Types and flags used to represent the media graph elements

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\mediactl\(linux-master)(Documentation)(userspace-api)(media)(mediactl)media-types.rst`, line 8)

Unknown directive type "tabularcolumns".

```
.. tabularcolumns:: |p{8.2cm}|p{9.3cm}|
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\mediactl\(linux-master)(Documentation)(userspace-api)(media)(mediactl)media-types.rst`, line 48)

Unknown directive type "cssclass".

```
.. cssclass:: longtable
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\mediactl\(linux-master)(Documentation)(userspace-api)(media)(mediactl)media-types.rst`, line 50)

Unknown directive type "flat-table".

```
.. flat-table:: Media entity functions
    :header-rows:  0
    :stub-columns: 0

    *  -  ``MEDIA_ENT_F_UNKNOWN`` and
          ``MEDIA_ENT_F_V4L2_SUBDEV_UNKNOWN``
       -  Unknown entity. That generally indicates that a driver didn't
          initialize properly the entity, which is a Kernel bug

    *  -  ``MEDIA_ENT_F_IO_V4L``
       -  Data streaming input and/or output entity.

    *  -  ``MEDIA_ENT_F_IO_VBI``
       -  V4L VBI streaming input or output entity

    *  -  ``MEDIA_ENT_F_IO_SWRADIO``
       -  V4L Software Digital Radio (SDR) streaming input or output entity

    *  -  ``MEDIA_ENT_F_IO_DTV``
       -  DVB Digital TV streaming input or output entity

    *  -  ``MEDIA_ENT_F_DTV_DEMOD``
       -  Digital TV demodulator entity.

    *  -  ``MEDIA_ENT_F_TS_DEMUX``
       -  MPEG Transport stream demux entity. Could be implemented on
          hardware or in Kernelspace by the Linux DVB subsystem.

    *  -  ``MEDIA_ENT_F_DTV_CA``
       -  Digital TV Conditional Access module (CAM) entity

    *  -  ``MEDIA_ENT_F_DTV_NET_DECAP``
       -  Digital TV network ULE/MLE desencapsulation entity. Could be
          implemented on hardware or in Kernelspace

    *  -  ``MEDIA_ENT_F_CONN_RF``
       -  Connector for a Radio Frequency (RF) signal.

    *  -  ``MEDIA_ENT_F_CONN_SVIDEO``
       -  Connector for a S-Video signal.

    *  -  ``MEDIA_ENT_F_CONN_COMPOSITE``
       -  Connector for a RGB composite signal.

    *  -  ``MEDIA_ENT_F_CAM_SENSOR``
       -  Camera video sensor entity.
    *  -  ``MEDIA_ENT_F_FLASH``
       -  Flash controller entity.

    *  -  ``MEDIA_ENT_F_LENS``
       -  Lens controller entity.

    *  -  ``MEDIA_ENT_F_ATV_DECODER``
       -  Analog video decoder, the basic function of the video decoder is
          to accept analogue video from a wide variety of sources such as
          broadcast, DVD players, cameras and video cassette recorders, in
          either NTSC, PAL, SECAM or HD format, separating the stream into
          its component parts, luminance and chrominance, and output it in
          some digital video standard, with appropriate timing signals.

    *  -  ``MEDIA_ENT_F_TUNER``
       -  Digital TV, analog TV, radio and/or software radio tuner, with
          consists on a PLL tuning stage that converts radio frequency (RF)
          signal into an Intermediate Frequency (IF). Modern tuners have
          internally IF-PLL decoders for audio and video, but older models
          have those stages implemented on separate entities.

    *  -  ``MEDIA_ENT_F_IF_VID_DECODER``
       -  IF-PLL video decoder. It receives the IF from a PLL and decodes
          the analog TV video signal. This is commonly found on some very
          old analog tuners, like Philips MK3 designs. They all contain a
          tda9887 (or some software compatible similar chip, like tda9885).
          Those devices use a different I2C address than the tuner PLL.

    *  -  ``MEDIA_ENT_F_IF_AUD_DECODER``
       -  IF-PLL sound decoder. It receives the IF from a PLL and decodes
          the analog TV audio signal. This is commonly found on some very
          old analog hardware, like Micronas msp3400, Philips tda9840,
          tda985x, etc. Those devices use a different I2C address than the
          tuner PLL and should be controlled together with the IF-PLL video
          decoder.

    *  -  ``MEDIA_ENT_F_AUDIO_CAPTURE``
       -  Audio Capture Function Entity.

    *  -  ``MEDIA_ENT_F_AUDIO_PLAYBACK``
       -  Audio Playback Function Entity.

    *  -  ``MEDIA_ENT_F_AUDIO_MIXER``
       -  Audio Mixer Function Entity.

    *  -  ``MEDIA_ENT_F_PROC_VIDEO_COMPOSER``
       -  Video composer (blender). An entity capable of video
          composing must have at least two sink pads and one source
          pad, and composes input video frames onto output video
          frames. Composition can be performed using alpha blending,
          color keying, raster operations (ROP), stitching or any other
          means.

    *  -  ``MEDIA_ENT_F_PROC_VIDEO_PIXEL_FORMATTER``
       -  Video pixel formatter. An entity capable of pixel formatting
          must have at least one sink pad and one source pad. Read
          pixel formatters read pixels from memory and perform a subset
          of unpacking, cropping, color keying, alpha multiplication
          and pixel encoding conversion. Write pixel formatters perform
          a subset of dithering, pixel encoding conversion and packing
          and write pixels to memory.

    *  -  ``MEDIA_ENT_F_PROC_VIDEO_PIXEL_ENC_CONV``
       -  Video pixel encoding converter. An entity capable of pixel
          encoding conversion must have at least one sink pad and one
          source pad, and convert the encoding of pixels received on
          its sink pad(s) to a different encoding output on its source
          pad(s). Pixel encoding conversion includes but isn't limited
          to RGB to/from HSV, RGB to/from YUV and CFA (Bayer) to RGB
          conversions.

    *  -  ``MEDIA_ENT_F_PROC_VIDEO_LUT``
       -  Video look-up table. An entity capable of video lookup table
          processing must have one sink pad and one source pad. It uses
          the values of the pixels received on its sink pad to look up
          entries in internal tables and output them on its source pad.
          The lookup processing can be performed on all components
          separately or combine them for multi-dimensional table
          lookups.
    *  -  ``MEDIA_ENT_F_PROC_VIDEO_SCALER``
       -  Video scaler. An entity capable of video scaling must have
          at least one sink pad and one source pad, and scale the
          video frame(s) received on its sink pad(s) to a different
          resolution output on its source pad(s). The range of
          supported scaling ratios is entity-specific and can differ
          between the horizontal and vertical directions (in particular
          scaling can be supported in one direction only). Binning and
          sub-sampling (occasionally also referred to as skipping) are
          considered as scaling.

    *  -  ``MEDIA_ENT_F_PROC_VIDEO_STATISTICS``
       -  Video statistics computation (histogram, 3A, etc.). An entity
          capable of statistics computation must have one sink pad and
          one source pad. It computes statistics over the frames
          received on its sink pad and outputs the statistics data on
          its source pad.

    *  -  ``MEDIA_ENT_F_PROC_VIDEO_ENCODER``
       -  Video (MPEG, HEVC, VPx, etc.) encoder. An entity capable of
          compressing video frames. Must have one sink pad and at least
          one source pad.

    *  -  ``MEDIA_ENT_F_PROC_VIDEO_DECODER``
       -  Video (MPEG, HEVC, VPx, etc.) decoder. An entity capable of
          decompressing a compressed video stream into uncompressed video
          frames. Must have one sink pad and at least one source pad.

    *  -  ``MEDIA_ENT_F_PROC_VIDEO_ISP``
       -  An Image Signal Processor (ISP) device. ISPs generally are one of a
          kind devices that have their specific control interfaces using a
          combination of custom V4L2 controls and IOCTLs, and parameters
          supplied in a metadata buffer.

    *  -  ``MEDIA_ENT_F_VID_MUX``
       - Video multiplexer. An entity capable of multiplexing must have at
         least two sink pads and one source pad, and must pass the video
         frame(s) received from the active sink pad to the source pad.

    *  -  ``MEDIA_ENT_F_VID_IF_BRIDGE``
       - Video interface bridge. A video interface bridge entity must have at
         least one sink pad and at least one source pad. It receives video
         frames on its sink pad from an input video bus of one type (HDMI, eDP,
         MIPI CSI-2, etc.), and outputs them on its source pad to an output
         video bus of another type (eDP, MIPI CSI-2, parallel, etc.).

    *  -  ``MEDIA_ENT_F_DV_DECODER``
       -  Digital video decoder. The basic function of the video decoder is
          to accept digital video from a wide variety of sources
          and output it in some digital video standard, with appropriate
          timing signals.

    *  -  ``MEDIA_ENT_F_DV_ENCODER``
       -  Digital video encoder. The basic function of the video encoder is
          to accept digital video from some digital video standard with
          appropriate timing signals (usually a parallel video bus with sync
          signals) and output this to a digital video output connector such
          as HDMI or DisplayPort.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\mediactl\(linux-master)(Documentation)(userspace-api)(media)(mediactl)media-types.rst`, line 236)

Unknown directive type "tabularcolumns".

```
.. tabularcolumns:: |p{5.5cm}|p{12.0cm}|
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\mediactl\(linux-master)(Documentation)(userspace-api)(media)(mediactl)media-types.rst`, line 242)

Unknown directive type "flat-table".

```
.. flat-table:: Media entity flags
    :header-rows:  0
    :stub-columns: 0
    *  -  ``MEDIA_ENT_FL_DEFAULT``
       -  Default entity for its type. Used to discover the default audio,
          VBI and video devices, the default camera sensor, etc.

    *  -  ``MEDIA_ENT_FL_CONNECTOR``
       -  The entity represents a connector.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\mediactl\(linux-master) (Documentation) (userspace-api) (media) (mediactl)media-types.rst`, line 254)

Unknown directive type "tabularcolumns".

```
.. tabularcolumns:: |p{6.5cm}|p{6.0cm}|p{4.8cm}|
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\mediactl\(linux-master) (Documentation) (userspace-api) (media) (mediactl)media-types.rst`, line 277)

Unknown directive type "flat-table".

```
.. flat-table:: Media interface types
    :header-rows:  0
    :stub-columns: 0

    *  -  ``MEDIA_INTF_T_DVB_FE``
       -  Device node interface for the Digital TV frontend
       -  typically, /dev/dvb/adapter?/frontend?

    *  -  ``MEDIA_INTF_T_DVB_DEMUX``
       -  Device node interface for the Digital TV demux
       -  typically, /dev/dvb/adapter?/demux?

    *  -  ``MEDIA_INTF_T_DVB_DVR``
       -  Device node interface for the Digital TV DVR
       -  typically, /dev/dvb/adapter?/dvr?

    *  -  ``MEDIA_INTF_T_DVB_CA``
       -  Device node interface for the Digital TV Conditional Access
       -  typically, /dev/dvb/adapter?/ca?

    *  -  ``MEDIA_INTF_T_DVB_NET``
       -  Device node interface for the Digital TV network control
       -  typically, /dev/dvb/adapter?/net?

    *  -  ``MEDIA_INTF_T_V4L_VIDEO``
       -  Device node interface for video (V4L)
       -  typically, /dev/video?

    *  -  ``MEDIA_INTF_T_V4L_VBI``
       -  Device node interface for VBI (V4L)
       -  typically, /dev/vbi?

    *  -  ``MEDIA_INTF_T_V4L_RADIO``
       -  Device node interface for radio (V4L)
       -  typically, /dev/radio?

    *  -  ``MEDIA_INTF_T_V4L_SUBDEV``
       -  Device node interface for a V4L subdevice
       -  typically, /dev/v4l-subdev?

    *  -  ``MEDIA_INTF_T_V4L_SWRADIO``
       -  Device node interface for Software Defined Radio (V4L)
       -  typically, /dev/swradio?

    *  -  ``MEDIA_INTF_T_V4L_TOUCH``
       -  Device node interface for Touch device (V4L)
       -  typically, /dev/v4l-touch?

    *  -  ``MEDIA_INTF_T_ALSA_PCM_CAPTURE``
       -  Device node interface for ALSA PCM Capture
       -  typically, /dev/snd/pcmC?D?c

    *  -  ``MEDIA_INTF_T_ALSA_PCM_PLAYBACK``
       -  Device node interface for ALSA PCM Playback
       -  typically, /dev/snd/pcmC?D?p
    *  -  ``MEDIA_INTF_T_ALSA_CONTROL``
       -  Device node interface for ALSA Control
       -  typically, /dev/snd/controlC?

    *  -  ``MEDIA_INTF_T_ALSA_COMPRESS``
       -  Device node interface for ALSA Compress
       -  typically, /dev/snd/compr?

    *  -  ``MEDIA_INTF_T_ALSA_RAWMIDI``
       -  Device node interface for ALSA Raw MIDI
       -  typically, /dev/snd/midi?

    *  -  ``MEDIA_INTF_T_ALSA_HWDEP``
       -  Device node interface for ALSA Hardware Dependent
       -  typically, /dev/snd/hwC?D?

    *  -  ``MEDIA_INTF_T_ALSA_SEQUENCER``
       -  Device node interface for ALSA Sequencer
       -  typically, /dev/snd/seq

    *  -  ``MEDIA_INTF_T_ALSA_TIMER``
       -  Device node interface for ALSA Timer
       -  typically, /dev/snd/timer
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\mediactl\(linux-master) (Documentation) (userspace-api) (media) (mediactl)media-types.rst`, line 358)

Unknown directive type "tabularcolumns".

```
.. tabularcolumns:: |p{5.5cm}|p{12.0cm}|
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\mediactl\(linux-master) (Documentation) (userspace-api) (media) (mediactl)media-types.rst`, line 365)

Unknown directive type "flat-table".

```
.. flat-table:: Media pad flags
    :header-rows:  0
    :stub-columns: 0

    *  -  ``MEDIA_PAD_FL_SINK``
       -  Input pad, relative to the entity. Input pads sink data and are
          targets of links.

    *  -  ``MEDIA_PAD_FL_SOURCE``
       -  Output pad, relative to the entity. Output pads source data and
          are origins of links.

    *  -  ``MEDIA_PAD_FL_MUST_CONNECT``
       -  If this flag is set and the pad is linked to any other pad, then
          at least one of those links must be enabled for the entity to be
          able to stream. There could be temporary reasons (e.g. device
          configuration dependent) for the pad to need enabled links even
          when this flag isn't set; the absence of the flag doesn't imply
          there is none.
```

One and only one of `MEDIA_PAD_FL_SINK` and `MEDIA_PAD_FL_SOURCE` must be set for every pad.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\mediactl\(linux-master) (Documentation) (userspace-api) (media) (mediactl)media-types.rst`, line 389)

Unknown directive type "tabularcolumns".

```
.. tabularcolumns:: |p{5.5cm}|p{12.0cm}|
```

**System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-

master\Documentation\userspace-api\media\mediactl\(linux-master)(Documentation)(userspace-api)(media)(mediactl)media-types.rst, line 397)

Unknown directive type "flat-table".

```
.. flat-table:: Media link flags
    :header-rows:  0
    :stub-columns: 0

    *  -  ``MEDIA_LNK_FL_ENABLED``
       -  The link is enabled and can be used to transfer media data. When
          two or more links target a sink pad, only one of them can be
          enabled at a time.

    *  -  ``MEDIA_LNK_FL_IMMUTABLE``
       -  The link enabled state can't be modified at runtime. An immutable
          link is always enabled.

    *  -  ``MEDIA_LNK_FL_DYNAMIC``
       -  The link enabled state can be modified during streaming. This flag
          is set by drivers and is read-only for applications.

    *  -  ``MEDIA_LNK_FL_LINK_TYPE``
       -  This is a bitmask that defines the type of the link. Currently,
          two types of links are supported:

          .. _MEDIA-LNK-FL-DATA-LINK:

          ``MEDIA_LNK_FL_DATA_LINK`` if the link is between two pads

          .. _MEDIA-LNK-FL-INTERFACE-LINK:

          ``MEDIA_LNK_FL_INTERFACE_LINK`` if the link is between an
          interface and an entity
```