<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Authors</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>v1.0</td>
<td>10 June 2014</td>
<td>mzensus</td>
<td>Initial release.</td>
</tr>
<tr>
<td>v1.1</td>
<td>25 June 2014</td>
<td>mzensus</td>
<td>Corrections to Video Format conversion.</td>
</tr>
<tr>
<td>v1.2</td>
<td>8 July 2014</td>
<td>mzensus</td>
<td>Converted to non-confidential document.</td>
</tr>
<tr>
<td>v1.3</td>
<td>10 Dec 2014</td>
<td>mzensus</td>
<td>Added H.264 encoder features.</td>
</tr>
<tr>
<td>v2.0</td>
<td>13 Jan 2015</td>
<td>mzensus</td>
<td>Added Gstreamer-1.0 information.</td>
</tr>
<tr>
<td>v2.1</td>
<td>26 Feb 2015</td>
<td>mzensus</td>
<td>Added further usage information.</td>
</tr>
<tr>
<td>v2.2</td>
<td>30 Jun 2015</td>
<td>mzensus</td>
<td>Added Gstreamer option reference, and video encoder feature listing.</td>
</tr>
<tr>
<td>v2.3</td>
<td>03 Dec 2015</td>
<td>kstone</td>
<td>Added nvvidconv video-rotation and interpolation methods.</td>
</tr>
<tr>
<td>v2.4</td>
<td>06 Jul 2016</td>
<td>mzensus</td>
<td>Version and date updated for R21.5 release. No change to content.</td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS

Jetson TK1/Tegra Linux Driver Package Multimedia User Guide ........ 1

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gstreamer-0.10 Installation and Setup</td>
<td>2</td>
</tr>
<tr>
<td>Gstreamer-1.0 Installation and Setup</td>
<td>3</td>
</tr>
<tr>
<td>Decode Examples</td>
<td>4</td>
</tr>
<tr>
<td>Audio Decode Examples Using gst-launch-0.10</td>
<td>4</td>
</tr>
<tr>
<td>Audio Decode Examples Using gst-launch-1.0</td>
<td>4</td>
</tr>
<tr>
<td>Video Decode Examples Using gst-launch-0.10</td>
<td>5</td>
</tr>
<tr>
<td>Video Decode Examples Using gst-launch-1.0</td>
<td>5</td>
</tr>
<tr>
<td>Encode Examples</td>
<td>6</td>
</tr>
<tr>
<td>Audio Encode Examples Using gst-launch-0.10</td>
<td>6</td>
</tr>
<tr>
<td>Audio Encode Examples Using gst-launch-1.0</td>
<td>6</td>
</tr>
<tr>
<td>Video Encode Examples Using gst-launch-0.10</td>
<td>7</td>
</tr>
<tr>
<td>Supported H.264 Encoder Features with Gstreamer-0.10</td>
<td>8</td>
</tr>
<tr>
<td>Video Encode Examples Using gst-launch-1.0</td>
<td>9</td>
</tr>
<tr>
<td>Camera Capture with Gstreamer-0.10</td>
<td>9</td>
</tr>
<tr>
<td>Camera Capture with Gstreamer-1.0</td>
<td>10</td>
</tr>
<tr>
<td>Video Playback with Gstreamer-0.10</td>
<td>11</td>
</tr>
<tr>
<td>Video Playback with Gstreamer-1.0</td>
<td>11</td>
</tr>
<tr>
<td>Video Format Conversion with Gstreamer-0.10</td>
<td>12</td>
</tr>
<tr>
<td>raw-yuv Input Formats</td>
<td>12</td>
</tr>
<tr>
<td>raw-gray Input Formats</td>
<td>12</td>
</tr>
<tr>
<td>raw-yuv Output Formats</td>
<td>13</td>
</tr>
<tr>
<td>raw-gray Output Formats</td>
<td>13</td>
</tr>
<tr>
<td>RGB Output Formats</td>
<td>14</td>
</tr>
<tr>
<td>Video Format Conversion with Gstreamer-1.0</td>
<td>14</td>
</tr>
<tr>
<td>raw-yuv Input Formats</td>
<td>14</td>
</tr>
<tr>
<td>raw-gray Input Formats</td>
<td>14</td>
</tr>
<tr>
<td>raw-gray Output Formats</td>
<td>15</td>
</tr>
<tr>
<td>Video Scaling with Gstreamer-0.10</td>
<td>15</td>
</tr>
<tr>
<td>raw-yuv Input Formats</td>
<td>15</td>
</tr>
<tr>
<td>raw-gray Input Formats</td>
<td>15</td>
</tr>
<tr>
<td>raw-yuv Output Formats</td>
<td>16</td>
</tr>
<tr>
<td>raw-gray Output Formats</td>
<td>16</td>
</tr>
<tr>
<td>RGB Output Formats</td>
<td>17</td>
</tr>
<tr>
<td>NVIDIA Input and Output Formats</td>
<td>17</td>
</tr>
<tr>
<td>Video Scaling with Gstreamer-1.0</td>
<td>18</td>
</tr>
<tr>
<td>raw-yuv Input Formats</td>
<td>18</td>
</tr>
<tr>
<td>raw-gray Input Formats</td>
<td>18</td>
</tr>
</tbody>
</table>
raw-yuv Output Formats ................................................................. 18
raw-gray Output Formats ............................................................. 19
NVIDIA Input and Output Formats ................................................ 19
Video Transcode with Gstreamer-0.10 ......................................... 19
Video Transcode with Gstreamer-1.0 ........................................... 21
Video Rotation with Gstreamer-1.0 .............................................. 23
Interpolation Methods for Video Scaling .................................... 24

Nvgstcapture-1.0 Option Reference ............................................. 26
Nvgstcapture Application Options .............................................. 26
CSI Camera Runtime Commands ................................................ 27
USB Camera Runtime Commands .............................................. 29

Video Encoder Features ............................................................... 31
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
- Nvgstcaptre-1.0 Option Reference
- Video Encoder Features
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 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) R21.4 release package for Jetson TK1.

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

<table>
<thead>
<tr>
<th>Video Decoder</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nv_omx_h264dec</td>
<td>OpenMAX IL H.264/AVC video decoder</td>
</tr>
<tr>
<td>nv_omx_mpeg4dec</td>
<td>OpenMAX IL MPEG-4 video decoder</td>
</tr>
<tr>
<td>nv_omx_vp8dec</td>
<td>OpenMAX IL VP8 video decoder</td>
</tr>
<tr>
<td>nv_omx_h263dec</td>
<td>OpenMAX IL H.263 video decoder</td>
</tr>
</tbody>
</table>

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

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

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

<table>
<thead>
<tr>
<th>Video Sink</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nv_omx_videosink</td>
<td>OpenMAX IL videosink element</td>
</tr>
<tr>
<td>nv_omx_hdmi_videosink</td>
<td>OpenMAX IL HDMI videosink element</td>
</tr>
</tbody>
</table>
GSTREAMER-1.0 INSTALLATION AND SETUP

This section describes how to install and configure Gstreamer.

To install Gstreamer-1.0

- Install Gstreamer-1.0 on the 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:

<table>
<thead>
<tr>
<th>Video Decoder</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>omxh264dec</td>
<td>OpenMAX IL H.264 Video Decoder</td>
</tr>
<tr>
<td>omxmpeg4videodec</td>
<td>OpenMAX IL MPEG4 Video Decoder</td>
</tr>
<tr>
<td>omxvp8dec</td>
<td>OpenMAX IL VP8 Video Decoder</td>
</tr>
<tr>
<td>omxh263dec</td>
<td>OpenMAX IL H.263 video decoder</td>
</tr>
</tbody>
</table>

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

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

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

<table>
<thead>
<tr>
<th>Video Sink</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>noverlaysink</td>
<td>OpenMAX IL videosink element</td>
</tr>
<tr>
<td>nvhdmioverviewsink</td>
<td>OpenMAX IL HDMI videosink element</td>
</tr>
</tbody>
</table>

Gstreamer version 1.0 includes the following EGL image video sinks:

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

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_hdmi_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_hdmi_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)
ENCODE EXAMPLES

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

Audio Encode Examples Using gst-launch-0.10

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

AAC Encode (OSS software encode)

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

AMR-WB Encode (OSS software encode)

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

Audio Encode Examples Using gst-launch-1.0

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

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

AMR-WB Encode (OSS software encode)

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

Video Encode Examples Using gst-launch-0.10

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

H.264 Encode (NVIDIA accelerated encode)

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

VP8 Encode (NVIDIA accelerated encode)

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

MPEG-4 Encode (OSS software encode)

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

H.263 Encode (OSS software encode)

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

Image Encode

$ gst-launch-0.10 videotestsrc num-buffers=1 ! 'video/x-raw-yuv, width=(int)1280, height=(int)720, format=(fourcc)I420' ! nvjpegenc ! filesink location=test.jpg -e
Supported H.264 Encoder Features with Gstreamer-0.10

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

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

Set I-frame interval

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

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

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

Set rate control mode

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

Set quantization range for P and I frame

The format for the range is the following:

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

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

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

Set quality level

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

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

Video Encode Examples Using gst-launch-1.0

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

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

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

**VP8 Encode (NVIDIA accelerated encode)**

```bash
$ 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)**

```bash
$ 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)**

```bash
$ 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**

```bash
$ 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 R21.4 release is `nvgstcapture-0.10`. For usage information enter the following command:
$ nvgstcapture-0.10 --help

The `nvgstcapture-0.10` application uses the `v4l2src` plugin to capture still images and video.

The following table shows USB camera support.

<table>
<thead>
<tr>
<th>USB Camera Support</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>YUV</td>
<td>Preview display</td>
</tr>
<tr>
<td></td>
<td>Image capture (VGA, 640 x 480)</td>
</tr>
<tr>
<td></td>
<td>Video capture (480p, 720p, H.264/VP8 encode)</td>
</tr>
<tr>
<td>MJPEG</td>
<td>Preview display</td>
</tr>
<tr>
<td></td>
<td>Image capture VGA, 640 x 480</td>
</tr>
<tr>
<td></td>
<td>720p, 1280 x 720</td>
</tr>
<tr>
<td></td>
<td>Video capture (480p, 720p, 1080p, MJPEG encode)</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>USB Camera Support</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>YUV</td>
<td>Preview display</td>
</tr>
<tr>
<td></td>
<td>Image capture (VGA, 640 x 480)</td>
</tr>
<tr>
<td></td>
<td>Video capture (480p, 720p, H.264/VP8 encode)</td>
</tr>
</tbody>
</table>

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

$ gst-launch-1.0 v4l2src device="/dev/video0" ! "video/x-raw, width=640, height=480, format=(string)I420" ! xvimagesink -e
VIDEO PLAYBACK WITH GSTREAMER-0.10

The default playback application in the R21.4 release is nvgstplayer-0.10. For usage information enter the following command:

```bash
$ 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)**

```bash
$ 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)**

```bash
$ 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)**

```bash
$ 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:

```bash
$ 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)**

```bash
$ 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 nvgстplayer-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.

#### 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 nvrм-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,
```
Converting raw-gray to nvrm-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-nvrm-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.

```bash
$ 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.

raw-yuv Input Formats

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

Converting raw-yuv to nv-yuv with scaling

```bash
$ 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 nvrmb-uv with scaling

```bash
$ gst-launch-0.10 videotestsrc ! 'video/x-raw-yuv, width=(int)1280,
height=(int)720, format=(fourcc)NV12' ! nvvidconv ! 'video/x-nv-uv-
vuy, 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

```bash
$ 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-uv-
vuy, width=(int)640, height=(int)480, format=(fourcc)I420' ! nv OMX_h264enc
! qtmux ! filesink location=test.mp4 -e
```
Converting raw-gray to nvrm-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-nvrm-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-nv-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-nvrm-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' ! ffmpegcolorspace ! 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' ! ffmpegcolorspace ! 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:

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

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 nvrm-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-nvrm-yuv,
   width=640, height=480' ! nv_omx_h264enc ! qtmux ! filesink
   location=test.mp4 -e
```

Scaling nvrm-yuv

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

```bash
$ gst-launch-0.10 filesrc location=1280x720_30p.mp4 ! qtdemux name=mux ! nv_omx_h264dec ! 'video/x-nvrm-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.

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

raw-gray Input Formats

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

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

raw-yuv Output Formats

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

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

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

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

NVIDIA Input and Output Formats

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

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

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 the following video formats.

**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)**

```bash
$ 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)**

```bash
$ 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)**

```bash
$ 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)**

```bash
$ 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)**

```bash
$ 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)**

```bash
$ 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> ! qtdmux 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> ! qtdmux 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> ! qtdmux name=demux
demux.video_0 ! queue ! h264parse ! omxh264dec ! nvvidconv !
omxvp8enc ! qt mux 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> ! qtdmux
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> ! qtdmux
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 !
```
VIDEO ROTATION WITH GSTREAMER-1.0

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

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

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

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

To rotate video 90 degrees counterclockwise

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

```
$ 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:

```bash
$ 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:

```bash
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:

```bash
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:

```bash
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

<table>
<thead>
<tr>
<th>Method</th>
<th>Property Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>nearest</td>
<td>0</td>
</tr>
<tr>
<td>linear</td>
<td>1</td>
</tr>
<tr>
<td>smart (default)</td>
<td>2</td>
</tr>
<tr>
<td>bilinear</td>
<td>3</td>
</tr>
</tbody>
</table>

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

To use bilinear interpolation method for scaling

- Enter the following command:

```bash
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
```
This section describes the options available in the `nvgstcapture-1.0` application.

**NVGSTCAPTURE APPLICATION OPTIONS**

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

<table>
<thead>
<tr>
<th>Application Options</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Option</strong></td>
</tr>
<tr>
<td>--prev_res</td>
</tr>
<tr>
<td>--image_res</td>
</tr>
<tr>
<td>--video_res</td>
</tr>
<tr>
<td>-m, --mode</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>-v, --video_enc</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>-j, --image_enc</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>-k, --file_type</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>--svs</td>
</tr>
<tr>
<td>--file-name</td>
</tr>
<tr>
<td>--camsrc</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Option</td>
</tr>
<tr>
<td>-----------------</td>
</tr>
<tr>
<td>-w, --whitebalance</td>
</tr>
<tr>
<td>-s, --scene-mode</td>
</tr>
<tr>
<td>-c, --color-effect</td>
</tr>
<tr>
<td>--auto-exposure</td>
</tr>
<tr>
<td>--flash</td>
</tr>
<tr>
<td>--flicker</td>
</tr>
<tr>
<td>--contrast</td>
</tr>
<tr>
<td>--saturation</td>
</tr>
<tr>
<td>--edge-enhancement</td>
</tr>
<tr>
<td>--tnr_strength</td>
</tr>
<tr>
<td>--tnr_mode</td>
</tr>
</tbody>
</table>

### Help Options

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

### CSI CAMERA RUNTIME COMMANDS

CSI camera runtime commands are described in the following table.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>h</td>
<td>Help</td>
<td>-</td>
</tr>
<tr>
<td>q</td>
<td>Quit</td>
<td>-</td>
</tr>
<tr>
<td>mo:&lt;value&gt;</td>
<td>Set capture mode</td>
<td>1-image 2-video</td>
</tr>
<tr>
<td>gmo</td>
<td>Get capture mode</td>
<td>-</td>
</tr>
<tr>
<td>wb:&lt;value&gt;</td>
<td>Set white balance mode</td>
<td>0-off 1-auto 2-incandescent 3-fluorescent 4-warm-fluorescent 5-daylight</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
<td>Options</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>gwb</td>
<td>Get white balance mode</td>
<td>6-cloudy-daylight 7-twilight 8-shade</td>
</tr>
<tr>
<td>scm:&lt;value&gt;</td>
<td>Set scene mode</td>
<td>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</td>
</tr>
<tr>
<td>gcm</td>
<td>Get scene mode</td>
<td>-</td>
</tr>
<tr>
<td>ce:&lt;value&gt;</td>
<td>Set color effect mode</td>
<td>1-off 2-mono 3-negative 4-solarize 5-sepia 6-posterize 7-aqua</td>
</tr>
<tr>
<td>gce</td>
<td>Get color effect mode</td>
<td>-</td>
</tr>
<tr>
<td>ae:&lt;value&gt;</td>
<td>Set auto-exposure mode</td>
<td>1-off 2-on 3-OnAutoFlash 4-OnAlwaysFlash 5-OnFlashRedEye</td>
</tr>
<tr>
<td>gae</td>
<td>Get auto exposure mode</td>
<td>-</td>
</tr>
<tr>
<td>f:&lt;value&gt;</td>
<td>Set flash mode</td>
<td>0-off 1-on 2-torch 3-auto</td>
</tr>
<tr>
<td>gf</td>
<td>Get flash mode</td>
<td>-</td>
</tr>
<tr>
<td>fl:&lt;value&gt;</td>
<td>Set flash detection and avoidance mode</td>
<td>0-off 1-50 Hz 2-60 Hz 3-auto</td>
</tr>
<tr>
<td>gfl</td>
<td>Get flash detection and avoidance mode</td>
<td>-</td>
</tr>
<tr>
<td>ct:&lt;value&gt;</td>
<td>Set contrast</td>
<td>0-1, e.g., ct:0.75</td>
</tr>
<tr>
<td>gct</td>
<td>Get contrast</td>
<td>-</td>
</tr>
<tr>
<td>st:&lt;value&gt;</td>
<td>Set saturation</td>
<td>0-2, e.g., st:1.25</td>
</tr>
</tbody>
</table>
**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.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>h</strong></td>
<td>Help</td>
<td>-</td>
</tr>
<tr>
<td><strong>q</strong></td>
<td>Quit</td>
<td>-</td>
</tr>
</tbody>
</table>
| **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.

<table>
<thead>
<tr>
<th>Value</th>
<th>Command</th>
<th>Description</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Start recording video</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0</td>
<td>Stop recording video</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><code>pcr:&lt;value&gt;</code></td>
<td>Set preview resolution</td>
<td>0-176x144, 1-320x240, 2-640x480, 3-1280x720</td>
<td>-</td>
</tr>
<tr>
<td><code>gpcr</code></td>
<td>Get preview resolution</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><code>gicr</code></td>
<td>Get image capture resolution</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td><code>gvcr</code></td>
<td>Get video capture resolution</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Gstreamer-1.0 and gstreamer-0.10 support the following features, respectively:

<table>
<thead>
<tr>
<th>Video Encoder Feature</th>
<th>gst-openmax (gstreamer-0.10)</th>
<th>gst-omx (nvgstcapture-1.0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>H.264 Baseline / Main profile</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>bitrate</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>insert-spspsatidr</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>rc-mode</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>iframeinterval</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>qp-range</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>temporal-tradeoff</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>bit-packetization</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>quality-level</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>low-latency</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>framerate</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>slice-header spacing</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>force-IDR</td>
<td>✔</td>
<td></td>
</tr>
</tbody>
</table>
Notice

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

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

Trademarks

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

Copyright

© 2015, 2016 NVIDIA Corporation. All rights reserved.