Edge Video Analytics SDK
User's Manual

PM Review

Manual Rev.: 0.5 Preliminary
Revision Date: August 14, 2020
Part Number: 50-1Z320-1000

Preface

Copyright
Copyright © 2020 ADLINK Technology, Inc. This document contains proprietary information protected by copyright. All rights are reserved. No part of this manual may be reproduced by any mechanical, electronic, or other means in any form without prior written permission of the manufacturer.

Disclaimer
The information in this document is subject to change without prior notice in order to improve reliability, design, and function and does not represent a commitment on the part of the manufacturer. In no event will the manufacturer be liable for direct, indirect, special, incidental, or consequential damages arising out of the use or inability to use the product or documentation, even if advised of the possibility of such damages.

Trademarks
Product names mentioned herein are used for identification purposes only and may be trademarks and/or registered trademarks of their respective companies.

Revision History

Revision	Description	Date	By
0.1	Preliminary release for Edge Video Analytics SDK User's Manual	2020-06-04	RA
0.2	2nd draft from markup	2020-07-03	RA
0.3	3rd draft from R&D	2020-07-30	AT
0.4	Add the model optimizer information and more detail of the adtranslator element.	2020-08-06	AT
0.5	Add a note in the introduction chapter. Modify ADLINK element version to 2.0. Replace autovideosink to xvimagesink.	2020-08-14	AT

Table of contents

Introduction

The ADLINK Edge Video Analytics (EVA) SDK provides an integrated development environment (IDE) for developers wanting to build computer vision and video analytic solutions with deep learning technology. The SDK provides the required building blocks to help with the implementation of edge inference solutions. Customers can select either the Intel® Distribution of OpenVINO™ Toolkit, NVIDIA TensorRT™, or both to accelerate deep learning inference. This SDK can help customers easily migrate or integrate heterogeneous hardware platforms. ADLINK offers a selection of pre-verified platforms with specific hardware configurations optimized for vision and video analytic applications requiring little additional engineering effort by developers. The EVA SDK integrates and verifies different building blocks required for Edge Video Analytic applications. The building blocks are implemented as GStreamer plugins. Customers can select a specific plugin based on their requirements, utilizing it in their development pipeline to quickly build their application. These plugins can be categorized as follows.

- Image/video capture: Supports different image capture solutions with different camera standards.
- Image processing: Provides the required algorithms needed before or after the inference.
- Image analytics: Provides an optimized INTEL/NVIDIA inference engine to accelerate deep learning inference.
- Connector plugin: Provides network connections for uploading images and inference results to a network service.

Note	The examples using gst-launch in this user's manual are for reference only. The "property" values such as the file name and path must be modified according to the actual situation.

Image and Video Capture Plugins

The following table lists the elements of the supported image and video capture plugins.

Element Name	Plugin Name	Vendor	Version	Description
pylonsrc	Pylonsrc	PlayGineering	2.0	A GStreamer element that uses Basler's USB3 Vision cameras to capture images
filesrc	coreelements	GStreamer	1.14.5	Reads from an arbitrary point in a file
rtspsrc	rtsp	GStreamer	1.14.5	Receives network data via RTSP (RFC 2326)
v4l2src	video4linux2	GStreamer	1.14.5	Reads frames from a Video4Linux2 device
multifilesrc	multifile	GStreamer	1.14.5	Reads buffers from sequentially named files

Element Descriptions

This section describes each element of the video capture plugin in greater detail.

pylonsrc

This is a GStreamer element that uses Basler's USB3 Vision cameras and Basler's GigE Vision cameras for image capture.

When multiple cameras are available, use the camera parameter to specify the specific camera for image capture. To find out the IDs for each camera, launch the pipeline without specifying the camera parameter, and the plugin will output the camera IDs.

The `width` and `height` parameters are used to set a custom resolution. To find out the maximum resolution of the camera, refer to the camera's technical specifications, or launch the plugin with logging tuned to loglevel4 (GST_DEBUG=pylonsrc:4). Note that the camera saves the manually set resolution. Launching a pipeline with a camera while specifying a resolution will result in the following runs using the same resolution unless the device is reconnected or a different resolution is specified.

Resolution offsets can be specified to move the image. Possible offsets are calculated by `max(width|height)-current(width|height)`. The parameters to specify offsets

are `offsetx` and `offsety`. To center the image, use `centerx` and `centery`. Note that setting the centering parameters will cause the plugin to ignore the offset values.

The ADLINK EVA SDK uses the pylonsrc source code that includes in gst-plugins-vision. https://github.com/joshdoe/gst-plugins-vision/tree/master/sys/pylon

For more details, refer to the gst-pylonsrc documentation at https://gitlab.com/zingmars/gst-pylonsrc/-/blob/master/README.MD. Note that use 'pixel-format' property instead of 'imageformat' property when referencing this document.

Note	Download and install the ADLINK EVA SDK to access the pylonsrc plugin. For more details, refer to the ADLINK EVA Installation Guide.

Examples Using gst-launch

Displaying a pylon video with ycbcr422_8 format.

```
gst-launch-1.0 pylonsrc pixel-format=ycbcr422_8 ! videoconvert !
xvimagesink
```

Recording a pylon video at 150fps.

```
gst-launch-1.0 pylonsrc limitbandwidth=false sensorreadoutmode=fast
fps=150 ! videoconvert ! matroskamux ! filesink location='recording.mkv'
```

Setting the resolution width and height to 960x540 and displaying the pylon video.

```
gst-launch-1.0 pylonsrc width= 960 height=540 ! videoconvert ! xvimagesink
```

Centering the pylon video and display.

```
gst-launch-1.0 pylonsrc centerx=true centery=true ! videoconvert !
xvimagesink
```

Note	It is recommended to use xvimagesink to display the window.

Hierarchy

```
GObject
      └──GInitiallyUnowned
            └──GstObject
                  └──GstElement
                        └──GstBaseSrc
                              └──GstPushSrc
                                    └──GstPylonsrc
```

Pad Templates

src

ANY

Presence – *always*

Direction – *src*

Properties

camera

"camera" gint

(Number) Camera ID as defined by Basler's API. If only one camera is connected this parameter will be ignored and the camera will be used. If there are multiple cameras and this parameter is not defined, the plugin will output a list of available cameras and their IDs. Note that if there are multiple cameras available to the API and the camera parameter is not defined then this plugin will not run. Range: 0 to 100

Flags: Read, Write

Default value: 9999

height

```
"height" gint
```

(Pixel) The height of the picture. Note that the camera will remember this setting, and will use values from the previous runs if you relaunch without specifying this parameter. Reconnect the camera or use the reset parameter to reset. Range: 0 to 10000

Flags: Read, Write

Default value: 0

width

```
"width" gint
```

(Pixels) The width of the picture. Note that the camera will remember this setting, and will use values from the previous runs if you relaunch without specifying this parameter. Reconnect the camera or use the reset parameter to reset. Range: 0 to 10000

Flags: Read, Write

Default value: 0

binningh

```
"binningh" gint
```

(Pixels) The number of pixels to be binned in the horizontal direction. Note that the camera will remember this setting, and will use values from the previous runs if you relaunch without specifying this parameter. Reconnect the camera or use the reset parameter to reset. Range: 1 to 6

Flags: Read, Write

Default value: 1

binningv

```
"binningv" gint
```

(Pixels) The number of pixels to be binned in the vertical direction. Note that the camera will remember this setting, and will use values from the previous runs if you relaunch without specifying this parameter. Reconnect the camera or use the reset parameter to reset. Range: 1 to 6

Flags: Read, Write

Default value: 1

limitbandwidth

```
"limitbandwidth" gboolean
```

(true/false) Bandwidth limit mode. CAUTION! Disabling this will allow the camera to reach higher frames per second, but can damage the camera. Running the plugin without specifying this parameter will reset the value stored on the camera to `true`.

Flags: Read, Write

Default value: true

maxbandwidth

```
"maxbandwidth" gint64
```

(Bytes per second) This property sets the maximum bandwidth the camera can use. The camera will only use as much as it needs for the specified resolution and framerate. This setting will have no effect if limitbandwidth is set to off. Note that the camera will remember this setting, and will use values from the previous runs if you relaunch without specifying this parameter. Reconnect the camera or use the reset parameter to reset. Range: 0 to 999999999

Flags: Read, Write

Default value: 0

sensorreadoutmode

```
"sensorreadoutmode" gchararray
```

(normal/fast) This property changes the sensor readout mode. Fast will allow for faster framerates but might cause quality loss. It might be required that either increasing the maximum bandwidth or disabling bandwidth limits be configured for this to cause any noticeable change. Running the plugin without specifying this parameter will reset the value stored on the camera to "normal".

Flags: Read, Write

Default value: "normal"

acquisitionframerateenable

```
"acquisitionframerateenable" gboolean
```

(true/false) Enables the use of custom fps values. This parameter will be set to true if the fps property is set. Running the plugin without specifying this parameter will reset the value stored on the camera to false.

Flags: Read, Write

Default value: false

fps

```
"fps" gdouble
```

(Frames per second) Sets the framerate of the video coming from the camera. Setting the value too high might cause the plugin to crash. Note that if the pipeline causes the computer to hang, or stall, then the resulting video will not be in the resolution that was set. Setting this parameter will set acquisitionframerateenable to true. The value of this parameter will be saved to the camera, but it will have no effect unless either this or the acquisitionframerateenable parameters are set. Reconnect the camera or use the reset parameter to reset. Range: 0.0 to 1024.0

Flags: Read, Write

Default value: 0.0

lightsource

```
"lightsource" gchararray
```

(off, 2800k, 5000k, 6500k) Changes the color balance settings to those defined by the presets. For best results, select a color balance setting closest to the environment's lighting. Running the plugin without specifying this parameter will reset the value stored on the camera to "5000k".

Flags: Read, Write

Default value: "5000k"

autoexposure

```
"autoexposure" gchararray
```

(off, once, continuous) Controls whether the camera will try to adjust the exposure settings. Setting this parameter to anything but "off" will override the exposure parameter. Running the plugin without specifying this parameter will reset the value stored on the camera to "off".

Flags: Read, Write

Default value: "off"

exposure

```
"exposure" gdouble
```

(Microseconds) Exposure time for the camera in microseconds, but only has an effect if autoexposure is set to off (default). Higher numbers will cause a lower frame rate. Note that the camera will remember this setting, and will use values from the previous runs if you relaunch without specifying this parameter. Reconnect the camera or use the reset parameter to reset. Range: 0.0 to 1000000.0

Flags: Read, Write

Default value: 0.0

autowhitebalance

```
"autowhitebalance" gchararray
```

(off, once, continuous) Controls whether the camera will try to adjust the white balance settings. Setting this parameter to anything but "off" will override the exposure parameter. Running the plugin without specifying this parameter will reset the value stored on the camera to "off".

Flags: Read, Write

Default value: "off"

balancered

```
"balancered" gdouble
```

Specifies the red color balance. The autowhitebalance setting must be set to "off" for this property to have any effect. Note that this value gets saved on the camera, and running this plugin again without specifying this value will cause the previous value to be used. Use the reset parameter or reconnect the camera to reset. Range: 0.0 to 15.9

Flags: Read, Write

Default value: 999

balancegreen

```
"balancegreen" gdouble
```

Specifies the green color balance. The autowhitebalance setting must be set to "off" for this property to have any effect. Note that this value gets saved on the camera, and running this plugin again without specifying this value will cause the previous value to be used. Use the reset parameter or reconnect the camera to reset. Range: 0.0 to 15.9

Flags: Read, Write

Default value: 999

balanceblue

```
"balanceblue" gdouble
```

Specifies the blue color balance. The autowhitebalance setting must be set to "off" for this property to have any effect. Note that this value gets saved on the camera, and running this plugin again without specifying this value will cause the previous value to be used. Use the reset parameter or reconnect the camera to reset. Range: 0.0 to 15.9

Flags: Read, Write

Default value: 999

colorredhue

```
"colorredhue" gdouble
```

Specifies the red hue. Note that this value gets saved on the camera, and running this plugin again without specifying this value will cause the previous value to be used. Use the reset parameter or reconnect the camera to reset. Range: -4.0 to 3.9

Flags: Read, Write

Default value: 999

colorredsaturation

```
"colorredsaturation" gdouble
```

Specifies the red saturation. Note that this value gets saved on the camera, and running this plugin again without specifying this value will cause the previous value to be used. Use the reset parameter or reconnect the camera to reset. Range: 0.0 to 1.9

Flags: Read, Write

Default value: 0.0

coloryellowhue

```
"coloryellowhue" gdouble
```

Specifies the yellow hue. Note that this value gets saved on the camera, and running this plugin again without specifying this value will cause the previous value to be used. Use the reset parameter or reconnect the camera to reset. Range: -4.0 to 3.9

Flags: Read, Write

Default value: 999

coloryellowsaturation

```
"coloryellowsaturation" gdouble
```

Specifies the yellow saturation. Note that this value gets saved on the camera, and running this plugin again without specifying this value will cause the previous value to be used. Use the reset parameter or reconnect the camera to reset. Range: 0.0 to 1.9

Flags: Read, Write

Default value: 999

colorgreenhue

```
"colorgreenhue" gdouble
```

Specifies the green hue. Note that this value gets saved on the camera, and running this plugin again without specifying this value will cause the previous value to be used. Use the reset parameter or reconnect the camera to reset. Range: -4.0 to 3.9

Flags: Read, Write

Default value: 999

colorgreensaturation

```
"colorgreensaturation" gdouble
```

Specifies the green saturation. Note that this value gets saved on the camera, and running this plugin again without specifying this value will cause the previous value to be used. Use the reset parameter or reconnect the camera to reset. Range: 0.0 to 1.9

Flags: Read, Write

Default value: 999

colorcyanhue

```
"colorcyanhue" gdouble
```

Specifies the cyan hue. Note that this value gets saved on the camera, and running this plugin again without specifying this value will cause the previous value to be used. Use the reset parameter or reconnect the camera to reset. Range: -4.0 to 3.9

Flags: Read, Write

Default value: 999

colorcyansaturation

```
"colorcyansaturation" gdouble
```

Specifies the cyan saturation. Note that this value gets saved on the camera, and running this plugin again without specifying this value will cause the previous value to be used. Use the reset parameter or reconnect the camera to reset. Range: 0.0 to 1.9

Flags: Read, Write

Default value: 999

colorbluehue

```
"colorbluehue" gdouble
```

Specifies the blue hue. Note that this value gets saved on the camera, and running this plugin again without specifying this value will cause the previous value to be used. Use the reset parameter or reconnect the camera to reset. Range: -4.0 to 3.9

Flags: Read, Write

Default value: 0.0

colorbluesaturation

```
"colorbluesaturation" gdouble
```

Specifies the blue saturation. Note that this value gets saved on the camera, and running this plugin again without specifying this value will cause the previous value to be used. Use the reset parameter or reconnect the camera to reset. Range: 0.0 to 1.9

Flags: Read, Write

Default value: 999

colormagentahue

```
"colormagentahue" gdouble
```

Specifies the magenta hue. Note that this value gets saved on the camera, and running this plugin again without specifying this value will cause the previous value to be used. Use the reset parameter or reconnect the camera to reset. Range: -4.0 to 3.9

Flags: Read, Write

Default value: 999

colormagentasaturation

```
"colormagentasaturation" gdouble
```

Specifies the magenta saturation. Note that this value gets saved on the camera, and running this plugin again without specifying this value will cause the previous value to be used. Use the reset parameter or reconnect the camera to reset. Range: 0.0 to 1.9

Flags: Read, Write

Default value: 999

autogain

```
"autogain" gchararray
```

(off, once, continuous) Controls whether the camera will try to adjust the gain settings. Setting this parameter to anything but "off" will override the exposure parameter. Running the plugin without specifying this parameter will reset the value stored on the camera to "off".

Flags: Read, Write

Default value: "off"

gain

```
"gain" gdouble
```

(dB) Sets the gain added on the camera before sending the frame to the computer. The value of this parameter will be saved to the camera, but it will be set to 0 every time this plugin is launched without specifying gain, or it will be overridden if the autogain parameter is set to anything that is not "off". Reconnect the camera or use the reset parameter to reset the stored value. Range: 0.0 to 12.0

Flags: Read, Write

Default value: 0.0

blacklevel

```
"blacklevel" gdouble
```

(DN) Sets the stream's black level. This parameter is processed on the camera before the picture is sent to the computer. The value of this parameter will be saved to the camera, but it will be set to 0 every time this plugin is launched without specifying this parameter. Reconnect the camera or use the reset parameter to reset the stored value. Range: 0.0 to 63.75

Flags: Read, Write

Default value: 0.0

gamma

```
"gamma" gdouble
```

Sets the gamma correction value. This parameter is processed on the camera before the picture is sent to the computer. The value of this parameter will be saved to the camera, but it will be set to 1.0 every time this plugin is launched without specifying this parameter. Reconnect the camera or use the reset parameter to reset the stored value. Range: 0.0 to 3.9

Flags: Read, Write

Default value: 1.0

reset

```
"reset" gchararray
```

(off, before, after). Controls whether or when the camera's settings will be reset. Setting this to "before" will wipe the settings before the camera initialization begins. Setting this to "after" will reset the device once the pipeline closes. This can be useful for debugging or when using the camera with other software that does not reset the camera settings before use (such as PylonViewerApp).

Flags: Read, Write

Default value: "off"

testimage

```
"testimage" gint
```

(1 to 6) Specifies a test image to show instead of a video stream. It is useful for debugging and is disabled by default. Range: 0 to 6

Flags: Read, Write

Default value: 0

continuous

```
"continuous" gboolean
```

(true/false) Used to switch between triggered and continuous mode. Set to “false” to switch to triggered mode.

Flags: Read, Write

Default value: True

pixel-format

```
"pixel-format" gchararray
```

(Mono8/BayerBG8/BayerGR8/BayerRG8/BayerGB8/RGB8/BGR8/YCbCr422_8/YUV422Packed) Determines the pixel format for sending frames. The default is auto which will query the camera for supported pixel formats and allow GStreamer to negotiate the format with downstream elements.

Flags: Read, Write

Default value: "auto"

userid

```
"userid" gchararry
```

(<string>) Sets the device custom ID.

Flags: Read, Write

Default value: NULL

demosaicing

```
"demosaicing" gboolean
```

(true/false) Switches between simple and Basler's demosaicing (PGI) mode. This does not work if bayer output is used.

Flags: Read, Write

Default value: False

noisereduction

```
"noisereduction" gdouble
```

Specifies the amount of noise reduction to apply. To use this, Basler's demosaicing mode must be enabled. Setting this will enable demosaicing mode. Range: 0.0 to 2.0

Flags: Read, Write

Default value: 999

sharpnessenhancement

```
"sharpnessenhancement" gdouble
```

Specifies the amount of sharpness enhancement to apply. To use this, Basler's demosaicing mode must be enabled. Setting this will enable demosaicing mode. Range: 1.0 to 3.98

Flags: Read, Write

Default value: 999

offsetx

```
"offsetx" gint
```

(0 to 10000) Determines the vertical offset. Note that the maximum offset value is calculated during initialization, and will not be shown in this output.

Flags: Read, Write

Default value: 99999

centerx

```
"centerx" gboolean
```

(true/false) Setting this to true will center the horizontal offset and cause the plugin to ignore the offsetx value.

Flags: Read, Write

Default value: False

offsety

```
"offsety" gint
```

(0 to 10000) Determines the vertical offset. Note that the maximum offset value is calculated during initialization, and will not be shown in this output.

Flags: Read, Write

Default value: 99999

centery

```
"centery" gboolean
```

(true/false) Setting this to true will center the vertical offset and cause the plugin to ignore the offsety value.

Flags: Read, Write

Default value: False

flipx

```
"flipx" gboolean
```

(true/false) Setting this to true will flip the image horizontally.

Flags: Read, Write

Default value: False

flipy

```
"flipy" gboolean
```

(true/false) Setting this to true will flip the image vertically.

Flags: Read, Write

Default value: False

exposureupperlimit

```
"exposureupperlimit" gdouble
```

(105 to 1000000) Sets the upper limit for the auto exposure function.

Flags: Read, Write

Default value: 999

exposurelowerlimit

```
"exposurelowerlimit" gdouble
```

(105 to 1000000) Sets the lower limit for the auto exposure function.

Flags: Read, Write

Default value: 999

gainupperlimit

```
"gainupperlimit" gdouble
```

(0 to 12.00921) Sets the upper limit for the auto gain function.

Flags: Read, Write

Default value: 999

gainlowerlimit

```
"gainlowerlimit" gdouble
```

(0 to 12.00921) Sets the lower limit for the auto gain function.

Flags: Read, Write

Default value: 999

autoprofile

```
"autoprofile" gchararray
```

(gain/exposure) When the auto functions are on, this determines whether to focus on minimizing gain or exposure.

Flags: Read, Write

Default value: "default"

autobrightnesstarget

```
"autobrightnesstarget" gdouble
```

(0.19608 to 0.80392) Sets the auto exposure brightness target value.

Flags: Read, Write

Default value: 999

transformationselector

```
"transformationselector" gchararray
```

(RGBRGB, RGBYUV, YUVRGB) Sets the type of color transformation done by the color transformation selectors.

Flags: Read, Write

Default value: "default"

transformation00

```
"transformation00" gdouble
```

Gain00 transformation selector. Range: -8.0 to 7.96875

Flags: Read, Write

Default value: 999

transformation01

```
"transformation01" gdouble
```

Gain01 transformation selector. Range: -8.0 to 7.96875

Flags: Read, Write

Default value: 999

transformation02

```
"transformation02" gdouble
```

Gain02 transformation selector. Range: -8.0 to 7.96875

Flags: Read, Write

Default value: 999

transformation10

```
"transformation10" gdouble
```

Gain10 transformation selector. Range: -8.0 to 7.96875

Flags: Read, Write

Default value: 999

transformation11

```
"transformation11" gdouble
```

Gain11 transformation selector. Range: -8.0 to 7.96875

Flags: Read, Write

Default value: 999

transformation12

```
"transformation12" gdouble
```

Gain12 transformation selector. Range: -8.0 to 7.96875

Flags: Read, Write

Default value: 999

transformation20

```
"transformation20" gdouble
```

Gain20 transformation selector. Range: -8.0 to 7.96875

Flags: Read, Write

Default value: 999

transformation21

```
"transformation21" gdouble
```

Gain21 transformation selector. Range: -8.0 to 7.96875

Flags: Read, Write

Default value: 999

transformation22

```
"transformation22" gdouble
```

Gain22 transformation selector. Range: -8.0 to 7.96875

Flags: Read, Write

Default value: 999

maxtransfersize

```
"maxtransfersize" gdouble
```

The default value is appropriate for most applications. Reducing the value may cause a higher CPU load. USB host adapter drivers may require decreasing the value in case the application fails to receive the image stream. The maximum value for the Maximum Transfer Size depends on the operating system version and may be limited by the host adapter drivers. Range: 65536 to 4194304

Flags: Read, Write

Default value: 1045876

---

filesrc

This element reads data from a file on the local system. For more details, refer to the official GStreamer documentation at:
https://gstreamer.freedesktop.org/documentation/coreelements/filesrc.html?gi-language=c#filesrc

Examples Using gst-launch

Play the song.ogg audio file located in the current working directory.

```
gst-launch-1.0 filesrc location=song.ogg ! decodebin ! audioconvert !
audioresample ! autoaudiosink
```

Play the video.avi video file located in the current working directory.

```
gst-launch-1.0 filesrc location=video.avi ! decodebin ! videoconvert !
xvimagesink
```

The following file formats are supported in GStreamer: avi, m4v, mov, mp4, mpg (MPEG-1/MPEG-2), flv, mkv, and wmv.

---

rtspsrc

This element receives data over a network via RTSP (RFC 2326). For more details, refer to the official documentation at: https://gstreamer.freedesktop.org/documentation/rtsp/rtspsrc.html?gi-language=c#rtspsrc-page

Examples Using gst-launch

Establish a connection to an RTSP server and send the raw RTP packets to a fakesink.

```
gst-launch-1.0 rtspsrc location=rtsp://some.server/url ! fakesink
```

Launch an RTSP video stream from a Bosch camera with the H.264 codec. The example specifies the IP address as 192.168.1.20 with a user ID of ooo, and xxx as the password.

```
gst-launch-1.0 rtspsrc location=rtsp://192.168.1.20 user-id=ooo user-
pw=xxx ! rtph264depay ! h264parse ! avdec_h264 ! xvimagesink sync=false
```

v4l2src

This element can be used to capture video from v4l2 devices, such as webcams and TV tuner cards.

For more details, refer to the official documentation at:
https://gstreamer.freedesktop.org/documentation/video4linux2/v4l2src.html?gi-language=c#v4l2src-page

Examples Using gst-launch

This pipeline shows the video captured from a webcam with jpeg images.

```
gst-launch-1.0 v4l2src ! jpegdec ! xvimagesink
```

This pipeline shows the video captured from a webcam with jpeg images and the buffer number set to output before sending EOS. For auto-function testing, the output automatically stops when the buffer is reached.

```
gst-launch-1.0 v4l2src num-buffers=100 ! jpegdec ! xvimagesink
```

---

multifilesrc

This element reads buffers from sequentially named files. If used together with an image decoder, the “caps” property or a capsfilter must be used to force to caps containing a framerate, otherwise the image decoders send EOS after the first frame. A videorate element is needed to set timestamps on all buffers after the first one in accordance with the corresponding framerate.

File names are created by replacing "%d" with the index using `printf()`.

For more details, refer to the official documentation at:
https://gstreamer.freedesktop.org/documentation/multifile/multifilesrc.html?gi-language=c#multifilesrc-page

Example Using gst-launch

This pipeline displays a video by joining multiple sequentially named PNG files (img.0000.png, img.0001.png, etc.).

```
gst-launch-1.0 multifilesrc location="img.%04d.png" index=0
caps="image/png,framerate=\(fraction\)12/1" ! pngdec ! videoconvert !
videorate ! xvimagesink
```

Image Processing Plugins

The following table lists the element of the supported image processing plugins.

Element Name	Plugin Name	Vendor	Version	Description
jpegdec	jpeg	GStreamer	1.14.5	Decodes jpeg images
jpegenc	jpeg	GStreamer	1.14.5	Encodes jpeg images
pngdec	png	GStreamer	1.14.5	Decodes png images. If there is no framerate set on sink caps, it sends EOS after the first picture.
pngenc	png	GStreamer	1.14.5	Encodes png images
gdkpixbufdec	gdkpixbuf	GStreamer	1.14.5	Decodes images in a video stream using GdkPixbuf
avenc_tiff	libva	GStreamer	1.14.5	libav tiff encoder
avenc_bmp	libva	GStreamer	1.14.5	libav bmp encoder
videorate	gstvideorate	GStreamer	1.14.5	Drops/duplicates/adjusts timestamps on video frames to make a perfect stream
videoconvert	videoconvert	GStreamer	1.14.5	Converts video frames between video formats
x264enc	x264	GStreamer	1.14.5	H264 encoder
x265enc	x265	GStreamer	1.14.5	HEVC encoder
msdkh264enc	msdk	GStreamer	1.14.5	Intel MSDK H264 encoder in gst-pulings-bad mode
msdkh265enc	msdk	GStreamer	1.14.5	Intel MSDK H265 encoder in gst-pulings-bad mode
msdkh264dec	msdk	GStreamer	1.14.5	Intel MSDK H264 decoder in gst-pulings-bad mode
msdkh265dec	msdk	GStreamer	1.14.5	Intel MSDK H265 encoder in gst-pulings-bad mode
avdec_h264	libva	GStreamer	1.14.5	libav h264 decoder
avdec_h265	libva	GStreamer	1.14.5	libav hevc decoder

Element Name	Plugin Name	Vendor	Version	Description
videoscale	videoscale	GStreamer	1.14.5	Resizes video frame
videocrop	videocrop	GStreamer	1.14.5	Crops video to a user-defined region
admetawriter	admetadata	ADLINK	2.0	Writes admeta data to stream
admetadrawer	admetadrawer	ADLINK	2.0	Draws admeta data to stream
admetadumper	admetadumper	ADLINK	2.0	Dumps admeta data to console
admetadebuger	admetadebuger	ADLINK	2.0	Writes fake admeta data to stream
nvv4l2decoder	nvvideo4linux2	NVIDIA DeepStream	4.0.2	Decodes video streams via V4L2 API
nvv4l2h264enc	nvvideo4linux2	NVIDA DeepStream	4.0.2	Encodes H.264 video streams via V4L2 AP
nvv4l2h265enc	nvvideo4linux2	NVIDA DeepStream	4.0.2	Encodes H.265 video streams via V4L2 AP

Element Descriptions

This section describes each element of the image processing plugin in greater detail.

jpegdec

This element decodes jpeg images.

For more details, refer to the official documentation at:
https://gstreamer.freedesktop.org/documentation/jpeg/jpegdec.html?gi-language=c

Examples Using gst-launch

This pipeline transcodes a jpeg to png and saves it as a png file.

```
gst-launch-1.0 filesrc location="frameJ1.jpeg" ! jpegdec ! videoconvert !
```

```
pngenc ! filesink location="frameJP1.png"
```

Similarly, jpeg images can be transcoded to tiff, bmp, or raw image formats.

```
gst-launch-1.0 filesrc location="frameJ1.jpeg" ! jpegdec ! videoconvert !
avenc_tiff ! filesink location="frameJT1.tiff"
```

```
gst-launch-1.0 filesrc location="frameJ1.jpeg" ! jpegdec ! videoconvert !
avenc_bmp ! filesink location="frameJB1.bmp"
```

```
gst-launch-1.0 filesrc location="frameJ1.jpeg" ! jpegdec ! videoconvert !
filesink location="frameJR1.raw"
```

The "idct-method" property controls the jpeg transcoding method.

1. islow (0) – Slow, but accurate integer algorithm
2. ifast (1) – Faster, but less accurate integer method
3. float (2) – Floating-point: accurate, but faster speed depends on better hardware

For example,

```
gst-launch-1.0 filesrc location="frame1.jpeg"  ! jpegdec idct-
method=float ! videoconvert ! filesink location="frameP1.raw"
```

---

jpegenc

This element encodes jpeg images.

For more details, refer to the official documentation at:
https://gstreamer.freedesktop.org/documentation/jpeg/jpegenc.html?gi-language=c

Example Using gst-launch

This pipeline encodes the video test pattern to jpeg images and saves them.

```
gst-launch-1.0 videotestsrc ! queue ! jpegenc ! multifilesink
location="frame%d.jpeg"
```

The "idct-method" property controls the jpeg encoding method.

1. islow (0) – Slow, but accurate integer algorithm
2. ifast (1) – Faster, but less accurate integer method
3. float (2) – Floating-point: accurate, but faster speed depends on better hardware

The "quality" property sets the encoding quality within a range of 0 to 100, with a default setting of 85.

---

pngdec

This element decodes png images.

For more details, refer to the official documentation at:
https://gstreamer.freedesktop.org/documentation/png/pngdec.html?gi-language=c

Examples Using gst-launch

This pipeline transcodes a png to jpeg and saves it as a jpeg file.

```
gst-launch-1.0 filesrc location="frameP1.png" ! pngdec ! videoconvert !
jpegenc ! filesink location="framePJ1.jpeg"
```

Similarly, png images can be transcoded to tiff, bmp, or raw image formats.

```
gst-launch-1.0 filesrc location="frameP1.png" ! pngdec ! videoconvert !
avenc_tiff ! filesink location="framePT1.tiff"
```

```
gst-launch-1.0 filesrc location="frameP1.png" ! pngdec ! videoconvert !
avenc_bmp ! filesink location="frameB1.bmp"
```

```
gst-launch-1.0 filesrc location="frameP1.png" ! pngdec ! videoconvert !
filesink location="framePR1.raw"
```

pngenc

This element encodes png images.

For more details, refer to the official documentation at:
https://gstreamer.freedesktop.org/documentation/png/pngenc.html?gi-language=c

Example Using gst-launch

This pipeline encodes the video test pattern to png images and saves them.

```
gst-launch-1.0 videotestsrc ! queue ! pngenc ! multifilesink
location="frame%d.png"
```

The "compression-level" property sets the PNG compression level within a range of 0 to 9, with a default setting of 6. A higher number creates a smaller file size.

---

gdkpixbufdec

This element decodes images in a video stream using GdkPixbuf.

For more details, refer to the official documentation at:
https://gstreamer.freedesktop.org/documentation/gdkpixbuf/gdkpixbufdec.html?gi-language=c#gdkpixbufdec-page

Examples Using gst-launch

These pipelines transcode a tiff or bmp to png and saves it as a png file.

```
gst-launch-1.0 filesrc location="frameT1.tiff" ! gdkpixbufdec !
videoconvert ! pngenc ! filesink location="frameTP1.png"
```

```
gst-launch-1.0 filesrc location="frameB1.bmp" ! gdkpixbufdec !
videoconvert ! pngenc ! filesink location="frameBP1.png"
```

Similarly, tiff or bmp images can be transcoded to other image formats.

```
gst-launch-1.0 filesrc location="frameB1.bmp" ! gdkpixbufdec !
videoconvert ! avenc_tiff ! filesink location="frameBT1.tiff"
```

```
gst-launch-1.0 filesrc location="frameT1.tiff" ! gdkpixbufdec !
videoconvert ! avenc_bmp ! filesink location="frameTB1.bmp"
```

```
gst-launch-1.0 filesrc location="frameT1.tiff" ! gdkpixbufdec !
videoconvert ! filesink location="frameTR1.raw"
```

```
gst-launch-1.0 filesrc location="frameB1.bmp" ! gdkpixbufdec !
videoconvert ! filesink location="frameBR1.raw"
```

avenc_tiff

This element encodes images in tiff format (based on libav).

For more details, refer to the official documentation at:
https://gstreamer.freedesktop.org/documentation/libav/avenc_tiff.html?gi-language=c#avenc_tiff-page

Example Using gst-launch

This pipeline encodes the video test pattern to tiff images and saves them.

```
gst-launch-1.0 videotestsrc ! queue ! avenc_tiff ! multifilesink
location="frame%d.tiff"
```

---

avenc_bmp

This element encodes images in bmp format (based on libav).

For more details, refer to the official documentation at:
https://gstreamer.freedesktop.org/documentation/libav/avenc_bmp.html?gi-language=c#avenc_bmp-page

Example Using gst-launch

This pipeline encodes the video test pattern to bmp images and saves them.

```
gst-launch-1.0 videotestsrc ! queue ! avenc_bmp ! multifilesink
location="frame%d.bmp"
```

---

videorate

This element takes an incoming stream of timestamped video frames and produces a perfect stream that matches the source pad's frame rate.

The operation is performed by dropping and duplicating frames. No algorithm is used to interpolate frames.

By default the element will simply negotiate the same frame rate on its source and sink pad.

This operation is useful to link to elements that require a perfect stream. Typical examples are formats that do not store timestamps for video frames, but only store a frame rate, like Ogg and AVI.

A conversion to a specific frame rate can be forced by using filtered caps on the source pad.

For more details, refer to the official documentation at: https://gstreamer.freedesktop.org/documentation/videorate/index.html?gi-language=c#videorate

Examples Using gst-launch

There are two ways to set the stream frame rate.

1. Directly use caps types to define the properties of the stream. The syntax is:

<type>[,<property>=<value>]...

The supported video types are listed in "***List of Defined Media Types***" from the GStreamer documentation at: https://gstreamer.freedesktop.org/documentation/plugin-development/advanced/media-types.html?gi-language=c#table-of-video-types

The following example shows how to directly set the stream frame rate.

```
gst-launch-1.0 videotestsrc ! video/x-raw, framerate=25/1 ! xvimagesink
```

This will set the stream frame rate to 25. This frame rate setting method is only supported when the pipeline is generated at the beginning of the stream, meaning there

is no opportunity to have two different frame rates in the pipeline. To set two different frame rates in the pipeline, follow the example below.

2. The "videorate" element is used to adjust the frame rate of the video in the pipeline.

For example, the following pipeline has a framerate of 25.

```
gst-launch-1.0 videotestsrc ! video/x-raw, framerate=25/1 ! gaussianblur !
videoconvert ! xvimagesink
```

To change the frame rate to 5 before applying xvimagesink, add the videorate element before it. The videorate will change the frame rate to adopt the coming media type setting in its src pad.

```
gst-launch-1.0 videotestsrc ! video/x-raw, framerate=25/1 ! gaussianblur !
videorate ! video/x-raw, framerate=5/1 ! videoconvert ! xvimagesink
```

---

videoconvert

This element converts from one color space to another (e.g. RGB to YUV). It can also convert between different YUV formats (e.g. I420, NV12, YUY2…) or RGB format arrangements (e.g. RGBA, ARGB, BGRA…).

This is normally the first choice when solving negotiation problems. When not needed, because its upstream and downstream elements can already understand each other, it acts in pass-through mode having minimal impact on performance.

As a general rule, always use `videoconvert` whenever you use elements whose Caps are unknown at design time, like `autovideosink`, or that can vary depending on external factors, like decoding a user-provided file.

For more details, refer to the official documentation at:
https://gstreamer.freedesktop.org/documentation/videoconvert/index.html?gi-language=c#videoconvert-page

Example Using gst-launch

This pipeline outputs a test video generated in YUY2 format in a video window.

```
gst-launch-1.0 videotestsrc ! video/x-raw,format=YUY2 ! videoconvert !
xvimagesink
```

If the video sink selected does not support YUY2, videoconvert will automatically convert the video to a format understood by the video sink.

---

videoscale

This element resizes video frames. By default the element will try to negotiate to the same size on the source and sinkpad so that no scaling is needed. It is therefore safe to insert this element in a pipeline to get more robust behavior without any cost if no scaling is needed.

This element supports a wide range of color spaces including various YUV and RGB formats and is therefore generally able to operate anywhere in a pipeline.

For more details, refer to the official documentation at: https://gstreamer.freedesktop.org/documentation/videoscale/index.html?gi-language=c#videoscale-page

Example Using gst-launch

This pipeline resizes the frames from videotestsrc to 720x576 and displays it via xvimagesink.

```
gst-launch-1.0 videotestsrc ! videoscale ! video/x-
raw,width=720,height=576 ! xvimagesink
```

---

videocrop

This element crops video frames, meaning it can remove parts of the picture on the left, right, top or bottom of the picture and output a smaller picture with the unwanted parts removed.

For more details, refer to the official documentation at:
https://gstreamer.freedesktop.org/documentation/videocrop/videocrop.html?gi-language=c#videocrop

Note	No special effort is made to handle chroma-subsampled formats in the case of odd-valued cropping and compensation for sub-unit chroma plane shifts in the case where the left or top property is set to an odd number.

Example Using gst-launch

```
gst-launch-1.0 videotestsrc ! videocrop top=42 left=1 right=4 bottom=0 !
ximagesink
```

---

x264enc

This element encodes raw video into H264 compressed data, otherwise known as MPEG-4 AVC (Advanced Video Codec).

For more details, refer to the official documentation at:
https://gstreamer.freedesktop.org/documentation/x264/?gi-language=c

Example Using gst-launch

This pipeline encodes a test video source to H264 using fixed quantization, and muxes it in a Matroska container.

```
gst-launch-1.0 videotestsrc num-buffers=1000 ! x264enc pass=quant !
matroskamux ! filesink location=x264enc.mkv
```

Note	Some settings, including the default settings, may lead to considerable latency (i.e. frame buffering) in the encoder. This may cause problems with pipeline stalling in non-trivial pipelines, because the encoder latency is often considerably higher than the default size of a simple queue element. Such problems are caused by one of the queues in the other non-x264enc streams/branches filling up and blocking upstream. They can be fixed by relaxing the default time/size/buffer limits on the queue elements in the non-x264 branches, or using a (single) multiqueue element for all branches. Another workaround for this problem is setting the tune=zerolatency property, but this will affect overall encoding quality so may not be appropriate for every use case.

x265enc

This element encodes raw video into H265 compressed data.

For more details, refer to the official documentation at:
https://gstreamer.freedesktop.org/documentation/x265/?gi-language=c

Example Using gst-launch

This pipeline encodes a test video source to H265 and muxes it in a Matroska container.

```
gst-launch-1.0 videotestsrc num-buffers=2000 ! x265enc ! h265parse !
matroskamux ! filesink location=x265enc.mkv
```

---

avdec_h264

This element is a libav h264 decoder.

For more details, refer to the official documentation at:
https://gstreamer.freedesktop.org/documentation/libav/avdec_h264.html?gi-language=c

Example Using gst-launch

This pipeline decodes an H.264 video file.

```
gst-launch-1.0 filesrc location=sample_1080p_h264.mp4 ! qtdemux !
h264parse ! queue ! avdec_h264 ! xvimagesink
```

---

avdev_h265

This element is a libav HEVC decoder.

For more details, refer to the official documentation at:
https://gstreamer.freedesktop.org/documentation/libav/avdec_h265.html?gi-language=c

Example Using gst-launch

This pipeline decodes an HEVC video file.

```
gst-launch-1.0 filesrc location=4K_h265.mkv ! matroskademux ! h265parse !
avdec_h265 ! videoconvert ! xvimagesink sync=false
```

---

msdkh264enc

This element is an Intel® Media SDK H.264 encoder.

Note	• Only support Intel platform. • Refer to the EVA SDK Installation Guide to install OpenVino toolkits and Media SDK for GStreamer before using the MSDK plugin.

Example Using gst-launch

This pipeline encodes the video test pattern as H.264 image and saves it as an mkv file.

```
gst-launch-1.0 videotestsrc num-buffers=1000 ! msdkh264enc ! h264parse !
matroskamux ! filesink location=msdkh264enc.mkv
```

Hierarchy

```
GObject
    └──GInitiallyUnowned
        └──GstObject
            └──GstElement
                └──GstVideoEncoder
                    └──GstMsdkEnc
                        └──GstMsdkH264Enc
```

Pad Templates

sink

```
video/x-raw:
          format: { (string)NV12, (string)I420, (string)YV12, (string)YUY2,
(string)UYVY, (string)BGRA }
       framerate: [ 0/1, 2147483647/1 ]
           width: [ 16, 2147483647 ]
          height: [ 16, 2147483647 ]
 interlace-mode: progressive
```

Presence – *always*

Direction – *sink*

src

```
video/x-h264:
      framerate: [ 0/1, 2147483647/1 ]
          width: [ 1, 2147483647 ]
         height: [ 1, 2147483647 ]
  stream-format: byte-stream
      alignment: au
        profile: { (string)high, (string)main, (string)baseline,
(string)constrained-baseline }
```

Presence – *always*

Direction – *src*

Properties

hardware

"hardware" gboolean

Enables hardware encoders

Flags: Read, Write

Default value: True

async-depth

"async-depth" guint

Depth of asynchronous pipeline. Range: 1 to 20

Flags: Read, Write

Default value: 4

target-usage

"target-usage" guint

1: Best quality, 4: Balanced, 7: Best speed. Range: 1 to 7

Flags: Read, Write

Default value: 4

rate-control

"rate-control" GstMsdkEncRateControl

Rate control method

Flags: Read, Write

Default value: cbr (1)

GstMsdkEncRateControl

Members

`cbr` (*1*) – Constant Bitrate
vbr (*2*) – Variable Bitrate
cqp (*3*) – Constant Quantizer
avbr (*4*) – Average Bitrate
la_vbr (*8*) – VBR with look ahead (Non HRD compliant)
icq (*9*) – Intelligent CQP
`vcm` (*10*) – Video Conferencing Mode (Non HRD compliant)
`la_icq` (*11*) – Intelligent CQP with LA (Non HRD compliant)
`la_hrd` (*13*) – HRD compliant LA
`qvbr` (*14*) – VBR with CQP

bitrate

```
"bitrate" guint
```

Bitrate in kbit/sec. Range: 1 to 2048000

Flags: Read, Write

Default value: 2048

max-frame-size

```
"max-frame-size" guint
```

Maximum possible size (kb) of any compressed frames (0: auto-calculate). Range: 0 to 65535

Flags: Read, Write

Default value: 0

max-vbv-bitrate

```
"max-vbv-bitrate" guint
```

Maximum bitrate (kbit/sec) at which data enters the video buffering verifier (0: auto-calculate). Range: 0 to 65535

Flags: Read, Write

Default value: 0

accuracy

```
"accuracy" guint
```

The AVBR accuracy per one tenth of one percent. Range: 0 to 65535

Flags: Read, Write

Default value: 0

convergence

```
"convergence" guint
```

The AVBR convergence per 100 frames. Range: 0 to 65535

Flags: Read, Write

Default value: 0

rc-lookahead

```
"rc-lookahead" guint
```

Number of frames to look ahead for rate control. Range: 0 to 100

Flags: Read, Write

Default value: 0

qpi

```
"qpi" guint
```

Constant quantizer for I frames (0 unlimited). Also used as ICQQuality or QVBRQuality for different RateControl methods. Range: 0 to 51

Flags: Read, Write

Default value: 0

qpp

```
"qpp" guint
```

Constant quantizer for P frames (0 unlimited). Range: 0 to 51

Flags: Read, Write

Default value: 0

qpb

```
"qpb" guint
```

Constant quantizer for B frames (0 unlimited) Range: 0 to 51

Flags: Read, Write

Default value: 0

gop-size

```
"gop-size" guint
```

GOP Size. Range: 0 to 2147483647

Flags: Read, Write

Default value: 256

ref-frames

```
"ref-frames" guint
```

Number of reference frames. Range: 0 to 2147483647

Flags: Read, Write

Default value: 1

i-frames

```
"i-frames" guint
```

Number of I frames between IDR frames. Range: 0 to 2147483647

Flags: Read, Write

Default value: 0

b-frames

```
"b-frames" guint
```

Number of B frames between I and P frames. Range: 0 to 2147483647

Flags: Read, Write

Default value: 0

num-slices

```
"num-slices" guint
```

Number of slices per frame. Zero tells the encoder to choose any slice partitioning allowed by the codec standard. Range: 0 to 2147483647

Flags: Read, Write

Default value: 0

mbbrc

```
"mbbrc" GstMsdkEncMbBitrateControl
```

Macroblock level bitrate control

Flags: Read, Write

Default value: 32, "off"

GstMsdkEncMbBitrateControl

Members

(0): auto - SDK decides what to do
(32): off - Disable Macroblock level bit rate control
(16): on - Enable Macroblock level bit rate control

i-adapt

```
"i-adapt" GstMsdkEncAdaptiveI
```

Adaptive I-Frame Insertion control

Flags: Read, Write

Default value: 32, "off"

GstMsdkEncAdaptiveI

Members

(0): auto - SDK decides what to do
(32): off - Disable Adaptive I frame insertion
(16): on - Enable Adaptive I frame insertion

b-adapt

```
"b-adapt" GstMsdkEncAdaptiveB
```

Adaptive B-Frame Insertion control

Flags: Read, Write

Default value: 32, "off"

GstMsdkEncAdaptiveB

Member

(0): auto - SDK decides what to do
(32): off - Disable Adaptive B-Frame insertion
(16): on - Enable Adaptive B-Frame insertion

cabac

```
"cabac" gboolean
```

Enable CABAC entropy coding

Flags: Read, Write

Default value: true

low-power

```
"low-power" gboolean
```

Enable low power mode

Flags: Read, Write

Default value: false

frame-packing

```
"frame-packing" GstMsdkH264EncFramePacking
```

Set frame packing mode for stereoscopic content.

Flags: Read, Write

Default value: Default: -1, "none"

GstMsdkH264EncFramePacking

Members

(-1): none (default)
(3): side-by-side
(7): top-bottom

rc-lookahead-ds

```
"rc-lookahead-ds" GstMsdkEncRCLookAheadDownsampling
```

Downsampling mode in look ahead bitrate control.

Flags: Read, Write

Default value: 0, "default"

GstMsdkEncRCLookAheadDownsampling

Members

(0): default - SDK decides what to do
(1): off - No downsampling
(2): 2x - Downsample 2 times before estimation
(3): 4x - Downsample 4 times before estimation

trellis

```
"trellis" GstMsdkEncTrellisQuantization
```

Enable Trellis Quantization

Flags: Read, Write

Default value: 0x00000000, "None"

GstMsdkEncTrellisQuantization

Members

(0x00000000): None - Disable for all frames
(0x00000002): i - Enable for I frames
(0x00000004): p - Enable for P frames

(0x00000008): b - Enable for B frames

max-slice-size

```
"max-slice-size" guint
```

Maximum slice size in bytes (if enabled MSDK will ignore the control over num-slices). Range: 0 to 4294967295

Flags: Read, Write

Default value: 0

b-pyramid

```
"b-pyramid" gboolean
```

Enable B-Pyramid reference structure

Flags: Read, Write

Default value: false

msdkh265enc

This element is an Intel® Media SDK H.265 encoder.

Note	• Only support Intel platform. • Refer to the EVA SDK Installation Guide to install OpenVino toolkits and Media SDK for GStreamer before using the MSDK plugin.

Example Using gst-launch

This pipeline encodes the video test pattern as H.265 image and saves it as an mkv file.

```
st-launch-1.0 videotestsrc num-buffers=1000 ! msdkh265enc ! h265parse !
matroskamux ! filesink location=msdkh265enc.mkv
```

Hierarchy

```
GObject
    └──GInitiallyUnowned
        └──GstObject
            └──GstElement
                └──GstVideoEncoder
                    └──GstMsdkEnc
                        └──GstMsdkH265Enc
```

Pad Templates

sink

```
video/x-raw:
          format: { (string)NV12, (string)I420, (string)YV12, (string)YUY2,
(string)UYVY, (string)BGRA }
       framerate: [ 0/1, 2147483647/1 ]
           width: [ 16, 2147483647 ]
          height: [ 16, 2147483647 ]
 interlace-mode: progressive
```

Presence – *always*

Direction – *sink*

src

```
video/x-h265:
      framerate: [ 0/1, 2147483647/1 ]
          width: [ 1, 2147483647 ]
         height: [ 1, 2147483647 ]
  stream-format: byte-stream
      alignment: au
        profile: main
```

Presence – *always*

Direction – *src*

Properties

hardware

"hardware" gboolean

Enables hardware encoders

Flags: Read, Write

Default value: True

async-depth

"async-depth" guint

Depth of asynchronous pipeline. Range: 1 to 20

Flags: Read, Write

Default value: 4

target-usage

"target-usage" guint

1: Best quality, 4: Balanced, 7: Best speed. Range: 1 to 7

Flags: Read, Write

Default value: 4

rate-control

"rate-control" GstMsdkEncRateControl

Rate control method

Flags: Read, Write

Default value: cbr (1)

GstMsdkEncRateControl

Members

`cbr` (*1*) – Constant Bitrate
vbr (*2*) – Variable Bitrate
cqp (*3*) – Constant Quantizer
avbr (*4*) – Average Bitrate
la_vbr (*8*) – VBR with look ahead (Non HRD compliant)
icq (*9*) – Intelligent CQP
`vcm` (*10*) – Video Conferencing Mode (Non HRD compliant)
`la_icq` (*11*) – Intelligent CQP with LA (Non HRD compliant)
`la_hrd` (*13*) – HRD compliant LA
`qvbr` (*14*) – VBR with CQP

bitrate

```
"bitrate" guint
```

Bitrate in kbit/sec. Range: 1 to 2048000

Flags: Read, Write

Default value: 2048

max-frame-size

```
"max-frame-size" guint
```

Maximum possible size (kb) of any compressed frames (0: auto-calculate). Range: 0 to 65535

Flags: Read, Write

Default value: 0

max-vbv-bitrate

```
"max-vbv-bitrate" guint
```

Maximum bitrate (kbit/sec) at which data enters the video buffering verifier (0: auto-calculate). Range: 0 to 65535

Flags: Read, Write

Default value: 0

accuracy

```
"accuracy" guint
```

The AVBR accuracy per one tenth of one percent. Range: 0 to 65535

Flags: Read, Write

Default value: 0

convergence

```
"convergence" guint
```

The AVBR convergence per 100 frames. Range: 0 to 65535

Flags: Read, Write

Default value: 0

rc-lookahead

```
"rc-lookahead" guint
```

Number of frames to look ahead for rate control. Range: 0 to 100

Flags: Read, Write

Default value: 0

qpi

```
"qpi" guint
```

Constant quantizer for I frames (0 unlimited). Also used as ICQQuality or QVBRQuality for different RateControl methods. Range: 0 to 51

Flags: Read, Write

Default value: 0

qpp

```
"qpp" guint
```

Constant quantizer for P frames (0 unlimited). Range: 0 to 51

Flags: Read, Write

Default value: 0

qpb

```
"qpb" guint
```

Constant quantizer for B frames (0 unlimited) Range: 0 to 51

Flags: Read, Write

Default value: 0

gop-size

```
"gop-size" guint
```

GOP Size. Range: 0 to 2147483647

Flags: Read, Write

Default value: 256

ref-frames

```
"ref-frames" guint
```

Number of reference frames. Range: 0 to 2147483647

Flags: Read, Write

Default value: 1

i-frames

`"i-frames" guint`

Number of I frames between IDR frames. Range: 0 to 2147483647

Flags: Read, Write

Default value: 0

b-frames

`"b-frames" guint`

Number of B frames between I and P frames. Range: 0 to 2147483647

Flags: Read, Write

Default value: 0

---

msdkh264dec

This element is an Intel® Media SDK H.264 decoder.

Note	• Only support Intel platform. • Refer to the EVA SDK Installation Guide to install OpenVino toolkits and Media SDK for GStreamer before using the MSDK plugin.

Example Using gst-launch

This pipeline decodes and displays an H.264 format image.

```
gst-launch-1.0 filesrc location=sample_720p.h264 ! h264parse !
msdkh264dec ! videoconvert ! xvimagesink
```

Hierarchy

```
GObject
    └──GInitiallyUnowned
        └──GstObject
            └──GstElement
                └──GstVideoDecoder
                    └──GstMsdkDec
                        └──GstMsdkH264Dec
```

Pad Templates

sink

```
video/x-h264:
          width: [ 1, 2147483647 ]
         height: [ 1, 2147483647 ]
  stream-format: byte-stream
      alignment: au
        profile: { (string)high, (string)main, (string)baseline,
(string)constrained-baseline }
```

Presence – *always*

Direction – *sink*

src

```
video/x-raw:
          format: { (string)NV12 }
       framerate: [ 0/1, 2147483647/1 ]
           width: [ 16, 2147483647 ]
          height: [ 16, 2147483647 ]
 interlace-mode: progressive
```

Presence – *always*

Direction – *src*

Properties

hardware

```
"hardware" gboolean
```

Enables hardware decoders

Flags: Read, Write

Default value: True

async-depth

```
"async-depth" guint
```

Depth of asynchronous pipeline. Range: 1 to 20

Flags: Read, Write

Default value: 4

---

msdkh265dec

This element is an Intel® Media SDK H.265 decoder.

Note	• Only support Intel platform. • Refer to the EVA SDK Installation Guide to install OpenVino toolkits and Media SDK for GStreamer before using the MSDK plugin.

Example Using gst-launch

This pipeline decodes and displays an H.265 format image.

```
gst-launch-1.0 filesrc location=4K_h265.mkv ! matroskademux ! h265parse !
avdec_h265 ! videoconvert ! xvimagesink sync=false
```

Hierarchy

```
GObject
      └──GInitiallyUnowned
            └──GstObject
                  └──GstElement
                        └──GstVideoDecoder
                              └──GstMsdkDec
                                    └──GstMsdkH265Dec
```

Pad Templates

sink

```
video/x-h265:
          width: [ 1, 2147483647 ]
         height: [ 1, 2147483647 ]
  stream-format: byte-stream
      alignment: au
        profile: main
```

Presence – *always*

Direction – *sink*

src

```
video/x-raw:
          format: { (string)NV12 }
      framerate: [ 0/1, 2147483647/1 ]
          width: [ 16, 2147483647 ]
         height: [ 16, 2147483647 ]
 interlace-mode: progressive
```

Presence – *always*

Direction – *src*

Properties

hardware

```
"hardware" gboolean
```

Enables hardware decoders

Flags: Read, Write

Default value: True

async-depth

```
"async-depth" guint
```

Depth of asynchronous pipeline. Range: 1 to 20

Flags: Read, Write

Default value: 4

---

admetawriter

Writes admeta data to stream with device information read from a packed text file. Also provides the ability to count frames on an input stream to add to the admeta frame id field.

Note	This must be the first plugin to add metadata to the stream, so it must be used before any other plugins that add to admeta data (admetadrawer, admetadebuger, advino, adrt)

Example Using gst-launch

```
gst-launch-1.0 videotestsrc ! admetawriter devinfo=dev1.txt count-
frame=TRUE ! videoconvert ! ximagesink
```

Hierarchy

```
GObject
   └── GInitiallyUnowned
         └── GstObject
               └── GstElement
                     └── AdMetaWriter
```

Pad Templates

sink

```
video/x-raw:
          format: {RGBx, xRGB, BGRx, xBGR, RGBA, ARGB, BGRA, ABGR, RGB, BGR,
AYUV, YUY2, YVYU, UYVY, I420, YV12, RGB16, RGB15, GRAY8, NV12, NV21,
GRAY16_LE, GRAY16_BE}
           width: [ 0, 2147483647 ]
          height: [ 0, 2147483647 ]
       framerate: [ 0/1, 2147483647/1 ]
```

Presence – *always*

Direction – *sink*

src

```
video/x-raw:
          format: {RGBx, xRGB, BGRx, xBGR, RGBA, ARGB, BGRA, ABGR, RGB, BGR,
AYUV, YUY2, YVYU, UYVY, I420, YV12, RGB16, RGB15, GRAY8, NV12, NV21,
GRAY16_LE, GRAY16_BE}
           width: [ 0, 2147483647 ]
          height: [ 0, 2147483647 ]
       framerate: [ 0/1, 2147483647/1 ]
```

Presence – *always*

Direction – *sink*

Properties

devinfo

"devinfo" gchararray

Provides the location to the device information text file. The format of the device information text file is as follows.

```
0
0
10.9.2.1.2
8088
0
ADLINK2
0
0
0
0
0
0
0
```

The following table lists the data definitions represented by each line of the device information text file.

Rows	Field name	Description	Sample value
1	dev_id	Vision Device host interface (ex. /dev/video0 or /dev/ttyUSB0)	1006

2	fw_version	Vision Device firmware version	3.v.0
3	ip_address	Host machine IP Address	11.0.0.1.2
4	kind	Vision device kind enum(OASYS defined)	Vision
5	mac_address	Host address	00:11:22:33:44:55
6	manufacturer	Vision Device manufacturer	ADLINK2
7	model	Vision Device model	NEONi1000
8	port	Connection port	8088
9	protocol	ProtocolKind enum describing how the device communicates	V4L2
10	serial	Vision Device serial identifier	AD0123456
11	status	DeviceStatus enum(OASYS defined)	OK
12	stream_id	Stream publisher ID	Stream1
13	uri	Video Interface URI (rtsp://xx/h264)	0

Flags: Read / Write

Default value: NULL

count-frame

"count-frame" gboolean

Provides an option to add frame number counting to the stream. Some src plugins will use offset data to automatically generate this information.

Flags: Read / Write

Default value: false

admetadrawer

Draws admeta data to the stream.

Example Using gst-launch

```
gst-launch-1.0 videotestsrc ! admetadrawer ! videoconvert ! ximagesink
```

Hierarchy

```
GObject
   └── GInitiallyUnowned
         └── GstObject
               └── GstElement
                     └──AdMetaDrawer
```

Pad Templates

sink

```
video/x-raw:
          format: {RGBx, xRGB, BGRx, xBGR, RGBA, ARGB, BGRA, ABGR, RGB, BGR,
AYUV, YUY2, YVYU, UYVY, I420, YV12, RGB16, RGB15, GRAY8, NV12, NV21,
GRAY16_LE, GRAY16_BE}
           width: [ 0, 2147483647 ]
          height: [ 0, 2147483647 ]
       framerate: [ 0/1, 2147483647/1 ]
```

Presence – *always*

Direction – *sink*

src

```
video/x-raw:
         format: {RGBx, xRGB, BGRx, xBGR, RGBA, ARGB, BGRA, ABGR, RGB, BGR,
AYUV, YUY2, YVYU, UYVY, I420, YV12, RGB16, RGB15, GRAY8, NV12, NV21,
GRAY16_LE, GRAY16_BE}
          width: [ 0, 2147483647 ]
         height: [ 0, 2147483647 ]
      framerate: [ 0/1, 2147483647/1 ]
```

Presence – *always*

Direction – *src*

---

admetadumper

Prints some admeta data to the console.

The following is an example of printing metadata fields.

"========== Adlink batch metadata =========="

"Device[***$idx***] ID: ***$stream_id***"

"Frame[***$idx***] Num:***$frame_id*** (***$width***, ***$height***, ***$depth***, ***$channels***) for Device ***$device_idx***"

"Class[***$index***] label:***$label*** prop: ***$prob***"

"Box[***$obj_id***] (***$x1***,***$y1***)-(***$x2***,***$y2***) label:***$obj_label*** prob:***$prob***"

"Seg[***$s_idx***] (***$w***,***$h***) label id:***$label_id*** label:***$label***"

"==="

- Device information is from DeviceInfoData structure. The ***$idx*** is the order of the input media device.
- Frame information is from VideoFrameData structure. The ***$idx*** is the order of the input video frame.
- Class information is from ClassificationResult structure.
- Box information is from DetectionBoxResult structure.
- Seg information is from SegmentResult structure. The ***$s_idx*** is the order of the segment number. The (***$w***,***$h***) are the width and height of the inference.

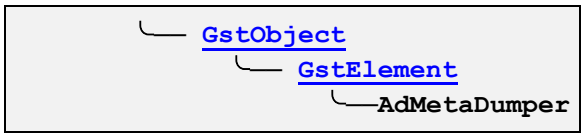
For more details, refer to ADLINK EVA SDK Programming Guide.

Example Using gst-launch

```
gst-launch-1.0 videotestsrc ! admetadumper ! videoconvert ! ximagesink
```

Hierarchy

```
GObject
   └── GInitiallyUnowned
```



Pad Templates

sink

`ANY`

Presence – *always*

Direction – *sink*

src

`ANY`

Presence – *always*

Direction – *src*

Properties

segment_num

"segment" guint

Number of segmentation results that will be printed on the Terminal. Range: 0 to 64.

Flags: Read, Write

Default value: 10

---

admetadebuger

Writes fake admeta data to the stream, but is currently only able to write box or class metadata.

Only one inference result can be written at a time. To write multiple inference results to metadata, the element can be duplicated as many times as necessary.

Example Using gst-launch

```
gst-launch-1.0 videotestsrc ! admetadebuger type=box x1=0.2 y1=0.4 x2=0.6
y2=0.8 class="foobar"  ! videoconvert ! ximagesink
```

Hierarchy

```
GObject
   └── GInitiallyUnowned
         └── GstObject
               └── GstElement
                     └──AdMetaDebuger
```

Pad Templates

sink

```
ANY
```

Presence – *always*

Direction – *sink*

src

```
ANY
```

Presence – *always*

Direction – *src*

Property

type

```
"type" DataTypePattern
```

Type of written inference data.

Flags: Read, Write

Default value: 0, "class"

DataTypePattern

Members

(0): class - Class data (id, class, prob)
(1): box - Box data (id, x1, y1, x2, y2, prob)

id

```
"id" guint
```

Id of inference data.

Flags: Read, Write

Default value: 0

class

```
"class" gchararray
```

The class name of the inference result. If “class” is NULL, the class will be the string “test” with its suffix as a random integer between 0 and 1000.

Flags: Read, Write

Default value: NULL

x1

`"x1" gfloat`

The X-axis coordinate is at the left side of the bounding box. It should be a ratio of the width, so the value is between 0 and 1.

Flags: Read, Write

Default value: 0.0

y1

`"y1" gfloat`

The Y-axis coordinate is in the top of the bounding box. It should be a ratio of the width, so the value is between 0 and 1

Flags: Read, Write

Default value: 0.0

x2

`"x2" gfloat`

The X-axis coordinate is at the right side of the bounding box. It should be a ratio of the width, so the value is between 0 and 1

Flags: Read, Write

Default value: 1.0

y2

`"y2" gfloat`

The Y-axis coordinate is at the bottom of the bounding box. It should be a ratio of the width, so the value is between 0 and 1.

Flags: Read, Write

Default value: 1.0

prob

`"prob" gfloat`

Probability of inference result between 0 and 1.

Flags: Read, Write

Default value: 1.0

---

nvv4l2decoder

The OSS Gst-nvvideo4linux2 plugin leverages the hardware decoding engines on Jetson and DGPU platforms by interfacing with libv4l2 plugins. It supports H.264, H.265, JPEG and MJPEG formats. The plugin accepts an encoded bitstream and NVDEC hardware engine to decode the bitstream. The decoded output is in NV12 format.

For more details, refer to the official documentation at: https://docs.nvidia.com/metropolis/deepstream/4.0.2/plugin-manual/index.html#page/DeepStream_Plugin_Manual%2Fdeepstream_plugin_details.02.12.html%23wwpID0E0NM0HA

Note	Only support NVIDIA platform.

Example Using gst-launch

This pipeline decodes an H.264 video file.

```
gst-launch-1.0 filesrc location=sample_1080p_h264.mp4 ! qtdemux !
h264parse ! nvv4l2decoder ! nvvideoconvert ! xvimagesink
```

---

nvv4l2h264enc

The OSS Gst-nvvideo4linux2 plugin leverages the hardware accelerated encoding engine available on Jetson and dGPU platforms by interfacing with libv4l2 plugins. The plugin accepts RAW data in I420 format. It uses the NVENC hardware engine to encode RAW input. Encoded output is in elementary bitstream supported formats.

For more details, refer to the official documentation at:
https://docs.nvidia.com/metropolis/deepstream/4.0.2/plugin-manual/index.html#page/DeepStream_Plugin_Manual%2Fdeepstream_plugin_details.02.12.html%23wwpID0E0YL0HA

Note	Only support NVIDIA platform.

Example Using gst-launch

This pipeline encodes video to the H.264 format.

```
gst-launch-1.0 videotestsrc num-buffers=1000 ! "video/x-
raw,format=(string)I420" ! nvvideoconvert ! "video/x-raw(memory:NVMM)" !
nvv4l2h264enc ! h264parse ! matroskamux ! filesink location=x1.mkv
```

nvv4l2h265enc

The OSS Gst-nvvideo4linux2 plugin leverages the hardware accelerated encoding engine available on Jetson and dGPU platforms by interfacing with libv4l2 plugins. The plugin accepts RAW data in I420 format. It uses the NVENC hardware engine to encode RAW input. Encoded output is in elementary bitstream supported formats.

For more details, refer to the official documentation at:
https://docs.nvidia.com/metropolis/deepstream/4.0.2/plugin-manual/index.html#page/DeepStream_Plugin_Manual%2Fdeepstream_plugin_details.02.12.html%23wwpID0E0YL0HA

Note	Only support NVIDIA platform.

Example Using gst-launch

This pipeline encodes video to the H.265 format.

```
gst-launch-1.0 videotestsrc num-buffers=2000 ! "video/x-
raw,format=(string)I420" ! nvvideoconvert ! "video/x-raw(memory:NVMM)" !
nvv4l2h265enc ! h265parse ! matroskamux ! filesink
location=nvv4l2h265enc.mkv
```

Image Analytic Plugins

The following table lists the element of the supported image analytic plugins.

Element Name	Plugin Name	Vendor	Version	Description
adtranslator	adtranslator	ADLINK	2.0	Translates an inference result into human readable format
adrt	adrt	ADLINK	2.0	Initializes TensorRT inference engine plugins to infer a data stream
advino	advino	ADLINK	2.0	Initializes OpenVINO inference engine plugins to infer a data stream
advideobatch	adbatch	ADLINK	2.0	Batch inputs video streams together
adsplitbatch	adbatch	ADLINK	2.0	Splits batched streams into a single stream

Element Descriptions

This section describes each element of the image analytic plugin in greater detail.

adtranslator

This element implements a buffer list chain function. Previous elements should push a buffer list into this element.

The last buffer the list will be seen as an inference result, passing buffer and dimension information assigned by the user to the translator function.

As the translator function is user-selected, it is important that the user understand the format of the inference result in order to avoid unexpected results from passing incorrect dimensions or using the wrong translator.

Some neural networks require users to provide a model input size to convert to the correct coordinates.

Examples Using gst-launch

Using the OpenVINO inference

```
gst-launch-1.0 videotestsrc ! advino model=classification.xml device=GPU !
adtranslator dims=1,1000 topology=Classifier label=label.txt ! ximagesink
```

```
gst-launch-1.0 videotestsrc ! advino model=ssd.xml device=GPU ! adtranslator
dims=1,1,200,7 topology=ssd label=label.txt ! ximagesink
```

```
gst-launch-1.0 videotestsrc ! advino model=segment.xml device=GPU !
adtranslator dims=1,1024,2048 topology=segment label=label.txt ! ximagesink
```

```
gst-launch-1.0 videotestsrc ! advino model=yolov3.xml device=GPU !
adtranslator dims=1,255,26,26,1,255,52,52,1,255,13,13 input_width=416
input_height=416 topology=yolov3 engine-type=openvino label=label.txt !
ximagesink
```

Using the TensorRT inference

```
gst-launch-1.0 videotestsrc ! adrt model=mobilenetv2.engine batch=1
scale=0.017 ! adtranslator topology=classifier dims=1,1000
label=labels.txt ! ximagesink
```

```
gst-launch-1.0 videotestsrc ! adrt model=ssd_mobilenetv2_tx2.engine batch=1
device=0 scale=0.0078 mean="0 0 0" ! adtranslator topology=ssd
dims=1,1,100,7 label=ssd_mobilenetv2_RT_labels.txt ! ximagesink
```

```
gst-launch-1.0 videotestsrc ! adrt model=yolov3_608_tx2.engine scale=0.004
mean="0 0 0" device=0 batch=1 ! adtranslator label=yolo_RT_labels.txt
topology=yolov3 dims=1,255,19,19,1,255,38,38,1,255,76,76 input_width=608
engine-type=tensorrt ! ximagesink
```

```
gst-launch-1.0 videotestsrc ! adrt model=FCN-Alexnet-Cityscapes-
SD_tx2.engine scale=1.0 mean="0 0 0" device=0 batch=1 ! adtranslator
label=cityscapes-labels.txt topology=fcn dims=1,3,512,1024,21,10,26 !
```

```
ximagesink
```

The different topologies have their definition of the dimension. The following table lists the dimension format corresponding to the topology.

Topology	Dimension
classifier	dims = n,X • n: batch size • X: output dimension, commonly equal to object classes
ssd	dims = n,1, X,7 • n: batch size • X: Number of detectable objects; depend on training setting of model. Common values are 100,200,400
yolov3	dims = n,a,x,x,n,a,y,y,n,a,z,z • n: batch size • a: a value related to number of object classes (X). Calculated by a = (X + 5) x 3. i.e : 80 classes mean a = (80 + 5) x3 = 255 • x,y,z : value depend on output layer of Yolov3 model, normal value is: input_layer_dimension/32, input_layer_dimension/16, input_layer_dimension/8. i.e : input layer is 416x416 so x,y,z is 13,26,52. Order depends on models.
segment	dims = n,h,w dims = n,c,h,w (get highest probability in each c channel for every pixel of width and height) • n: batch size • c: number of channels of output image • h: height of output layer dimension • w: width of output layer dimension
fcn	dims = n,c,h,w,X,a,b • n: batch size • c: number of channels of output image • h: height of output image • w: width of output image • X: number of object classes • a: height of output layer dimension • b: width of output layer dimension

Hierarchy

```
GObject
      └──GInitiallyUnowned
            └──GstObject
                  └──GstElement
                        └──Adtranslator
```

Pad Templates

sink

```
ANY
```

Presence – *always*

Direction – *sink*

src

```
ANY
```

Presence – *always*

Direction – *src*

Properties

topology

```
"topology" AdAITopologyPattern
```

Topology of neural network:

- classifier
- yolov3
- ssd
- segment
- fcn

Flags: Read, Write

Default value: 0, "none"

AdAITopologyPattern

Members

(0): none
(1): classifier
(2): yolov3
(3): ssd
(4): segment
(6): fcn

Engine-type

`"engine-type" AdAIEnginePattern`

Engine of inference element:

- openvino (1)
- tensorrt (2)

Flags: Read, Write

Default value: 0, "none"

Note	This parameter only works with Yolov3 topology. Other topology can use with both types of engines so no need to set.

AdAIEnginePattern

Members

(0): none - Non-specific engine
(1): openvino - OpenVINO inference engine
(2): tensorrt - TensorRT inference engine

label

```
"label" gchararray
```

The path of a label file.

The label file content needs to conform to the following format:

- Each line is one class
- Class can be string or number

Flags: Read, Write

Default value: NULL

dims

```
"dims" gchar
```

Dimension of inference result, separated by a comma.

Flags: Read, Write

Default value: NULL

input-width

```
"input-width" gint
```

Neural network's input width. If the property isn't assigned a value, it is the same as input-height. Required by Yolo model.

Flags: Read, Write

Default value: 0

Input-height

```
"input-height" gint
```

Neural network's input height. If the property isn’t assigned a value, it is the same as input-width. Required by Yolo model.

Flags: Read, Write

Default value: 0

---

adrt

Initializes the TensorRT inference engine plugins to infer a data stream with the following parameters.

1. TensorRT optimized engine model path
2. Scale (must be a float number)
3. Mean (must be a string of 3 numbers representing a vector of mean values used in training)

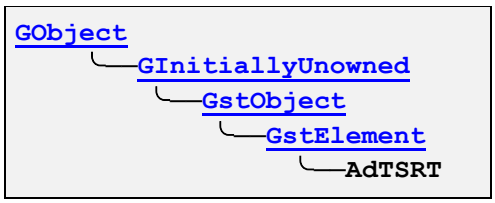
For more details about converting a model for TensorRT, refer to the chapter Convert for TensorRT Model.

Note	Only support NVIDIA platform.

Example Using gst-launch

```
gst-launch-1.0 videotestsrc pattern=snow ! adrt model=classification.engine scale=0.017 mean="103.94 116.78 123.68" batch=1 device=0 !fakesink
```

Hierarchy



Pad Templates

sink

```
video/x-raw:
         format: {BGR}
          width: [ 0, 2147483647 ]
         height: [ 0, 2147483647 ]
      framerate: [ 0/1, 2147483647/1 ]
video/x-raw(ANY):
         format: {BGR}
          width: [ 0, 2147483647 ]
         height: [ 0, 2147483647 ]
      framerate: [ 0/1, 2147483647/1 ]
```

Presence – *always*

Direction – *sink*

src

```
video/x-raw:
         format: {BGR}
          width: [ 0, 2147483647 ]
         height: [ 0, 2147483647 ]
      framerate: [ 0/1, 2147483647/1 ]
video/x-raw(ANY):
         format: {BGR}
          width: [ 0, 2147483647 ]
         height: [ 0, 2147483647 ]
      framerate: [ 0/1, 2147483647/1 ]
```

Presence – *always*

Direction – *sink*

Properties

model

"model" gchararray

The path where the TensorRT optimized engine is located.

Flags: Read / Write

Default value: NULL

scale

```
"scale" gfloat
```

Sets the scale needed to apply to input data. Scale is to normalize the input to corresponding range. Please refer to https://en.wikipedia.org/wiki/Feature_scaling

Flags: Read / Write

Default value: 1.0

mean

```
"mean" gchararray
```

Sets the mean vector value needed to apply to input data. The mean value of the dataset is the mean value of the pixels of all the images across all the color channels (e.g. RBG). Grey scale images will have just one mean value and color images like ImageNet will have 3 mean values.

Flags: Read / Write

Default value: "103.94 116.78 123.68"

device

```
"device" gchararray
```

Selects which GPU will be used to inference, use GPU index start from 0

Flags: Read / Write

Default value: "0"

batch

`"batch" guit`

Batch size of input to inference(number of image to inference at same time) need at least 1

Flags: Read / Write

Default value: 1

---

advino

Initializes the OpenVINO inference engine plugins to infer a data stream with following parameters.

1. OpenVINO optimized engine Model path
2. Inference device type: (1) CPU<default>; (2) GPU; (3) VPU; (4) FPGA; (5) MYRIAD (Reference: OpenVINO support devices link)

Some MYRIAD boards contain multiple MYRIAD cores, so users can inference a specific core or select one via the OpenVINO inference scheduler.

Below command can query the available devices.

```
$ python3 -c "from openvino.inference_engine import IECore;
print(IECore().available_devices)"

['CPU', 'GPU', 'HDDL', 'MYRIAD.7.1-ma2480', 'MYRIAD.7.2-ma2480']
```

This element pushes a buffer list where the last buffer is the inference result raw data needed to be analyzed by next element, and the other result is the original incoming frame buffer.

It will have unexpected result if you pipe an element that does not implement a buffer list chain function.

For more details about converting a model for OpenVINO, refer to the chapter Convert for OpenVINO Model.

Note	Only support Intel platform.

Example Using gst-launch

```
gst-launch-1.0 videotestsrc pattern=snow ! advino model=yolov3.xml !
fakesink
```

Hierarchy

```
GObject
      └──GInitiallyUnowned
            └──GstObject
                  └──GstElement
                        └──AdVINO
```

Pad Templates

sink

```
video/x-raw:
          format: { BGR }
           width: [ 0, 2147483647 ]
          height: [ 0, 2147483647 ]
      framerate: [ 0/1, 2147483647/1 ]
video/x-raw(ANY):
          format: { BGR }
           width: [ 0, 2147483647 ]
          height: [ 0, 2147483647 ]
      framerate: [ 0/1, 2147483647/1 ]
```

Presence – *always*

Direction – *sink*

src

```
video/x-raw:
          format: { BGR }
           width: [ 0, 2147483647 ]
          height: [ 0, 2147483647 ]
      framerate: [ 0/1, 2147483647/1 ]
video/x-raw(ANY):
          format: { BGR }
           width: [ 0, 2147483647 ]
          height: [ 0, 2147483647 ]
      framerate: [ 0/1, 2147483647/1 ]
```

Presence – *always*

Direction – *sink*

Properties

model

`"model" gchararray`

The path where the OpenVINO optimized engine is located.

Flags: Read / Write

Default value: NULL

device

`"device" gchararray`

Deep learning device, ex: CPU, GPU or MYRIAD. User may also using following command to list available devices. "python3 -c 'from openvino.inference_engine import IECore; print(IECore().available_devices)"'

Flags: Read / Write

Default value: "CPU"

batch

`"batch" guit`

Batch size of input to inference(number of image to inference at same time) need at least 1.

Flags: Read / Write

Default value: 1

scale

`"scale" gfloat`

Sets the scale needed to apply to input data.

Flags: Read / Write

Default value: 1.0

mean

`"mean" gchararray`

Sets the mean vector value needed to apply to input data.

Flags: Read / Write

Default value: NULL

---

advideobatch

This element batches multiple streams to a single vertical stream fitting the data format of the inference engine and collects all admeta data of the individual streams into batch meta data.

Example Using gst-launch

```
gst-launch-1.0 advideobatch name=mix ! videoconvert ! ximagesink \
 videotestsrc pattern="red" ! admetawriter devinfo=dev1.txt count-
frame=TRUE ! mix.sink_0 \
 videotestsrc pattern="blue" ! admetawriter devinfo=dev1.txt count-
frame=TRUE ! mix.sink_1
```

Hierarchy

```
GObject
   └── GInitiallyUnowned
         └── GstObject
               └── GstElement
                     └──AdVideoBatch
```

Pad Templates

sink_%u

```
video/x-raw:
          format: { BGR }
           width: [ 0, 2147483647 ]
          height: [ 0, 2147483647 ]
       framerate: [ 0/1, 2147483647/1 ]
```

Presence – *always*

Direction – *sink*

src

```
video/x-raw:
          format: { BGR }
           width: [ 0, 2147483647 ]
          height: [ 0, 2147483647 ]
      framerate: [ 0/1, 2147483647/1 ]
```

Presence – *always*

Direction – *src*

---

adsplitbatch

A batched stream is a stream in which each of its frames is the combination of several smaller frames from several sources. This element will split a batched stream frame back to its original component frames and display it sequentially as a single stream.

This element must run after an advideobatch element with admeta data in its stream.

Note	The fps of the stream at sink pad will be equal to the number of batched streams multiplied with the fps of the src stream. For example, a stream of 30fps with a batch of 4 will generate a stream of 120fps.

Example Using gst-launch

```
gst-launch-1.0 advideobatch name=mix ! videoconvert ! adsplitbatch !
videoconvert ! ximagesink \
 videotestsrc pattern="red" ! admetawriter devinfo=dev1.txt count-
frame=TRUE ! mix.sink_0 \
 videotestsrc pattern="blue" ! admetawriter devinfo=dev1.txt count-
frame=TRUE ! mix.sink_1 \
```

Hierarchy

```
GObject
   └── GInitiallyUnowned
         └── GstObject
               └── GstElement
                     └──AdSplitBatch
```

Pad Templates

sink

```
video/x-raw:
          format: {BGR}
           width: [ 0, 2147483647 ]
          height: [ 0, 2147483647 ]
      framerate: [ 0/1, 2147483647/1 ]
```

Presence – *always*

Direction – *sink*

src

```
video/x-raw:
          format: {BGR}
           width: [ 0, 2147483647 ]
          height: [ 0, 2147483647 ]
      framerate: [ 0/1, 2147483647/1 ]
```

Presence – *always*

Direction – *src*

Connector Plugins

The following table lists the element of the supported connector plugins.

Element Name	Plugin Name	Vendor	Version	Description
adedgesrc	adedge	ADLINK	2.0	Subscribes to ADLINK edge river and receives data.
adedgesink	adedge	ADLINK	2.0	Publishes data to ADLINK river.

Element Descriptions

This section describes each element of the connector plugins in greater detail.

adedgesrc

Subscribes to ADLINK edge river and receives data.

1. Set data model (Tag Group)
2. Convert the edge TagGroup to relative tag (metadata)
3. Listen to the river and pass the data downstream in the pipeline
4. Exchange color format from edge river to pipeline
5. Based on VideoFrameData, synchronize the rest of the data in the river

Note	Note that there exists a "***blocksize***" property that inherits from GstBaseSrc. Set this value at least greater than the data to be pushed. For example, push out an RGB pixel format with (width, height) = (640, 480). At least set the blocksize=921600 (640x480x3).

Example Using gst-launch

```
gst-launch-1.0 adedgesrc ! videoconvert ! ximagesink
```

Hierarchy

```
GObject
      └──GInitiallyUnowned
            └──GstObject
                  └──GstElement
                        └──GstBaseSrc
                              └──GstPushSrc
                                    └──AdEdgeSrc
```

Pad Templates

src

```
video/x-raw:
          format: { I420, NV12, BGR }
           width: [ 1, 2147483647 ]
          height: [ 1, 2147483647 ]
       framerate: [ 0/1, 2147483647/1 ]
```

Presence – *always*

Direction – *src*

Properties

taggroups

```
"taggroups" gchararray
```

Defines the tag groups' name; for example: taggroups=VideoFrameData.

If there is more than one tag group used, use the '+' delimiter to separate them. For example: taggroups=VideoFrameData+ClassificationData.

- VideoFrameData: publishes image data to edge river
- DetectionBoxData: publishes detection inference results to edge river
- SegmentationData: publishing segmentation inference result to edge river.
- ClassificationData: publishes classification inference results to edge river
- DeviceInfoData: publishes camera device information to edge river

VideoFrameData is the required tag group for synchronization.

Flags: Read, Write

Default value: NULL

Note	1. adedgesrc will synchronize all separated tag groups based on the frame from the river. If adedgesink does not publish any frame data to the river, adedgesrc cannot trigger the synchronization process. 2. Setting the property blocksize derived from GstBaseSrc bigger than real image data size is recommended. 3. Even if the Run Length Encoding is used in plugin adedge for segmentation results, it is strongly recommended not to transmit high dimensional inference segmentation data.

adedgesink

Publishes data to ADLINK river.

1. Set data model
2. Convert the tag(metadata) to the relative edge TagGroups
3. Receive data from upstream and publish to the river
4. Exchange color format from pipeline to edge river

Example Using gst-launch

```
gst-launch-1.0 videotestsrc ! adedgesink taggroups=VideoFrameData
```

Hierarchy

```
GObject
    └──GInitiallyUnowned
        └──GstObject
            └──GstElement
                └──GstBaseSink
                    └──GstVideoSink
                        └──AdEdgeSink
```

Pad Templates

sink

```
video/x-raw:
         format: { I420, NV12, BGR }
          width: [ 1 , 2147483647 ]
         height: [ 1 , 2147483647 ]
      framerate: [ 0/1, 2147483647/1 ]
image/jpeg:
          width: [ 16, 65535 ]
         height: [ 16, 65535 ]
      framerate: [ 0/1, 2147483647/1 ]
     sof-marker: { (int)0, (int)1, (int)2, (int)4, (int)9 }
```

Presence – *always*

Direction – *sink*

Properties

taggroups

`"taggroups" gchararray`

Defines the tag groups' name; for example: taggroups=VideoFrameData.

If there is more than one tag group used, use the '+' delimiter to separate them. For example: taggroups=VideoFrameData+ClassificationData.

- VideoFrameData: publishes image data to edge river.
- DetectionBoxData: publishes detection inference result to edge river.
- SegmentationData: publishing segmentation inference result to edge river.
- ClassificationData: publishes classification inference result to edge river.
- DeviceInfoData: publishes camera device information to edge river.

adedgesink should set taggroups=VideoFrameData for publishing frame data, otherwise adedgesrc will not pass any frame data downstream.

Flags: Read, Write

Default value: NULL

Convert for OpenVINO Model

Model Optimizer Developer Guide

Model Optimizer is a cross-platform command-line tool that facilitates the transition between the training and deployment environment, performs static model analysis, and adjusts deep learning models for optimal execution on end-point target devices. Refer to the official documentation at https://docs.openvinotoolkit.org/latest/_docs_MO_DG_Deep_Learning_Model_Optimizer_DevGuide.html

Converting a Caffe* Model

For more details of converting a Caffe* Model, refer to the official documentation at: https://docs.openvinotoolkit.org/latest/_docs_MO_DG_prepare_model_convert_model_Convert_Model_From_Caffe.html

Converting a TensorFlow* Model

For more details of converting a TensorFlow* Model, refer to the official documentation at: https://docs.openvinotoolkit.org/latest/_docs_MO_DG_prepare_model_convert_model_Convert_Model_From_TensorFlow.html

Converting an ONNX* Model

For more details of converting a ONNX* Model, refer to the official documentation at: https://docs.openvinotoolkit.org/latest/_docs_MO_DG_prepare_model_convert_model_Convert_Model_From_ONNX.html

Converting a Tensorflow YoloV3 Model

For more details of converting a Tensorflow YoloV3 Model, refer to the official documentation at: https://docs.openvinotoolkit.org/latest/openvino_docs_MO_DG_prepare_model_convert_model_tf_specific_Convert_YOLO_From_Tensorflow.html

Convert for TensorRT Model

Basic understanding of Deep Learning Neural network is required.

It is very important that each GPU architecture can only use the converted engine of the corresponding Compute Capability (CC) (https://en.wikipedia.org/wiki/CUDA) For example, TX2 CC is 6.2 so engine converted on TX2 can only run on other device with CC 6.2; they cannot run on other PASCAL GPU like P1000 which has CC 6.1 or Tesla P100 which has CC 6.0

Please download and install the TensorRT command-line wrapper tool – "trtexec" from https://github.com/NVIDIA/TensorRT/tree/master/samples/opensource/trtexec

If TensorRT is installed, trtexec is prebuilt and installed on the following path.

```
/usr/src/tensorrt/bin/trtexec
```

Build TensorRT engine parameters

TensorRT supports the ONNX model, the Caffe model, and the UFF model.

The following tables list the parameter of the three models in trtexec tool.

ONNX model

Parameters	Meaning
--onnx=<file>	Path to ONNX model
--output=<name>[,<name>]*	Output layer name, ONNX not required

Caffe model

Parameters	Meaning
--model=<file>	Path to Caffe weight
--deploy=<file>	Path to Caffe prototxt
--output=<name>[,<name>]*	Output layer name, required

UFF model

Parameters	Meaning
--uff=<file>	Path to UFF model file
--uffInput=<name>,X,Y,Z	Input layer and dimension, required
--output=<name>[,<name>]*	Output layer name, required
--uffNHWC	If specified, the input X,Y,Z will be used as H,W,C Default is C,H,W

The following table lists the build options in trtexec tool.

Build options

Parameter	Meaning
--calib=<file>	Read INT8 calibration cache file, required if --int8
--explicitBatch	Use explicit batch sizes when building the engine (default = implicit)
--fp16	Enable fp16 mode (default = disabled)
--int8	Run in int8 mode (default = disabled)
--maxBatch=<int>	Set max batch size and build an implicit batch engine (default = 1)
--plugins=<file>	Plugin library (.so) to load (can be specified multiple times) This is customized plugins which contained unsupported layers, need to be in CUDA code
--saveEngine=<file>	Save the TensorRT engine as <file>
--workspace=N	Set workspace size to N megabytes (default = 16) Set as big as possible for maximum speed of converting.

Convert Tensorflow models

TensorRT does not support the Tensorflow model. The Tensorflow models must be converted to a UFF model or an ONNX model.

Convert Tensorflow to UFF

For more details of converting a TensorFlow model stream to a UFF mode, refer to the official documentation at:
https://docs.nvidia.com/deeplearning/tensorrt/api/python_api/uff/uff.html

Convert Tensorflow to ONNX

For more details of converting a TensorFlow model stream to a UFF mode, refer to the official documentation at**:** https://github.com/onnx/tensorflow-onnx

Samples

Convert Yolov3.onnx to FP32 or FP16 engine file

Please follow this link to convert yolo to onnx: https://github.com/jkjung-avt/tensorrt_demos/blob/master/yolo/yolo_to_onnx.py

More information can be found in: https://github.com/jkjung-avt/tensorrt_demos/tree/master/yolo

FP32:

```
/usr/src/tensorrt/bin/trtexec --onnx=yolov3.onnx --workspace=3000 --
maxBatch=4 --verbose --saveEngine=yolov3.engine
```

FP16:

```
/usr/src/tensorrt/bin/trtexec --onnx=yolov3.onnx --workspace=3000 --
maxBatch=4 --fp16 --verbose --saveEngine=yolov3_fp16.engine
```

Explain:

Parameter	Meaning
--onnx=yolov3.onnx	Load onnx file path
--workspace=3000	Set GPU memory in MB to allow TensorRT to use to build engine, Best to around 80% of all memory
--maxBatch=4	Build engine will allow maximum input batch size =4
--saveEngine=yolov3.engine	save the final engine to yolov3.engine file
--verbose	display more information
--fp16	set model to fp16 precision
--buildOnly	(Optional) Reduce the 10s for testing model, only build and save engine

Convert googlenet caffe model to FP32 or FP16 engine file

Download model from https://github.com/BVLC/caffe/tree/master/models/bvlc_googlenet.

Caffemodel contains two files: prototxt file and caffemodel file. To inference, we normally require the deploy.prototxt file.

When converting the Caffe model, we need to input the output layer name. In the above link, the input layer is "Input" and the output layer is "prob"

FP32:

```
/usr/src/tensorrt/bin/trtexec --model=bvlc_googlenet.caffemodel --
deploy=deploy.prototxt --output=prob --workspace=3000 --maxBatch=4 --
verbose --saveEngine=googlenet.engine
```

FP16:

```
/usr/src/tensorrt/bin/trtexec --model=bvlc_googlenet.caffemodel --
deploy=deploy.prototxt --output=prob --workspace=3000 --fp16 --maxBatch=4 -
-verbose --saveEngine=googlenet_fp16.engine
```

Explain:

Parameter	Meaning
--deploy=deploy.prototxt	Load prototxt file path
--model=bvlc_googlenet.caffemodel	Load model file path
--output=prob	Set output layer name
--workspace=3000	Set GPU memory in MB to allow TensorRT to use to build engine, Best to around 80% of all memory
--maxBatch=4	Build engine will allow maximum input batch size =4
--saveEngine=googlenet.engine	save the final engine to googlenet.engine file
--fp16	set model to fp16 precision

Parameter	Meaning
--verbose	display more information
--buildOnly	(Optional) Reduce the 10s for testing model, only build and save engine

Convert ssd_inception_v2 uff model to FP32 or FP16 engine file

The models are generated from
https://github.com/NVIDIA/TensorRT/tree/master/samples/opensource/sampleUffSSD

When converting the UFF model, we need to input the output layer and input layer name. In the above link, the input layer is "Input" with dimension of 3x300x300 and the output layer is "NMS"

FP32:

```
/usr/src/tensorrt/bin/trtexec --uff=sample_ssd_relu6.uff --output=NMS --
uffInput=Input,3,300,300 --workspace=3000 --saveEngine=ssdv2.engine --
verbose --maxbatch=4
```

FP16:

```
/usr/src/tensorrt/bin/trtexec --uff=sample_ssd_relu6.uff --output=NMS --
uffInput=Input,3,300,300 --workspace=3000 --saveEngine=ssdv2_fp16.engine --
verbose --maxbatch=4 --fp16
```

Explain:

Parameter	Meaning
--uff	Load uff file path
--uffInput=Input,3,300,300	Set input layer name and input layer dimension
--output=NMS	Set output layer name
--workspace=3000	Set GPU memory in MB to allow TensorRT to use to build engine, Best to around 80% of all memory
--maxBatch=4	Build engine will allow maximum input batch size =4
--saveEngine=mobilenet_ssdv2.engine	save the final engine to ssdv2.engine file

Parameter	Meaning
--fp16	set model to fp16 precision
--verbose	display more information
--buildOnly	(Optional) Reduce the 10s for testing model, only build and save engine