Sub-device Interface

The complex nature of V4L2 devices, where hardware is often made of several integrated circuits that need to interact with each other in a controlled way, leads to complex V4L2 drivers. The drivers usually reflect the hardware model in software, and model the different hardware components as software blocks called sub-devices.

V4L2 sub-devices are usually kernel-only objects. If the V4L2 driver implements the media device API, they will automatically inherit from media entities. Applications will be able to enumerate the sub-devices and discover the hardware topology using the media entities, pads and links enumeration API.

In addition to make sub-devices discoverable, drivers can also choose to make them directly configurable by applications. When both the sub-device driver and the V4L2 device driver support this, sub-devices will feature a character device node on which ioctls can be called to

- query, read and write sub-devices controls
- subscribe and unsubscribe to events and retrieve them
- negotiate image formats on individual pads

Sub-device character device nodes, conventionally named `/dev/v4l-subdev*`, use major number 81.

Drivers may opt to limit the sub-device character devices to only expose operations that do not modify the device state. In such a case the sub-devices are referred to as `read-only` in the rest of this documentation, and the related restrictions are documented in individual ioctls.

Controls

Most V4L2 controls are implemented by sub-device hardware. Drivers usually merge all controls and expose them through video device nodes. Applications can control all sub-devices through a single interface.

Complex devices sometimes implement the same control in different pieces of hardware. This situation is common in embedded platforms, where both sensors and image processing hardware implement identical functions, such as contrast adjustment, white balance or faulty pixels correction. As the V4L2 controls API doesn't support several identical controls in a single device, all but one of the identical controls are hidden.

Applications can access those hidden controls through the sub-device node with the V4L2 control API described in :ref:`control`. The ioctls behave identically as when issued on V4L2 device nodes, with the exception that they deal only with controls implemented in the sub-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-subdev.rst, line 55); *backlink***
>
> Unknown interpreted text role "ref".

Depending on the driver, those controls might also be exposed through one (or several) V4L2 device nodes.

Events

V4L2 sub-devices can notify applications of events as described in :ref:`event`. The API behaves identically as when used on V4L2 device nodes, with the exception that it only deals with events generated by the sub-device. Depending on the driver, those events might also be reported on one (or several) V4L2 device nodes.

> **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-subdev.rst, line 68); *backlink***
>
> Unknown interpreted text role "ref".

Pad-level Formats

> **Warning**
>
> Pad-level formats are only applicable to very complex devices that need to expose low-level format configuration to user space. Generic V4L2 applications do *not* need to use the API described in this section.

> **Note**
>
> For the purpose of this section, the term *format* means the combination of media bus data format, frame width and frame height.

Image formats are typically negotiated on video capture and output devices using the format and :ref:`selection`

<VIDIOC_SUBDEV_G_SELECTION>` ioctls. The driver is responsible for configuring every block in the video pipeline according to the requested format at the pipeline input and/or output.

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-subdev.rst`, line 92); *backlink*

Unknown interpreted text role "ref".

For complex devices, such as often found in embedded systems, identical image sizes at the output of a pipeline can be achieved using different hardware configurations. One such example is shown on :ref:`pipeline-scaling`, where image scaling can be performed on both the video sensor and the host image processing hardware.

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-subdev.rst`, line 98); *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-subdev.rst`, line 107)

Unknown directive type "kernel-figure".

```
.. kernel-figure:: pipeline.dot
    :alt:   pipeline.dot
    :align: center

    Image Format Negotiation on Pipelines

    High quality and high speed pipeline configuration
```

The sensor scaler is usually of less quality than the host scaler, but scaling on the sensor is required to achieve higher frame rates. Depending on the use case (quality vs. speed), the pipeline must be configured differently. Applications need to configure the formats at every point in the pipeline explicitly.

Drivers that implement the :ref:`media API <media-controller-intro>` can expose pad-level image format configuration to applications. When they do, applications can use the :ref:`VIDIOC_SUBDEV_G_FMT <VIDIOC_SUBDEV_G_FMT>` and :ref:`VIDIOC_SUBDEV_S_FMT <VIDIOC_SUBDEV_G_FMT>` ioctls. to negotiate formats on a per-pad basis.

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-subdev.rst`, line 123); *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-subdev.rst`, line 123); *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-subdev.rst`, line 123); *backlink*

Unknown interpreted text role "ref".

Applications are responsible for configuring coherent parameters on the whole pipeline and making sure that connected pads have compatible formats. The pipeline is checked for formats mismatch at :ref:`VIDIOC_STREAMON <VIDIOC_STREAMON>` time, and an `EPIPE` error code is then returned if the configuration is invalid.

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-subdev.rst`, line 130); *backlink*

Unknown interpreted text role "ref".

Pad-level image format configuration support can be tested by calling the :ref:`VIDIOC_SUBDEV_G_FMT` ioctl on pad 0. If the driver returns an `EINVAL` error code pad-level format configuration is not supported by the sub-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-subdev.rst`, line 136); *backlink*
>
> Unknown interpreted text role "ref".

Format Negotiation

Acceptable formats on pads can (and usually do) depend on a number of external parameters, such as formats on other pads, active links, or even controls. Finding a combination of formats on all pads in a video pipeline, acceptable to both application and driver, can't rely on formats enumeration only. A format negotiation mechanism is required.

Central to the format negotiation mechanism are the get/set format operations. When called with the `which` argument set to :ref:`V4L2_SUBDEV_FORMAT_TRY <VIDIOC_SUBDEV_G_FMT>`, the :ref:`VIDIOC_SUBDEV_G_FMT <VIDIOC_SUBDEV_G_FMT>` and :ref:`VIDIOC_SUBDEV_S_FMT <VIDIOC_SUBDEV_G_FMT>` ioctls operate on a set of formats parameters that are not connected to the hardware configuration. Modifying those 'try' formats leaves the device state untouched (this applies to both the software state stored in the driver and the hardware state stored in the device itself).

> **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-subdev.rst`, line 151); *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-subdev.rst`, line 151); *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-subdev.rst`, line 151); *backlink*
>
> Unknown interpreted text role "ref".

While not kept as part of the device state, try formats are stored in the sub-device file handles. A :ref:`VIDIOC_SUBDEV_G_FMT <VIDIOC_SUBDEV_G_FMT>` call will return the last try format set *on the same sub-device file handle*. Several applications querying the same sub-device at the same time will thus not interact with each other.

> **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-subdev.rst`, line 161); *backlink*
>
> Unknown interpreted text role "ref".

To find out whether a particular format is supported by the device, applications use the :ref:`VIDIOC_SUBDEV_S_FMT <VIDIOC_SUBDEV_G_FMT>` ioctl. Drivers verify and, if needed, change the requested `format` based on device requirements and return the possibly modified value. Applications can then choose to try a different format or accept the returned value and continue.

> **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-subdev.rst`, line 168); *backlink*
>
> Unknown interpreted text role "ref".

Formats returned by the driver during a negotiation iteration are guaranteed to be supported by the device. In particular, drivers guarantee that a returned format will not be further changed if passed to an :ref:`VIDIOC_SUBDEV_S_FMT <VIDIOC_SUBDEV_G_FMT>` call as-is (as long as external parameters, such as formats on other pads or links' configuration are not changed).

> **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-subdev.rst`, line 176); *backlink*
>
> Unknown interpreted text role "ref".

Drivers automatically propagate formats inside sub-devices. When a try or active format is set on a pad, corresponding formats on other pads of the same sub-device can be modified by the driver. Drivers are free to modify formats as required by the device. However, they should comply with the following rules when possible:

- Formats should be propagated from sink pads to source pads. Modifying a format on a source pad should not modify the format on any sink pad.
- Sub-devices that scale frames using variable scaling factors should reset the scale factors to default values when sink pads formats are modified. If the 1:1 scaling ratio is supported, this means that source pads formats should be reset to the sink pads formats.

Formats are not propagated across links, as that would involve propagating them from one sub-device file handle to another. Applications must then take care to configure both ends of every link explicitly with compatible formats. Identical formats on the two ends of a link are guaranteed to be compatible. Drivers are free to accept different formats matching device requirements as being compatible.

:ref:`sample-pipeline-config` shows a sample configuration sequence for the pipeline described in :ref:`pipeline-scaling` (table columns list entity names and pad numbers).

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-subdev.rst, line 205); *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-subdev.rst, line 205); *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-subdev.rst, line 216)

Unknown directive type "tabularcolumns".

```
.. tabularcolumns:: |p{2.0cm}|p{2.1cm}|p{2.1cm}|p{2.1cm}|p{2.1cm}|p{2.1cm}|p{2.1cm}|
```

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-subdev.rst, line 220)

Unknown directive type "flat-table".

```
.. flat-table:: Sample Pipeline Configuration
    :header-rows:  1
    :stub-columns: 0
    :widths: 5 5 5 5 5 5 5

    * -
      - Sensor/0

        format
      - Frontend/0

        format
      - Frontend/1

        format
      - Scaler/0

        format
      - Scaler/0

        compose selection rectangle
      - Scaler/1

        format
    * - Initial state
      - 2048x1536

        SGRBG8_1X8
      - (default)
      - (default)
      - (default)
      - (default)
      - (default)
    * - Configure frontend sink format
```

```
      - 2048x1536

        SGRBG8_1X8
      - *2048x1536*

        *SGRBG8_1X8*
      - *2046x1534*

        *SGRBG8_1X8*
      - (default)
      - (default)
      - (default)
    * - Configure scaler sink format
      - 2048x1536

        SGRBG8_1X8
      - 2048x1536

        SGRBG8_1X8
      - 2046x1534

        SGRBG8_1X8
      - *2046x1534*

        *SGRBG8_1X8*
      - *0,0/2046x1534*
      - *2046x1534*

        *SGRBG8_1X8*
    * - Configure scaler sink compose selection
      - 2048x1536

        SGRBG8_1X8
      - 2048x1536

        SGRBG8_1X8
      - 2046x1534

        SGRBG8_1X8
      - 2046x1534

        SGRBG8_1X8
      - *0,0/1280x960*
      - *1280x960*

        *SGRBG8_1X8*
```

1. Initial state. The sensor source pad format is set to its native 3MP size and V4L2_MBUS_FMT_SGRBG8_1X8 media bus code. Formats on the host frontend and scaler sink and source pads have the default values, as well as the compose rectangle on the scaler's sink pad.
2. The application configures the frontend sink pad format's size to 2048x1536 and its media bus code to V4L2_MBUS_FMT_SGRBG_1X8. The driver propagates the format to the frontend source pad.
3. The application configures the scaler sink pad format's size to 2046x1534 and the media bus code to V4L2_MBUS_FMT_SGRBG_1X8 to match the frontend source size and media bus code. The media bus code on the sink pad is set to V4L2_MBUS_FMT_SGRBG_1X8. The driver propagates the size to the compose selection rectangle on the scaler's sink pad, and the format to the scaler source pad.
4. The application configures the size of the compose selection rectangle of the scaler's sink pad 1280x960. The driver propagates the size to the scaler's source pad format.

When satisfied with the try results, applications can set the active formats by setting the `which` argument to `V4L2_SUBDEV_FORMAT_ACTIVE`. Active formats are changed exactly as try formats by drivers. To avoid modifying the hardware state during format negotiation, applications should negotiate try formats first and then modify the active settings using the try formats returned during the last negotiation iteration. This guarantees that the active format will be applied as-is by the driver without being modified.

Selections: cropping, scaling and composition

Many sub-devices support cropping frames on their input or output pads (or possible even on both). Cropping is used to select the area of interest in an image, typically on an image sensor or a video decoder. It can also be used as part of digital zoom implementations to select the area of the image that will be scaled up.

Crop settings are defined by a crop rectangle and represented in a struct :c:type:`v4l2_rect` by the coordinates of the top left corner and the rectangle size. Both the coordinates and sizes are expressed in pixels.

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-subdev.rst`, line 346); *backlink*

Unknown interpreted text role "c:type".

As for pad formats, drivers store try and active rectangles for the selection targets :ref:`v4l2-selections-common`.

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-subdev.rst, line 351); *backlink*

Unknown interpreted text role "ref".

On sink pads, cropping is applied relative to the current pad format. The pad format represents the image size as received by the sub-device from the previous block in the pipeline, and the crop rectangle represents the sub-image that will be transmitted further inside the sub-device for processing.

The scaling operation changes the size of the image by scaling it to new dimensions. The scaling ratio isn't specified explicitly, but is implied from the original and scaled image sizes. Both sizes are represented by struct :c:type:`v4l2_rect`.

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-subdev.rst, line 360); *backlink*

Unknown interpreted text role "c:type".

Scaling support is optional. When supported by a subdev, the crop rectangle on the subdev's sink pad is scaled to the size configured using the :ref:`VIDIOC_SUBDEV_S_SELECTION <VIDIOC_SUBDEV_G_SELECTION>` IOCTL using `V4L2_SEL_TGT_COMPOSE` selection target on the same pad. If the subdev supports scaling but not composing, the top and left values are not used and must always be set to 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-subdev.rst, line 365); *backlink*

Unknown interpreted text role "ref".

On source pads, cropping is similar to sink pads, with the exception that the source size from which the cropping is performed, is the COMPOSE rectangle on the sink pad. In both sink and source pads, the crop rectangle must be entirely contained inside the source image size for the crop operation.

The drivers should always use the closest possible rectangle the user requests on all selection targets, unless specifically told otherwise. `V4L2_SEL_FLAG_GE` and `V4L2_SEL_FLAG_LE` flags may be used to round the image size either up or down. :ref:`v4l2-selection-flags`

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-subdev.rst, line 379); *backlink*

Unknown interpreted text role "ref".

Types of selection targets

Actual targets

Actual targets (without a postfix) reflect the actual hardware configuration at any point of time. There is a BOUNDS target corresponding to every actual target.

BOUNDS targets

BOUNDS targets is the smallest rectangle that contains all valid actual rectangles. It may not be possible to set the actual rectangle as large as the BOUNDS rectangle, however. This may be because e.g. a sensor's pixel array is not rectangular but cross-shaped or round. The maximum size may also be smaller than the BOUNDS rectangle.

Order of configuration and format propagation

Inside subdevs, the order of image processing steps will always be from the sink pad towards the source pad. This is also reflected in the order in which the configuration must be performed by the user: the changes made will be propagated to any subsequent stages. If this behaviour is not desired, the user must set `V4L2_SEL_FLAG_KEEP_CONFIG` flag. This flag causes no propagation of the changes are allowed in any circumstances. This may also cause the accessed rectangle to be adjusted by the driver, depending on the properties of the underlying hardware.

The coordinates to a step always refer to the actual size of the previous step. The exception to this rule is the sink compose rectangle, which refers to the sink compose bounds rectangle --- if it is supported by the hardware.

1. Sink pad format. The user configures the sink pad format. This format defines the parameters of the image the entity receives through the pad for further processing.
2. Sink pad actual crop selection. The sink pad crop defines the crop performed to the sink pad format.

3. Sink pad actual compose selection. The size of the sink pad compose rectangle defines the scaling ratio compared to the size of the sink pad crop rectangle. The location of the compose rectangle specifies the location of the actual sink compose rectangle in the sink compose bounds rectangle.
4. Source pad actual crop selection. Crop on the source pad defines crop performed to the image in the sink compose bounds rectangle.
5. Source pad format. The source pad format defines the output pixel format of the subdev, as well as the other parameters with the exception of the image width and height. Width and height are defined by the size of the source pad actual crop selection.

Accessing any of the above rectangles not supported by the subdev will return `EINVAL`. Any rectangle referring to a previous unsupported rectangle coordinates will instead refer to the previous supported rectangle. For example, if sink crop is not supported, the compose selection will refer to the sink pad format dimensions instead.

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-subdev.rst, line 454)

Unknown directive type "kernel-figure".

```
.. kernel-figure:: subdev-image-processing-crop.svg
    :alt:   subdev-image-processing-crop.svg
    :align: center

    **Figure 4.5. Image processing in subdevs: simple crop example**
```

In the above example, the subdev supports cropping on its sink pad. To configure it, the user sets the media bus format on the subdev's sink pad. Now the actual crop rectangle can be set on the sink pad --- the location and size of this rectangle reflect the location and size of a rectangle to be cropped from the sink format. The size of the sink crop rectangle will also be the size of the format of the subdev's source pad.

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-subdev.rst, line 471)

Unknown directive type "kernel-figure".

```
.. kernel-figure:: subdev-image-processing-scaling-multi-source.svg
    :alt:   subdev-image-processing-scaling-multi-source.svg
    :align: center

    **Figure 4.6. Image processing in subdevs: scaling with multiple sources**
```

In this example, the subdev is capable of first cropping, then scaling and finally cropping for two source pads individually from the resulting scaled image. The location of the scaled image in the cropped image is ignored in sink compose target. Both of the locations of the source crop rectangles refer to the sink scaling rectangle, independently cropping an area at location specified by the source crop rectangle from it.

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-subdev.rst, line 487)

Unknown directive type "kernel-figure".

```
.. kernel-figure:: subdev-image-processing-full.svg
    :alt:    subdev-image-processing-full.svg
    :align:  center

    **Figure 4.7. Image processing in subdevs: scaling and composition with multiple sinks and source
```

The subdev driver supports two sink pads and two source pads. The images from both of the sink pads are individually cropped, then scaled and further composed on the composition bounds rectangle. From that, two independent streams are cropped and sent out of the subdev from the source pads.

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-subdev.rst, line 500)

Unknown directive type "toctree".

```
.. toctree::
    :maxdepth: 1

    subdev-formats
```