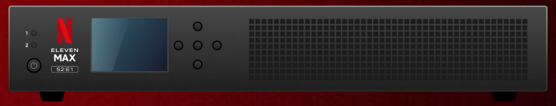
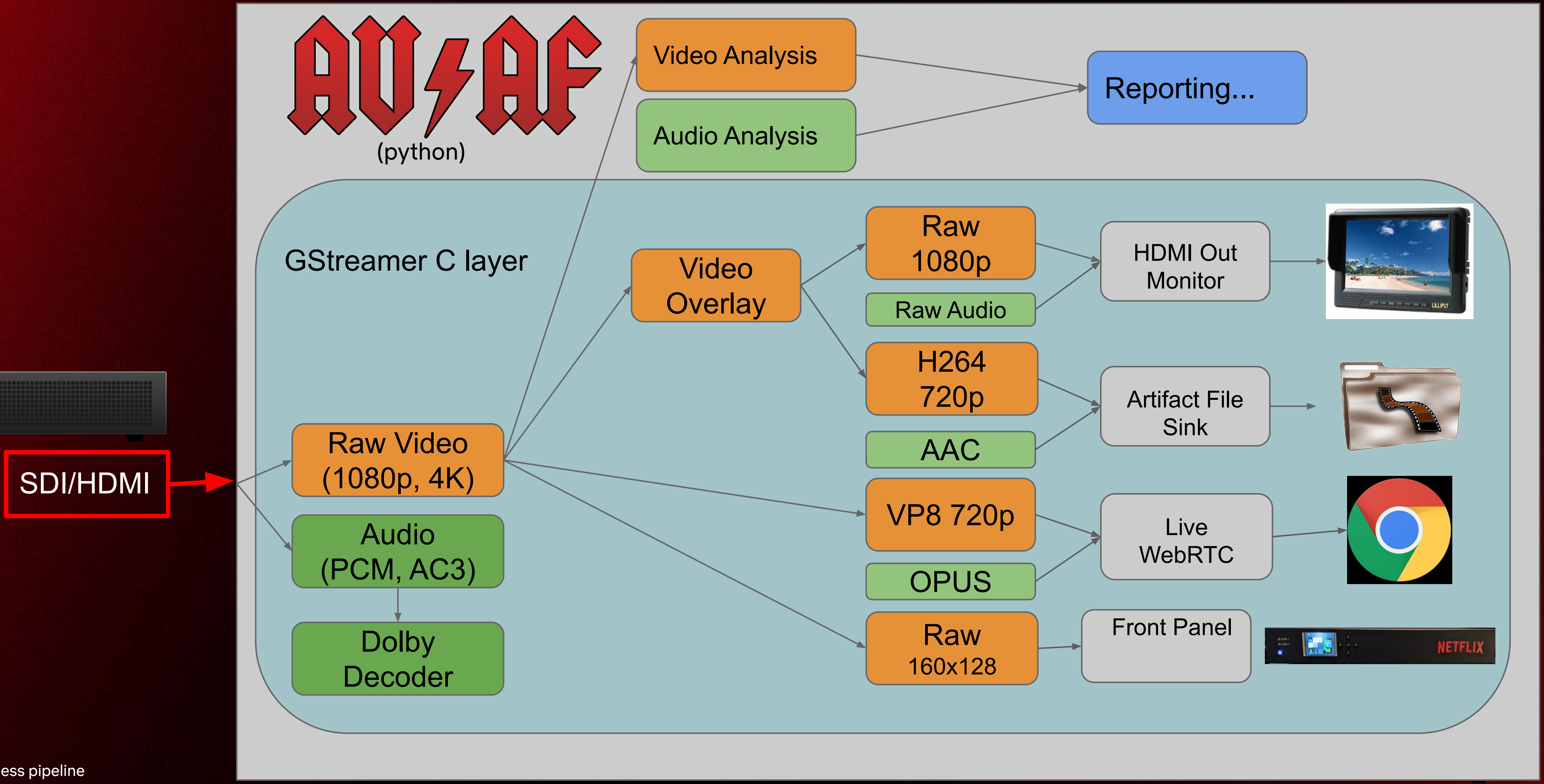
unixfd: Zero-copy multi-process pipeline

Xavier Claessens
October 7, 2024

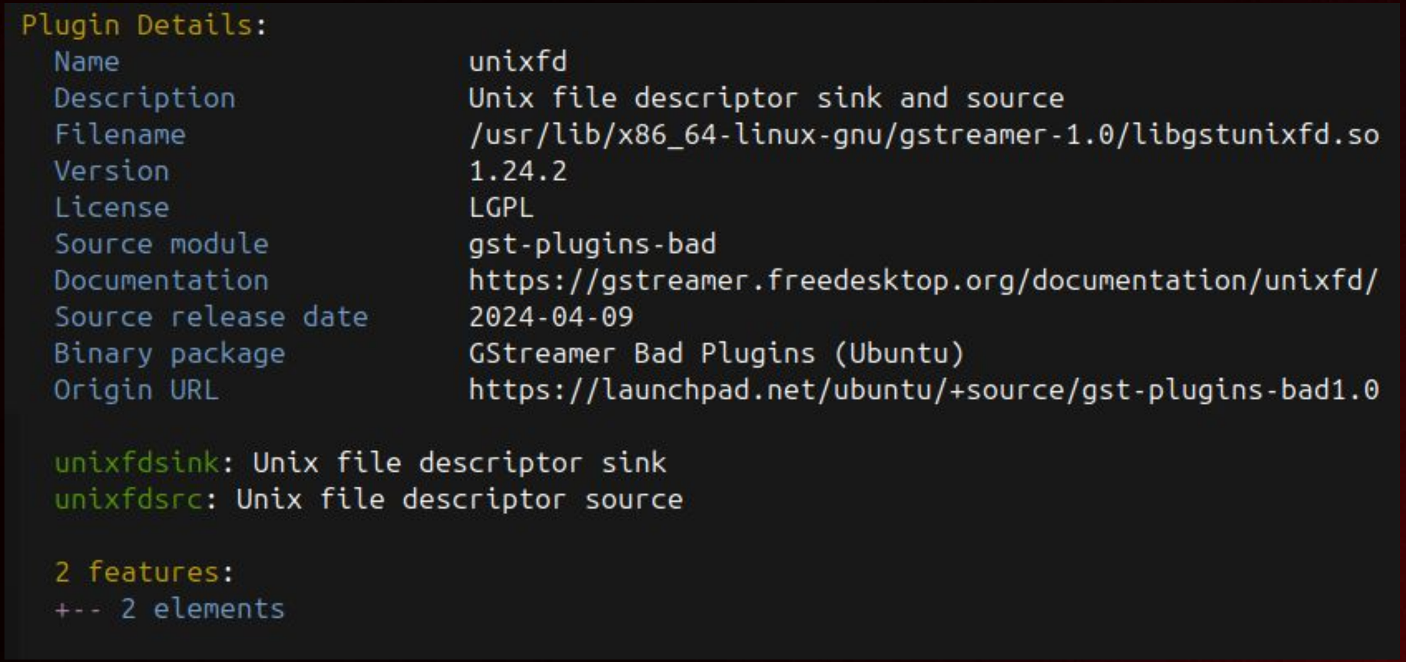
Unixfd - Context





Unixfd - What?

- Sink and source elements
- Communicate over a unix socket
- Share file descriptors from source to sink element
- 1:n relation like tee
- Since GStreamer 1.24



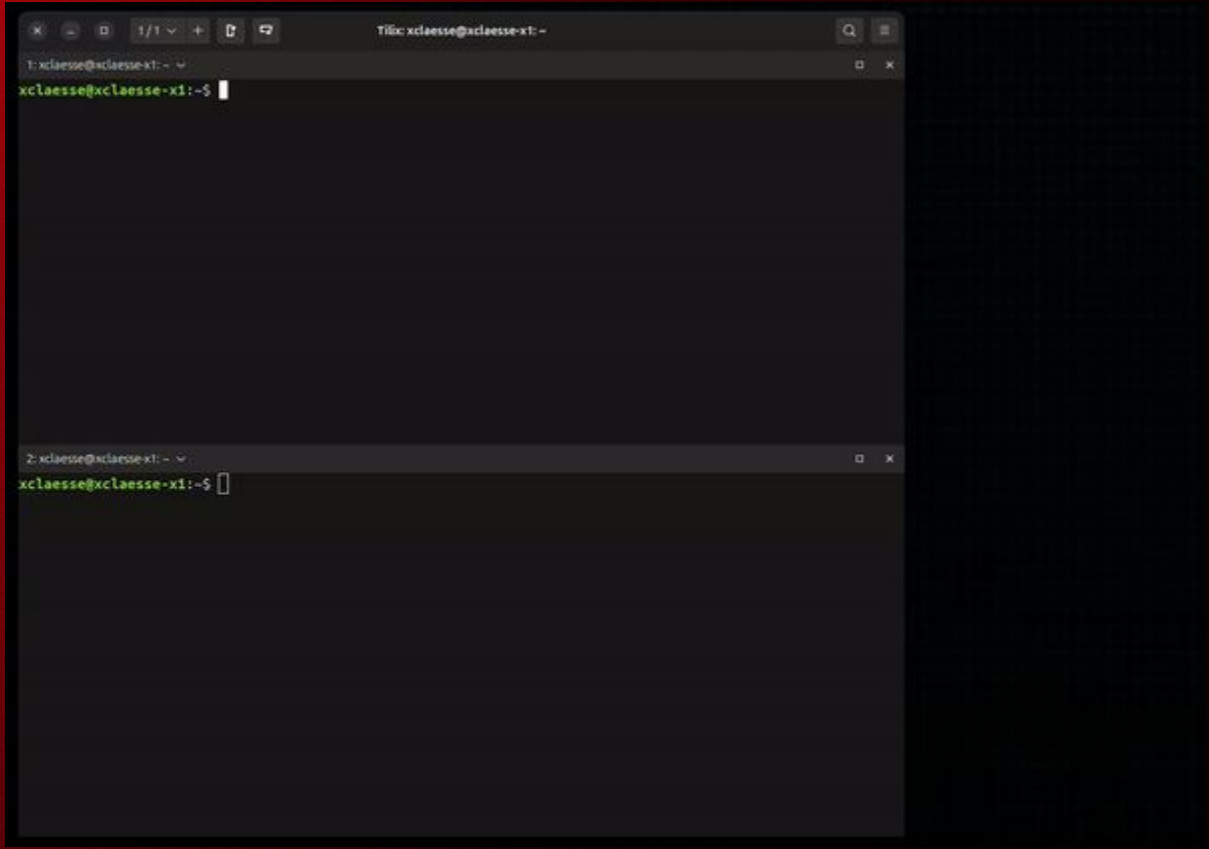
```
Plugin Details:
  Name                     unixfd
  Description              Unix file descriptor sink and source
  Filename                 /usr/lib/x86_64-linux-gnu/gstreamer-1.0/libgstunixfd.so
  Version                  1.24.2
  License                  LGPL
  Source module            gst-plugins-bad
  Documentation            https://gstreamer.freedesktop.org/documentation/unixfd/
  Source release date      2024-04-09
  Binary package           GStreamer Bad Plugins (Ubuntu)
  Origin URL               https://launchpad.net/ubuntu/+source/gst-plugins-bad1.0

  unixfdsink: Unix file descriptor sink
  unixfdsrc: Unix file descriptor source

  2 features:
  +-- 2 elements
```

Unixfd - Why?

- Isolate parts of your pipeline in different processes for security, stability, …
- Zero-copy, sharing memfd, dmabuf, …

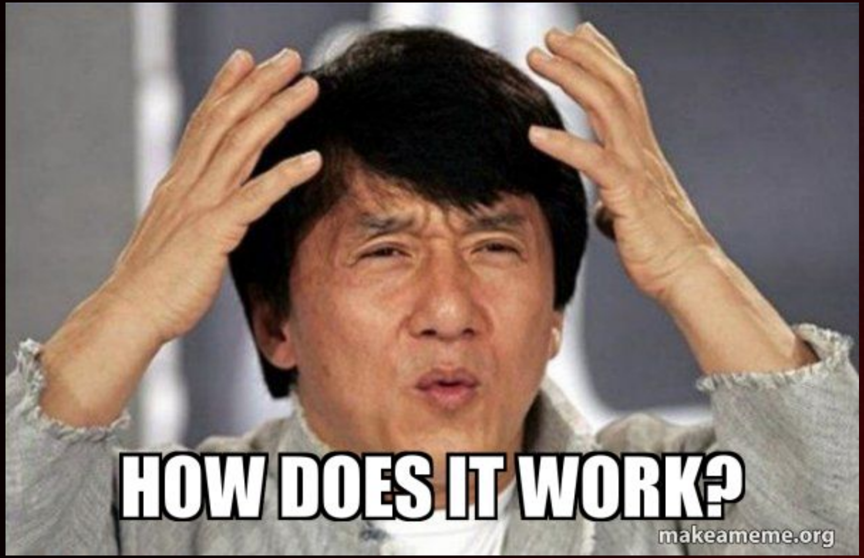
Prior art in GStreamer - shmsink, shmsrc
- Copies buffers into a shared memory.
- Does not send caps, metas, etc.
- Single shared memory area of fixed size, not suitable for large raw video buffers.
- SAT has similar shmdata.

Unixfd - How?

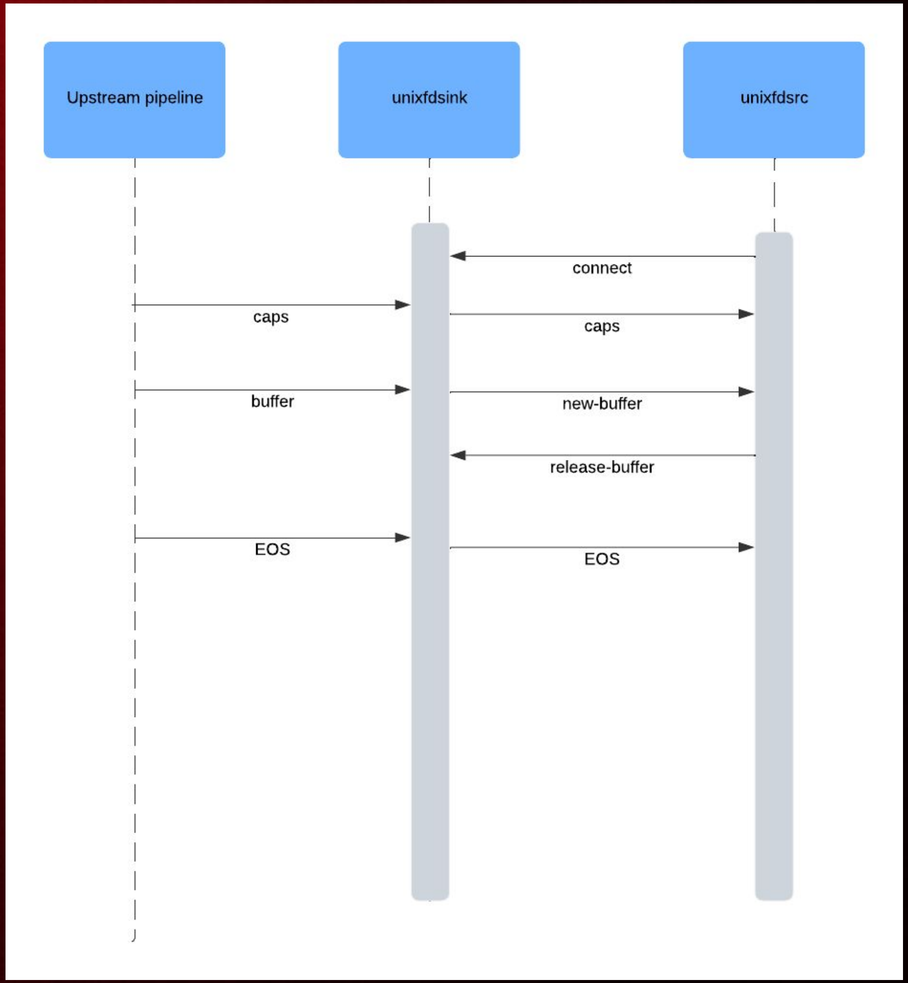

Tilix: xclaesse@xclaesse-x1: ~
1: xclaesse@xclaesse-x1: ~
2: xclaesse@xclaesse-x1: ~

```
xclaesse@xclaesse-x1:~$
```

```
xclaesse@xclaesse-x1:~$
```



Unixfd - How?


Upstream pipeline
unixfdsink
unixfdsrc
connect
caps
caps
buffer
new-buffer
release-buffer
EOS
EOS

Unixfd - How?

Timestamps and clocks:

- Upstream and downstreams pipelines on different clocks.
- Convert to/from system monotonic clock.
- Takes current segment into account: gst_segment_to_running_time().

Unixfd - How?

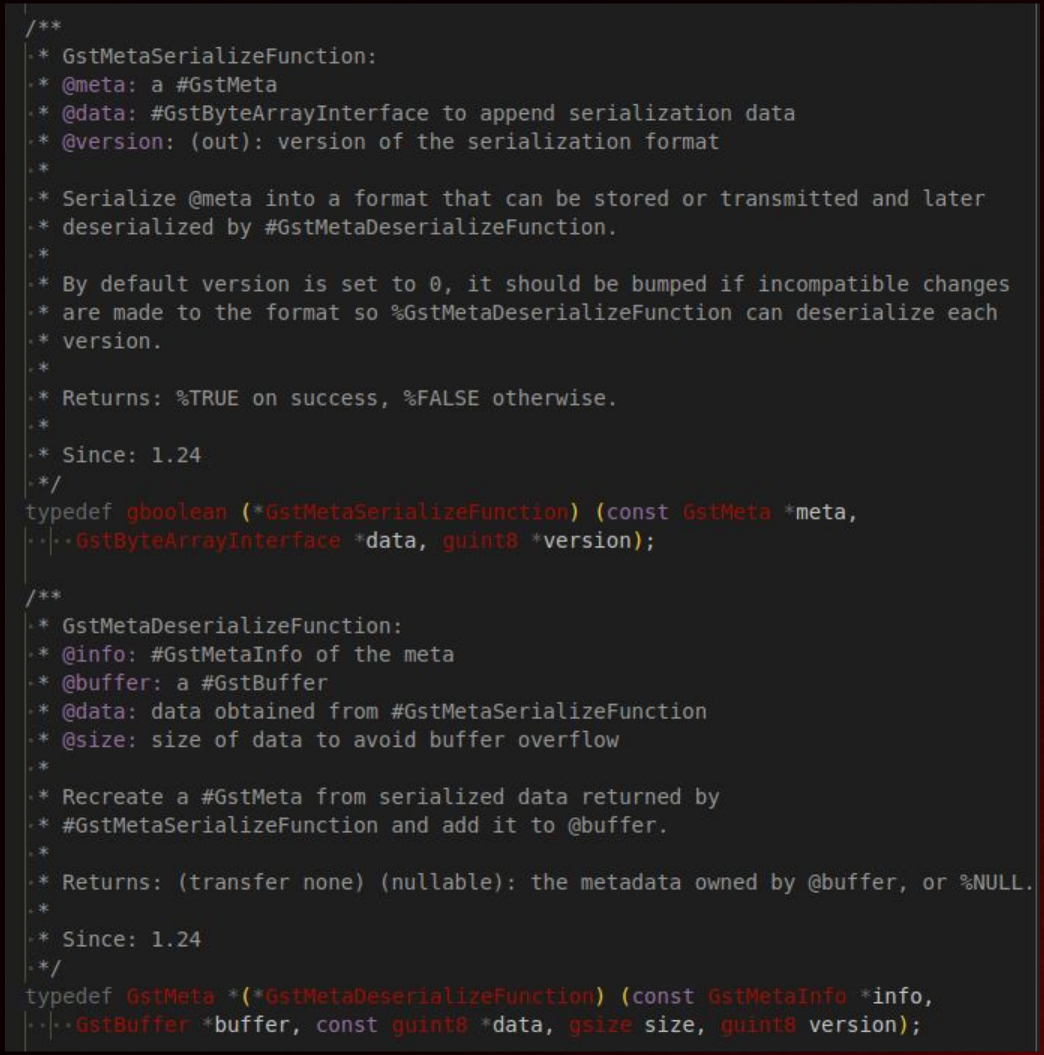
Wait, you said zero-copy!

- GstShmAllocator: a memfd allocator.
- Used by waylandsink too.
- F_SEAL_SHRINK but not (yet) F_SEAL_FUTURE_WRITE !5684.
- Fallbacks to shm.
- unixfdsink: propose allocation.
- GstBaseSrc: decide allocation.

Unixfd - How?

But, but, but… you need metas!

- gst_meta_(de)serialize()
- GstAudioMeta
- GstVideoMeta
- GstReferenceTimestampMeta
- GstCustomMeta
- More to come…



```
/**
 * GstMetaSerializeFunction:
 * @meta: a #GstMeta
 * @data: #GstByteArrayInterface to append serialization data
 * @version: (out): version of the serialization format
 *
 * Serialize @meta into a format that can be stored or transmitted and later
 * deserialized by #GstMetaDeserializeFunction.
 *
 * By default version is set to 0, it should be bumped if incompatible changes
 * are made to the format so %GstMetaDeserializeFunction can deserialize each
 * version.
 *
 * Returns: %TRUE on success, %FALSE otherwise.
 *
 * Since: 1.24
 */
typedef gboolean (*GstMetaSerializeFunction) (const GstMeta *meta,
    GstByteArrayInterface *data, guint8 *version);

/**
 * GstMetaDeserializeFunction:
 * @info: #GstMetaInfo of the meta
 * @buffer: a #GstBuffer
 * @data: data obtained from #GstMetaSerializeFunction
 * @size: size of data to avoid buffer overflow
 *
 * Recreate a #GstMeta from serialized data returned by
 * #GstMetaSerializeFunction and add it to @buffer.
 *
 * Returns: (transfer none) (nullable): the metadata owned by @buffer, or %NULL.
 *
 * Since: 1.24
 */
typedef GstMeta *(*GstMetaDeserializeFunction) (const GstMetaInfo *info,
    GstBuffer *buffer, const guint8 *data, gsize size, guint8 version);
```

Unixfd - Next?

- Implement serialization for remaining metas
- Serialize metas with gdppay?
- Bidirectional events/queries?
- Make a copy if source does not use shm allocator - patch from Nicolas.
- Share your ideas!

Unixfd - Demo

- With stock GStreamer from Ubuntu 24.04 LTS.

Thank You

Xavier Claessens