NAME

ffmpeg-devices - FFmpeg devices

DESCRIPTION

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

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.

INPUT DEVICES

Input devices are configured elements in FFmpeg which enable accessing 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 "-devices" of the ff* tools will display the list of supported input devices.

A description of the currently available input devices follows.

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**>

Options

sample_rate
Set the sample rate in Hz. Default is 48000.

channels
Set the number of channels. Default is 2.

android_camera

Android camera input device.

This input devices uses the Android Camera2 NDK API which is available on devices with API level 24+. The availability of android_camera is autodetected during configuration.

This device allows capturing from all cameras on an Android device, which are integrated into the Camera2 NDK API.

The available cameras are enumerated internally and can be selected with the *camera_index* parameter. The input file string is discarded.

Generally the back facing camera has index 0 while the front facing camera has index 1.

Options

video_size
Set the video size given as a string such as 640x480 or hd720. Falls back to the first available configuration reported by Android if requested video size is not available or by default.

framerate
Set the video framerate. Falls back to the first available configuration reported by Android if requested framerate is not available or by default (-1).

camera_index
Set the index of the camera to use. Default is 0.

input_queue_size
Set the maximum number of frames to buffer. Default is 5.

avfoundation

AVFoundation input device.

AVFoundation is the currently recommended framework by Apple for streamgrabbing on OSX >= 10.7 as well as on iOS.

The input filename has to be given in the following syntax:

```
        -i "[[VIDEO]:[AUDIO]]"
```

The first entry selects the video input while the latter selects the audio input. The stream has to be specified by the device name or the device index as shown by the device list. Alternatively, the video and/or audio input device can be chosen by index using the

```
    B<-video_device_index E<lt>INDEXE<gt>>
```

and/or

```
    B<-audio_device_index E<lt>INDEXE<gt>>
```

, overriding any device name or index given in the input filename.

All available devices can be enumerated by using **-list_devices true**, listing all device names and corresponding indices.

There are two device name aliases:

"default"
  Select the AVFoundation default device of the corresponding type.

"none"
  Do not record the corresponding media type. This is equivalent to specifying an empty device name or index.

Options

AVFoundation supports the following options:

-list_devices <TRUE|FALSE>
  If set to true, a list of all available input devices is given showing all device names and indices.

-video_device_index <INDEX>
  Specify the video device by its index. Overrides anything given in the input filename.

-audio_device_index <INDEX>
  Specify the audio device by its index. Overrides anything given in the input filename.

-pixel_format <FORMAT>
  Request the video device to use a specific pixel format. If the specified format is not supported, a list of available formats is given and the first one in this list is used instead. Available pixel formats are: "monob, rgb555be, rgb555le, rgb565be, rgb565le, rgb24, bgr24, 0rgb, bgr0, 0bgr, rgb0,
  bgr48be, uyvy422, yuva444p, yuva444p16le, yuv444p, yuv422p16, yuv422p10, yuv444p10,
  yuv420p, nv12, yuyv422, gray"

-framerate
  Set the grabbing frame rate. Default is "ntsc", corresponding to a frame rate of "30000/1001".

-video_size
  Set the video frame size.

-capture_cursor
  Capture the mouse pointer. Default is 0.

-capture_mouse_clicks

Capture the screen mouse clicks. Default is 0.

-capture_raw_data

Capture the raw device data. Default is 0. Using this option may result in receiving the underlying data delivered to the AVFoundation framework. E.g. for muxed devices that sends raw DV data to the framework (like tape-based camcorders), setting this option to false results in extracted video frames captured in the designated pixel format only. Setting this option to true results in receiving the raw DV stream untouched.

Examples

- Print the list of AVFoundation supported devices and exit:

```
$ ffmpeg -f avfoundation -list_devices true -i ""
```

- Record video from video device 0 and audio from audio device 0 into out.avi:

```
$ ffmpeg -f avfoundation -i "0:0" out.avi
```

- Record video from video device 2 and audio from audio device 1 into out.avi:

```
$ ffmpeg -f avfoundation -video_device_index 2 -i ":1" out.avi
```

- Record video from the system default video device using the pixel format bgr0 and do not record any audio into out.avi:

```
$ ffmpeg -f avfoundation -pixel_format bgr0 -i "default:none" out.avi
```

- Record raw DV data from a suitable input device and write the output into out.dv:

```
$ ffmpeg -f avfoundation -capture_raw_data true -i "zr100:none" out.dv
```

bktr

BSD video input device.

Options

framerate

Set the frame rate.

video_size

Set the video frame size. Default is "vga".

standard

Available values are:

pal
ntsc
secam
paln
palm
ntscj

decklink

The decklink input device provides capture capabilities for Blackmagic DeckLink devices.

To enable this input device, you need the Blackmagic DeckLink SDK and you need to configure with the appropriate "--extra-cflags" and "--extra-ldflags". On Windows, you need to run the IDL files through **widl**.

DeckLink is very picky about the formats it supports. Pixel format of the input can be set with **raw_format**. Framerate and video size must be determined for your device with **-list_formats 1**. Audio sample rate is always 48 kHz and the number of channels can be 2, 8 or 16. Note that all audio channels are bundled in one single audio track.

Options

list_devices

If set to **true**, print a list of devices and exit. Defaults to **false**. This option is deprecated, please use the "-sources" option of ffmpeg to list the available input devices.

list_formats

If set to **true**, print a list of supported formats and exit. Defaults to **false**.

format_code <FourCC>

This sets the input video format to the format given by the FourCC. To see the supported values of your device(s) use **list_formats**. Note that there is a FourCC **’pal ’** that can also be used as **pal** (3 letters). Default behavior is autodetection of the input video format, if the hardware supports it.

raw_format

Set the pixel format of the captured video. Available values are:

auto This is the default which means 8-bit YUV 422 or 8-bit ARGB if format autodetection is used, 8-bit YUV 422 otherwise.

uyvy422
8-bit YUV 422.

yuv422p10
10-bit YUV 422.

argb
8-bit RGB.

bgra
8-bit RGB.

rgb10
10-bit RGB.

teletext_lines
If set to nonzero, an additional teletext stream will be captured from the vertical ancillary data. Both SD PAL (576i) and HD (1080i or 1080p) sources are supported. In case of HD sources, OP47 packets are decoded.

This option is a bitmask of the SD PAL VBI lines captured, specifically lines 6 to 22, and lines 318 to 335. Line 6 is the LSB in the mask. Selected lines which do not contain teletext information will be ignored. You can use the special **all** constant to select all possible lines, or **standard** to skip lines 6, 318 and 319, which are not compatible with all receivers.

For SD sources, ffmpeg needs to be compiled with "--enable-libzvbi". For HD sources, on older (pre-4K) DeckLink card models you have to capture in 10 bit mode.

channels
Defines number of audio channels to capture. Must be **2**, **8** or **16**. Defaults to **2**.

duplex_mode
Sets the decklink device duplex/profile mode. Must be **unset**, **half**, **full**, **one_sub_device_full**, **one_sub_device_half**, **two_sub_device_full**, **four_sub_device_half** Defaults to **unset**.

Note: DeckLink SDK 11.0 have replaced the duplex property by a profile property. For the DeckLink Duo 2 and DeckLink Quad 2, a profile is shared between any 2 sub-devices that utilize the same connectors. For the DeckLink 8K Pro, a profile is shared between all 4 sub-devices. So DeckLink 8K Pro support four profiles.

Valid profile modes for DeckLink 8K Pro(with DeckLink SDK >= 11.0): **one_sub_device_full**, **one_sub_device_half**, **two_sub_device_full**, **four_sub_device_half**

Valid profile modes for DeckLink Quad 2 and DeckLink Duo 2: **half**, **full**

timecode_format
Timecode type to include in the frame and video stream metadata. Must be **none**, **rp188vitc**, **rp188vitc2**, **rp188ltc**, **rp188hfr**, **rp188any**, **vitc**, **vitc2**, or **serial**. Defaults to **none** (not included).

In order to properly support 50/60 fps timecodes, the ordering of the queried timecode types for **rp188any** is HFR, VITC1, VITC2 and LTC for >30 fps content. Note that this is slightly different to the ordering used by the DeckLink API, which is HFR, VITC1, LTC, VITC2.

video_input
Sets the video input source. Must be **unset**, **sdi**, **hdmi**, **optical_sdi**, **component**, **composite** or **s_video**. Defaults to **unset**.

audio_input
Sets the audio input source. Must be **unset**, **embedded**, **aes_ebu**, **analog**, **analog_xlr**, **analog_rca** or **microphone**. Defaults to **unset**.

video_pts
Sets the video packet timestamp source. Must be **video**, **audio**, **reference**, **wallclock** or **abs_wallclock**. Defaults to **video**.

audio_pts
Sets the audio packet timestamp source. Must be **video**, **audio**, **reference**, **wallclock** or **abs_wallclock**. Defaults to **audio**.

draw_bars
If set to **true**, color bars are drawn in the event of a signal loss. Defaults to **true**.

queue_size
Sets maximum input buffer size in bytes. If the buffering reaches this value, incoming frames will be dropped. Defaults to **1073741824**.

audio_depth
Sets the audio sample bit depth. Must be **16** or **32**. Defaults to **16**.

decklink_copyts
If set to **true**, timestamps are forwarded as they are without removing the initial offset. Defaults to **false**.

timestamp_align
Capture start time alignment in seconds. If set to nonzero, input frames are dropped till the system timestamp aligns with configured value. Alignment difference of up to one frame duration is tolerated. This is useful for maintaining input synchronization across N different hardware devices deployed for 'N-way' redundancy. The system time of different hardware devices should be synchronized with protocols such as NTP or PTP, before using this option. Note that this method is not foolproof. In some border cases input synchronization may not happen due to thread scheduling jitters in the OS. Either sync could go wrong by 1 frame or in a rarer case **timestamp_align** seconds. Defaults to **0**.

wait_for_tc (*bool*)
Drop frames till a frame with timecode is received. Sometimes serial timecode isn't received with the first input frame. If that happens, the stored stream timecode will be inaccurate. If this option is set to **true**, input frames are dropped till a frame with timecode is received. Option *timecode_format* must be specified. Defaults to **false**.

enable_klv(*bool*)
If set to **true**, extracts KLV data from VANC and outputs KLV packets. KLV VANC packets are joined based on MID and PSC fields and aggregated into one KLV packet. Defaults to **false**.

Examples

- List input devices:

```
ffmpeg -sources decklink
```

- List supported formats:

```
ffmpeg -f decklink -list_formats 1 -i 'Intensity Pro'
```

- Capture video clip at 1080i50:

```
ffmpeg -format_code Hi50 -f decklink -i 'Intensity Pro' -c:a copy -c:v copy output.avi
```

- Capture video clip at 1080i50 10 bit:

  ```
  ffmpeg -raw_format yuv422p10 -format_code Hi50 -f decklink -i 'UltraStudio Mini Recorder' -c:a copy -c:v
  ```

- Capture video clip at 1080i50 with 16 audio channels:

  ```
  ffmpeg -channels 16 -format_code Hi50 -f decklink -i 'UltraStudio Mini Recorder' -c:a copy -c:v copy outpu
  ```

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 or alternative name..

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 the same name (starts at 0, defaults to 0).

audio_device_number
Set audio device number for devices with the 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**>

video_pin_name
Select video capture pin to use by name or alternative name.

audio_pin_name
Select audio capture pin to use by name or alternative name.

crossbar_video_input_pin_number
Select video input pin number for crossbar device. This will be routed to the crossbar device's Video Decoder output pin. Note that changing this value can affect future invocations (sets a new default) until system reboot occurs.

crossbar_audio_input_pin_number
Select audio input pin number for crossbar device. This will be routed to the crossbar device's Audio Decoder output pin. Note that changing this value can affect future invocations (sets a new default) until system reboot occurs.

show_video_device_dialog
If set to **true**, before capture starts, popup a display dialog to the end user, allowing them to change video filter properties and configurations manually. Note that for crossbar devices, adjusting values in this dialog may be needed at times to toggle between PAL (25 fps) and NTSC (29.97) input frame rates, sizes, interlacing, etc. Changing these values can enable different scan rates/frame rates and avoiding green bars at the bottom, flickering scan lines, etc. Note that with some devices, changing these properties can also affect future invocations (sets new defaults) until system reboot occurs.

show_audio_device_dialog
If set to **true**, before capture starts, popup a display dialog to the end user, allowing them to change audio filter properties and configurations manually.

show_video_crossbar_connection_dialog
If set to **true**, before capture starts, popup a display dialog to the end user, allowing them to manually modify crossbar pin routings, when it opens a video device.

show_audio_crossbar_connection_dialog
If set to **true**, before capture starts, popup a display dialog to the end user, allowing them to manually modify crossbar pin routings, when it opens an audio device.

show_analog_tv_tuner_dialog
If set to **true**, before capture starts, popup a display dialog to the end user, allowing them to manually modify TV channels and frequencies.

show_analog_tv_tuner_audio_dialog
If set to **true**, before capture starts, popup a display dialog to the end user, allowing them to manually modify TV audio (like mono vs. stereo, Language A,B or C).

audio_device_load
Load an audio capture filter device from file instead of searching it by name. It may load additional parameters too, if the filter supports the serialization of its properties to. To use this an audio capture source has to be specified, but it can be anything even fake one.

audio_device_save
Save the currently used audio capture filter device and its parameters (if the filter supports it) to a file. If a file with the same name exists it will be overwritten.

video_device_load
Load a video capture filter device from file instead of searching it by name. It may load additional

parameters too, if the filter supports the serialization of its properties to. To use this a video capture source has to be specified, but it can be anything even fake one.

video_device_save
Save the currently used video capture filter device and its parameters (if the filter supports it) to a file. If a file with the same name exists it will be overwritten.

use_video_device_timestamps
If set to **false**, the timestamp for video frames will be derived from the wallclock instead of the timestamp provided by the capture device. This allows working around devices that provide unreliable timestamps.

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"
```

- Specify pin names to capture by name or alternative name, specify alternative device name:

```
$ ffmpeg -f dshow -audio_pin_name "Audio Out" -video_pin_name 2 -i video=video="@device_pnp_\\?\pci
```

- Configure a crossbar device, specifying crossbar pins, allow user to adjust video capture properties at startup:

```
        $ ffmpeg -f dshow -show_video_device_dialog true -crossbar_video_input_pin_number 0
             -crossbar_audio_input_pin_number 3 -i video="AVerMedia BDA Analog Capture":audio="AVerMedia B
```

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.

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

To record from the framebuffer device */dev/fb0* with **ffmpeg**:

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

You can take a single screenshot image with the command:

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

Options

framerate

Set the frame rate. Default is 25.

gdigrab

Win32 GDI-based screen capture device.

This device allows you to capture a region of the display on Windows.

There are two options for the input filename:

```
        desktop
```

or

```
        title=<window_title>
```

The first option will capture the entire desktop, or a fixed region of the desktop. The second option will instead capture the contents of a single window, regardless of its position on the screen.

For example, to grab the entire desktop using **ffmpeg**:

```
ffmpeg -f gdigrab -framerate 6 -i desktop out.mpg
```

Grab a 640x480 region at position "10,20":

```
ffmpeg -f gdigrab -framerate 6 -offset_x 10 -offset_y 20 -video_size vga -i desktop out.mpg
```

Grab the contents of the window named "Calculator"

```
ffmpeg -f gdigrab -framerate 6 -i title=Calculator out.mpg
```

Options

draw_mouse
Specify whether to draw the mouse pointer. Use the value 0 to not draw the pointer. Default value is 1.

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.

Note that *show_region* is incompatible with grabbing the contents of a single window.

For example:

```
ffmpeg -f gdigrab -show_region 1 -framerate 6 -video_size cif -offset_x 10 -offset_y 20 -i desktop out.mpg
```

video_size
Set the video frame size. The default is to capture the full screen if *desktop* is selected, or the full window size if *title=window_title* is selected.

offset_x

When capturing a region with *video_size*, set the distance from the left edge of the screen or desktop.

Note that the offset calculation is from the top left corner of the primary monitor on Windows. If you have a monitor positioned to the left of your primary monitor, you will need to use a negative *offset_x* value to move the region to that monitor.

offset_y

When capturing a region with *video_size*, set the distance from the top edge of the screen or desktop.

Note that the offset calculation is from the top left corner of the primary monitor on Windows. If you have a monitor positioned above your primary monitor, you will need to use a negative *offset_y* value to move the region to that monitor.

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.

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 maximum 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 its 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.

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 -dvbuffer 100000 out.mpg
  ```

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/**>

Options

channels
Set the number of channels. Default is 2.

kmsgrab

KMS video input device.

Captures the KMS scanout framebuffer associated with a specified CRTC or plane as a DRM object that can be passed to other hardware functions.

Requires either DRM master or CAP_SYS_ADMIN to run.

If you don't understand what all of that means, you probably don't want this. Look at **x11grab** instead.

Options

device
DRM device to capture on. Defaults to **/dev/dri/card0**.

format
Pixel format of the framebuffer. This can be autodetected if you are running Linux 5.7 or later, but needs to be provided for earlier versions. Defaults to **bgr0**, which is the most common format used by the Linux console and Xorg X server.

format_modifier
Format modifier to signal on output frames. This is necessary to import correctly into some APIs. It can be autodetected if you are running Linux 5.7 or later, but will need to be provided explicitly when needed in earlier versions. See the libdrm documentation for possible values.

crtc_id
KMS CRTC ID to define the capture source. The first active plane on the given CRTC will be used.

plane_id
KMS plane ID to define the capture source. Defaults to the first active plane found if neither **crtc_id** nor **plane_id** are specified.

framerate
Framerate to capture at. This is not synchronised to any page flipping or framebuffer changes - it just defines the interval at which the framebuffer is sampled. Sampling faster than the framebuffer update rate will generate independent frames with the same content. Defaults to 30.

Examples

- Capture from the first active plane, download the result to normal frames and encode. This will only work if the framebuffer is both linear and mappable - if not, the result may be scrambled or fail to download.

```
        ffmpeg -f kmsgrab -i - -vf 'hwdownload,format=bgr0' output.mp4
```

- Capture from CRTC ID 42 at 60fps, map the result to VAAPI, convert to NV12 and encode as H.264.

```
        ffmpeg -crtc_id 42 -framerate 60 -f kmsgrab -i - -vf 'hwmap=derive_device=vaapi,scale_vaapi=w=1920:h=1
```

- To capture only part of a plane the output can be cropped - this can be used to capture a single window, as long as it has a known absolute position and size. For example, to capture and encode the middle quarter of a 1920x1080 plane:

```
        ffmpeg -f kmsgrab -i - -vf 'hwmap=derive_device=vaapi,crop=960:540:480:270,scale_vaapi=960:540:nv12'
```

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. The filtergraph is specified through the option **graph**.

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.

The suffix "+subcc" can be appended to the output label to create an extra stream with the closed captions packets attached to that output (experimental; only for EIA-608 / CEA-708 for now). The subcc streams are created after all the normal streams, in the order of the corresponding stream. For example, if there is "out19+subcc", "out7+subcc" and up to "out42", the stream #43 is subcc for stream #7 and stream #44 is subcc for stream #19.

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*.

dumpgraph

Dump graph to stderr.

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]"
```

- Dump decoded frames to images and closed captions to a file (experimental):

```
ffmpeg -f lavfi -i "movie=test.ts[out0+subcc]" -map v frame%08d.png -map s -c copy -f rawvideo subcc.bin
```

libcdio

Audio-CD input device based on libcdio.

To enable this input device during configuration you need libcdio installed on your system. It requires the configure option "--enable-libcdio".

This device allows playing and grabbing from an Audio-CD.

For example to copy with **ffmpeg** the entire Audio-CD in */dev/sr0*, you may run the command:

```
ffmpeg -f libcdio -i /dev/sr0 cd.wav
```

Options

speed

Set drive reading speed. Default value is 0.

The speed is specified CD-ROM speed units. The speed is set through the libcdio "cdio_cddap_speed_set" function. On many CD-ROM drives, specifying a value too large will result in using the fastest speed.

paranoia_mode

Set paranoia recovery mode flags. It accepts one of the following values:

disable
verify

overlap
neverskip
full

Default value is **disable**.

For more information about the available recovery modes, consult the paranoia project documentation.

libdc1394

IIDC1394 input device, based on libdc1394 and libraw1394.

Requires the configure option "--enable-libdc1394".

Options

framerate
Set the frame rate. Default is "ntsc", corresponding to a frame rate of "30000/1001".

pixel_format
Select the pixel format. Default is "uyvy422".

video_size
Set the video size given as a string such as "640x480" or "hd720". Default is "qvga".

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 one 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*.

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**.

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.

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**>

Options

sample_rate
Set the sample rate in Hz. Default is 48000.

channels
Set the number of channels. Default is 2.

pulse

PulseAudio input device.

To enable this output device you need to configure FFmpeg with "--enable-libpulse".

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

To list the PulseAudio source devices and their properties you can invoke the command **pactl list sources**.

More information about PulseAudio can be found on <**http://www.pulseaudio.org**>.

Options

server
Connect to a specific PulseAudio server, specified by an IP address. Default server is used when not provided.

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

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

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

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

frame_size
This option does nothing and is deprecated.

fragment_size
Specify the size in bytes of the minimal buffering fragment in PulseAudio, it will affect the audio latency. By default it is set to 50 ms amount of data.

wallclock
Set the initial PTS using the current time. Default is 1.

Examples

Record a stream from default device:

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

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
```

Options

sample_rate
Set the sample rate in Hz. Default is 48000.

channels
Set the number of channels. Default is 2.

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**:

- List supported formats for a video4linux2 device:

  ```
  ffplay -f video4linux2 -list_formats all /dev/video0
  ```

- 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/**>.

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 one 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".

use_libv4l2
Use libv4l2 (v4l-utils) conversion functions. Default is 0.

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.

Options

video_size
Set the video frame size.

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

x11grab

X11 video input device.

To enable this input device during configuration you need libxcb installed on your system. It will be automatically detected during configuration.

This device allows one 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 **xdpyinfo** 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 -framerate 25 -video_size cif -i :0.0 out.mpg
```

Grab at position "10,20":

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

Options

select_region
Specify whether to select the grabbing area graphically using the pointer. A value of 1 prompts the user to select the grabbing area graphically by clicking and dragging. A single click with no dragging will select the whole screen. A region with zero width or height will also select the whole screen. This option overwrites the *video_size*, *grab_x*, and *grab_y* options. Default value is 0.

draw_mouse
Specify whether to draw the mouse pointer. A value of 0 specifies 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 -framerate 25 -video_size 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 -framerate 25 -video_size 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.

region_border

Set the region border thickness if **-show_region 1** is used. Range is 1 to 128 and default is 3 (XCB-based x11grab only).

For example:

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

With *follow_mouse*:

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

window_id

Grab this window, instead of the whole screen. Default value is 0, which maps to the whole screen (root window).

The id of a window can be found using the **xwininfo** program, possibly with options -tree and -root.

If the window is later enlarged, the new area is not recorded. Video ends when the window is closed, unmapped (i.e., iconified) or shrunk beyond the video size (which defaults to the initial window size).

This option disables options **follow_mouse** and **select_region**.

video_size

Set the video frame size. Default is the full desktop or window.

grab_x
grab_y

Set the grabbing region coordinates. They are expressed as offset from the top left corner of the X11 window and correspond to the *x_offset* and *y_offset* parameters in the device name. The default value for both options is 0.

OUTPUT DEVICES

Output devices are configured elements in FFmpeg that can 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 "-devices" of the ff* tools will display the list of enabled output devices.

A description of the currently available output devices follows.

alsa

ALSA (Advanced Linux Sound Architecture) output device.

Examples

- Play a file on default ALSA device:

```
ffmpeg -i INPUT -f alsa default
```

- Play a file on soundcard 1, audio device 7:

```
ffmpeg -i INPUT -f alsa hw:1,7
```

AudioToolbox

AudioToolbox output device.

Allows native output to CoreAudio devices on OSX.

The output filename can be empty (or "-") to refer to the default system output device or a number that refers to the device index as shown using: "-list_devices true".

Alternatively, the audio input device can be chosen by index using the

```
B<-audio_device_index E<lt>INDEXE<gt>>
```

, overriding any device name or index given in the input filename.

All available devices can be enumerated by using **-list_devices true**, listing all device names, UIDs and corresponding indices.

Options

AudioToolbox supports the following options:

-audio_device_index <INDEX>
Specify the audio device by its index. Overrides anything given in the output filename.

Examples

- Print the list of supported devices and output a sine wave to the default device:

```
$ ffmpeg -f lavfi -i sine=r=44100 -f audiotoolbox -list_devices true -
```

- Output a sine wave to the device with the index 2, overriding any output filename:

```
$ ffmpeg -f lavfi -i sine=r=44100 -f audiotoolbox -audio_device_index 2 -
```

caca

CACA output device.

This output device allows one 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**>

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".

Examples

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

```
ffmpeg -i INPUT -c:v 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 -
```

decklink

The decklink output device provides playback capabilities for Blackmagic DeckLink devices.

To enable this output device, you need the Blackmagic DeckLink SDK and you need to configure with the appropriate "--extra-cflags" and "--extra-ldflags". On Windows, you need to run the IDL files through **widl**.

DeckLink is very picky about the formats it supports. Pixel format is always uyvy422, framerate, field order and video size must be determined for your device with **-list_formats 1**. Audio sample rate is always 48 kHz.

Options

list_devices
If set to **true**, print a list of devices and exit. Defaults to **false**. This option is deprecated, please use the "-sinks" option of ffmpeg to list the available output devices.

list_formats
If set to **true**, print a list of supported formats and exit. Defaults to **false**.

preroll
Amount of time to preroll video in seconds. Defaults to **0.5**.

duplex_mode
Sets the decklink device duplex/profile mode. Must be **unset**, **half**, **full**, **one_sub_device_full**, **one_sub_device_half**, **two_sub_device_full**, **four_sub_device_half** Defaults to **unset**.

Note: DeckLink SDK 11.0 have replaced the duplex property by a profile property. For the DeckLink Duo 2 and DeckLink Quad 2, a profile is shared between any 2 sub-devices that utilize the same connectors. For the DeckLink 8K Pro, a profile is shared between all 4 sub-devices. So DeckLink 8K Pro support four profiles.

Valid profile modes for DeckLink 8K Pro(with DeckLink SDK >= 11.0): **one_sub_device_full**, **one_sub_device_half**, **two_sub_device_full**, **four_sub_device_half**

Valid profile modes for DeckLink Quad 2 and DeckLink Duo 2: **half**, **full**

timing_offset
Sets the genlock timing pixel offset on the used output. Defaults to **unset**.

link Sets the SDI video link configuration on the used output. Must be **unset**, **single** link SDI, **dual** link SDI or **quad** link SDI. Defaults to **unset**.

sqd Enable Square Division Quad Split mode for Quad-link SDI output. Must be **unset**, **true** or **false**. Defaults to **unset**.

level_a
Enable SMPTE Level A mode on the used output. Must be **unset**, **true** or **false**. Defaults to **unset**.

vanc_queue_size
Sets maximum output buffer size in bytes for VANC data. If the buffering reaches this value, outgoing VANC data will be dropped. Defaults to **1048576**.

Examples

- List output devices:

```
ffmpeg -sinks decklink
```

- List supported formats:

```
ffmpeg -i test.avi -f decklink -list_formats 1 'DeckLink Mini Monitor'
```

- Play video clip:

```
ffmpeg -i test.avi -f decklink -pix_fmt uyvy422 'DeckLink Mini Monitor'
```

- Play video clip with non-standard framerate or video size:

```
ffmpeg -i test.avi -f decklink -pix_fmt uyvy422 -s 720x486 -r 24000/1001 'DeckLink Mini Monitor'
```

fbdev

Linux framebuffer output 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.

Options

xoffset
yoffset
Set x/y coordinate of top left corner. Default is 0.

Examples

Play a file on framebuffer device */dev/fb0*. Required pixel format depends on current framebuffer settings.

```
ffmpeg -re -i INPUT -c:v rawvideo -pix_fmt bgra -f fbdev /dev/fb0
```

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

opengl

OpenGL output device.

To enable this output device you need to configure FFmpeg with "--enable-opengl".

This output device allows one to render to OpenGL context. Context may be provided by application or default SDL window is created.

When device renders to external context, application must implement handlers for following messages: "AV_DEV_TO_APP_CREATE_WINDOW_BUFFER" - create OpenGL context on current thread. "AV_DEV_TO_APP_PREPARE_WINDOW_BUFFER" - make OpenGL context current. "AV_DEV_TO_APP_DISPLAY_WINDOW_BUFFER" - swap buffers. "AV_DEV_TO_APP_DESTROY_WINDOW_BUFFER" - destroy OpenGL context. Application is also required to inform a device about current resolution by sending "AV_APP_TO_DEV_WINDOW_SIZE" message.

Options

background
Set background color. Black is a default.

no_window
Disables default SDL window when set to non-zero value. Application must provide OpenGL context and both "window_size_cb" and "window_swap_buffers_cb" callbacks when set.

window_title

Set the SDL window title, if not specified default to the filename specified for the output device. Ignored when **no_window** is set.

window_size

Set preferred window size, can be a string of the form widthxheight or a video size abbreviation. If not specified it defaults to the size of the input video, downscaled according to the aspect ratio. Mostly usable when **no_window** is not set.

Examples

Play a file on SDL window using OpenGL rendering:

```
ffmpeg  -i INPUT -f opengl "window title"
```

oss

OSS (Open Sound System) output device.

pulse

PulseAudio output device.

To enable this output device you need to configure FFmpeg with "--enable-libpulse".

More information about PulseAudio can be found on <**http://www.pulseaudio.org**>

Options

server

Connect to a specific PulseAudio server, specified by an IP address. Default server is used when not provided.

name

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

stream_name

Specify the stream name PulseAudio will use when showing active streams, by default it is set to the specified output name.

device

Specify the device to use. Default device is used when not provided. List of output devices can be obtained with command **pactl list sinks**.

buffer_size
buffer_duration
Control the size and duration of the PulseAudio buffer. A small buffer gives more control, but requires more frequent updates.

buffer_size specifies size in bytes while **buffer_duration** specifies duration in milliseconds.

When both options are provided then the highest value is used (duration is recalculated to bytes using stream parameters). If they are set to 0 (which is default), the device will use the default PulseAudio duration value. By default PulseAudio set buffer duration to around 2 seconds.

prebuf
Specify pre-buffering size in bytes. The server does not start with playback before at least **prebuf** bytes are available in the buffer. By default this option is initialized to the same value as **buffer_size** or **buffer_duration** (whichever is bigger).

minreq
Specify minimum request size in bytes. The server does not request less than **minreq** bytes from the client, instead waits until the buffer is free enough to request more bytes at once. It is recommended to not set this option, which will initialize this to a value that is deemed sensible by the server.

Examples

Play a file on default device on default server:

```
ffmpeg  -i INPUT -f pulse "stream name"
```

sdl

SDL (Simple DirectMedia Layer) output device.

"sdl2" can be used as alias for "sdl".

This output device allows one 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/**>

Options

window_borderless
Set SDL window border off. Default value is 0 (enable window border).

window_enable_quit
Enable quit action (using window button or keyboard key) when non-zero value is provided. Default value is 1 (enable quit action).

window_fullscreen
Set fullscreen mode when non-zero value is provided. Default value is zero.

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.

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

window_x
window_y
Set the position of the window on the screen.

Interactive commands

The window created by the device can be controlled through the following interactive commands.

q, ESC
Quit the device immediately.

Examples

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

```
ffmpeg -i INPUT -c:v rawvideo -pix_fmt yuv420p -window_size qcif -f sdl "SDL output"
```

sndio

sndio audio output device.

v4l2

Video4Linux2 output device.

xv

XV (XVideo) output device.

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

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_id

When set to non-zero value then device doesn’t create new window, but uses existing one with provided *window_id*. By default this options is set to zero and device creates its own window.

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. Ignored when *window_id* is set.

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. Ignored when *window_id* is set.

window_title

Set the window title, if not specified default to the filename specified for the output device. Ignored when *window_id* is set.

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

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
```

SEE ALSO

ffmpeg(1), **ffplay**(1), **ffprobe**(1), **libavdevice**(3)

AUTHORS

The FFmpeg developers.

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

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