## DOCUMENT CHANGE HISTORY

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Authors</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>v1.0</td>
<td>01 May 2015</td>
<td>mzensius</td>
<td>Initial release.</td>
</tr>
<tr>
<td>v1.1</td>
<td>30 Jun 2015</td>
<td>mzensius</td>
<td>Added rotation and scaling commands, other new content.</td>
</tr>
<tr>
<td>v1.2</td>
<td>03 Nov 2015</td>
<td>emilyh</td>
<td>Changes for 23.1</td>
</tr>
<tr>
<td>v1.3</td>
<td>19 Nov 2015</td>
<td>mzensius</td>
<td>Added note for display export.</td>
</tr>
<tr>
<td>v1.4</td>
<td>17 Dec 2015</td>
<td>hlang</td>
<td>Updated gst-nvivafilter sample pipelines. Updated steps to build gstreamer manually.</td>
</tr>
<tr>
<td>v1.5</td>
<td>08 Jan 2016</td>
<td>kstone</td>
<td>Added nvvidconv interpolation method.</td>
</tr>
<tr>
<td>v1.5</td>
<td>29 Jan 2016</td>
<td>hlang</td>
<td>Additional syntax changes for 23.2 release</td>
</tr>
<tr>
<td>v2.0</td>
<td>11 May 2016</td>
<td>mzensius</td>
<td>Minor change to nvgstcapture options.</td>
</tr>
<tr>
<td>v3.0</td>
<td>11 Aug 2016</td>
<td>mzensius</td>
<td>Versioned for 24.2 release. Gstreamer-0.10 content removed. Also Adds Video Cropping example, interpolation methods for video scaling, EGLStream producer example, and an EGL Image transform example.</td>
</tr>
<tr>
<td>v3.1</td>
<td>06 Oct 2016</td>
<td>mzensius</td>
<td>Minor updates to video encoder features.</td>
</tr>
<tr>
<td>v3.1.1</td>
<td>21 Nov 2016</td>
<td>mzensius</td>
<td>Changed title of document.</td>
</tr>
<tr>
<td>v3.2</td>
<td>12 Jan 2017</td>
<td>mzensius</td>
<td>Adds H.264/H.265 encoder documentation. Also corrects the Gstreamer-1.0 installation procedure.</td>
</tr>
<tr>
<td>3.2</td>
<td>3 Mar 2017</td>
<td>hlang</td>
<td>Update date/moniker for L4T 27.1 release. No other updates.</td>
</tr>
<tr>
<td>3.3</td>
<td>20 Jul 2017</td>
<td>mzensius</td>
<td>Minor edit to command syntax, and update of date/moniker for L4T 28.1 release.</td>
</tr>
</tbody>
</table>
# Table of Contents

**Accelerated GStreamer User Guide** ................................................................. 1  
Gstreamer-1.0 Installation and Setup .......................................................... 2  
Decode Examples .................................................................................. 3  
  - Audio Decode Examples Using gst-launch-1.0 ........................................ 4  
  - Video Decode Examples Using gst-launch-1.0 ........................................ 4  
Encode Examples ............................................................................. 5  
  - Audio Encode Examples Using gst-launch-1.0 ........................................ 5  
  - Video Encode Examples Using gst-launch-1.0 ........................................ 6  
Supported H.264/H.265 Encoder Features with Gstreamer-1.0 ....................... 7  
Camera Capture with Gstreamer-1.0 ....................................................... 10  
Video Playback with Gstreamer-1.0 .......................................................... 11  
Video Format Conversion with Gstreamer-1.0 .......................................... 11  
  - raw-yuv Input Formats ........................................................................ 11  
  - raw-gray Input Formats ...................................................................... 12  
  - raw-yuv Output Formats .................................................................... 12  
  - raw-gray Output Formats .................................................................. 12  
Video Scaling with Gstreamer-1.0 .............................................................. 12  
  - raw-yuv Input Formats ........................................................................ 13  
  - raw-gray Input Formats ...................................................................... 13  
  - raw-yuv Output Formats .................................................................... 13  
  - raw-gray Output Formats .................................................................. 13  
NVIDIA Input and Output Formats .............................................................. 14  
Video Cropping with Gstreamer-1.0 ............................................................. 14  
Video Transcode with Gstreamer-1.0 ......................................................... 15  
CUDA Video Post-Processing with GStreamer-1.0 ....................................... 17  
  - gst-videocuda .................................................................................... 17  
  - gst-nvivafilter .................................................................................. 17  
Video Rotation with Gstreamer-1.0 ............................................................ 18  
Interpolation Methods for Video Scaling .................................................... 19  
EGLStream Producer Example ................................................................... 20  
EGL Image Transform Example .................................................................. 20  

**GStreamer Build Instructions** .................................................................. 21  

**Nvgstcapture-1.0 Option Reference** ......................................................... 24  
  - Nvgstcapture Application Options ....................................................... 24  
  - CSI Camera Supported Resolutions ..................................................... 26  
  - CSI Camera Runtime Commands ......................................................... 27
USB Camera Runtime Commands ............................................................ 29
Notes............................................................................................. 30

Video Encoder Features .................................................................. 32

Supported Cameras............................................................................. 33
  CSI Cameras................................................................. 33
  USB 2.0 Cameras ............................................................... 33
  Industrial Camera Details .......................................................... 34
This document is a user guide for the Gstreamer version 1.0 based accelerated solution included in NVIDIA® Tegra® Linux Driver Package for Ubuntu Linux 16.04 on platforms including Tegra X1 devices.

This document contains the following sections:

- Gstreamer-1.0 Installation and Setup
- Decode Examples
- Encode Examples
- Camera Capture with Gstreamer-1.0
- Video Playback with Gstreamer-1.0
- Video Format Conversion with Gstreamer-1.0
- Video Scaling with Gstreamer-1.0
- Video Cropping with Gstreamer-1.0
- Video Transcode with Gstreamer-1.0
- CUDA Video Post-Processing with Gstreamer-1.0
- Video Rotation with Gstreamer-1.0
- Interpolation Methods for Video Scaling
- EGLStream Producer Example
- EGL Image Transform Example
- Gstreamer Build Instructions
- Nvgstcapture-1.0 Option Reference
- Video Encoder Features
- Supported USB Camera
GSTREAMER-1.0 INSTALLATION AND SETUP

This section describes how to install and configure GStreamer.

To install Gstreamer-1.0

- Install Gstreamer-1.0 on the platform with the following commands:

  sudo add-apt-repository universe
  sudo add-apt-repository multiverse
  sudo apt-get update
  sudo apt-get install gstreamer1.0-tools gstreamer1.0-alsa gstreamer1.0-plugins-base gstreamer1.0-plugins-good gstreamer1.0-plugins-bad gstreamer1.0-plugins-ugly gstreamer1.0-libav
  sudo apt-get install libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev libgstreamer-plugins-good1.0-dev libgstreamer-plugins-bad1.0-dev

To check the Gstreamer-1.0 version

- Check the Gstreamer-1.0 version with the following command:

  gst-inspect-1.0 --version

Gstreamer version 1.0 includes the following gst-omx video decoders:

<table>
<thead>
<tr>
<th>Video Decoder</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>omxh265dec</td>
<td>OpenMAX IL H.265 Video Decoder</td>
</tr>
<tr>
<td>omxh264dec</td>
<td>OpenMAX IL H.264 Video Decoder</td>
</tr>
<tr>
<td>omxmpeg4videodec</td>
<td>OpenMAX IL MPEG4 Video Decoder</td>
</tr>
<tr>
<td>omxvp8dec</td>
<td>OpenMAX IL VP8 Video Decoder</td>
</tr>
<tr>
<td>omxvp9dec</td>
<td>OpenMAX IL VP9 video decoder</td>
</tr>
</tbody>
</table>

Gstreamer version 1.0 includes the following gst-omx video encoders:

<table>
<thead>
<tr>
<th>Video Encoders</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>omxh264enc</td>
<td>OpenMAX IL H.264/AVC video encoder</td>
</tr>
<tr>
<td>omxh265enc</td>
<td>OpenMAX IL H.265/AVC video encoder</td>
</tr>
<tr>
<td>omxvp8enc</td>
<td>OpenMAX IL VP8 video encoder</td>
</tr>
</tbody>
</table>

Gstreamer version 1.0 includes the following gst-omx video sinks:

<table>
<thead>
<tr>
<th>Video Sink</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nvoverlaysink</td>
<td>OpenMAX IL videosink element</td>
</tr>
<tr>
<td>nvhdmioverlay (deprecated)</td>
<td>OpenMAX IL HDMI videosink element</td>
</tr>
</tbody>
</table>
Gstreamer version 1.0 includes the following egl image video sinks:

<table>
<thead>
<tr>
<th>Video Sink</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nveglglessink</td>
<td>EGL/GLES videosink element</td>
</tr>
</tbody>
</table>

Gstreamer version 1.0 includes the following proprietary NVIDIA plugins:

<table>
<thead>
<tr>
<th>Video Sink</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nvvidconv</td>
<td>Video format conversion &amp; scaling</td>
</tr>
<tr>
<td>nveglstreamsrc</td>
<td>Acts as Gstreamer Source Component, accepts EGLStream from EGLStream producer</td>
</tr>
<tr>
<td>nvvideosink</td>
<td>Video Sink Component. Accepts YUV-I420 format and produces EGLStream (RGBA)</td>
</tr>
<tr>
<td>nvegltransform</td>
<td>Video transform element for NVMM to EGLImage (supported with nveglglessink only)</td>
</tr>
</tbody>
</table>

Gstreamer version 1.0 includes the following libjpeg based JPEG image video encode/decode plugins:

<table>
<thead>
<tr>
<th>Video Sink</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nvjpegenc</td>
<td>JPEG encoder element</td>
</tr>
<tr>
<td>nvjpegdec</td>
<td>JPEG decoder element</td>
</tr>
</tbody>
</table>

Note: Execute the following command on the target before starting the video decode pipeline using gst-launch or nvgstplayer.

export DISPLAY=:0
Start the X server with xinit &, if it is not already running.

DECODE EXAMPLES

The examples in this section show how you can perform audio and video decode with Gstreamer.

Note: Gstreamer version 0.10 support is deprecated in Linux for Tegra (L4T) Release 24.2. Use of Gstreamer version 1.0 is recommended for development.
Audio Decode Examples Using gst-launch-1.0

The following examples show how you can perform audio decode using Gstreamer-1.0.

**AAC Decode (OSS software decode)**

```bash
gst-launch-1.0 filesrc location=<filename.mp4> ! qtdemux name=demux
demux.audio_0 ! queue ! avdec_aac ! audioconvert ! alsasink -e
```

**AMR-WB Decode (OSS software decode)**

```bash
gst-launch-1.0 filesrc location=<filename.mp4> ! qtdemux name=demux
demux.audio_0 ! queue ! avdec_amrwb ! audioconvert ! alsasink -e
```

**AMR-NB Decode (OSS software decode)**

```bash
gst-launch-1.0 filesrc location=<filename.mp4> ! qtdemux name=demux
demux.audio_0 ! queue ! avdec_amrn nb ! audioconvert ! alsasink -e
```

**MP3 Decode (OSS software decode)**

```bash
gst-launch-1.0 filesrc location=<filename.mp3> ! mpegaudioparse !
avdec_mp3 ! audioconvert ! alsasink -e
```

- **Note:** To route audio over HDMI, set the alsasink property device to the following:
  ```bash
  hw:Tegra,3
  ```

Video Decode Examples Using gst-launch-1.0

The following examples show how you can perform video decode on Gstreamer-1.0.

**H.264 Decode (NVIDIA accelerated decode)**

```bash
gst-launch-1.0 filesrc location=<filename.mp4> ! qtdemux name=demux
demux.video_0 ! queue ! h264parse ! omxh264dec ! nveglglessink -e
```

**H.265 Decode (NVIDIA accelerated decode)**

```bash
gst-launch-1.0 filesrc location=<filename.mp4> ! qtdemux name=demux
demux.video_0 ! queue ! h265parse ! omxh265dec ! nvoverlaysink -e
```

- **Note:** Decoding H.265 streams requires Gstreamer version 1.4.x or later, including support for h265parse and qtdemux. See Gstreamer Build Instructions in this guide for details.
VP8 Decode (NVIDIA accelerated decode)

```bash
gst-launch-1.0 filesrc location=<filename.mp4> ! qtdemux name=demux
demux.video_0 ! queue ! omxvp8dec ! nvoverlaysink -e
```

Note: When you do not use the primary display to render video, use the `display-id` property of `nvoverlaysink`. For example, refer to the pipeline below.

```bash
gst-launch-1.0 filesrc location=<filename.mp4> ! qtdemux name=demux
demux.video_0 ! queue ! omxvp8dec ! nvoverlaysink display-id=1 -e
```

MPEG-4 Decode (NVIDIA accelerated decode)

```bash
gst-launch-1.0 filesrc location=<filename.mp4> ! qtdemux name=demux
demux.video_0 ! queue ! mpeg4videoparse ! omxmpeg4videodec ! nveglglessink -e
```

Image Decode

```bash
gst-launch-1.0 filesrc location=<filename.jpg> ! nvjpegdec !
imagefreeze ! xvimagesink -e
```

ENCODE EXAMPLES

The examples in this section show how you can perform audio and video encode with Gstreamer.

Audio Encode Examples Using `gst-launch-1.0`

The following examples show how you can perform audio encode on Gstreamer-1.0.

**AAC Encode (OSS software encode)**

```bash
gst-launch-1.0 audiotestsrc ! 'audio/x-raw, format=(string)S16LE,
layout=(string)interleaved, rate=(int)44100, channels=(int)2' !
voaacenc ! qtmux ! filesink location=test.mp4 -e
```

**AMR-WB Encode (OSS software encode)**

```bash
gst-launch-1.0 audiotestsrc ! 'audio/x-raw, format=(string)S16LE,
layout=(string)interleaved, rate=(int)16000, channels=(int)1' !
voamrwbenc ! qtmux ! filesink location=test.mp4 -e
```
Video Encode Examples Using gst-launch-1.0

The following examples show how you can perform video encode with Gstreamer-1.0.

**H.264 Encode (NVIDIA accelerated encode)**

```sh
gst-launch-1.0 videotestsrc ! 'video/x-raw, format=(string)I420, width=(int)640, height=(int)480' ! omxh264enc ! 'video/x-h264, stream-format=(string)byte-stream' ! h264parse ! qtmux ! filesink location=test.mp4 -e
```

**H.265 Encode (NVIDIA accelerated encode)**

```sh
gst-launch-1.0 videotestsrc ! 'video/x-raw, format=(string)I420, width=(int)640, height=(int)480' ! omxh265enc ! filesink location=test.h265 -e
```

**VP8 Encode (NVIDIA accelerated encode)**

```sh
gst-launch-1.0 videotestsrc ! 'video/x-raw, format=(string)I420, width=(int)640, height=(int)480' ! omxvp8enc ! qtmux ! filesink location=test.mp4 -e
```

**MPEG-4 Encode (OSS software encode)**

```sh
gst-launch-1.0 videotestsrc ! 'video/x-raw, format=(string)I420, width=(int)640, height=(int)480' ! avenc_mpeg4 ! qtmux ! filesink location=test.mp4 -e
```

**H.263 Encode (OSS software encode)**

```sh
gst-launch-1.0 videotestsrc ! 'video/x-raw, format=(string)I420, width=(int)704, height=(int)576' ! avenc_h263 ! qtmux ! filesink location=test.mp4 -e
```

**Image Encode**

```sh
gst-launch-1.0 videotestsrc num-buffers=1 ! 'video/x-raw, width=(int)640, height=(int)480, format=(string)I420' ! nvjpegenc ! filesink location=test.jpg -e
```
Supported H.264/H.265 Encoder Features with Gstreamer-1.0

This section describes example gst-launch-1.0 usage for features supported by the NVIDIA accelerated H.264/H.265 encoder.

**Note:** Display detailed information on omxh264enc or omxh265enc encoder properties with the `gst-inspect-1.0 [omxh264enc | omxh265enc]` command.

**Set I-frame interval**

```
gst-launch-1.0 videotestsrc num-buffers=200 ! 'video/x-raw, width=(int)1280, height=(int)720, format=(string)I420' ! omxh264enc iframeinterval=100 ! qtmux ! filesink location=test.mp4 -e
```

**Set temporal-tradeoff (the rate the encoder should drop frames)**

```
gst-launch-1.0 videotestsrc num-buffers=200 ! 'video/x-raw, width=(int)1280, height=(int)720, format=(string)I420' ! omxh264enc temporal-tradeoff=1 ! qtmux ! filesink location=test.mp4 -e
```

Configuring temporal tradeoff causes the encoder to intentionally, periodically, drop input frames. The following modes are supported:

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Disable</td>
</tr>
<tr>
<td>1</td>
<td>Drop 1 in 5 frames</td>
</tr>
<tr>
<td>2</td>
<td>Drop 1 in 3 frames</td>
</tr>
<tr>
<td>3</td>
<td>Drop 1 in 2 frames</td>
</tr>
<tr>
<td>4</td>
<td>Drop 2 in 3 frames</td>
</tr>
</tbody>
</table>

**Set rate control mode**

```
gst-launch-1.0 videotestsrc num-buffers=200 ! 'video/x-raw, width=(int)1280, height=(int)720, format=(string)I420' ! omxh264enc control-rate=1 ! qtmux ! filesink location=test.mp4 -e
```

The following modes are supported:

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Disable</td>
</tr>
<tr>
<td>1</td>
<td>Variable bit rate</td>
</tr>
<tr>
<td>2</td>
<td>Constant bit rate</td>
</tr>
</tbody>
</table>
3 Variable bit rate with frame skip. The encoder skips frames as necessary to meet the target bit rate.

4 Constant bit rate with frame skip

Set quantization range for I, P and B frame

The format for the range is the following:

“<I_range>:<P_range>:<B_range>”

Where <I_range>, <P_range> and <B_range> are each expressed as hyphenated values, as shown in the following example:

```sh
gst-launch-1.0 videotestsrc num-buffers=200 ! 'video/x-raw, width=(int)1280, height=(int)720, format=(string)I420' ! omxh264enc qp-range="10,30:10,35:10,35" ! qtmux ! filesink location=test.mp4 -e
```

The range of B frames does not take effect if the number of B frames is 0.

Set hardware preset level

```sh
gst-launch-1.0 videotestsrc num-buffers=200 ! 'video/x-raw, width=(int)1280, height=(int)720, format=(string)I420' ! omxh264enc preset-level=0 ! qtmux ! filesink location=test.mp4 -e
```

The following modes are supported:

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>UltraFastPreset</td>
</tr>
<tr>
<td>1</td>
<td>FastPreset</td>
</tr>
<tr>
<td></td>
<td>Only Integer Pixel (integer-pel) block motion is estimated. For I/P macroblock mode decision, only Intra 16 x 16 cost is compared with Inter modes costs. Supports Intra 16 x 16 and Intra 4 x 4 modes.</td>
</tr>
<tr>
<td>2</td>
<td>MediumPreset</td>
</tr>
<tr>
<td></td>
<td>Supports up to Half Pixel (half-pel) block motion estimation. For an I/P macroblock mode decision, only Intra 16 x 16 cost is compared with Inter modes costs. Supports Intra 16 x 16 and Intra 4 x 4 modes.</td>
</tr>
<tr>
<td>3</td>
<td>SlowPreset</td>
</tr>
<tr>
<td></td>
<td>Supports up to Quarter Pixel (Qpel) block motion estimation. For an I/P macroblock mode decision, Intra 4 x 4 as well as Intra 16 x 16 cost is compared with Inter modes costs. Supports Intra 16 x 16 and Intra 4 x 4 modes.</td>
</tr>
</tbody>
</table>
Set profile

gst-launch-1.0 videotestsrc num-buffers=200 ! 'video/x-raw,
width=(int)1280, height=(int)720, format=(string)I420' ! omxh264enc
profile=8 ! qtmux ! filesink location=test.mp4 -e

From omxh264enc, the following profiles are supported:

<table>
<thead>
<tr>
<th>Profile</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Baseline profile</td>
</tr>
<tr>
<td>2</td>
<td>Main profile</td>
</tr>
<tr>
<td>8</td>
<td>High profile</td>
</tr>
</tbody>
</table>

Set number of B frames between two reference frames

gst-launch-1.0 videotestsrc num-buffers=200 ! 'video/x-raw,
width=(int)1280, height=(int)720, format=(string)I420' ! omxh264enc
num-B-Frames=2 ! qtmux ! filesink location=test.mp4 –e

Note: B-frame-encoding is not supported with omxh265enc.

Insert SPS and PPS at IDR

gst-launch-1.0 videotestsrc num-buffers=200 ! 'video/x-raw,
width=(int)1280, height=(int)720, format=(string)I420' ! omxh264enc
insert-sps-pps=1 ! qtmux ! filesink location=test.mp4 –e

If enabled, a sequence parameter set (SPS) and a picture parameter set (PPS) are inserted before each IDR frame in the H.264/H.265 stream.

Enable two-pass CBR

gst-launch-1.0 videotestsrc num-buffers=200 ! 'video/x-raw,
width=(int)1280, height=(int)720, format=(string)I420' ! omxh264enc
EnableTwopassCBR=1 control-rate=2 ! qtmux ! filesink location=test.mp4 –e

Two-pass CBR must be enabled along with constant bit rate (control-rate=2).

Set virtual buffer size

gst-launch-1.0 videotestsrc num-buffers=200 ! 'video/x-raw,
width=(int)1280, height=(int)720, format=(string)I420' ! omxh264enc
vbv-size=10 ! qtmux ! filesink location=test.mp4 –e
If the buffer size of decoder or network bandwidth is limited, configuring virtual buffer size can cause video stream generation to correspond to the limitations according to the following formula:

\[ \text{virtual buffer size} = \text{vbv-size} \times \frac{\text{bitrate}}{\text{fps}} \]

**Slice-header-spacing with spacing in terms of MB**

```
gst-launch-1.0 videotestsrc num-buffers=200 ! 'video/x-raw, width=(int)1280, height=(int)720, format=(string)I420' ! omxh264enc slice-header-spacing=200 bit-packetization=0 ! qtmux ! filesink location=test.mp4 -e
```

The parameter `bit-packetization=0` configures the network abstraction layer (NAL) packet as macroblock (MB)-based, and `slice-header-spacing=200` configures each NAL packet as 200 MB at maximum.

**Slice header spacing with spacing in terms of number of bits**

```
gst-launch-1.0 videotestsrc num-buffers=200 ! 'video/x-raw, width=(int)1280, height=(int)720, format=(string)I420' ! omxh264enc slice-header-spacing=1024 bit-packetization=1 ! qtmux ! filesink location=test1.mp4 -e
```

The parameter `bit-packetization=1` configures the network abstraction layer (NAL) packet as size-based, and `slice-header-spacing=1024` configures each NAL packet as 1024 bytes at maximum.

## CAMERA CAPTURE WITH GSTREAMER-1.0

For `nvgstcapture-1.0` usage information enter the following command:

```
nvgstcapture-1.0 --help
```

The `nvgstcapture-1.0` application uses the `v4l2src` plugin to capture still images and video.

The following table shows USB camera support.

<table>
<thead>
<tr>
<th>USB Camera Support</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>YUV</td>
<td>Preview display</td>
</tr>
<tr>
<td></td>
<td>Image capture (VGA, 640 x 480)</td>
</tr>
<tr>
<td></td>
<td>Video capture (480p, 720p, H.264/VP8 encode)</td>
</tr>
</tbody>
</table>
raw-yuv Capture (I420 format) and preview display with xvimagesink

```
gst-launch-1.0 v4l2src device="/dev/video0" ! "video/x-raw, width=640, height=480, format=(string)I420" ! xvimagesink -e
```

**VIDEO PLAYBACK WITH GSTREAMER-1.0**

For nvgstplayer-1.0 usage information enter the following command:

```
nvgstplayer-1.0 --help
```

Video can be output to HD displays using the HDMI connector on the platform. The Gstreamer-1.0 application supports currently the following video sinks:

**Overlay Sink (Video playback on overlay in full-screen mode)**

```
gst-launch-1.0 filesrc location=<filename.mp4> ! qtdemux name=demux ! h264parse ! omxh264dec ! nvoverlaysink –e
```

**nveglglessink (Windowed video playback, NVIDIA EGL/GLES videosink)**

```
gst-launch-1.0 filesrc location=<filename.mp4> ! qtdemux name=demux ! h264parse ! omxh264dec ! nveglglessink –e
```

This nvgstplayer-1.0 application supports specific window position and dimensions for windowed playback:

```
nvgstplayer-1.0 –i <filename> --window-x=300 --window-y=300 --window-width=500 --window-height=500
```

**VIDEO FORMAT CONVERSION WITH GSTREAMER-1.0**

The NVIDIA proprietary nvvidconv Gstreamer-1.0 plug-in allows conversion between OSS (raw) video formats and NVIDIA video formats. The nvvidconv plug-in currently supports the format conversions described in this section

**raw-yuv Input Formats**

Currently nvvidconv supports the I420, UYVY, and NV12 raw-yuv input formats.

```
gst-launch-1.0 videotestsrc ! 'video/x-raw, format=(string)UYVY, width=(int)1280, height=(int)720' ! nvvidconv !
```
'video/x-raw(memory:NVMM), format=(string)I420' ! omxh264enc ! 'video/x-h264, stream-format=(string)byte-stream' ! h264parse ! qtmux ! filesink location=test.mp4 -e

**raw-gray Input Formats**

Currently **nvvidconv** supports the GRAY8 raw-gray input format.

```
gst-launch-1.0 videotestsrc ! 'video/x-raw, format=(string)GRAY8, width=(int)1280, height=(int)720' ! nvvidconv ! 'video/x-raw(memory:NVMM), format=(string)I420' ! omxh264enc ! 'video/x-h264, stream-format=(string)byte-stream' ! h264parse ! qtmux ! filesink location=test.mp4 -e
```

**raw-yuv Output Formats**

Currently **nvvidconv** supports the I420 and UYVY the raw-yuv output formats.

```
gst-launch-1.0 filesrc location=640x480_30p.mp4 ! qtdemux ! queue ! h264parse ! omxh264dec ! nvvidconv ! 'video/x-raw, format=(string)UYVY' ! xvimagesink -e
```

**raw-gray Output Formats**

Currently **nvvidconv** supports the GRAY8 raw-gray output format.

```
gst-launch-1.0 filesrc location=640x480_30p.mp4 ! qtdemux ! queue ! h264parse ! omxh264dec ! nvvidconv ! 'video/x-raw, format=(string)GRAY8' ! videoconvert ! xvimagesink -e
```

**VIDEO SCALING WITH GSTREAMER-1.0**

The NVIDIA proprietary **nvvidconv** Gstreamer-1.0 plug-in also allows you to perform video scaling. The **nvvidconv** plug-in currently supports scaling with the format conversions described in this section.
raw-yuv Input Formats

Currently `nvvidconv` supports the I420, UYVY, and NV12 raw-yuv input formats for scaling.

```
gst-launch-1.0 videotestsrc ! 'video/x-raw, format=(string)I420, width=(int)1280, height=(int)720' ! nvvidconv ! 'video/x-raw(memory:NVMM), width=(int)640, height=(int)480, format=(string)I420' ! omxh264enc ! 'video/x-h264, stream-format=(string)byte-stream' ! h264parse ! qtmux ! filesink location=test.mp4 –e
```

raw-gray Input Formats

Currently `nvvidconv` supports the GRAY8 raw-gray input format for scaling.

```
gst-launch-1.0 videotestsrc ! 'video/x-raw, format=(string)GRAY8, width=(int)1280, height=(int)720' ! nvvidconv ! 'video/x-raw(memory:NVMM), width=(int)640, height=(int)480, format=(string)I420' ! omxh264enc ! 'video/x-h264, stream-format=(string)byte-stream' ! h264parse ! qtmux ! filesink location=test.mp4 –e
```

raw-yuv Output Formats

Currently `nvvidconv` supports the I420 and UYVY raw-yuv output formats for scaling.

```
gst-launch-1.0 filesrc location=1280x720_30p.mp4 ! qtdemux ! queue ! h264parse ! omxh264dec ! nvvidconv ! 'video/x-raw, format=(string)I420, width=640, height=480' ! xvimagesink –e
```

raw-gray Output Formats

Currently `nvvidconv` supports the GRAY8 raw-gray output format for scaling.

```
gst-launch-1.0 filesrc location=1280x720_30p.mp4 ! qtdemux ! queue ! h264parse ! omxh264dec ! nvvidconv ! 'video/x-raw, format=(string)GRAY8, width=640, height=480' ! videoconvert ! xvimagesink –e
```
NVIDIA Input and Output Formats

Currently `nvvidconv` supports the NVIDIA input and output formats for scaling described in the following table:

<table>
<thead>
<tr>
<th>Input Format</th>
<th>Output Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>NV12</td>
<td>NV12</td>
</tr>
<tr>
<td>I420</td>
<td>I420</td>
</tr>
<tr>
<td>RGBA</td>
<td></td>
</tr>
</tbody>
</table>

To scale between NVIDIA formats

Scale between NVIDIA Formats with the following commands:

```
gst-launch-1.0 filesrc location=1280x720_30p.mp4 ! qtdemux ! h264parse ! omxh264dec ! nvvidconv ! 'video/x-raw(memory:NVMM), width=(int)640, height=(int)480, format=(string)I420' ! omxh264enc ! qtmux ! filesink location=test.mp4 –e
```

```
gst-launch-1.0 filesrc location=1280x720_30p.mp4 ! qtdemux ! h264parse ! omxh264dec ! nvvidconv ! 'video/x-raw(memory:NVMM), width=(int)640, height=(int)480, format=(string)RGBA' ! nvoverlaysink –e
```

```
gst-launch-1.0 nvcamerasrc fpsRange="30 30" ! 'video/x-raw(memory:NVMM), width=(int)1920, height=(int)1080, format=(string)I420, framerate=(fraction)30/1' ! nvtee ! nvvidconv ! 'video/x-raw(memory:NVMM), width=(int)640, height=(int)480, format=(string)NV12' ! omxh264enc ! qtmux ! filesink location=test.mp4 –e
```

VIDEO CROPPING WITH GSTREAMER-1.0

The NVIDIA proprietary `nvvidconv` Gstreamer-1.0 plug-in also allows you to perform video cropping.

To crop video

Crop video with the following commands:

```
gst-launch-1.0 filesrc location=<filename_1080p.mp4> ! qtdemux ! h264parse ! omxh264dec ! nvvidconv left=400 right=1520 top=200 bottom=880 ! nvoverlaysink display-id=1 –e
```
VIDEO TRANSCODE WITH GSTREAMER-1.0

You can perform video transcoding between the following video formats.

H.264 Decode to VP8 Encode (NVIDIA-accelerated decode to NVIDIA-accelerated encode)

```bash
gst-launch-1.0 filesrc location=<filename.mp4> ! qtdemux name=demux
demux.video_0 ! queue ! h264parse ! omxh264dec ! nvvidconv ! 'video/x-
raw(memory:NVMM), format=(string)I420' ! omxvp8enc ! qtmux name=mux !
filesink location=<Transcoded_filename.mp4> demux.audio_0 ! queue !
aacparse ! mux.audio_0 -e
```

VP8 Decode to H.264 Encode (NVIDIA-accelerated decode to NVIDIA-accelerated encode)

```bash
gst-launch-1.0 filesrc location=<filename.mp4> ! qtdemux
name=demux demux.video_0 ! queue ! omxvp8dec ! nvvidconv ! 'video/x-
raw(memory:NVMM), format=(string)I420' ! omxh264enc ! qtmux name=mux !
filesink location=<Transcoded_filename.mp4> demux.audio_0 ! queue !
aacparse ! mux.audio_0 -e
```

MPEG-4 Decode to VP8 Encode (NVIDIA-accelerated decode to NVIDIA-accelerated encode)

```bash
gst-launch-1.0 filesrc location=<filename.mp4> ! qtdemux
name=demux demux.video_0 ! queue ! mpeg4videoparse ! omxmpeg4videodec !
nvvidconv ! 'video/x-raw(memory:NVMM), format=(string)I420' !
omxvp8enc ! qtmux name=mux ! filesink
location=<Transcoded_filename.mp4> demux.audio_0 ! queue ! aacparse !
mux.audio_0 -e
```

MPEG-4 Decode to H.264 Encode (NVIDIA-accelerated decode to NVIDIA-accelerated encode)

```bash
gst-launch-1.0 filesrc location=<filename.mp4> ! qtdemux
name=demux demux.video_0 ! queue ! mpeg4videoparse ! omxmpeg4videodec !
nvvidconv ! 'video/x-raw(memory:NVMM), format=(string)I420' !
omxh264enc ! qtmux name=mux ! filesink
location=<Transcoded_filename.mp4> demux.audio_0 ! queue ! aacparse !
mux.audio_0 -e
```

H.264 Decode to MPEG-4 Encode (NVIDIA-accelerated decode to OSS software encode)

```bash
gst-launch-1.0 filesrc location=<filename.mp4> ! qtdemux
name=demux demux.video_0 ! queue ! h264parse ! omxh264dec ! nvvidconv !
```
VP8 Decode to MPEG-4 Encode (NVIDIA-accelerated decode to OSS software encode)

```gstreamer
avenc_mpeg4 ! qtmux name=mux ! filesink
location=<Transcoded_filename.mp4> demux.audio_0 ! queue ! aacparse !
mux.audio_0 -e
```

H.264 Decode to Theora Encode (NVIDIA-accelerated decode to OSS software encode)

```gstreamer
gst-launch-1.0 filesrc location=<filename.mp4> ! qtdemux
name=demux demux.video_0 ! queue ! omxvp8dec ! nvvidconv !
avenc_mpeg4 ! qtmux name=mux ! filesink
location=<Transcoded_filename.mp4> demux.audio_0 ! queue ! aacparse !
mux.audio_0 -e
```

VP8 Decode to Theora Encode (NVIDIA-accelerated decode to OSS software encode)

```gstreamer
gst-launch-1.0 filesrc location=<filename.mp4> ! qtdemux name=demux
demux.video_0 ! queue ! h264parse ! omxh264dec ! nvvidconv ! theoraenc
! oggmux name=mux ! filesink location=<Transcoded_filename.ogg> -e
```

MPEG-4 Decode to Theora Encode (NVIDIA-accelerated decode to OSS software encode)

```gstreamer
gst-launch-1.0 filesrc location=<filename.mp4> ! qtdemux name=demux
demux.video_0 ! queue ! mpeg4videoparse ! omxmpeg4videodec !
nvvidconv ! theoraenc ! oggmux name=mux ! filesink
location=<Transcoded_filename.ogg> -e
```
CUDA VIDEO POST-PROCESSING WITH GSTREAMER-1.0

This section describes Gstreamer-1.0 plug-ins for CUDA post-processing operations.

gst-videocuda

This Gstreamer-1.0 plug-in performs CUDA post-processing operations on decoder-provided EGL images and render video using nveglglessink.

The following are sample pipeline creation and application usage commands.

Sample decode pipeline

```
gst-launch-1.0 filesrc location=<filename_h264_1080p.mp4> ! qtdemux name=demux ! h264parse ! omxh264dec ! videocuda ! nveglglessink max-lateness=-1 -e
```

Sample decode command

```
nvgstplayer-1.0 -i <filename_h264_1080p.mp4> --svd="omxh264dec" --svc="videocuda" --svs="nveglglessink # max-lateness=-1" --disable-vnative --no-audio --window-x=0 --window-y=0 --window-width=960 --window-height=540
```

gst-nvivafilter

This NVIDIA proprietary Gstreamer-1.0 plug-in performs pre/post and CUDA post-processing operations on CSI camera captured or decoded frames, and renders video using overlay video sink or video encode.

Sample decode pipeline

```
gst-launch-1.0 filesrc location=<filename.mp4> ! qtdemux ! h264parse ! omxh264dec ! nvivafilter cuda-process=true customer-lib-name="libnvsample_cudaprocess.so" ! 'video/x-raw(memory:NVMM), format=(string)NV12' ! nvoverlaysink -e
```

Sample CSI Camera pipeline

```
gst-launch-1.0 nvccamerasrc fpsRange="30 30" ! 'video/x-raw(memory:NVMM), width=(int)3840, height=(int)2160, format=(string)I420, framerate=(fraction)30/1' ! nvtee ! nvivafilter cuda-process=true customer-lib-name="libnvsample_cudaprocess.so" ! 'video/x-raw(memory:NVMM), format=(string)NV12' ! nvoverlaysink -e
```
Note: See nvsample_cudaprocess_src.tbz2 package for the libnvsample_cudaprocess.so library sources. A Sample CUDA implementation of libnvsample_cudaprocess.so can be replaced by a custom CUDA implementation.

VIDEO ROTATION WITH GSTREAMER-1.0

The NVIDIA proprietary nvvidconv Gstreamer-1.0 plug-in also allows you to perform video rotation operations.

The following table shows the supported values for the nvvidconv flip-method property.

<table>
<thead>
<tr>
<th>Flip Method</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>identity - no rotation (default)</td>
<td>0</td>
</tr>
<tr>
<td>counterclockwise - 90 degrees</td>
<td>1</td>
</tr>
<tr>
<td>rotate - 180 degrees</td>
<td>2</td>
</tr>
<tr>
<td>clockwise - 90 degrees</td>
<td>3</td>
</tr>
<tr>
<td>horizontal flip</td>
<td>4</td>
</tr>
<tr>
<td>upper right diagonal flip</td>
<td>5</td>
</tr>
<tr>
<td>vertical flip</td>
<td>6</td>
</tr>
<tr>
<td>upper-left diagonal</td>
<td>7</td>
</tr>
</tbody>
</table>

Note: Get information on nvvidconv flip-method property with the gst-inspect-1.0 nvvidconv command.

To rotate video 90 degrees counterclockwise

- To rotate video 90 degrees in a counterclockwise direction, enter the following command.

```bash
gst-launch-1.0 filesrc location=<filename.mp4>! qtdemux name=demux ! h264parse ! omxh264dec ! nvvidconv flip-method=1 ! 'video/x-raw(memory:NVMM), format=(string)I420' ! nvoverlaysink -e
```

To rotate video 90 degrees clockwise

- To rotate video 90 degrees in a clockwise direction, enter the following command:

```bash
gst-launch-1.0 filesrc location=<filename.mp4> ! qtdemux name=demux ! h264parse ! omxh264dec ! nvvidconv flip-method=3 ! 'video/x-raw(memory:NVMM), format=(string)I420' ! omxh264enc ! qtmux ! filesink location=test.mp4 -e
```
Rotate 180 degrees

To rotate video 180 degrees, enter the following command:

```bash
gst-launch-1.0 nvcamerasrc fpsRange="30.0 30.0" ! 'video/x-raw(memory:NVMM), width=(int)1920, height=(int)1080, format=(string)I420, framerate=(fraction)30/1' ! nvtee ! nvvidconv flip-method=2 ! 'video/x-raw(memory:NVMM), format=(string)I420' ! nvoverlaysink -e
```

To scale and rotate video 90 degrees counterclockwise

To scale and rotate video 90 degrees counterclockwise, enter the following command:

```bash
gst-launch-1.0 filesrc location=<filename_1080p.mp4> ! qtdemux ! h264parse ! omxh264dec ! nvvidconv flip-method=1 ! 'video/x-raw(memory:NVMM), width=(int)480, height=(int)640, format=(string)I420' ! nvoverlaysink -e
```

To scale and rotate video 90 degrees clockwise

To scale and rotate video 90 degrees clockwise, enter the following command:

```bash
gst-launch-1.0 nvcamerasrc fpsRange="30.0 30.0" ! 'video/x-raw(memory:NVMM), width=(int)1920, height=(int)1080, format=(string)I420, framerate=(fraction)30/1' ! nvtee ! nvvidconv flip-method=3 ! 'video/x-raw(memory:NVMM), width=(int)480, height=(int)640, format=(string)I420' ! nvoverlaysink -e
```

To scale and rotate video 180 degrees

To scale and rotate video 180 degrees, enter the following command:

```bash
gst-launch-1.0 filesrc location=<filename_1080p.mp4> ! qtdemux ! h264parse ! omxh264dec ! nvvidconv flip-method=2 ! 'video/x-raw(memory:NVMM), width=(int)640, height=(int)480, format=(string)I420' ! nvoverlaysink -e
```

**INTERPOLATION METHODS FOR VIDEO SCALING**

The NVIDIA proprietary `nvvidconv` Gstreamer-1.0 plug-in allows you to choose the interpolation method used for scaling.

The following table shows the supported values for the `nvvidconv interpolation-method` property.
<table>
<thead>
<tr>
<th>Interpolation Method</th>
<th>Property Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>nearest</td>
<td>0</td>
</tr>
<tr>
<td>linear</td>
<td>1</td>
</tr>
<tr>
<td>smart (default)</td>
<td>2</td>
</tr>
<tr>
<td>bilinear</td>
<td>3</td>
</tr>
</tbody>
</table>

Note: Get information on `nvvidconv interpolation-method` property with the `gst-inspect-1.0 nvvidconv` command.

To use bilinear interpolation method for scaling

- Enter the following command:

```
gst-launch-1.0 filesrc location=<filename_1080p.mp4> ! qtdemux name=demux ! h264parse ! omxh264dec ! nvvidconv interpolation-method=3 ! 'video/x-raw(memory:NVMM), format=(string)I420, width=1280, height=720' ! nvoverlaysink -e
```

EGLSTREAM PRODUCER EXAMPLE

The NVIDIA-proprietary `nveglstreamsrc` and `nvvideo2sink` Gstreamer-1.0 plug-ins allow simulation of an EGLStream producer pipeline (for preview only.)

To simulate an EGLStream producer pipeline

- Enter the following command:

```
nvgstcapture-1.0 --camsrc=3 --nvvideo2sink-create-eglstream
```

EGL IMAGE TRANSFORM EXAMPLE

The NVIDIA proprietary `nvegltransform` Gstreamer-1.0 plug-in allows simulation of an EGLImage transform pipeline.

To simulate an EGL Image transform pipeline

- Enter the following command:

```
gst-launch-1.0 filesrc location=<filename_h264_1080p.mp4> ! qtdemux ! h264parse ! omxh264dec ! nvvidconv ! 'video/x-raw(memory:NVMM), width=(int)1280, height=(int)720, format=(string)NV12' ! nvegltransform ! nveglglessink -e
```
This section provides a procedure for building current versions of gstreamer.

**Using gst-install to build GStreamer**

This release contains the `git-install` script to install a specific GStreamer version. To install, execute:

```
gst-install [--prefix=<install_path>] [--version=<version>]
```

Where:
- `<install_path>` is the location where you are installing GStreamer
- `<version>` is the GStreamer version

For example:

```
gst-install --prefix=/home/ubuntu/gst-1.6.0 --version=1.6.0
```

**To build GStreamer manually**

1. Download the latest version of gstreamer available at:

   `http://gstreamer.freedesktop.org/src/`

   The following are the files you need from version 1.6.0:
   - `gstreamer-1.6.0.tar.xz`
   - `gst-plugins-base-1.6.0.tar.xz`
   - `gst-plugins-good-1.6.0.tar.xz`
   - `gst-plugins-bad-1.6.0.tar.xz`
   - `gst-plugins-ugly-1.6.0.tar.xz`
2. Install needed packages with the following command:

```
sudo apt-get install build-essential dpkg-dev flex bison autotools-dev automake liborc-dev autopoint libtool gtk-doc-tools libgstreamer1.0-dev
```

3. In the `~` directory, create a `gst_<version>` directory, where `<version>` is the version number of gstreamer you are building.

4. Copy the downloaded tar.xz files to the `gst_<version>` directory.

5. Uncompress the tar.xz files in the `gst_<version>` directory.

6. Set the `PKG_CONFIG_PATH` with the following command:

```
export PKG_CONFIG_PATH=/home/ubuntu/gst_1.6.0/out/lib/pkgconfig
```

7. Build gstreamer (in this example, gstreamer-1.6.0) with the following commands:

```
./configure --prefix=/home/ubuntu/gst_1.6.0/out
make
make install
```

8. Build gst-plugins-base-1.6.0 with the following commands:

```
sudo apt-get install libxv-dev libasound2-dev libtheora-dev libogg-dev libvorbis-dev
./configure --prefix=/home/ubuntu/gst_1.6.0/out
make
make install
```

9. Build gst-plugins-good-1.6.0 with the following commands:

```
sudo apt-get install libbz2-dev libv4l-dev libvpx-dev libjack-jackd2-dev libsoup2.4-dev libpulse-dev
./configure --prefix=/home/ubuntu/gst_1.6.0/out
make
make install
```

10. Obtain and build gst-plugins-bad-1.6.0 with the following commands:

```
sudo apt-get install faad libfaad-dev libfaac-dev
./configure --prefix=/home/ubuntu/gst_1.6.0/out
make
make install
```
11. Obtain and build gst-plugins-ugly-1.6.0 with the following commands:

```
sudo apt-get install libx264-dev libmad0-dev
./configure --prefix=/home/ubuntu/gst_1.6.0/out
make
make install
```

12. Set the LD_LIBRARY_PATH environment variable with the following command:

```
export LD_LIBRARY_PATH=/home/ubuntu/gst_1.6.0/out/lib/
```

13. Copy the nvidia gstreamer-1.0 libraries to the gst_1.6.0 plugin directory using the following command:

```
cd /usr/lib/arm-linux-gnueabihf/gstreamer-1.0/
cp libgstnv* libnvgst* libgstomx.so ~/gst_1.6.0/out/lib/gstreamer-1.0/
```

The nvidia gstreamer-1.0 libraries include:

```
libgstnvcamera.so
libgstnvegllessink.so
libgstnveglstreams.src.so
libgstnvegltransform.so
libgstnvvivafilter.so
libgstnvvidconv.so
libgstnvvideosink.so
libnvgstjpeg.so
libgstomx.so
```
This section describes the options available in the `nvgstcapture-1.0` application.

**NVGSTCAPTURE APPLICATION OPTIONS**

Nvgstcapture-1.0 command-line options are described in the following table.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>--prev_res</td>
<td>Preview area width and height, e.g., <code>--prev_res=3</code></td>
<td>-</td>
</tr>
<tr>
<td>--cus-prev-res</td>
<td>Custom preview width and height for CSI only</td>
<td>-</td>
</tr>
<tr>
<td>--image_res</td>
<td>Image width and height, e.g., <code>--image_res=3</code></td>
<td>-</td>
</tr>
<tr>
<td>--video_res</td>
<td>Video width and height, e.g., <code>--video_res=3</code></td>
<td>-</td>
</tr>
<tr>
<td>-m, --mode</td>
<td>Capture mode.</td>
<td>1-Still</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2-Video</td>
</tr>
<tr>
<td>-v, --video_enc</td>
<td>Video encoder type.</td>
<td>0-H.264 (hardware)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1-VP8 (hardware)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2-MPEG-4 (software)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3-H.263 (software)</td>
</tr>
<tr>
<td>-b, --enc-bitrate</td>
<td>Video encoding Bit-rate(in bytes)</td>
<td>Example: <code>--enc-bitrate=4000000</code></td>
</tr>
<tr>
<td>--enc-profile</td>
<td>Video encoder profile (only for H.264)</td>
<td>0-Baseline</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1-Main</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2-High</td>
</tr>
<tr>
<td>-j, --image_enc</td>
<td>Image encoder type.</td>
<td>0-jpeg_SW[jpegenc]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1-jpeg_HW[nvjpegenc]</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
<td>Values</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>-k, --file_type</td>
<td>Container file type.</td>
<td>0-MP4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1-3GP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2-AVI</td>
</tr>
<tr>
<td>--cap-dev-node</td>
<td>Video capture device node.</td>
<td>0=/dev/video0[default]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1=/dev/video1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2=/dev/video2</td>
</tr>
<tr>
<td>--svs</td>
<td>Chain for video preview.</td>
<td>-</td>
</tr>
<tr>
<td>--file-name</td>
<td>File name for capture.</td>
<td>“nvcamtest” is used by default.</td>
</tr>
<tr>
<td>--camsrc</td>
<td>Camera source.</td>
<td>0-v4l2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1-csi (default)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2-videotest</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3-eglstream</td>
</tr>
<tr>
<td>--orientation</td>
<td>Camera sensor orientation value(CSI only)</td>
<td>-</td>
</tr>
<tr>
<td>-w, --whitebalance</td>
<td>White balance value for capture.</td>
<td>(CSI only)</td>
</tr>
<tr>
<td>-s, --scene-mode</td>
<td>Camera scene-mode value.</td>
<td>(CSI only)</td>
</tr>
<tr>
<td>-c, --color-effect</td>
<td>Camera color effect value.</td>
<td>(CSI only)</td>
</tr>
<tr>
<td>--auto-exposure</td>
<td>Camera auto-exposure value.</td>
<td>(CSI only)</td>
</tr>
<tr>
<td>--flash</td>
<td>Camera flash value.</td>
<td>(CSI only)</td>
</tr>
<tr>
<td>--flicker</td>
<td>Camera flicker detection and avoidance mode value.</td>
<td>(CSI only)</td>
</tr>
<tr>
<td>--contrast</td>
<td>Camera contrast value.</td>
<td>(CSI only)</td>
</tr>
<tr>
<td>--saturation</td>
<td>Camera saturation value.</td>
<td>(CSI only)</td>
</tr>
<tr>
<td>--edge-enhancement</td>
<td>Camera edge enhancement value.</td>
<td>(CSI only)</td>
</tr>
<tr>
<td>--tnr_strength</td>
<td>Camera TNR strength value.</td>
<td>(CSI only)</td>
</tr>
<tr>
<td>--tnr_mode</td>
<td>Camera TNR mode value.</td>
<td>(CSI only)</td>
</tr>
<tr>
<td>--sensor-id</td>
<td>Camera Sensor ID value.</td>
<td>(CSI only)</td>
</tr>
<tr>
<td>--display-id</td>
<td>Display ID value (for nvoverlay only)</td>
<td>-</td>
</tr>
<tr>
<td>--eglstream-id</td>
<td>Select EGLStreamProducerID value (for CSI EGLStream). Default is 0.</td>
<td>-</td>
</tr>
<tr>
<td>--aeRegion</td>
<td>ROI for AE coordinates (top, left, bottom, right) and weight, in that order.</td>
<td>(CSI only)</td>
</tr>
</tbody>
</table>

Example: 
```
--aeRegion="30 40 200 200 1.2"
```
### Option Reference

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>--wbRegion</td>
<td>ROI for AWB coordinates (top, left, bottom, right) and weight in that order. (CSI only)</td>
<td>Example: --wbRegion=&quot;30 40 200 200 1.2&quot;</td>
</tr>
<tr>
<td>--fpsRange</td>
<td>FPS range values (low, high) (CSI only)</td>
<td>Example: --fpsRange=&quot;15 30&quot;</td>
</tr>
<tr>
<td>--wbGains</td>
<td>White Balance (WB) gains values (R, GR, GB, B) in that order. (CSI only)</td>
<td>Example: --wbGains=&quot;1.2 1.4 0.8 1.6&quot;</td>
</tr>
<tr>
<td>--overlayConfig</td>
<td>Overlay Configuration Options index and coordinates in (index, x_pos, y_pos, width, height) order.</td>
<td>Example: --overlayConfig=&quot;0, 0, 0, 1280, 720&quot;</td>
</tr>
<tr>
<td>--enable-meta</td>
<td>Enables Sensor MetaData reporting if the sensor has the capability to provide the embedded metadata.</td>
<td>-</td>
</tr>
<tr>
<td>--eglConfig</td>
<td>EGL window Coordinates (x_pos, y_pos) in that order.</td>
<td>Example: --eglConfig=&quot;50 100&quot;</td>
</tr>
<tr>
<td>--enable-exif</td>
<td>Enable Exif data</td>
<td>-</td>
</tr>
<tr>
<td>--dump-bayer</td>
<td>Dump bayer data in addition to image capture</td>
<td>-</td>
</tr>
<tr>
<td>--exposure-time</td>
<td>Capture exposure time value. (CSI only)</td>
<td>Example: --exposure-time=0.033</td>
</tr>
</tbody>
</table>

#### Help Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>-h, --help</td>
<td>Show help options.</td>
<td>-</td>
</tr>
<tr>
<td>--help-all</td>
<td>Show all help options.</td>
<td>-</td>
</tr>
<tr>
<td>--help-gst</td>
<td>Show Gstreamer options.</td>
<td>-</td>
</tr>
</tbody>
</table>

### CSI CAMERA SUPPORTED RESOLUTIONS

CSI camera supports the following image resolutions:

- 640x480
- 1280x720
- 1920x1080
- 2104x1560
- 2592x1944
- 2616x1472
- 3840x2160
- 3896x2192
- 4208x3120
CSI CAMERA RUNTIME COMMANDS

CSI camera runtime commands are described in the following table.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>h</td>
<td>Help</td>
<td>-</td>
</tr>
<tr>
<td>q</td>
<td>Quit</td>
<td>-</td>
</tr>
<tr>
<td>mo:&lt;value&gt;</td>
<td>Set capture mode</td>
<td>1-image</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2-video</td>
</tr>
<tr>
<td>gmo</td>
<td>Get capture mode</td>
<td>-</td>
</tr>
<tr>
<td>sid:&lt;value&gt;</td>
<td>Set sensor ID</td>
<td>-</td>
</tr>
<tr>
<td>gsid</td>
<td>Get sensor ID</td>
<td>-</td>
</tr>
<tr>
<td>so:&lt;val&gt;</td>
<td>Set sensor orientation</td>
<td>(0): none</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(1): Rotate counter-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>clockwise 90 degrees</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(2): Rotate 180 degrees</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(3): Rotate clockwise 90</td>
</tr>
<tr>
<td></td>
<td></td>
<td>degrees</td>
</tr>
<tr>
<td>gso</td>
<td>Get sensor orientation</td>
<td>-</td>
</tr>
<tr>
<td>wb:&lt;value&gt;</td>
<td>Set white balance mode</td>
<td>0-off</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1-auto</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2-incandescent</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3-fluorescent</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4-warm-fluorescent</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5-daylight</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6-cloudy-daylight</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7-twilight</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8-shade</td>
</tr>
<tr>
<td>gwb</td>
<td>Get white balance mode</td>
<td>-</td>
</tr>
<tr>
<td>scm:&lt;value&gt;</td>
<td>Set scene mode</td>
<td>0-face-priority</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1-action</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2-portrait</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3-landscape</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4-night</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5-night-portrait</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6-theatre</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7-beach</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8-snow</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9-sunset</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10-steady-photo</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11-fireworks</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12-sports</td>
</tr>
<tr>
<td></td>
<td></td>
<td>13-party</td>
</tr>
<tr>
<td></td>
<td></td>
<td>14-candle-light</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15-barcode</td>
</tr>
<tr>
<td>gcm</td>
<td>Get scene mode</td>
<td>-</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
<td>Values</td>
</tr>
<tr>
<td>---------</td>
<td>--------------</td>
<td>--------</td>
</tr>
<tr>
<td>ce: &lt;value&gt;</td>
<td>Set color effect mode</td>
<td>1-off, 2-mono, 3-negative, 4-solarize, 5-sepia, 6-posterize, 7-aqua</td>
</tr>
<tr>
<td>gce</td>
<td>Get color effect mode</td>
<td>-</td>
</tr>
<tr>
<td>ae: &lt;value&gt;</td>
<td>Set auto-exposure mode</td>
<td>1-off, 2-on, 3-OnAutoFlash, 4-OnAlwaysFlash, 5-OnFlashRedEye</td>
</tr>
<tr>
<td>gae</td>
<td>Get auto exposure mode</td>
<td>-</td>
</tr>
<tr>
<td>f: &lt;value&gt;</td>
<td>Set flash mode</td>
<td>0-off, 1-on, 2-torch, 3-auto</td>
</tr>
<tr>
<td>gf</td>
<td>Get flash mode</td>
<td>-</td>
</tr>
<tr>
<td>fl: &lt;value&gt;</td>
<td>Set flicker detection and avoidance mode</td>
<td>0-off, 1-50 Hz, 2-60 Hz, 3-auto</td>
</tr>
<tr>
<td>gfl</td>
<td>Get flicker detection and avoidance mode</td>
<td>-</td>
</tr>
<tr>
<td>ct: &lt;value&gt;</td>
<td>Set contrast</td>
<td>0-1, e.g., ct:0.75</td>
</tr>
<tr>
<td>gct</td>
<td>Get contrast</td>
<td>-</td>
</tr>
<tr>
<td>st: &lt;value&gt;</td>
<td>Set saturation</td>
<td>0-2, e.g., st:1.25</td>
</tr>
<tr>
<td>gst</td>
<td>Get saturation</td>
<td>-</td>
</tr>
<tr>
<td>ext: &lt;value&gt;</td>
<td>Set exposure time (in seconds)</td>
<td>e.g., ext:0.033</td>
</tr>
<tr>
<td>gext</td>
<td>Get exposure time</td>
<td>-</td>
</tr>
<tr>
<td>ee: &lt;value&gt;</td>
<td>Set edge enhancement</td>
<td>0-1, e.g., ee:0.75</td>
</tr>
<tr>
<td>gee</td>
<td>Get edge enhancement</td>
<td>-</td>
</tr>
<tr>
<td>aer: &lt;value&gt;</td>
<td>Set ROI coordinates for AE (top, left, bottom, right) and weight</td>
<td>e.g., aer:20 20 400 400 1.2</td>
</tr>
<tr>
<td>gaer</td>
<td>Get ROI for AE</td>
<td>-</td>
</tr>
<tr>
<td>wbr: &lt;value&gt;</td>
<td>Set ROI coordinates for AWB (top, left, bottom, right) and weight</td>
<td>e.g., wbr:20 20 400 400 1.2</td>
</tr>
<tr>
<td>gwbr</td>
<td>Get ROI for AE</td>
<td>-</td>
</tr>
<tr>
<td>fpsr: &lt;value&gt;</td>
<td>Set FPS range (low, high)</td>
<td>e.g., fpsr:15 30</td>
</tr>
<tr>
<td>gfpsr</td>
<td>Get FPS range</td>
<td>-</td>
</tr>
<tr>
<td>wbg: &lt;value&gt;</td>
<td>Set WB gains (R, GR, GB, B)</td>
<td>e.g., wbg:1.2 2.2 0.8 1.6</td>
</tr>
</tbody>
</table>
### USB CAMERA RUNTIME COMMANDS

USB camera runtime commands are described in the following table.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>h</td>
<td>Help</td>
<td>-</td>
</tr>
<tr>
<td>q</td>
<td>Quit</td>
<td>-</td>
</tr>
<tr>
<td>mo:&lt;value&gt;</td>
<td>Set capture mode</td>
<td>1-image 2-video</td>
</tr>
<tr>
<td>gmo</td>
<td>Get capture mode</td>
<td>-</td>
</tr>
<tr>
<td>j</td>
<td>Capture one image.</td>
<td>-</td>
</tr>
<tr>
<td>jx&lt;delay&gt;</td>
<td>Capture after a delay of &lt;delay&gt;, e.g., jx5000 to capture after a 5-second delay</td>
<td>-</td>
</tr>
<tr>
<td>j:&lt;value&gt;</td>
<td>Capture &lt;count&gt; number of images in succession, e.g., j:6 to capture 6 images.</td>
<td>-</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
<td>Notes</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>-------</td>
</tr>
<tr>
<td>br:&lt;val&gt;</td>
<td>Sets encoding bit-rate (in bytes)</td>
<td>Example: br:4000000</td>
</tr>
<tr>
<td>gbr</td>
<td>Gets encoding bit-rate (in bytes)</td>
<td>-</td>
</tr>
<tr>
<td>ep:&lt;val&gt;</td>
<td>Sets encoding profile (for H.264 only)</td>
<td>Example: ep:1 (0): Baseline (1): Main (2): High</td>
</tr>
<tr>
<td>gep</td>
<td>Gets encoding profile (for H.264 only)</td>
<td>-</td>
</tr>
<tr>
<td>Enter+f</td>
<td>Forces IDR frame on video encoder (for H.264 only)</td>
<td>-</td>
</tr>
</tbody>
</table>

**NOTES**

- The nvgstcapture-1.0 application generates image and video output files in the same directory as the application itself.
- Filenames for image and video content are in the formats nvcamtest<counter>.jpg and nvcamtest<counter>.mp4 respectively, where <counter> is a counter starting from 0 every time you run the application. Rename or move files between runs to avoid overwriting results you want to save.
- Default H.263 encode resolution is 704x576(4CIF) in AVI container formats. Use --camsrc=2 for H.263 video encode.
- The `nvgstcapture-1.0` application supports native capture (video only) mode by default.
- Advance features, like setting zoom, brightness, exposure, and whitebalance levels, are not supported for USB camera.
The Gstreamer-1.0-based gst-omx video encoders support the following features, respectively:

<table>
<thead>
<tr>
<th>Video Encoder Feature</th>
<th>H264enc</th>
<th>H265enc</th>
<th>Vp8enc</th>
</tr>
</thead>
<tbody>
<tr>
<td>profile (Baseline / Main / High)</td>
<td>✓ (all)</td>
<td>✓ (Main)</td>
<td>✓</td>
</tr>
<tr>
<td>bitrate</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>insert-spspsatidr</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>control-rate</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>iframeinterval</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>qp-range</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>temporal-tradeoff</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>bit-packetization</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>preset-level</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>low-latency</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>slice-header spacing</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>force-IDR</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>vbv-size</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>sliceintrarefreshenable</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>sliceintrarefreshinterval</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>EnableTwoPassCBR</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>num-B-Frames</td>
<td>✓</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
This section describes the supported cameras.

**CSI CAMERAS**

- Jetson TX1 currently supports only 1 CSI RAW BAYER sensor.
- The platform has been validated with a single OV5693 sensor for capture on L4T.
- The camera module is interfaced with the Tegra platform via MIPI-CSI.
- Tested using the nvgstcapture application.

**USB 2.0 CAMERAS**

The following cameras have been validated on Tegra platforms for Android and L4T with USB 2.0 ports. These cameras are UVC compliant.

- Logitech c920 (preferred)
  

- Logitech c910
  
  [http://www.amazon.com/Logitech-HD-Pro-Webcam-C910/dp/B003M2YT96](http://www.amazon.com/Logitech-HD-Pro-Webcam-C910/dp/B003M2YT96)

- Rocketfish™ HD Webcam Pro
  

- Creative Live! Cam Socialize HD 1080
  
  [http://support.creative.com/Products/ProductDetails.aspx?catID=218&CatName=WebCameras&subCatID=231&subCatName=MIDI+Keyboards&prodID=20165&prodName=Live!+Cam+Socialize+HD+1080&bTopTwenty=1&VARSET=prodfaq:PRODFAQ_20165,VARSET=CategoryID:218](http://support.creative.com/Products/ProductDetails.aspx?catID=218&CatName=WebCameras&subCatID=231&subCatName=MIDI+Keyboards&prodID=20165&prodName=Live!+Cam+Socialize+HD+1080&bTopTwenty=1&VARSET=prodfaq:PRODFAQ_20165,VARSET=CategoryID:218)
INDUSTRIAL CAMERA DETAILS

The following USB 3.0 Industrial cameras are supported on Jetson-TX1 under L4T:

- See3CAM_CU130
  - USB 3.0
  - UVC compliant
  - 3840 x 2160 at 30 FPS | 4224 x 3156 at 13 FPS
  - Purpose - Embedded Navigation
  - Test using the nvgstcapture app.
  - Issues encountered:
    - FPS cannot be fixed. Changes based on exposure.
    - FPS cannot be changed. Needs payment to vendor to get the support added to their firmware.

- MQ003CG-CM
  - USB 3.0
  - Non-UVC compliant
  - 640 x 480 at 500 FPS
  - Purpose - Embedded Robotics
  - Installation and Verification on Jetson TX1:
    1. Add the user to the plugdev group:
       ```
       sudo gpasswd -a ubuntu plugdev
       ```
       Re-login.
    2. Install tools for the application:
       ```
       apt-get install libgstreamer0.10-dev libgstreamer-plugins-base0.10-dev libgtk2.0-dev g++
       ```
    3. Download XIMEA Linux Software Package:
       ```
       wget http://www.ximea.com/downloads/recent/XIMEA_Linux_SP.tgz
       ```
       Untar:
       ```
       tar xzf XIMEA_Linux_SP.tgz
       cd package
       ```
4. Open the install file and replace

```bash
elif [ "${arch:0:3}" == "arm" ]
with
elif [ "$arch" == "aarch64" ]
```

5. Start installation:

```bash
./install
```

Install USB3 camera:

```bash
./install -cam_usb30
```

Install graphical desktop:

```bash
sudo apt-get update
sudo apt-get install ubuntu-desktop
```

6. Reboot. The system boots to the graphical desktop.
7. To access sample applications:

   - xiSample: run from `/package/bin` folder
   - streamViewer
     ```bash
     make from `/package/examples/streamViewer` folder
     run from the `/package/bin` folder
     ```
Notice

ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OR CONDITION OF TITLE, MERCHANTABILITY, SATISFACTORY QUALITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT, ARE HEREBY EXCLUDED TO THE MAXIMUM EXTENT PERMITTED BY LAW.

Information furnished is believed to be accurate and reliable. However, NVIDIA Corporation assumes no responsibility for the consequences of use of such information or for any infringement of patents or other rights of third parties that may result from its use. No license is granted by implication or otherwise under any patent or patent rights of NVIDIA Corporation. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. NVIDIA Corporation products are not authorized for use as critical components in life support devices or systems without express written approval of NVIDIA Corporation.

Trademarks

NVIDIA and the NVIDIA logo are trademarks or registered trademarks of NVIDIA Corporation in the United States and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright

© 2015, 2016, 2017 NVIDIA Corporation. All rights reserved.