



JETSON TK1/TEGRA LINUX DRIVER PACKAGE MULTIMEDIA USER GUIDE

DA_07303-2.5 | October 6, 2016
Advance Information | Subject to Change

Release R21.5



DOCUMENT CHANGE HISTORY

DA_07303-2.5

Version	Date	Authors	Description of Change
v1.0	10 June 2014	mzensius	Initial release.
v1.1	25 June 2014	mzensius	Corrections to Video Format conversions.
v1.2	8 July 2014	mzensius	Converted to non-confidential document.
v1.3	10 Dec 2014	mzensius	Added H.264 encoder features.
v2.0	13 Jan 2015	mzensius	Added Gstreamer-1.0 information.
v2.1	26 Feb 2015	mzensius	Added further usage information.
v2.2	30 Jun 2015	mzensius	Added Gstreamer option reference, and video encoder feature listing.
v2.3	03 Dec 2015	kstone	Added nvvidconv video-rotation and interpolation methods.
v2.4	06 Jul 2016	mzensius	Version and date updated for R21.5 release. No change to content.
v2.5	06 Oct 2016	mzensius	Minor updates to video encoder features, also adds Gstreamer 0.10 deprecation notice.

TABLE OF CONTENTS

Jetson TK1/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	4
Video Decode Examples Using gst-launch-0.10	5
Video Decode Examples Using gst-launch-1.0	6
Encode Examples	6
Audio Encode Examples Using gst-launch-0.10	6
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
Camera Capture with Gstreamer-1.0	10
Video Playback with Gstreamer-0.10	11
Video Playback with Gstreamer-1.0	12
Video Format Conversion with Gstreamer-0.10	12
raw-yuv Input Formats	13
raw-gray Input Formats	13
raw-yuv Output Formats	13
raw-gray Output Formats	14
RGB Output Formats	14
Video Format Conversion with Gstreamer-1.0	14
raw-yuv Input Formats	15
raw-gray Input Formats	15
raw-gray Output Formats	15
Video Scaling with Gstreamer-0.10	15
raw-yuv Input Formats	15
raw-gray Input Formats	16
raw-yuv Output Formats	16
raw-gray Output Formats	17
RGB Output Formats	17
NVIDIA Input and Output Formats	18
Video Scaling with Gstreamer-1.0	18
raw-yuv Input Formats	19
raw-gray Input Formats	19

raw-yuv Output Formats	19
raw-gray Output Formats	19
NVIDIA Input and Output Formats	20
Video Transcode with Gstreamer-0.10	20
Video Transcode with Gstreamer-1.0.....	22
Video Rotation with Gstreamer-1.0	23
Interpolation Methods for Video Scaling	25
Nvgstcapture-1.0 Option Reference	27
Nvgstcapture Application Options	27
CSI Camera Runtime Commands.....	28
USB Camera Runtime Commands.....	30
Video Encoder Features	32

JETSON TK1/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 the Jetson TK1 platform.

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-0.10](#)
- ▶ [Video Format Conversion with Gstreamer-1.0](#)
- ▶ [Video Scaling with Gstreamer-0.10](#)
- ▶ [Video Scaling with Gstreamer-1.0](#)
- ▶ [Video Transcode with Gstreamer-0.10](#)
- ▶ [Video Transcode with Gstreamer-1.0](#)
- ▶ [Video Rotation with Gstreamer-1.0](#)
- ▶ [Nvgstcapre-1.0 Option Reference](#)
- ▶ [Video Encoder Features](#)

GSTREAMER-0.10 INSTALLATION AND SETUP

This section describes how to install and configure Gstreamer.



Note: Gstreamer version 0.10 support is deprecated in Linux for Tegra (L4T) Release 21.5. Use of Gstreamer version 1.0 is recommended for development.

To install Gstreamer-0.10

- Install Gstreamer-0.10 on the Jetson TK1 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) R21.5 release package for Jetson TK1.

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
nv_omx_h263dec	OpenMAX IL H.263 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	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 Jetson TK1 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
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
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	OpenMAX IL HDMI videosink element

Gstreamer version 1.0 includes the following EGL image video sinks:

Video Sink	Description
------------	-------------

nveglglessink

EGL/GLES videosink element

DECODE EXAMPLES

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



Note: Gstreamer version 0.10 support is deprecated in Linux for Tegra (L4T) Release 21.5. Use of Gstreamer version 1.0 is recommended for development.

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_hdm1_videosink -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_hdm1_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_hdm1_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
```

VP8 Decode (NVIDIA accelerated decode)

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



Note: Gstreamer version 0.10 support is deprecated in Linux for Tegra (L4T) Release 21.5. Use of Gstreamer version 1.0 is recommended for development.

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' ! voamrbenc ! 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
```

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 R21.5 release is `nvgstcapture-0.10`.



Note: Gstreamer version 0.10 support is deprecated in Linux for Tegra (L4T) Release 21.5. Use of Gstreamer version 1.0 is recommended for development.

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 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 R21.5 release is nvgstplayer-0.10.



Note: Gstreamer version 0.10 support is deprecated in Linux for Tegra (L4T) Release 21.5. Use of Gstreamer version 1.0 is recommended for development.

For usage information enter the following command:

```
$ nvgstplayer-0.10 --help
```

Video can be output to HD displays using the HDMI connector on the Jetson TK1 platform. The Gstreamer-0.10 application supports currently the following video sinks:

HDMI 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_hdmi_videosink -v -e
```

HDMI 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_hdmi_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 Jetson TK1 platform. The Gstreamer-1.0 application supports currently the following video sinks:

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

```
$ gst-launch-1.0 filesrc location=<filename.mp4> ! qtdemux name=demux !
h264parse ! omxh264dec ! nvhdmioverlaysink -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-0.10

The NVIDIA proprietary nvvidconv Gstreamer-0.10 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.



Note: Gstreamer version 0.10 support is deprecated in Linux for Tegra (L4T) Release 21.5. Use of Gstreamer version 1.0 is recommended for development.

raw-yuv Input Formats

Currently `nvvidconv` supports the following raw-yuv input formats: I420, YV12, YUY2, UYVY, YVYU, Y444, and NV12.

Converting raw-yuv to nv-yuv

```
$ gst-launch-0.10 videotestsrc ! 'video/x-raw-yuv, width=(int)1280, height=(int)720, format=(fourcc)YUY2' ! nvvidconv ! 'video/x-nv-yuv' ! nv_omx_h264enc ! qtmux ! filesink location=test.mp4 -e
```

Converting raw-yuv to nvrn-yuv

```
$ gst-launch-0.10 videotestsrc ! 'video/x-raw-yuv, width=(int)1280, height=(int)720, format=(fourcc)YUY2' ! nvvidconv ! 'video/x-nv-yuv' ! nv_omx_h264enc ! qtmux ! filesink location=test.mp4 -e
```

raw-gray Input Formats

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

Converting raw-gray to nv-yuv

```
$ gst-launch-0.10 videotestsrc num-buffers=300 ! 'video/x-raw-gray, bpp=(int)8, depth=(int)8, width=(int)640, height=(int)480, framerate=(fraction)30/1' ! nvvidconv ! 'video/x-nv-yuv, format=(fourcc)I420' ! nv_omx_h264enc ! qtmux ! filesink location=test.mp4 -e
```

Converting raw-gray to nvrn-yuv

```
$ gst-launch-0.10 videotestsrc num-buffers=300 ! 'video/x-raw-gray, bpp=(int)8, depth=(int)8, width=(int)640, height=(int)480, framerate=(fraction)30/1' ! nvvidconv ! 'video/x-nvrn-yuv, format=(fourcc)I420' ! nv_omx_h264enc ! qtmux ! filesink location=test.mp4 -e
```

raw-yuv Output Formats

Currently `nvvidconv` supports the following raw-yuv output formats: I420, YUY2, UYVY, and YVYU.

Converting nv-yuv to raw-yuv

```
$ gst-launch-0.10 filesrc location=640x480_30p.mp4 ! qtdemux name=demux ! nv_omx_h264dec ! 'video/x-nv-yuv' ! nvvidconv ! xvimagesink -e
```

Converting nvrm-yuv to raw-yuv

```
$ gst-launch-0.10 filesrc location=640x480_30p.mp4 ! qtdemux name=demux
! nv_omx_h264dec ! 'video/x-nvrm-yuv' ! nvvidconv ! 'video/x-raw-yuv,
format=(fourcc)UYVY' ! xvimagesink -e
```

raw-gray Output Formats

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

Converting nv-yuv to raw-gray

```
$ gst-launch-0.10 filesrc location=640x480_30p.mp4 ! qtdemux name=demux
! nv_omx_h264dec ! 'video/x-nv-yuv' ! nvvidconv ! 'video/x-raw-gray' !
ffmpegcolorspace ! xvimagesink -e
```

Converting nvrm-yuv to raw-gray

```
$ gst-launch-0.10 filesrc location=640x480_30p.mp4 ! qtdemux name=demux
! nv_omx_h264dec ! 'video/x-nvrm-yuv' ! nvvidconv ! 'video/x-raw-gray'
! ffmpegcolorspace ! xvimagesink -e
```

RGB Output Formats

Currently `nvvidconv` supports the following RGB output formats: BGRA, RGBA, BGRx, and RGBx.

Converting nv-yuv to raw-rgb

```
$ gst-launch-0.10 filesrc location=640x480_30p.mp4! qtdemux name=mux !
nv_omx_h264dec ! 'video/x-nv-yuv' ! nvvidconv ! ximagesink -e
```

Converting nvrm-yuv to raw-rgb

```
$ gst-launch-0.10 filesrc location=640x480_30p.mp4! qtdemux name=mux !
nv_omx_h264dec ! 'video/x-nvrm-yuv' ! nvvidconv ! ximagesink -e
```

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-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-0.10

The NVIDIA proprietary `nvvidconv` Gstreamer-0.10 plug-in also allows you to perform video scaling. The `nvvidconv` plug-in currently supports scaling with the format conversions described in this section.



Note: Gstreamer version 0.10 support is deprecated in Linux for Tegra (L4T) Release 21.5. Use of Gstreamer version 1.0 is recommended for development.

raw-yuv Input Formats

Currently `nvvidconv` supports the following raw-yuv input formats for scaling: I420, YUY2, UYVY, YVYU, Y444, and NV12.

Converting raw-yuv to nv-yuv with scaling

```
$ gst-launch-0.10 videotestsrc ! 'video/x-raw-yuv, width=(int)1280, height=(int)720, format=(fourcc)I420' ! nvvidconv ! 'video/x-nv-yuv, width=(int)640, height=(int)480' ! nv_omx_h264enc ! qtmux ! filesink location=test.mp4 -e
```

Converting raw-yuv to nvrn-yuv with scaling

```
$ gst-launch-0.10 videotestsrc ! 'video/x-raw-yuv, width=(int)1280, height=(int)720, format=(fourcc)NV12' ! nvvidconv ! 'video/x-nvrn-yuv, width=(int)640, height=(int)480' ! nv_omx_h264enc ! qtmux ! filesink location=test.mp4 -e
```

raw-gray Input Formats

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

Converting raw-gray to nv-yuv with scaling

```
$ gst-launch-0.10 videotestsrc num-buffers=300 ! 'video/x-raw-gray, bpp=(int)8, depth=(int)8, width=(int)1280, height=(int)720, framerate=(fraction)30/1' ! nvvidconv ! 'video/x-nv-yuv, width=(int)640, height=(int)480, format=(fourcc)I420' ! nv_omx_h264enc ! qtmux ! filesink location=test.mp4 -e
```

Converting raw-gray to nvrn-yuv with scaling

```
$ gst-launch-0.10 videotestsrc num-buffers=300 ! 'video/x-raw-gray, bpp=(int)8, depth=(int)8, width=(int)1920, height=(int)1080, framerate=(fraction)30/1' ! nvvidconv ! 'video/x-nvrn-yuv, width=(int)640, height=(int)480, format=(fourcc)I420' ! nv_omx_h264enc ! qtmux ! filesink location=test.mp4 -e
```

raw-yuv Output Formats

Currently nvvidconv supports the following raw-yuv output formats for scaling: I420, YUY2, UYVY, and YVYU.

Converting nv-yuv to raw-yuv with scaling

```
$ gst-launch-0.10 filesrc location=1280x720_30p.mp4 ! qtdemux name=demux ! nv_omx_h264dec ! 'video/x-nv-yuv' ! nvvidconv ! 'video/x-raw-yuv, width=(int)640, height=(int)480, format=(fourcc)YUY2' ! xvimagesink -e
```

Converting nvrm-yuv to raw-yuv with scaling

```
$ gst-launch-0.10 filesrc location=1280x720_30p.mp4 ! qtdemux
name=demux ! nv_omx_h264dec ! 'video/x-nvrm-yuv' ! nvvidconv !
'video/x-raw-yuv, width=(int)640, height=(int)480, format=(fourcc)UYVY'
! xvimagesink -e
```

raw-gray Output Formats

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

Converting nv-yuv to raw-gray with scaling

```
$ gst-launch-0.10 filesrc location=1280x720_30p.mp4 ! qtdemux
name=demux ! nv_omx_h264dec ! 'video/x-nv-yuv' ! nvvidconv ! 'video/x-
raw-gray, bpp=(int)8, depth=(int)8, width=(int)320, height=(int)240' !
ffmpegcolospace ! xvimagesink -e
```

Converting nvrm-yuv to raw-gray

```
$ gst-launch-0.10 filesrc location=1280x720_30p.mp4 ! qtdemux
name=demux ! nv_omx_h264dec ! 'video/x-nvrm-yuv' ! nvvidconv !
'video/x-raw-gray, bpp=(int)8, depth=(int)8, width=(int)640,
height=(int)480' ! ffmpegcolospace ! xvimagesink -e
```

RGB Output Formats

Currently `nvvidconv` supports the following RGB output formats for scaling: BGRA, RGBA, BGRx, and RGBx.

Converting nv-yuv to raw-rgb with scaling

```
$ gst-launch-0.10 filesrc location=1280x720_30p.mp4! qtdemux name=mux !
nv_omx_h264dec ! 'video/x-nv-yuv' ! nvvidconv ! 'video/x-raw-rgb,
width=(int)640, height=(int)480' ! ximagesink -e
```

Converting nvrm-yuv to raw-rgb

```
$ gst-launch-0.10 filesrc location=1280x720_30p.mp4! qtdemux name=mux !
nv_omx_h264dec ! 'video/x-nvrm-yuv' ! nvvidconv ! 'video/x-raw-rgb,
width=(int)640, height=(int)480' ! ximagesink -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 <code>gst-openmax</code> decoder output format.
I420	NVIDIA <code>gst-openmax</code> encoder input format.

Scaling nv-yuv

```
$ gst-launch-0.10 filesrc location=1280x720_30p.mp4 ! qtdemux name=mux
! nv_omx_h264dec ! 'video/x-nv-yuv' ! nvvidconv ! 'video/x-nv-yuv,
width=640, height=480' ! nv_omx_h264enc ! qtmux ! filesink
location=test.mp4 -e
```

Converting nv-yuv to nvrn-yuv with scaling

```
$ gst-launch-0.10 filesrc location=1280x720_30p.mp4 ! qtdemux name=mux
! nv_omx_h264dec ! 'video/x-nv-yuv' ! nvvidconv ! 'video/x-nvrn-yuv,
width=640, height=480' ! nv_omx_h264enc ! qtmux ! filesink
location=test.mp4 -e
```

Scaling nvrn-yuv

```
$ gst-launch-0.10 filesrc location=1280x720_30p.mp4 ! qtdemux name=mux
! nv_omx_h264dec ! 'video/x-nvrn-yuv' ! nvvidconv ! 'video/x-nvrn-yuv,
width=640, height=480' ! nv_omx_h264enc ! qtmux ! filesink
location=test.mp4 -e
```

Converting nvrn-yuv to nv-yuv with scaling

```
$ gst-launch-0.10 filesrc location=1280x720_30p.mp4 ! qtdemux name=mux
! nv_omx_h264dec ! 'video/x-nvrn-yuv' ! nvvidconv ! 'video/x-nv-yuv,
width=640, height=480' ! nv_omx_h264enc ! qtmux ! filesink
location=test.mp4 -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.

Scaling between nv 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' !
nvhdmioverlaysink -e
```

VIDEO TRANSCODE WITH GSTREAMER-0.10

You can perform video transcoding between video formats as described in this section.



Note: Gstreamer version 0.10 support is deprecated in Linux for Tegra (L4T) Release 21.5. Use of Gstreamer version 1.0 is recommended for development.

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

```
$ gst-launch-0.10 filesrc location=<filename.mp4> ! qtdemux name=demux
demux.video_00 ! queue ! nv_omx_h264dec ! nv_omx_vp8enc ! qtmux
name=mux ! filesink location=<Transcoded_filename.mp4> demux.audio_00 !
queue ! aacparse ! mux.audio_00 -e
```

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

```
$ gst-launch-0.10 filesrc location=<filename.mp4> ! qtdemux name=demux
demux.video_00 ! queue ! nv_omx_vp8dec ! nv_omx_h264enc ! qtmux
name=mux ! filesink location=<Transcoded_filename.mp4> demux.audio_00 !
queue ! aacparse ! mux.audio_00 -e
```


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

```
$ gst-launch-0.10 filesrc location=<filename.mp4> ! qtdemux name=demux
demux.video_00 ! queue ! nv_omx_mpeg4dec ! nv_omx_vp8enc ! qtmux
name=mux ! filesink location=<Transcoded_filename.mp4> demux.audio_00 !
queue ! aacparse ! mux.audio_00 -e
```

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

```
$ gst-launch-0.10 filesrc location=<filename.mp4> ! qtdemux name=demux
demux.video_00 ! queue ! nv_omx_mpeg4dec ! nv_omx_h264enc ! qtmux
name=mux ! filesink location=<Transcoded_filename.mp4> demux.audio_00 !
queue ! aacparse ! mux.audio_00 -v -e
```

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

```
$ gst-launch-0.10 filesrc location=<filename.mp4> ! qtdemux name=demux
demux.video_00 ! queue ! nv_omx_h264dec ! ffenc_mpeg4 ! qtmux
name=mux ! filesink location=<Transcoded_filename.mp4> demux.audio_00 !
queue ! aacparse ! mux.audio_00 -e
```

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

```
$ gst-launch-0.10 filesrc location=<filename.mp4> ! qtdemux name=demux
demux.video_00 ! queue ! nv_omx_vp8dec ! ffenc_mpeg4 ! qtmux
name=mux ! filesink location=<Transcoded_filename.mp4> demux.audio_00 !
queue ! aacparse ! mux.audio_00 -e
```

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

```
$ gst-launch-0.10 filesrc location=<filename.mp4> ! qtdemux name=demux
demux.video_00 ! queue ! nv_omx_h264dec ! theoraenc ! oggmux
name=mux ! filesink location=<Transcoded_filename.ogg> demux.audio_00 !
queue ! faad ! audioconvert ! vorbisenc ! mux. -e
```

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

```
$ gst-launch-0.10 filesrc location=<filename.mp4> ! qtdemux name=demux
demux.video_00 ! queue ! nv_omx_vp8dec ! theoraenc ! oggmux
name=mux ! filesink location=<Transcoded_filename.ogg> demux.audio_00 !
queue ! faad ! audioconvert ! vorbisenc ! mux. -e
```

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

```
$ gst-launch-0.10 filesrc location=<filename.mp4> ! qtdemux name=demux
demux.video_00 ! queue ! nv_omx_mpeg4dec ! theoraenc ! oggmux
name=mux ! filesink location=<Transcoded_filename.ogg> demux.audio_00 !
queue ! faad ! audioconvert ! vorbisenc ! mux. -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
```

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	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' ! nvhdmioverlaysink -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 ! xvimagesink -e
```

To rotate 180 degrees

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

```
$ gst-launch-1.0 filesrc location=<filename.mp4> ! qtdemux name=demux !
h264parse ! omxh264dec ! nvvidconv flip-method=2 ! omxh264enc ! qtmux !
filesink location=test.mp4 -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" !
nvhdmioverlaysink -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" !
nvhdmioverlaysink -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 `nvvidconv interpolation-method` property with the `gst-inspect-1.0 nvvidconv` command.

To use bilinear interpolation method for scaling

- Enter the following command:

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

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
--prev_res	Preview area width and height, e.g., --prev_res=3	-
--image_res	Image width and height, e.g., --image_res=3	-
--video_res	Video width and height, e.g., --video_res=3	-
-m, --mode	Capture mode.	1-Still 2-Video
-v, --video_enc	Video encoder type.	0-H.264 (hardware) 1-VP8(hardware) 2-MPEG-4 (software) 3-H.263 (software)
-j, --image_enc	Image encoder type.	0-jpeg_SW[jpegenc] 1-jpeg_HW[nvjpegenc]
-k, --file_type	Container file type.	0-MP4 1-3GP 2-AVI
--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

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

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	-
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	
ee:<value>	Set edge enhancement	0-1, e.g., ee:0.75
gee	Get edge enhancement	-
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	-

VIDEO ENCODER FEATURES

The Gstreamer-1.0 and gstreamer-0.10-based gst-omx video encoders support the following features, respectively:

Video Encoder Feature	H264enc	Vp8enc
profile (Baseline / Main / High)	✓ (all)	✓
bitrate	✓	✓
insert-spsppsatidr	✓	✓
control-rate	✓	✓
iframeinterval	✓	✓
qp-range	✓	✓
temporal-tradeoff	✓	✓
bit-packetization	✓	✓
quality-level	✓	✓
low-latency	✓	✓
slice-header spacing	✓	-
force-IDR	✓	✓
vbv-size	✓	✓
sliceintrarefreshenable	✓	-
sliceintrarefreshinterval	✓	-
EnableTwoPassCBR	✓	✓
num-B-Frames	✓	-

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.