ioctl VIDIOC_EXPBUF

Name

VIDIOC_EXPBUF - Export a buffer as a DMABUF file descriptor.

Synopsis

```
int ioctl(int fd, VIDIOC_EXPBUF, struct v4l2_exportbuffer *argp)
```

Arguments

`fd`

> File descriptor returned by :c:func:`open()`.

`argp`

> Pointer to struct :c:type:`v4l2_exportbuffer`.

Description

This ioctl is an extension to the :ref:`memory mapping <mmap>` I/O method, therefore it is available only for `V4L2_MEMORY_MMAP` buffers. It can be used to export a buffer as a DMABUF file at any time after buffers have been allocated with the :ref:`VIDIOC_REQBUFS` ioctl.

Unknown interpreted text role "ref".

To export a buffer, applications fill struct :c:type:`v4l2_exportbuffer`. The `type` field is set to the same buffer type as was previously used with struct :c:type:`v4l2_requestbuffers` `type`. Applications must also set the `index` field. Valid index numbers range from zero to the number of buffers allocated with :ref:`VIDIOC_REQBUFS` (struct :c:type:`v4l2_requestbuffers` `count`) minus one. For the multi-planar API, applications set the `plane` field to the index of the plane to be exported. Valid planes range from zero to the maximal number of valid planes for the currently active format. For the single-planar API, applications must set `plane` to zero. Additional flags may be posted in the `flags` field. Refer to a manual for open() for details. Currently only O_CLOEXEC, O_RDONLY, O_WRONLY, and O_RDWR are supported. All other fields must be set to zero. In the case of multi-planar API, every plane is exported separately using multiple :ref:`VIDIOC_EXPBUF` calls.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\(linux-master)(Documentation)(userspace-api)(media)(v4l)vidioc-expbuf.rst`, **line 40**); *backlink*
>
> Unknown interpreted text role "c:type".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\(linux-master)(Documentation)(userspace-api)(media)(v4l)vidioc-expbuf.rst`, **line 40**); *backlink*
>
> Unknown interpreted text role "c:type".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\(linux-master)(Documentation)(userspace-api)(media)(v4l)vidioc-expbuf.rst`, **line 40**); *backlink*
>
> Unknown interpreted text role "ref".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\(linux-master)(Documentation)(userspace-api)(media)(v4l)vidioc-expbuf.rst`, **line 40**); *backlink*
>
> Unknown interpreted text role "c:type".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\(linux-master)(Documentation)(userspace-api)(media)(v4l)vidioc-expbuf.rst`, **line 40**); *backlink*
>
> Unknown interpreted text role "ref".

After calling :ref:`VIDIOC_EXPBUF` the `fd` field will be set by a driver. This is a DMABUF file descriptor. The application may pass it to other DMABUF-aware devices. Refer to :ref:`DMABUF importing <dmabuf>` for details about importing DMABUF files into V4L2 nodes. It is recommended to close a DMABUF file when it is no longer used to allow the associated memory to be reclaimed.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\(linux-master)(Documentation)(userspace-api)(media)(v4l)vidioc-expbuf.rst`, **line 58**); *backlink*
>
> Unknown interpreted text role "ref".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\(linux-master)(Documentation)(userspace-api)(media)(v4l)vidioc-expbuf.rst`, **line 58**); *backlink*
>
> Unknown interpreted text role "ref".

Examples

```c
int buffer_export(int v4lfd, enum v4l2_buf_type bt, int index, int *dmafd)
{
    struct v4l2_exportbuffer expbuf;

    memset(&expbuf, 0, sizeof(expbuf));
```

```c
    expbuf.type = bt;
    expbuf.index = index;
    if (ioctl(v4lfd, VIDIOC_EXPBUF, &expbuf) == -1) {
        perror("VIDIOC_EXPBUF");
        return -1;
    }

    *dmafd = expbuf.fd;

    return 0;
}
```

```c
int buffer_export_mp(int v4lfd, enum v4l2_buf_type bt, int index,
    int dmafd[], int n_planes)
{
    int i;

    for (i = 0; i < n_planes; ++i) {
        struct v4l2_exportbuffer expbuf;

        memset(&expbuf, 0, sizeof(expbuf));
        expbuf.type = bt;
        expbuf.index = index;
        expbuf.plane = i;
        if (ioctl(v4lfd, VIDIOC_EXPBUF, &expbuf) == -1) {
            perror("VIDIOC_EXPBUF");
            while (i)
                close(dmafd[--i]);
            return -1;
        }
        dmafd[i] = expbuf.fd;
    }

    return 0;
}
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\(linux-master)(Documentation)(userspace-api)(media)(v4l)vidioc-expbuf.rst`, line 113)

Unknown directive type "c:type".

```
.. c:type:: v4l2_exportbuffer
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\(linux-master)(Documentation)(userspace-api)(media)(v4l)vidioc-expbuf.rst`, line 115)

Unknown directive type "tabularcolumns".

```
.. tabularcolumns:: |p{4.4cm}|p{4.4cm}|p{8.5cm}|
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\(linux-master)(Documentation)(userspace-api)(media)(v4l)vidioc-expbuf.rst`, line 117)

Unknown directive type "flat-table".

```
.. flat-table:: struct v4l2_exportbuffer
    :header-rows:  0
    :stub-columns: 0
    :widths:       1 1 2

    * - __u32
      - ``type``
      - Type of the buffer, same as struct
        :c:type:`v4l2_format` ``type`` or struct
        :c:type:`v4l2_requestbuffers` ``type``, set
        by the application. See :c:type:`v4l2_buf_type`
    * - __u32
      - ``index``
      - Number of the buffer, set by the application. This field is only
        used for :ref:`memory mapping <mmap>` I/O and can range from
        zero to the number of buffers allocated with the
        :ref:`VIDIOC_REQBUFS` and/or
        :ref:`VIDIOC_CREATE_BUFS` ioctls.
```

```
    * - __u32
      - ``plane``
      - Index of the plane to be exported when using the multi-planar API.
        Otherwise this value must be set to zero.
    * - __u32
      - ``flags``
      - Flags for the newly created file, currently only ``O_CLOEXEC``,
        ``O_RDONLY``, ``O_WRONLY``, and ``O_RDWR`` are supported, refer to
        the manual of open() for more details.
    * - __s32
      - ``fd``
      - The DMABUF file descriptor associated with a buffer. Set by the
        driver.
    * - __u32
      - ``reserved[11]``
      - Reserved field for future use. Drivers and applications must set
        the array to zero.
```

Return Value

On success 0 is returned, on error -1 and the `errno` variable is set appropriately. The generic error codes are described at the :ref:`Generic Error Codes <gen-errors>` chapter.

EINVAL
> A queue is not in MMAP mode or DMABUF exporting is not supported or `flags` or `type` or `index` or `plane` fields are invalid.