefore, it would select the ac3 codec for the second audio stream.

A stream specifier can match several streams, so that the option is applied to all of them. E.g. the stream specifier in `-b:a 128k` matches all audio streams.

An empty stream specifier matches all streams. For example, `-codec copy` or `-codec: copy` would copy all the streams without reencoding.

Possible forms of stream specifiers are:

‘*`stream_index`*’

> Matches the stream with this index. E.g. `-threads:1 4` would set the thread count for the second stream to 4.

‘`stream_type[:stream_index]`’

stream_type is one of following: ’v’ for video, ’a’ for audio, ’s’ for subtitle, ’d’ for data, and ’t’ for attachments. If *stream_index* is given, then it matches stream number *stream_index* of this type. Otherwise, it matches all streams of this type.

‘`p:program_id[:stream_index]`’

If *stream_index* is given, then it matches the stream with number *stream_index* in the program with the id *program_id*. Otherwise, it matches all streams in the program.

‘`#stream_id`’

Matches the stream by a format-specific ID.

3.2 Generic options

These options are shared amongst the ff* tools.

‘`-L`’

Show license.

‘`-h, -?, -help, --help [arg]`’

Show help. An optional parameter may be specified to print help about a specific item.

Possible values of *arg* are:

‘`decoder=decoder_name`’

Print detailed information about the decoder named *decoder_name*. Use the ‘`-decoders`’ option to get a list of all decoders.

‘`encoder=encoder_name`’

Print detailed information about the encoder named *encoder_name*. Use the ‘`-encoders`’ option to get a list of all encoders.

‘`demuxer=demuxer_name`’

Print detailed information about the demuxer named *demuxer_name*. Use the ‘`-formats`’ option to get a list of all demuxers and muxers.

‘`muxer=muxer_name`’

Print detailed information about the muxer named *muxer_name*. Use the ‘`-formats`’ option to get a list of all muxers and demuxers.

‘`filter=`*`filter_name`*’

Print detailed information about the filter name *filter_name*. Use the ‘`-filters`’ option to get a list of all filters.

‘`-version`’

Show version.

‘`-formats`’

Show available formats.

‘`-codecs`’

Show all codecs known to libavcodec.

Note that the term ’codec’ is used throughout this documentation as a shortcut for what is more correctly called a media bitstream format.

‘`-decoders`’

Show available decoders.

‘`-encoders`’

Show all available encoders.

‘`-bsfs`’

Show available bitstream filters.

‘`-protocols`’

Show available protocols.

‘`-filters`’

Show available libavfilter filters.

‘`-pix_fmts`’

Show available pixel formats.

‘`-sample_fmts`’

Show available sample formats.

‘`-layouts`’

Show channel names and standard channel layouts.

‘`-loglevel [repeat+]`*`loglevel`*` | -v [repeat+]`*`loglevel`*’

Set the logging level used by the library. Adding "repeat+" indicates that repeated log output should not be compressed to the first line and the "Last message repeated n times" line will be omitted. "repeat" can also be used alone. If "repeat" is used alone, and with no prior loglevel set, the default loglevel will be used. If multiple loglevel parameters are given, using ’repeat’ will not change the loglevel. *loglevel* is a number or a string containing one of the following values:

‘`quiet`’

Show nothing at all; be silent.

‘`panic`’

Only show fatal errors which could lead the process to crash, such as and assert failure. This is not currently used for anything.

‘`fatal`’

Only show fatal errors. These are errors after which the process absolutely cannot continue after.

‘`error`’

Show all errors, including ones which can be recovered from.

‘`warning`’

Show all warnings and errors. Any message related to possibly incorrect or unexpected events will be shown.

‘`info`’

Show informative messages during processing. This is in addition to warnings and errors. This is the default value.

‘`verbose`’

Same as `info`, except more verbose.

‘`debug`’

Show everything, including debugging information.

By default the program logs to stderr, if coloring is supported by the terminal, colors are used to mark errors and warnings. Log coloring can be disabled setting the environment variable `AV_LOG_FORCE_NOCOLOR` or `NO_COLOR`, or can be forced setting the environment variable

`AV_LOG_FORCE_COLOR`. The use of the environment variable `NO_COLOR` is deprecated and will be dropped in a following FFmpeg version.

‘`-report`’

Dump full command line and console output to a file named `*program*-*YYYYMMDD*-*HHMMSS*.log` in the current directory. This file can be useful for bug reports. It also implies `-loglevel verbose`.

Setting the environment variable `FFREPORT` to any value has the same effect. If the value is a ’:’-separated key=value sequence, these options will affect the report; options values must be escaped if they contain special characters or the options delimiter ’:’ (see the “Quoting and escaping” section in the ffmpeg-utils manual). The following option is recognized:

‘`file`’

set the file name to use for the report; `%p` is expanded to the name of the program, `%t` is expanded to a timestamp, `%%` is expanded to a plain `%`

Errors in parsing the environment variable are not fatal, and will not appear in the report.

‘`-cpuflags flags (`*`global`*`)`’

Allows setting and clearing cpu flags. This option is intended for testing. Do not use it unless you know what you’re doing.

```
ffmpeg -cpuflags -sse+mmx ...
ffmpeg -cpuflags mmx ...
ffmpeg -cpuflags 0 ...
```

Possible flags for this option are:

‘`x86`’
- ‘`mmx`’
- ‘`mmxext`’
- ‘`sse`’
- ‘`sse2`’
- ‘`sse2slow`’
- ‘`sse3`’
- ‘`sse3slow`’
- ‘`ssse3`’
- ‘`atom`’
- ‘`sse4.1`’
- ‘`sse4.2`’
- ‘`avx`’
- ‘`xop`’

        ‘`fma4`’
        ‘`3dnow`’
        ‘`3dnowext`’
        ‘`cmov`’
    ‘`ARM`’
        ‘`armv5te`’
        ‘`armv6`’
        ‘`armv6t2`’
        ‘`vfp`’
        ‘`vfpv3`’
        ‘`neon`’
    ‘`PowerPC`’
        ‘`altivec`’
    ‘`Specific Processors`’
        ‘`pentium2`’
        ‘`pentium3`’
        ‘`pentium4`’
        ‘`k6`’
        ‘`k62`’
        ‘`athlon`’
        ‘`athlonxp`’
        ‘`k8`’

‘`-opencl_options options (`*`global`*`)`’

> Set OpenCL environment options. This option is only available when FFmpeg has been compiled with `--enable-opencl`.
>
> *options* must be a list of *key*=*value* option pairs separated by ’:’. See the “OpenCL Options” section in the ffmpeg-utils manual for the list of supported options.

3.3 AVOptions

These options are provided directly by the libavformat, libavdevice and libavcodec libraries. To see the list of available AVOptions, use the ‘`-help`’ option. They are separated into two categories:

‘`generic`’

> These options can be set for any container, codec or device. Generic options are listed under AVFormatContext options for containers/devices and under AVCodecContext options for codecs.

‘`private`’

> These options are specific to the given container, device or codec. Private options are listed under their corresponding containers/devices/codecs.

For example to write an ID3v2.3 header instead of a default ID3v2.4 to an MP3 file, use the '`id3v2_version`' private option of the MP3 muxer:

```
ffmpeg -i input.flac -id3v2_version 3 out.mp3
```

All codec AVOptions are obviously per-stream, so the chapter on stream specifiers applies to them

Note '`-nooption`' syntax cannot be used for boolean AVOptions, use '`-option 0`'/'`-option 1`'.

Note2 old undocumented way of specifying per-stream AVOptions by prepending v/a/s to the options name is now obsolete and will be removed soon.

3.4 Main options

'`-f` *`format`*'

Force format to use.

'`-unit`'

Show the unit of the displayed values.

'`-prefix`'

Use SI prefixes for the displayed values. Unless the "-byte_binary_prefix" option is used all the prefixes are decimal.

'`-byte_binary_prefix`'

Force the use of binary prefixes for byte values.

'`-sexagesimal`'

Use sexagesimal format HH:MM:SS.MICROSECONDS for time values.

'`-pretty`'

Prettify the format of the displayed values, it corresponds to the options "-unit -prefix -byte_binary_prefix -sexagesimal".

'`-of, -print_format` *`writer_name`*`[=`*`writer_options`*`]`'

Set the output printing format.

writer_name specifies the name of the writer, and *writer_options* specifies the options to be passed to the writer.

For example for printing the output in JSON format, specify:

```
-print_format json
```

For more details on the available output printing formats, see the Writers section below.

‘`-sections`’

Print sections structure and section information, and exit. The output is not meant to be parsed by a machine.

‘`-select_streams` *`stream_specifier`*’

Select only the streams specified by *stream_specifier*. This option affects only the options related to streams (e.g. `show_streams`, `show_packets`, etc.).

For example to show only audio streams, you can use the command:

```
ffprobe -show_streams -select_streams a INPUT
```

To show only video packets belonging to the video stream with index 1:

```
ffprobe -show_packets -select_streams v:1 INPUT
```

‘`-show_data`’

Show payload data, as an hexadecimal and ASCII dump. Coupled with ‘`-show_packets`’, it will dump the packets’ data. Coupled with ‘`-show_streams`’, it will dump the codec extradata.

The dump is printed as the "data" field. It may contain newlines.

‘`-show_error`’

Show information about the error found when trying to probe the input.

The error information is printed within a section with name "ERROR".

‘`-show_format`’

Show information about the container format of the input multimedia stream.

All the container format information is printed within a section with name "FORMAT".

‘`-show_format_entry` *`name`*’

Like '`-show_format`', but only prints the specified entry of the container format information, rather than all. This option may be given more than once, then all specified entries will be shown.

This option is deprecated, use `show_entries` instead.

'`-show_entries` *`section_entries`*'

Set list of entries to show.

Entries are specified according to the following syntax. *section_entries* contains a list of section entries separated by `:`. Each section entry is composed by a section name (or unique name), optionally followed by a list of entries local to that section, separated by `,`.

If section name is specified but is followed by no `=`, all entries are printed to output, together with all the contained sections. Otherwise only the entries specified in the local section entries list are printed. In particular, if `=` is specified but the list of local entries is empty, then no entries will be shown for that section.

Note that the order of specification of the local section entries is not honored in the output, and the usual display order will be retained.

The formal syntax is given by:

```
LOCAL_SECTION_ENTRIES ::= SECTION_ENTRY_NAME[,LOCAL_SECTION_ENTRIES]
SECTION_ENTRY         ::= SECTION_NAME[=[LOCAL_SECTION_ENTRIES]]
SECTION_ENTRIES       ::= SECTION_ENTRY[:SECTION_ENTRIES]
```

For example, to show only the index and type of each stream, and the PTS time, duration time, and stream index of the packets, you can specify the argument:

```
packet=pts_time,duration_time,stream_index : stream=index,codec_type
```

To show all the entries in the section "format", but only the codec type in the section "stream", specify the argument:

```
format : stream=codec_type
```

To show all the tags in the stream and format sections:

```
format_tags : format_tags
```

To show only the `title` tag (if available) in the stream sections:

```
stream_tags=title
```

‘`-show_packets`’

Show information about each packet contained in the input multimedia stream.

The information for each single packet is printed within a dedicated section with name "PACKET".

‘`-show_frames`’

Show information about each frame contained in the input multimedia stream.

The information for each single frame is printed within a dedicated section with name "FRAME".

‘`-show_streams`’

Show information about each media stream contained in the input multimedia stream.

Each media stream information is printed within a dedicated section with name "STREAM".

‘`-show_chapters`’

Show information about chapters stored in the format.

Each chapter is printed within a dedicated section with name "CHAPTER".

‘`-count_frames`’

Count the number of frames per stream and report it in the corresponding stream section.

‘`-count_packets`’

Count the number of packets per stream and report it in the corresponding stream section.

‘`-show_private_data, -private`’

Show private data, that is data depending on the format of the particular shown element. This option is enabled by default, but you may need to disable it for specific uses, for example when creating XSD-compliant XML output.

‘`-show_program_version`’

Show information related to program version.

Version information is printed within a section with name "PROGRAM_VERSION".

‘`-show_library_versions`’

Show information related to library versions.

Version information for each library is printed within a section with name "LIBRARY_VERSION".

‘`-show_versions`’

Show information related to program and library versions. This is the equivalent of setting both ‘`-show_program_version`’ and ‘`-show_library_versions`’ options.

‘`-bitexact`’

Force bitexact output, useful to produce output which is not dependent on the specific build.

‘`-i` *`input_file`*’

Read *input_file*.

4. Writers

A writer defines the output format adopted by `ffprobe`, and will be used for printing all the parts of the output.

A writer may accept one or more arguments, which specify the options to adopt. The options are specified as a list of *key*=*value* pairs, separated by ":".

A description of the currently available writers follows.

4.1 default

Default format.

Print each section in the form:

```
[SECTION]
key1=val1
...
keyN=valN
[/SECTION]
```

Metadata tags are printed as a line in the corresponding FORMAT or STREAM section, and are prefixed by the string "TAG:".

A description of the accepted options follows.

‘`nokey, nk`’

If set to 1 specify not to print the key of each field. Default value is 0.

‘`noprint_wrappers, nw`’

If set to 1 specify not to print the section header and footer. Default value is 0.

4.2 compact, csv

Compact and CSV format.

The `csv` writer is equivalent to `compact`, but supports different defaults.

Each section is printed on a single line. If no option is specifid, the output has the form:

```
section|key1=val1| ... |keyN=valN
```

Metadata tags are printed in the corresponding "format" or "stream" section. A metadata tag key, if printed, is prefixed by the string "tag:".

The description of the accepted options follows.

‘`item_sep, s`’

Specify the character to use for separating fields in the output line. It must be a single printable character, it is "|" by default ("," for the `csv` writer).

‘`nokey, nk`’

If set to 1 specify not to print the key of each field. Its default value is 0 (1 for the `csv` writer).

‘`escape, e`’

Set the escape mode to use, default to "c" ("csv" for the `csv` writer).

It can assume one of the following values:

‘`c`’

Perform C-like escaping. Strings containing a newline (’\n’), carriage return (’\r’), a tab (’\t’), a form feed (’\f’), the escaping character (’\’) or the item separator character *SEP* are escaped using C-like fashioned escaping, so that a newline is converted to the sequence "\n", a carriage return to "\r", ’\’ to "\\" and the separator *SEP* is converted to "*SEP*".

‘`csv`’

Perform CSV-like escaping, as described in RFC4180. Strings containing a newline (’\n’), a carriage return (’\r’), a double quote (’"’), or *SEP* are enclosed in double-quotes.

‘`none`’

Perform no escaping.

‘`print_section, p`’

Print the section name at the begin of each line if the value is `1`, disable it with value set to `0`. Default value is `1`.

4.3 flat

Flat format.

A free-form output where each line contains an explicit key=value, such as "streams.stream.3.tags.foo=bar". The output is shell escaped, so it can be directly embedded in sh scripts as long as the separator character is an alphanumeric character or an underscore (see *sep_char* option).

The description of the accepted options follows.

‘`sep_char, s`’

Separator character used to separate the chapter, the section name, IDs and potential tags in the printed field key.

Default value is ’.’.

‘`hierarchical, h`’

Specify if the section name specification should be hierarchical. If set to 1, and if there is more than one section in the current chapter, the section name will be prefixed by the name of the chapter. A value of 0 will disable this behavior.

Default value is 1.

4.4 ini

INI format output.

Print output in an INI based format.

The following conventions are adopted:

- all key and values are UTF-8
- ’.’ is the subgroup separator
- newline, ’\t’, ’\f’, ’\b’ and the following characters are escaped
- ’\’ is the escape character
- ’#’ is the comment indicator
- ’=’ is the key/value separator

- ’:’ is not used but usually parsed as key/value separator

This writer accepts options as a list of *key*=*value* pairs, separated by ":".

The description of the accepted options follows.

‘`hierarchical, h`’

> Specify if the section name specification should be hierarchical. If set to 1, and if there is more than one section in the current chapter, the section name will be prefixed by the name of the chapter. A value of 0 will disable this behavior.
>
> Default value is 1.

4.5 json

JSON based format.

Each section is printed using JSON notation.

The description of the accepted options follows.

‘`compact, c`’

> If set to 1 enable compact output, that is each section will be printed on a single line. Default value is 0.

For more information about JSON, see http://www.json.org/.

4.6 xml

XML based format.

The XML output is described in the XML schema description file ‘`ffprobe.xsd`’ installed in the FFmpeg datadir.

An updated version of the schema can be retrieved at the url http://www.ffmpeg.org/schema/ffprobe.xsd, which redirects to the latest schema committed into the FFmpeg development source code tree.

Note that the output issued will be compliant to the ‘`ffprobe.xsd`’ schema only when no special global output options (‘`unit`’, ‘`prefix`’, ‘`byte_binary_prefix`’, ‘`sexagesimal`’ etc.) are specified.

The description of the accepted options follows.

‘`fully_qualified, q`’

If set to 1 specify if the output should be fully qualified. Default value is 0. This is required for generating an XML file which can be validated through an XSD file.

'`xsd_compliant, x`'

If set to 1 perform more checks for ensuring that the output is XSD compliant. Default value is 0. This option automatically sets '`fully_qualified`' to 1.

For more information about the XML format, see http://www.w3.org/XML/.

5. Timecode

`ffprobe` supports Timecode extraction:

- MPEG1/2 timecode is extracted from the GOP, and is available in the video stream details ('`-show_streams`', see *timecode*).
- MOV timecode is extracted from tmcd track, so is available in the tmcd stream metadata ('`-show_streams`', see *TAG:timecode*).
- DV, GXF and AVI timecodes are available in format metadata ('`-show_format`', see *TAG:timecode*).

6. See Also

ffprobe-all, ffmpeg, ffplay, ffserver, ffmpeg-utils, ffmpeg-scaler, ffmpeg-resampler, ffmpeg-codecs, ffmpeg-bitstream-filters, ffmpeg-formats, ffmpeg-devices, ffmpeg-protocols, ffmpeg-filters

7. Authors

The FFmpeg developers.

For details about the authorship, see the Git history of the project (git://source.ffmpeg.org/ffmpeg), e.g. by typing the command `git log` in the FFmpeg source directory, or browsing the online repository at http://source.ffmpeg.org.

Maintainers for the specific components are listed in the file '`MAINTAINERS`' in the source code tree.

This document was generated by *john* on *September 24, 2013* using *texi2html 1.82*.