GstStream API

GStreamer Conference 2016, Berlin
October 11th 2015

Edward Hervey a.k.a `bilboed`

edward@centricular.com

gstreamer

Centricular

- Playbin3 and Decodebin3
- New GstStream API
- Purpose and Usage
- Examples

Summary of past year

- Decodebin3 and playbin3 have landed
- CPU and memory improvements …
- … but also more use-cases
- And a new unified and generic API to deal with “Streams”

Streams

- What you would say out loud
  - A video stream, an audio stream, …
- Higher level than elements and pads
- Cross-element concept
  - “This stream is created by a demuxer, parsed by a parser, decoded by a decoder, goes through queues and is displayed in this sink”
- A new Object to describe that:
  - GstStream

GstStream object

- GstObject subclass (refcounted)
  - Id (same as STREAM_START stream-id)
  - GstStreamType and GstStreamFlags
    - GST_STREAM_TYPE_{AUDIO, VIDEO, TEXT, …}
  - GstCaps
  - GstTagList
- Collection of information stored in various places

```
GstStream *gst_stream_new             (const gchar *stream_id,
                       GstCaps *caps,
                       GstStreamType type,
                       GstStreamFlags flags);

const gchar *gst_stream_get_stream_id (GstStream *stream);

void           gst_stream_set_stream_flags (GstStream *stream, GstStreamFlags flags);
GstStreamFlags gst_stream_get_stream_flags (GstStream *stream);

void          gst_stream_set_stream_type (GstStream *stream, GstStreamType
stream_type);
GstStreamType gst_stream_get_stream_type (GstStream *stream);

void        gst_stream_set_tags (GstStream *stream, GstTagList *tags);
GstTagList *gst_stream_get_tags (GstStream *stream);

void     gst_stream_set_caps (GstStream *stream, GstCaps *caps);
GstCaps *gst_stream_get_caps (GstStream *stream);
```

GstStream object

- Created by any element that introduces a new stream (sources, demuxers, …)
  - Parsebin automatically creates it for you if needed
- Conveyed in GST_EVENT_STREAM_START
  - No changes needed for elements
- Query with gst_pad_get_stream(..);
- Elements can refine/extend information
  - Tags,
  - Caps,

Streams

- A stream is rarely alone …
- … and it would be great to know which streams are present in a pipeline in an easy way (bus message ?)

GstStreamCollection

- A immutable collection of GstStream
  - Usually created by demuxers or other elements that can offer a “collection” of streams
  - Doesn’t mean each stream is actually present/exposed
    - Multi-angle DVD, alternate online streams, ….

GstStreamCollection

```
GstStreamCollection *gst_stream_collection_new (const gchar *upstream_id);

const gchar *gst_stream_collection_get_upstream_id (GstStreamCollection
*collection);

guint gst_stream_collection_get_size (GstStreamCollection *collection);
GstStream *gst_stream_collection_get_stream (GstStreamCollection *collection,
guint index);

gboolean gst_stream_collection_add_stream (GstStreamCollection *collection,
                        GstStream *stream);
```

Sending GstStreamCollection

- GST_EVENT_STREAM_COLLECTION
  - Sticky downstream event
  - Elements can know what streams are available upstream
- GST_MESSAGE_STREAM_COLLECTION
  - User, application, bin, can be informed of available streams
  - No longer need to fiddle with element, pads, probes, … to know available streams :)
  - Can react synchronously to select streams ASAP

Selecting Streams

- Aka: Actually doing something with all these objects
- Elements and application can know the various types of streams available
  - How to specify which one should actually be exposed or selected ?

Selecting Streams

- GST_EVENT_SELECT_STREAMS
  - List of stream-id to be selected
    - Which you got from GstStreamCollection and GstStream
- Elements can now reliably know **which** streams will be needed downstream
  - Avoid processing (cpu/memory/io)
  - Avoid downloading
  - Hidden streams to activate (Alternate HLS/DASH, switching DVB channel, ...)

Selecting Streams

- gst_event_new_select_streams(GList*);
- gst_event_parse_select_streams(GstEvent*, GList **);
- To be clarified for 1.12:
  - Specifying that we **might** want other streams to be present for fast switching or not (fast switching vs never need them)
  - Handling multiple collection (ex: collection from dvbsrc and collection from tsdemux)

Selecting Streams

- It might take time for the requested selection to become active
- When this happens, the element in charge of the selection posts a GST_MESSAGE_STREAMS_SELECTED on the bus
  - Applications can update their UI accordingly
- Will also be posted by decodebin3 if no SELECT_STREAMS event was sent
  - You know what the initial selection is

Dynamic streams, what about pads ?

- We can now provide list of available streams, unrelated to presence of pads
- Historical handling of “updating streams”
  - Emit “no-more-pads”
  - Add new pads, including for streams already present
  - Emit “no-more-pads”
  - Push EOS on old pads no longer needed,
  - Remove old pads no longer needed

Downside of historical behaviour

- If you wanted to add/remove a new stream
  - You would have to “break” your other streams (end up being a “new” discontinous stream)
  - Seems stupid tbh …
- But we can indicate which streams will be available now ! We could just add/remove pads when we want !
- Sure … but we mustn’t break existing behaviour

Indicate you can handle new behaviour

- GST_BIN_FLAG_STREAMS_AWARE
- You are telling children elements that they can add/remove pads at any time
  - Provided they post the new collection before hand
- Ensures backwards-compatibility
  - Element just check the parent element flag
- This is totally not something that was implemented yesterday
  - At all
    - Really

Example: tsdemux

- Mpeg-ts is a streaming format
- You can have new streams appearing or going away at any time
  - Ex: switching to a movie with new audio tracks or subtitles
- If parent is “streams-aware”
  - Re-use existing program
  - Create new streams and collection
  - Post collection
  - Add/Remove pads

Questions ?