

---

**v4l2ctl**

***Release 0.1a5***

**Feb 28, 2021**



---

## Contents:

---

<b>1</b>	<b>What v4l2ctl is</b>	<b>1</b>
<b>2</b>	<b>Project Status</b>	<b>3</b>
<b>3</b>	<b>The documentation</b>	<b>5</b>
3.1	v4l2ctl package . . . . .	5
<b>4</b>	<b>Internal documentation</b>	<b>11</b>
4.1	Internal documentation . . . . .	11
<b>5</b>	<b>Copyright and licence</b>	<b>15</b>
5.1	See also . . . . .	15
	<b>Python Module Index</b>	<b>17</b>
	<b>Index</b>	<b>19</b>



# CHAPTER 1

---

## What v4l2ctl is

---

v4l2ctl is a python package to control v4L2 (Video 4 Linux 2) drivers.



## CHAPTER 2

---

### Project Status

---

The project is still in its early development stages. It can, however, already be used to read the name, driver, version and capabilities of a v4l2 device driver.





## 3.1 v4l2ctl package

### 3.1.1 Module contents

**class** `v4l2ctl.V4l2Device` (*device*='/dev/video0')

Initialize the V4l2Device object and read its basic information.

**Keyword Arguments** **device** (*str*, *path-like*, *int*) – the video device (default `r"/dev/video0"`) if an *int* is given, it is assumed to be number after “video” in “/dev”.

**Raises** `IoctlError` – if a non-video device file is given.

**buffer\_type**

The buffer type (see `V4l2BufferType`) required for several operations. This attribute does not change anything in the device itself. It is used by other operations.

**bus**

The bus through which this device is connected (read-only).

**capabilities**

The device specific capabilities (read-only). These are the capabilities associated with this dev-file only. The physical device can have more than one dev-file, and hence more capabilities. See `physical_capabilities`.

**close()**

Flush and close the IO object.

This method has no effect if the file is already closed.

**cropping\_capabilities**

The cropping capabilities (read-only). These are the cropping capabilities of this video device.

**Only valid for these buffer types:**

- `V4l2BufferType.VIDEO_CAPTURE`

- `V4l2BufferType.VIDEO_CAPTURE_MPLANE`
- `V4l2BufferType.VIDEO_OUTPUT`
- `V4l2BufferType.VIDEO_OUTPUT_MPLANE`
- `V4l2BufferType.VIDEO_OVERLAY`

**cropping\_rectangle**

The cropping rectangle (see `V4l2Rectangle`).

---

**Note:** The cropping rectange is specfic to the set buffer type. (See [buffer\\_type](#))

---

**device**

The device file (read-only).

**driver**

The linux driver (read-only).

**fileno()**

Returns underlying file descriptor if one exists.

`OSError` is raised if the IO object does not use a file descriptor.

**flush()**

Flush write buffers, if applicable.

This is not implemented for read-only and non-blocking streams.

**formats**

A generator for the suported formats by this video device.

---

**Note:** The formats are specfic to the set buffer type. (See [buffer\\_type](#))

---

**isatty()**

Return whether this is an ‘interactive’ stream.

Return `False` if it can’t be determined.

**iter\_buffer\_formats(buffer\_type)**

Iterate over the formats supported by a certain buffer.

**Keyword Arguments** **buffer\_type** – see [V4l2BufferType](#).

**Returns** a generator

**static iter\_devices(skip\_links=True)**

Return an iterator over the available v4l2 devices.

**Keyword Arguments** **skip\_links** (*bool*) – skip links and return every device only once (default `True`)

**Returns** an iterator

**name**

The card name (read-only).

**physical\_capabilities**

The general physical capabilities (read-only). These are the capabilities associated with the physical device as a while, and not limited to this dev-file only.

**readable()**

Return whether object was opened for reading.

If False, read() will raise OSError.

**readline** (*size=-1*)

Read and return a line from the stream.

If size is specified, at most size bytes will be read.

The line terminator is always b'n' for binary files; for text files, the newlines argument to open can be used to select the line terminator(s) recognized.

**readlines** (*hint=-1*)

Return a list of lines from the stream.

hint can be specified to control the number of lines read: no more lines will be read if the total size (in bytes/characters) of all lines so far exceeds hint.

**seekable()**

Return whether object supports random access.

If False, seek(), tell() and truncate() will raise OSError. This method may need to do a test seek().

**supported\_buffer\_types**

The supported buffer types by this video device (read-only).

**version**

The kernel version as a string (read-only).

**version\_tuple**

The kernel version as a tuple (read-only).

**writable()**

Return whether object was opened for writing.

If False, write() will raise OSError.

**writelines** (*lines*)

Write a list of lines to stream.

Line separators are not added, so it is usual for each of the lines provided to have a line separator at the end.

**class v4l2ctl.V4l2Capabilities**

The v4l2 capability flags.

These are the flags defining the supported capabilities of a V4l2 devince.

**Example**

Check if device /dev/video0 supports video capturing:

```
vid_dev = VideoDevice(r"/dev/video0")
if CapabilityFlags.VIDEO_CAPTURE in vid_dev.capabilities:
    start_recording()
```

**ASYNCIO = 33554432**

Async I/O.

**AUDIO = 131072**

Has audio support.

**DEVICE\_CAPS = 2147483648**  
Sets device capabilities field.

**EXT\_PIX\_FORMAT = 2097152**  
Supports the extended pixel format.

**HW\_FREQ\_SEEK = 1024**  
Can do hardware frequency seek.

**META\_CAPTURE = 8388608**  
Is a metadata capture device.

**META\_OUTPUT = 134217728**  
Is a metadata output device.

**MODULATOR = 524288**  
Has a modulator.

**RADIO = 262144**  
Is a radio device.

**RDS\_CAPTURE = 256**  
RDS data capture.

**RDS\_OUTPUT = 2048**  
Is an RDS encoder.

**READWRITE = 16777216**  
Read/write systemcalls.

**SDR\_CAPTURE = 1048576**  
Is a SDR capture device.

**SDR\_OUTPUT = 4194304**  
Is a SDR output device.

**SLICED\_VBI\_CAPTURE = 64**  
Is a sliced VBI capture device.

**SLICED\_VBI\_OUTPUT = 128**  
Is a sliced VBI output device.

**STREAMING = 67108864**  
Streaming I/O ioctls.

**TOUCH = 268435456**  
Is a touch device.

**TUNER = 65536**  
Has a tuner.

**VBI\_CAPTURE = 16**  
Is a raw VBI capture device.

**VBI\_OUTPUT = 32**  
Is a raw VBI output device.

**VIDEO\_CAPTURE = 1**  
Is a video capture device.

**VIDEO\_CAPTURE\_MPLANE = 4096**  
Is a video capture device that supports multiplanar formats.

**VIDEO\_M2M = 32768**

Is a video mem-to-mem device.

**VIDEO\_M2M\_MPLANE = 16384**

Is a video mem-to-mem device that supports multiplanar formats.

**VIDEO\_OUTPUT = 2**

Is a video output device.

**VIDEO\_OUTPUT\_MPLANE = 8192**

Is a video output device that supports multiplanar formats.

**VIDEO\_OUTPUT\_OVERLAY = 512**

Can do video output overlay.

**VIDEO\_OVERLAY = 4**

Can do video overlay.

**class v4l2ctl.V4l2BufferType**

The v4l2 buffer types.

Used with **:attribute:'enum\_fmt'**.

**META\_CAPTURE = 13**

Buffer for metadata capture, see Metadata Interface.

**META\_OUTPUT = 14**

Buffer for metadata output, see Metadata Interface.

**SDR\_CAPTURE = 11**

Buffer for Software Defined Radio (SDR) capture stream, see Software Defined Radio Interface (SDR).

**SDR\_OUTPUT = 12**

Buffer for Software Defined Radio (SDR) output stream, see Software Defined Radio Interface (SDR).

**SLICED\_VBI\_CAPTURE = 6**

Buffer of a sliced VBI capture stream, see Sliced VBI Data Interface.

**SLICED\_VBI\_OUTPUT = 7**

Buffer of a sliced VBI output stream, see Sliced VBI Data Interface.

**VBI\_CAPTURE = 4**

Buffer of a raw VBI capture stream, see Raw VBI Data Interface.

**VBI\_OUTPUT = 5**

Buffer of a raw VBI output stream, see Raw VBI Data Interface.

**VIDEO\_CAPTURE = 1**

Buffer of a single-planar video capture stream, see Video Capture Interface.

**VIDEO\_CAPTURE\_MPLANE = 9**

Buffer of a multi-planar video capture stream, see Video Capture Interface.

**VIDEO\_OUTPUT = 2**

Buffer of a single-planar video output stream, see Video Output Interface.

**VIDEO\_OUTPUT\_MPLANE = 10**

Buffer of a multi-planar video output stream, see Video Output Interface.

**VIDEO\_OUTPUT\_OVERLAY = 8**

Buffer for video output overlay (OSD), see Video Output Overlay Interface.

**VIDEO\_OVERLAY = 3**

Buffer for video overlay, see Video Overlay Interface.

**class** v4l2ctl.V4l2Formats

An Enum-Container for all V4l2Formats. This class delegates its operations to the contained enums. For more information, see:

py:class:V4l2PixFormats

py:class:V4l2SdrFormats

py:class:V4l2TouchFormats

py:class:V4l2MetaFormats

**class** v4l2ctl.V4l2FormatDescFlags

The v4l2 format flags.

**COMPRESSED = 1**

This is a compressed format.

**EMULATED = 2**

This format is not native to the device but emulated through software (usually libv4l2), where possible try to use a native format instead for better performance.

**NONE = 0**

No flags are set.

**exception** v4l2ctl.IoctlError(*device, name, request, return\_code, extra\_msg=None*)

Raised when ioctl() returns a non-zero value.

**exception** v4l2ctl.FeatureNotSupported

## 4.1 Internal documentation

This is the documentation of the internals of v4l2ctl

### 4.1.1 Module ioctlmacros

### 4.1.2 Module v4l2interface

### 4.1.3 Module v4l2device

**exception** v4l2ctl.v4l2device.FeatureNotSupported

Bases: Exception

**\_\_module\_\_** = 'v4l2ctl.v4l2device'

**\_\_weakref\_\_**

list of weak references to the object (if defined)

**class** v4l2ctl.v4l2device.V4l2Device (device='/dev/video0')

Bases: io.IOBase

Initialize the V4l2Device object and read its basic information.

**Keyword Arguments** **device** (*str, path-like, int*) – the video device (default `r"/dev/video0"`) if an int is given, it is assumed to be number after “video” in “/dev”.

**Raises** IoctlError – if a non-video device file is given.

**\_\_abstractmethods\_\_** = frozenset()

**\_\_enter\_\_**()

**\_\_exit\_\_**(*exc\_type, exc, tb*)

`__init__ (device='/dev/video0')`

Initialize self. See `help(type(self))` for accurate signature.

`__iter__ ()`

Implement `iter(self)`.

`__module__ = 'v4l2ctl.v4l2device'`

`__next__ ()`

Implement `next(self)`.

`__repr__ ()`

Return `repr(self)`.

`_abc_impl = <_abc_data object>`

`_open ()`

**buffer\_type**

The buffer type (see `V4l2BufferType`) required for several operations. This attribute does not change anything in the device itself. It is used by other operations.

**bus**

The bus through which this device is connected (read-only).

**capabilities**

The device specific capabilities (read-only). These are the capabilities associated with this dev-file only. The physical device can have more than one dev-file, and hence more capabilities. See `physical_capabilities`.

**close ()**

Flush and close the IO object.

This method has no effect if the file is already closed.

**closed**

**cropping\_buffer\_types = [`V4l2BufferType.VIDEO_CAPTURE: 1`], `V4l2BufferType.VIDEO_CAPTURE`]**

**cropping\_capabilities**

The cropping capabilities (read-only). These are the cropping capabilities of this video device.

**Only valid for these buffer types:**

- `V4l2BufferType.VIDEO_CAPTURE`
- `V4l2BufferType.VIDEO_CAPTURE_MPLANE`
- `V4l2BufferType.VIDEO_OUTPUT`
- `V4l2BufferType.VIDEO_OUTPUT_MPLANE`
- `V4l2BufferType.VIDEO_OVERLAY`

**cropping\_rectangle**

The cropping rectangle (see `V4l2Rectangle`).

---

**Note:** The cropping rectangle is specific to the set buffer type. (See *buffer\_type*)

---

**device**

The device file (read-only).

**driver**

The linux driver (read-only).



**fileno()**

Returns underlying file descriptor if one exists.

OSError is raised if the IO object does not use a file descriptor.

**flush()**

Flush write buffers, if applicable.

This is not implemented for read-only and non-blocking streams.

**formats**

A generator for the supported formats by this video device.

---

**Note:** The formats are specific to the set buffer type. (See [buffer\\_type](#))

---

**isatty()**

Return whether this is an ‘interactive’ stream.

Return False if it can’t be determined.

**iter\_buffer\_formats(buffer\_type)**

Iterate over the formats supported by a certain buffer.

**Keyword Arguments** **buffer\_type** – see V4l2BufferType.

**Returns** a generator

**static iter\_devices(skip\_links=True)**

Return an iterator over the available v4l2 devices.

**Keyword Arguments** **skip\_links** (*bool*) – skip links and return every device only once (default True)

**Returns** an iterator

**name**

The card name (read-only).

**physical\_capabilities**

The general physical capabilities (read-only). These are the capabilities associated with the physical device as a whole, and not limited to this dev-file only.

**readable()**

Return whether object was opened for reading.

If False, read() will raise OSError.

**readline(size=-1)**

Read and return a line from the stream.

If size is specified, at most size bytes will be read.

The line terminator is always b’n’ for binary files; for text files, the newlines argument to open can be used to select the line terminator(s) recognized.

**readlines(hint=-1)**

Return a list of lines from the stream.

hint can be specified to control the number of lines read: no more lines will be read if the total size (in bytes/characters) of all lines so far exceeds hint.

**seekable()**

Return whether object supports random access.

If False, seek(), tell() and truncate() will raise OSError. This method may need to do a test seek().

**supported\_buffer\_types**

The supported buffer types by this video device (read-only).

**version**

The kernel version as a string (read-only).

**version\_tuple**

The kernel version as a tuple (read-only).

**writable()**

Return whether object was opened for writing.

If False, write() will raise OSError.

**writelines(*lines*)**

Write a list of lines to stream.

Line separators are not added, so it is usual for each of the lines provided to have a line separator at the end.

**class** v4l2ctl.v4l2device.V4l2DeviceIterator(*skip\_links*)

Bases: object

**\_\_dict\_\_** = mappingproxy({'\_\_module\_\_': 'v4l2ctl.v4l2device', '\_v4l2\_device\_prefixes':

**\_\_init\_\_**(*skip\_links*)

Initialize self. See help(type(self)) for accurate signature.

**\_\_iter\_\_**()

**\_\_module\_\_** = 'v4l2ctl.v4l2device'

**\_\_weakref\_\_**

list of weak references to the object (if defined)

**\_v4l2\_device\_prefixes** = ['video', 'radio', 'vbi', 'swradio', 'v4l-subdev']

Or check the website's genindex.

---

### Copyright and licence

---

Copyright 2020, Michael Israel  
Licensed under the [EURL](#)

#### 5.1 See also

If your purpose is to implement high end streaming applications or some sort of video processing, you might want to take a look at [gstreamer](#) and [opencv](#).



### V

`v4l2ctl`, [5](#)

`v4l2ctl.v4l2device`, [11](#)



## Symbols

[\\_\\_abstractmethods\\_\\_](#) ([v4l2ctl.v4l2device.V4l2Device](#) attribute), 11  
[\\_\\_dict\\_\\_](#) ([v4l2ctl.v4l2device.V4l2DeviceIterator](#) attribute), 14  
[\\_\\_enter\\_\\_](#)() ([v4l2ctl.v4l2device.V4l2Device](#) method), 11  
[\\_\\_exit\\_\\_](#)() ([v4l2ctl.v4l2device.V4l2Device](#) method), 11  
[\\_\\_init\\_\\_](#)() ([v4l2ctl.v4l2device.V4l2Device](#) method), 11  
[\\_\\_init\\_\\_](#)() ([v4l2ctl.v4l2device.V4l2DeviceIterator](#) method), 14  
[\\_\\_iter\\_\\_](#)() ([v4l2ctl.v4l2device.V4l2Device](#) method), 12  
[\\_\\_iter\\_\\_](#)() ([v4l2ctl.v4l2device.V4l2DeviceIterator](#) method), 14  
[\\_\\_module\\_\\_](#) ([v4l2ctl.v4l2device.FeatureNotSupported](#) attribute), 11  
[\\_\\_module\\_\\_](#) ([v4l2ctl.v4l2device.V4l2Device](#) attribute), 12  
[\\_\\_module\\_\\_](#) ([v4l2ctl.v4l2device.V4l2DeviceIterator](#) attribute), 14  
[\\_\\_next\\_\\_](#)() ([v4l2ctl.v4l2device.V4l2Device](#) method), 12  
[\\_\\_repr\\_\\_](#)() ([v4l2ctl.v4l2device.V4l2Device](#) method), 12  
[\\_\\_weakref\\_\\_](#) ([v4l2ctl.v4l2device.FeatureNotSupported](#) attribute), 11  
[\\_\\_weakref\\_\\_](#) ([v4l2ctl.v4l2device.V4l2DeviceIterator](#) attribute), 14  
[\\_abc\\_impl](#) ([v4l2ctl.v4l2device.V4l2Device](#) attribute), 12  
[\\_open](#)() ([v4l2ctl.v4l2device.V4l2Device](#) method), 12  
[\\_v4l2\\_device\\_prefixes](#) ([v4l2ctl.v4l2device.V4l2DeviceIterator](#) attribute), 14

## A

[ASYNCIO](#) ([v4l2ctl.V4l2Capabilities](#) attribute), 7  
[AUDIO](#) ([v4l2ctl.V4l2Capabilities](#) attribute), 7

## B

[buffer\\_type](#) ([v4l2ctl.V4l2Device](#) attribute), 5  
[buffer\\_type](#) ([v4l2ctl.v4l2device.V4l2Device](#) attribute), 12  
[bus](#) ([v4l2ctl.V4l2Device](#) attribute), 5  
[bus](#) ([v4l2ctl.v4l2device.V4l2Device](#) attribute), 12

## C

[capabilities](#) ([v4l2ctl.V4l2Device](#) attribute), 5  
[capabilities](#) ([v4l2ctl.v4l2device.V4l2Device](#) attribute), 12  
[close](#)() ([v4l2ctl.V4l2Device](#) method), 5  
[close](#)() ([v4l2ctl.v4l2device.V4l2Device](#) method), 12  
[closed](#) ([v4l2ctl.v4l2device.V4l2Device](#) attribute), 12  
[COMPRESSED](#) ([v4l2ctl.V4l2FormatDescFlags](#) attribute), 10  
[cropping\\_buffer\\_types](#) ([v4l2ctl.v4l2device.V4l2Device](#) attribute), 12  
[cropping\\_capabilities](#) ([v4l2ctl.V4l2Device](#) attribute), 5  
[cropping\\_capabilities](#) ([v4l2ctl.v4l2device.V4l2Device](#) attribute), 12  
[cropping\\_rectangle](#) ([v4l2ctl.V4l2Device](#) attribute), 6  
[cropping\\_rectangle](#) ([v4l2ctl.v4l2device.V4l2Device](#) attribute), 12

## D

[device](#) ([v4l2ctl.V4l2Device](#) attribute), 6  
[device](#) ([v4l2ctl.v4l2device.V4l2Device](#) attribute), 12  
[DEVICE\\_CAPS](#) ([v4l2ctl.V4l2Capabilities](#) attribute), 7  
[driver](#) ([v4l2ctl.V4l2Device](#) attribute), 6

driver (*v4l2ctl.v4l2device.V4l2Device attribute*), 12

## E

EMULATED (*v4l2ctl.V4l2FormatDescFlags attribute*), 10

EXT\_PIX\_FORMAT (*v4l2ctl.V4l2Capabilities attribute*), 8

## F

FeatureNotSupported, 10, 11

fileno() (*v4l2ctl.V4l2Device method*), 6

fileno() (*v4l2ctl.v4l2device.V4l2Device method*), 12

flush() (*v4l2ctl.V4l2Device method*), 6

flush() (*v4l2ctl.v4l2device.V4l2Device method*), 13

formats (*v4l2ctl.V4l2Device attribute*), 6

formats (*v4l2ctl.v4l2device.V4l2Device attribute*), 13

## H

HW\_FREQ\_SEEK (*v4l2ctl.V4l2Capabilities attribute*), 8

## I

IoctlError, 10

isatty() (*v4l2ctl.V4l2Device method*), 6

isatty() (*v4l2ctl.v4l2device.V4l2Device method*), 13

iter\_buffer\_formats() (*v4l2ctl.V4l2Device method*), 6

iter\_buffer\_formats() (*v4l2ctl.v4l2device.V4l2Device method*), 13

iter\_devices() (*v4l2ctl.V4l2Device static method*), 6

iter\_devices() (*v4l2ctl.v4l2device.V4l2Device static method*), 13

## M

META\_CAPTURE (*v4l2ctl.V4l2BufferType attribute*), 9

META\_CAPTURE (*v4l2ctl.V4l2Capabilities attribute*), 8

META\_OUTPUT (*v4l2ctl.V4l2BufferType attribute*), 9

META\_OUTPUT (*v4l2ctl.V4l2Capabilities attribute*), 8

MODULATOR (*v4l2ctl.V4l2Capabilities attribute*), 8

## N

name (*v4l2ctl.V4l2Device attribute*), 6

name (*v4l2ctl.v4l2device.V4l2Device attribute*), 13

NONE (*v4l2ctl.V4l2FormatDescFlags attribute*), 10

## P

physical\_capabilities (*v4l2ctl.V4l2Device attribute*), 6

physical\_capabilities (*v4l2ctl.v4l2device.V4l2Device attribute*), 13

## R

RADIO (*v4l2ctl.V4l2Capabilities attribute*), 8

RDS\_CAPTURE (*v4l2ctl.V4l2Capabilities attribute*), 8

RDS\_OUTPUT (*v4l2ctl.V4l2Capabilities attribute*), 8

readable() (*v4l2ctl.V4l2Device method*), 6

readable() (*v4l2ctl.v4l2device.V4l2Device method*), 13

readline() (*v4l2ctl.V4l2Device method*), 7

readline() (*v4l2ctl.v4l2device.V4l2Device method*), 13

readlines() (*v4l2ctl.V4l2Device method*), 7

readlines() (*v4l2ctl.v4l2device.V4l2Device method*), 13

READWRITE (*v4l2ctl.V4l2Capabilities attribute*), 8

## S

SDR\_CAPTURE (*v4l2ctl.V4l2BufferType attribute*), 9

SDR\_CAPTURE (*v4l2ctl.V4l2Capabilities attribute*), 8

SDR\_OUTPUT (*v4l2ctl.V4l2BufferType attribute*), 9

SDR\_OUTPUT (*v4l2ctl.V4l2Capabilities attribute*), 8

seekable() (*v4l2ctl.V4l2Device method*), 7

seekable() (*v4l2ctl.v4l2device.V4l2Device method*), 13

SLICED\_VBI\_CAPTURE (*v4l2ctl.V4l2BufferType attribute*), 9

SLICED\_VBI\_CAPTURE (*v4l2ctl.V4l2Capabilities attribute*), 8

SLICED\_VBI\_OUTPUT (*v4l2ctl.V4l2BufferType attribute*), 9

SLICED\_VBI\_OUTPUT (*v4l2ctl.V4l2Capabilities attribute*), 8

STREAMING (*v4l2ctl.V4l2Capabilities attribute*), 8

supported\_buffer\_types (*v4l2ctl.V4l2Device attribute*), 7

supported\_buffer\_types (*v4l2ctl.v4l2device.V4l2Device attribute*), 14

## T

TOUCH (*v4l2ctl.V4l2Capabilities attribute*), 8

TUNER (*v4l2ctl.V4l2Capabilities attribute*), 8

## V

V4l2BufferType (*class in v4l2ctl*), 9

V4l2Capabilities (*class in v4l2ctl*), 7

v4l2ctl (*module*), 5

v4l2ctl.v4l2device (*module*), 11

V4l2Device (*class in v4l2ctl*), 5

V4l2Device (*class in v4l2ctl.v4l2device*), 11

V4l2DeviceIterator (*class in v4l2ctl.v4l2device*), 14

V4l2FormatDescFlags (*class in v4l2ctl*), 10

V4l2Formats (*class in v4l2ctl*), 9

VBI\_CAPTURE (*v4l2ctl.V4l2BufferType attribute*), 9

VBI\_CAPTURE (*v4l2ctl.V4l2Capabilities attribute*), 8

VBI\_OUTPUT (*v4l2ctl.V4l2BufferType attribute*), 9



VBI\_OUTPUT (*v4l2ctl.V4l2Capabilities attribute*), 8  
 version (*v4l2ctl.V4l2Device attribute*), 7  
 version (*v4l2ctl.v4l2device.V4l2Device attribute*), 14  
 version\_tuple (*v4l2ctl.V4l2Device attribute*), 7  
 version\_tuple (*v4l2ctl.v4l2device.V4l2Device attribute*), 14  
 VIDEO\_CAPTURE (*v4l2ctl.V4l2BufferType attribute*), 9  
 VIDEO\_CAPTURE (*v4l2ctl.V4l2Capabilities attribute*), 8  
 VIDEO\_CAPTURE\_MPLANE (*v4l2ctl.V4l2BufferType attribute*), 9  
 VIDEO\_CAPTURE\_MPLANE (*v4l2ctl.V4l2Capabilities attribute*), 8  
 VIDEO\_M2M (*v4l2ctl.V4l2Capabilities attribute*), 8  
 VIDEO\_M2M\_MPLANE (*v4l2ctl.V4l2Capabilities attribute*), 9  
 VIDEO\_OUTPUT (*v4l2ctl.V4l2BufferType attribute*), 9  
 VIDEO\_OUTPUT (*v4l2ctl.V4l2Capabilities attribute*), 9  
 VIDEO\_OUTPUT\_MPLANE (*v4l2ctl.V4l2BufferType attribute*), 9  
 VIDEO\_OUTPUT\_MPLANE (*v4l2ctl.V4l2Capabilities attribute*), 9  
 VIDEO\_OUTPUT\_OVERLAY (*v4l2ctl.V4l2BufferType attribute*), 9  
 VIDEO\_OUTPUT\_OVERLAY (*v4l2ctl.V4l2Capabilities attribute*), 9  
 VIDEO\_OVERLAY (*v4l2ctl.V4l2BufferType attribute*), 9  
 VIDEO\_OVERLAY (*v4l2ctl.V4l2Capabilities attribute*), 9

## W

writable() (*v4l2ctl.V4l2Device method*), 7  
 writable() (*v4l2ctl.v4l2device.V4l2Device method*), 14  
 writelines() (*v4l2ctl.V4l2Device method*), 7  
 writelines() (*v4l2ctl.v4l2device.V4l2Device method*), 14