## 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>
</tbody>
</table>
# TABLE OF CONTENTS

## Tegra X1/Tegra Linux Driver Package Multimedia User Guide
- Gstreamer-0.10 Installation and Setup .................................................. 2
- Gstreamer-1.0 Installation and Setup .................................................... 3
- Decode Examples .................................................................................. 4
  - Audio Decode Examples Using gst-launch-0.10 .................................. 4
  - Audio Decode Examples Using gst-launch-1.0 .................................... 4
  - Video Decode Examples Using gst-launch-0.10 ................................... 5
  - Video Decode Examples Using gst-launch-1.0 .................................... 6
- Encode Examples .................................................................................. 7
  - Audio Encode Examples Using gst-launch-0.10 .................................. 7
  - Audio Encode Examples Using gst-launch-1.0 .................................... 7
  - Video Encode Examples Using gst-launch-0.10 ................................... 7
- Supported H.264 Encoder Features with Gstreamer-0.10 ....................... 8
  - Video Encode Examples Using gst-launch-1.0 ................................... 9
- Camera Capture with Gstreamer-0.10 .................................................. 10
- Video Playback with Gstreamer-0.10 .................................................. 11
- Video Playback with Gstreamer-1.0 .................................................. 12
- Video Format Conversion with Gstreamer-1.0 ...................................... 13
  - raw-yuv Input Formats ..................................................................... 13
  - raw-gray Input Formats ................................................................... 13
  - raw-yuv Output Formats ................................................................. 13
  - raw-gray Output Formats ............................................................... 13
- Video Scaling with Gstreamer-1.0 ........................................................ 14
  - raw-yuv Input Formats ..................................................................... 14
  - raw-gray Input Formats ................................................................... 14
  - raw-yuv Output Formats ................................................................. 14
  - raw-gray Output Formats ............................................................... 14
- NVIDIA Input and Output Formats ..................................................... 15
- 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

## GStreamer Build Instructions .............................................................. 20

## Nvgstcapture-1.0 Option Reference .................................................... 23
Nvgstcapture Application Options ................................................................. 23
CSI Camera Supported Resolutions .......................................................... 25
CSI Camera Runtime Commands .............................................................. 25
USB Camera Runtime Commands .............................................................. 28
Notes ...................................................................................................... 29

Video Encoder Features ........................................................................ 30

Supported Cameras ............................................................................... 31
  CSI Cameras ...................................................................................... 31
  USB 2.0 Cameras .............................................................................. 31

INDUSTRIAL CAMERA DETAILS .............................................................. 32
This document is a user guide for the Gstreamer (versions 0.10 and 1.0) based accelerated solution included in NVIDIA® Tegra® Linux Driver Package for Ubuntu Linux 14.04 on platforms including Tegra X1 devices.

This document contains the following sections:

- Gstreamer-0.10 Installation and Setup
- Gstreamer-1.0 Installation and Setup
- Decode Examples
- Encode Examples
- Camera Capture with Gstreamer-0.10
- Camera Capture with Gstreamer-1.0
- Video Playback with Gstreamer-0.10
- Video Playback with Gstreamer-1.0
- Video Format Conversion with Gstreamer-1.0
- Video Scaling with Gstreamer-1.0
- Video Transcode with Gstreamer-1.0
- CUDA Video Post-Processing with Gstreamer-1.0
- Video Rotation with Gstreamer-1.0
- Gstreamer Build Instructions
- Nvgstcapture-1.0 Option Reference
- Video Encoder Features
- Supported USB Camera
GSTREAMER-0.10 INSTALLATION AND SETUP

This section describes how to install and configure Gstreamer.

To install Gstreamer-0.10

- Install Gstreamer-0.10 on the target platform with the following command:

```
sudo apt-get install gstreamer-tools gstreamer0.10-alsa
               gstreamer0.10-plugins-base  gstreamer0.10-plugins-good
               gstreamer0.10-plugins-bad gstreamer0.10-plugins-ugly
```

To check the Gstreamer-0.10 version

- Check the Gstreamer-0.10 version with the following command:

```
gst-inspect-0.10 --version
```

Note: Gstreamer version 0.10 plugins are included pre-installed in Linux for Tegra (L4T) 23.00 release package.

Gstreamer version 0.10 includes the following gst-openmax video decoders:

<table>
<thead>
<tr>
<th>Video Decoder</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nv_omx_h264dec</td>
<td>OpenMAX IL H.264/AVC video decoder</td>
</tr>
<tr>
<td>nv_omx_mpeg4dec</td>
<td>OpenMAX IL MPEG-4 video decoder</td>
</tr>
<tr>
<td>nv_omx_vp8dec</td>
<td>OpenMAX IL VP8 video decoder</td>
</tr>
</tbody>
</table>

Gstreamer version 0.10 includes the following gst-openmax video encoders:

<table>
<thead>
<tr>
<th>Video Encoders</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nv_omx_h264enc</td>
<td>OpenMAX IL H.264/AVC video encoder</td>
</tr>
<tr>
<td>nv_omx_vp8enc</td>
<td>OpenMAX IL VP8 video encoder</td>
</tr>
</tbody>
</table>

Gstreamer version 0.10 includes the following gst-openmax video sinks:

<table>
<thead>
<tr>
<th>Video Sink</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nv_omx_videosink</td>
<td>OpenMAX IL videosink element</td>
</tr>
<tr>
<td>nv_omx_hdmi_videosink</td>
<td>OpenMAX IL HDMI videosink element</td>
</tr>
<tr>
<td>(deprecated)</td>
<td></td>
</tr>
</tbody>
</table>
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 command:

```
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
```

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>omxh263dec</td>
<td>OpenMAX IL H.263 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>nvooverlaysink</td>
<td>OpenMAX IL videosink element</td>
</tr>
<tr>
<td>nvhdmioverlaysink (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>nveglstreams</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>

**DECODE EXAMPLES**

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

**Audio Decode Examples Using gst-launch-0.10**

The following examples show how you can perform audio decode using Gstreamer-0.10.

**AAC Decode (OSS software decode)**

```bash
gst-launch-0.10 filesrc location=<filename.mp4> ! qtdemux name=demux
demux.audio_00 ! queue ! ffdec_aac ! alsasink -e
```

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

```bash
gst-launch-0.10 filesrc location=<filename.mp4> ! qtdemux name=demux
demux.audio_00 ! queue ! ffdec_amrwb ! audioconvert ! alsasink -e
```

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

```bash
gst-launch-0.10 filesrc location=<filename.mp4> ! qtdemux name=demux
demux.audio_00 ! queue ! ffdec_amrnb ! audioconvert ! alsasink -e
```

**MP3 Decode (OSS software decode)**

```bash
gst-launch-0.10 filesrc location=<filename.mp3> ! mpegaudioparse !
ffdec_mp3 ! audioconvert ! alsasink -e
```

**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_amrnb ! 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 `aux_plug`.

Video Decode Examples Using `gst-launch-0.10`

The following examples show how you can perform video decode using Gstreamer-0.10.

H.264 Decode (NVIDIA accelerated decode)

```bash
gst-launch-0.10 filesrc location=<filename.mp4> ! qtdemux name=demux
demux.video_00 ! queue ! nv_omx_h264dec ! nv_omx_hdmi_videosink -e
```

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

```bash
gst-launch-0.10 filesrc location=<filename.mp4> ! qtdemux name=demux
demux.video_00 ! queue ! nv_omx_h264dec ! nv_omx_videosink display-id=1 -e
```

VP8 Decode (NVIDIA accelerated decode)

```bash
gst-launch-0.10 filesrc location=<filename.mp4> ! qtdemux name=demux
demux.video_00 ! queue ! nv_omx_vp8dec ! nv_omx_videosink -e
```
MPEG-4 Decode (NVIDIA accelerated decode)

gst-launch-0.10 filesrc location=<filename.mp4> ! qtdemux name=demux
demux.video_00 ! queue ! nv_omx_mpeg4dec ! nv_omx_videosink -e

Image Decode

gst-launch-0.10 filesrc location=<filename.jpg> ! nvjpegdec ! freeze !
xvimagesink -e

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)

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)

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)

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.

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)

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

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-0.10

The following examples show how you can perform audio encode using Gstreamer-0.10.

AAC Encode (OSS software encode)

gst-launch-0.10 audiotestsrc ! 'audio/x-raw-int, rate=(int)44100, channels=(int)2' ! ffenc_aac ! qtmux ! filesink location=test.mp4 -e

AMR-WB Encode (OSS software encode)

gst-launch-0.10 audiotestsrc ! 'audio/x-raw-int, rate=(int)16000, channels=(int)1' ! voamrwbenc ! qtmux ! filesink location=test.mp4 -e

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)

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)

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-0.10

The following examples show how you can perform video encode using Gstreamer-0.10.
H.264 Encode (NVIDIA accelerated encode)

gst-launch-0.10 videotestsrc ! 'video/x-raw-yuv, width=(int)1280, height=(int)720, format=(fourcc)I420' ! nv_omx_h264enc ! qtmux ! filesink location=test.mp4 -e

VP8 Encode (NVIDIA accelerated encode)

gst-launch-0.10 videotestsrc ! 'video/x-raw-yuv, width=(int)1280, height=(int)720, format=(fourcc)I420' ! nv_omx_vp8enc ! qtmux ! filesink location=test.mp4 -e

MPEG-4 Encode (OSS software encode)

gst-launch-0.10 videotestsrc ! 'video/x-raw-yuv, width=(int)1280, height=(int)720, format=(fourcc)I420' ! ffenc_mpeg4 ! qtmux ! filesink location=test.mp4 -e

H.263 Encode (OSS software encode)

gst-launch-0.10 videotestsrc ! 'video/x-raw-yuv, width=(int)704, height=(int)576, format=(fourcc)I420' ! ffenc_h263 ! qtmux ! filesink location=test.mp4 -e

Image Encode

gst-launch-0.10 videotestsrc num-buffers=1 ! 'video/x-raw-yuv, width=(int)1280, height=(int)720, format=(fourcc)I420' ! nvjpegenc ! filesink location=test.jpg -e

Supported H.264 Encoder Features with Gstreamer-0.10

This section describes example gst-launch-0.10 usage for features supported by the NVIDIA accelerated H.264 encoder.

Note: Display detailed information on nv_omx_h264enc encoder properties with the gst-inspect-0.10 nv_omx_h264enc command.

Set I-frame interval

gst-launch-0.10 videotestsrc num-buffers=200 ! 'video/x-raw-yuv, width=(int)1280, height=(int)720, format=(fourcc)I420' ! nv_omx_h264enc ! iframeinterval=100 ! qtmux ! filesink location=test.mp4 -e

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

```bash
gst-launch-0.10 videotestsrc num-buffers=200 ! 'video/x-raw-yuv, width=(int)1280, height=(int)720, format=(fourcc)I420' ! nv_omx_h264enc rc-mode=0 ! qtmux ! filesink location=test.mp4 -e
```

Set quantization range for P and I frame

The format for the range is the following:

```
"<P_range>:<I_range>"
```

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

```bash
gst-launch-0.10 videotestsrc num-buffers=200 ! 'video/x-raw-yuv, width=(int)1280, height=(int)720, format=(fourcc)I420' ! nv_omx_h264enc qp-range="10-51:5-30" ! qtmux ! filesink location=test.mp4 -e
```

Set quality level

```bash
gst-launch-0.10 videotestsrc num-buffers=200 ! 'video/x-raw-yuv, width=(int)1280, height=(int)720, format=(fourcc)I420' ! nv_omx_h264enc quality-level=2 ! qtmux ! filesink location=test.mp4 -e
```

Set low latency attribute

```bash
gst-launch-0.10 videotestsrc num-buffers=200 ! 'video/x-raw-yuv, width=(int)1280, height=(int)720, format=(fourcc)I420' ! nv_omx_h264enc low-latency=1 ! 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)

```bash
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)

```
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)

```
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)

```
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)

```
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

```
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
```

**CAMERA CAPTURE WITH GSTREAMER-0.10**

The default image capture application in the R23.00 release is `nvgstcapture-0.10`. For usage information enter the following command:

```
nvgstcapture-0.10 --help
```

The `nvgstcapture-0.10` 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/VP9 encode)</td>
</tr>
</tbody>
</table>
MJPEG

<table>
<thead>
<tr>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preview display</td>
</tr>
<tr>
<td>Image capture</td>
</tr>
<tr>
<td>VGA, 640 x 480</td>
</tr>
<tr>
<td>720p, 1280 x 720</td>
</tr>
<tr>
<td>Video capture (480p, 720p, 1080p, MJPEG encode)</td>
</tr>
</tbody>
</table>

**raw-yuv Capture (I420 format) and preview display with xvimagesink**

```plaintext
gst-launch-0.10 v4l2src device="/dev/video0" ! "video/x-raw-yuv, width=640, height=480, format=(fourcc)I420" ! xvimagesink -v -e
```

**CAMERA CAPTURE WITH GSTREAMER-1.0**

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

```plaintext
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**

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

**VIDEO PLAYBACK WITH GSTREAMER-0.10**

The default playback application in the R23.00 release is `nvgstplayer-0.10`. For usage information enter the following command:

```plaintext
nvgstplayer-0.10 --help
```

Video can be output to HD displays using the HDMI connector on the platform. The Gstreamer-0.10 application supports currently the following video sinks:
Overlay Sink (Video playback on overlay in full-screen mode)

```
gst-launch-0.10 filesrc location=<filename.mp4> ! qtdemux name=demux
demux.video_00 ! queue ! nv_omx_h264dec ! nv OMX_Videosink -v -e
```

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

```
gst-launch-0.10 filesrc location=<filename.mp4> ! qtdemux name=demux
demux.video_00 ! queue ! nv_omx_h264dec ! nv OMX_Videosink overlay-
x=300 overlay-y=300 overlay-w=500 overlay-h=500 -v -e
```

If you specify values for overlay-x and overlay-y, you must also specify values for overlay-w and overlay-h.

Xvimagesink (Windowed video playback)

```
gst-launch-0.10 filesrc location=<filename.mp4> ! qtdemux name=demux
demux.video_00 ! queue ! nv_omx_h264dec ! 'video/x-nv-yuv' ! nvvidconv
! xvimagesink -v -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 you to convert 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)' ! 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)' ! 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.

```bash
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' ! 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.

```bash
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' ! 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.

```bash
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.

```bash
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>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NV12</td>
<td>NVIDIA gst-omx decoder output format.</td>
</tr>
<tr>
<td>I420</td>
<td>NVIDIA gst-omx encoder input format.</td>
</tr>
</tbody>
</table>

To scale between NVIDIA formats

- Scale between NVIDIA Formats with the following commands:

```bash
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)I420' ! nvoverlaysink –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 ! 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 ! 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 ! omxvp8enc ! qtmux name=mux ! filesink
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 ! omxh264enc ! qtmux name=mux ! filesink
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 !
avenc_mpeg4 ! qtmux name=mux ! filesink
demux.audio_0 ! queue ! aacparse !
mux.audio_0 -e
```

VP8 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 ! omxvp8dec ! nvvidconv !
avenc_mpeg4 ! qtmux name=mux ! filesink
demux.audio_0 ! queue ! aacparse !
mux.audio_0 -e
```

H.264 Decode to Theora 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 ! theoraenc !
oggmux name=mux ! filesink location=<Transcoded_filename.ogg> -e
```

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

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

**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**
Sample CSI Camera pipeline

gst-launch-1.0 nvcamerasrc fpsRange="30 30" ! 'video/x-raw(memory:NVMM), width=(int)3840, height=(int)2160, format=(string)I420, framerate=(fraction)30/1' ! nvtee ! nvvidconv cuda-process=true customer-lib-name="libsample_process.so" ! nvoverlaysink -e

VIDEO ROTATION WITH GSTREAMER-1.0

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

\begin{table}[h]
\centering
\begin{tabular}{|c|c|}
\hline
\textbf{flip-method} & \textbf{Property value} \\
\hline
counterclockwise - 90 degrees & 1 \\
rotate - 180 degrees & 2 \\
clockwise - 90 degrees & 3 \\
\hline
\end{tabular}
\caption{nvvidconv property details for rotation support}
\end{table}

\begin{itemize}
\item Note: Get information on \textit{nvvidconv} \textit{flip-method} property with the \texttt{gst-inspect-1.0} command.
\end{itemize}

To rotate video 90 degrees counterclockwise

\begin{itemize}
\item To rotate video 90 degrees in a counterclockwise direction, enter the following command:
\end{itemize}

\begin{verbatim}
   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
\end{verbatim}

To rotate video 90 degrees clockwise

\begin{itemize}
\item To rotate video 90 degrees in a clockwise direction, enter the following command:
\end{itemize}

\begin{verbatim}
   gst-launch-1.0 filesrc location=<filename.mp4> ! qtdemux name=demux ! h264parse ! omxh264dec ! nvvidconv flip-method=3 ! omxh264enc ! qtmux ! filesink location=test.mp4 -e
\end{verbatim}
Rotate 180 degrees

- To rotate video 180 degrees, enter the following command:

```
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:

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

```
gst-launch-1.0 filesrc location=<filename_1080p.mp4> ! qtdemux ! h264parse ! omxh264dec ! nvvidconv ! "video/x-raw(memory:NVMM), width=(int)640, height=(int)480, format=(string)I420" ! nvvidconv flip-method=3 ! omxh264enc ! qtmux ! filesink location=test.mp4 -e
```

To scale and rotate video 180 degrees

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

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


   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:

```bash
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:

```bash
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:

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

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

```bash
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:

```bash
sudo apt-get install libbz2-dev libv4l-dev libvpx-dev libjack-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:

```bash
sudo apt-get install faad libfaad-dev libfaac-dev libg1l1-mesa-dev libgles2-mesa-dev libopencv-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 nvidia_gstreamer-1.0 libraries from /usr/lib/arm-linux-gnueabihf/gstreamer-1.0/ to the gst-1.6.0/out/lib/gstreamer-1.0/ directory.
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., --prev_res=3</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., --image_res=3</td>
<td>-</td>
</tr>
<tr>
<td>--video_res</td>
<td>Video width and height, e.g., --video_res=3</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:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>--enc-bitrate=4000000</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>-k, --file_type</td>
<td>Container file type.</td>
<td>0-MP4</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
<td>Available Values</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>--cap-dev-node</td>
<td>Video capture device node.</td>
<td>0=/dev/video0[default] 1=/dev/video1 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 1csi (default) 2-videotest 3-eglstream</td>
</tr>
<tr>
<td>-w, --whitebalance</td>
<td>White balance value for capture. (CSI only)</td>
<td></td>
</tr>
<tr>
<td>-s, --scene-mode</td>
<td>Camera scene-mode value. (CSI only)</td>
<td></td>
</tr>
<tr>
<td>-c, --color-effect</td>
<td>Camera color effect value. (CSI only)</td>
<td></td>
</tr>
<tr>
<td>--auto-exposure</td>
<td>Camera auto-exposure value. (CSI only)</td>
<td></td>
</tr>
<tr>
<td>--flash</td>
<td>Camera flash value. (CSI only)</td>
<td></td>
</tr>
<tr>
<td>--flicker</td>
<td>Camera flicker detection and avoidance mode value. (CSI only)</td>
<td></td>
</tr>
<tr>
<td>--contrast</td>
<td>Camera contrast value. (CSI only)</td>
<td></td>
</tr>
<tr>
<td>--saturation</td>
<td>Camera saturation value. (CSI only)</td>
<td></td>
</tr>
<tr>
<td>--edge-enhancement</td>
<td>Camera edge enhancement value. (CSI only)</td>
<td></td>
</tr>
<tr>
<td>--tnr_strength</td>
<td>Camera TNR strength value. (CSI only)</td>
<td></td>
</tr>
<tr>
<td>--tnr_mode</td>
<td>Camera TNR mode value. (CSI only)</td>
<td></td>
</tr>
<tr>
<td>--sensor-id</td>
<td>Camera Sensor ID value. (CSI only)</td>
<td></td>
</tr>
<tr>
<td>--display-id</td>
<td>Display ID value (for nvoverlaysink 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. (CSI only)</td>
<td>Example: --aeRegion=&quot;30 40 200 200 1.2“</td>
</tr>
<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“</td>
</tr>
<tr>
<td>--fpsRange</td>
<td>FPS range values (low, high) (CSI only)</td>
<td>Example: --fpsRange=&quot;15 30“</td>
</tr>
</tbody>
</table>
--wbGains White Balance (WB) gains values (R, GR, GB, B) in that order. (CSI only)
Example: --wbGains="1.2 1.4 0.8 1.6"

--stitchConfig Stitching Configuration Options
sensor ID and coordinates in (sensor_id, x_pos, y_pos, width, height) order.
Example: --stitchConfig="0, 0, 1920, 1080, 1, 1920, 0, 1920, 1080"

--overlayConfig Overlay Configuration Options
index and coordinates in (index, x_pos, y_pos, width, height) order.
Example: --overlayConfig="0, 0, 1280, 720"

--enable-meta Enables Sensor MetaData reporting if the sensor has the capability to provide the embedded metadata.

--eglConfig EGL window Coordinates (x_pos y_pos) in that order.
Example: --eglConfig="50 100"

--exposure-time Capture exposure time value. (CSI only)
Example: --exposure-time=0.033

<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
- 2160x1560
- 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>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>ce:&lt;value&gt;</td>
<td>Set color effect mode</td>
<td>1-off</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2-mono</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3-negative</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4-solarize</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5-sepia</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6-posterize</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2-on</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3-OnAutoFlash</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4-OnAlwaysFlash</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5-OnFlashRedEye</td>
</tr>
<tr>
<td>gae</td>
<td>Get auto exposure 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>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 flash detection and avoidance mode</td>
<td>0-off, 1-50 Hz, 2-60 Hz, 3-auto</td>
</tr>
<tr>
<td>gfl</td>
<td>Get flash 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</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</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>
<tr>
<td>gwbg</td>
<td>Get WB gains</td>
<td></td>
</tr>
<tr>
<td>ts:&lt;value&gt;</td>
<td>Set TNR strength</td>
<td>0-1, e.g., ts:0.75</td>
</tr>
<tr>
<td>gts</td>
<td>Get TNR strength</td>
<td></td>
</tr>
<tr>
<td>tnr:&lt;value&gt;</td>
<td>Set TNR mode</td>
<td>0-Original, 1-Outdoor-low-light, 2-Outdoor-medium-light, 3-Outdoor-high-light, 4-Indoor-low-light, 5-Indoor-medium-light, 6-Indoor-high-light</td>
</tr>
<tr>
<td>gtnr</td>
<td>Get TNR mode</td>
<td></td>
</tr>
<tr>
<td>j</td>
<td>Capture one image</td>
<td></td>
</tr>
</tbody>
</table>
### jx<delay>
Capture after a delay of <delay>, e.g., jx5000 to capture after a 5-second delay.

### j:<value>
Capture <count> number of images in succession, e.g., j:6 to capture 6 images.

### 1
Start recording video.

### 0
Stop recording video.

### gpcr
Get preview resolution.

### gicr
Get image capture resolution.

### gvcr
Get video capture resolution.

## 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>1</td>
<td>Start recording video</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>Stop recording video</td>
<td></td>
</tr>
<tr>
<td>pcr:&lt;value&gt;</td>
<td>Set preview resolution</td>
<td>0-176x144 1-320x240 2-640x480 3-1280x720</td>
</tr>
<tr>
<td>gpcr</td>
<td>Get preview resolution</td>
<td></td>
</tr>
<tr>
<td>gicr</td>
<td>Get image capture resolution</td>
<td></td>
</tr>
<tr>
<td>gvcr</td>
<td>Get video capture resolution</td>
<td></td>
</tr>
<tr>
<td>br:&lt;value&gt;</td>
<td>Set encoding bit rate (in bytes)</td>
<td>e.g., br:4000000</td>
</tr>
<tr>
<td>gbr</td>
<td>Get encoding bit rate</td>
<td></td>
</tr>
</tbody>
</table>
Runtime video encoder configuration options are described in the following table.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>cdn:&lt;value&gt;</td>
<td>Set capture device node</td>
<td>0-/dev/video0 1-/dev/video1 2-/dev/video2</td>
</tr>
<tr>
<td>gcdn</td>
<td>Get capture device node</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.
Gstreamer-1.0 and gstreamer-0.10 support the following features, respectively:

<table>
<thead>
<tr>
<th>Video Encoder Feature</th>
<th>gst-openmax</th>
<th>gst-omx</th>
</tr>
</thead>
<tbody>
<tr>
<td>H.264 Baseline / Main profile</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>bitrate</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>insert-spspsatidr</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>rc-mode</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>control-rate</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td>iframeinterval</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>qp-range</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>temporal-tradeoff</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>bit-packetization</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>quality-level</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>low-latency</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>slice-header spacing</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>force-IDR</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>profile</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>vvb-size</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>sliceintrarefreshenable</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>sliceintrarefreshinterval</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)
  
  [Link](http://www.logitech.com/en-in/product/hd-pro-webcam-c920)

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

- Rocketfish™ HD Webcam Pro
  
  [Link](http://www.rocketfishproducts.com/products/computer-accessories/RF-HDWEB10.html?supportTab=open)

- Creative Live! Cam Socialize HD 1080
  
  [Link](http://support.creative.com/Products/ProductDetails.aspx?catID=218&CatName=WebCameras&subCatID=231&subCatName=MIDI+Keyboards&prodID=20165&productName=Live%3dCam%3dSocialize%3dHD%3d1080&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:

       ```bash
       sudo gpasswd -a ubuntu plugdev
       ```

       Re-login.

    2. Install tools for the application:

       ```bash
       apt-get install libgstreamer0.10-dev libgstreamer-plugins-base0.10-dev libgtk2.0-dev g++
       ```

    3. Download XIMEA Linux Software Package:

       ```bash
       wget http://www.ximea.com/downloads/recent/XIMEA_Linux_SP.tgz
       ```

       Untar:

       ```bash
       tar xzf XIMEA_Linux_SP.tgz
cd package
       ```
4. Open the install file and replace

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

with

```bash
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 apps:

   - xiSample: run from `/package/bin` folder
   - streamViewer
     - 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 ON-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 NVIDIA Corporation. All rights reserved.