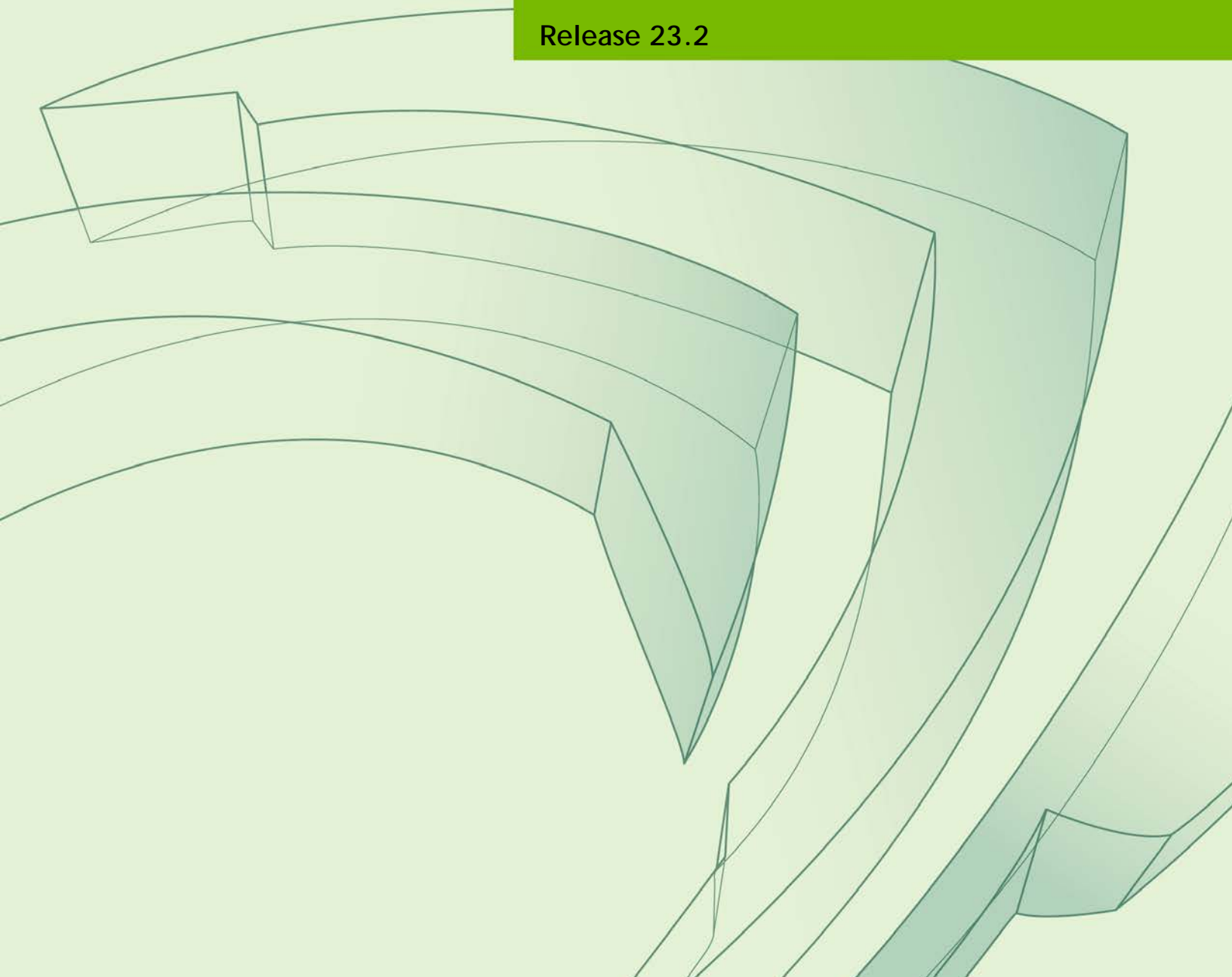




TEGRA X1/TEGRA LINUX DRIVER PACKAGE MULTIMEDIA USER GUIDE

DA_07303_1.5 | January 8, 2016
Advance Information | Subject to Change

Release 23.2



DOCUMENT CHANGE HISTORY

DA_07303_1.4

Version	Date	Authors	Description of Change
v1.0	01 May 2015	mzensius	Initial release.
v1.1	30 Jun 2015	mzensius	Added rotation and scaling commands, other new content.
v1.2	03 Nov 2015	emilyh	Changes for 23.1
v1.3	19 Nov 2015	mzensius	Added note for display export.
v1.4	17 Dec 2015	hlang	Updated gst-nvivafilter sample pipelines. Updated steps to build gstreamer manually.
v1.5	8 Jan 2016	kstone	added nvidiaconv interpolation method.

TABLE OF CONTENTS

Tegra X1/Tegra Linux Driver Package Multimedia User Guide	1
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	5
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	8
Supported H.264 Encoder Features with Gstreamer-0.10	9
Video Encode Examples Using gst-launch-1.0	10
Camera Capture with Gstreamer-0.10	11
Camera Capture with Gstreamer-1.0	11
Video Playback with Gstreamer-0.10	12
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	14
raw-gray Output Formats	14
Video Scaling with Gstreamer-1.0	14
raw-yuv Input Formats	14
raw-gray Input Formats	14
raw-yuv Output Formats	15
raw-gray Output Formats	15
NVIDIA Input and Output Formats	15
Video Transcode with Gstreamer-1.0	16
CUDA Video Post-Processing with GStreamer-1.0	18
gst-videocuda	18
gst-nvivafilter	18
Video Rotation with Gstreamer-1.0	19
Interpolation with GStreamer-1.0	20
GStreamer Build Instructions	22

Nvgstcapture-1.0 Option Reference	25
Nvgstcapture Application Options	25
CSI Camera Supported Resolutions	27
CSI Camera Runtime Commands	27
USB Camera Runtime Commands	30
Notes.....	31
Video Encoder Features	32
Supported Cameras.....	33
CSI Cameras.....	33
USB 2.0 Cameras	33
INDUSTRIAL CAMERA DETAILS	34

TEGRA X1/TEGRA LINUX DRIVER PACKAGE MULTIMEDIA USER GUIDE

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:

Video Decoder	Description
nv_omx_h264dec	OpenMAX IL H.264/AVC video decoder
nv_omx_mpeg4dec	OpenMAX IL MPEG-4 video decoder
nv_omx_vp8dec	OpenMAX IL VP8 video decoder

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

Video Encoders	Description
nv_omx_h264enc	OpenMAX IL H.264/AVC video encoder
nv_omx_vp8enc	OpenMAX IL VP8 video encoder

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

Video Sink	Description
nv_omx_videosink	OpenMAX IL videosink element
nv_omx_hdmi_videosink (deprecated)	OpenMAX IL HDMI videosink element

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:

Video Decoder	Description
omxh265dec	OpenMAX IL H.265 Video Decoder
omxh264dec	OpenMAX IL H.264 Video Decoder
omxmpeg4videodec	OpenMAX IL MPEG4 Video Decoder
omxvp8dec	OpenMAX IL VP8 Video Decoder
omxh263dec	OpenMAX IL H.263 video decoder

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

Video Encoders	Description
omxh264enc	OpenMAX IL H.264/AVC video encoder
omxh265enc	OpenMAX IL H.265/AVC video encoder
omxvp8enc	OpenMAX IL VP8 video encoder

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

Video Sink	Description
nvoverlaysink	OpenMAX IL videosink element
nvhdmioverlaysink (deprecated)	OpenMAX IL HDMI videosink element

Gstreamer version 1.0 includes the following egl image video sinks:

Video Sink	Description
nveglglessink	EGL/GLES videosink element

Gstreamer version 1.0 includes the following proprietary NVIDIA plugins:

Video Sink	Description
nveglstreamsrc	Acts as Gstreamer Source Component, accepts EGLStream from EGLStream producer
nvvideosink	Video Sink Component. Accepts YUV-I420 format and produces EGLStream (RGBA)
nvegltransform	Video transform element for NVMM to EGLImage (supported with nveglglessink only)



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.

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)

```
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)

```
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)

```
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)

```
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)

```
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)

```
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)

```
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)

```
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)

```
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.

```
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)

```
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' ! voamrbenc ! 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)

```
gst-launch-0.10 videotestsrc num-buffers=200 ! 'video/x-raw-yuv,
width=(int)1280, height=(int)720, format=(fourcc)I420' ! nv_omx_h264enc
temporal-tradeoff=1 ! qtmux ! filesink location=test.mp4 -e
```

Set rate control mode

```
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:

```
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

```
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

```
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)

```
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.

USB Camera Support	Feature
YUV	Preview display
	Image capture (VGA, 640 x 480)
	Video capture (480p, 720p, H.264/VP8/VP9 encode)
MJPEG	Preview display
	Image capture VGA, 640 x 480 720p, 1280 x 720
	Video capture (480p, 720p, 1080p, MJPEG encode)

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

```
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:

```
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.

USB Camera Support	Feature
YUV	Preview display
	Image capture (VGA, 640 x 480)
	Video capture (480p, 720p, H.264/VP8 encode)

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-0.10

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

```
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.

```
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.

```
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.

```
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:

Format	Description
NV12	NVIDIA gst-omx decoder output format.
I420	NVIDIA gst-omx encoder input format.

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)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)

```
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)

```
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)

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

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

```
gst-launch-1.0 filesrc location=<filename.mp4> ! qtdemux
name=demux demux.video_0 ! queue ! h264parse ! omxh264dec ! nvvidconv !
avenc_mpeg4 ! qtmux name=mux ! filesink
location=<Transcoded_filename.mp4> demux.audio_0 ! queue ! aacparse !
mux.audio_0 -e
```

VP8 Decode to MPEG-4 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 !
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)

```
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)

```
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
```

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 ! 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.

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="libsampler_process.so" ! 'video/x-raw(memory:NVMM),
format=(string)NV12' ! nvoverlaysink -e
```

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 ! nvivafilter
cuda-process=true customer-lib-name="libsampler_process.so" ! 'video/x-
raw(memory:NVMM), format=(string)NV12' ! nvoverlaysink -e
```

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.

Flip Method	Property value
identity - no rotation (default)	0
counterclockwise - 90 degrees	1
rotate - 180 degrees	2
clockwise - 90 degrees	3
horizontal flip	4
upper right diagonal flip	5
vertical flip	6
upper-left diagonal	7

 **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.

```
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:

```
gst-launch-1.0 filesrc location=<filename.mp4> ! qtdemux name=demux !
h264parse ! omxh264dec ! nvvidconv flip-method=3 !
omxh264enc ! qtmux ! filesink location=test.mp4 -e
```

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
```

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.

Interpolation Method	Property Value
nearest	0
linear	1
smart (default)	2

bilinear	3
----------	---



Note: Get information on `nvidconv interpolation-method` property with the `gst-inspect-1.0 nvidconv` 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 ! nvidconv interpolation-
method=3 ! 'video/x-raw(memory:NVMM), format=(string)I420,
width=1280, height=720' ! nvoverlaysink -e
```

GSTREAMER BUILD INSTRUCTIONS

This section provides a procedure for building current versions of gstreamer.

Using gst-install to build GStreamer

This release contains the `gst-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 libg11-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 the `nvidia gstreamer-1.0` libraries to the `gst_1.6.0` plugin directory using the following command:

```
cd /usr/lib/arm-linux-gnueabi/hf/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
libgstnveglglessink.so
libgstnveglstreamsrc.so
libgstnvegltransform.so
libgstnvivafilter.so
libgstnvvidconv.so
libgstnvvideosink.so
libnvgstjpeg.so
libgstomx.so
```

NVGSTCAPTURE-1.0 OPTION REFERENCE

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.

Application Options		
Option	Description	Notes
<code>--prev_res</code>	Preview area width and height, e.g., <code>--prev_res=3</code>	-
<code>--cus-prev-res</code>	Custom preview width and height for CSI only	-
<code>--image_res</code>	Image width and height, e.g., <code>--image_res=3</code>	-
<code>--video_res</code>	Video width and height, e.g., <code>--video_res=3</code>	-
<code>-m, --mode</code>	Capture mode.	1-Still 2-Video
<code>-v, --video_enc</code>	Video encoder type.	0-H.264 (hardware) 1-VP8(hardware) 2-MPEG-4 (software) 3-H.263 (software)
<code>-b, --enc-bitrate</code>	Video encoding Bit-rate(in bytes)	Example: <code>--enc-bitrate=4000000</code>
<code>--enc-profile</code>	Video encoder profile (only for H.264)	0-Baseline 1-Main 2-High
<code>-j, --image_enc</code>	Image encoder type.	0-jpeg_SW[jpegenc] 1-jpeg_HW[nvjpegenc]
<code>-k, --file_type</code>	Container file type.	0-MP4

		1-3GP 2-AVI
--cap-dev-node	Video capture device node.	0=/dev/video0[default] 1=/dev/video1 2=/dev/video2
--svs	Chain for video preview.	-
--file-name	File name for capture.	"nvcamtest" is used by default.
--camsrc	Camera source.	0-v4l2 1-csi (default) 2-videotest 3-eglstream
-w, --whitebalance	White balance value for capture. (CSI only)	-
-s, --scene-mode	Camera scene-mode value. (CSI only)	-
-c, --color-effect	Camera color effect value. (CSI only)	-
--auto-exposure	Camera auto-exposure value. (CSI only)	-
--flash	Camera flash value. (CSI only)	-
--flicker	Camera flicker detection and avoidance mode value. (CSI only)	-
--contrast	Camera contrast value. (CSI only)	-
--saturation	Camera saturation value. (CSI only)	-
--edge-enhancement	Camera edge enhancement value. (CSI only)	-
--tnr_strength	Camera TNR strength value. (CSI only)	-
--tnr_mode	Camera TNR mode value. (CSI only)	-
--sensor-id	Camera Sensor ID value. (CSI only)	-
--display-id	Display ID value (for nvoverlaysink only)	-
--eglstream-id	Select EGLStreamProducerID value (for CSI EGLStream). Default is 0.	-
--aeRegion	ROI for AE coordinates (top, left, bottom, right) and weight, in that order. (CSI only)	Example: --aeRegion="30 40 200 200 1.2"
--wbRegion	ROI for AWB coordinates (top, left, bottom, right) and weight in that order. (CSI only)	Example: --wbRegion="30 40 200 200 1.2"
--fpsRange	FPS range values (low, high) (CSI only)	Example: --fpsRange="15 30"

--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, 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
Help Options		
Option	Description	Notes
-h, --help	Show help options.	-
--help-all	Show all help options.	-
--help-gst	Show Gstreamer options.	-

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.

Command	Description	Notes
h	Help	-
q	Quit	-
mo:<value>	Set capture mode	1-image 2-video
gmo	Get capture mode	-
sid:<value>	Set sensor ID	-
gsid	Get sensor ID	-
wb:<value>	Set white balance mode	0-off 1-auto 2-incandescent 3-fluorescent 4-warm-fluorescent 5-daylight 6-cloudy-daylight 7-twilight 8-shade
gwb	Get white balance mode	-
scm:<value>	Set scene mode	0-face-priority 1-action 2-portrait 3-landscape 4-night 5-night-portrait 6-theatre 7-beach 8-snow 9-sunset 10-steady-photo 11-fireworks 12-sports 13-party 14-candle-light 15-barcode
gcm	Get scene mode	-
ce:<value>	Set color effect mode	1-off 2-mono 3-negative 4-solarize 5-sepia 6-posterize 7-aqua
gce	Get color effect mode	-
ae:<value>	Set auto-exposure mode	1-off 2-on 3-OnAutoFlash 4-OnAlwaysFlash 5-OnFlashRedEye
gae	Get auto exposure mode	-

f:<value>	Set flash mode	0-off 1-on 2-torch 3-auto
gf	Get flash mode	-
fl:<value>	Set flash detection and avoidance mode	0-off 1-50 Hz 2-60 Hz 3-auto
gfl	Get flash detection and avoidance mode	-
ct:<value>	Set contrast	0-1, e.g., ct:0.75
gct	Get contrast	-
st:<value>	Set saturation	0-2, e.g., st:1.25
gst	Get saturation	-
ext:<value>	Set exposure time (in seconds)	e.g., ext:0.033
gext	Get exposure time	-
ee:<value>	Set edge enhancement	0-1, e.g., ee:0.75
gee	Get edge enhancement	-
aer:<value>	Set ROI coordinates for AE (top, left, bottom, right) and weight	e.g., aer:20 20 400 400 1.2
gaer	Get ROI for AE	-
wbr:<value>	Set ROI coordinates for AWB (top, left, bottom, right) and weight	e.g., wbr:20 20 400 400 1.2
gwbr	Get ROI for AE	-
fpsr:<value>	Set FPS range (low, high)	e.g., fpsr:15 30
gfpsr	Get FPS range	-
wbg:<value>	Set WB gains (R, GR, GB, B)	e.g., wbg:1.2 2.2 0.8 1.6
gwbg	Get WB gains	-
ts:<value>	Set TNR strength	0-1, e.g., ts:0.75
gts	Get TNR strength	-
tnr:<value>	Set TNR mode	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
gtnr	Get TNR mode	-
j	Capture one image.	-

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.

Command	Description	Notes
h	Help	-
q	Quit	-
mo:<value>	Set capture mode	1-image 2-video
gmo	Get capture mode	-
j	Capture one image.	-
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	-
pcr:<value>	Set preview resolution	0-176x144 1-320x240 2-640x480 3-1280x720
gpcr	Get preview resolution	-
gicr	Get image capture resolution	-
gvcr	Get video capture resolution	-
br:<value>	Set encoding bit rate (in bytes)	e.g., br:4000000
gbr	Get encoding bit rate	-

cdn:<value>	Set capture device node	0-/dev/video0 1-/dev/video1 2-/dev/video2
gcdn	Get capture device node	-

Runtime video encoder configuration options are described in the following table.

Command	Description	Notes
br:<val>	Sets encoding bit-rate (in bytes)	Example: br:4000000
gbr	Gets encoding bit-rate (in bytes)	-
ep:<val>	Sets encoding profile (for H.264 only)	Example: ep:1 (0): Baseline (1): Main (2): High
gep	Gets encoding profile (for H.264 only)	-
Enter+f	Forces IDR frame on video encoder (for H.264 only)	-

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.

VIDEO ENCODER FEATURES

Gstreamer-1.0 and gstreamer-0.10 support the following features, respectively:

Video Encoder Feature	gst-openmax (gstreamer-0.10)	gst-omx (nvgstcapture-1.0)
H.264 Baseline / Main profile	✓	✓
bitrate	✓	✓
insert-spsppsatidr	✓	✓
rc-mode	✓	-
control-rate	-	✓
iframeinterval	✓	✓
qp-range	✓	
temporal-tradeoff	✓	✓
bit-packetization	✓	✓
quality-level	✓	✓
low-latency	✓	✓
slice-header spacing	✓	✓
force-IDR	✓	✓
profile		✓
vbv-size		✓
sliceintrarefreshenable		✓
sliceintrarefreshinterval		✓

SUPPORTED CAMERAS

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)
<http://www.logitech.com/en-in/product/hd-pro-webcam-c920>
- ▶ Logitech c910
<http://www.amazon.com/Logitech-HD-Pro-Webcam-C910/dp/B003M2YT96>
- ▶ Rocketfish™ HD Webcam Pro
<http://www.rocketfishproducts.com/products/computer-accessories/RF-HDWEB10.html?supportTab=open>
- ▶ Creative Live! Cam Socialize HD 1080
http://support.creative.com/Products/ProductDetails.aspx?catID=218&CatName=Web+Cameras&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

<http://www.e-consystems.com/UltraHD-USB-Camera.asp>

- 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

<http://www.ximea.com/en/products/usb3-vision-cameras-xiq-line/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

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

with

```
elif [ "$arch" == "aarch64" ]
```

5. Start installation:

```
./install
```

Install USB3 camera:

```
./install -cam_usb30
```

Install graphical desktop:

```
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, 2016 NVIDIA Corporation. All rights reserved.