FFmpeg Devices Documentation

Table of Contents

1. Description

This document describes the input and output devices provided by the libavdevice library.

2. Device Options

The libavdevice library provides the same interface as libavformat. Namely, an input device is considered like a demuxer, and an output device like a muxer, and the interface and generic device options are the same provided by libavformat (see the ffmpeg-formats manual).

In addition each input or output device may support so-called private options, which are specific for that component.

Options may be set by specifying *-option value* in the FFmpeg tools, or by setting the value explicitly in the device `AVFormatContext` options or using the ‘`libavutil/opt.h`’ API for programmatic use.

3. Input Devices

Input devices are configured elements in FFmpeg which allow to access the data coming from a multimedia device attached to your system.

When you configure your FFmpeg build, all the supported input devices are enabled by default. You can list all available ones using the configure option "–list-indevs".

You can disable all the input devices using the configure option "–disable-indevs", and selectively enable an input device using the option "–enable-indev=*INDEV*", or you can disable a particular input device using the option "–disable-indev=*INDEV*".

The option "-formats" of the ff* tools will display the list of supported input devices (amongst the demuxers).

A description of the currently available input devices follows.

3.1 alsa

ALSA (Advanced Linux Sound Architecture) input device.

To enable this input device during configuration you need libasound installed on your system.

This device allows capturing from an ALSA device. The name of the device to capture has to be an ALSA card identifier.

An ALSA identifier has the syntax:

```
hw:CARD[,DEV[,SUBDEV]]
```

where the *DEV* and *SUBDEV* components are optional.

The three arguments (in order: *CARD*,*DEV*,*SUBDEV*) specify card number or identifier, device number and subdevice number (-1 means any).

To see the list of cards currently recognized by your system check the files ‘`/proc/asound/cards`’ and ‘`/proc/asound/devices`’.

For example to capture with `ffmpeg` from an ALSA device with card id 0, you may run the command:

```
ffmpeg -f alsa -i hw:0 alsaout.wav
```

For more information see: http://www.alsa-project.org/alsa-doc/alsa-lib/pcm.html

3.2 bktr

BSD video input device.

3.3 dshow

Windows DirectShow input device.

DirectShow support is enabled when FFmpeg is built with the mingw-w64 project. Currently only audio and video devices are supported.

Multiple devices may be opened as separate inputs, but they may also be opened on the same input, which should improve synchronism between them.

The input name should be in the format:

```
TYPE=NAME[:TYPE=NAME]
```

where *TYPE* can be either *audio* or *video*, and *NAME* is the device's name.

3.3.1 Options

If no options are specified, the device's defaults are used. If the device does not support the requested options, it will fail to open.

‘`video_size`’

Set the video size in the captured video.

‘`framerate`’

Set the frame rate in the captured video.

‘`sample_rate`’

Set the sample rate (in Hz) of the captured audio.

‘`sample_size`’

Set the sample size (in bits) of the captured audio.

‘`channels`’

Set the number of channels in the captured audio.

‘`list_devices`’

If set to ‘`true`’, print a list of devices and exit.

‘`list_options`’

If set to ‘`true`’, print a list of selected device's options and exit.

‘`video_device_number`’

Set video device number for devices with same name (starts at 0, defaults to 0).

‘`audio_device_number`’

Set audio device number for devices with same name (starts at 0, defaults to 0).

‘`pixel_format`’

Select pixel format to be used by DirectShow. This may only be set when the video codec is not set or set to rawvideo.

‘`audio_buffer_size`’

Set audio device buffer size in milliseconds (which can directly impact latency, depending on the device). Defaults to using the audio device’s default buffer size (typically some multiple of 500ms). Setting this value too low can degrade performance. See also http://msdn.microsoft.com/en-us/library/windows/desktop/dd377582(v=vs.85).aspx

3.3.2 Examples

- Print the list of DirectShow supported devices and exit:

```
$ ffmpeg -list_devices true -f dshow -i dummy
```

- Open video device *Camera*:

```
$ ffmpeg -f dshow -i video="Camera"
```

- Open second video device with name *Camera*:

```
$ ffmpeg -f dshow -video_device_number 1 -i video="Camera"
```

- Open video device *Camera* and audio device *Microphone*:

```
$ ffmpeg -f dshow -i video="Camera":audio="Microphone"
```

- Print the list of supported options in selected device and exit:

```
$ ffmpeg -list_options true -f dshow -i video="Camera"
```

3.4 dv1394

Linux DV 1394 input device.

3.5 fbdev

Linux framebuffer input device.

The Linux framebuffer is a graphic hardware-independent abstraction layer to show graphics on a computer monitor, typically on the console. It is accessed through a file device node, usually ‘`/dev/fb0`’.

For more detailed information read the file Documentation/fb/framebuffer.txt included in the Linux source tree.

To record from the framebuffer device ‘`/dev/fb0`’ with `ffmpeg`:

```
ffmpeg -f fbdev -r 10 -i /dev/fb0 out.avi
```

You can take a single screenshot image with the command:

```
ffmpeg -f fbdev -frames:v 1 -r 1 -i /dev/fb0 screenshot.jpeg
```

See also http://linux-fbdev.sourceforge.net/, and fbset(1).

3.6 iec61883

FireWire DV/HDV input device using libiec61883.

To enable this input device, you need libiec61883, libraw1394 and libavc1394 installed on your system. Use the configure option `--enable-libiec61883` to compile with the device enabled.

The iec61883 capture device supports capturing from a video device connected via IEEE1394 (FireWire), using libiec61883 and the new Linux FireWire stack (juju). This is the default DV/HDV input method in Linux Kernel 2.6.37 and later, since the old FireWire stack was removed.

Specify the FireWire port to be used as input file, or "auto" to choose the first port connected.

3.6.1 Options

‘`dvtype`’

Override autodetection of DV/HDV. This should only be used if auto detection does not work, or if usage of a different device type should be prohibited. Treating a DV device as HDV (or vice versa) will not work and result in undefined behavior. The values ‘`auto`’, ‘`dv`’ and ‘`hdv`’ are supported.

‘`dvbuffer`’

Set maxiumum size of buffer for incoming data, in frames. For DV, this is an exact value. For HDV, it is not frame exact, since HDV does not have a fixed frame size.

‘`dvguid`’

Select the capture device by specifying it’s GUID. Capturing will only be performed from the specified device and fails if no device with the given GUID is found. This is useful to select the input if multiple devices are connected at the same time. Look at /sys/bus/firewire/devices to find out the GUIDs.

3.6.2 Examples

- Grab and show the input of a FireWire DV/HDV device.

```
ffplay -f iec61883 -i auto
```

- Grab and record the input of a FireWire DV/HDV device, using a packet buffer of 100000 packets if the source is HDV.

```
ffmpeg -f iec61883 -i auto -hdvbuffer 100000 out.mpg
```

3.7 jack

JACK input device.

To enable this input device during configuration you need libjack installed on your system.

A JACK input device creates one or more JACK writable clients, one for each audio channel, with name *client_name*:input_*N*, where *client_name* is the name provided by the application, and *N* is a number which identifies the channel. Each writable client will send the acquired data to the FFmpeg input device.

Once you have created one or more JACK readable clients, you need to connect them to one or more JACK writable clients.

To connect or disconnect JACK clients you can use the `jack_connect` and `jack_disconnect` programs, or do it through a graphical interface, for example with `qjackctl`.

To list the JACK clients and their properties you can invoke the command `jack_lsp`.

Follows an example which shows how to capture a JACK readable client with `ffmpeg`.

```
# Create a JACK writable client with name "ffmpeg".
$ ffmpeg -f jack -i ffmpeg -y out.wav

# Start the sample jack_metro readable client.
$ jack_metro -b 120 -d 0.2 -f 4000

# List the current JACK clients.
$ jack_lsp -c
system:capture_1
system:capture_2
system:playback_1
system:playback_2
ffmpeg:input_1
metro:120_bpm

# Connect metro to the ffmpeg writable client.
$ jack_connect metro:120_bpm ffmpeg:input_1
```

For more information read: http://jackaudio.org/

3.8 lavfi

Libavfilter input virtual device.

This input device reads data from the open output pads of a libavfilter filtergraph.

For each filtergraph open output, the input device will create a corresponding stream which is mapped to the generated output. Currently only video data is supported. The filtergraph is specified through the option ‘`graph`’.

3.8.1 Options

‘`graph`’

Specify the filtergraph to use as input. Each video open output must be labelled by a unique string of the form "out*N*", where *N* is a number starting from 0 corresponding to the mapped input stream generated by the device. The first unlabelled output is automatically assigned to the "out0" label, but all the others need to be specified explicitly.

If not specified defaults to the filename specified for the input device.

‘`graph_file`’

Set the filename of the filtergraph to be read and sent to the other filters. Syntax of the filtergraph is the same as the one specified by the option *graph*.

3.8.2 Examples

- Create a color video stream and play it back with `ffplay`:

```
ffplay -f lavfi -graph "color=c=pink [out0]" dummy
```

- As the previous example, but use filename for specifying the graph description, and omit the "out0" label:

```
ffplay -f lavfi color=c=pink
```

- Create three different video test filtered sources and play them:

```
ffplay -f lavfi -graph "testsrc [out0]; testsrc,hflip [out1]; testsrc,negate [out2]" test3
```

- Read an audio stream from a file using the amovie source and play it back with `ffplay`:

```
ffplay -f lavfi "amovie=test.wav"
```

- Read an audio stream and a video stream and play it back with `ffplay`:

```
ffplay -f lavfi "movie=test.avi[out0];amovie=test.wav[out1]"
```

3.9 libdc1394

IIDC1394 input device, based on libdc1394 and libraw1394.

3.10 openal

The OpenAL input device provides audio capture on all systems with a working OpenAL 1.1 implementation.

To enable this input device during configuration, you need OpenAL headers and libraries installed on your system, and need to configure FFmpeg with `--enable-openal`.

OpenAL headers and libraries should be provided as part of your OpenAL implementation, or as an additional download (an SDK). Depending on your installation you may need to specify additional flags via the `--extra-cflags` and `--extra-ldflags` for allowing the build system to locate the OpenAL headers and libraries.

An incomplete list of OpenAL implementations follows:

Creative

The official Windows implementation, providing hardware acceleration with supported devices and software fallback. See http://openal.org/.

OpenAL Soft

Portable, open source (LGPL) software implementation. Includes backends for the most common sound APIs on the Windows, Linux, Solaris, and BSD operating systems. See http://kcat.strangesoft.net/openal.html.

Apple

OpenAL is part of Core Audio, the official Mac OS X Audio interface. See http://developer.apple.com/technologies/mac/audio-and-video.html

This device allows to capture from an audio input device handled through OpenAL.

You need to specify the name of the device to capture in the provided filename. If the empty string is provided, the device will automatically select the default device. You can get the list of the supported devices by using the option *list_devices*.

3.10.1 Options

‘`channels`’

Set the number of channels in the captured audio. Only the values ‘`1`’ (monaural) and ‘`2`’ (stereo) are currently supported. Defaults to ‘`2`’.

‘`sample_size`’

Set the sample size (in bits) of the captured audio. Only the values ‘`8`’ and ‘`16`’ are currently supported. Defaults to ‘`16`’.

‘`sample_rate`’

Set the sample rate (in Hz) of the captured audio. Defaults to ‘`44.1k`’.

‘`list_devices`’

If set to ‘`true`’, print a list of devices and exit. Defaults to ‘`false`’.

3.10.2 Examples

Print the list of OpenAL supported devices and exit:

```
$ ffmpeg -list_devices true -f openal -i dummy out.ogg
```

Capture from the OpenAL device ‘DR-BT101 via PulseAudio’:

```
$ ffmpeg -f openal -i ’DR-BT101 via PulseAudio’ out.ogg
```

Capture from the default device (note the empty string ” as filename):

```
$ ffmpeg -f openal -i ’’ out.ogg
```

Capture from two devices simultaneously, writing to two different files, within the same `ffmpeg` command:

```
$ ffmpeg -f openal -i ’DR-BT101 via PulseAudio’ out1.ogg -f openal -i ’ALSA Default’ out2.ogg
```

Note: not all OpenAL implementations support multiple simultaneous capture - try the latest OpenAL Soft if the above does not work.

3.11 oss

Open Sound System input device.

The filename to provide to the input device is the device node representing the OSS input device, and is usually set to ‘`/dev/dsp`’.

For example to grab from ‘`/dev/dsp`’ using `ffmpeg` use the command:

```
ffmpeg -f oss -i /dev/dsp /tmp/oss.wav
```

For more information about OSS see: http://manuals.opensound.com/usersguide/dsp.html

3.12 pulse

pulseaudio input device.

To enable this input device during configuration you need libpulse-simple installed in your system.

The filename to provide to the input device is a source device or the string "default"

To list the pulse source devices and their properties you can invoke the command `pactl list sources`.

```
ffmpeg -f pulse -i default /tmp/pulse.wav
```

3.12.1 *server* AVOption

The syntax is:

```
-server server name
```

Connects to a specific server.

3.12.2 *name* AVOption

The syntax is:

```
-name application name
```

Specify the application name pulse will use when showing active clients, by default it is the LIBAVFORMAT_IDENT string

3.12.3 *stream_name* AVOption

The syntax is:

```
-stream_name stream name
```

Specify the stream name pulse will use when showing active streams, by default it is "record"

3.12.4 *sample_rate* AVOption

The syntax is:

```
-sample_rate samplerate
```

Specify the samplerate in Hz, by default 48kHz is used.

3.12.5 *channels* AVOption

The syntax is:

```
-channels N
```

Specify the channels in use, by default 2 (stereo) is set.

3.12.6 *frame_size* AVOption

The syntax is:

```
-frame_size bytes
```

Specify the number of byte per frame, by default it is set to 1024.

3.12.7 *fragment_size* AVOption

The syntax is:

```
-fragment_size bytes
```

Specify the minimal buffering fragment in pulseaudio, it will affect the audio latency. By default it is unset.

3.13 sndio

sndio input device.

To enable this input device during configuration you need libsndio installed on your system.

The filename to provide to the input device is the device node representing the sndio input device, and is usually set to '`/dev/audio0`'.

For example to grab from '`/dev/audio0`' using `ffmpeg` use the command:

```
ffmpeg -f sndio -i /dev/audio0 /tmp/oss.wav
```

3.14 video4linux2, v4l2

Video4Linux2 input video device.

"v4l2" can be used as alias for "video4linux2".

If FFmpeg is built with v4l-utils support (by using the `--enable-libv4l2` configure option), it is possible to use it with the `-use_libv4l2` input device option.

The name of the device to grab is a file device node, usually Linux systems tend to automatically create such nodes when the device (e.g. an USB webcam) is plugged into the system, and has a name of the kind '`/dev/videoN`', where *N* is a number associated to the device.

Video4Linux2 devices usually support a limited set of *width*x*height* sizes and frame rates. You can check which are supported using `-list_formats all` for Video4Linux2 devices. Some devices, like TV cards, support one or more standards. It is possible to list all the supported standards using `-list_standards all`.

The time base for the timestamps is 1 microsecond. Depending on the kernel version and configuration, the timestamps may be derived from the real time clock (origin at the Unix Epoch) or the monotonic clock (origin usually at boot time, unaffected by NTP or manual changes to the clock). The '`-timestamps abs`' or '`-ts abs`' option can be used to force conversion into the real time clock.

Some usage examples of the video4linux2 device with `ffmpeg` and `ffplay`:

- Grab and show the input of a video4linux2 device:

```
ffplay -f video4linux2 -framerate 30 -video_size hd720 /dev/video0
```

- Grab and record the input of a video4linux2 device, leave the frame rate and size as previously set:

```
ffmpeg -f video4linux2 -input_format mjpeg -i /dev/video0 out.mpeg
```

For more information about Video4Linux, check http://linuxtv.org/.

3.14.1 Options

'`standard`'

Set the standard. Must be the name of a supported standard. To get a list of the supported standards, use the '`list_standards`' option.

'`channel`'

Set the input channel number. Default to -1, which means using the previously selected channel.

'`video_size`'

Set the video frame size. The argument must be a string in the form *WIDTH*x*HEIGHT* or a valid size abbreviation.

'`pixel_format`'

Select the pixel format (only valid for raw video input).

'`input_format`'

Set the preferred pixel format (for raw video) or a codec name. This option allows to select the input format, when several are available.

‘framerate’

Set the preferred video frame rate.

‘list_formats’

List available formats (supported pixel formats, codecs, and frame sizes) and exit.

Available values are:

‘all’

Show all available (compressed and non-compressed) formats.

‘raw’

Show only raw video (non-compressed) formats.

‘compressed’

Show only compressed formats.

‘list_standards’

List supported standards and exit.

Available values are:

‘all’

Show all supported standards.

‘timestamps, ts’

Set type of timestamps for grabbed frames.

Available values are:

‘default’

Use timestamps from the kernel.

‘abs’

Use absolute timestamps (wall clock).

‘mono2abs’

Force conversion from monotonic to absolute timestamps.

Default value is `default`.

3.15 vfwcap

VfW (Video for Windows) capture input device.

The filename passed as input is the capture driver number, ranging from 0 to 9. You may use "list" as filename to print a list of drivers. Any other filename will be interpreted as device number 0.

3.16 x11grab

X11 video input device.

This device allows to capture a region of an X11 display.

The filename passed as input has the syntax:

```
[hostname]:display_number.screen_number[+x_offset,y_offset]
```

hostname:*display_number*.*screen_number* specifies the X11 display name of the screen to grab from. *hostname* can be omitted, and defaults to "localhost". The environment variable `DISPLAY` contains the default display name.

x_offset and *y_offset* specify the offsets of the grabbed area with respect to the top-left border of the X11 screen. They default to 0.

Check the X11 documentation (e.g. man X) for more detailed information.

Use the `dpyinfo` program for getting basic information about the properties of your X11 display (e.g. grep for "name" or "dimensions").

For example to grab from ‘`:0.0`’ using `ffmpeg`:

```
ffmpeg -f x11grab -r 25 -s cif -i :0.0 out.mpg
```

Grab at position `10,20`:

```
ffmpeg -f x11grab -r 25 -s cif -i :0.0+10,20 out.mpg
```

3.16.1 Options

‘`draw_mouse`’

Specify whether to draw the mouse pointer. A value of `0` specify not to draw the pointer. Default value is `1`.

‘`follow_mouse`’

Make the grabbed area follow the mouse. The argument can be `centered` or a number of pixels *PIXELS*.

When it is specified with "centered", the grabbing region follows the mouse pointer and keeps the pointer at the center of region; otherwise, the region follows only when the mouse pointer reaches within *PIXELS* (greater than zero) to the edge of region.

For example:

```
ffmpeg -f x11grab -follow_mouse centered -r 25 -s cif -i :0.0 out.mpg
```

To follow only when the mouse pointer reaches within 100 pixels to edge:

```
ffmpeg -f x11grab -follow_mouse 100 -r 25 -s cif -i :0.0 out.mpg
```

‘`framerate`’

Set the grabbing frame rate. Default value is `ntsc`, corresponding to a frame rate of `30000/1001`.

‘`show_region`’

Show grabbed region on screen.

If *show_region* is specified with `1`, then the grabbing region will be indicated on screen. With this option, it is easy to know what is being grabbed if only a portion of the screen is grabbed.

For example:

```
ffmpeg -f x11grab -show_region 1 -r 25 -s cif -i :0.0+10,20 out.mpg
```

With *follow_mouse*:

```
ffmpeg -f x11grab -follow_mouse centered -show_region 1 -r 25 -s cif -i :0.0 out.mpg
```

‘`video_size`’

Set the video frame size. Default value is `vga`.

4. Output Devices

Output devices are configured elements in FFmpeg which allow to write multimedia data to an output device attached to your system.

When you configure your FFmpeg build, all the supported output devices are enabled by default. You can list all available ones using the configure option "–list-outdevs".

You can disable all the output devices using the configure option "–disable-outdevs", and selectively enable an output device using the option "–enable-outdev=*OUTDEV*", or you can disable a particular input device using the option "–disable-outdev=*OUTDEV*".

The option "-formats" of the ff* tools will display the list of enabled output devices (amongst the muxers).

A description of the currently available output devices follows.

4.1 alsa

ALSA (Advanced Linux Sound Architecture) output device.

4.2 caca

CACA output device.

This output device allows to show a video stream in CACA window. Only one CACA window is allowed per application, so you can have only one instance of this output device in an application.

To enable this output device you need to configure FFmpeg with `--enable-libcaca`. libcaca is a graphics library that outputs text instead of pixels.

For more information about libcaca, check: http://caca.zoy.org/wiki/libcaca

4.2.1 Options

‘`window_title`’

Set the CACA window title, if not specified default to the filename specified for the output device.

‘`window_size`’

Set the CACA window size, can be a string of the form *width*x*height* or a video size abbreviation. If not specified it defaults to the size of the input video.

‘driver’

Set display driver.

‘algorithm’

Set dithering algorithm. Dithering is necessary because the picture being rendered has usually far more colours than the available palette. The accepted values are listed with `-list_dither algorithms`.

‘antialias’

Set antialias method. Antialiasing smoothens the rendered image and avoids the commonly seen staircase effect. The accepted values are listed with `-list_dither antialiases`.

‘charset’

Set which characters are going to be used when rendering text. The accepted values are listed with `-list_dither charsets`.

‘color’

Set color to be used when rendering text. The accepted values are listed with `-list_dither colors`.

‘list_drivers’

If set to ‘`true`’, print a list of available drivers and exit.

‘list_dither’

List available dither options related to the argument. The argument must be one of `algorithms`, `antialiases`, `charsets`, `colors`.

4.2.2 Examples

- The following command shows the `ffmpeg` output is an CACA window, forcing its size to 80x25:

```
ffmpeg -i INPUT -vcodec rawvideo -pix_fmt rgb24 -window_size 80x25 -f caca -
```

- Show the list of available drivers and exit:

```
ffmpeg -i INPUT -pix_fmt rgb24 -f caca -list_drivers true -
```

- Show the list of available dither colors and exit:

```
ffmpeg -i INPUT -pix_fmt rgb24 -f caca -list_dither colors -
```

4.3 oss

OSS (Open Sound System) output device.

4.4 sdl

SDL (Simple DirectMedia Layer) output device.

This output device allows to show a video stream in an SDL window. Only one SDL window is allowed per application, so you can have only one instance of this output device in an application.

To enable this output device you need libsdl installed on your system when configuring your build.

For more information about SDL, check: http://www.libsdl.org/

4.4.1 Options

‘`window_title`’

Set the SDL window title, if not specified default to the filename specified for the output device.

‘`icon_title`’

Set the name of the iconified SDL window, if not specified it is set to the same value of *window_title*.

‘`window_size`’

Set the SDL window size, can be a string of the form *width*x*height* or a video size abbreviation. If not specified it defaults to the size of the input video, downscaled according to the aspect ratio.

4.4.2 Examples

The following command shows the `ffmpeg` output is an SDL window, forcing its size to the qcif format:

```
ffmpeg -i INPUT -vcodec rawvideo -pix_fmt yuv420p -window_size qcif -f sdl "SDL output"
```

4.5 sndio

sndio audio output device.

4.6 xv

XV (XVideo) output device.

This output device allows to show a video stream in a X Window System window.

4.6.1 Options

‘`display_name`’

Specify the hardware display name, which determines the display and communications domain to be used.

The display name or DISPLAY environment variable can be a string in the format *hostname*[:*number*[.*screen_number*]].

hostname specifies the name of the host machine on which the display is physically attached. *number* specifies the number of the display server on that host machine. *screen_number* specifies the screen to be used on that server.

If unspecified, it defaults to the value of the DISPLAY environment variable.

For example, `dual-headed:0.1` would specify screen 1 of display 0 on the machine named “dual-headed”.

Check the X11 specification for more detailed information about the display name format.

‘`window_size`’

Set the created window size, can be a string of the form *width*x*height* or a video size abbreviation. If not specified it defaults to the size of the input video.

‘`window_x`’
‘`window_y`’

Set the X and Y window offsets for the created window. They are both set to 0 by default. The values may be ignored by the window manager.

‘`window_title`’

Set the window title, if not specified default to the filename specified for the output device.

For more information about XVideo see http://www.x.org/.

4.6.2 Examples

- Decode, display and encode video input with `ffmpeg` at the same time:

```
ffmpeg -i INPUT OUTPUT -f xv display
```

- Decode and display the input video to multiple X11 windows:

```
ffmpeg -i INPUT -f xv normal -vf negate -f xv negated
```

5. See Also

ffmpeg, ffplay, ffprobe, ffserver, libavdevice

6. 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*.

FFmpeg Devices Documentation

Table of Contents

1. Description

This document describes the input and output devices provided by the libavdevice library.

2. Device Options

The libavdevice library provides the same interface as libavformat. Namely, an input device is considered like a demuxer, and an output device like a muxer, and the interface and generic device options are the same provided by libavformat (see the ffmpeg-formats manual).

In addition each input or output device may support so-called private options, which are specific for that component.

Options may be set by specifying *-option value* in the FFmpeg tools, or by setting the value explicitly in the device `AVFormatContext` options or using the ‘`libavutil/opt.h`’ API for programmatic use.

3. Input Devices

Input devices are configured elements in FFmpeg which allow to access the data coming from a multimedia device attached to your system.

When you configure your FFmpeg build, all the supported input devices are enabled by default. You can list all available ones using the configure option "–list-indevs".

You can disable all the input devices using the configure option "–disable-indevs", and selectively enable an input device using the option "–enable-indev=*INDEV*", or you can disable a particular input device using the option "–disable-indev=*INDEV*".

The option "-formats" of the ff* tools will display the list of supported input devices (amongst the demuxers).

A description of the currently available input devices follows.

3.1 alsa

ALSA (Advanced Linux Sound Architecture) input device.

To enable this input device during configuration you need libasound installed on your system.

This device allows capturing from an ALSA device. The name of the device to capture has to be an ALSA card identifier.

An ALSA identifier has the syntax:

```
hw:CARD[,DEV[,SUBDEV]]
```

where the *DEV* and *SUBDEV* components are optional.

The three arguments (in order: *CARD*,*DEV*,*SUBDEV*) specify card number or identifier, device number and subdevice number (-1 means any).

To see the list of cards currently recognized by your system check the files ‘`/proc/asound/cards`’ and ‘`/proc/asound/devices`’.

For example to capture with `ffmpeg` from an ALSA device with card id 0, you may run the command:

```
ffmpeg -f alsa -i hw:0 alsaout.wav
```

For more information see: http://www.alsa-project.org/alsa-doc/alsa-lib/pcm.html

3.2 bktr

BSD video input device.

3.3 dshow

Windows DirectShow input device.

DirectShow support is enabled when FFmpeg is built with the mingw-w64 project. Currently only audio and video devices are supported.

Multiple devices may be opened as separate inputs, but they may also be opened on the same input, which should improve synchronism between them.

The input name should be in the format:

```
TYPE=NAME[:TYPE=NAME]
```

where *TYPE* can be either *audio* or *video*, and *NAME* is the device's name.

3.3.1 Options

If no options are specified, the device's defaults are used. If the device does not support the requested options, it will fail to open.

‘`video_size`’

Set the video size in the captured video.

‘`framerate`’

Set the frame rate in the captured video.

‘`sample_rate`’

Set the sample rate (in Hz) of the captured audio.

‘`sample_size`’

Set the sample size (in bits) of the captured audio.

‘`channels`’

Set the number of channels in the captured audio.

‘`list_devices`’

If set to ‘`true`’, print a list of devices and exit.

‘`list_options`’

If set to ‘`true`’, print a list of selected device's options and exit.

‘`video_device_number`’

Set video device number for devices with same name (starts at 0, defaults to 0).

‘`audio_device_number`’

Set audio device number for devices with same name (starts at 0, defaults to 0).

‘`pixel_format`’

Select pixel format to be used by DirectShow. This may only be set when the video codec is not set or set to rawvideo.

‘`audio_buffer_size`’

Set audio device buffer size in milliseconds (which can directly impact latency, depending on the device). Defaults to using the audio device’s default buffer size (typically some multiple of 500ms). Setting this value too low can degrade performance. See also http://msdn.microsoft.com/en-us/library/windows/desktop/dd377582(v=vs.85).aspx

3.3.2 Examples

- Print the list of DirectShow supported devices and exit:

```
$ ffmpeg -list_devices true -f dshow -i dummy
```

- Open video device *Camera*:

```
$ ffmpeg -f dshow -i video="Camera"
```

- Open second video device with name *Camera*:

```
$ ffmpeg -f dshow -video_device_number 1 -i video="Camera"
```

- Open video device *Camera* and audio device *Microphone*:

```
$ ffmpeg -f dshow -i video="Camera":audio="Microphone"
```

- Print the list of supported options in selected device and exit:

```
$ ffmpeg -list_options true -f dshow -i video="Camera"
```

3.4 dv1394

Linux DV 1394 input device.

3.5 fbdev

Linux framebuffer input device.

The Linux framebuffer is a graphic hardware-independent abstraction layer to show graphics on a computer monitor, typically on the console. It is accessed through a file device node, usually ‘`/dev/fb0`’.

For more detailed information read the file Documentation/fb/framebuffer.txt included in the Linux source tree.

To record from the framebuffer device ‘`/dev/fb0`’ with `ffmpeg`:

```
ffmpeg -f fbdev -r 10 -i /dev/fb0 out.avi
```

You can take a single screenshot image with the command:

```
ffmpeg -f fbdev -frames:v 1 -r 1 -i /dev/fb0 screenshot.jpeg
```

See also http://linux-fbdev.sourceforge.net/, and fbset(1).

3.6 iec61883

FireWire DV/HDV input device using libiec61883.

To enable this input device, you need libiec61883, libraw1394 and libavc1394 installed on your system. Use the configure option `--enable-libiec61883` to compile with the device enabled.

The iec61883 capture device supports capturing from a video device connected via IEEE1394 (FireWire), using libiec61883 and the new Linux FireWire stack (juju). This is the default DV/HDV input method in Linux Kernel 2.6.37 and later, since the old FireWire stack was removed.

Specify the FireWire port to be used as input file, or "auto" to choose the first port connected.

3.6.1 Options

‘`dvtype`’

> Override autodetection of DV/HDV. This should only be used if auto detection does not work, or if usage of a different device type should be prohibited. Treating a DV device as HDV (or vice versa) will not work and result in undefined behavior. The values ‘`auto`’, ‘`dv`’ and ‘`hdv`’ are supported.

‘`dvbuffer`’

Set maxiumum size of buffer for incoming data, in frames. For DV, this is an exact value. For HDV, it is not frame exact, since HDV does not have a fixed frame size.

‘`dvguid`’

Select the capture device by specifying it’s GUID. Capturing will only be performed from the specified device and fails if no device with the given GUID is found. This is useful to select the input if multiple devices are connected at the same time. Look at /sys/bus/firewire/devices to find out the GUIDs.

3.6.2 Examples

- Grab and show the input of a FireWire DV/HDV device.

```
ffplay -f iec61883 -i auto
```

- Grab and record the input of a FireWire DV/HDV device, using a packet buffer of 100000 packets if the source is HDV.

```
ffmpeg -f iec61883 -i auto -hdvbuffer 100000 out.mpg
```

3.7 jack

JACK input device.

To enable this input device during configuration you need libjack installed on your system.

A JACK input device creates one or more JACK writable clients, one for each audio channel, with name *client_name*:input_*N*, where *client_name* is the name provided by the application, and *N* is a number which identifies the channel. Each writable client will send the acquired data to the FFmpeg input device.

Once you have created one or more JACK readable clients, you need to connect them to one or more JACK writable clients.

To connect or disconnect JACK clients you can use the `jack_connect` and `jack_disconnect` programs, or do it through a graphical interface, for example with `qjackctl`.

To list the JACK clients and their properties you can invoke the command `jack_lsp`.

Follows an example which shows how to capture a JACK readable client with `ffmpeg`.

```
# Create a JACK writable client with name "ffmpeg".
$ ffmpeg -f jack -i ffmpeg -y out.wav

# Start the sample jack_metro readable client.
$ jack_metro -b 120 -d 0.2 -f 4000

# List the current JACK clients.
$ jack_lsp -c
system:capture_1
system:capture_2
system:playback_1
system:playback_2
ffmpeg:input_1
metro:120_bpm

# Connect metro to the ffmpeg writable client.
$ jack_connect metro:120_bpm ffmpeg:input_1
```

For more information read: http://jackaudio.org/

3.8 lavfi

Libavfilter input virtual device.

This input device reads data from the open output pads of a libavfilter filtergraph.

For each filtergraph open output, the input device will create a corresponding stream which is mapped to the generated output. Currently only video data is supported. The filtergraph is specified through the option ‘`graph`’.

3.8.1 Options

‘`graph`’

Specify the filtergraph to use as input. Each video open output must be labelled by a unique string of the form "out*N*", where *N* is a number starting from 0 corresponding to the mapped input stream generated by the device. The first unlabelled output is automatically assigned to the "out0" label, but all the others need to be specified explicitly.

If not specified defaults to the filename specified for the input device.

‘`graph_file`’

Set the filename of the filtergraph to be read and sent to the other filters. Syntax of the filtergraph is the same as the one specified by the option *graph*.

3.8.2 Examples

- Create a color video stream and play it back with `ffplay`:

```
ffplay -f lavfi -graph "color=c=pink [out0]" dummy
```

- As the previous example, but use filename for specifying the graph description, and omit the "out0" label:

```
ffplay -f lavfi color=c=pink
```

- Create three different video test filtered sources and play them:

```
ffplay -f lavfi -graph "testsrc [out0]; testsrc,hflip [out1]; testsrc,negate [out2]" test3
```

- Read an audio stream from a file using the amovie source and play it back with `ffplay`:

```
ffplay -f lavfi "amovie=test.wav"
```

- Read an audio stream and a video stream and play it back with `ffplay`:

```
ffplay -f lavfi "movie=test.avi[out0];amovie=test.wav[out1]"
```

3.9 libdc1394

IIDC1394 input device, based on libdc1394 and libraw1394.

3.10 openal

The OpenAL input device provides audio capture on all systems with a working OpenAL 1.1 implementation.

To enable this input device during configuration, you need OpenAL headers and libraries installed on your system, and need to configure FFmpeg with `--enable-openal`.

OpenAL headers and libraries should be provided as part of your OpenAL implementation, or as an additional download (an SDK). Depending on your installation you may need to specify additional flags via the `--extra-cflags` and `--extra-ldflags` for allowing the build system to locate the OpenAL headers and libraries.

An incomplete list of OpenAL implementations follows:

Creative

The official Windows implementation, providing hardware acceleration with supported devices and software fallback. See http://openal.org/.

OpenAL Soft

Portable, open source (LGPL) software implementation. Includes backends for the most common sound APIs on the Windows, Linux, Solaris, and BSD operating systems. See http://kcat.strangesoft.net/openal.html.

Apple

OpenAL is part of Core Audio, the official Mac OS X Audio interface. See http://developer.apple.com/technologies/mac/audio-and-video.html

This device allows to capture from an audio input device handled through OpenAL.

You need to specify the name of the device to capture in the provided filename. If the empty string is provided, the device will automatically select the default device. You can get the list of the supported devices by using the option *list_devices*.

3.10.1 Options

‘`channels`’

Set the number of channels in the captured audio. Only the values ‘`1`’ (monaural) and ‘`2`’ (stereo) are currently supported. Defaults to ‘`2`’.

‘`sample_size`’

Set the sample size (in bits) of the captured audio. Only the values ‘`8`’ and ‘`16`’ are currently supported. Defaults to ‘`16`’.

‘`sample_rate`’

Set the sample rate (in Hz) of the captured audio. Defaults to ‘`44.1k`’.

‘`list_devices`’

If set to ‘`true`’, print a list of devices and exit. Defaults to ‘`false`’.

3.10.2 Examples

Print the list of OpenAL supported devices and exit:

```
$ ffmpeg -list_devices true -f openal -i dummy out.ogg
```

Capture from the OpenAL device ‘DR-BT101 via PulseAudio’:

```
$ ffmpeg -f openal -i ’DR-BT101 via PulseAudio’ out.ogg
```

Capture from the default device (note the empty string ” as filename):

```
$ ffmpeg -f openal -i ’’ out.ogg
```

Capture from two devices simultaneously, writing to two different files, within the same `ffmpeg` command:

```
$ ffmpeg -f openal -i ’DR-BT101 via PulseAudio’ out1.ogg -f openal -i ’ALSA Default’ out2.ogg
```

Note: not all OpenAL implementations support multiple simultaneous capture - try the latest OpenAL Soft if the above does not work.

3.11 oss

Open Sound System input device.

The filename to provide to the input device is the device node representing the OSS input device, and is usually set to ‘`/dev/dsp`’.

For example to grab from ‘`/dev/dsp`’ using `ffmpeg` use the command:

```
ffmpeg -f oss -i /dev/dsp /tmp/oss.wav
```

For more information about OSS see: http://manuals.opensound.com/usersguide/dsp.html

3.12 pulse

pulseaudio input device.

To enable this input device during configuration you need libpulse-simple installed in your system.

The filename to provide to the input device is a source device or the string "default"

To list the pulse source devices and their properties you can invoke the command `pactl list sources`.

```
ffmpeg -f pulse -i default /tmp/pulse.wav
```

3.12.1 *server* AVOption

The syntax is:

```
-server server name
```

Connects to a specific server.

3.12.2 *name* AVOption

The syntax is:

```
-name application name
```

Specify the application name pulse will use when showing active clients, by default it is the LIBAVFORMAT_IDENT string

3.12.3 *stream_name* AVOption

The syntax is:

```
-stream_name stream name
```

Specify the stream name pulse will use when showing active streams, by default it is "record"

3.12.4 *sample_rate* AVOption

The syntax is:

```
-sample_rate samplerate
```

Specify the samplerate in Hz, by default 48kHz is used.

3.12.5 *channels* AVOption

The syntax is:

```
-channels N
```

Specify the channels in use, by default 2 (stereo) is set.

3.12.6 *frame_size* AVOption

The syntax is:

```
-frame_size bytes
```

Specify the number of byte per frame, by default it is set to 1024.

3.12.7 *fragment_size* AVOption

The syntax is:

```
-fragment_size bytes
```

Specify the minimal buffering fragment in pulseaudio, it will affect the audio latency. By default it is unset.

3.13 sndio

sndio input device.

To enable this input device during configuration you need libsndio installed on your system.

The filename to provide to the input device is the device node representing the sndio input device, and is usually set to '`/dev/audio0`'.

For example to grab from '`/dev/audio0`' using `ffmpeg` use the command:

```
ffmpeg -f sndio -i /dev/audio0 /tmp/oss.wav
```

3.14 video4linux2, v4l2

Video4Linux2 input video device.

"v4l2" can be used as alias for "video4linux2".

If FFmpeg is built with v4l-utils support (by using the `--enable-libv4l2` configure option), it is possible to use it with the `-use_libv4l2` input device option.

The name of the device to grab is a file device node, usually Linux systems tend to automatically create such nodes when the device (e.g. an USB webcam) is plugged into the system, and has a name of the kind '`/dev/videoN`', where *N* is a number associated to the device.

Video4Linux2 devices usually support a limited set of *width*x*height* sizes and frame rates. You can check which are supported using `-list_formats all` for Video4Linux2 devices. Some devices, like TV cards, support one or more standards. It is possible to list all the supported standards using `-list_standards all`.

The time base for the timestamps is 1 microsecond. Depending on the kernel version and configuration, the timestamps may be derived from the real time clock (origin at the Unix Epoch) or the monotonic clock (origin usually at boot time, unaffected by NTP or manual changes to the clock). The '`-timestamps abs`' or '`-ts abs`' option can be used to force conversion into the real time clock.

Some usage examples of the video4linux2 device with `ffmpeg` and `ffplay`:

- Grab and show the input of a video4linux2 device:

```
ffplay -f video4linux2 -framerate 30 -video_size hd720 /dev/video0
```

- Grab and record the input of a video4linux2 device, leave the frame rate and size as previously set:

```
ffmpeg -f video4linux2 -input_format mjpeg -i /dev/video0 out.mpeg
```

For more information about Video4Linux, check http://linuxtv.org/.

3.14.1 Options

'`standard`'

Set the standard. Must be the name of a supported standard. To get a list of the supported standards, use the '`list_standards`' option.

'`channel`'

Set the input channel number. Default to -1, which means using the previously selected channel.

'`video_size`'

Set the video frame size. The argument must be a string in the form *WIDTH*x*HEIGHT* or a valid size abbreviation.

'`pixel_format`'

Select the pixel format (only valid for raw video input).

'`input_format`'

Set the preferred pixel format (for raw video) or a codec name. This option allows to select the input format, when several are available.

‘framerate’

Set the preferred video frame rate.

‘list_formats’

List available formats (supported pixel formats, codecs, and frame sizes) and exit.

Available values are:

‘all’

Show all available (compressed and non-compressed) formats.

‘raw’

Show only raw video (non-compressed) formats.

‘compressed’

Show only compressed formats.

‘list_standards’

List supported standards and exit.

Available values are:

‘all’

Show all supported standards.

‘timestamps, ts’

Set type of timestamps for grabbed frames.

Available values are:

‘default’

Use timestamps from the kernel.

‘abs’

Use absolute timestamps (wall clock).

‘mono2abs’

Force conversion from monotonic to absolute timestamps.

Default value is `default`.

3.15 vfwcap

VfW (Video for Windows) capture input device.

The filename passed as input is the capture driver number, ranging from 0 to 9. You may use "list" as filename to print a list of drivers. Any other filename will be interpreted as device number 0.

3.16 x11grab

X11 video input device.

This device allows to capture a region of an X11 display.

The filename passed as input has the syntax:

```
[hostname]:display_number.screen_number[+x_offset,y_offset]
```

hostname:*display_number*.*screen_number* specifies the X11 display name of the screen to grab from. *hostname* can be omitted, and defaults to "localhost". The environment variable `DISPLAY` contains the default display name.

x_offset and *y_offset* specify the offsets of the grabbed area with respect to the top-left border of the X11 screen. They default to 0.

Check the X11 documentation (e.g. man X) for more detailed information.

Use the `dpyinfo` program for getting basic information about the properties of your X11 display (e.g. grep for "name" or "dimensions").

For example to grab from ‘`:0.0`’ using `ffmpeg`:

```
ffmpeg -f x11grab -r 25 -s cif -i :0.0 out.mpg
```

Grab at position `10,20`:

```
ffmpeg -f x11grab -r 25 -s cif -i :0.0+10,20 out.mpg
```

3.16.1 Options

‘`draw_mouse`’

Specify whether to draw the mouse pointer. A value of `0` specify not to draw the pointer. Default value is `1`.

‘`follow_mouse`’

Make the grabbed area follow the mouse. The argument can be `centered` or a number of pixels *PIXELS*.

When it is specified with "centered", the grabbing region follows the mouse pointer and keeps the pointer at the center of region; otherwise, the region follows only when the mouse pointer reaches within *PIXELS* (greater than zero) to the edge of region.

For example:

```
ffmpeg -f x11grab -follow_mouse centered -r 25 -s cif -i :0.0 out.mpg
```

To follow only when the mouse pointer reaches within 100 pixels to edge:

```
ffmpeg -f x11grab -follow_mouse 100 -r 25 -s cif -i :0.0 out.mpg
```

‘`framerate`’

Set the grabbing frame rate. Default value is `ntsc`, corresponding to a frame rate of `30000/1001`.

‘`show_region`’

Show grabbed region on screen.

If *show_region* is specified with `1`, then the grabbing region will be indicated on screen. With this option, it is easy to know what is being grabbed if only a portion of the screen is grabbed.

For example:

```
ffmpeg -f x11grab -show_region 1 -r 25 -s cif -i :0.0+10,20 out.mpg
```

With *follow_mouse*:

```
ffmpeg -f x11grab -follow_mouse centered -show_region 1 -r 25 -s cif -i :0.0 out.mpg
```

‘`video_size`’

Set the video frame size. Default value is `vga`.

4. Output Devices

Output devices are configured elements in FFmpeg which allow to write multimedia data to an output device attached to your system.

When you configure your FFmpeg build, all the supported output devices are enabled by default. You can list all available ones using the configure option "–list-outdevs".

You can disable all the output devices using the configure option "–disable-outdevs", and selectively enable an output device using the option "–enable-outdev=*OUTDEV*", or you can disable a particular input device using the option "–disable-outdev=*OUTDEV*".

The option "-formats" of the ff* tools will display the list of enabled output devices (amongst the muxers).

A description of the currently available output devices follows.

4.1 alsa

ALSA (Advanced Linux Sound Architecture) output device.

4.2 caca

CACA output device.

This output device allows to show a video stream in CACA window. Only one CACA window is allowed per application, so you can have only one instance of this output device in an application.

To enable this output device you need to configure FFmpeg with `--enable-libcaca`. libcaca is a graphics library that outputs text instead of pixels.

For more information about libcaca, check: http://caca.zoy.org/wiki/libcaca

4.2.1 Options

‘`window_title`’

Set the CACA window title, if not specified default to the filename specified for the output device.

‘`window_size`’

Set the CACA window size, can be a string of the form *width*x*height* or a video size abbreviation. If not specified it defaults to the size of the input video.

‘driver’

Set display driver.

‘algorithm’

Set dithering algorithm. Dithering is necessary because the picture being rendered has usually far more colours than the available palette. The accepted values are listed with `-list_dither algorithms`.

‘antialias’

Set antialias method. Antialiasing smoothens the rendered image and avoids the commonly seen staircase effect. The accepted values are listed with `-list_dither antialiases`.

‘charset’

Set which characters are going to be used when rendering text. The accepted values are listed with `-list_dither charsets`.

‘color’

Set color to be used when rendering text. The accepted values are listed with `-list_dither colors`.

‘list_drivers’

If set to ‘`true`’, print a list of available drivers and exit.

‘list_dither’

List available dither options related to the argument. The argument must be one of `algorithms`, `antialiases`, `charsets`, `colors`.

4.2.2 Examples

- The following command shows the `ffmpeg` output is an CACA window, forcing its size to 80x25:

```
ffmpeg -i INPUT -vcodec rawvideo -pix_fmt rgb24 -window_size 80x25 -f caca -
```

- Show the list of available drivers and exit:

```
ffmpeg -i INPUT -pix_fmt rgb24 -f caca -list_drivers true -
```

- Show the list of available dither colors and exit:

```
ffmpeg -i INPUT -pix_fmt rgb24 -f caca -list_dither colors -
```

4.3 oss

OSS (Open Sound System) output device.

4.4 sdl

SDL (Simple DirectMedia Layer) output device.

This output device allows to show a video stream in an SDL window. Only one SDL window is allowed per application, so you can have only one instance of this output device in an application.

To enable this output device you need libsdl installed on your system when configuring your build.

For more information about SDL, check: http://www.libsdl.org/

4.4.1 Options

‘`window_title`’

Set the SDL window title, if not specified default to the filename specified for the output device.

‘`icon_title`’

Set the name of the iconified SDL window, if not specified it is set to the same value of *window_title*.

‘`window_size`’

Set the SDL window size, can be a string of the form *width*x*height* or a video size abbreviation. If not specified it defaults to the size of the input video, downscaled according to the aspect ratio.

4.4.2 Examples

The following command shows the `ffmpeg` output is an SDL window, forcing its size to the qcif format:

```
ffmpeg -i INPUT -vcodec rawvideo -pix_fmt yuv420p -window_size qcif -f sdl "SDL output"
```

4.5 sndio

sndio audio output device.

4.6 xv

XV (XVideo) output device.

This output device allows to show a video stream in a X Window System window.

4.6.1 Options

‘`display_name`’

Specify the hardware display name, which determines the display and communications domain to be used.

The display name or DISPLAY environment variable can be a string in the format *hostname*[:*number*[.*screen_number*]].

hostname specifies the name of the host machine on which the display is physically attached. *number* specifies the number of the display server on that host machine. *screen_number* specifies the screen to be used on that server.

If unspecified, it defaults to the value of the DISPLAY environment variable.

For example, `dual-headed:0.1` would specify screen 1 of display 0 on the machine named “dual-headed”.

Check the X11 specification for more detailed information about the display name format.

‘`window_size`’

Set the created window size, can be a string of the form *width*x*height* or a video size abbreviation. If not specified it defaults to the size of the input video.

‘`window_x`’
‘`window_y`’

Set the X and Y window offsets for the created window. They are both set to 0 by default. The values may be ignored by the window manager.

‘`window_title`’

Set the window title, if not specified default to the filename specified for the output device.

For more information about XVideo see http://www.x.org/.

4.6.2 Examples

- Decode, display and encode video input with `ffmpeg` at the same time:

```
ffmpeg -i INPUT OUTPUT -f xv display
```

- Decode and display the input video to multiple X11 windows:

```
ffmpeg -i INPUT -f xv normal -vf negate -f xv negated
```

5. See Also

ffmpeg, ffplay, ffprobe, ffserver, libavdevice

6. 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*.