Memory-to-memory Stateless Video Decoder Interface

A stateless decoder is a decoder that works without retaining any kind of state between processed frames. This means that each frame is decoded independently of any previous and future frames, and that the client is responsible for maintaining the decoding state and providing it to the decoder with each decoding request. This is in contrast to the stateful video decoder interface, where the hardware and driver maintain the decoding state and all the client has to do is to provide the raw encoded stream and dequeue decoded frames in display order.

This section describes how user-space ("the client") is expected to communicate with stateless decoders in order to successfully decode an encoded stream. Compared to stateful codecs, the decoder/client sequence is simpler, but the cost of this simplicity is extra complexity in the client which is responsible for maintaining a consistent decoding state.

Stateless decoders make use of the :ref:`media-request-api`. A stateless decoder must expose the `V4L2_BUF_CAP_SUPPORTS_REQUESTS` capability on its `OUTPUT` queue when :c:func:`VIDIOC_REQBUFS` or :c:func:`VIDIOC_CREATE_BUFS` are invoked.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\[linux-master][Documentation][userspace-api][media][v4l]dev-stateless-decoder.rst, line 24); *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]dev-stateless-decoder.rst, line 24); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\[linux-master][Documentation][userspace-api][media][v4l]dev-stateless-decoder.rst, line 24); *backlink*

Unknown interpreted text role "c:func".

Depending on the encoded formats supported by the decoder, a single decoded frame may be the result of several decode requests (for instance, H.264 streams with multiple slices per frame). Decoders that support such formats must also expose the `V4L2_BUF_CAP_SUPPORTS_M2M_HOLD_CAPTURE_BUF` capability on their `OUTPUT` queue.

Querying capabilities

1. To enumerate the set of coded formats supported by the decoder, the client calls :c:func:`VIDIOC_ENUM_FMT` on the `OUTPUT` queue.

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\[linux-master][Documentation][userspace-api][media][v4l]dev-stateless-decoder.rst, line 38); *backlink*

   Unknown interpreted text role "c:func".

   - The driver must always return the full set of supported `OUTPUT` formats, irrespective of the format currently set on the `CAPTURE` queue.
   - Simultaneously, the driver must restrain the set of values returned by codec-specific capability controls (such as H.264 profiles) to the set actually supported by the hardware.

2. To enumerate the set of supported raw formats, the client calls :c:func:`VIDIOC_ENUM_FMT` on the `CAPTURE` queue.

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\[linux-master][Documentation][userspace-api][media][v4l]dev-stateless-decoder.rst, line 48); *backlink*

   Unknown interpreted text role "c:func".

   - The driver must return only the formats supported for the format currently active on the `OUTPUT` queue.

- Depending on the currently set OUTPUT format, the set of supported raw formats may depend on the value of some codec-dependent controls. The client is responsible for making sure that these controls are set before querying the CAPTURE queue. Failure to do so will result in the default values for these controls being used, and a returned set of formats that may not be usable for the media the client is trying to decode.

3. The client may use :c:func:`VIDIOC_ENUM_FRAMESIZES` to detect supported resolutions for a given format, passing desired pixel format in :c:type:`v4l2_frmsizeenum`'s `pixel_format`.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\[linux-master][Documentation][userspace-api][media][v4l]dev-stateless-decoder.rst, line 61); *backlink***
>
> Unknown interpreted text role "c:func".

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\[linux-master][Documentation][userspace-api][media][v4l]dev-stateless-decoder.rst, line 61); *backlink***
>
> Unknown interpreted text role "c:type".

4. Supported profiles and levels for the current OUTPUT format, if applicable, may be queried using their respective controls via :c:func:`VIDIOC_QUERYCTRL`.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\[linux-master][Documentation][userspace-api][media][v4l]dev-stateless-decoder.rst, line 65); *backlink***
>
> Unknown interpreted text role "c:func".

Initialization

1. Set the coded format on the OUTPUT queue via :c:func:`VIDIOC_S_FMT`.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\[linux-master][Documentation][userspace-api][media][v4l]dev-stateless-decoder.rst, line 72); *backlink***
>
> Unknown interpreted text role "c:func".

- **Required fields:**

  `type`
  : a `V4L2_BUF_TYPE_*` enum appropriate for OUTPUT.

  `pixelformat`
  : a coded pixel format.

  `width`, `height`
  : coded width and height parsed from the stream.

  other fields
  : follow standard semantics.

> **Note**
>
> Changing the OUTPUT format may change the currently set CAPTURE format. The driver will derive a new CAPTURE format from the OUTPUT format being set, including resolution, colorimetry parameters, etc. If the client needs a specific CAPTURE format, it must adjust it afterwards.

2. Call :c:func:`VIDIOC_S_EXT_CTRLS` to set all the controls (parsed headers, etc.) required by the OUTPUT format to enumerate the CAPTURE formats.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\[linux-master][Documentation][userspace-api][media][v4l]dev-stateless-decoder.rst, line 96);**

backlink

Unknown interpreted text role "c:func".

3. Call :c:func:`VIDIOC_G_FMT` for `CAPTURE` queue to get the format for the destination buffers parsed/decoded from the bytestream.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\[linux-master][Documentation][userspace-api][media][v4l]dev-stateless-decoder.rst, line 99); *backlink*

Unknown interpreted text role "c:func".

- **Required fields:**

`type`
: a `V4L2_BUF_TYPE_*` enum appropriate for `CAPTURE`.

- **Returned fields:**

`width`, `height`
: frame buffer resolution for the decoded frames.

`pixelformat`
: pixel format for decoded frames.

`num_planes` (for _MPLANE `type` only)
: number of planes for pixelformat.

`sizeimage`, `bytesperline`
: as per standard semantics; matching frame buffer format.

Note

The value of `pixelformat` may be any pixel format supported for the `OUTPUT` format, based on the hardware capabilities. It is suggested that the driver chooses the preferred/optimal format for the current configuration. For example, a YUV format may be preferred over an RGB format, if an additional conversion step would be required for RGB.

4. *[optional]* Enumerate `CAPTURE` formats via :c:func:`VIDIOC_ENUM_FMT` on the `CAPTURE` queue. The client may use this ioctl to discover which alternative raw formats are supported for the current `OUTPUT` format and select one of them via :c:func:`VIDIOC_S_FMT`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\[linux-master][Documentation][userspace-api][media][v4l]dev-stateless-decoder.rst, line 129); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\[linux-master][Documentation][userspace-api][media][v4l]dev-stateless-decoder.rst, line 129); *backlink*

Unknown interpreted text role "c:func".

Note

The driver will return only formats supported for the currently selected `OUTPUT` format and currently set controls, even if more formats may be supported by the decoder in general.

For example, a decoder may support YUV and RGB formats for resolutions 1920x1088 and lower, but only YUV for higher resolutions (due to hardware limitations). After setting a resolution of 1920x1088 or lower as the `OUTPUT` format, :c:func:`VIDIOC_ENUM_FMT` may return a set of YUV and RGB pixel formats, but after setting a resolution higher than 1920x1088, the driver will not return RGB pixel formats, since they are unsupported for this resolution.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\[linux-

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\[linux-master][Documentation][userspace-api][media][v4l]dev-stateless-decoder.rst, line 140); *backlink*

Unknown interpreted text role "c:func".

5. *[optional]* Choose a different `CAPTURE` format than suggested via :c:func:`VIDIOC_S_FMT` on `CAPTURE` queue. It is possible for the client to choose a different format than selected/suggested by the driver in :c:func:`VIDIOC_G_FMT`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\[linux-master][Documentation][userspace-api][media][v4l]dev-stateless-decoder.rst, line 148); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\[linux-master][Documentation][userspace-api][media][v4l]dev-stateless-decoder.rst, line 148); *backlink*

Unknown interpreted text role "c:func".

   - **Required fields:**

     `type`
       a `V4L2_BUF_TYPE_*` enum appropriate for `CAPTURE`.

     `pixelformat`
       a raw pixel format.

     `width`, `height`
       frame buffer resolution of the decoded stream; typically unchanged from what was returned with :c:func:`VIDIOC_G_FMT`, but it may be different if the hardware supports composition and/or scaling.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\[linux-master][Documentation][userspace-api][media][v4l]dev-stateless-decoder.rst, line 162); *backlink*

Unknown interpreted text role "c:func".

   After performing this step, the client must perform step 3 again in order to obtain up-to-date information about the buffers size and layout.

6. Allocate source (bytestream) buffers via :c:func:`VIDIOC_REQBUFS` on `OUTPUT` queue.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\[linux-master][Documentation][userspace-api][media][v4l]dev-stateless-decoder.rst, line 169); *backlink*

Unknown interpreted text role "c:func".

   - **Required fields:**

     `count`
       requested number of buffers to allocate; greater than zero.

     `type`
       a `V4L2_BUF_TYPE_*` enum appropriate for `OUTPUT`.

     `memory`
       follows standard semantics.

   - **Return fields:**

     `count`
       actual number of buffers allocated.

- If required, the driver will adjust `count` to be equal or bigger to the minimum of required number of `OUTPUT` buffers for the given format and requested count. The client must check this value after the ioctl returns to get the actual number of buffers allocated.

7. Allocate destination (raw format) buffers via :c:func:`VIDIOC_REQBUFS` on the `CAPTURE` queue.

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\[linux-master][Documentation][userspace-api][media][v4l]dev-stateless-decoder.rst, line 193); *backlink*

   Unknown interpreted text role "c:func".

   - **Required fields:**

     `count`
     : requested number of buffers to allocate; greater than zero. The client is responsible for deducing the minimum number of buffers required for the stream to be properly decoded (taking e.g. reference frames into account) and pass an equal or bigger number.

     `type`
     : a `V4L2_BUF_TYPE_*` enum appropriate for `CAPTURE`.

     `memory`
     : follows standard semantics. `V4L2_MEMORY_USERPTR` is not supported for `CAPTURE` buffers.

   - **Return fields:**

     `count`
     : adjusted to allocated number of buffers, in case the codec requires more buffers than requested.

   - The driver must adjust count to the minimum of required number of `CAPTURE` buffers for the current format, stream configuration and requested count. The client must check this value after the ioctl returns to get the number of buffers allocated.

8. Allocate requests (likely one per `OUTPUT` buffer) via

   :c:func:`MEDIA_IOC_REQUEST_ALLOC` on the media device.

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\[linux-master][Documentation][userspace-api][media][v4l]dev-stateless-decoder.rst, line 223); *backlink*

   Unknown interpreted text role "c:func".

9. Start streaming on both `OUTPUT` and `CAPTURE` queues via

   :c:func:`VIDIOC_STREAMON`.

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\[linux-master][Documentation][userspace-api][media][v4l]dev-stateless-decoder.rst, line 226); *backlink*

   Unknown interpreted text role "c:func".

Decoding

For each frame, the client is responsible for submitting at least one request to which the following is attached:

- The amount of encoded data expected by the codec for its current configuration, as a buffer submitted to the `OUTPUT` queue. Typically, this corresponds to one frame worth of encoded data, but some formats may allow (or require) different amounts per unit.
- All the metadata needed to decode the submitted encoded data, in the form of controls relevant to the format being decoded.

The amount of data and contents of the source `OUTPUT` buffer, as well as the controls that must be set on the request, depend on the active coded pixel format and might be affected by codec-specific extended controls, as stated in documentation of each format.

If there is a possibility that the decoded frame will require one or more decode requests after the current one in order to be produced, then the client must set the `V4L2_BUF_FLAG_M2M_HOLD_CAPTURE_BUF` flag on the `OUTPUT` buffer. This will result in the (potentially partially) decoded `CAPTURE` buffer not being made available for dequeueing, and reused for the next decode request if the timestamp of the next `OUTPUT` buffer has not changed.

A typical frame would thus be decoded using the following sequence:

1. Queue an `OUTPUT` buffer containing one unit of encoded bytestream data for the decoding request, using :c:func:`VIDIOC_QBUF`.

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\[linux-master][Documentation][userspace-api][media][v4l]dev-stateless-decoder.rst, line 255); *backlink*

   Unknown interpreted text role "c:func".

   - **Required fields:**

     `index`
     : index of the buffer being queued.

     `type`
     : type of the buffer.

     `bytesused`
     : number of bytes taken by the encoded data frame in the buffer.

     `flags`
     : the `V4L2_BUF_FLAG_REQUEST_FD` flag must be set. Additionally, if we are not sure that the current decode request is the last one needed to produce a fully decoded frame, then `V4L2_BUF_FLAG_M2M_HOLD_CAPTURE_BUF` must also be set.

     `request_fd`
     : must be set to the file descriptor of the decoding request.

     `timestamp`
     : must be set to a unique value per frame. This value will be propagated into the decoded frame's buffer and can also be used to use this frame as the reference of another. If using multiple decode requests per frame, then the timestamps of all the `OUTPUT` buffers for a given frame must be identical. If the timestamp changes, then the currently held `CAPTURE` buffer will be made available for dequeuing and the current request will work on a new `CAPTURE` buffer.

2. Set the codec-specific controls for the decoding request, using :c:func:`VIDIOC_S_EXT_CTRLS`.

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\[linux-master][Documentation][userspace-api][media][v4l]dev-stateless-decoder.rst, line 287); *backlink*

   Unknown interpreted text role "c:func".

   - **Required fields:**

     `which`
     : must be `V4L2_CTRL_WHICH_REQUEST_VAL`.

     `request_fd`
     : must be set to the file descriptor of the decoding request.

     other fields
     : other fields are set as usual when setting controls. The `controls` array must contain all the codec-specific controls required to decode a frame.

   Note

   It is possible to specify the controls in different invocations of :c:func:`VIDIOC_S_EXT_CTRLS`, or to overwrite a previously set control, as long as `request_fd` and `which` are properly set. The controls state at the moment of request submission is the one that will be considered.

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\[linux-master][Documentation][userspace-api][media][v4l]dev-stateless-decoder.rst, line 305); *backlink*

   Unknown interpreted text role "c:func".

   Note

The order in which steps 1 and 2 take place is interchangeable.

3. Submit the request by invoking :c:func:`MEDIA_REQUEST_IOC_QUEUE` on the request FD.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\[linux-master][Documentation][userspace-api][media][v4l]dev-stateless-decoder.rst, line 314); *backlink*

Unknown interpreted text role "c:func".

If the request is submitted without an OUTPUT buffer, or if some of the required controls are missing from the request, then :c:func:`MEDIA_REQUEST_IOC_QUEUE` will return -ENOENT. If more than one OUTPUT buffer is queued, then it will return -EINVAL. :c:func:`MEDIA_REQUEST_IOC_QUEUE` returning non-zero means that no CAPTURE buffer will be produced for this request.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\[linux-master][Documentation][userspace-api][media][v4l]dev-stateless-decoder.rst, line 317); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\[linux-master][Documentation][userspace-api][media][v4l]dev-stateless-decoder.rst, line 317); *backlink*

Unknown interpreted text role "c:func".

CAPTURE buffers must not be part of the request, and are queued independently. They are returned in decode order (i.e. the same order as coded frames were submitted to the OUTPUT queue).

Runtime decoding errors are signaled by the dequeued CAPTURE buffers carrying the V4L2_BUF_FLAG_ERROR flag. If a decoded reference frame has an error, then all following decoded frames that refer to it also have the V4L2_BUF_FLAG_ERROR flag set, although the decoder will still try to produce (likely corrupted) frames.

Buffer management while decoding

Contrary to stateful decoders, a stateless decoder does not perform any kind of buffer management: it only guarantees that dequeued CAPTURE buffers can be used by the client for as long as they are not queued again. "Used" here encompasses using the buffer for compositing or display.

A dequeued capture buffer can also be used as the reference frame of another buffer.

A frame is specified as reference by converting its timestamp into nanoseconds, and storing it into the relevant member of a codec-dependent control structure. The :c:func:`v4l2_timeval_to_ns` function must be used to perform that conversion. The timestamp of a frame can be used to reference it as soon as all its units of encoded data are successfully submitted to the OUTPUT queue.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\[linux-master][Documentation][userspace-api][media][v4l]dev-stateless-decoder.rst, line 344); *backlink*

Unknown interpreted text role "c:func".

A decoded buffer containing a reference frame must not be reused as a decoding target until all the frames referencing it have been decoded. The safest way to achieve this is to refrain from queueing a reference buffer until all the decoded frames referencing it have been dequeued. However, if the driver can guarantee that buffers queued to the CAPTURE queue are processed in queued order, then user-space can take advantage of this guarantee and queue a reference buffer when the following conditions are met:

1. All the requests for frames affected by the reference frame have been queued, and
2. A sufficient number of CAPTURE buffers to cover all the decoded referencing frames have been queued.

When queuing a decoding request, the driver will increase the reference count of all the resources associated with reference frames. This means that the client can e.g. close the DMABUF file descriptors of reference frame buffers if it won't need them afterwards.

Seeking

In order to seek, the client just needs to submit requests using input buffers corresponding to the new stream position. It must however be aware that resolution may have changed and follow the dynamic resolution change sequence in that case. Also depending on the codec used, picture parameters (e.g. SPS/PPS for H.264) may have changed and the client is responsible for making sure that a valid state is sent to the decoder.

The client is then free to ignore any returned CAPTURE buffer that comes from the pre-seek position.

Pausing

In order to pause, the client can just cease queuing buffers onto the OUTPUT queue. Without source bytestream data, there is no data to process and the codec will remain idle.

Dynamic resolution change

If the client detects a resolution change in the stream, it will need to perform the initialization sequence again with the new resolution:

1. If the last submitted request resulted in a CAPTURE buffer being held by the use of the V4L2_BUF_FLAG_M2M_HOLD_CAPTURE_BUF flag, then the last frame is not available on the CAPTURE queue. In this case, a V4L2_DEC_CMD_FLUSH command shall be sent. This will make the driver dequeue the held CAPTURE buffer.
2. Wait until all submitted requests have completed and dequeue the corresponding output buffers.
3. Call :c:func:`VIDIOC_STREAMOFF` on both the OUTPUT and CAPTURE queues.

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\[linux-master][Documentation][userspace-api][media][v4l]dev-stateless-decoder.rst, line 403); ***backlink***

   Unknown interpreted text role "c:func".

4. Free all CAPTURE buffers by calling :c:func:`VIDIOC_REQBUFS` on the CAPTURE queue with a buffer count of zero.

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\[linux-master][Documentation][userspace-api][media][v4l]dev-stateless-decoder.rst, line 406); ***backlink***

   Unknown interpreted text role "c:func".

5. Perform the initialization sequence again (minus the allocation of OUTPUT buffers), with the new resolution set on the OUTPUT queue. Note that due to resolution constraints, a different format may need to be picked on the CAPTURE queue.

Drain

If the last submitted request resulted in a CAPTURE buffer being held by the use of the V4L2_BUF_FLAG_M2M_HOLD_CAPTURE_BUF flag, then the last frame is not available on the CAPTURE queue. In this case, a V4L2_DEC_CMD_FLUSH command shall be sent. This will make the driver dequeue the held CAPTURE buffer.

After that, in order to drain the stream on a stateless decoder, the client just needs to wait until all the submitted requests are completed.