GStreamer

Examples of running GStreamer from the PetaLinux command line are provided below. To see the description of gstreamer elements and properties used in each of them, use the gst-inspect-1.0 command.

For example, to get description of each parameters for "omxh264dec" element, enter the following at the command prompt:

```
gst-inspect-1.0 omxh264dec
```

H.264 Decoding

Decode H.264 based input file and display it over monitor connected to Display port

```
gst-launch-1.0 filesrc location="input-file.mp4" ! qtdemux name=demux demux.video_0
! h264parse ! omxh264dec ! queue max-size-bytes=0 ! kmssink
bus-id=fd4a0000.zynqmp-display fullscreen-overlay=1
```

H.265 Decoding

Decode H.265 based input file and display it over monitor connected to Display port

```
gst-launch-1.0 filesrc location="input-file.mp4" ! qtdemux name=demux demux.video_0
! h265parse ! omxh265dec ! queue max-size-bytes=0 ! kmssink
bus-id=fd4a0000.zynqmp-display fullscreen-overlay=1
```

Note: Input-file.mp4 can be of any of the following formats:

- 4:2:0 8-bit
- 4:2:2 8-bit
- 4:2:0 10-bit
- 4:2:2 10-bit

High Bitrate Bitstream Decoding

To reduce frame decoding time for bitstreams greater than 100 Mb/s at 4kP30, use the options below:

- Increase internal decoder buffers (internal-entropy-buffers parameter) to 9 or 10.
- Add a queue at decoder input side

The command below decodes an H.264 MP4 file using an increased number of internal entropy buffers and displays it via DisplayPort.

```
gst-launch-1.0 filesrc location="input-file.mp4" ! qtdemux name=demux demux.video_0
! h264parse ! queue max-size-bytes=0 ! omxh264dec internal-entropy-buffers=10 !
queue max-size-bytes=0 ! kmssink bus-id=fd4a0000.zynqmp-display fullscreen-
overlay=1
```

H.264 Encoding

Encode input 3840×2160 4:2:0 8-bit (NV12) YUV file to H.264.

```
gst-launch-1.0 filesrc location="input-file.yuv" ! rawvideoparse format=nv12
width=3840 height=2160 framerate=30/1 ! omxh264enc ! filesink
location="output.h264"
```

Encoding 3840×2160 4:2:2 8-bit (NV16) YUV file to H.264.

```
gst-launch-1.0 filesrc location="input-file.yuv" ! rawvideoparse format=nv16
width=3840 height=2160 framerate=30/1 ! omxh264enc ! filesink
location="output.h264"
```

Encoding 3840×2160 4:2:0 10-bit (NV12_10LE32) YUV file to H.264.

```
gst-launch-1.0 filesrc location="input-file.yuv" ! rawvideoparse format=nv12-10le32
width=3840 height=2160 framerate=30/1 ! omxh264enc ! filesink
location="output.h264"
```

Encoding 3840×2160 4:2:2 10-bit (NV16_10LE32) YUV file to H.264.

```
gst-launch-1.0 filesrc location="input-file.yuv" ! rawvideoparse format=nv16-10le32
width=3840 height=2160 framerate=30/1 ! omxh264enc ! filesink
location="output.h264"
```

H.265 Encoding

Encoding 3840×2160 4:2:0 8-bit (NV12) YUV file to H.265.

```
gst-launch-1.0 filesrc location="input-file.yuv" ! rawvideoparse format=nv12
width=3840 height=2160 framerate=30/1 ! omxh265enc ! filesink
location="output.h265"
```

Encoding 3840×2160 4:2:2 8-bit (NV16) YUV file to H.265.

```
gst-launch-1.0 filesrc location="input-file.yuv" ! rawvideoparse format=nv16
width=3840 height=2160 framerate=30/1 ! omxh265enc ! filesink
location="output.h265"
```

Encoding 3840×2160 4:2:0 10-bit (NV12_10LE32) YUV file to H.265.

```
gst-launch-1.0 filesrc location="input-file.yuv" ! rawvideoparse format=nv12-10le32
width=3840 height=2160 framerate=30/1 ! omxh265enc ! filesink
location="output.h265"
```

Encoding 3840×2160 4:2:2-10-bit (NV16_10LE32) YUV file to H.265.

```
gst-launch-1.0 filesrc location="input-file.yuv" ! rawvideoparse format=nv16-10le32
width=3840 height=2160 framerate=30/1 ! omxh265enc ! filesink
location="output.h265"
```

Note: The command lines above assume the file input-file.yuv is in the format specified.

Transcode from H.264 to H.265

Convert H.264 based input container format file into H.265 format

```
gst-launch-1.0 filesrc location="input-h264-file.mp4" ! qtdemux name=demux
demux.video_0 ! h264parse ! video/x-h264, alignment=au ! omxh264dec low-latency=0
! omxh265enc ! video/x-h265, alignment=au ! filesink location="output.h265"
```

Transcode from H.265 to H.264

Convert H.265 based input container format file into H.264 format

```
gst-launch-1.0 filesrc location="input-h265-file.mp4" ! qtdemux name=demux
demux.video_0 ! h265parse ! video/x-h265, alignment=au ! omxh265dec low-latency=0
! omxh264enc ! video/x-h264, alignment=au ! filesink location="output.h264
```

Multistream Decoding

Decode H.265 input file using four decoder elements simultaneously, saving them to separate files

```
gst-launch-1.0 filesrc location=input_1920x1080.mp4 ! qtdemux ! h265parse ! tee
name=t
t. ! queue ! omxh265dec ! queue max-size-bytes=0 !
filesink location="output_0_1920x1080.yuv"
t. ! queue ! omxh265dec ! queue max-size-bytes=0 !
filesink location="output_1_1920x1080.yuv"
t. ! queue ! omxh265dec ! queue max-size-bytes=0 !
filesink location="output_2_1920x1080.yuv"
t. ! queue ! omxh265dec ! queue max-size-bytes=0 !
filesink location="output_3_1920x1080.yuv"
```

Note: tee element is used to feed same input file into 4 decoder instances, user can use separate gst-launch-1.0 application to fed different inputs.

Multistream Encoding

Encode input YUV file into eight streams by using eight encoder elements simultaneously.

```
gst-launch-1.0 filesrc location=input_nv12_1920x1080.yuv ! rawvideoparse
width=1920 height=1080 format=nv12 framerate=30/1 ! tee name=t
t. ! queue ! omxh264enc control-rate=2 target-bitrate=8000 ! video/x-
h264, profile=high ! filesink location="ouput_0.h264"
t. ! queue ! omxh264enc control-rate=2 target-bitrate=8000 ! video/x-
h264, profile=high ! filesink location="ouput_1.h264"
t. ! queue ! omxh264enc control-rate=2 target-bitrate=8000 ! video/x-
h264, profile=high ! filesink location="ouput_2.h264"
t. ! queue ! omxh264enc control-rate=2 target-bitrate=8000 ! video/x-
h264, profile=high ! filesink location="ouput_3.h264"
t. ! queue ! omxh264enc control-rate=2 target-bitrate=8000 ! video/x-
h264, profile=high ! filesink location="ouput_4.h264"
t. ! queue ! omxh264enc control-rate=2 target-bitrate=8000 ! video/x-
h264, profile=high ! filesink location="ouput_5.h264"
t. ! queue ! omxh264enc control-rate=2 target-bitrate=8000 ! video/x-
h264, profile=high ! filesink location="ouput_6.h264"
t. ! queue ! omxh264enc control-rate=2 target-bitrate=8000 ! video/x-
h264, profile=high ! filesink location="ouput_7.h264"
```

Note: The tee element is used to feed the same input file into eight encoder instances. You can separate the gst-launch-1.0 application to be fed with different inputs.

For alternate input YUV formats following changes are required in above pipeline:

Format/Profile	Arguments
4:2:2 8-bit (NV16)	format=nv16 profile=high-4:2:2
4:2:0 10-bit (NV12_10LE32)	format=nv12-10le32 profile=high-10
4:2:2 10-bit (NV16_10LE32)	format=nv16-10le32 profile=high-4:2:2
4:0:0 8-bit (GRAY8)	Not Supported
4:0:0 10-bit (GRAY10_LE32)	Not Supported

Transcoding and Streaming via Ethernet

```
gst-launch-1.0 filesrc location="test_0003.mp4" ! qtdemux ! h264parse ! omxh264dec
! omxh265enc control-rate=2 target-bitrate=20000 ! h265parse ! queue ! rtph265pay
! udpsink host=192.168.1.1 port=50000 buffer-size=20000000 async=false max-
lateness=-1 qos-dscp=60
```

Note: 192.168.1.1 is an example client IP address. You may need to modify this with actual client IP address.

Streaming via Ethernet and Decoding to the Display Pipeline

```
gst-launch-1.0 udpsrc port=50000 caps="application/x-rtp, media=video,
clock-rate=90000, payload=96, encoding-name=H265" buffer-size=20000000 !
rtpjitterbuffer latency=1000 ! rtph265depay ! h265parse ! omxh265dec ! queue !
kmssink bus-id=fd4a0000.zynqmp-display fullscreen-overlay=1 sync=false
```

Recommended Settings for Streaming

The following are the recommended settings for streaming:

- Low-latency RC with low-delay-p gop-mode, gdr-mode=horizontal, periodicity-idr=Picture Height in MBs
- Low-latency RC with low-delay-p gop-mode and periodicity-idr=twice the framerate.
- CBR RC with low-delay-p gop-mode, gdr-mode=horizontal, periodicity-idr=Picture Height in MBs
- CBR RC with low-delay-p gop-mode and periodicity-idr=twice the framerate, cpb-size="1000"

For AVC, 1 MB=16x16 Pixels for HEVC, 1MB=32x32 pixels, so user need to calculate picture height in Mbs= roundup(Height,64)/#Mb rows

Note:

- If you are not using buffer-size property of `udpsrc`, then you must set it manually using `sysctl` command as per their network bandwidth utilization and requirements.

```
-> sysctl -w net.core.rmem_default=60000000
```

- VBR is not a preferred mode of streaming.

Verified GStreamer Elements

Table 12-10: **Verified GStreamer Elements**

Element	Description
filesink	Writes incoming data to a file in the local file system
filesrc	Reads data from a file in the local file system
h264parse	Parses a H.264 encoded stream
h265parse	Parses a H.265 encoded stream
kmssink	Renders video frames directly in a plane of a DRM device
omxh264dec	Decodes OpenMAX H.264 video
omxh264enc	Encodes OpenMAX H.264 video
omxh265dec	Decodes OpenMAX H.265 video
omxh265enc	Encodes OpenMAX H.265 video
qtdemux	Demuxes a .mov file into raw or compressed audio and/or video streams.
queue	Queues data until one of the limits specified by the “max-size-buffers”, “max-size-bytes” or “max-size-time” properties has been reached
rtph264depay	Extracts an H.264 video payload from an RTP packet stream
rtph264pay	Encapsulates an H.264 video in an RTP packet stream
rtph265depay	Extracts an H.265 video payload from an RTP packet stream
rtph265pay	Encapsulates an H.265 video in an RTP packet stream
rtpjitterbuffer	Reorders and removes duplicate RTP packets as they are received from a network source
tee	Splits data to multiple pads
udpsink	Sinks UDP packets to the network
udpsrc	Reads UDP packets from the network
v4l2src	Captures video from v4l2 devices, like webcams and television tuner cards
rawvideoparse	Converts a byte stream into video frames