How to Become a GStreamer Developer

August 5, 2016 • Laurent Pinchart

It's Monday morning, you're as fresh as a daisy, ready to finally test the V4L2 deinterlacer driver you've been working on for the past few months with GStreamer. With a cup of tea in one hand, a scone in the other one, and your keyboard in the third, you fire up your GStreamer test pipeline:

```
$ gst-launch-1.0 --gst-debug-no-color \
        videotestsrc pattern=ball num-buffers=120 ! \
        video/x-raw,format=NV12,width=1920,height=1080,framerate=60/1,interlace-mode=progressive !
\
        interlace field-pattern=0 ! \
        v4l2video0convert ! \
        queue ! \
        video/x-raw,format=NV12,width=1920,height=1080,interlace-mode=progressive ! \
        x264enc ! \
        h264parse ! \
        mp4mux ! \
        filesink location=fdp1.interlace.smpte.NV12-NV12.1920.1080.mp4
```

and get greeted by the following message (line numbers added for clarity):

```
1.  Setting pipeline to PAUSED ...
2.  Pipeline is PREROLLING ...
3.  ERROR: from element /GstPipeline:pipeline0/GstVideoTestSrc:videotestsrc0: Internal data flow
error.
4.  Additional debug info:
5.  gstbasesrc.c(2948): gst_base_src_loop (): /GstPipeline:pipeline0/GstVideoTestSrc:videotestsrc0:
6.  streaming task paused, reason not-negotiated (-4)
7.  ERROR: pipeline doesn't want to preroll.
8.  Setting pipeline to NULL ...
9.  Freeing pipeline ...
```

The tea and scone feel sorry for you for a split second, the time it takes for you to swallow them and free all your hands for the fight. This Monday morning will be long.

GStreamer Logging

Luckily for you GStreamer comes with an extensive logging and debugging infrastructure that could save your day (but don't bet on lunch though, miracles are not included in the package). Put your Rambo knife back in the drawer without hurting yourself, we will instead use the much more versatile weapon of GST_DEBUG.

The GST_DEBUG environment variable controls GStreamer logging. Its usage is documented in details in the **GStreamer basic tutorials**. When the variable is set on the command line, gst-launch will log to the standard output all messages matching the filter specified by GST_DEBUG. Filters are expressed as a comma-separated list of category:level pairs, with categories being free-formed strings defined by GStreamer elements, and levels integers ranging from 0 (no logging) to 5 (log all messages).

The first step of your root cause analysis should be to raise the default logging level from 1 (error messages) to 2 (warning messages) for all categories. Good doctors listen to their patients, there's no reason to ignore GStreamer's complains for the sole reason that it's a piece of software and not a human. If nothing else, it should remember your nice gesture and be thankful on the day of the singularity.

```
$ GST_DEBUG="*:2" gst-launch-1.0 --gst-debug-no-color \
        videotestsrc pattern=ball num-buffers=120 ! \
        video/x-raw,format=NV12,width=1920,height=1080,framerate=60/1,interlace-mode=progressive !
\
        interlace field-pattern=0 ! \
        v4l2video0convert ! \
        queue ! \
        video/x-raw,format=NV12,width=1920,height=1080,interlace-mode=progressive ! \
        x264enc ! \
        h264parse ! \
        mp4mux ! \
        filesink location=fdp1.interlace.smpte.NV12-NV12.1920.1080.mp4
```

The log is now slightly longer, and definitely more cryptic.

```
1.  Setting pipeline to PAUSED ...
2.  Pipeline is PREROLLING ...
3.  0:00:00.066795027   871     0x2ce51050 WARN           basetransform
gstbasetransform.c:1414:gst_base_transform_setcaps:<v4l2video0convert0> transform could not
transform video/x-raw, format=(string)NV12, width=(int)1920, height=(int)1080, framerate=
(fraction)30/1, interlace-mode=(string)interleaved, colorimetry=(string)bt601, pixel-aspect-ratio=
(fraction)1/1 in anything we support
4.  0:00:00.113798615   871     0x2ce51050 WARN           basetransform
gstbasetransform.c:1414:gst_base_transform_setcaps:<v4l2video0convert0> transform could not
transform video/x-raw, format=(string)NV12, width=(int)1920, height=(int)1080, framerate=
(fraction)30/1, interlace-mode=(string)interleaved, colorimetry=(string)bt601, pixel-aspect-ratio=
(fraction)1/1 in anything we support
5.  0:00:00.164307177   871     0x2ce51050 WARN           basetransform
```

```
gstbasetransform.c:1414:gst_base_transform_setcaps:<v4l2video0convert0> transform could not
transform video/x-raw, format=(string)NV12, width=(int)1920, height=(int)1080, framerate=
(fraction)30/1, interlace-mode=(string)interleaved, colorimetry=(string)bt601, pixel-aspect-ratio=
(fraction)1/1 in anything we support
6.   0:00:00.164786579   871     0x2ce51050 WARN                 basesrc
gstbasesrc.c:2947:gst_base_src_loop:<videotestsrc0> error: Internal data flow error.
7.   0:00:00.164814419   871     0x2ce51050 WARN                 basesrc
gstbasesrc.c:2947:gst_base_src_loop:<videotestsrc0> error: streaming task paused, reason not-
negotiated (-4)
8.   ERROR: from element /GstPipeline:pipeline0/GstVideoTestSrc:videotestsrc0: Internal data flow
error.
9.   0:00:00.165227941   871     0x2ce51050 WARN           basetransform
gstbasetransform.c:1414:gst_base_transform_setcaps:<v4l2video0convert0> transform could not
transform video/x-raw, format=(string)NV12, width=(int)1920, height=(int)1080, framerate=
(fraction)30/1, interlace-mode=(string)interleaved, colorimetry=(string)bt601, pixel-aspect-ratio=
(fraction)1/1 in anything we support
10.  Additional debug info:
11.  gstbasesrc.c(2947): gst_base_src_loop (): /GstPipeline:pipeline0/GstVideoTestSrc:videotestsrc0:
streaming task paused, reason not-negotiated (-4)
12.  ERROR: pipeline doesn't want to preroll.
13.  Setting pipeline to NULL ...
14.  0:00:00.165453182   871     0x2ce51050 WARN           basetransform
gstbasetransform.c:1414:gst_base_transform_setcaps:<v4l2video0convert0> transform could not
transform video/x-raw, format=(string)NV12, width=(int)1920, height=(int)1080, framerate=
(fraction)30/1, interlace-mode=(string)interleaved, colorimetry=(string)bt601, pixel-aspect-ratio=
(fraction)1/1 in anything we support
15.  Freeing pipeline ...
```

It's now time to resist the temptation to reach back for your Rambo knife. Take a deep breath, your fingers (or what remains of them if you tried to debug GStreamer on your own before reading this article) will thank you. Stay focussed with me for a minute, I will show you that those messages are not satanic incantations that forebode the impeding materialization of a demon (even if you run on FreeBSD).

Log Messages

GStreamer log messages start with a timestamp, process ID, thread ID, log level, category, source file and line, function and name of the object that issued the message. The first message is thus to be interpreted as follows.

Timestamp	0:00:00.066795027
Process ID	871
Thread ID	0x2ce51050
Log level	WARN
Category	basetransform
File name	gstbasetransform.c
Line number	1414
Function	gst_base_transform_setcaps
Object	<v4l2video0convert0>
Message	transform could not transform video/x-raw, format=(string)NV12, width=(int)1920, height=(int)1080, framerate=(fraction)30/1, interlace-mode=(string)interleaved, colorimetry=(string)bt601, pixel-aspect-ratio=(fraction)1/1 in anything we support

While the timestamp, process ID and thread ID are very useful to debug complex problems that involve multiple processes and/or threads, you can ignore them for now. The log level can be glanced at, but shouldn't surprise you in this case as you've requested GStreamer to output warning messages only. If you had forgotten that already, a second cup of tea might help (and good luck finding your keys when going back home at the end of the day).

The object provides more interesting information. GStreamer labels objects with with instance numbers, 'v4l2video0convert0' is the first instance of the element 'v4l2video0convert'. Our pipeline contains a single instance of that that element, we have now very likely narrowed down the issue to one element instance. Let's request more information from GStreamer.

```
$ gst-inspect-1.0 v4l2video0convert
Factory Details:
  Rank                     none (0)
  Long-name                V4L2 Video Converter
  Klass                    Filter/Converter/Video/Scaler
  Description              Transform streams via V4L2 API
  Author                   Nicolas Dufresne

Plugin Details:
  Name                     video4linux2
  Description              elements for Video 4 Linux
  Filename                 /usr/lib/gstreamer-1.0/libgstvideo4linux2.so
  Version                  1.9.1
  License                  LGPL
  Source module            gst-plugins-good
  Source release date      2016-06-06
  Binary package           GStreamer Good Plug-ins source release
  Origin URL               Unknown package origin

GObject
```

```
 +----GInitiallyUnowned
       +----GstObject
             +----GstElement
                   +----GstBaseTransform
                         +----GstV4l2Transform
                               +----v4l2video0convert
...
```

The v4l2video0convert element inherits the GstV4l2Transform class, itself inheriting the GstBaseTransform class. This matches the category, file name and function name from the log message. From the name of the GstBaseTransform class only we can guess that it implements support for elements that transform data, and this is confirmed by the **GStreamer documentation**.

You now have an entry point to the code base, both in terms of class and of code location (by file name and line number). Beware! This is no time to rush, you little Padawan. I see you, all eager to open your favourite text editor (which is obviously vim, as for all **real Jedis^Wprogrammers**), rejoicing at the expectation to get your hands dirty with the code. There is much more ancient lore to be learnt from the almighty log before you will be ready to dive into the arcane depths of the source code.

Let's look at the error message itself.

```
transform could not transform video/x-raw, format=(string)NV12, width=(int)1920, height=(int)1080,
framerate=(fraction)30/1, interlace-mode=(string)interleaved, colorimetry=(string)bt601, pixel-
aspect-ratio=(fraction)1/1 in anything we support
```

(Emphasis added). This long list of semi-random values is core to GStreamer's operation and is called **capabilities** (abbreviated to 'caps'). They describe the type of data that is transmitted between two pads, or that one pad supports. The element here complains that it can't transform data it receives on its sink pad, as described by the caps, to a format that the rest of the pipeline can support.

Log Categories

It's now time to pump up the log level and harvest debugging information. Setting the log level to 5 for all categories would produce an insane amount of messages, but with the knowledge we have gained we can be more clever than that. As the warning is output by the GstBaseTransform class code for v4l2video0convert0, let's target the categories related to all parent classes of the object.

Category names can't be queried dynamically, but they can usually be guessed as they are most often equal to the class name without the Gst prefix and in all lower-case. If that fails, they can be found in the source code, initialized by the GST_DEBUG_CATEGORY_INIT() macro.

```
    GST_DEBUG_CATEGORY_INIT (gst_v4l2_transform_debug, "v4l2transform", 0,
        "V4L2 Converter");
```

sys/v4l2/gstv4l2transform.c

Running the same pipeline with additional debugging

```
$ GST_DEBUG="basetransform:5,v4l2transform:5" gst-launch-1.0 --gst-debug-no-color \
        videotestsrc pattern=ball num-buffers=120 ! \
        video/x-raw,format=NV12,width=1920,height=1080,framerate=60/1,interlace-mode=progressive !
\
        interlace field-pattern=0 ! \
        v4l2video0convert ! \
        queue ! \
        video/x-raw,format=NV12,width=1920,height=1080,interlace-mode=progressive ! \
        x264enc ! \
        h264parse ! \
        mp4mux ! \
        filesink location=fdp1.interlace.smpte.NV12-NV12.1920.1080.mp4
```

produces more than 400 log messages. Reading them all would only be interesting if you're suffering from synesthesia. I will spare you the feeling of despair caused by being lost in the abysmal depths of the endless logs and walk from the previous warning message backwards. The timestamp, process ID, thread ID and log level are removed for clarity.

Capabilities

```
1.  basetransform gstbasetransform.c:704:gst_base_transform_query_caps:<capsfilter1:sink> filter
caps  video/x-raw, framerate=(fraction)30/1, interlace-mode=(string)interleaved, format=(string){
RGB15, RGB16, BGRA, BGRx, ARGB, xRGB, RGB, BGR, NV16, NV61, NV12, NV21, UYVY, YUY2, YVYU, I420 },
width=(int)[ 1, 32768 ], height=(int)[ 1, 32768 ]
2.  basetransform gstbasetransform.c:707:gst_base_transform_query_caps:<capsfilter1:sink> our
template  ANY
3.  basetransform gstbasetransform.c:709:gst_base_transform_query_caps:<capsfilter1:sink>
intersected video/x-raw, framerate=(fraction)30/1, interlace-mode=(string)interleaved, format=
(string){ RGB15, RGB16, BGRA, BGRx, ARGB, xRGB, RGB, BGR, NV16, NV61, NV12, NV21, UYVY, YUY2, YVYU,
I420 }, width=(int)[ 1, 32768 ], height=(int)[ 1, 32768 ]
4.  basetransform gstbasetransform.c:533:gst_base_transform_transform_caps:<capsfilter1> transform
caps (direction = 2)
5.  basetransform gstbasetransform.c:560:gst_base_transform_transform_caps:<capsfilter1> to: EMPTY
6.  basetransform gstbasetransform.c:714:gst_base_transform_query_caps:<capsfilter1:sink>
transformed  EMPTY
7.  basetransform gstbasetransform.c:730:gst_base_transform_query_caps:<capsfilter1:sink> peer
filter caps EMPTY
```

```
8.  basetransform gstbasetransform.c:733:gst_base_transform_query_caps:<capsfilter1:sink> peer
filter caps are empty
9.  basetransform gstbasetransform.c:793:gst_base_transform_query_caps:<capsfilter1> returning
EMPTY
10. basetransform gstbasetransform.c:1174:gst_base_transform_find_transform:<v4l2video0convert0>
Resulted in EMPTY
11. basetransform gstbasetransform.c:1260:gst_base_transform_find_transform:<v4l2video0convert0>
transform could not transform video/x-raw, format=(string)NV12, width=(int)1920, height=(int)1080,
framerate=(fraction)30/1, interlace-mode=(string)interleaved, colorimetry=(string)bt601, pixel-
aspect-ratio=(fraction)1/1 in anything we support
12. basetransform gstbasetransform.c:1414:gst_base_transform_setcaps:<v4l2video0convert0> transform
could not transform video/x-raw, format=(string)NV12, width=(int)1920, height=(int)1080, framerate=
(fraction)30/1, interlace-mode=(string)interleaved, colorimetry=(string)bt601, pixel-aspect-ratio=
(fraction)1/1 in anything we support
```

The capsfilter elements correspond to the capabilities filtering lines in the gst-launch pipeline. capsfilter1 being the second instance, it corresponds to

video/x-raw,format=NV12,width=1920,height=1080,interlace-mode=progressive

The log shows that the filter produced an empty set of caps from the input caps it received

video/x-raw, framerate=(fraction)30/1, interlace-mode=(string)interleaved, format=(string){ RGB15, RGB16, BGRA, BGRx, ARGB, xRGB, RGB, BGR, NV16, NV61, NV12, NV21, UYVY, YUY2, YVYU, I420 }, width=(int)[1, 32768], height=(int)[1, 32768]

This behaviour is normal, as filtering the input 'interlace-mode=interleaved' with a 'interlace-mode=progressive' filter leaves no option. If your mind hadn't burnt up from its hopeless random wandering in the logs' abyss, you would now be wondering why the filter received an interleaved format on its sink pad, given that the previous element in the pipeline, v4l2video0convert0, corresponds to your deinterlacer and is supposed to produce progressive video. To answer that question let's look at the previous lines from the log.

```
1.  basetransform gstbasetransform.c:533:gst_base_transform_transform_caps:<v4l2video0convert0>
transform caps (direction = 2)
2.  v4l2transform gstv4l2transform.c:416:gst_v4l2_transform_transform_caps:<v4l2video0convert0>
transformed video/x-raw, format=(string)NV12, width=(int)1920, height=(int)1080, framerate=
(fraction)30/1, interlace-mode=(string)interleaved, colorimetry=(string)bt601, pixel-aspect-ratio=
(fraction)1/1 into video/x-raw, framerate=(fraction)30/1, interlace-mode=(string)interleaved
3.  basetransform gstbasetransform.c:560:gst_base_transform_transform_caps:<v4l2video0convert0> to:
video/x-raw, framerate=(fraction)30/1, interlace-mode=(string)interleaved
4.  basetransform gstbasetransform.c:1142:gst_base_transform_find_transform:<v4l2video0convert0>
intersecting against padtemplate video/x-raw, format=(string){ RGB15, RGB16, BGRA, BGRx, ARGB,
xRGB, RGB, BGR, NV16, NV61, NV12, NV21, UYVY, YUY2, YVYU, I420 }, width=(int)[ 1, 32768 ], height=
(int)[ 1, 32768 ], framerate=(fraction)[ 0/1, 2147483647/1 ]
5.  basetransform gstbasetransform.c:1164:gst_base_transform_find_transform:<v4l2video0convert0>
transform returned non fixed  video/x-raw, framerate=(fraction)30/1, interlace-mode=
(string)interleaved, format=(string){ RGB15, RGB16, BGRA, BGRx, ARGB, xRGB, RGB, BGR, NV16, NV61,
NV12, NV21, UYVY, YUY2, YVYU, I420 }, width=(int)[ 1, 32768 ], height=(int)[ 1, 32768 ]
6.  basetransform gstbasetransform.c:1171:gst_base_transform_find_transform:<v4l2video0convert0>
Checking peer caps with filter video/x-raw, framerate=(fraction)30/1, interlace-mode=
(string)interleaved, format=(string){ RGB15, RGB16, BGRA, BGRx, ARGB, xRGB, RGB, BGR, NV16, NV61,
NV12, NV21, UYVY, YUY2, YVYU, I420 }, width=(int)[ 1, 32768 ], height=(int)[ 1, 32768 ]
```

Line 2 tells us that the v4l2video0convert0 element transformed the caps it received on its sink pad

video/x-raw, format=(string)NV12, width=(int)1920, height=(int)1080, framerate=(fraction)30/1, interlace-mode=(string)interleaved, colorimetry=(string)bt601, pixel-aspect-ratio=(fraction)1/1

into

video/x-raw, framerate=(fraction)30/1, interlace-mode=(string)interleaved

This transformation is performed by the GstV4l2Transform class in the gst_v4l2_transform_transform_caps() function. The GstBaseTransform class then intersects the caps with the caps supported by the v4l2video0convert0 element's source pad (line 4)

video/x-raw, format=(string){ RGB15, RGB16, BGRA, BGRx, ARGB, xRGB, RGB, BGR, NV16, NV61, NV12, NV21, UYVY, YUY2, YVYU, I420 }, width=(int)[1, 32768], height=(int)[1, 32768], framerate=(fraction)[0/1, 2147483647/1]

to produce the allowed caps (line 5)

video/x-raw, framerate=(fraction)30/1, interlace-mode=(string)interleaved, format=(string){ RGB15, RGB16, BGRA, BGRx, ARGB, xRGB, RGB, BGR, NV16, NV61, NV12, NV21, UYVY, YUY2, YVYU, I420 }, width=(int)[1, 32768], height=(int)[1, 32768]

The caps intersection computation by GstBaseTransform is correct, but is being incorrectly fed caps with 'interlace-mode=interleaved' by GstV4l2Transform. This is all we can find out from the log, you now have permission to dive into the source code.

I'm waiting.

GstV4l2Transform

Still waiting.

Not there yet ?

I will give you a hint, to reward you for reading this far. The code you are looking for is the implementation of gst_v4l2_transform_transform_caps(), in gstv4l2transform.c.

```
/* The caps can be transformed into any other caps with format info removed.
 * However, we should prefer passthrough, so if passthrough is possible,
 * put it first in the list. */
static GstCaps *
gst_v4l2_transform_transform_caps (GstBaseTransform * btrans,
    GstPadDirection direction, GstCaps * caps, GstCaps * filter)
{
  GstCaps *tmp, *tmp2;
  GstCaps *result;

  /* Get all possible caps that we can transform to */
  tmp = gst_v4l2_transform_caps_remove_format_info (caps);

  if (filter) {
    tmp2 = gst_caps_intersect_full (filter, tmp, GST_CAPS_INTERSECT_FIRST);
    gst_caps_unref (tmp);
    tmp = tmp2;
  }

  result = tmp;

  GST_DEBUG_OBJECT (btrans, "transformed %" GST_PTR_FORMAT " into %"
      GST_PTR_FORMAT, caps, result);

  return result;
}
```

sys/v4l2/gstv4l2transform.c

The function removes format information from the passed caps, to account for the ability of the element to convert between different video formats. This is implemented in the gst_v4l2_transform_caps_remove_format_info() function.

```
/* copies the given caps */
static GstCaps *
gst_v4l2_transform_caps_remove_format_info (GstCaps * caps)
{
  GstStructure *st;
  GstCapsFeatures *f;
  gint i, n;
  GstCaps *res;

  res = gst_caps_new_empty ();

  n = gst_caps_get_size (caps);
  for (i = 0; i < n; i++) {
    st = gst_caps_get_structure (caps, i);
    f = gst_caps_get_features (caps, i);

    /* If this is already expressed by the existing caps
     * skip this structure */
    if (i > 0 && gst_caps_is_subset_structure_full (res, st, f))
      continue;

    st = gst_structure_copy (st);
    /* Only remove format info for the cases when we can actually convert */
    if (!gst_caps_features_is_any (f)
        && gst_caps_features_is_equal (f,
            GST_CAPS_FEATURES_MEMORY_SYSTEM_MEMORY))
      gst_structure_remove_fields (st, "format", "colorimetry", "chroma-site",
          "width", "height", "pixel-aspect-ratio", NULL);

    gst_caps_append_structure_full (res, st, gst_caps_features_copy (f));
  }

  return res;
}
```

sys/v4l2/gstv4l2transform.c

Fixing the Problem

You are looking straight at the root cause of the problem. The function removes the format, colorimetry, chroma-site, width, height and pixel-aspect-ratio fields of the caps, but leaves the interlace-mode in place. I leave to you the honour of fixing it.

```
      gst_structure_remove_fields (st, "format", "colorimetry", "chroma-site",
          "width", "height", "pixel-aspect-ratio", "interlace-mode", NULL);
```

sys/v4l2/gstv4l2transform.c

It's now Monday evening, you're exhausted with your eyes all dry from staring at logs the whole day, your fingers barely resisting the envy to reach for your Rambo knife in the drawer, and your brain in a state that would be the subject of dirty

jokes among neuroscience PhD students. Congratulations, you're now a GStreamer developer.